

POWER BI MVP BOOK

A BOOK OF TRICKS AND TECHNIQUES FOR
WORKING WITH POWER BI
AUTHORED BY 21 POWER BI MVPS



First Edition
August 2019

FOREWORD BY ARUN ULAG
GENERAL MANAGER, POWER BI, MICROSOFT

ORGANIZED BY REZA RAD

Authors:

Anil Maharjan, Nepal
Indira Bandari, New Zealand
Liam Bastick, Australia
Ken Puls, Canada
Jesus Gil, Mexico
Reza Rad, New Zealand
Thomas LeBlanc, USA
Ike Ellis, USA
Matt Allington, Australia
Leila Etaati, New Zealand
Markus Ehrenmüller, Austria
Ashraf Ghonaim, Canada
Eduardo Castro, Costa Rica
Manohar Punna, Australia
Treb Gatte, USA
Gilbert Quevauvilliers, Australia
Michael Johnson, South Africa
Shree Khanal, Nepal
Asgeir Gunnarsson, Iceland
Greg Low, Australia
Gogula Aryalingam, Sri Lanka

Power BI MVP Book

First Edition

August 2019

High-Level Table of Contents

Part I: Getting Data

[Chapter 1: Using Power Query to tell your story from your Facebook Data](#)

[Chapter 2: Get Data from Multiple URLs Using Web By Example](#)

[Chapter 3: One URL, Many Tables](#)

Part II: Data Preparation

[Chapter 4: Creating Calendar Dimensions with Power Query](#)

[Chapter 5: Transform and combine your data with ETL tool named Power Query](#)

[Chapter 6: Creating a Shared Dimension in Power BI Using Power Query: Basics and Foundations of Modeling](#)

[Chapter 7: Data Modeling with Relational Databases](#)

Part III: DAX and Calculations

[Chapter 8: Up and Running with DAX as Quick as Possible](#)

[Chapter 9: Power BI is Not the Same as Excel](#)

Part IV: AI and Power BI

[Chapter 10: AI for Business Users in Dataflow and Power BI Desktop](#)

[Chapter 11: AI in Power BI Desktop](#)

[Chapter 12: Automated Machine Learning in Power BI](#)

Part V: Integration of Power BI with other services and tools

[Chapter 13: Power BI REST API](#)

[Chapter 14: Real-Time Streaming Datasets](#)

Part VI: Power BI for Enterprise

[Chapter 15: Introduction to Conversation-Centric Design TM](#)

[Chapter 16: Understanding when to move to Power BI Premium](#)

[Chapter 17: Incremental refresh](#)

[Chapter 18: Report Server Administration](#)

Part VII: Architecture

[Chapter 19: Governance](#)

[Chapter 20: Architecture of a Power BI Solution in an Enterprise Environment](#)

[Chapter 21: A Power BI-only Solution for Small Organizations](#)

[Table of Contents](#)

[High-Level Table of Contents](#)

[Foreword](#)

[Introduction](#)

[Who is this book for](#)

[Who this book is not for](#)

[How the book is organized](#)

[Chapter 1: Using Power Query to tell your story from your Facebook Data](#)

[Introduction](#)

[Power Query](#)

[Let's drill into your Facebook data to extract your story](#)

[Facebook Graph API](#)

[Power Query Analysis 1](#)

[Facebook Feed Trend Analysis](#)

[Power Query Analysis 2](#)

[Facebook Photos by Location Tracking](#)

[Summary](#)

[About the Author](#)

[Chapter 2: Get Data from Multiple URLs Using Web By Example](#)

[Get Data from Web By Example from a Single Web Page](#)

[Create a Table](#)

[Create a Parameter and a Function to get Data from Multiple Web Pages](#)

[1. Create a Parameter](#)

[2. Create a Function](#)

[3. Get Data from Multiple Web Pages](#)

[Summary](#)

[About the Author](#)

[Chapter 3: One URL, Many Tables](#)

[Part 1: Manual Retrieval of Data](#)

[Part 2: Custom Functions](#)

[Part 3: Unknown Number of Pages](#)

[Part 4: Fiddling with the URL](#)

[Part 5: Putting it All Together](#)

[About the Author](#)

[Chapter 4: Creating Calendar Dimensions with Power Query](#)

[To Create or not to Create?](#)

[Dynamic Calendars vs the Corporate Database](#)

[Doesn't Power BI Use Default Date Tables?](#)

[Sample Data](#)

[Creating a Dynamic Calendar Table](#)

[Recipe for StartDate and EndDate Queries](#)

[Building the Base Calendar table](#)

[Add any Additional Columns Needed](#)

[Add Fiscal Year Ends to Your Calendar](#)

[Fiscal Periods](#)

[Fiscal Year End](#)

[Building a 4-4-5, 4-5-4 or 5-4-4 \(ISO\) Calendar](#)

[Creating StartDate and EndDate queries for 4-4-5 calendars](#)

[Creating the "DayID" column](#)

[Creating the remaining PeriodID columns](#)

[Adding Other Fiscal Periods](#)

[Fiscal Year Columns](#)

[X of Year Columns](#)

[X of Quarter Columns](#)

[X of Month Columns](#)

[X of Week Columns](#)

[Start of X Columns](#)

[End of X Columns](#)

[Summary](#)

[About the Author](#)

[Chapter 5: Transform and combine your data with ETL tool named Power Query](#)

[What is Power Query?](#)

[Where is used Power Query?](#)

[Why do people love Power Query?](#)

[ETL: The concept](#)

[Building the ETL with Power Query](#)

[Why do we get taken to a blank pane?](#)

[Transform ribbon tab](#)

[Use first Row as Headers \(Promoted Headers task\)](#)

[Changing Data Type](#)

[Add Column tab](#)

[Adding custom column with M Formula](#)

[View tab](#)

[Advanced Editor](#)

[Summary](#)

[About the Author](#)

[Chapter 6: Creating a Shared Dimension in Power BI Using Power Query:
Basics and Foundations of Modeling](#)

[Sample Dataset](#)

[Design Challenge](#)

[Many-to-many Relationship Issue](#)

[Both-directional Relationship Issue](#)

[Master List Does Not Exist!](#)

[Shared Dimension: Solution](#)

[Creating Shared Dimension](#)

[Prepare sub-tables](#)

[Set all column names to be the same](#)

[Append all three tables](#)

[Remove Duplicates](#)

[Date Dimension](#)

[Best Practice Design: Star Schema and Shared Dimensions](#)

[Summary](#)

[About the Author](#)

[Chapter 7: Data Modeling with Relational Databases](#)

[Data Modeling](#)

[Relational Database](#)

[Data Type](#)

[Additional Fact Table](#)

[Many-to-Many or Bi-Directional Filter](#)

[Hierarchies](#)

[Additional Comments](#)

[Summary](#)

[About the Author](#)

[Chapter 8: Up and Running with DAX as Quick as Possible](#)

[Introduction](#)

[Your First DAX Expression](#)

[Your Second DAX Expression](#)

[Another Example](#)

[Calculated Tables](#)

[The CALCULATE Function](#)

[Variables and Return](#)

[Time Intelligence – YTD](#)

[Time Intelligence – PREVIOUSMONTH](#)

[X vs Non-X Functions](#)

[Best Practice: Organize your code](#)

[Best Practice: Naming Columns & Measures](#)

[Best Practice: Formatting](#)

[Other Resources](#)

[Summary](#)

[About the Author](#)

[Chapter 9: Power BI is Not the Same as Excel](#)

[Introduction](#)

[Some Things Are the Same in Power BI and Excel](#)

[Built with You in Mind](#)

[DAX is a Functional Language](#)

[DAX has Many Common Functions with Excel](#)

[Sometimes Functions Are Similar, But Have Small Differences](#)

[Many Things Are Very Different Between Power BI and Excel](#)

[Power BI is a Database, Excel is a Spreadsheet](#)

[Database](#)

[Tips to Get You Started as You Move to Power BI](#)

[DAX Calculated Columns vs Measures vs Tables](#)

[Using Visuals to Structure your Output](#)

[Filter First, Calculate Second](#)

[About the Author](#)

[Chapter 10: AI for Business Users in Dataflow and Power BI Desktop](#)

[Cognitive Service in Power BI](#)

[AI in Dataflow](#)

[Image Tag in Power BI](#)

[Key Influencer](#)

[List of Questions](#)

[Get it!](#)

[Use It!](#)

[Summary](#)

[About the Author](#)

[Chapter 11: AI in Power BI Desktop](#)

[Introduction](#)

[Linear Regression](#)

[Analytic Line](#)

[DAX](#)

[R Visual](#)

[Summary](#)

[Text Mining](#)

[Word Cloud](#)

[Azure Cognitive Services](#)

[Azure Machine Learning](#)

[R in Azure Machine Learning](#)

[R as a Power Query Transformation](#)

[R Visual](#)

[Summary](#)

[About the Author](#)

[Chapter 12: Automated Machine Learning in Power BI](#)

[What is Machine Learning \(ML\)?](#)

[What are the challenges of Traditional ML?](#)

[What is Automated Machine Learning \(AutoML\)?](#)

[Automated Machine Learning \(AutoML\) in Power BI](#)

[Enabling AutoML in your Power BI Premium Subscription](#)

[Creating an AutoML Model in Power BI](#)

[Creating an AutoML Model Step by Step](#)

[1- Data prep for creating ML Model:](#)

[2- Configuring the ML Model Inputs](#)

[3- ML Model Training](#)

[4- AutoML Model Explainability](#)

[5- AutoML Model Report](#)

[6- Applying the AutoML Model](#)

[Deep dive into the 3 types of ML Models](#)

[1- Binary Prediction Models](#)

[2- Classification Models](#)

[3- Regression Models](#)

[Summary](#)

[About the Author](#)

[Chapter 13: Power BI REST API](#)

[Getting ready to use Power BI REST API](#)

[Register your developer application](#)

[Register your application in Azure Portal](#)

[Preparing Visual Studio to use the Power BI REST API](#)

[Summary](#)

[About the Author](#)

[Chapter 14: Real-Time Streaming Datasets](#)

[Introduction](#)

[Real-Time Datasets](#)

[Push Datasets](#)

[Streaming Datasets](#)

[PubNub Datasets](#)

[Creating Real-Time Datasets](#)

[Power BI Service UI](#)

[Power BI REST API](#)

[Azure Stream Analytics](#)

[PubNub](#)

[Push Data to Streaming Datasets](#)

[Visualizing Real-Time Datasets](#)

[Streaming datasets](#)

[Push datasets](#)

[Summary](#)

[About the Author](#)

[Chapter 15: Introduction to Conversation-Centric Design™](#)

[“Spreadsheet on a Web Page”](#)

[What’s really going on here?](#)

[Conversation-Centric Design™ Overview](#)

[Why formal interactions?](#)

[Process Overview](#)

[A Real World Example](#)

[Summary](#)

[About the Author](#)

[Trebuel Gatte, CEO, MarqueeInsights.com](#)

[Chapter 16: Understanding when to move to Power BI Premium](#)

[Dedicated Performance](#)

[Free Users](#)

[XMLA End Points](#)

[Higher Refreshes](#)

[Incremental Refresh](#)

[Refreshes are quicker](#)

[Fewer resources are required](#)

[Dataset size larger than 1GB](#)

[Actual memory consumption](#)

[Paginated Reports](#)

[Geographic Distribution](#)

[Data Sovereignty](#)

[Performance](#)

[Dataflows](#)

[Linked Entities](#)

[Computed Entities](#)

[Incremental refresh](#)

[Parallel Execution of transformations](#)

[Enhanced Compute Engine](#)

[Monitoring for Power BI Premium](#)

[Summary](#)

[About the Author](#)

[Chapter 17: Incremental refresh](#)

[Introduction](#)

[What is incremental refresh and how does it work](#)

[Requirements](#)

[Premium Workspace](#)

[Query Folding data source](#)

[Transaction dates](#)

[Enabled Incremental Refresh feature](#)

[Setting up Incremental refresh](#)

[Step 1 – Create range parameters](#)

[Step 2: Filter dataset using parameters](#)

[Step 3: Configure incremental refresh in Power BI desktop](#)

[Step 4: Deploy and publish report](#)

[Looking under the covers](#)

[Limitations and things to look out for](#)

[Summary](#)

[About the Author](#)

[Chapter 18: Report Server Administration](#)

[Power BI Report Server](#)

[Power BI Report Server Can be installed on the following Microsoft operating systems](#)

[Web browser:](#)

[Download Power BI Report Server](#)

[Installing Power BI Report Server](#)

[Configuring the Power BI Report Server](#)

[Web Setup](#)

[Installing Power BI Desktop Report Server](#)

[Developing Reports with Power BI Report Server](#)

[Managing Datasets on the Report Server](#)

[Schedule Refresh Requirement](#)

[Resources/ References](#)

[Summary](#)

[About the Author](#)

[Chapter 19: Governance](#)

[Introduction](#)

[Process](#)

[1-Development process](#)

[2-Publishing Process](#)

[3-Sharing Process](#)

[4-Security Process](#)

[5-Naming standard process](#)

[6-Support process](#)

[7-Tenant Settings process](#)

[Training](#)

[Training categories](#)

[Monitoring](#)

[Artefact inventory](#)

[Monitoring usage](#)

[Monitoring the Power BI On-Premise Gateway](#)

[Roles](#)

[Power BI Administrator](#)

[Power BI Gateway Administrator](#)

[Data steward](#)

[Power BI Auditor](#)

[Power BI Supporter](#)

[Summary](#)

[About the Author](#)

[Chapter 20: Architecture of a Power BI Solution in an Enterprise Environment](#)

[The Big Picture](#)

[Identity Management](#)

[On-Premises Active Directory](#)

[Credential Spread Issues](#)

[Hybrid Identity](#)

[Integration Applications](#)

[Azure Active Directory Offerings](#)

[Enhanced Security on Terminations](#)

[Implementing Hybrid Identity](#)

[Application Integration](#)

[Authentication for External Users](#)

[Password-less Implementation](#)

[On-Premises Source Data](#)

[Dimensional Data Models](#)

[Staging and Cleansing Data](#)

[Analytic Data Models](#)

[Moving the data between databases](#)

[Paginated Reporting](#)

[Providing Power BI Access to the On-Premises Data](#)

[Enterprise Gateway](#)

[Power BI Service for Dashboards](#)

[Power BI Service](#)

[Power BI Workspaces](#)

[Connecting Other Applications](#)

[Summary](#)

[About the Author](#)

[Chapter 21: A Power BI-only Solution for Small Organizations](#)

[Background](#)

[Putting a Solution Together with Power BI](#)

[The Actors](#)

[The Solution](#)

[Data Movement and Processing](#)

[Security](#)

[Development life-cycle](#)

[Source control](#)

[Deployment](#)

[Summing it up](#)

[About the Author](#)

Foreword

Since Power BI's launch in July 2015, the product has literally taken the world by storm. In a world where business intelligence solutions were mostly on-premise, Power BI came in with a cloud first, software-as-a-service solution. BI solutions required time and patience to get up and running – Power BI launched with the goal of giving customers “5 seconds to sign up, 5 minutes to wow”. BI solutions were very expensive – with authoring tools upwards of \$1,500 per user and end-user licenses at \$500 or more – Power BI made BI truly accessible, Power BI Desktop was and is completely free, and end-user pricing is simply \$10 per user, per month. And finally, in a world where major BI vendors updated their products about once a year – Power BI was intensely focused on what our customers and community wanted us to build – ideas were submitted and voted on at ideas.powerbi.com, and we released new features weekly in the Power BI service, and a new release of Power BI Desktop every month. This resulted in customers and the community falling in love with the product, and in massive growth of Power BI. In a world that's awash with data, Power BI helps customers make sense of it all and truly drive a data culture – where every employee can make better decision based on data. In just 4 short years since launch, Power BI is now used in over 95% of the Fortune 500 and is recognized as the clear industry leader.

One of the most important factors in Power BI's success is the role of the community. I've known Reza and many of the authors of this book for pretty much most of Power BI's life. Reza and others deserve a lot of the credit for getting the Power BI word out, for helping our customers learn and understand the product, and they have been great partners for the Power BI engineering team in shaping the future of Power BI. They've applauded when we did things well, and called us out when we didn't quite measure up. We're very grateful for their contributions.

In this book, you get to learn directly from the folks who know Power BI best.

[Arun Ulag](#), General Manager, Power BI, Microsoft



Introduction

Power BI is the Microsoft cloud-based reporting tool, great for self-service analysis, as well as enterprise reporting. Power BI with this name first introduced in 2015, but the product has a long fundamental history using SQL Server Analysis Services engine and Excel Power Pivot. Learning Power BI is not just useful for BI developers, but also for data analysts to understand how to leverage the tool to analyze their data.

Power BI MVP book is not like a normal book that you use to learn Power BI. This book is a collective of articles from twenty one Power BI MVPs. Each chapter focused on a very particular subject. The author of that chapter as the subject matter expert shared their opinion and experience and knowledge with the reader specifically. This book is not covering all aspects of Power BI product and toolset, and it never intended to do so. For learning Power BI from beginner level to advanced, you have to read multiple books, and there are many books in the market for it. Some of those books are already used as a reference in chapters of this book.

Who is this book for

- Power BI Report Developers: If you are building reports with Power BI, this book has great tips that can help you in building better reports.
- Architects: If you are an architect and want to implement a strategy and governance for Power BI usage in your organization, this book has tips and chapters for you.
- Citizen Data Scientists: If you are not a data scientist, but want to learn easy ways of using AI functions with Power BI, this book has explained some tricks for you.
- .NET Developers: If you are a programmer or developer, who wants to extend possibilities of using Power BI, and embed that in your application, this book will show you methods for it.
- Consultant and Expert: If you are already using Power BI and consider yourself as an expert in the field, this book will give you a great experience that other experts are sharing with you in their real-world scenarios. This can be a good reference book in your bookshelves to go back and read some of the best practices time by time.

Who this book is not for

- This is not a book to learn Power BI from zero. There are beginner chapters, but there is no single story in the entire book to follow.
- This is not a book that teaches you all about Power BI. There are many subjects that are not covered in this book, because of time constraints, and also the scope of things to learn in the world of Power BI. This is not one book; learn it all.

How the book is organized

Each chapter is written on a specific subject by a different author. The book is organized in this way, that as a reader, you can choose any chapter without needing to read other chapters in advance, start from any chapter, and finish at any chapter you want. Here is what you will learn through this book:

In chapter 1, Anil Maharjan explains on how you can extract the story behind your Facebook data. He shows by using Power Query along with Power BI you can extract your Facebook data easily and analyze your own story by using your Facebook data. This chapter helps you to learn about Power Query and Power BI and shows you how to use self-service BI based on your Facebook data.

In chapter 2, Indira Bandari explains how to scrape data off a blog site (<http://radacad.com/blog>) using the “Add Data By Example” button in Power BI. She then uses Power Query to create a function that gets the blog URLs from a table and extract the data from these URLs. This process can be extended to extract data from any website that has a pattern.

In chapter 3, Liam Bastick considers how to import a multiple page table into Power Query when the URL does not appear to change. It’s harder than you might think and took the best ideas of several contributors to construct a practical solution. Nonetheless, it’s a very useful and important technique to learn and highlights common problems faced when extracting data from the internet.

In chapter 4, Ken Puls shows you how easy it is to create dynamic calendar tables on the fly with Power Query inside Power BI. Whether you need a standard 12-month calendar with a December 31 year end, a 12-month calendar with a March 31 year end, or even a 4-4-5 calendar, Power Query can create it for you if your IT department doesn’t have one you can use. And the best part? They’ll automatically update to cover the entire data range of your model!

In chapter 5, Jesus Gil explains how easy it is to use Power Query, the ease of use, application and implementation with the tool. It quickly explains the concept of ETL and how we can build it with Power Query, either by clicking or through the M language.

In chapter 6, Reza Rad explained some basics of Power BI modelling. He explained why it is important to have separate tables, and what are the advantages of having separate tables. He then explains how you can use Power Query to create fact tables and dimension tables, and build a star schema, and as

a result, build a better data model that answers the reporting requirements.

In Chapter 7, Thomas LeBlanc looks at the benefits of a good relational data model in Power BI. If someone is using a flat file and flattened table for Power BI, data modelling improves the re-usability of a single source of truth. Concepts covered include relationships between tables, using the correct data types for columns as well as measures for repeatable calculations. The relational database example is a dimensional model and the chapter concludes with a look at a many-to-many relationship with bi-directional filtering.

In Chapter 8, Ike Ellis will bring your Power BI skills to the next level by introducing you to the fundamental concepts of DAX. If you've avoided DAX because it seems like a complicated programmer feature, this chapter will show you that DAX is not that difficult. This chapter will show you the path to DAX mastery and will make DAX the first place you'll go when faced with Power BI challenges.

In chapter 9, Matt Allington explains what the differences are between Microsoft Excel and Power BI. Understanding what is the same and what is different is important for people that are trying to move from a traditional Excel world to a structured self-service BI world.

In chapter 10, Leila Etaati provides an overview of new AI capabilities and features in Power BI service and Power BI desktop. First, she explained how business users, using AI very easy, without writing any codes only with a couple of clicks. In this chapter, she shows two different possibilities of consuming AI. First how as a business user can analyse the text in Power BI service, next part, she explains how to use some AI-powered visuals such as Key Influencer in Power BI desktop to analyse the data without knowing the machine learning concepts.

In chapter 11, Markus Ehrenmüller-Jensen describes some of the many possibilities to leverage the use of Artificial Intelligence (AI) in Power BI Desktop. He took Linear Regression and Text Mining as an example to show you, how to make use of DAX, R, Power Query (M), Cognitive Services and Azure Machine Learning.

In Chapter 12, Ashraf Ghonaim explains the definition of Automated Machine Learning (AutoML) and how this breakthrough self-service feature empowers Power BI users to leverage machine learning capabilities and become true Citizen Data Scientists.

In chapter 13, Eduardo Castro explained how to use the Power BI REST API to administer and integrate Power BI with other applications. He shows how to use C# to manage workspaces, permissions and other administration related task using Power BI REST API.

In chapter 14, Manohar Punna introduces various streaming solutions available with Power BI Service. He takes you on a step-by-step implementation of these solutions using different scenarios. The learnings in this chapter give you hands-on experience in building real-time streaming solutions in Power BI.

In Chapter 15, Treb Gatte will guide you through designing your BI content so that it is aligned to the business need. You'll get an introduction to the Conversation-Centric Design Design™ approach that will help you with this process. It'll enable you to address two common problems in BI content development; ensuring you can manage scope easily and ensuring that the outcomes are aligned with where the end user should use the content.

In Chapter 16, Gilbert Quevauvilliers will look at Power BI Premium where there are a lot of options available to you with Power BI Premium. At times when there are too many options and it can potentially be a challenge to understand which features are applicable for your situation. In this chapter Gilbert will provide a better understanding of what these options are. By having a deeper understanding of the Power BI Premium features, it will allow you to make a more informed decision on looking to move to Power BI Premium.

In Chapter 17, Michael Johnson talks about how Incremental Refresh in Power BI is used to reduce the amount of data required to refresh reports improving both refresh time and reliability of these refreshes.

In chapter 18, Shree Khanal explains about the Power BI Report Server's report development, deployment and steps to host it. He highlights how interactive reports are now available on-premises servers and not just on the Power BI service. On top of that he explains the step-by-step process of installing, configuring and setting up the Power BI Report Server.

In chapter 19, Ásgeir Gunnarsson explains how you can tackle Power BI governance. You will learn about the four pillars of Power BI Governance strategy, processes, training, monitoring and roles. Ásgeir then goes into each pillar and explains what you need to think about when it comes to governance and what relevant documents can contain.

In chapter 20, Greg Low, shows all the core components and architecture that

make up a typical enterprise deployment of Power BI. Greg spends much of his time working in large financial enterprises. All of them want to implement Power BI but all of them are confused about how it would best fit into an enterprise environment.

In chapter 21, Gogula Aryalingam explains about how using only Power BI you can create a complete business intelligence solution for a small organization. You will be introduced to the structure upon which typical business intelligence is built, and how the same structure is leveraged to build the Power BI-only solution using whichever features are available.

Part I: Get Data

Chapter 1: Using Power Query to tell your story from your Facebook Data

Author: Anil Maharjan

This chapter is mainly for the one who is trying to extract the story behind their Facebook data by using Power Query. By using Power Query along with Power BI you can extract your Facebook data easily and analyze your own story by using your Facebook data. Power Query can connect data across a wide variety of sources. Facebook is just one of the data sources. This chapter helps you to learn about Power Query and Power BI and shows you how to use self-service BI based on your Facebook data.

Introduction

Most of the time of this weekend, I spent my time to extract the story behind my Facebook data by using Power Query. Power Query can connect data across a wide variety of sources, where Facebook is just one of the data sources. By using Power Query, you can extract your Facebook data easily and do analysis of your own story by using your Facebook data.

Power Query

Power Query is the Microsoft Data Connectivity and Data Preparation technology that enables business users to seamlessly access data stored in hundreds of data sources and reshape it to fit their needs, with an easy to use, engaging and no-code user experience.

Supported data sources include a wide range of file types, databases, Microsoft Azure services and many other third-party online services. **Power Query** also provides a [Custom Connectors SDK](#) so that third parties can create their own data connectors and seamlessly plug them into Power Query.

You can learn more about Power Query from the links below:

<https://docs.microsoft.com/en-us/power-query/power-query-what-is-power-query>

<https://docs.microsoft.com/en-us/power-query/power-query-quickstart-using-power-bi>

Previously, Microsoft Power Query for Excel was only an Excel add-in that enhanced the self-service Business Intelligence experience in Excel by simplifying data discovery, access and collaboration.

You can easily download the Power query Excel Add-In from the link below:

<http://www.microsoft.com/en-us/download/details.aspx?id=39379>

You can find more about Power View, Power Map, Power BI and Q&A from the official Microsoft Power BI site here:

<https://docs.microsoft.com/en-us/power-bi/power-bi-overview>

Let's drill into your Facebook data to extract your story

Start by opening the Power BI Desktop tool which is free one and can easily be downloaded and installed from the link: <https://powerbi.microsoft.com/en-us/desktop/>

Then once you have installed Power BI Desktop, go to the Get Data tab where you will see different data source connection types. Choose the option for more and it will open up as shown:

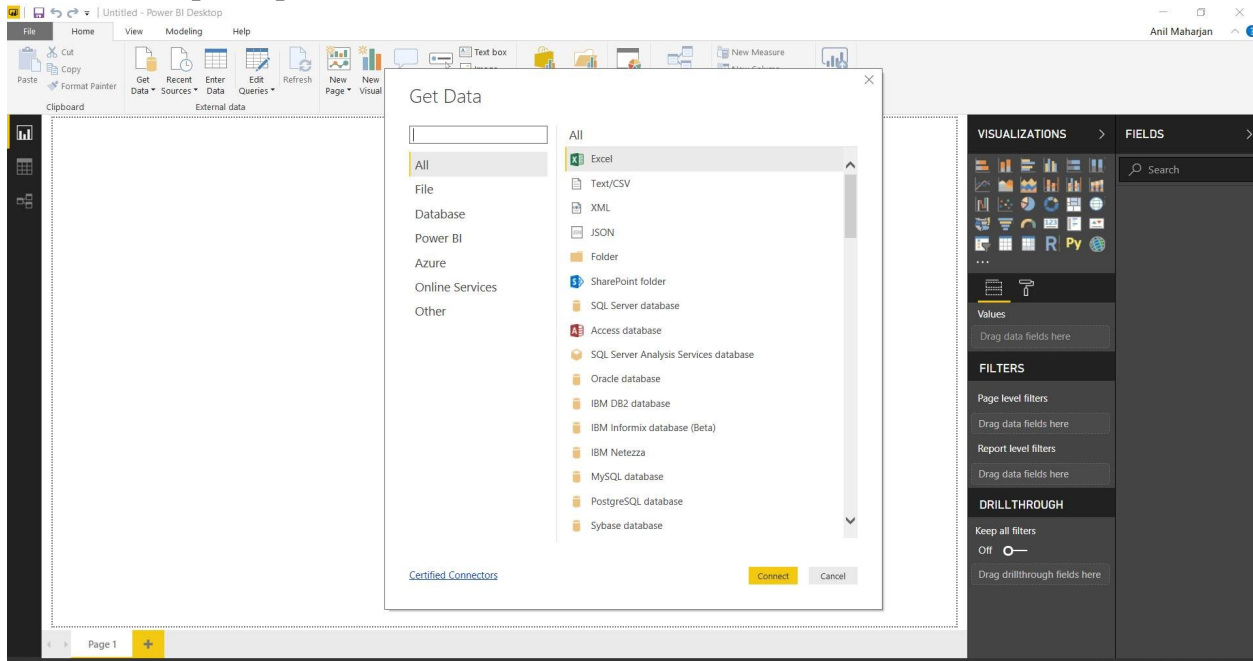


Figure 01-01: Launching the Power BI Desktop -Get Data tab

Choose the online services tab where you can see different online services data sources. Facebook is one of them. We'll connect to it to start the analysis.

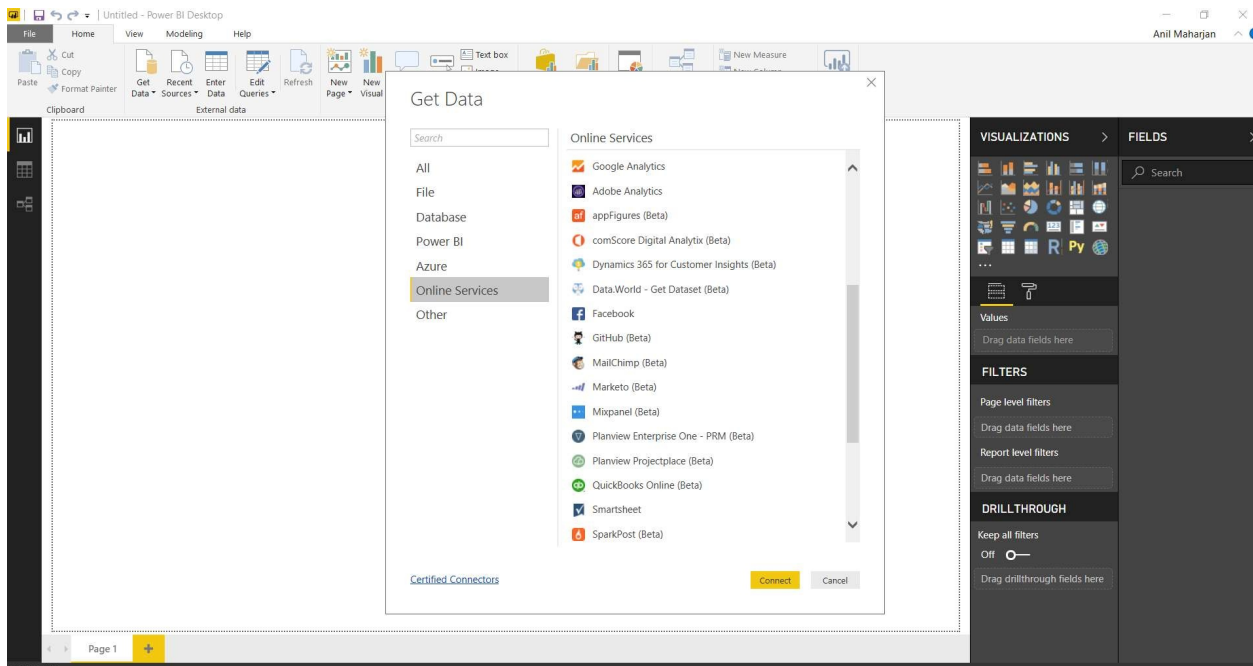


Figure 01-02: Selecting Online Services Facebook as a data source

You could also use the 'Edit Queries' tab in Power BI Desktop and it will open up the Power Query Editor as shown:

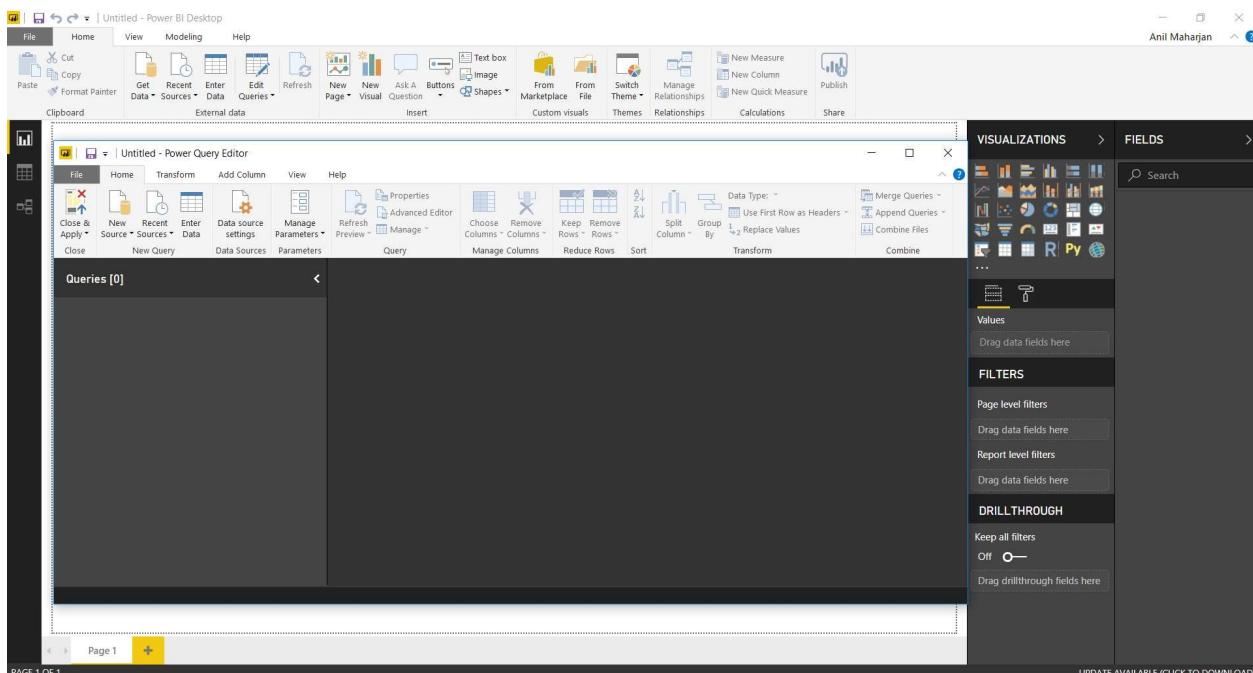


Figure 01-03: Editing Edit Queries section

From here, you can go to New Source tab -> Online Service -> Facebook which will open up as below and ask for you Facebook account credentials. Log in to Facebook and you are able to get your Facebook data as like feed, comments,

likes, friends etc.

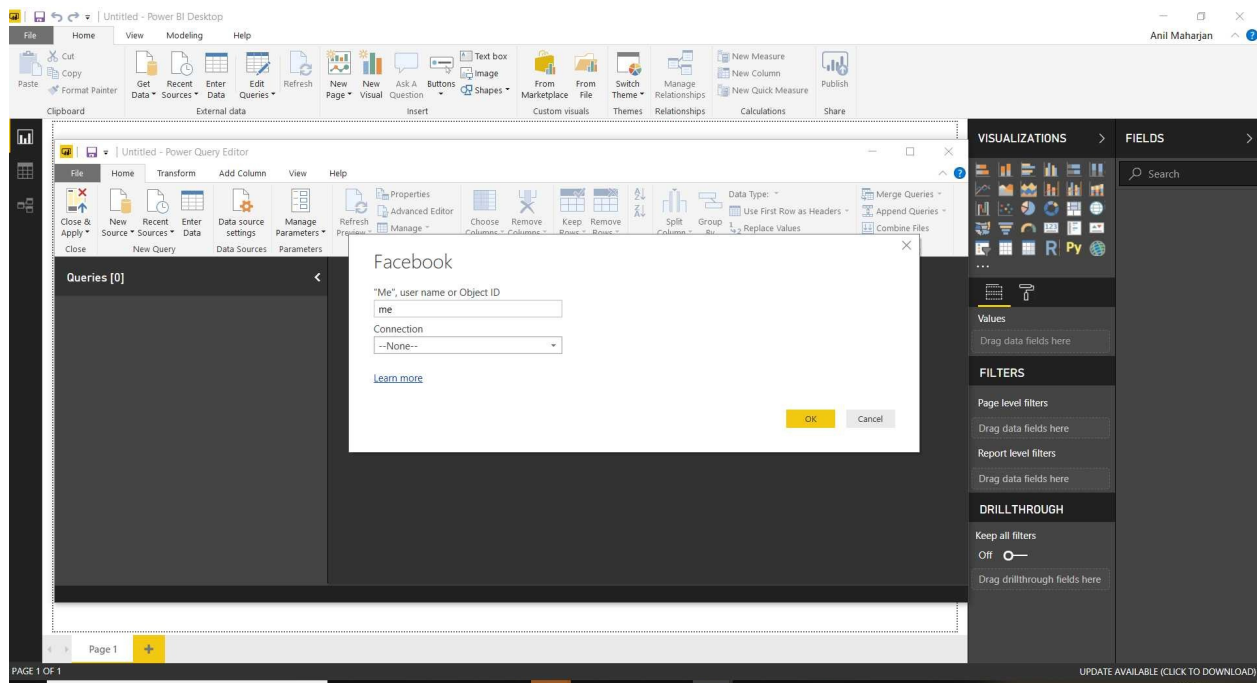


Figure 01-04: Connecting Facebook account credentials login in section

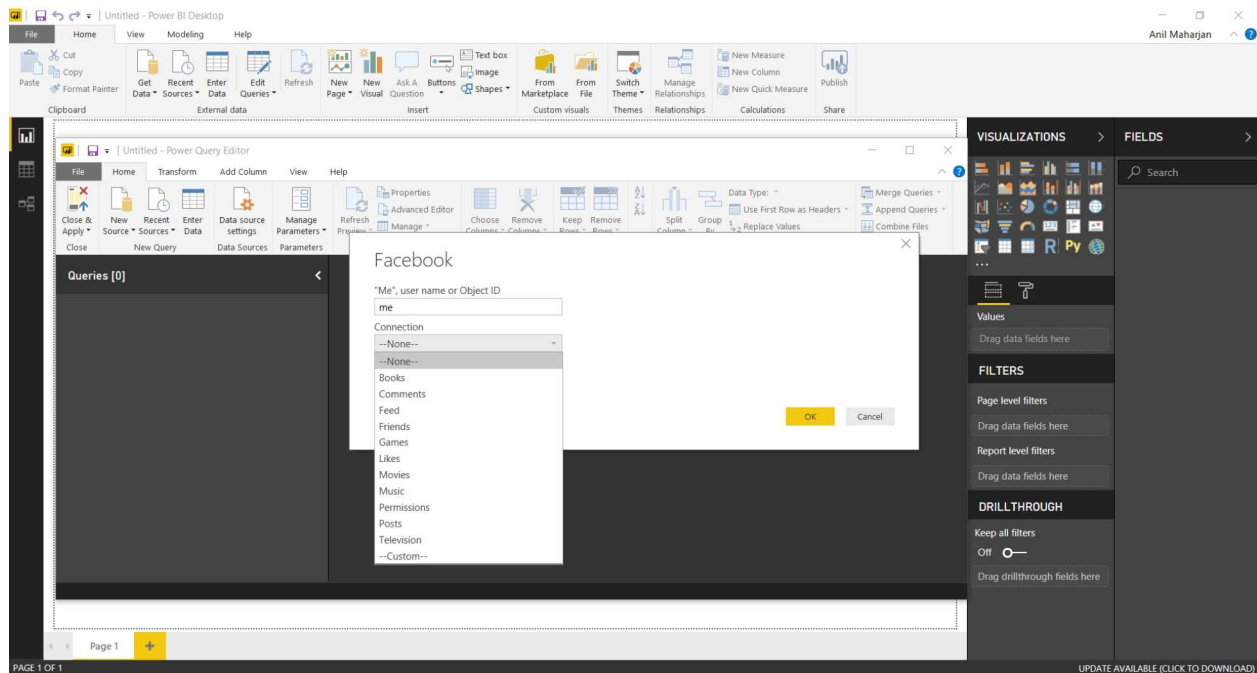


Figure 01-05: Different connection data list from Facebook

Once you are connected, you can select Feed from the dropdown list of connection tab, then it will fetch live data from your Facebook account as:

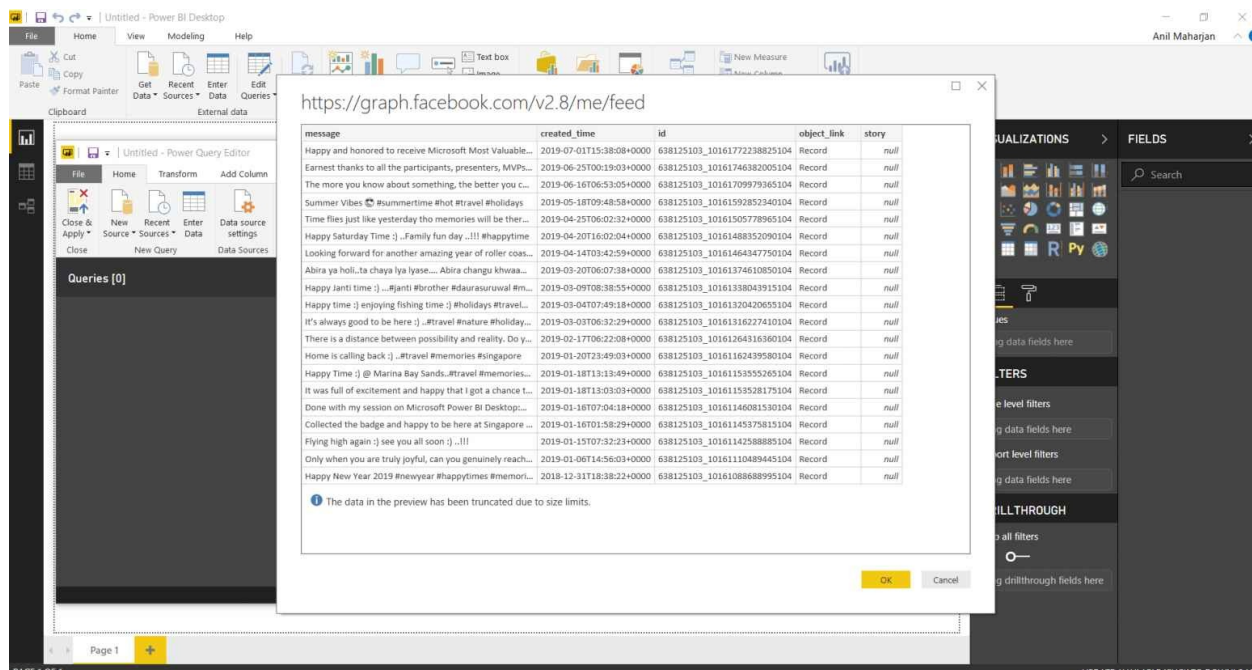


Figure 01-06: Getting Data Feed from Facebook account

Once you have clicked on it, it will load these data and make one Power Query as Query1 where we can rename it to Facebook Feed.

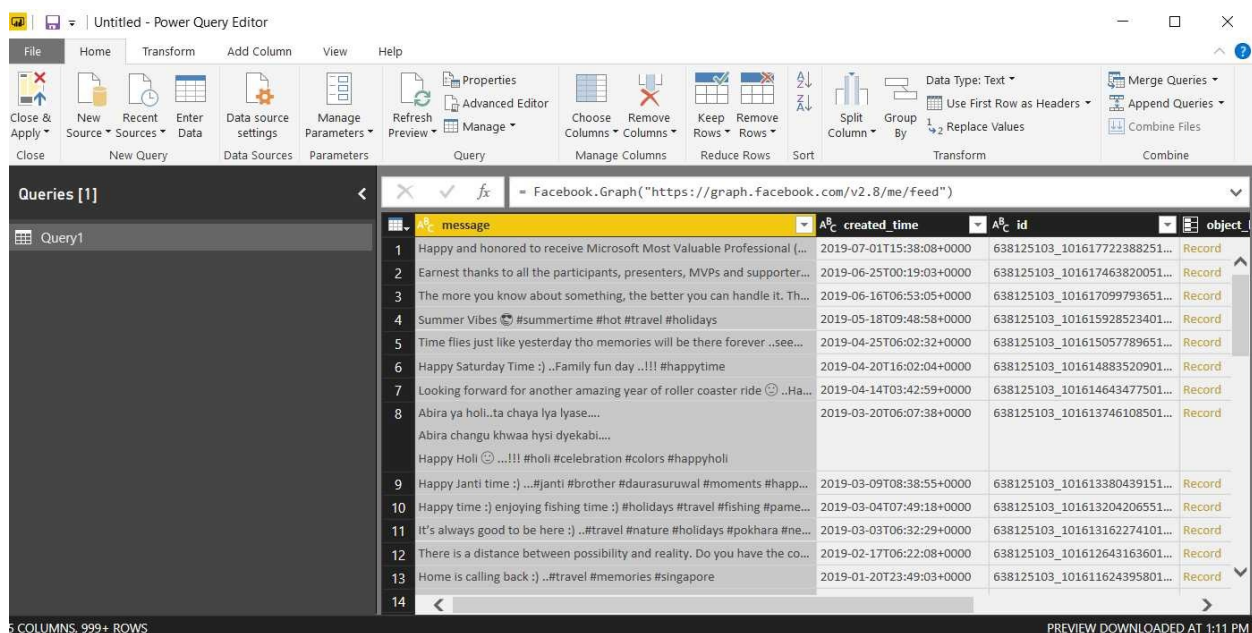


Figure 01-07: Power Query loading data from Facebook

Once you have renamed Query1 to Facebook Feed by right clicking and renaming, now you can edit using the Advanced Editor tab to edit the Power Query and you can see below Power Query which is used to connect with

Facebook and get the data. Here it is using Facebook Graph API v2.8.

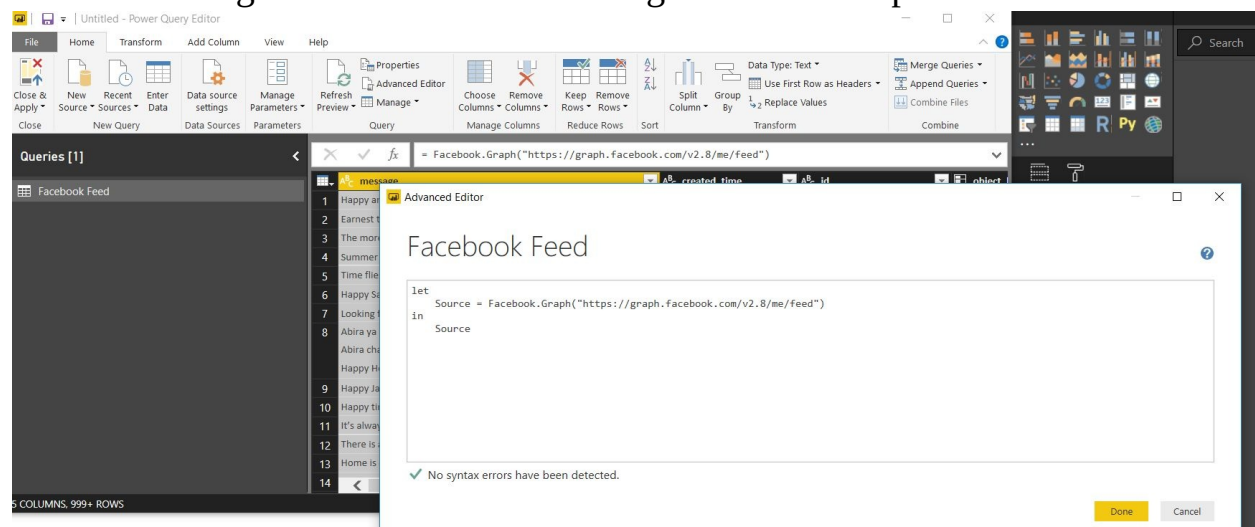


Figure 01-08: Power Query editor and Facebook Graph API connection

Facebook Graph API

The Graph API is the primary way to get data into and out of the Facebook platform. It's an HTTP-based API that apps can use to programmatically query data, post new stories, manage ads, upload photos, and perform a wide variety of other tasks.

One can learn more about Facebook Graph API from below link:

<https://developers.facebook.com/docs/graph-api/overview>

You also can edit the Power Query and add your own custom Power Query. If you need to select only certain year date data from Facebook, then we can select the particular year and do the analysis. For that, you need to edit your Power Query. One can you below Power Query to select only 2019 year date data from Facebook.

let

```
feed = Facebook.Graph("https://graph.facebook.com/v2.8/me/feed"),  
#"Added Custom" = Table.AddColumn(feed, "Year", each  
Date.Year(DateTime.FromText([created_time]))),
```

```
#"Filtered Rows" = Table.SelectRows(feed,each ([Year] = "2019"))
```

in

```
#"Added Custom",  
#"Filtered Rows" = Table.SelectRows(feed, each ([Year] = 2019))
```

in

```
#"Filtered Rows"
```

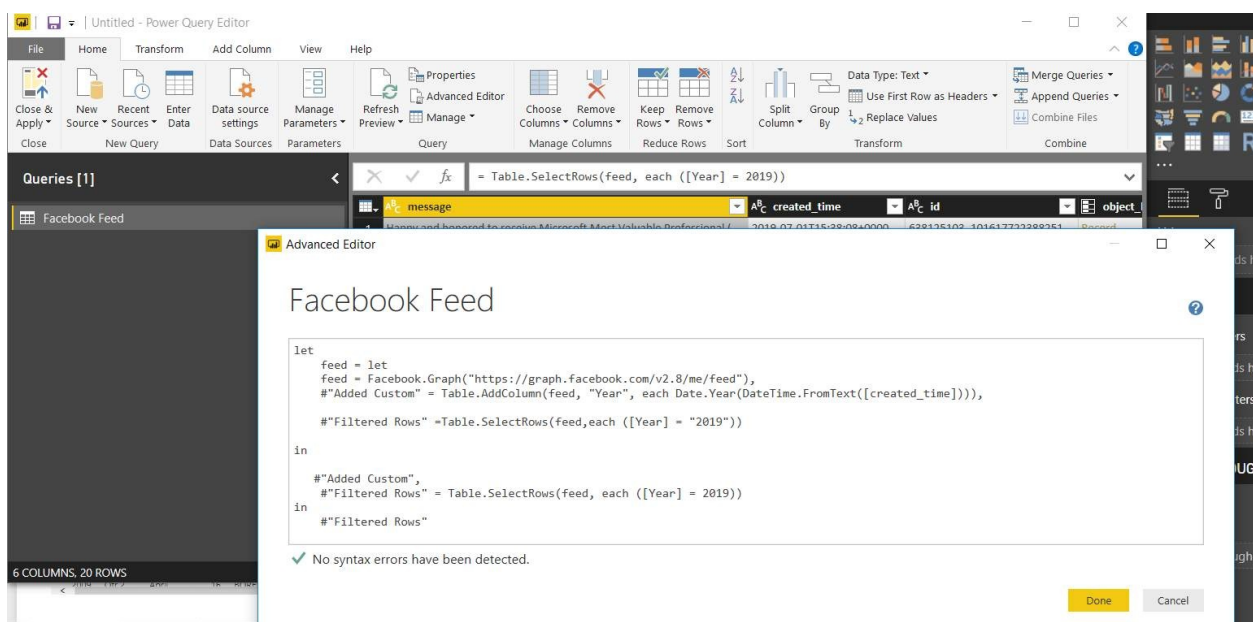


Figure 01-09: Custom Power Query with filter Date

Now once you click on the Close and Apply button and load the feed data from Facebook, let's start to extract some stories behind your own Facebook data.

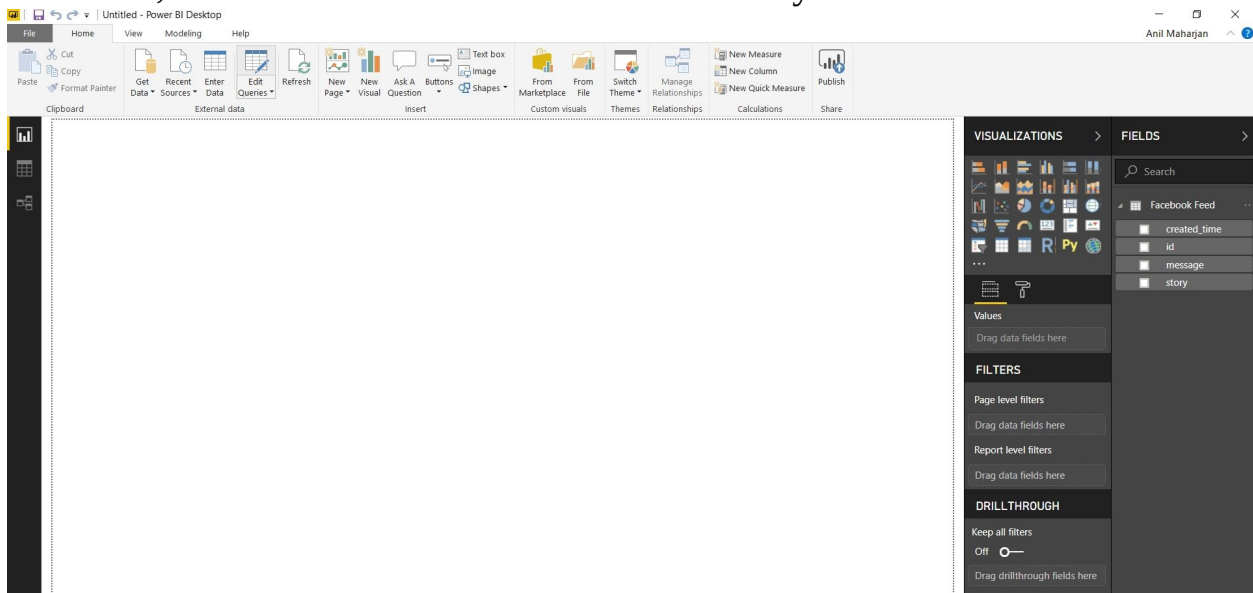


Figure 01-10: Loading data from Facebook

Power Query Analysis 1

Facebook Feed Trend Analysis

Now, let's see how many total feeds there are per year. For this analysis you need to change the *created_time* into date/time format by editing the Facebook feed Query:

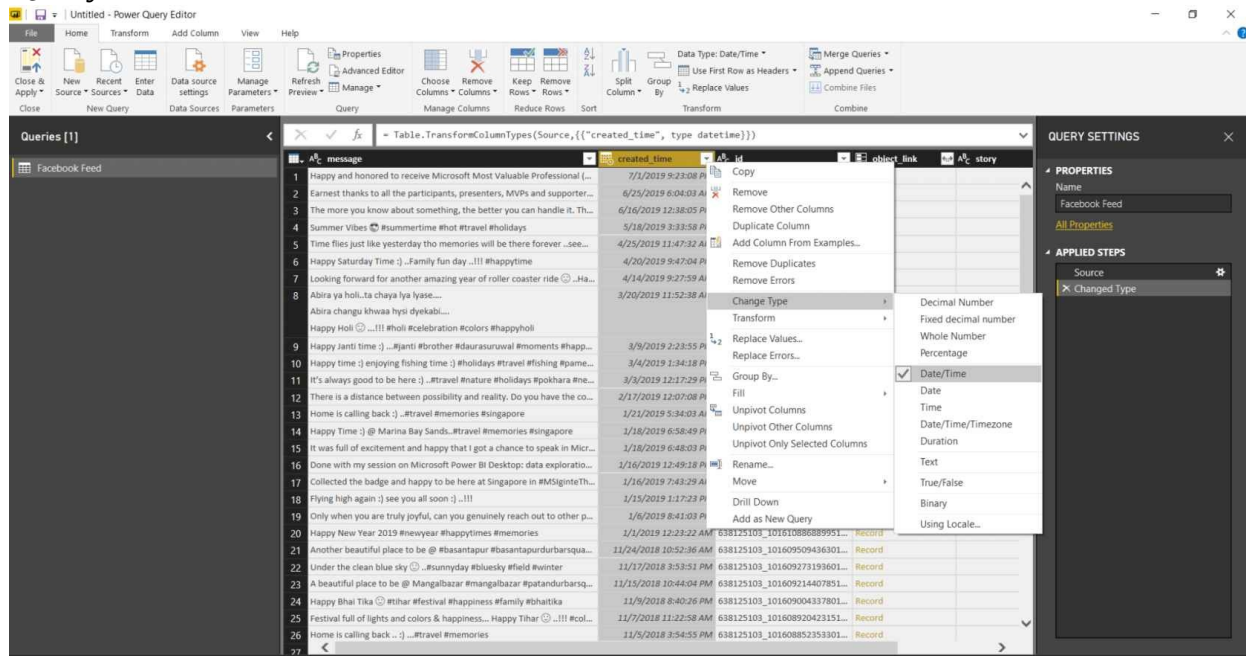


Figure 01-11: changing column format to Date/Time

Let's select Stacked Column Chart from the Visualization section then select *created_time* field in Axis and *id* field in Value. It will automatically make *id* field as count of id in the value section.

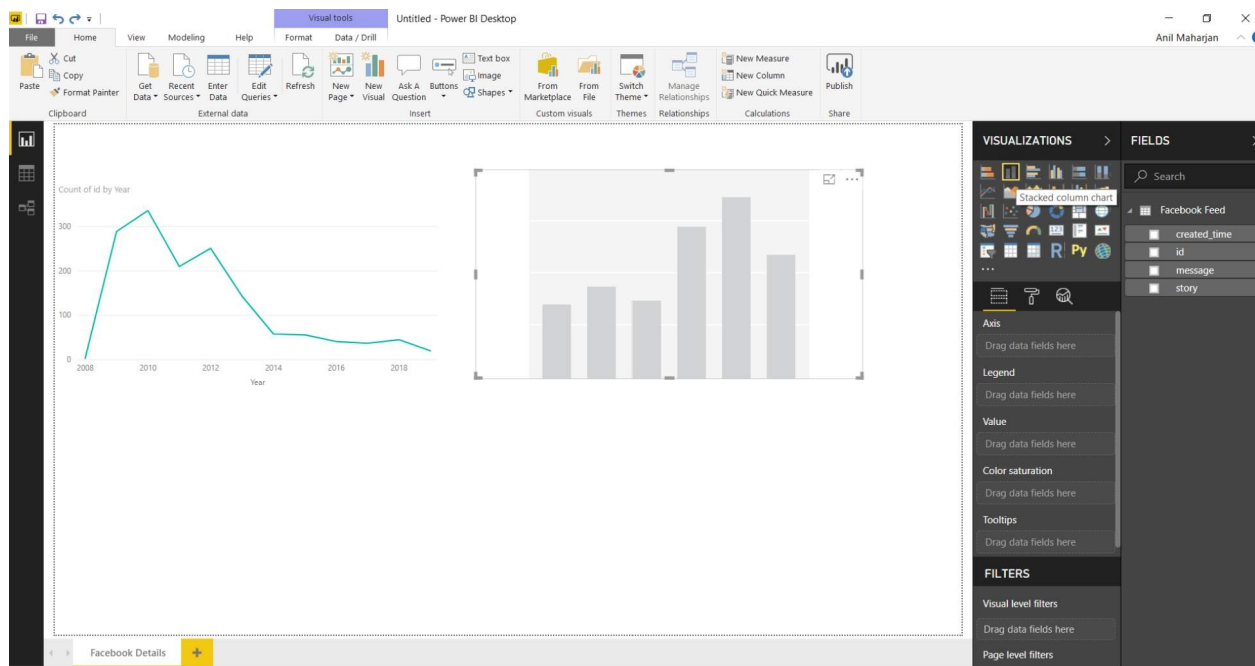


Figure 01-12: Visualization using Power BI Desktop

That will then show us how many total feeds we are doing in Facebook, year on year. Ultimately, this shows how much time you spent on Facebook too. In my context it clearly tells us that I have been using Facebook mostly in the years of 2008, 2009 and 2010. I had completed my computer engineering course in 2010 and I had most free time after my graduation. So, gradually it keeps on decreasing once I joined a company to work.

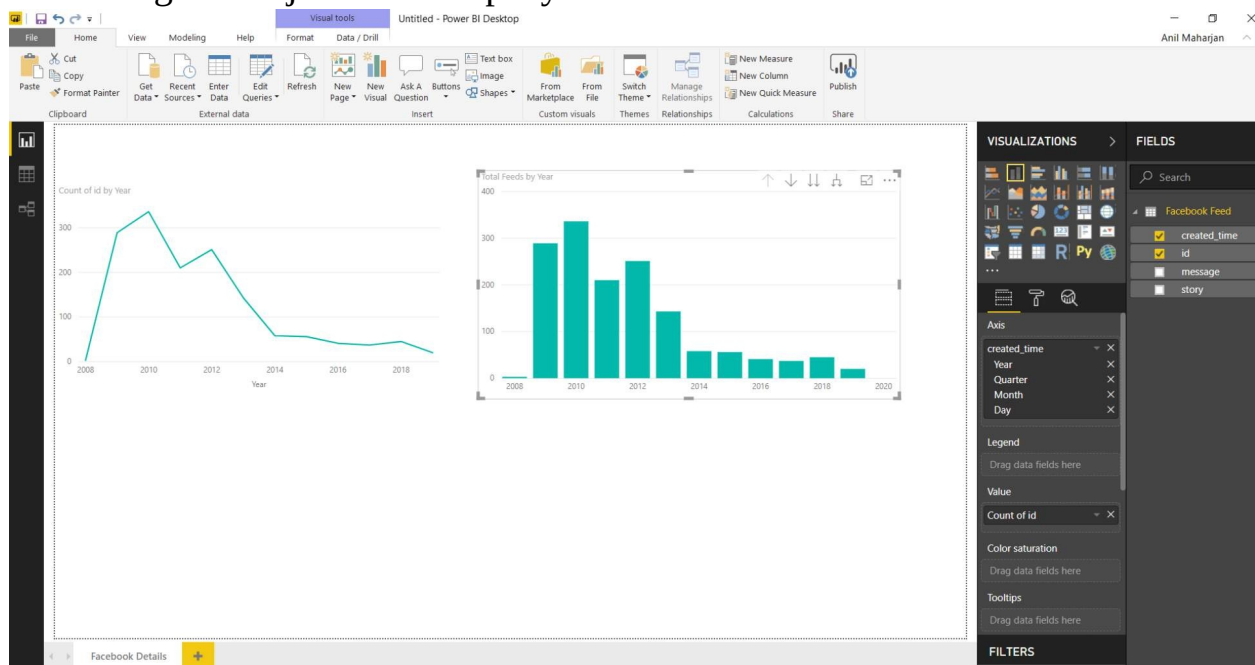


Figure 01-13: Visualization using Power BI Desktop

Similarly, you can add a table visualization chart and select *created_date* and *message* filed in order to see what status/feed or message you have posted on Facebook in any particular time. Here, I have saved this Power BI as Power BI MVP Book and added title of visualization as Facebook Feed Trend Analysis.

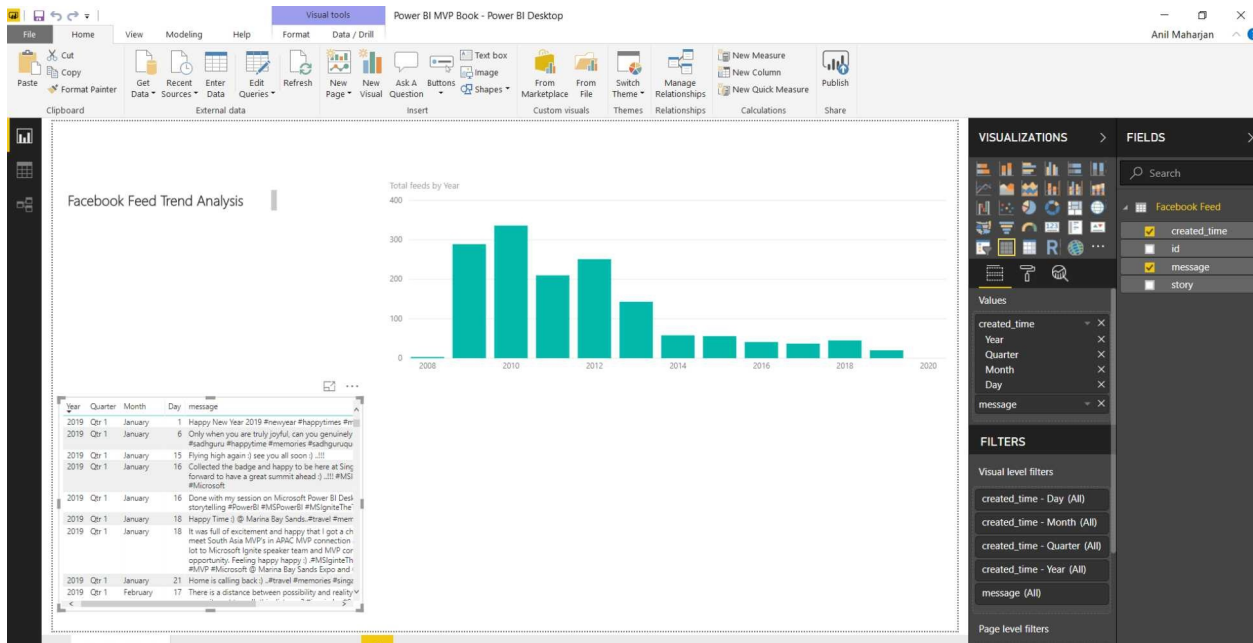


Figure 01-14: Visualization using Power BI Desktop

Then copy the first chart that we had created using Stacked Column Chart and changed into Line Chart where you will see the year on year trend analysis of total feeds. You can also drill down up to Day level.

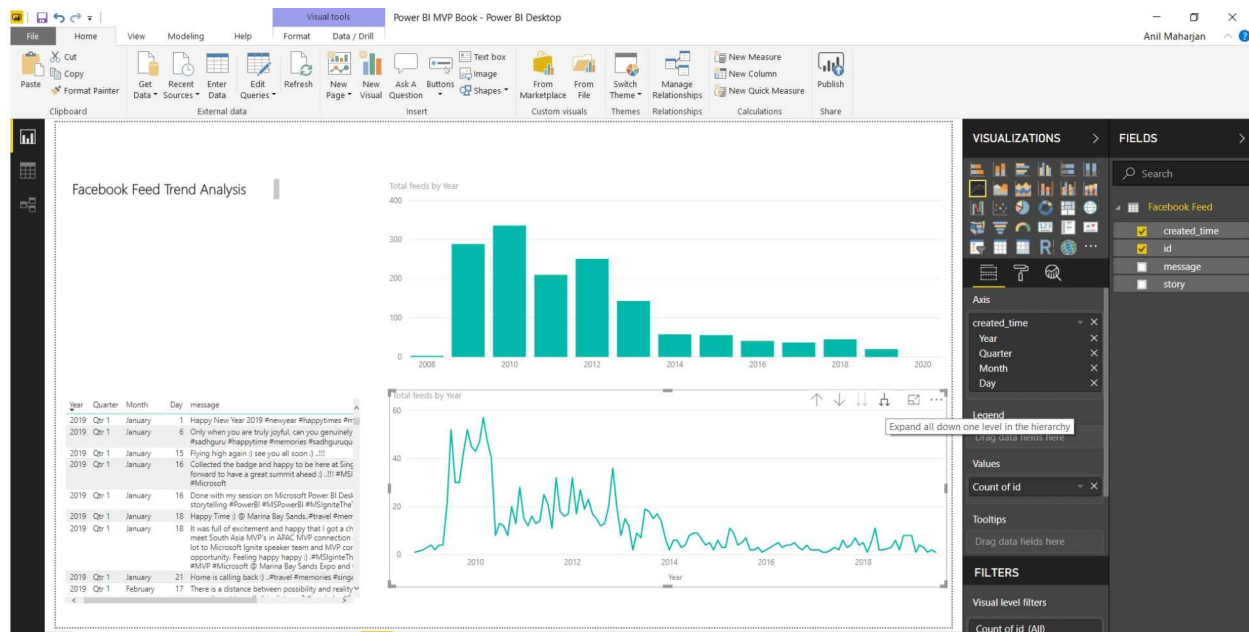


Figure 01-15: Visualization using Power BI Desktop

Power Query Analysis 2

Facebook Photos by Location Tracking

Now, let's see photos that have been taken on particular country or city that you have visited and check In on Facebook. For this analysis, you need to add new data feed and new worksheet page. To add page just you can see the plus sign in the bottom side.

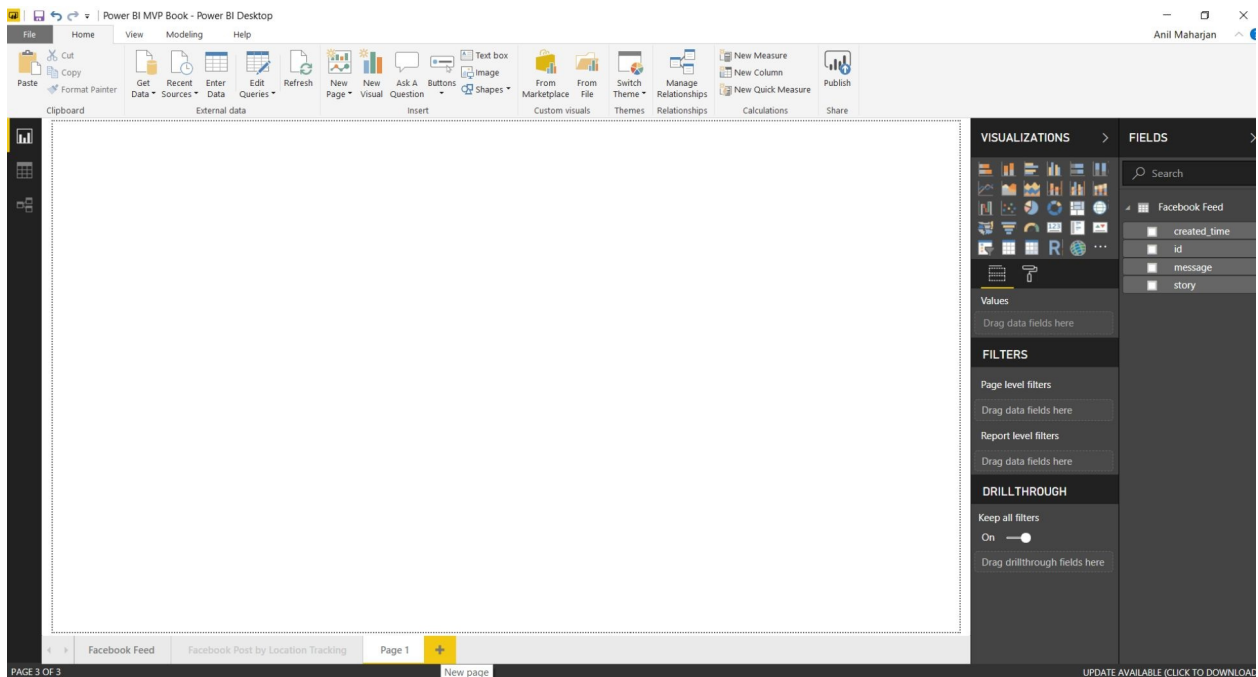


Figure 01-16: Adding new worksheet or page in Power BI Desktop

For adding new data feed from Facebook just repeat click on Get Data->Online Services->Facebook and then select Posts from the dropdown list.

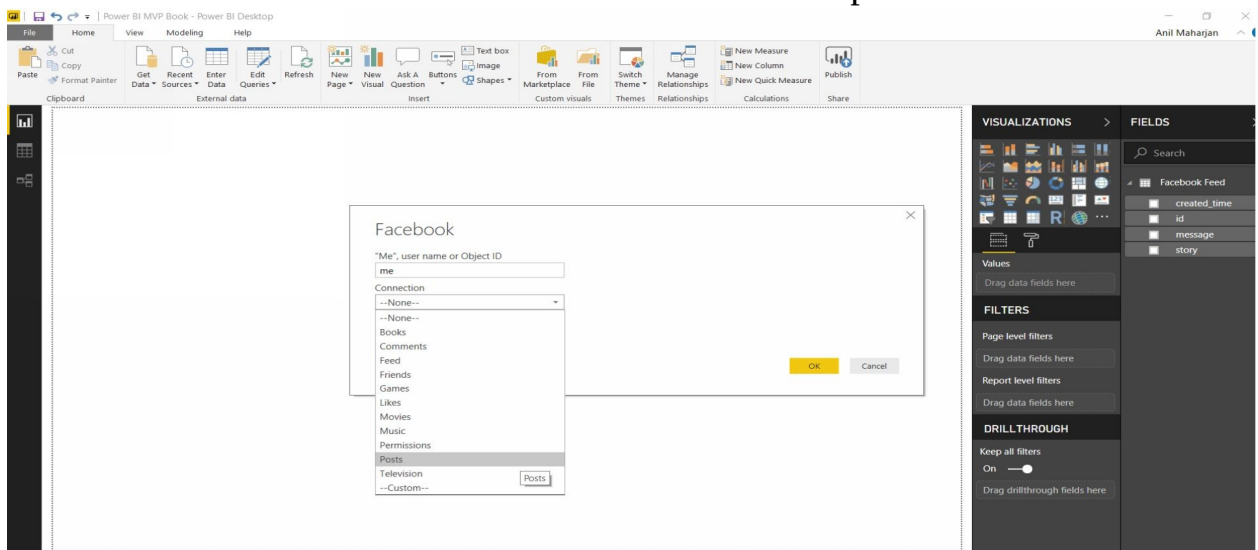


Figure 01-17: Adding new data feed from Facebook in Power BI Desktop

Here, you can also use Blank Query option in Get Data tab where we can write different Power Queries.

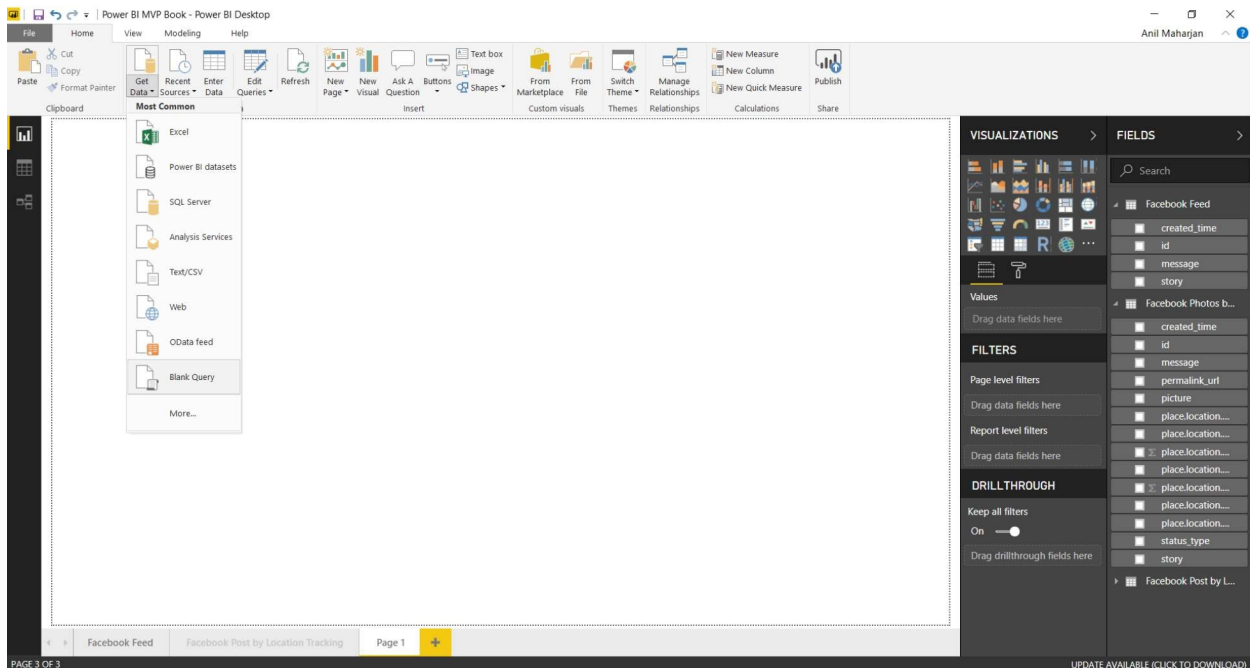


Figure 01-18: Using Blank Query in Power BI Desktop

Once you click on the Blank Query option from the Get Data tab dropdown list ,it will open up as Query1. Next go to the Advanced Editor option .

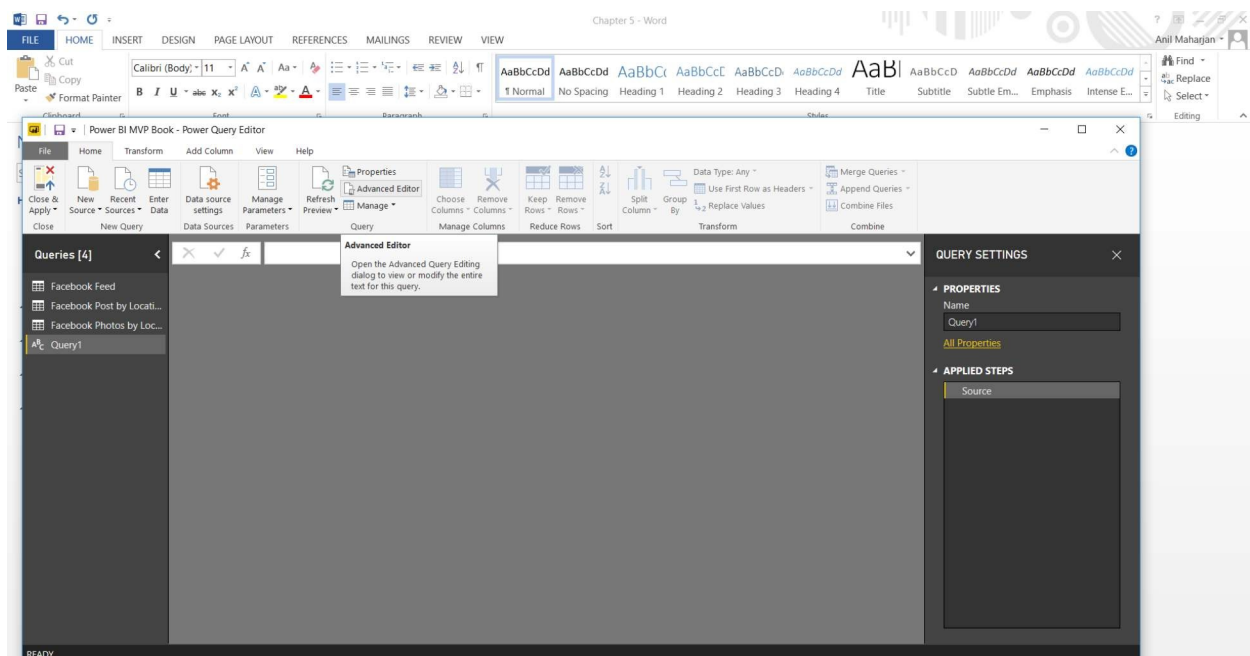


Figure 01-19: Using Blank Query in Power BI Desktop

Once you click Advanced Editor tab it will pop up to where you can write different Power Queries. For Facebook Photos by Location tracking, I will be using the below Power Query.

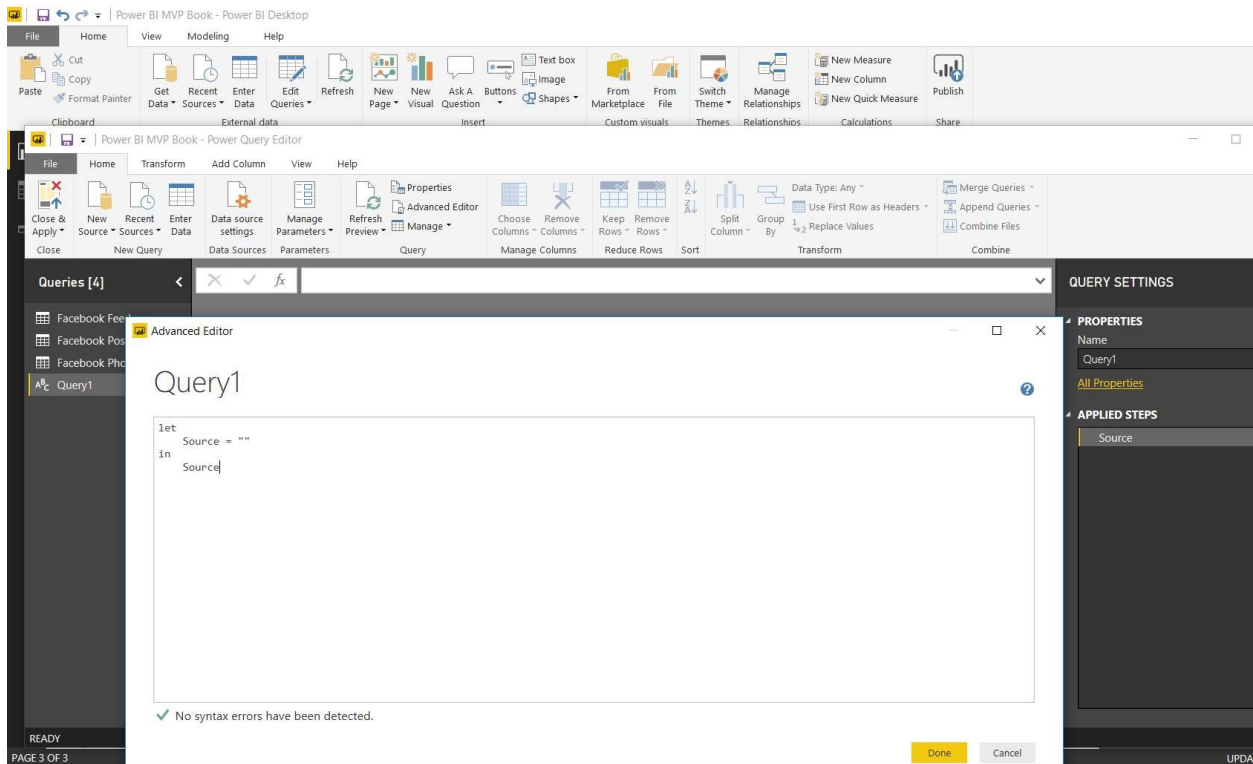


Figure 01-20: Using Blank Query in Power BI Desktop

let

```
Source = Facebook.Graph("https://graph.facebook.com/v2.8/Me/posts?
fields=place,message,story,status_type,created_time,id,permalink_url,picture&with=location"),
    #"Expanded place" = Table.ExpandRecordColumn(Source, "place", {"location"}, {"place.location"}),
    #"Expanded place.location" = Table.ExpandRecordColumn(#"Expanded place", "place.location",
{"city", "country", "latitude", "longitude", "street", "zip", "located_in"}, {"place.location.city",
"place.location.country", "place.location.latitude", "place.location.longitude", "place.location.street",
"place.location.zip", "place.location.located_in"}),
    #"Changed Type" = Table.TransformColumnTypes(#"Expanded place.location",
{{"place.location.latitude", type number}, {"place.location.longitude", type number}, {"created_time", type
datetime}}),
    #"Filtered Rows" = Table.SelectRows(#"Changed Type", each true)
in
    #"Filtered Rows"
```

After that, put the above Power Query in the Advanced Editor and rename Query1 to Facebook Photos by Location.

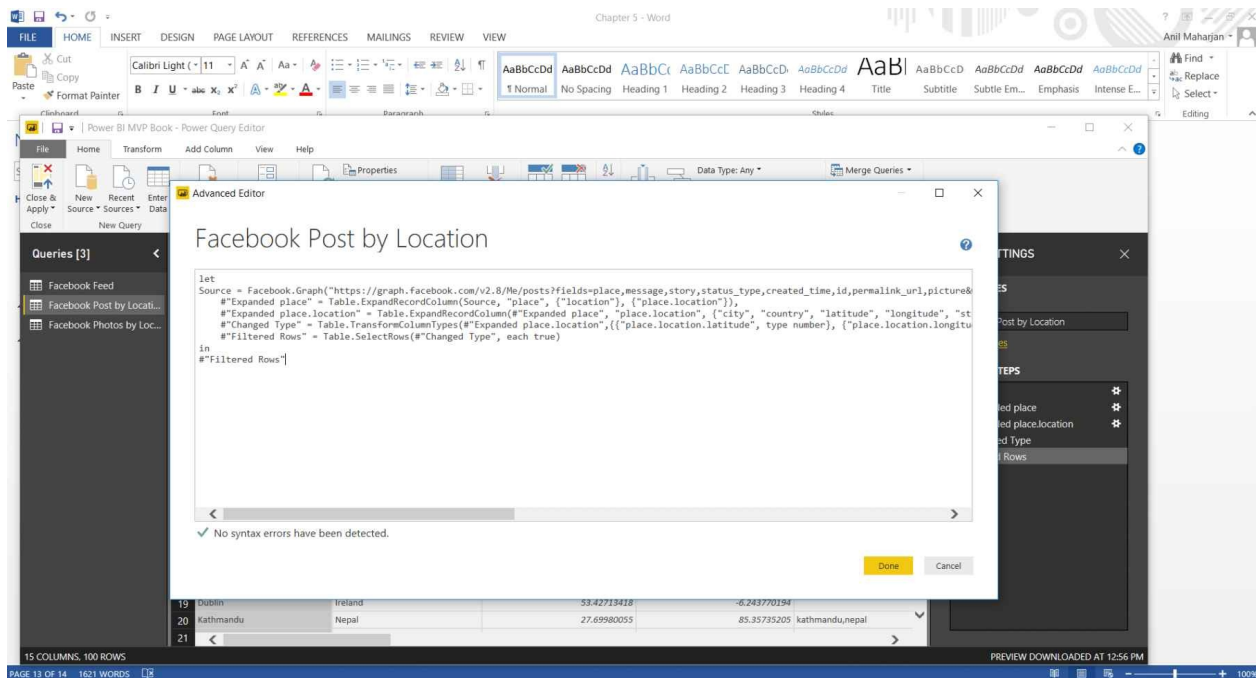


Figure 01-21: Using Blank Query in Power BI Desktop

Then you need to change the *picture*, *permalink_url* and *Geo location* fields with correct data category type in order to render picture as image, url as links and Geo location fields will work on maps. So for this, you need to go to the Modeling tab and select the appropriate Data Category value from dropdown list based on the column type.

Please map to below Data Category type.

Field	Data Category
<i>picture</i>	<i>Image URL</i>
<i>permalink_url</i>	<i>Web URL</i>
<i>city</i>	<i>City</i>
<i>country</i>	<i>Country/Region</i>
<i>latitude</i>	<i>Latitude</i>
<i>longitude</i>	<i>Longitude</i>

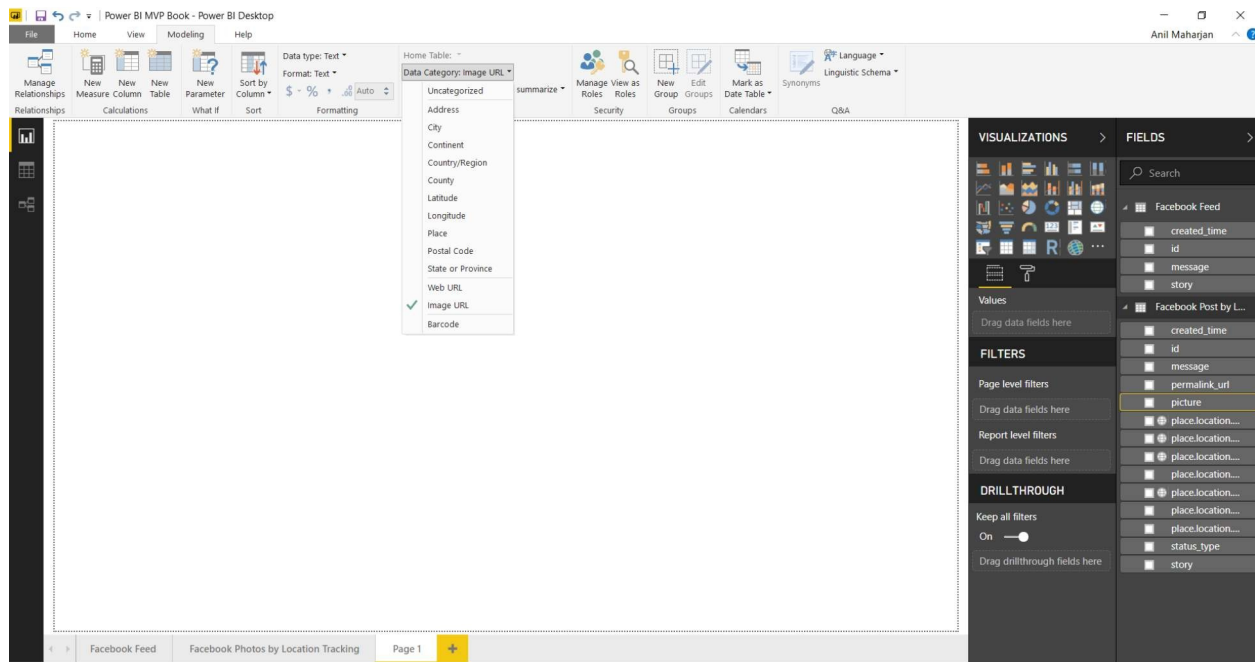


Figure 01-22: Modeling Data Category in Power BI Desktop

Now let's start the visualization to see the photos that have been taken on particular country or city that you have visited and check In on Facebook. For this select ArcGIS Maps from Visualizations section then select the *picture* in Size, *place.location.city* field into Location and *place.location.country* into the Color section.

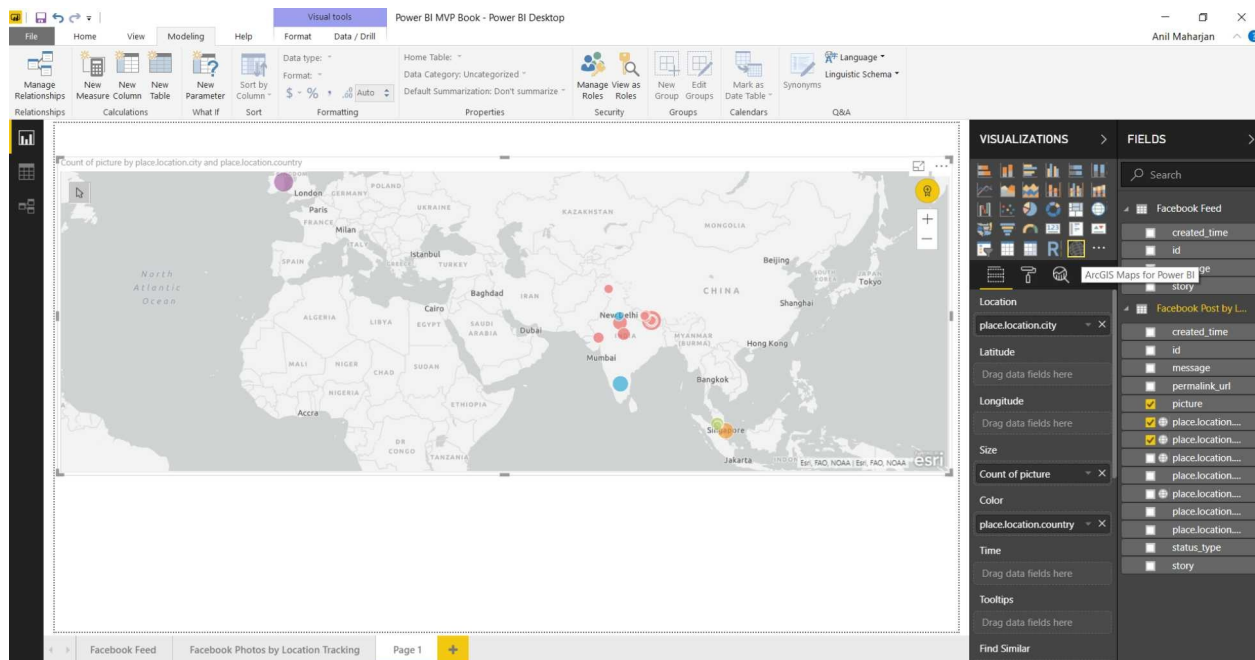


Figure 01-23: ArcGIS Maps Visualization in Power BI Desktop

Further select another Matrix Visualization tab where you select the *Picture* field in columns and *place.location.city* field in Values then it will show the Picture image and following location where you have taken that photo and checked-In in Facebook.

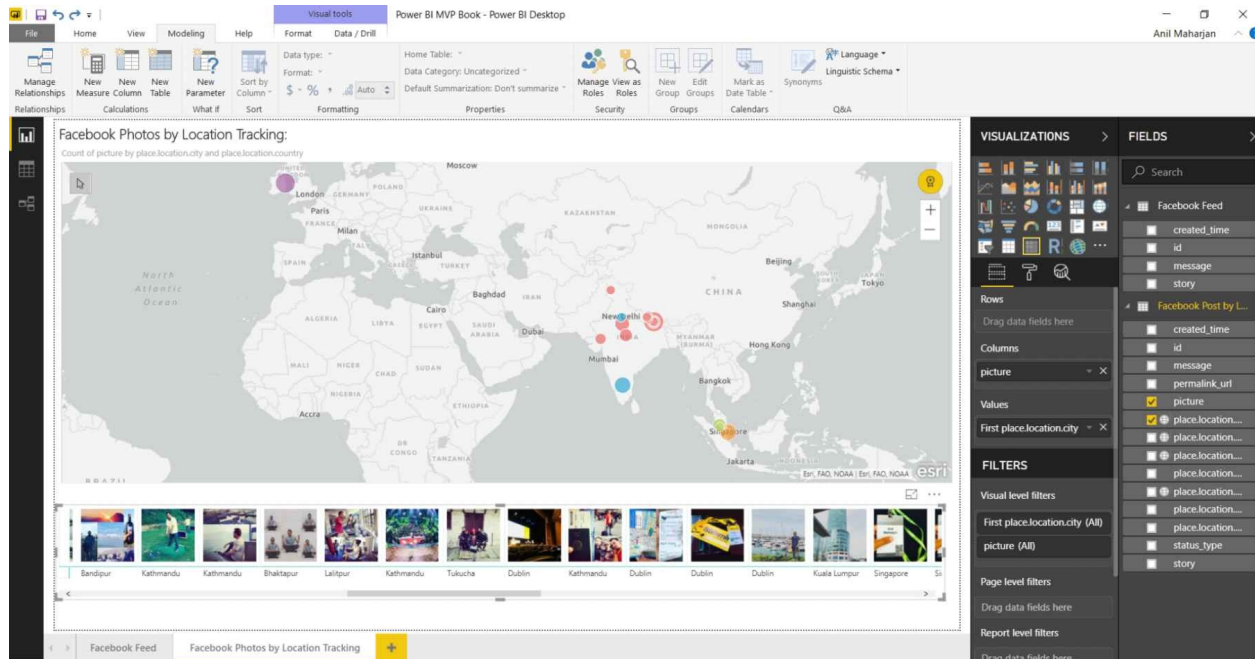


Figure 01-24: ArcGIS Maps Visualization and Photos Image in Power BI Desktop

From this Visualization analysis, you can easily see and track the places you have visited around the world and Photos taken or memories from the places that you have visited and had checked-in using Facebook. From above Map Visualization, I can clearly see that I have visited countries like Singapore, Malaysia and Ireland. We can select particular city or location and see selected pictures that we had posted during visit in that place. Below are some photos that I have posted while visiting the amazing city of Dublin. Quite good old memories.

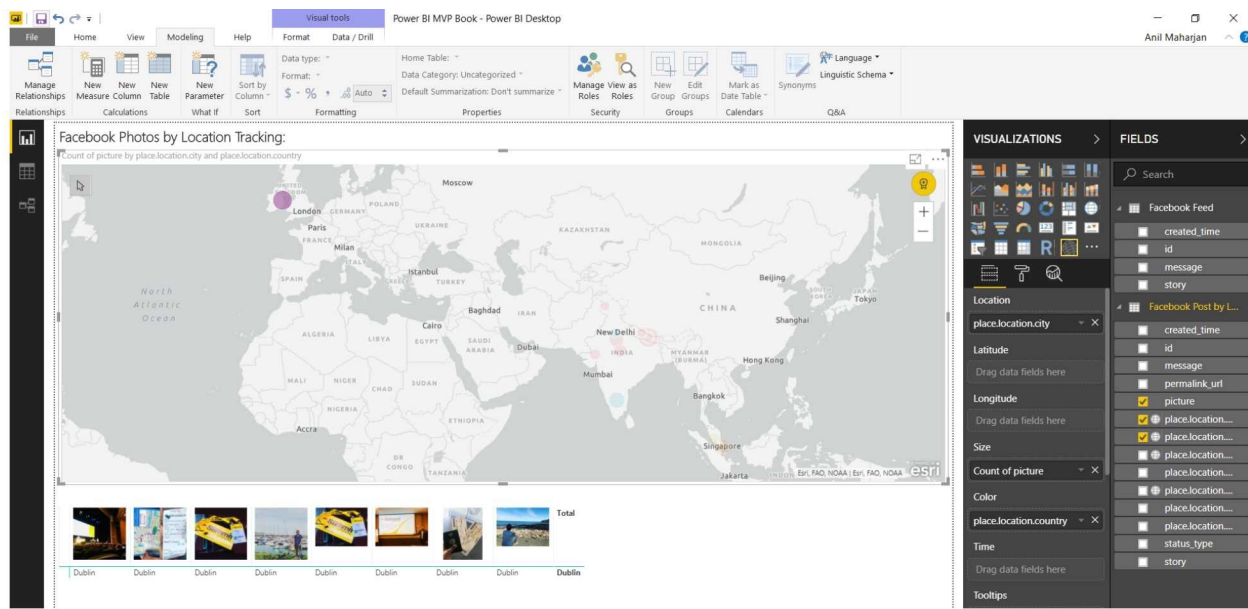


Figure 01-25: Country level filter Photos Image in Power BI Desktop

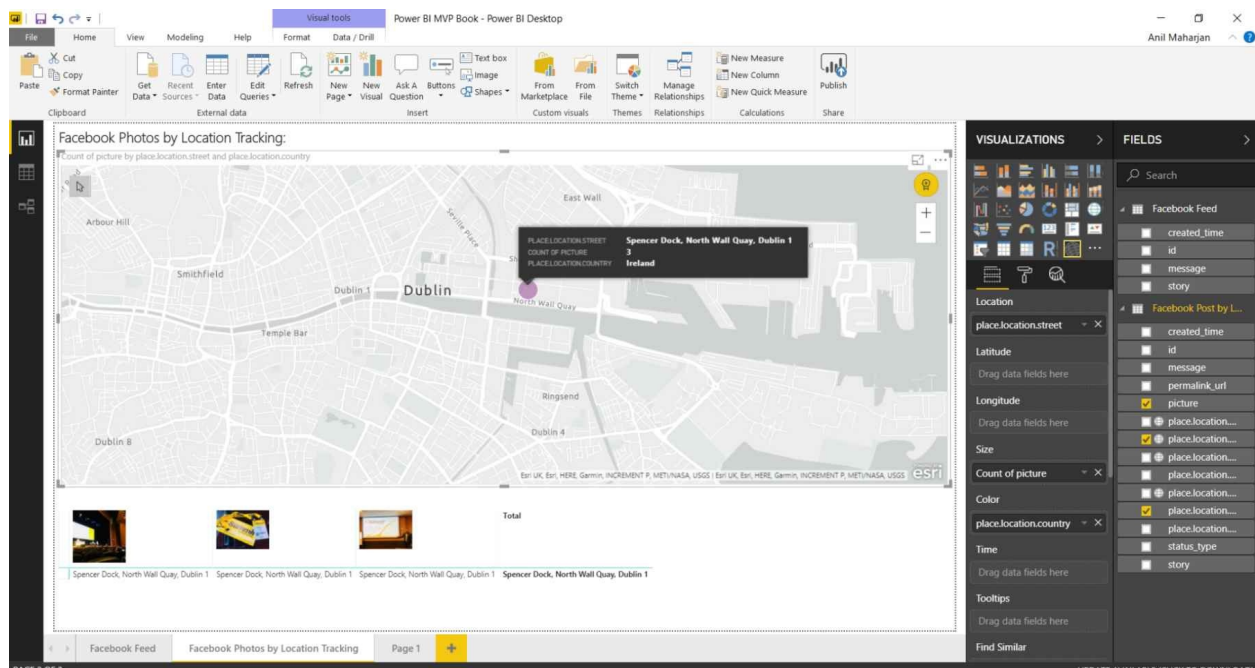


Figure 01-26: City level Drill Photos Image in Power BI Desktop

You can also publish this visualization in Power BI Service by just clicking Publish button. You should have account credentials in the Power BI Service. You can learn more from the link below:

<https://powerbi.microsoft.com/en-us/>

Summary

Power Query along with Power BI can actually tell us a story about you by using your Facebook data. I was happy to find out about the days I have spent on Facebook, the photo memories, and the places that I have visited. It also reminds me of my past college life.

This is such a cool tool, Power Query along with Power BI. You can visualize the things you just want to see.

About the Author



Anil Maharjan is a Microsoft Data Platform MVP, has more than 8 years of development & implementation experience in HealthCare Data Analytics, Telecommunication Industry as a BI Developer, Database consultant and Senior BI Engineer. He is a frequent blogger and speaker at local SQL Server User groups, Power BI User Group, SQL Saturday, Data and BI Summit 2018 – Dublin, Microsoft Ignite Singapore - 2019 and other SQL and Power BI Events. He is also an organizing member at the Himalayan SQL Server User Group and a User Group Leader of the Nepal Power BI User Group. Anil was a Program Committee member for PASS Summit 2014, DATA and BI Summit 2018-Dublin and Committee Track Leader for Power Platform Summit -2019 Australia.

Professional Blogging site:

<http://anilmaharjanonbi.wordpress.com>

<http://maharjananil.com.np>

Chapter 2: Get Data from Multiple URLs Using Web By Example

Author: Indira Bandari

In this chapter, there will be detailed steps that show you how to extract data from a web page that does not have a table that is shown straight away in the web connector.

Further, this chapter also covers how to get data from multiple webpages that have similar structures and will demonstrate the usage of parameters and functions.

The process will be described under three stages:

1. Get Data from Web By Example from a single webpage
2. Create a Table
3. Create a Parameter and Function Get data from Multiple URLs.

Get Data from Web By Example from a Single Web Page

The example depicted here is to visualise some of the [RADACAD blog](#) articles that are of interest in Power BI.

Consider the URL [AI Builder – AI Embedded in Power Apps Part 2](#) if we browse to that URL we will see its contents as shown below.

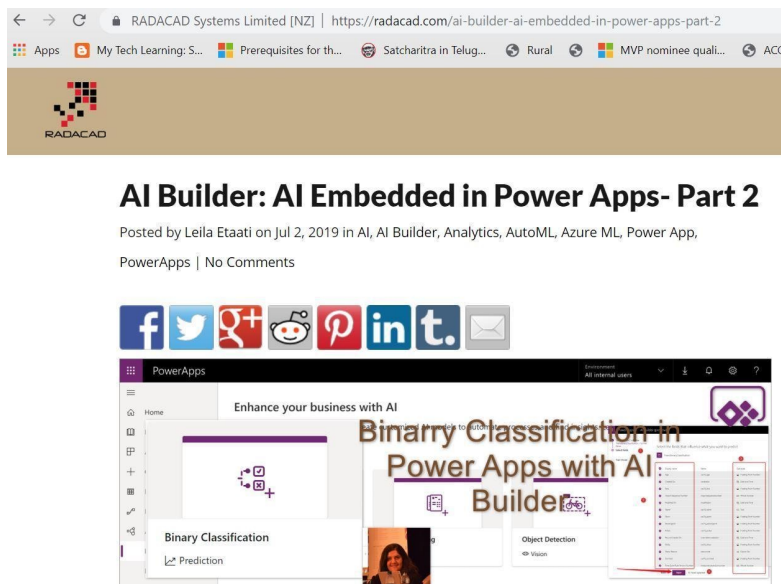


Figure 02-01: Blog URL

We will use Power BI import features to import the following columns: Title, Content, Posted By, Posted On, Categories *etc.*

Below is a step by step guide to import the above data.

Step 1: Open Power BI Desktop and click on the Get Data and Web as shown below:

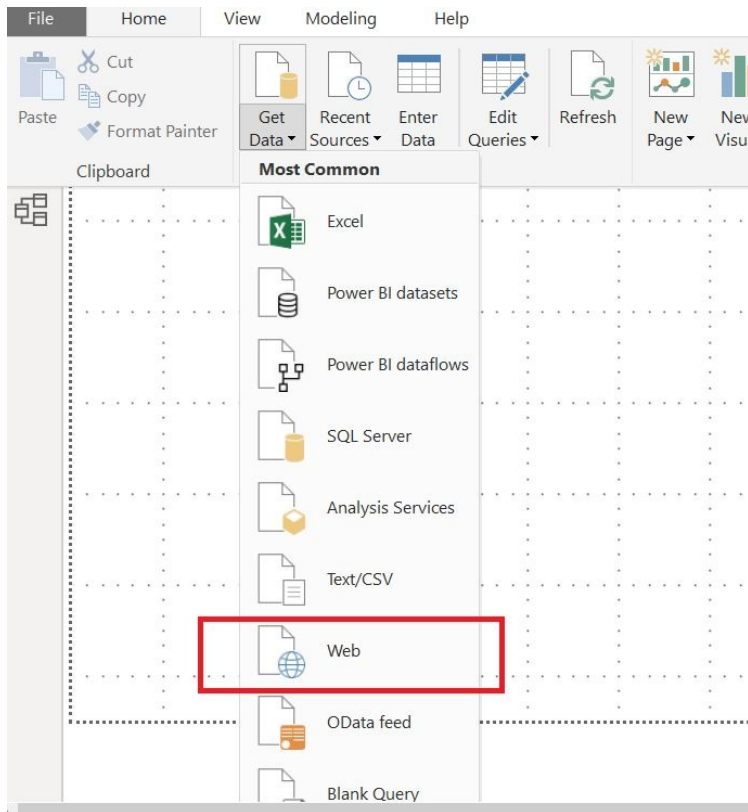


Figure 02-02: Get data from web button

Step 2: Enter the above URL that you have copied into the text box as shown below. Since there is no login required to view the blog articles, choose Basic and click OK.

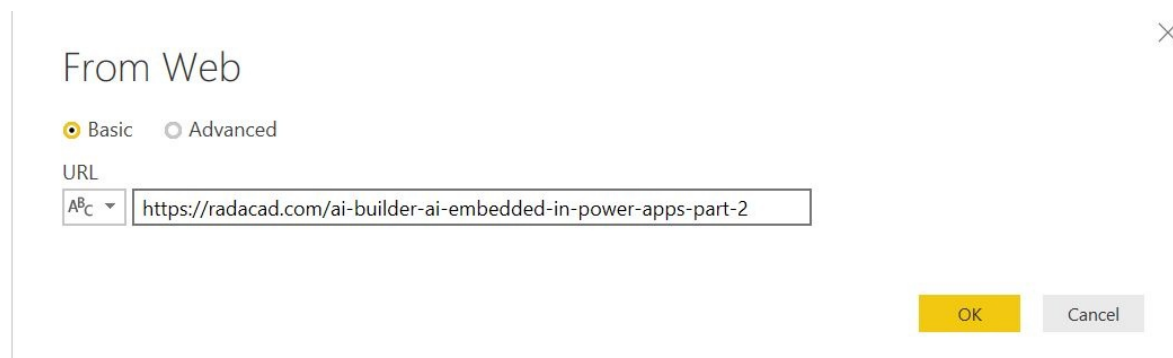


Figure 02-03: Get data from web -- provide URL

Step 3: The Navigator window is shown. If you observe closely, there is a Document that appears in the Display Options section on the left just below the link provided. When you click on the Document, the Table View on the Right side displays four columns – Kind, Name, Children and Text. This is of little use because you cannot easily get the information you want from the page.

If you move down to the bottom left corner of the page, there you see a button named 'Add table using examples'. Click on that button (highlighted).

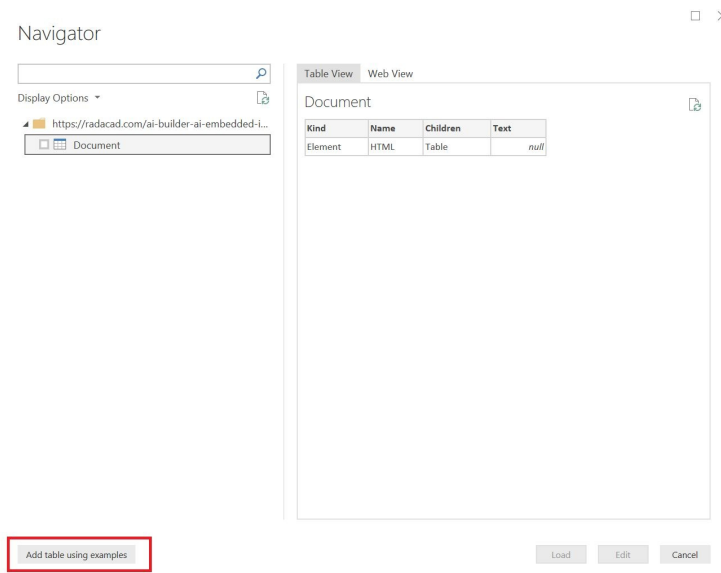


Figure 02-04: Add table using examples

Step 4: This takes you to the preview of the web page as shown below. The below image consists of two sections. One is the preview of the web page and the second section has one column with the name as Column1 as shown below. The web view might take a few seconds to load depending on your Internet connection as well as the amount of content on the page. Scroll to the content that you want to include from the first half of the section.

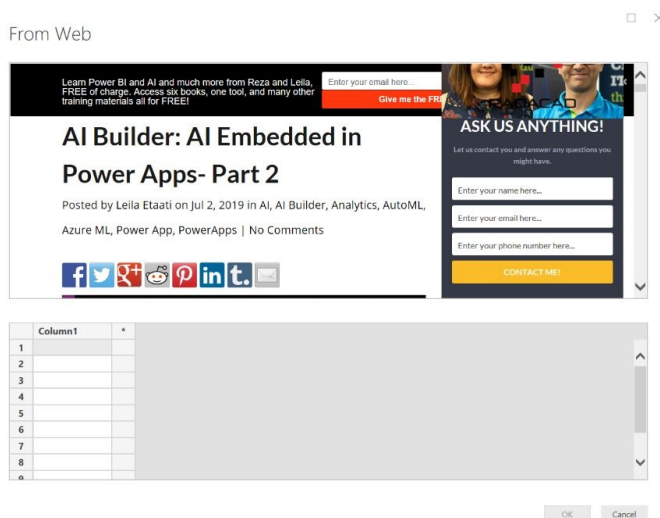


Figure 02-05: Add Column1

Step 5: Click on the first cell of the Column titled Column1. Start typing the

From Web


Learn Power BI and AI and much more from Reza and Leila. FREE of charge. Access six books, one tool, and many other training materials all for FREE!

Enter your email here
[View site the Full](#)

RADACAD Blog

AI Builder: AI Embedded in Power Apps- Part 2

Posted by Leila Etaati on Jul 2, 2019 in AI, AI Builder, Analytics, AutoML



ASK US ANYTHING!

Let us contact you and answer any questions you might have.

[CONTACT ME!](#)

Column1	
1	A
2	Atc: Ai
3	Atc: [ZT]
4	Ai and Power Bi: The Realm of the Impossible
5	Ai Builder
6	Ai Builder [Z]
7	Ai: Ai Builder: AI Embedded in Power Apps: Part 1
8	Ai: Ai Builder: AI Embedded in Power Apps: Part 1 Image Processing - Services : Deploy AutoML Model and Use it in Power BI- Part 3
9	Ai: Ai Builder: AI Embedded in Power Apps: Part 2

Poste...s, AutoML, Azure ML, Power Apps, PowerApps | No Comments

Ai: Ai Builder: AI Embedded in Power Apps: Part 2 ...ort in Power BI: Give the End User Information about the Data

13aug - 14aug 138.00 amnaug 14Data Science Training - Module 1: Power BI for Data Scientist ~ 2 Days Course Auckland Time Zone

19nov8:00 am- 4:00 amData Science Training - Module 4: AI and Cognitive Services in Applications - Auckland 1 Days Course

22oct- 23oct 228:00 amoct 23Data Science Training - Module 3: A...with Microsoft Services ~ 2 Days Course Auckland Time Zone

23jul8:00 am- 4:00 amData Science Training - Module 4: AI and Cognitive Services in Applications - Auckland 1 Days Course

24sep - 25sep 248:00 amsep 25Data Science Training - Module 2: Data Science with Microsoft Cloud ~ 2 Days Auckland Time Zone

2sep - 3sep 29:00 am- 4:30 pmRADACAD Bootcamp: Power BI and AI - Istanbul, Turkey

5sep9:00 am- 5:00 amPass Summit Seattle Pre-Con: Ai and Power BI: The Realm of the Impossible

8jul - 9jul 88:00 amjul 9Data Science Training - Module 2: Data Science with Microsoft Cloud ~ 2 Days Course, Sydney Australia

About Us Power BI Training Advanced Analytics Training RWIZ Powe...tended Our Trainings Courses RADACAD Blog Our Work Contact Us

Advanced Analytics Training

Step 6: Click on the grey area beside the first cell of the Column titled Column1.

Figure 02-07: Click for Column2

Step 7: That will create a new column titled Column 2.

From Web

Learn Power BI and AI and much more from Reza and Leila, FREE of charge. Access six books, one tool, and many other training materials all for FREE!

Enter your email here...

Give me the FREE!

AI Builder: AI Embedded in Power Apps- Part 2

Posted by Leila Etaati on Jul 2, 2019 in AI, AI Builder, Analytics, AutoML, Azure ML, Power App, PowerApps | No Comments

[f](#) [t](#) [g+](#) [r](#) [p](#) [in](#) [t.](#) [e](#)

ASK US ANYTHING!

Let us contact you and answer any questions you might have.

Enter your name here...

Enter your email here...

Enter your phone number here...

CONTACT ME!

	Column1	Column2	*
1	AI Builder: AI E...		
2			
3			
4			
5			
6			
7			
8			
9			

OK

Cancel

Figure 02-08: Add Column2

Step 8: Start typing the author as Leila and choose from the list displayed.

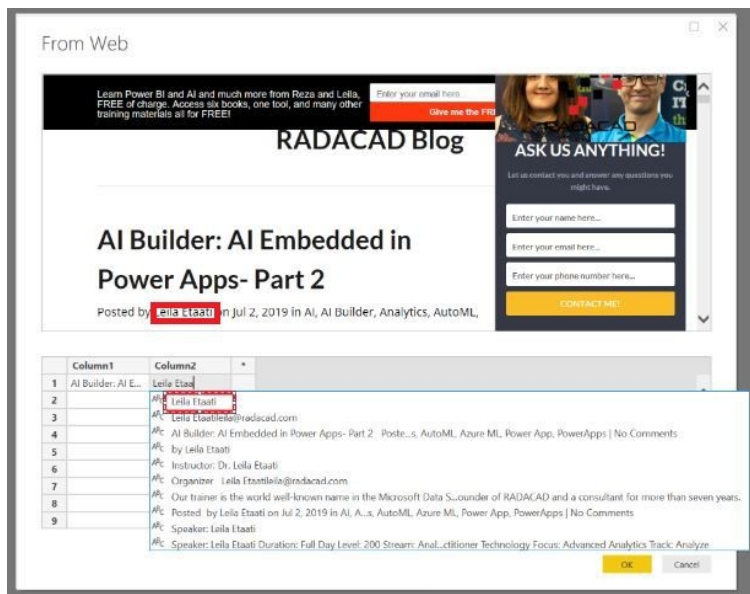


Figure 02-09: Choose Column2 from list

Step 9: Again, click on the grey area beside the first cell of the Column titled Column2 and this will create another column titled Column3.

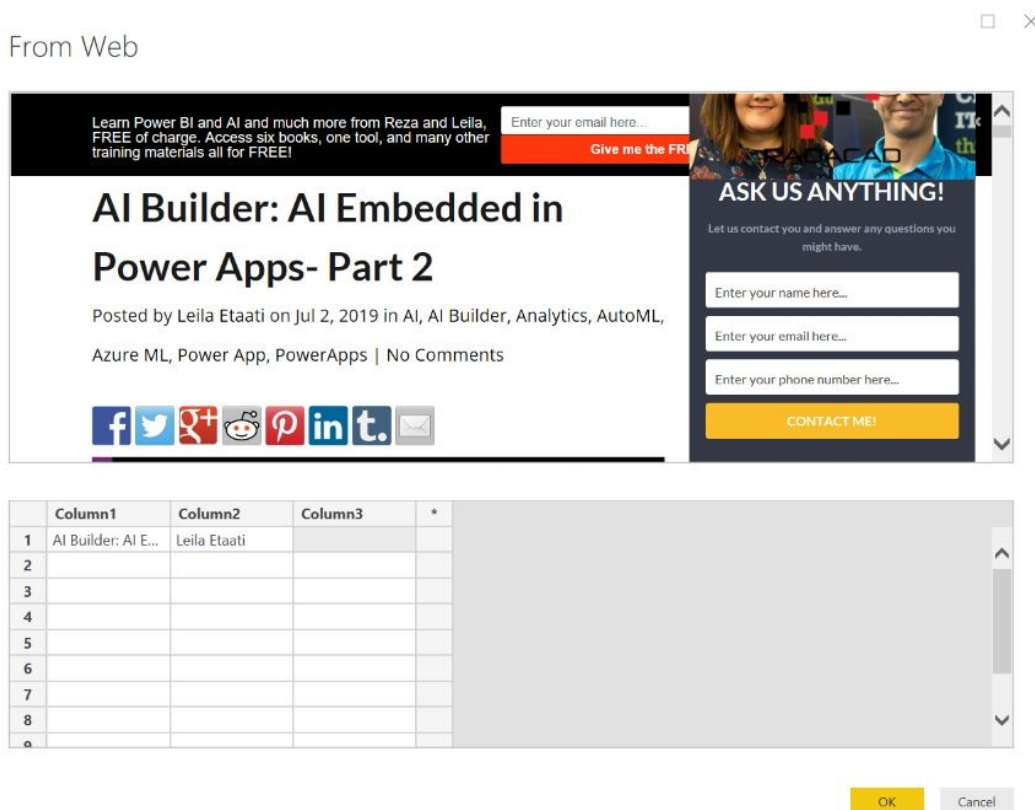


Figure 02-10: Add Column3

Step 10: Start typing the date as Jul 2 and choose from the prompt.

From Web

Learn Power BI and AI and much more from Reza and Leila, FREE of charge. Access six books, one tool, and many other training materials all for FREE!

Enter your email here...
Give me the FREE!

RADACAD Blog

AI Builder: AI Embedded in Power Apps- Part 2

Posted by Leila Etaati on Jul 2, 2019 in AI, AI Builder, Analytics, AutoML,

ASK US ANYTHING!
Let us contact you and answer any questions you might have.
Enter your name here...
Enter your email here...
Enter your phone number here...
CONTACT ME!

No CSS selector was found for the sample values you provided in the following column(s): Column3.

	Column1	Column2	Column3	*
1	AI Builder: AI E...	Leila Etaati	Jul 2, 2019	
2				
3				
4				
5				
6				
7				
8				

OK Cancel

Figure 02-11: Column3 Error

This error means that the data typed in, cannot be recognised automatically as there is No CSS selector defined for the date. Therefore, choose what you can see from the list, which is on Jul 2, 2019. Then the error is rectified.

From Web

Learn Power BI and AI and much more from Reza and Leila, FREE of charge. Access six books, one tool, and many other training materials all for FREE!

Enter your email here...
Give me the FREE!

RADACAD Blog

AI Builder: AI Embedded in Power Apps- Part 2

Posted by Leila Etaati on Jul 2, 2019 in AI, AI Builder, Analytics, AutoML,

ASK US ANYTHING!
Let us contact you and answer any questions you might have.
Enter your name here...
Enter your email here...
Enter your phone number here...
CONTACT ME!

	Column1	Column2	Column3	*
1	AI Builder: AI E...	Leila Etaati	on Ju	
2			on Jul 2, 2019	
3			AI Builder: AI Embedded in Power Apps- Part 2. Poste...s. AutoML, Azure ML, Power App, PowerApps No Comments	
4			I learned a lot from this course: It gave a great overview of wh...course) and step through the material again at your own pace.	
5			Note that Reza covers a lot of ground in this course, and it is...course) and step through the material again at your own pace.	
6			Posted: by Leila Etaati on Jul 2, 2019 in AI, A...s. AutoML, Azure ML, Power App, PowerApps No Comments	
7				
8				
9				

OK Cancel

Figure 02-12: Choose Column3 from list

Step 12: Create the rest of the columns Category, Comments and Content in a similar way.

From Web

	Title	Author	Date	Categories	Comments	Content	*
1	AI Builder: AI Inside Power Apps	Leila Etaati	on Jul 2, 2019	in AI, AI Builder...	No Comments	As mentioned i...	
2							
3							
4							
5							
6							
7							
8							
9							

OK Cancel

Figure 02-13: Populate remaining columns

Step 13: After you are happy with the columns, rename the columns as shown above and click OK. Once you click OK, you will be taken to the screen (below). Make sure that you click the 'Table 1' that is newly created at the bottom left corner of the Navigation pane as highlighted below. Check everything is ok and click Load.

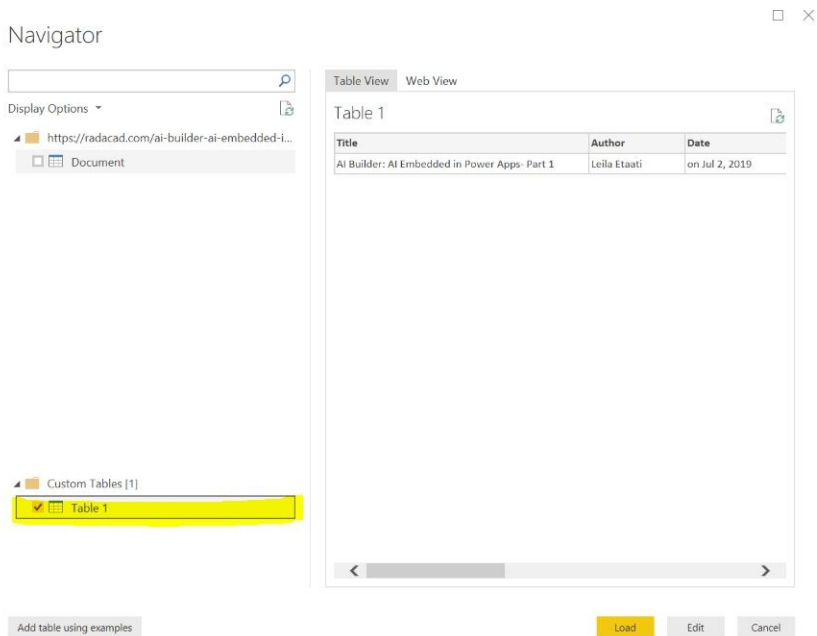


Figure 02-14: Add table from example Navigation Pane

Step 14: You will be taken to the below screen. Rename the Table as ‘Base Query’ as highlighted below.

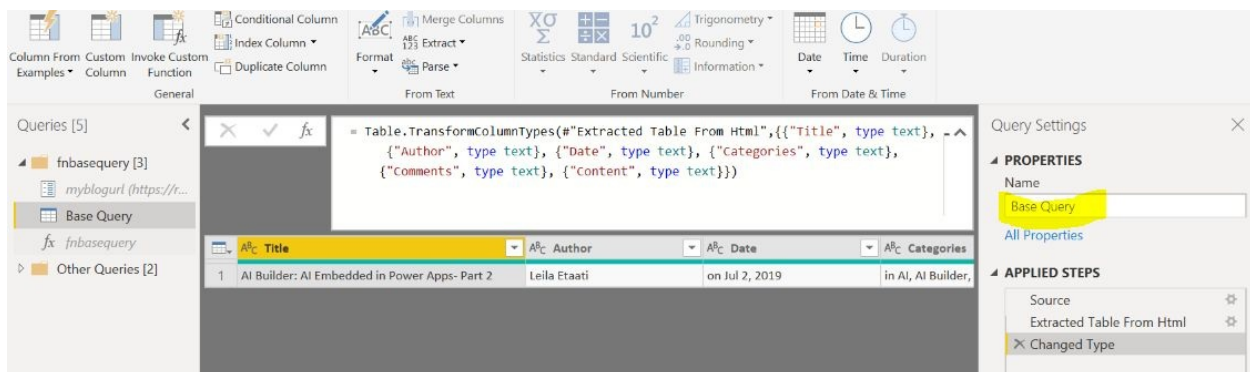


Figure 02-15: Base Query Table

That completes the first stage.

Create a Table

The next stage is to get data from the other blog articles that are of interest. For example, below is a list of URLs that are of interest to us to get into Power BI:

<https://radacad.com/ai-builder-power-apps-embed-with-ai-part-1>

<https://radacad.com/ai-builder-ai-embedded-in-power-apps-part-2>

<https://radacad.com/power-bi-shared-datasets-what-is-it-how-does-it-work-and-why-should-you-care>

<https://radacad.com/automated-machine-learning-data-profiling-in-azure-ml-services-part-4>

<https://radacad.com/azure-machine-learning-services-deploy-automl-model-and-use-it-in-power-bi-part-3>

<https://radacad.com/budget-vs-actual-zero-complexity-model-in-power-bi>

To do this, a table needs to be created where these URLs will be included in a column. Click on the ‘Enter Data’ button in the Power Query Editor as shown below:

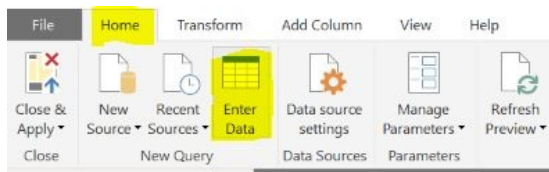


Figure 02-16: ‘Enter Data’ button in Power Query

The below screen appears:

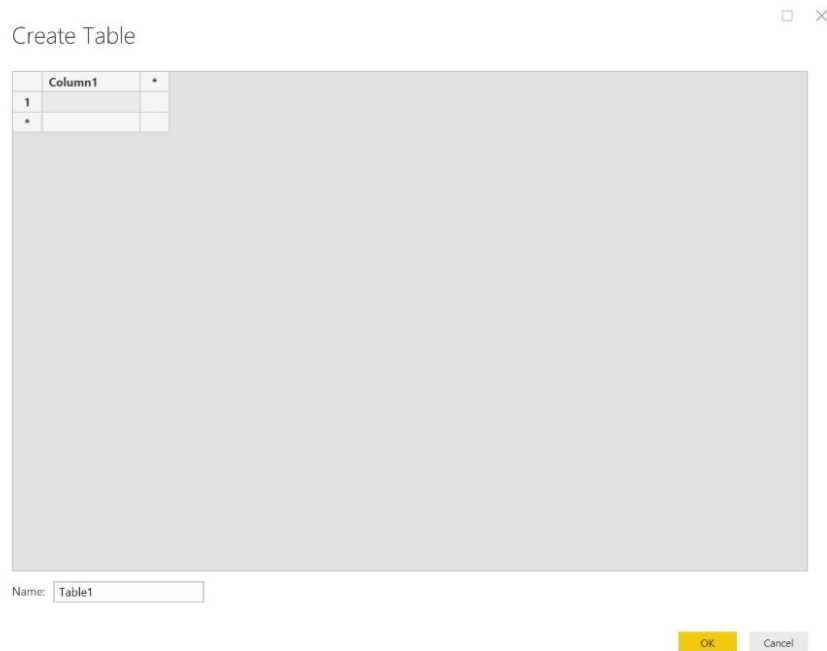


Figure 02-17: Create Table

Copy the URLs above and paste them in the first cell of the column.

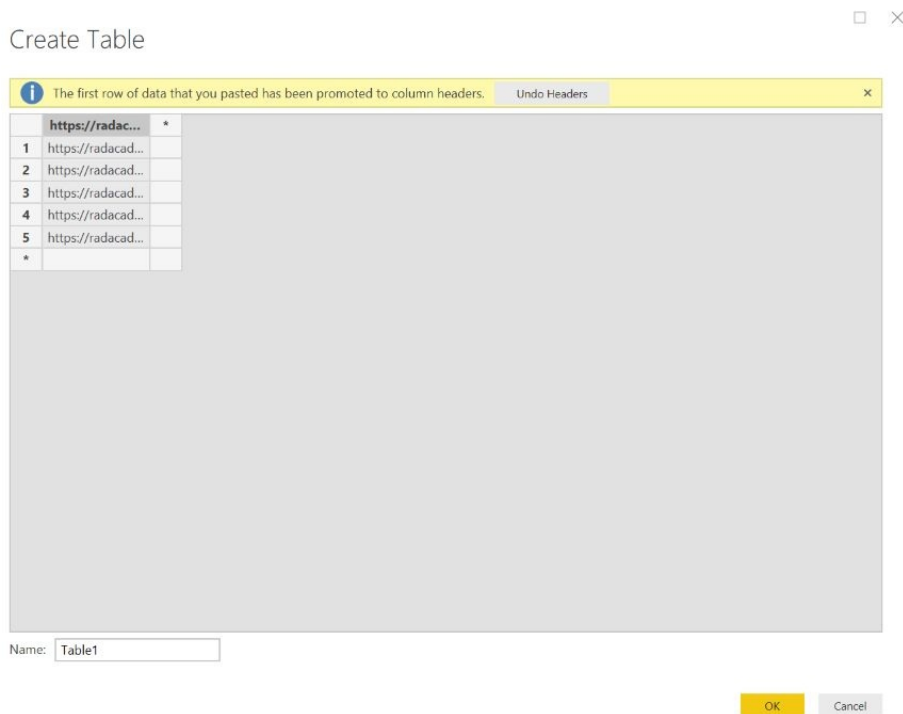


Figure 02-18: Populate Column1 with URLs

The above screen shows that the first row of data has been promoted to header. Click on 'Undo Headers' button. Rename the column to URL as shown below.

Create Table

	URL	*
1	https://radacad.com/ai-builder-power-apps-embed-with-ai-part-1	
2	https://radacad.com/ai-builder-ai-embedded-in-power-apps-part-2	
3	https://radacad.com/power-bi-shared-datasets-what-is-it-how-does-it-work-and-why-should-you-care	
4	https://radacad.com/automated-machine-learning-data-profiling-in-azure-ml-services-part-4	
5	https://radacad.com/azure-machine-learning-services-deploy-automl-model-and-use-it-in-power-bi-part-3	
6	https://radacad.com/budget-vs-actual-zero-complexity-model-in-power-bi	
*		

Name:

Figure 02-19: Rename Column Header

Rename the table as well to something meaningful (let's say URLs). Click OK. Now the table of URLs is ready.

Create a Parameter and a Function to get Data from Multiple Web Pages

1. Create a Parameter

In this stage, we will look at creating a parameter and a function.

To create a parameter, click on ‘Manage Parameters – New Parameter’ from the Home Ribbon:

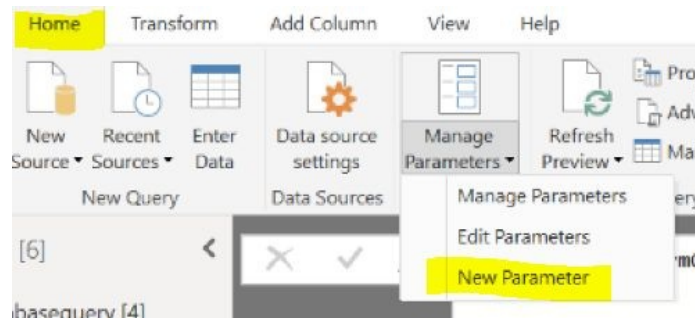


Figure 02-20: ‘Create New Parameter’ button

The screen below appears. Change the name of the new parameter to “myblogurl”, as well as the Type to Text and in the ‘Current Value’ field paste URL <https://radacad.com/ai-builder-ai-embedded-in-power-apps-part-2>.

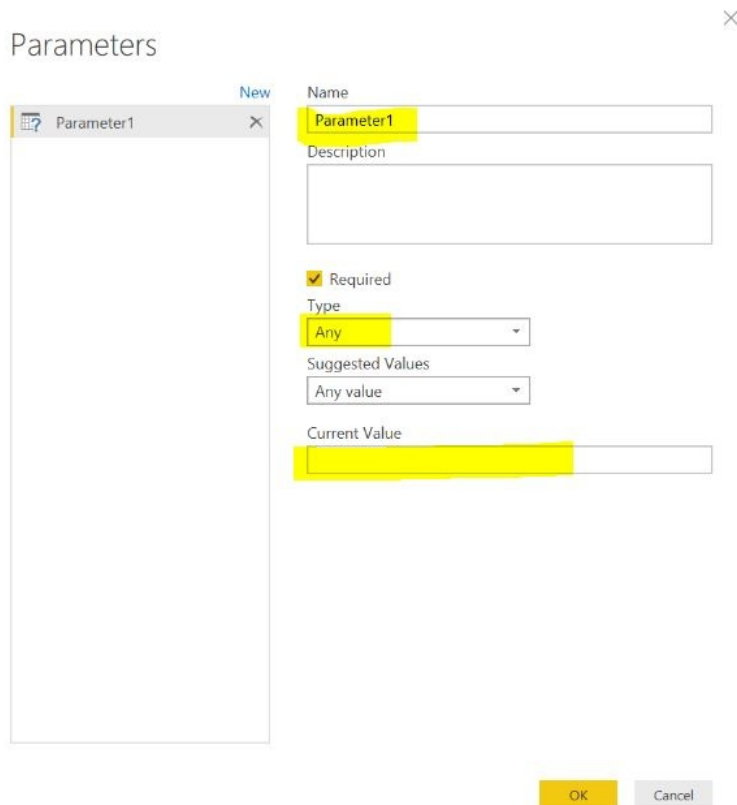


Figure 02-21: Parameters screen

Now the parameter is created. It looks as follows:

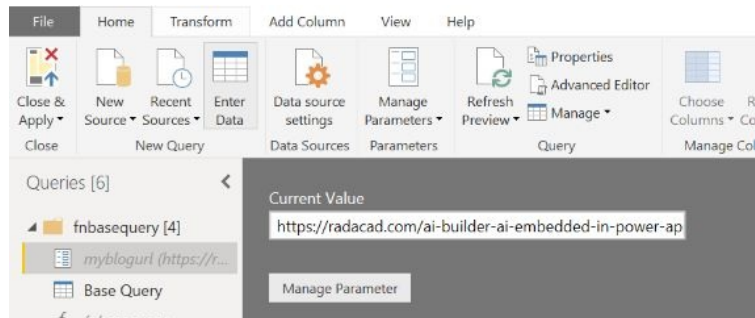


Figure 02-22: myblogurl Parameter created

2. Create a Function

The next task is to create a function. Below are the steps to create a function.

Step 1: In the Power Query Editor, click on the query you have created in the first stage named 'Base Query'. Then, click on 'Advance Editor':

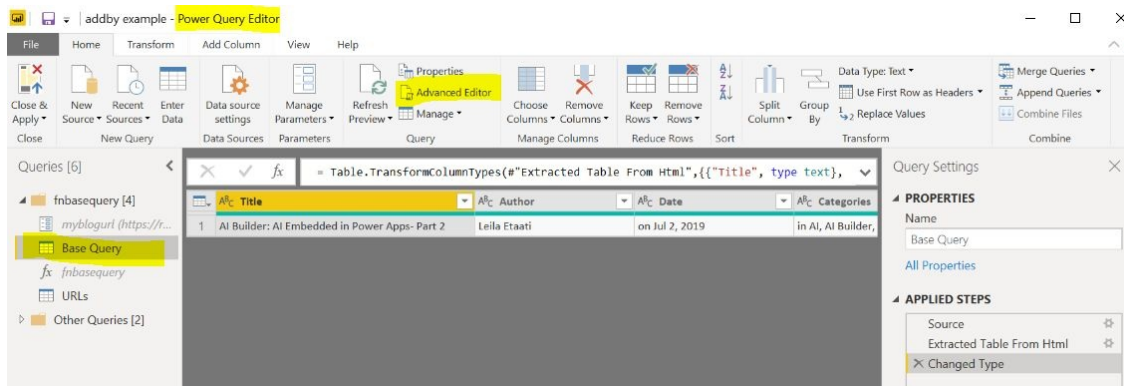


Figure 02-23: Advanced Editor button from Base Query

The below screen pops up. Look at the URL in the highlighted section:



Figure 02-24: Advanced Editor with the actual URL

Step 2: Replace the URL including the quotation marks with the name of the parameter (myblogurl) as shown below. Make sure that there are no syntax errors as highlighted in the bottom left corner of the image shown below.



Figure 02-25: Advanced Editor with the parameter

Step 3: Click OK. Now right click on the 'Base Query' and click on 'Create Function'.

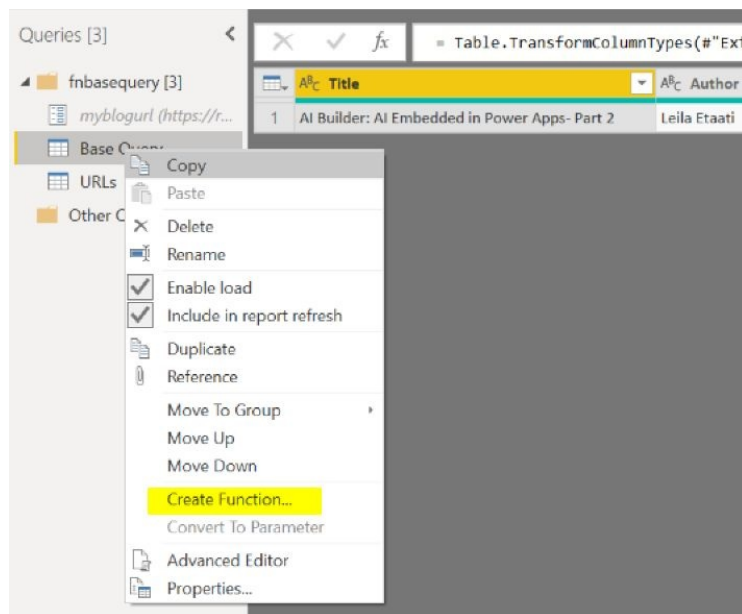


Figure 02-26: Create Function button

Step 4: The screen below pops up. Give a name to the function 'fnmyblogurl'.



Figure 02-27: Create Function screen

Step 5: Click OK. The created function should look as follows:

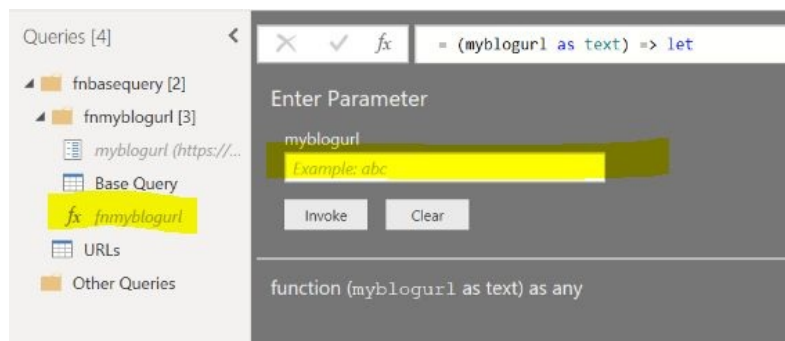


Figure 02-28: Function created

Now you can test the function by copying the URL <https://radacad.com/ai-builder-ai-embedded-in-power-apps-part-2> in the text box highlighted above and clicking Invoke. It should generate the following data:

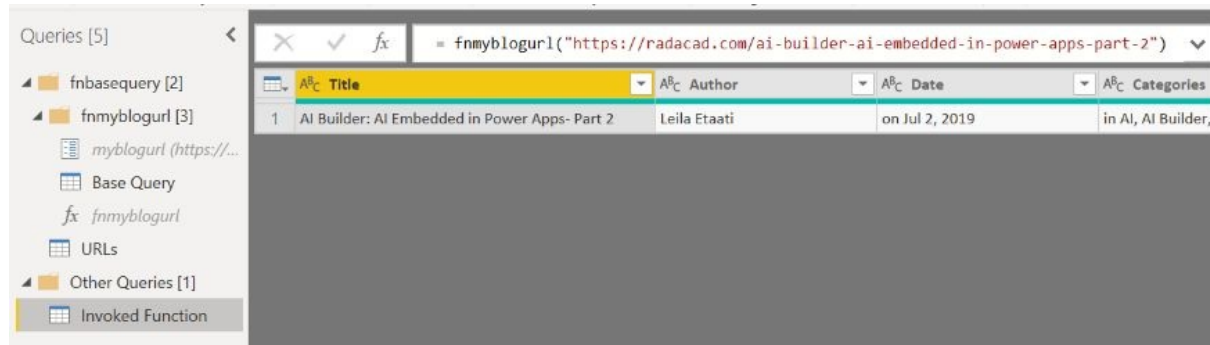


Figure 02-29: Invoked function

Now that your function is created, you can proceed to the next stage of getting the data from multiple URLs.

3. Get Data from Multiple Web Pages

In order get data from the multiple URLs, click on the table that was created, naming it as 'URLs'.

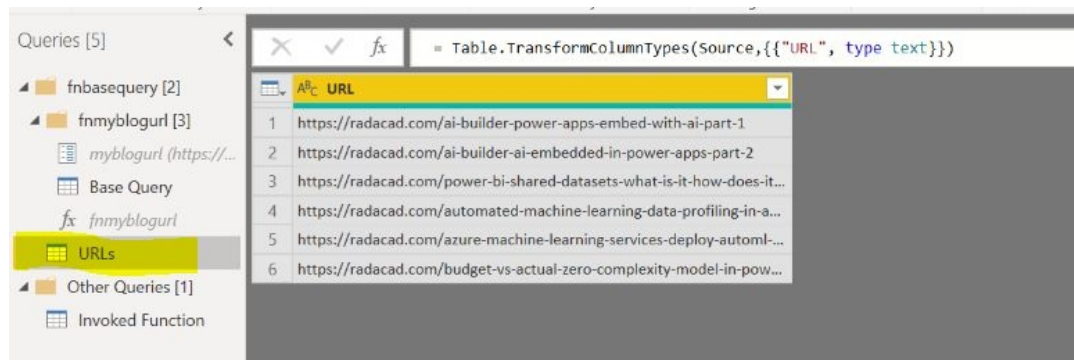


Figure 02-30: URLs table with URL column

Now we will use the created function to create a new column that contains the data for the URLs in the URL column. To do this click on the 'Add Column' tab in the ribbon and click on 'Invoke Custom Function'.

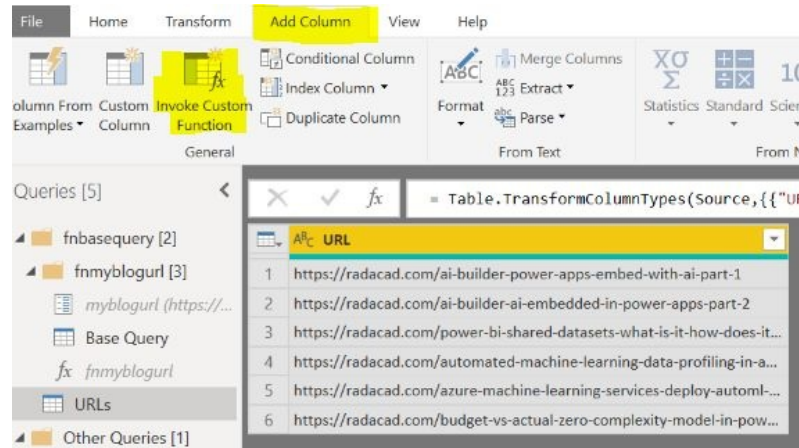


Figure 02-31: Add Column using Invoke Custom Function

The following dialog appears. Change the column name to 'blogdetails'. Choose the function query from the dropdown to fnmyblogurl. In the myblogurl dropdown, choose the field URL from the URLs table. In most cases it automatically takes that field if that is the only field in the table.

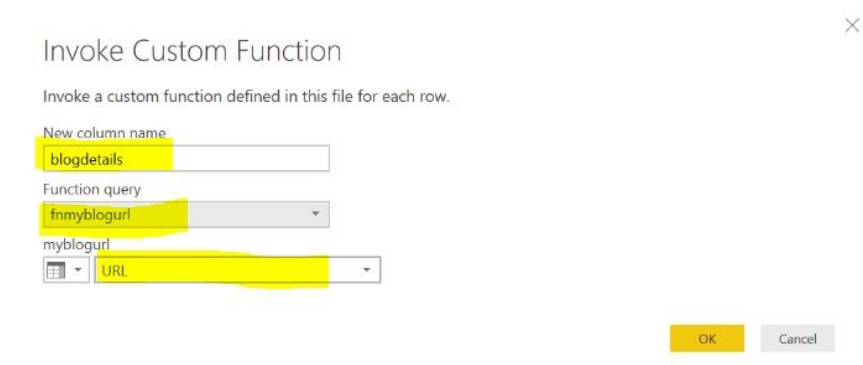


Figure 02-32: Invoke Custom Function dialog

Click OK. Now, there is a new column in the table URLs.

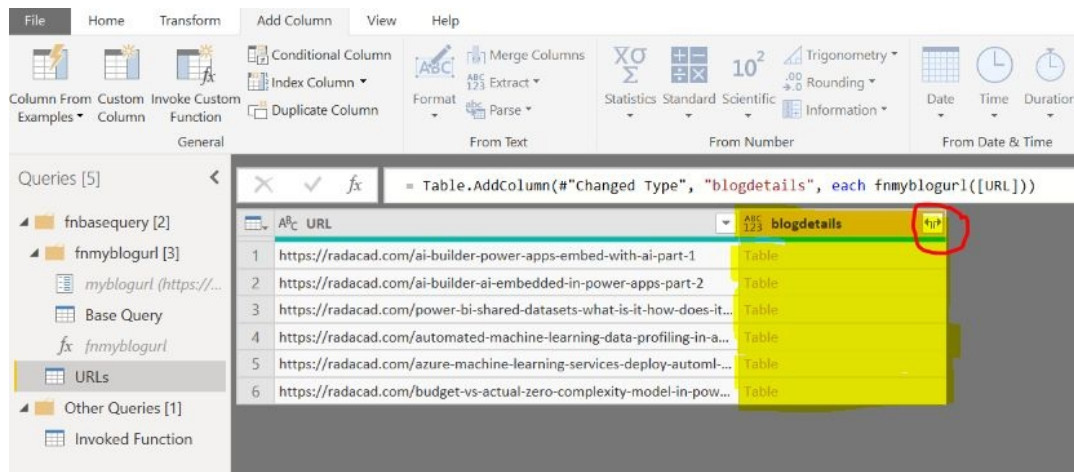


Figure 02-33: New column in URLs table

As you can see there is a Table link. You can expand the blog details column by clicking on the expand button as highlighted in red (above). The following dialog appears.

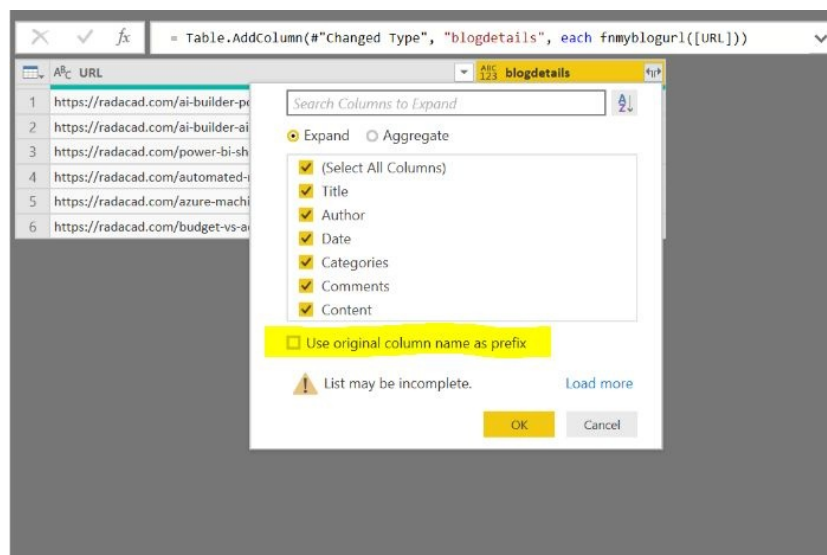


Figure 02-34: Expand button

Make sure that the check box named 'Use original column name as prefix' is unticked. Click OK.

Now the URLs table is populated with all the columns – Title, Date, Author, Categories, Comments and Content for each of the URLs you specified.

URL	Title	Author	Date	Categories	Comments
https://radacad.com/ai-builder-power-apps-embed-with-ai-part-1	AI Builder: AI Embedded in Power Apps- Part 1	Leila Etaati	on Jul 1, 2019	in AI, AI Builder, Analytics, AutoML, Azure ML, Power App	
https://radacad.com/ai-builder-ai-emb...	AI Builder: AI Embedded in Power Apps- Part 2	Leila Etaati	on Jul 2, 2019	in AI, AI Builder, Analytics, AutoML, Azure ML, Power App, PowerApps	No Comments
https://radacad.com/power-bi-sh...	Power BI Shared Datasets: What is it? How does it work? and Why sho...		on Jul 1, 2019	in Architecture, Power BI, Power BI from Rookie to Rockstar	
https://radacad.com/automated-machi...	Automated Machine Learning: Data Profiling in Azure ML Services – Pa...	Leila Etaati	on Jun 24, 2019	in AI, AutoML, Azure Machine Learning, Azure ML, Azure ML workbench	
https://radacad.com/azure-machine-le...	Azure Machine Learning Services : Deploy AutoML Model and Use It in...	Leila Etaati	on May 27, 2019	in AI, Analytics, AutoML, Azure Machine Learning, Azure ML, Azure ML...	
https://radacad.com/budget-vs-actual...	Budget vs Actual: Zero Complexity Model in Power BI		on Jun 18, 2019	in Modelling, Power BI, Power BI from Rookie to Rockstar, Power Query	

Figure 02-35: URLs table with gaps

Did you notice the highlighted parts being blank?

This can also be fixed. The following are the steps to obtain all of the details.

Go back to the Base Query in the Power Query Editor. On the right side in the 'Applied Steps' section, click on the wheel icon beside the 'Extract Table' from the HTML step as highlighted below.

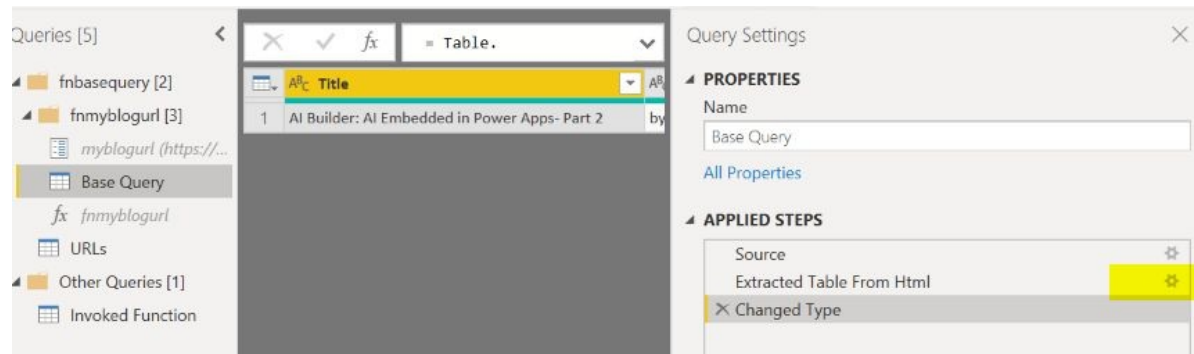


Figure 02-36: Extracted Table from HTML -- settings

The below screen pops up. In the Author field, instead of typing as just 'Leila Etaati', type it as shown below:

	Title	Author	Date	Categories	Comments	Content	*
1	AI Builder: AI E...	Leila Etaati	on Jul 2, 2019	in AI, AI Builder...	No Comments		
*		Leila Etaati					
		Leila Etaatileila@radacad.com					
		leila@radacad.com					
		Leila's latest tweets					
		AI Builder: AI Embedded in Power Apps- Part 2 Poste...s, AutoML, Azure ML, Power App, PowerAp					
		As part of SQL Saturday Auckland 2016 I attended an "Analytics w... into some of the more advan					
		by Leila Etaati					
		Event Details Register Here RADACAD Bootcamp is a 2-day...uage: English Full 2-days Bootcamp					
		Follow Leila on Twitter					
		Instructor: Dr. Leila Etaati					
		Kenny McMillan, Sports Physiologist / Data Analyst, Frankfurt, G...o wants to expand their data ana					

Figure 02-37: Repopulate Author column

Similarly, in the Comments field, instead of just typing 'No Comments', choose 'No Comments' as shown below:

	Title	Author	Date	Categories	Comments	Content	*
1	AI Builder: AI E...	by Leila Etaati	on Jul 2, 2019	in AI, AI Builder...	No c		
2					ABc No Comments		
*					ABc No Comments		
					ABc AI Builder: AI Embedded in Power Apps- P		
					ABc Posted by Leila Etaati on Jul 2, 2019 in AI,		

Figure 02-38: Repopulate Comments column

Click OK.

Now check the URLs table. The details for all the columns are filled in as displayed. In this way, you can just change the 'Base Query' and get the changes as reflected.

Table.ExpandTableColumn(#"Invoked Custom Function", "blogdetails", {"Title", "Author", "Date", "Categories", "Comments", "Content"}, {"Title", "Author", "Date", "Categories",						
URL	Title	Author	Date	Categories	Comments	
1 https://radacad.com/ai-builder-power...	AI Builder: AI Embedded in Power Apps- Part 1	by Leila Etaati	on Jul 1, 2019	in AI, AI Builder, Analytics, AutoML, Azure ML, Power App	No Com	
2 https://radacad.com/ai-builder-ai-emb...	AI Builder: AI Embedded in Power Apps- Part 2	by Leila Etaati	on Jul 2, 2019	in AI, AI Builder, Analytics, AutoML, Azure ML, Power App, PowerApps	No Com	
3 https://radacad.com/power-bi-shared...	Power BI Shared Datasets: What is it? How does it work? and Why sho...	by Reza Rad	on Jul 1, 2019	in Architecture, Power BI, Power BI from Rookie to Rockstar	4 Comm	
4 https://radacad.com/automated-machi...	Automated Machine Learning: Data Profiling in Azure ML Services – Pa...	by Leila Etaati	on Jun 24, 2019	in AI, AutoML, Azure Machine Learning, Azure ML, Azure ML workbenc...	No Com	
5 https://radacad.com/azure-machine-le...	Azure Machine Learning Services : Deploy AutoML Model and Use it in...	by Leila Etaati	on May 27, 2019	in AI, Analytics, AutoML, Azure Machine Learning, Azure ML, Azure ML...	No Com	
6 https://radacad.com/budget-vs-actual...	Budget vs Actual: Zero Complexity Model in Power BI	by Reza Rad	on Jun 18, 2019	in Modelling, Power BI, Power BI from Rookie to Rockstar, Power Query	No Com	

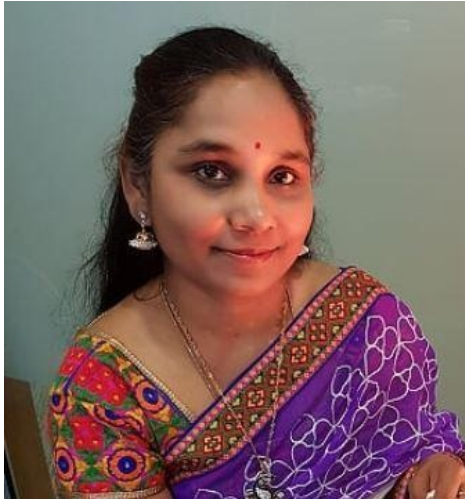
Figure 02-39: URLs Table with Full Details

Summary

Power Query is a pretty powerful tool you can use to extract data from Web as described in this chapter. There are other scenarios you can use to scrape data off the web and analyse in Power BI.

Happy Data Scraping!

About the Author



Indira Bandari Data Platform MVP

Indira is a Business Intelligence Consultant and an aspiring Data Scientist with over 15 years of experience in designing and developing data warehouses and analytical solutions. She has been awarded is a Microsoft Data Platform MVP in New Zealand. She has a Masters degree in Statistics and is passionate about data, analytics and learning data science.

Volunteering and sharing her knowledge are her passions. In her pastime, she teaches game development for primary school children. She also teaches database concepts and data visualizations to 10-15-year-olds.

She is a co-organiser for NZ Power BI User Group, Auckland AI Meetup Group and SQL Saturday Auckland. She is a speaker at various User Groups, Virtual Webinars, PASS Marathon, 24 Hours of PASS and SQL Saturdays in New Zealand and the Power Platform Summit in Australia.

Chapter 3: One URL, Many Tables

Author: Liam Bastick

Chapter abstract: This chapter considers how to import a multiple page table into Power Query where the URL does not appear to change. The approach was a true team effort – and although not pretty, it's a very useful technique!

Here's a rather awkward – yet common – problem. Consider the data from the following website <http://www.quanthockey.com/khl/seasons/2017-18-khl-players-stats.html>:



Rk	Name	Age	Pos	GP	G	A	P	PIM	+/-	PPG	SHG	GWG	G/GP	A/GP	P/GP
1	 Nigel Dawes	32	F	19	20	5	25	6	4	9	0	5	1.053	0.263	1.316
2	 Ilya Kovalchuk	34	F	24	17	11	28	12	2	10	0	5	0.708	0.458	1.167
3	 Nikita Gusev	25	F	24	13	19	32	2	12	3	0	2	0.542	0.792	1.333
4	 Sergei Mozyakin	36	F	23	12	14	26	2	-1	6	0	2	0.522	0.609	1.130
5	 Justin Azevedo	29	F	19	12	6	18	10	3	4	0	3	0.632	0.316	0.947
6	 Sergei Shirokov	31	F	24	11	14	25	12	17	4	1	1	0.458	0.583	1.042
7	 Vladimir Tkachyov	24	F	20	11	6	17	16	4	2	0	2	0.550	0.300	0.850
8	 Eeli Tolvanen	18	F	19	11	10	21	8	7	4	0	3	0.579	0.526	1.105
9	 Kirill Kaprizov	20	F	18	11	10	21	0	11	2	0	4	0.611	0.556	1.167
10	 Quinton Howden	25	F	23	10	6	16	16	-1	6	0	0	0.435	0.261	0.696

Figure 03-01: Example Data

The webpage is nicely set out and contains a table of hockey player statistics. The thing is, the embedded table actually has 17 pages of data and let's say we wish to extract all of this data for analysis elsewhere.

There's a problem though. When you click on the second or subsequent page of data, the URL for the website does not change. This seemingly defeats Power Query (or Power BI) as URLs for each page of table data are required.

So how may we extract all of the data? To answer this, let's get there in five steps.

Part 1: Manual Retrieval of Data

Now I know most of this book is on Power BI, but I like to present general solutions where possible. Power Query – in its guise as ‘Get & Transform’ – is available in both Excel and Power BI. Therefore, to show its versatility (and to be different!), please allow me to demonstrate this using Excel. Power BI works just as well – and very similarly too.

To manually import the data from this example website using Power Query, first open Excel, navigate to the ‘Data’ tab and click on the ‘New Query’ option, select the ‘Other Sources’ option followed by ‘Web’, viz.

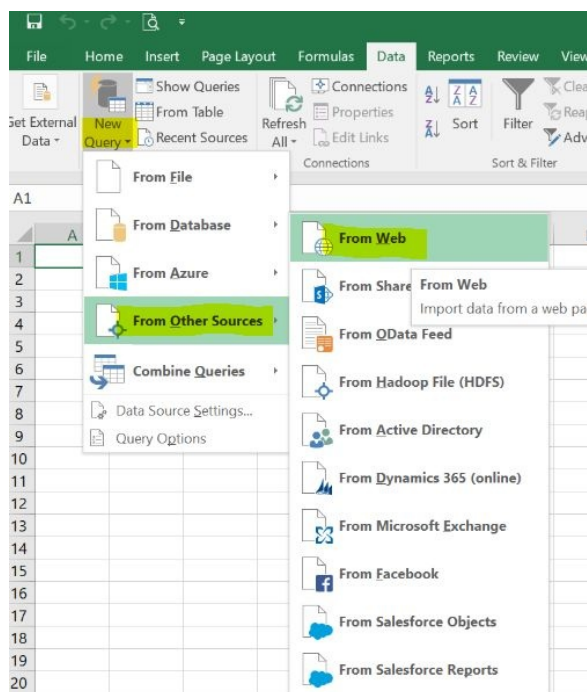


Figure 03-02: New Query

A dialog box will appear, allowing us to insert the URL. Next, click ‘OK’:



Figure 03-03: From Web Dialog

The 'Navigator' dialog box will appear, allowing us to select exactly which table to pull data from. At this point all looks good, however we should name the table, so click on 'Edit':

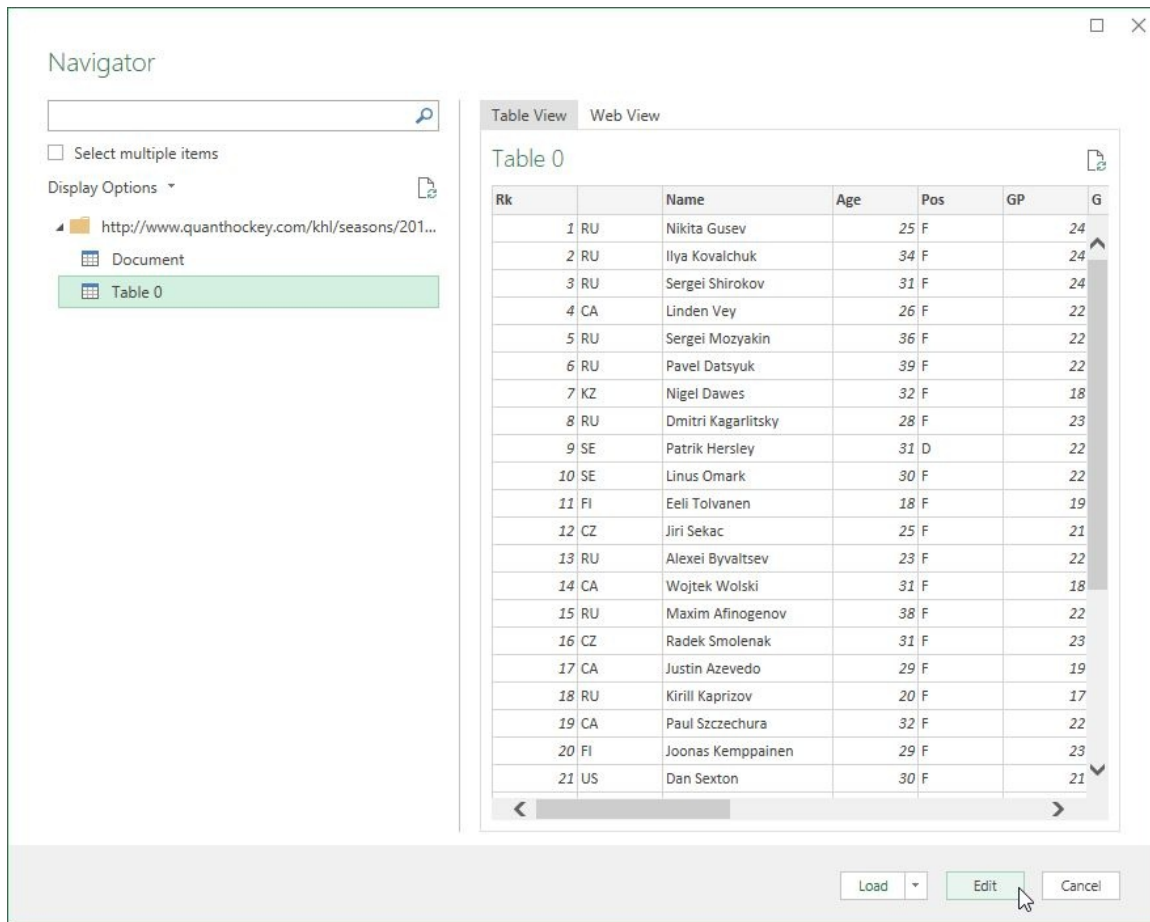


Figure 03-04: Navigator Dialog

In the 'Query Editor' dialog, we should give our query a friendly name, let's say 'HockeyData', then select 'Close & Load':

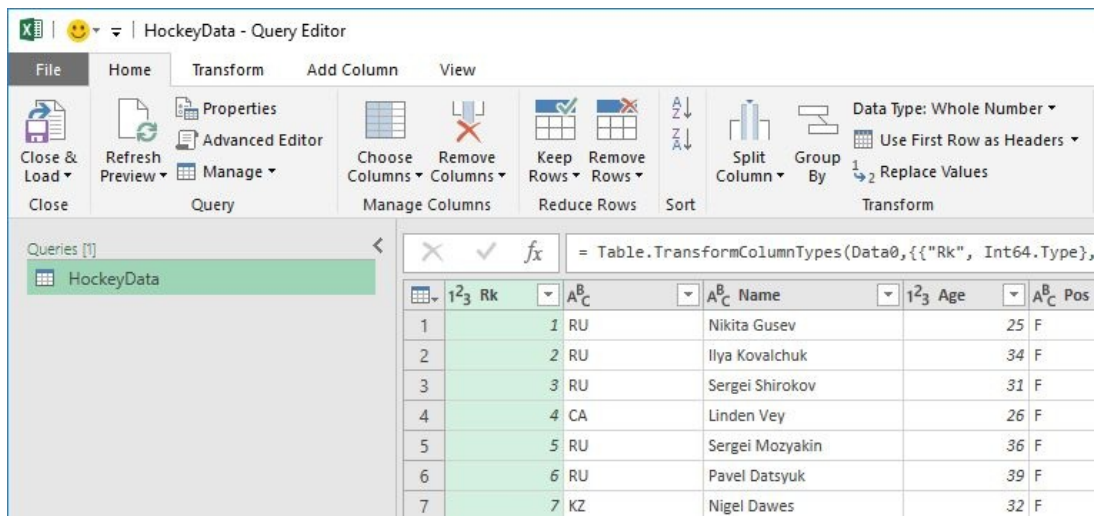


Figure 03-05: HockeyData

We can see that Power Query was only able to retrieve the first 50 entries:

S58																	
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
43	42	RU	Ilya Mikheyev	23	F	20	6	9	15	4	8	1	0	0	0.3	0.45	0.75
44	43	US	Casey Wellman	30	F	22	5	10	15	2	1	0	1	0	0.227	0.455	0.682
45	44	UA	Pavel Padakin	23	F	20	5	10	15	12	11	0	0	1	0.25	0.5	0.75
46	45	RU	Alexander Khokhlachev	24	F	18	5	10	15	6	4	2	0	2	0.278	0.556	0.833
47	46	RU	Pavel Chernov	27	F	22	6	8	14	24	7	1	0	0	0.273	0.364	0.636
48	47	RU	Dmitri Kugryshev	27	F	22	6	8	14	2	1	3	0	1	0.273	0.364	0.636
49	48	CA	Gilbert Brule	30	F	21	6	8	14	24	-1	3	0	1	0.286	0.381	0.667
50	49	CA	Eric O'Dell	27	F	18	6	8	14	12	12	1	0	0	0.333	0.444	0.778
51	50	CZ	Andrej Nestrasil	26	F	22	4	10	14	2	-6	0	0	2	0.182	0.455	0.636

Figure 03-06: First 50 Entries Only

This is because Power Query retrieves data based on the URL, and in this case our Power Query friendly hockey statistics website displays data using JavaScript to dynamically refresh the list of players. This enables the webpage to dynamically refresh the player list in one page, without changing the webpages' URL. Additionally, there's another problem: we also do not know how many pages of data this website has.

Therefore, to summarize, we have three key issues:

1. We are unable to manually pull all data from the website
2. We do not know how many pages of data the website has (and this may change over time)
3. The webpage does not change its URL when a new page of data is displayed.

Let's deal with them systematically.

Part 2: Custom Functions

Turning to the first issue identified, it's been noted that we are unable to manually retrieve all of the data just by importing it into Power Query. Several of us worked together collaboratively and the solution proposed and provided here was by Reza Rad utilising custom functions in Power Query.

A custom function is a query that is run by other queries. For those of you who know JavaScript, it is similar to what is known as an Object Method. The benefit of having a custom function is that we can repeat the same number of steps again and again.

Let's work through a simple example to illustrate a custom function's utility. For instance, we wish to retrieve the gross earnings of all of the movies that were released in that year, along with their current rank and their studio (referring to the website <http://www.boxofficemojo.com/yearly/chart/?yr=2017&p=.htm>). It does not matter which year we wish to begin with, so for this example we shall begin with 2017.

To launch Power Query / Get & Transform, launch Excel and head to the 'Data' tab and select 'New Query' -- 'Other Sources' -- 'Web' again. Using the default options, paste in the URL and click 'OK'.



Figure 03-07: From Web Dialog (again)

In the ensuing dialog, select 'Table 1' (as this is the data) and then click on 'Edit':

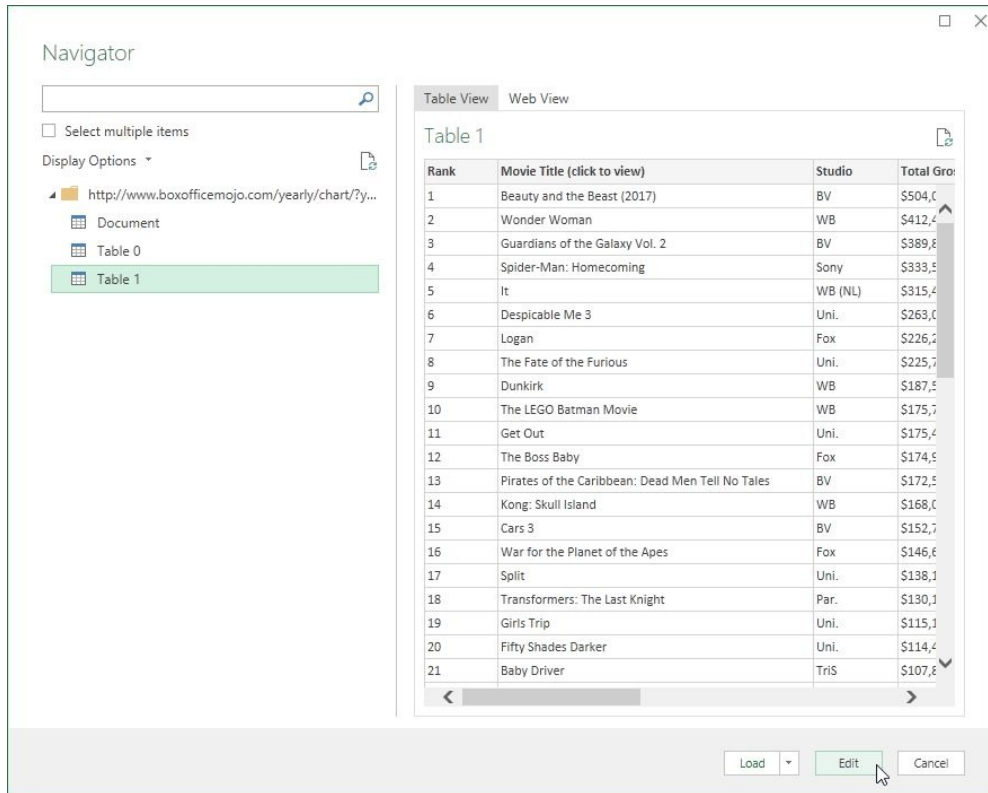


Figure 03-08: Editing Table 1

Now that we have the 'Query Editor' window open, we can define our parameter. Parameters are needed for custom functions to work.

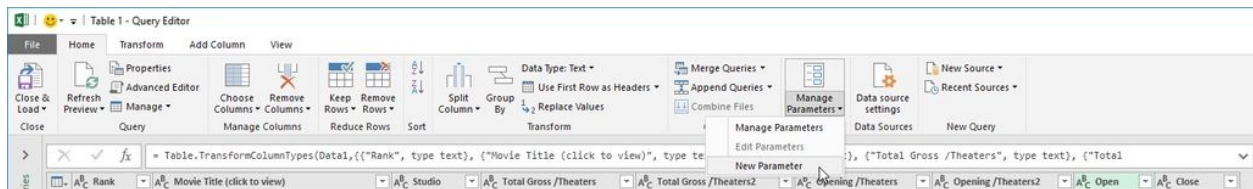


Figure 03-09: Creating a New Parameter

We create a simple parameter, set the name to 'Year' type to 'text' and the initial value to 2017:

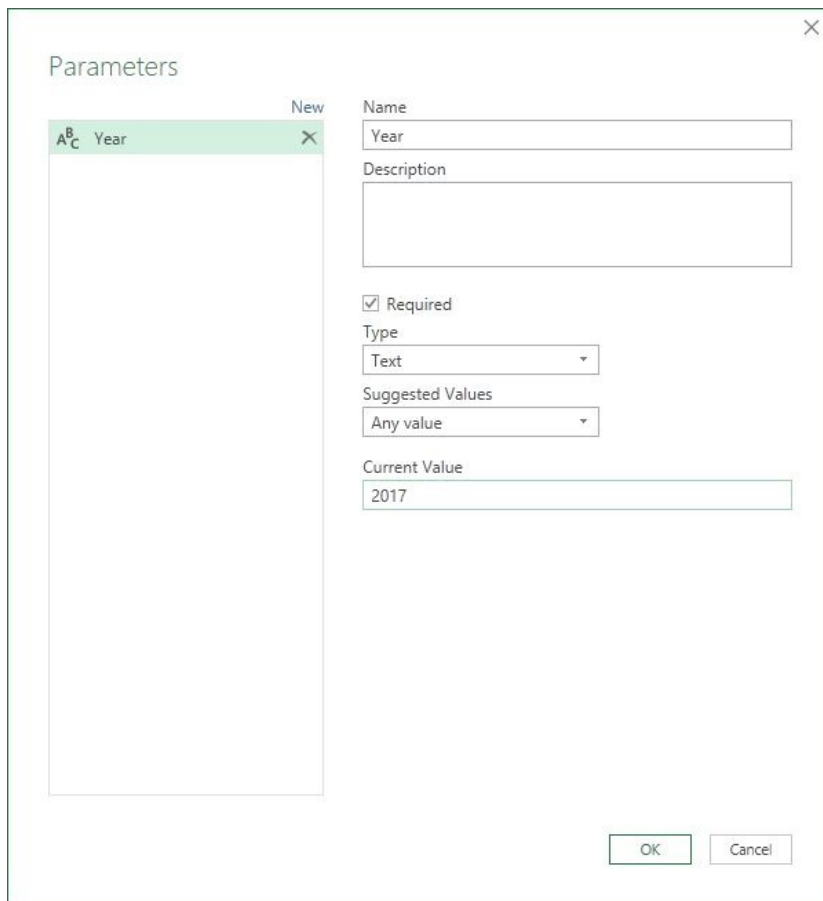


Figure 03-10: Parameters Dialog

We can now add a custom column. Click on 'Table 1', then on the 'Add Column' tab and then 'Custom Column':

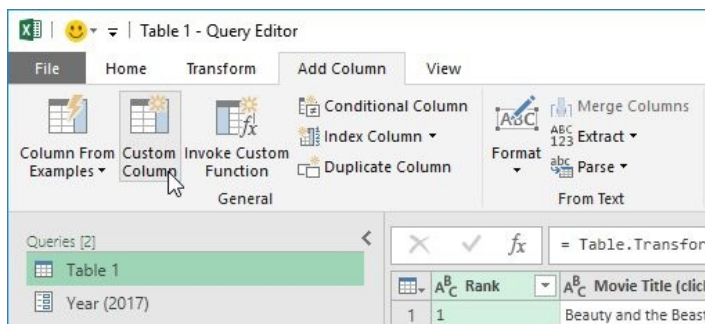


Figure 03-11: Custom Column

We give the custom column a name 'Year' and make it equal to the parameter 'Year'.

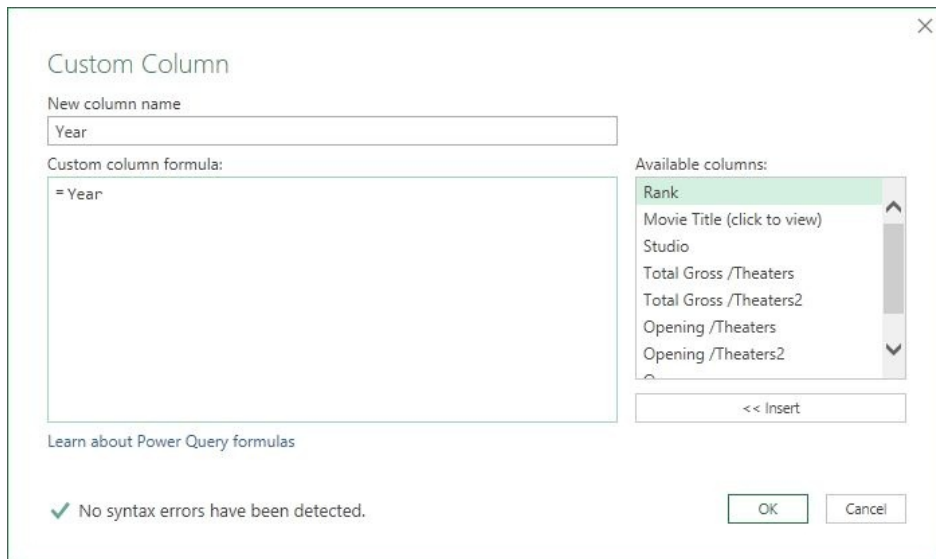


Figure 03-12: Creating the Parameter ‘Year’

Be sure to change the custom column’s data type to ‘Text’ too:

A ^B _C Open	A ^B _C Close	A ^B _C Year
3/17	7/13	2017
6/2	-	2017
5/5	9/21	2017
7/7	-	2017
8/8	-	2017

Figure 03-13: Making the ‘Year’ Data Type Text

The next step is to integrate our parameter into the URL. This allows us to dynamically change the URL, ultimately altering the source of the database on the desired year. Therefore, with ‘Table 1’ selected, click on the setting icon for the ‘Source’ step in the ‘Applied Steps’ section:

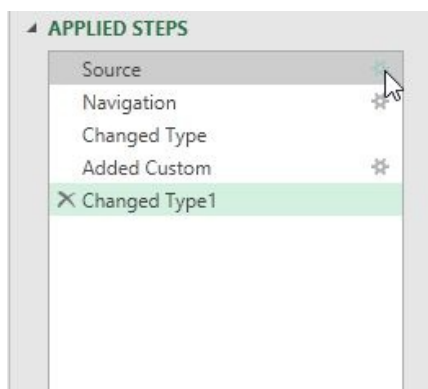


Figure 03-14: Applied Steps

Now let’s select the ‘Advanced’ option. Identify the part of the URL that has the

date, and enter the parameter in its place. We should also include the last element of the URL after the ‘Year’ parameter. We do this by adding to the URL with some copy and paste work.

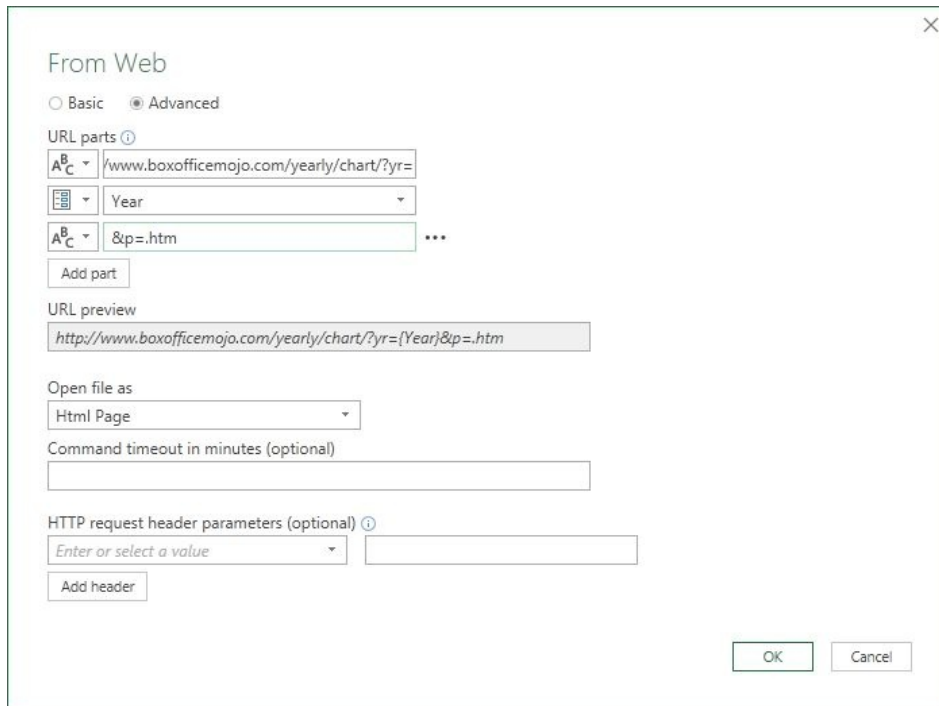


Figure 03-15: Modifying the URL

Once that is done click ‘OK’. We now have to convert the query into a function. Right click on the ‘Table 1’ query then select ‘Create Function...’:

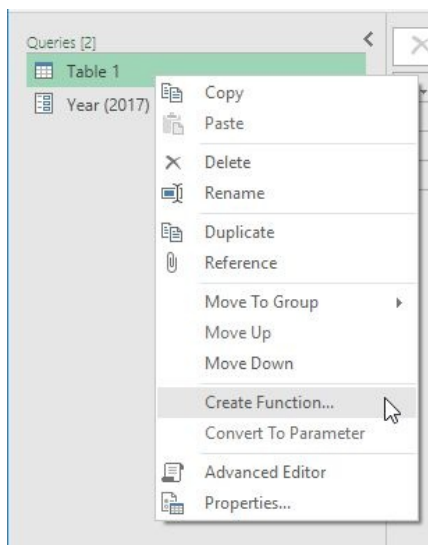


Figure 03-16: Creating a Function

Name the function ‘GetMovies’ then click ‘OK’.

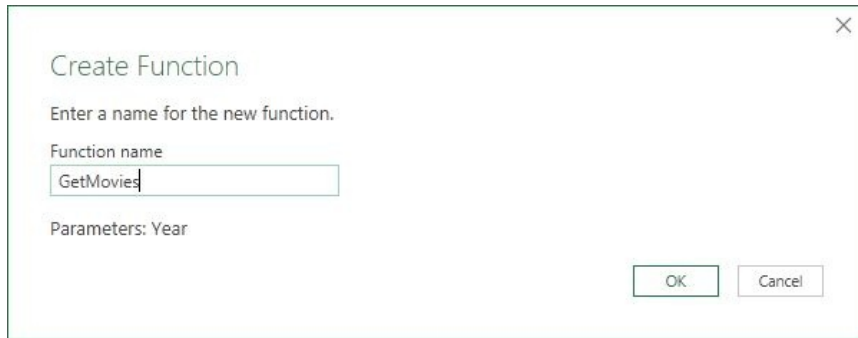


Figure 03-17: Naming the Function

There is now a group folder containing the original 'Table 1' query, the Year 2017 parameter, and the 'GetMovies' function.

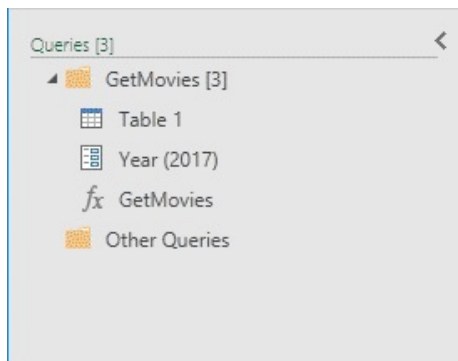


Figure 03-18: Noting the Three Queries

We have created a copy of the Table 1 query and called it 'GetMovies'. From now on, every time we call on 'GetMovies', Power Query will perform the same tasks in that defined order.

For simplicity, we will create a simple generator. We will use the **List.Numbers** function to create our generator. To do this, simply create a new query by navigating to the 'Data' tab, 'New Query', 'From Other Sources' and choose 'Blank Query'. Then enter the following formula in the formula bar:

=List.Numbers(2002,16)

and hit **ENTER**.

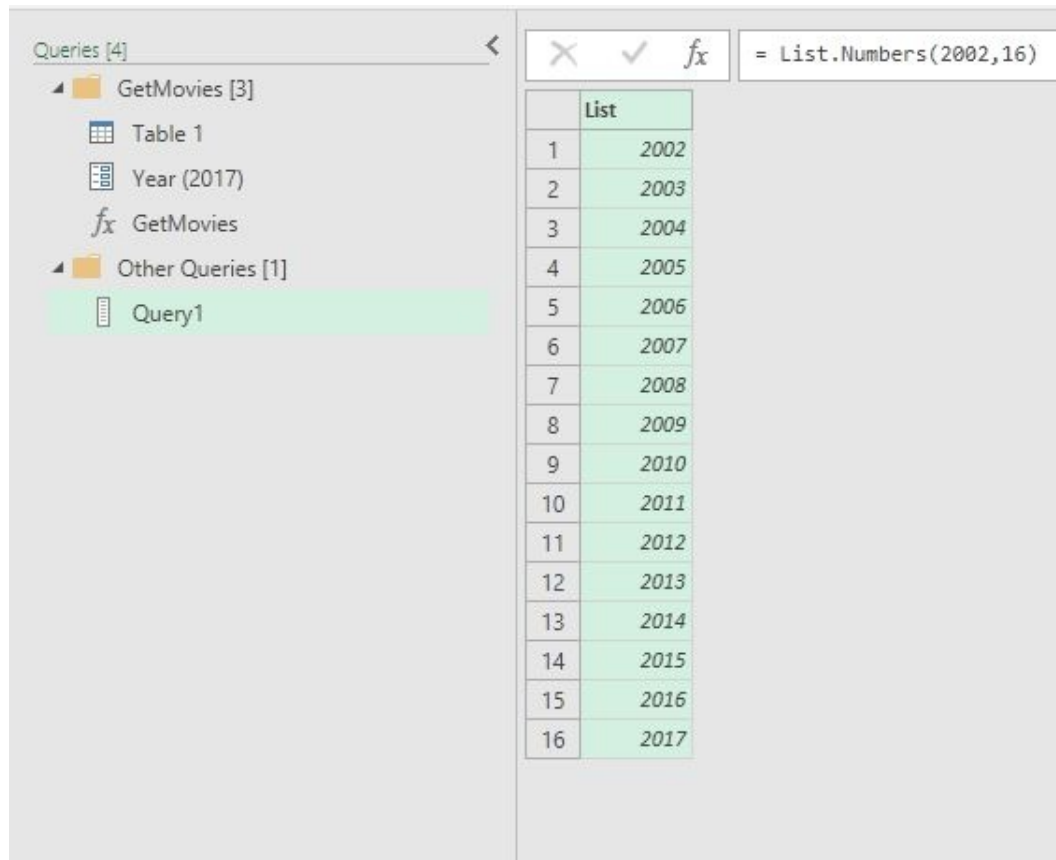


Figure 03-19: List.Numbers

This creates a list, which only has limited flexibility and use. Tables are much more flexible. We can convert the list into a table using the 'To Table' option located in the 'Convert' group as shown:

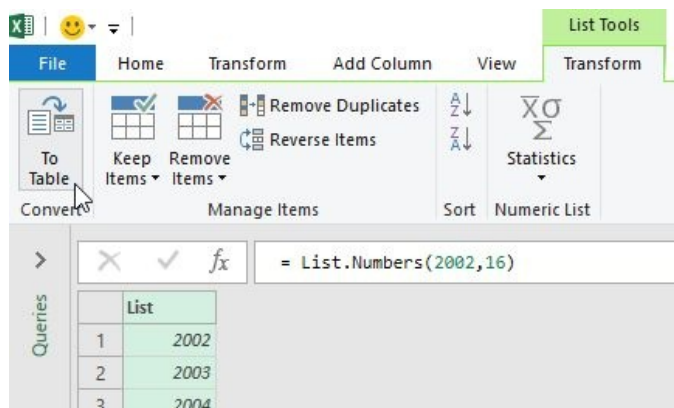


Figure 03-20: Convert to Table

The default conversion settings will suffice. Lastly, change the data type to 'Text'.

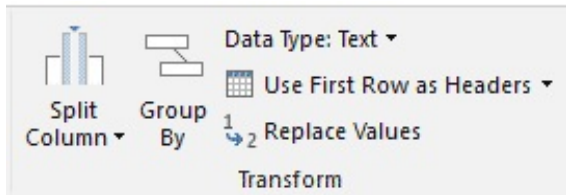


Figure 03-21: Data Type Text

With the 'Query1' query selected, invoke a custom function by going to the 'Add Column' tab and select the 'Invoke Custom Function' in the 'General' group:

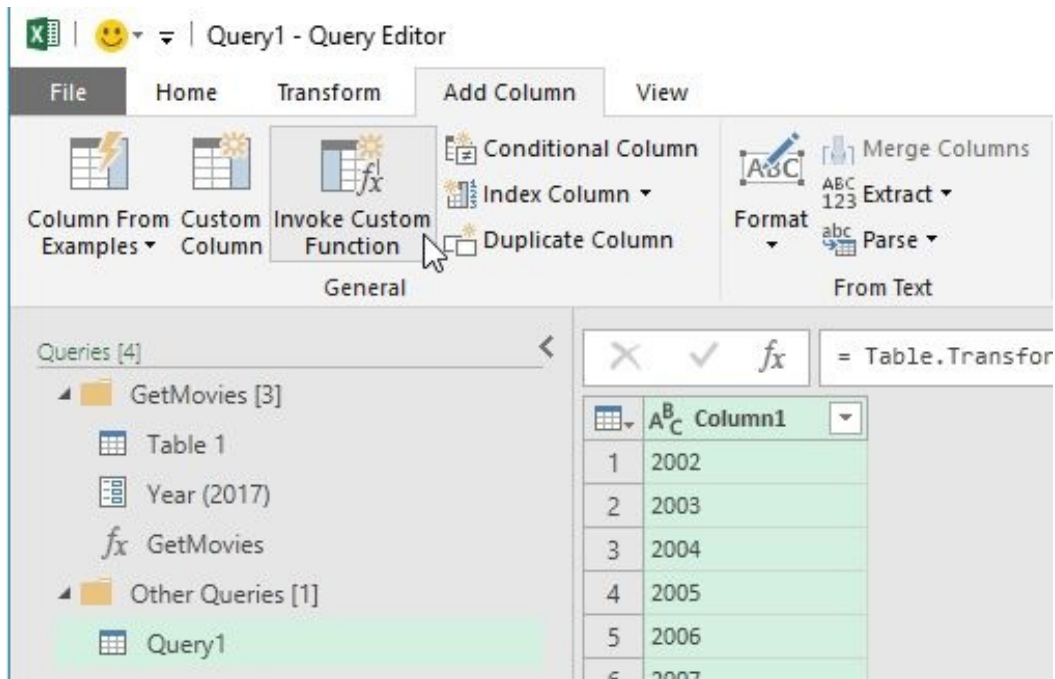


Figure 03-22: Invoke Custom Function

After naming the new column to 'GetMovieData', select the 'GetMovies' function and click 'OK':

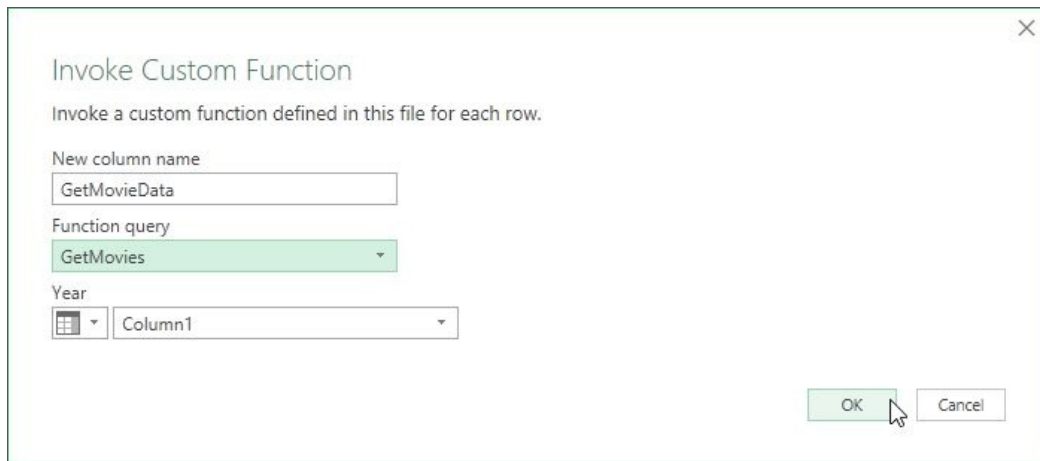


Figure 03-23: Invoke Custom Function Dialog

A new column will be added:

	Column1	GetMovieData
1	2002	Table
2	2003	Table
3	2004	Table
4	2005	Table
5	2006	Table
6	2007	Table
7	2008	Table
8	2009	Table
9	2010	Table
10	2011	Table
11	2012	Table
12	2013	Table
13	2014	Table
14	2015	Table
15	2016	Table
16	2017	Table

Figure 03-24: Column Added

Clicking on each individual Table line item will drill into the movie data for its corresponding year (or you may click on the white space to the right of 'Table' to show it in preview), *e.g.*

= #Invoked Custom Function"(4)[GetMovieData]							
A ^B _C Rank	A ^B _C Movie Title (click to view)	A ^B _C Studio	A ^B _C Total Gross /Theaters	A ^B _C Total Gross /Theaters2	A ^B _C Opening /Theaters	A ^B _C Opening /Theaters2	
1	Pirates of the Caribbean: Dead Man's Chest	BV	\$423,315,812	4,133	\$135,634,554	4,133	
2	Night at the Museum	Fox	\$250,863,268	3,768	\$30,433,781	3,685	
3	Cars	BV	\$244,082,982	3,988	\$60,119,509	3,985	
4	X-Men: The Last Stand	Fox	\$234,362,462	3,714	\$102,750,665	3,690	
5	The Da Vinci Code	Sony	\$217,536,138	3,757	\$77,073,388	3,735	

Figure 03-25: After Clicking on ‘Table’ in 2006

There are some limitations however:

- Editing the **M** script of the function will cause the function and query to collapse
- Custom functions cannot be scheduled to update in Power BI.

This shouldn't detract from the benefits. Moving on, click on the ‘Expand’ option as shown:

A ^B _C Column1	ABC 123 GetMovieData
1 2002	Table
2 2003	Table
3 2004	Table
4 2005	Table
5 2006	Table

Figure 03-26: ‘Expand’ Option

This reveals a compiled table with the top 100 movies for the given year:

= Table.ExpandTableColumn("#Invoked Custom Function", "GetMovieData", ("Rank", "Movie Title (click to view)", "Studio", "Total Gross /Theaters", "Total Gross /Theaters2", "Opening /Theaters", "Opening /Theaters2"))					
A ^B _C Column1	ABC 123 GetMovieData.Rank	ABC 123 GetMovieData.Movie Title (click to view)	ABC 123 GetMovieData.Studio	ABC 123 GetMovieData.Total Gross /Theaters	ABC 123 GetMovieData.Total Gross /Theaters2
87 2002	87	The Badger Sisters	Fox	\$30,361,410	2,944
88 2002	88	Bad Company	BV	\$30,160,161	2,944
89 2002	89	Ghost Ship	WB	\$30,113,491	2,787
90 2002	90	The New Guy	SonR	\$29,760,152	2,687
91 2002	91	SwimFan	Fox	\$28,564,995	2,860
92 2002	92	The Crocodile Hunter: Collision Course	MGM	\$28,442,574	2,535
93 2002	93	Brown Sugar	FoxS	\$27,563,891	1,378
94 2002	94	Blood Work	WB	\$26,235,081	2,525
95 2002	95	All About the Benjamins	NL	\$25,916,319	1,519
96 2002	96	Frida	Mira	\$25,885,000	794
97 2002	97	Jonah: A VeggieTales Movie	Art	\$25,581,229	1,625
98 2002	98	Beauty and the Beast (IMAX)	BV	\$25,487,190	68
99 2002	99	The Transporter	Fox	\$25,296,447	2,610
100 2002	100	The Sweetest Thing	Sony	\$24,718,164	2,670
101 2002	Summary of 480 Movies on Chart:				
102 2002	Totals:	\$9,206,344,777			
103 2002	Averages:	\$19,179,885			
104 2003	1	The Lord of the Rings: The Return of the King	NL	\$377,027,325	3,703
105 2003	2	Finding Nemo	BV	\$339,714,978	3,425
106 2003	3	Pirates of the Caribbean: The Curse of the Black Pearl	BV	\$305,413,918	3,416
107 2003	4	The Matrix Reloaded	WB	\$281,576,461	3,603
108 2003	5	Bruce Almighty	Uni	\$242,829,261	3,549
109 2003	6	X2: X-Men United	Fox	\$214,949,694	3,749
110 2003	7	Elf	NL	\$173,398,518	3,381

Figure 03-27: Expanded Table

The data still needs some cleaning up, but that's another story. This deals with manual importation of the data. However, what about the page number issue?

Part 3: Unknown Number of Pages

Again, I am not looking to take credit for the methodology here – I am just the one that put it all together. The solution to this part of the problem was produced by a combined effort of ideas from Matt Mason and Miguel Escobar.

Matt Mason's method adopts a brute force method to dealing with an unknown number of pages where it instructs Power Query to run through pages 1 to a given number (say, 10,000) but to stop when Power Query runs into an error or a 'null' value. He points out that if this method is used together with a third-party software such as Fiddler (more on Fiddler later), Power Query will be found trying to evaluate all 10,000 pages. Furthermore, if you try Matt's method now with newer versions of Power Query, you will receive an error claiming that you do not have access to the database. We need to modify this method!

This is where Miguel comes in and adjusts the code a little so that it does not adopt the brute force method anymore as well as fix this permissions bug in Power Query.

Building up from Matt Mason's model, we will only utilise his 'GetData' function. Open Power Query from Excel and we'll turn Matt's 'GetData' idea into a function:

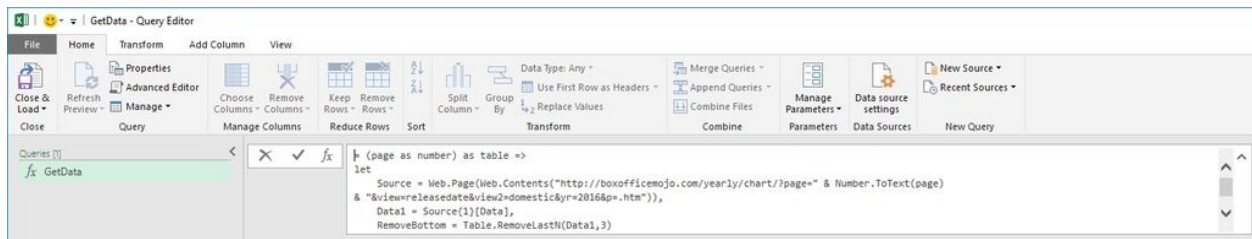


Figure 03-28: GetData Query

If you can't read it, don't worry! Let's go through the code line by line.

Now we create a whole new Query, using a 'Blank Query'. The first line of code to be entered is the **List.Generate** function:

=List.Generate()=>

The **()=>** function nomenclature essentially says that we will define a function with no parameter.

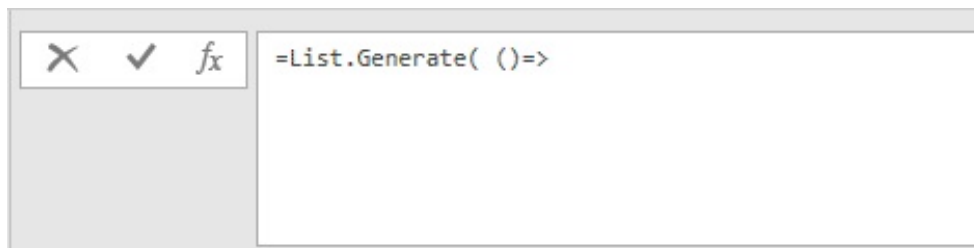


Figure 03-29: Building Up the Code #1

The next line is:

[Result= try GetData(1) otherwise null, Page = 1],

This line says try to **GetData**, but if it returns an error, return 'null' in Page 1:

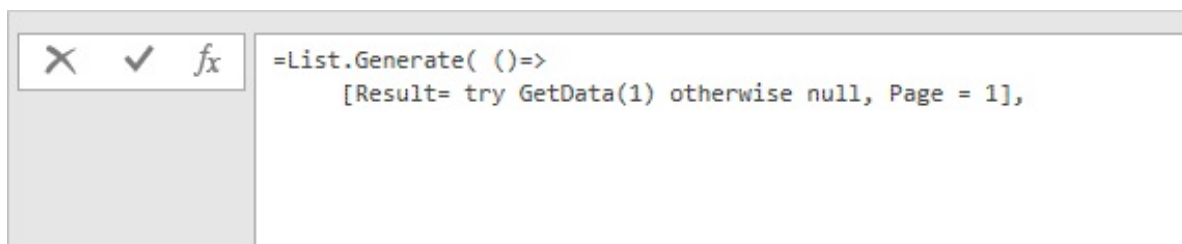


Figure 03-30: Building Up the Code #2

The next line:

each [Result] <> null,

specifies a condition, where the result cannot be null or perform this function as long as the **Result** is not equal to null.

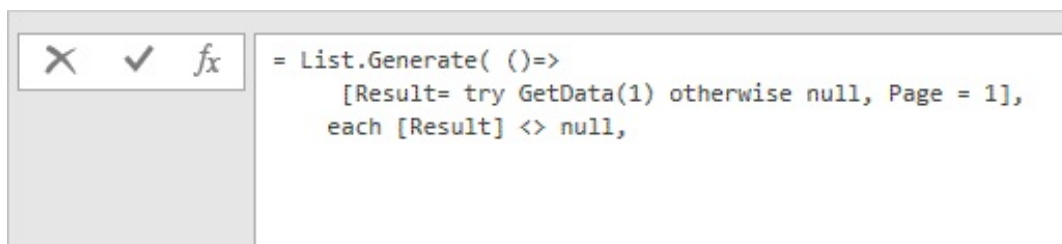


Figure 03-31: Building Up the Code #3

The next line increments the page to page 2:

each [Result = try GetData([Page]+1) otherwise null, Page = [Page]+1],

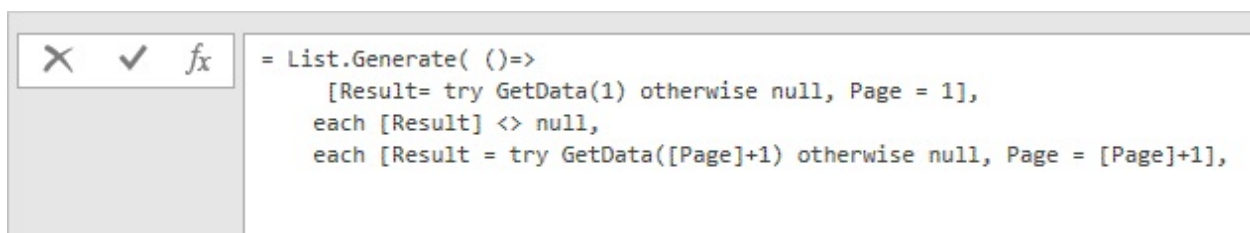


Figure 03-32: Building Up the Code #4

The last line in this function instructs Power Query to display the **Result** field: **each [Result]**)

Once we hit **ENTER**, we will see the list of tables:

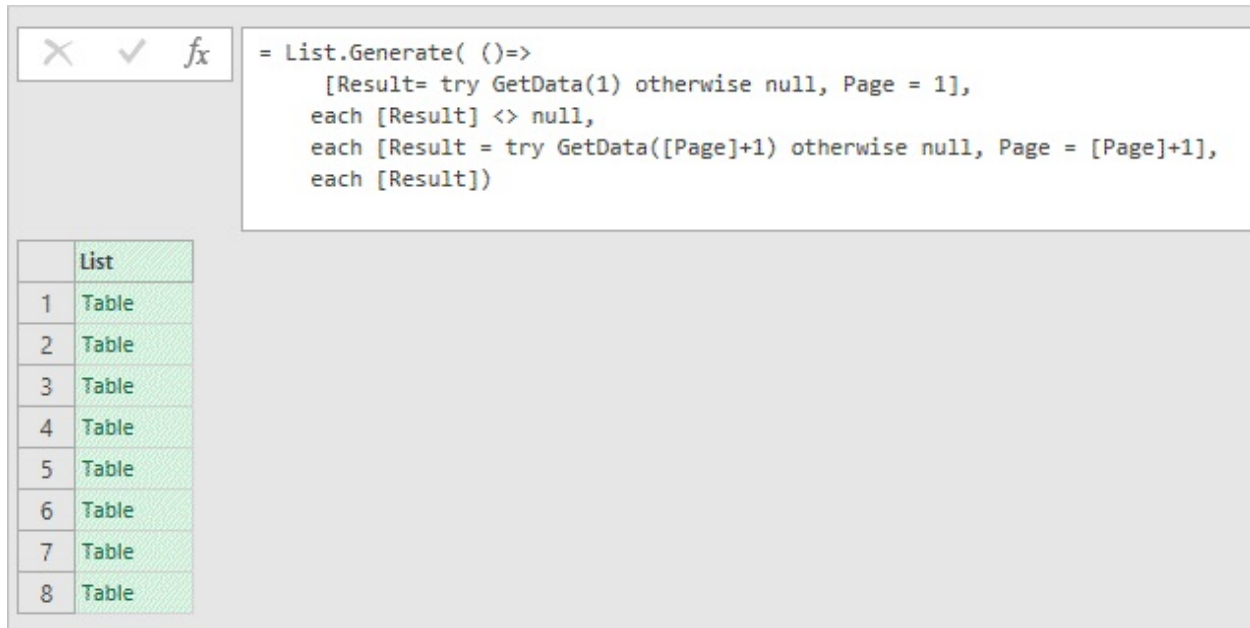


Figure 03-33: Building Up the Code #5

This is all of the different pages pulled of the domestic gross of 2016 from the Box Office Mojo website. Notice that Power Query does not try to evaluate 10,000 pages!

Now we go through the table and define each column's data type. While this is still a list, we can transform this into a table and expand the data:

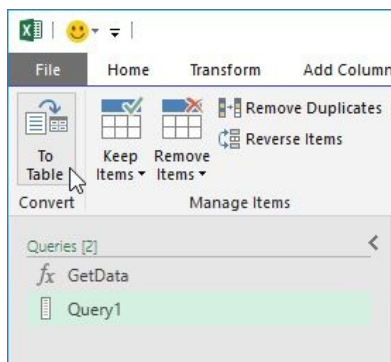
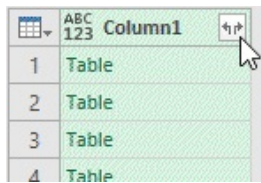


Figure 03-34: Transform to Table

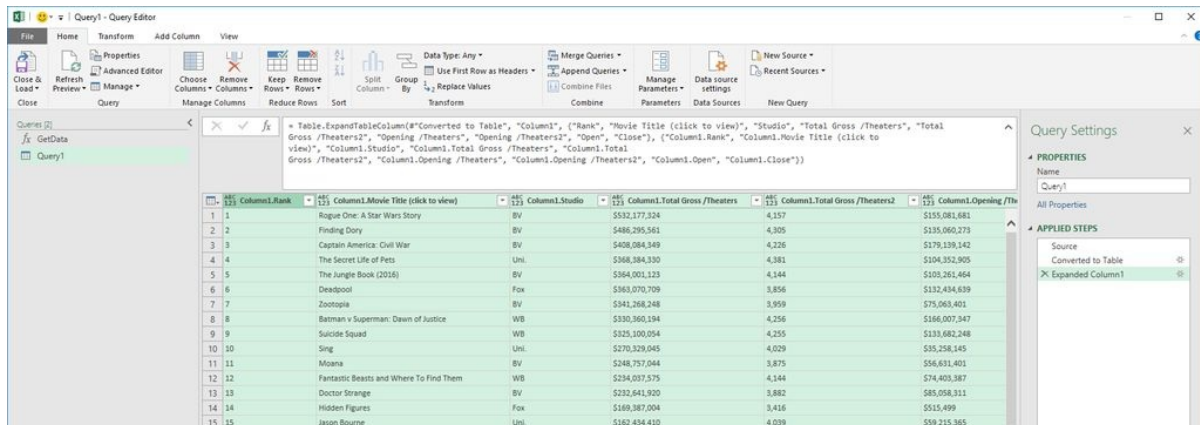
Once the table has been transformed, we can expand the table:



ABC 123	Column1
1	Table
2	Table
3	Table
4	Table

Figure 03-35: Expand the Table

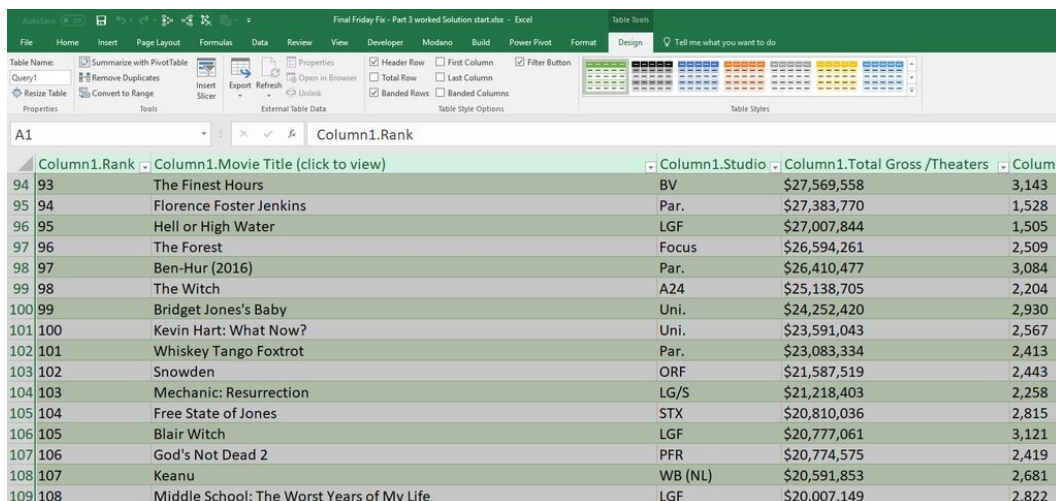
The expanded table should look something like this:



Column1.Rank	Column1.Movie Title (click to view)	Column1.Studio	Column1.Total Gross /Theaters	Column1.Total Gross /Theaters2	Column1.Opening /Theaters
1	Rogue One: A Star Wars Story	BV	\$532,177,324	4,157	\$155,081,681
2	Finding Dory	BV	\$486,295,561	4,305	\$135,060,273
3	Captain America: Civil War	BV	\$408,084,349	4,226	\$179,139,142
4	The Secret Life of Pets	Uni.	\$368,384,330	4,381	\$104,352,905
5	The Jungle Book (2016)	BV	\$364,001,123	4,184	\$109,361,464
6	Deadpool	Fox	\$363,070,709	3,856	\$132,414,639
7	Zootopia	BV	\$341,268,248	3,959	\$79,063,401
8	Batman v Superman: Dawn of Justice	WB	\$330,360,194	4,256	\$166,007,347
9	Suicide Squad	WB	\$325,100,054	4,255	\$133,682,248
10	Sing	Uni.	\$270,329,045	4,029	\$35,258,145
11	Moana	BV	\$248,757,044	3,875	\$56,631,401
12	Fantastic Beasts and Where to Find Them	WB	\$234,037,575	4,144	\$74,405,387
13	Doctor Strange	BV	\$232,641,920	3,882	\$85,058,311
14	Hidden Figures	Fox	\$169,387,004	3,416	\$515,499
15	Jason Bourne	Uni.	\$162,434,410	4,039	\$59,215,365

Figure 03-36: Expanded Table

Closing and loading will not result in an error but instead all of the movie data from the year of 2016 from the Box Office Mojo website:



Column1.Rank	Column1.Movie Title (click to view)	Column1.Studio	Column1.Total Gross /Theaters	Column1.Total Gross /Theaters2
94	The Finest Hours	BV	\$27,569,558	3,143
95	Florence Foster Jenkins	Par.	\$27,383,770	1,528
96	Hell or High Water	LGF	\$27,007,844	1,505
97	The Forest	Focus	\$26,594,261	2,509
98	Ben-Hur (2016)	Par.	\$26,410,477	3,084
99	The Witch	A24	\$25,138,705	2,204
100	Bridget Jones's Baby	Uni.	\$24,252,420	2,930
101	Kevin Hart: What Now?	Uni.	\$23,591,043	2,567
102	Whiskey Tango Foxtrot	Par.	\$23,083,334	2,413
103	Snowden	ORF	\$21,587,519	2,443
104	Mechanic: Resurrection	LG/S	\$21,218,403	2,258
105	Free State of Jones	STX	\$20,810,036	2,815
106	Blair Witch	LGF	\$20,777,061	3,121
107	God's Not Dead 2	PFR	\$20,774,575	2,419
108	Keanu	WB (NL)	\$20,591,853	2,681
109	Middle School: The Worst Years of My Life	LGF	\$20,007,149	2,822

Figure 03-37: All Movie Data

Now that we have dealt with the page number issue, let's move on...

Part 4: Fiddling with the URL

Next, let's head over to Telerik's software page (<https://www.telerik.com/fiddler>) to download Fiddler. This software will help us with this stage as it allows web session manipulation. When Windows has finished installing Fiddler, you should see something like this:

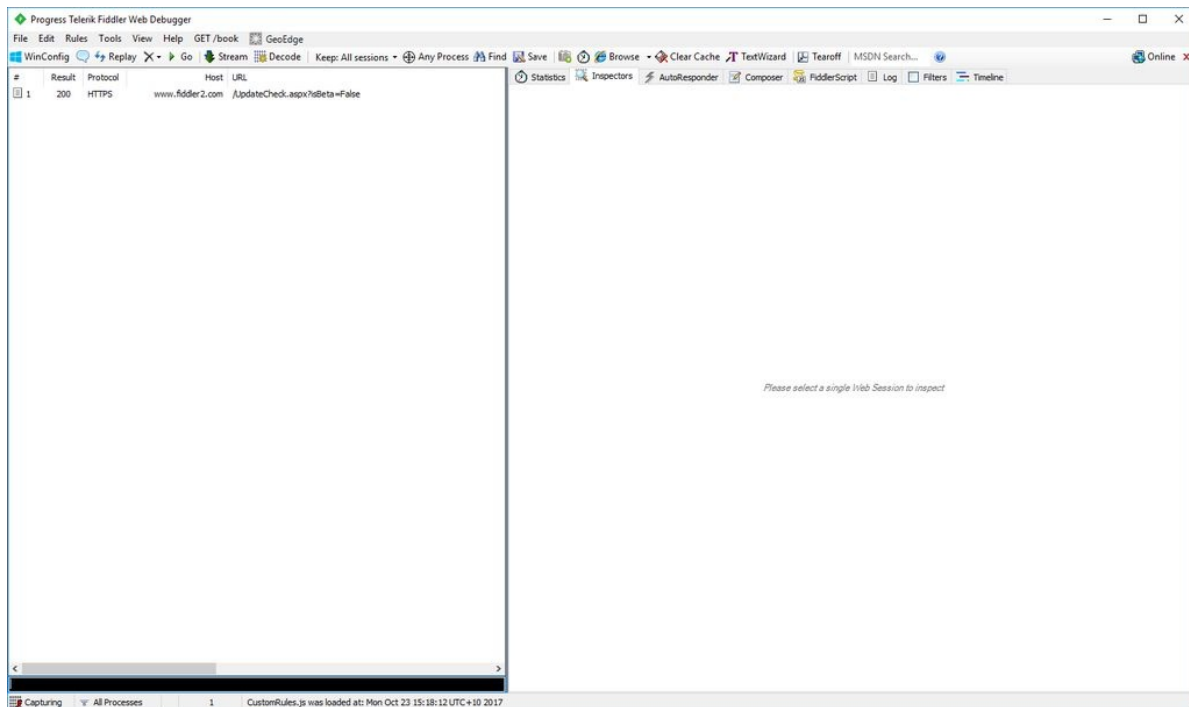


Figure 03-38: Fiddler

As the prompt 'Please select a single Web Session to inspect' suggests, we will open a window in the browser. Go ahead and navigate to the Hockey statistics website again (<http://www.quanthockey.com/khl/seasons/2017-18-khl-players-stats.html>) and we will start to see some interesting things appear on Fiddler:

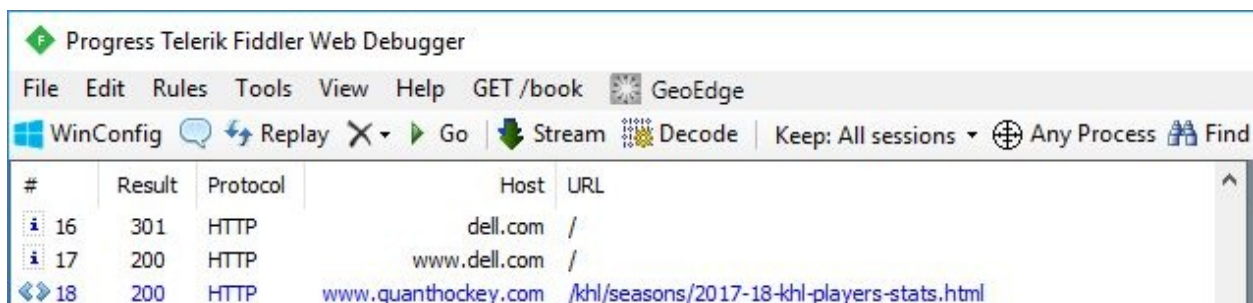


Figure 03-39: Fiddler Analysis 1

Fiddler takes the source of the URL and displays it here. Let's see what happens when we select page 2 of the Hockey Stats. Fiddler now returns with an

alternate URL:

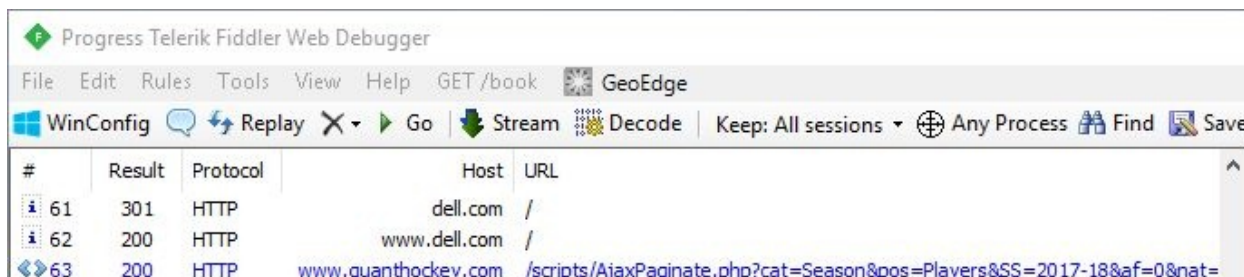


Figure 03-40: Fiddler Analysis 2

It seems to have been broken down into Seasons, and potentially pages. Let's copy it and save it to an Excel spreadsheet to aid us in discovering any patterns. Right click the line of URL and select 'Copy just URL'.

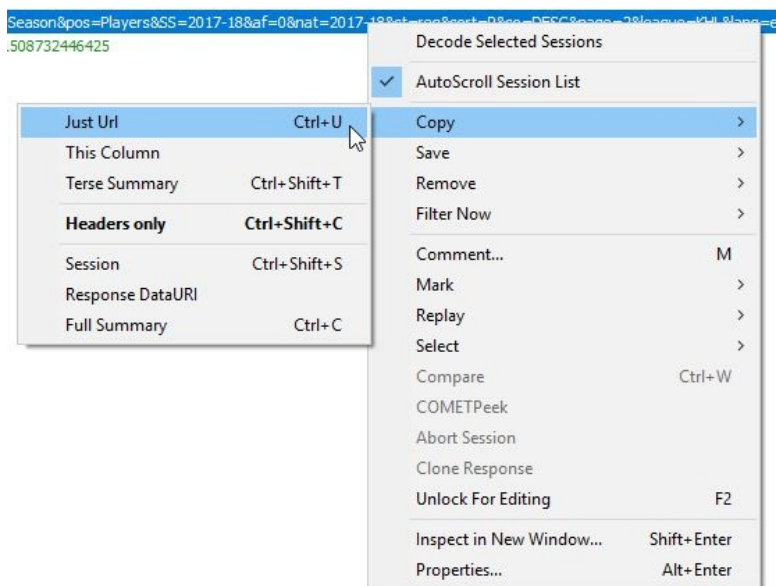


Figure 03-41: Copy Just URL

After repeating the process, a couple of times, we spot a pattern. Fiddler is able to retrieve the URL and break it down into pages! Therefore, we can finally use this to work with Power Query!

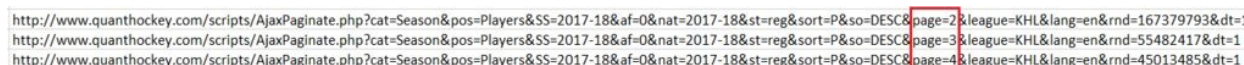


Figure 03-42: Spotting a Pattern

Now for the final part where we combine everything together.

Part 5: Putting it All Together

The first step is to create a new query in Power Query and create a new parameter:

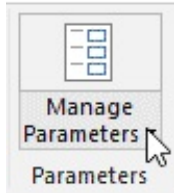


Figure 03-43: Manage Parameters

Let's name the parameter **PageNumber**, set it to a 'Decimal Number' type, and give it a current value of 1:

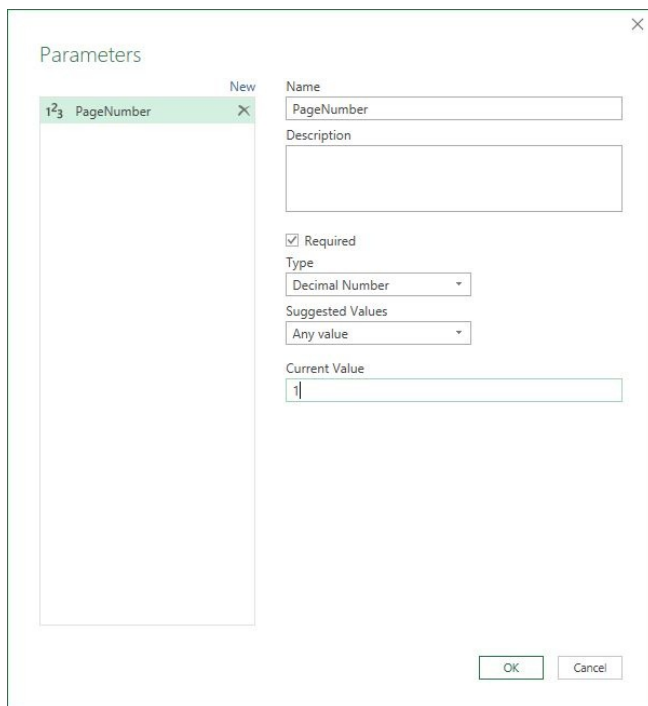


Figure 03-44: Parameters Dialog

Now create a new 'Blank Query' and paste the following (from before) into the formula bar:

Source =

Web.Page(Web.Contents("http://boxofficemojo.com/yearly/chart/?page=" & Number.ToText(page) & "&view=releasedate&view2=domestic&yr=2013&p=.htm"))),

Then modify it using the new URL provided from Fiddler:

```
=Web.Page(Web.Contents("http://www.quanthockey.com/scripts/AjaxPagin
cat=Season&pos=Players&SS=2017-18&af=0&nat=2017-
18&st=reg&sort=P&so=DESC&page=2&league=KHL&lang=en&rnd=1673
```

We also have to include the **PageNumber** parameter and the **Text.From** Power Query function to ensure that it is inserted into the URL as a text format. The following code should replace the page number (where the ampersand symbols mean concatenate or ‘join together’):

```
= "&Text.From(PageNumber)&"
```

yielding this:

```
=Web.Page(Web.Contents("http://www.quanthockey.com/scripts/AjaxPagin
cat=Season&pos=Players&SS=2017-18&af=0&nat=2017-
18&st=reg&sort=P&so=DESC&page="&Text.From(PageNumber)&"&leag
```

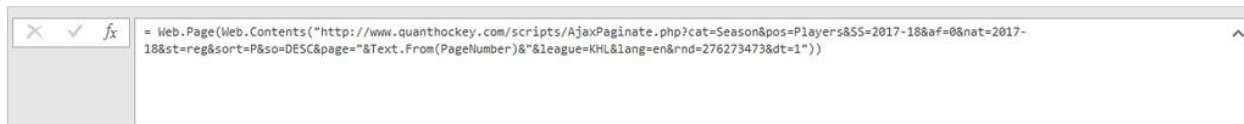


Figure 03-45: Example Code

As you can see, the **PageNumber** parameter has been linked into the URL. Hit **ENTER** and Power Query will return with a condensed table. The next step is to select the top right ‘Table’ option:

	A ^B _C Caption	A ^B _C Source	A ^B _C ClassName	A ^B _C Id	Data
1	null	Table	ps_tbl nowrap dt3	statistics	Table
2	Document	Service	null	null	Table

Figure 03-46: Selecting the Table Option

This will expand the table resulting in a table that only imports data from the first page, or the first 50 records:

44	44	RU	Alexander Khok...	24	F	18	5	10	15	6	4	2
45	45	RU	Pavel Chernov	27	F	22	6	8	14	24	7	1
46	46	RU	Dmitri Kugryshev	27	F	22	6	8	14	2	1	3
47	47	CA	Gilbert Brule	30	F	21	6	8	14	24	-1	3
48	48	CA	Eric O'Dell	27	F	18	6	8	14	12	12	1
49	49	CZ	Andrej Nestrail	26	F	22	4	10	14	2	-6	0
50	50	RU	Andrei Markov	39	D	21	4	10	14	8	-2	4

Figure 03-47: Table with Up to First 50 Records

Create a new blank query and copy this code in. It is a modified version of the **GetData** function:

```
= (PageNumber as number) => let
```



```
Source =
Web.Page(Web.Contents("http://www.quanthockey.com/scripts/AjaxPaginat
cat=Season&pos=Players&SS=2017-18&af=0&nat=2017-
18&st=reg&sort=P&so=DESC&page=" &Text.From(PageNumber)&" &leag
```

```
Data0 = Source{0}[Data],
```

```
#"Changed Type" = Table.TransformColumnTypes(Data0,{{"Rk",
Int64.Type}, {"", type text}, {"Name", type text}, {"Age", Int64.Type},
{"Pos", type text}, {"GP", Int64.Type}, {"G", Int64.Type}, {"A",
Int64.Type}, {"P", Int64.Type}, {"PIM", Int64.Type}, {"+/-", Int64.Type},
{"PPG", Int64.Type}, {"SHG", Int64.Type}, {"GWG", Int64.Type},
{"G/GP", type number}, {"A/GP", type number}, {"P/GP", type number}})
```

```
in
```

```
#"Changed Type"
```

The second section of code simply changes the data types accordingly for each column so that you don't have to do it!

Hit **ENTER** and rename the function to **PageData**:



Figure 03-48: PageData Function

Now, create another blank query and copy this code in (again, from before):

```
= List.Generate( ()=>
```

```
[Result= try PageData(1) otherwise null, Page = 1],
```

```
each [Result] <> null,
```

```
each [Result= try PageData(Page) otherwise null, Page = [Page] + 1],
```

```
each [Result])
```

Hit **ENTER** and change the name of the Query to **AllData**:

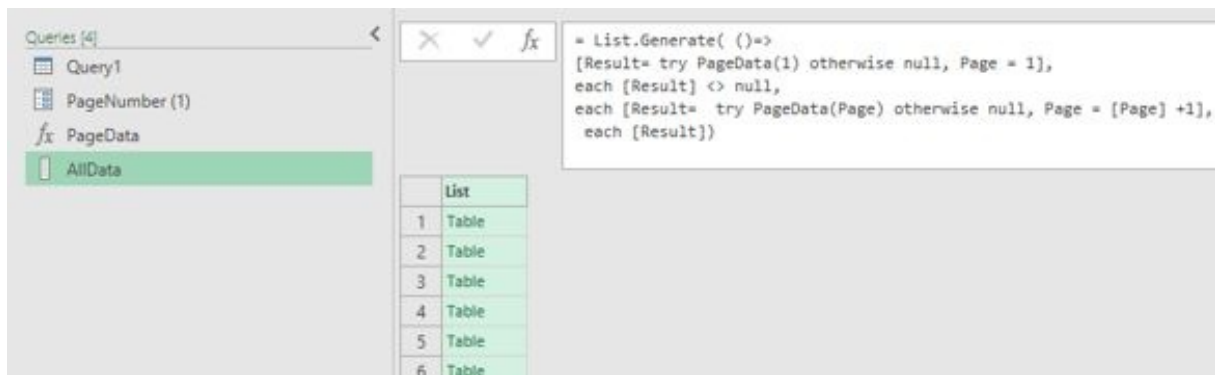


Figure 03-49: AllData

This time there are no modifications! We just need to convert this list into a table:

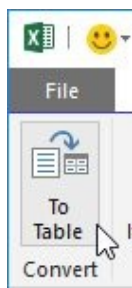


Figure 03-50: Convert to Table

Once Power Query has converted it into a table, we can expand the table:

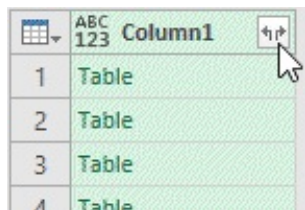


Figure 03-51: Expand Table (Again)

Expanding the table should yield this result, where Power Query is able to compile the entire list of Hockey players, not just the first 50:

	Column1.Rk	Column1.	Column1.Name	Column1.Age	Column1.Pos	Column1.GP	Column1.G	Column1.A	Column1.P
45	RU		Pavel Chernov	27	F	22	6	8	14
46	RU		Dmitri Kugryshev	27	F	22	6	8	14
47	CA		Gilbert Brule	30	F	21	6	8	14
48	CA		Eric O'Dell	27	F	18	6	8	14
49	CZ		Andrej Nestrasil	26	F	22	4	10	14
50	RU		Andrei Markov	39	D	21	4	10	14
51	CZ		Robin Hanzl	28	F	19	4	10	14
52	SE		Anton Lander	26	F	21	2	12	14
53	KZ		Kevin Dallman	36	D	20	2	12	14
54	LV		Miks Indrasis	27	F	23	8	5	13
55	RU		Sergei Kalinin	26	F	22	8	5	13
56	DK		Nicklas Jensen	24	F	18	7	6	13
57	KZ		Dustin Boyd	31	F	24	6	7	13
58	FI		Petri Kontiola	33	F	22	6	7	13
59	SE		Alexander Bergström	31	F	20	6	7	13
60	RU		Andrei Alexeyev	22	F	24	5	8	13
61	RU		Denis Parshin	31	F	23	5	8	13
62	CA		Colby Genoway	34	F	23	4	9	13
63	RU		Evgeny Lapenkov	33	F	22	4	9	13
64	CZ		Jakub Nakladal	30	D	22	4	9	13
65	RU		Valeri Nichushkin	22	F	18	8	4	12
66	RU		Daniil Vovchenko	21	F	23	7	5	12

Figure 03-52: Complete Table

We can now proceed to ‘Close & Load’.

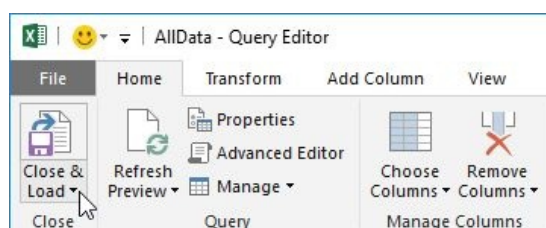


Figure 03-53: Close & Load

There you have it, all 829 Hockey Player stats (as at the time of writing) in one worksheet!

Column1.Rk	Column1.	Column1.Name	Column1.Age	Column1.Pos	Column1.GP	Column1.G	Column1.A	Column1.P	Column1.PIM	Column1.+/-
807	RU	Alexander Sharychenkov	26	G	1	0	0	0	0	
808	RU	Evgeny Ivannikov	26	G	1	0	0	0	0	
809	RU	Daniil Stalnov	23	D	1	0	0	0	4	-1
810	RU	Vladimir Sokhatsky	28	G	1	0	0	0	0	
811	RU	Alexander Trushkov	21	G	1	0	0	0	0	
812	RU	Sergei Magarilov	32	G	1	0	0	0	0	
813	RU	Ivan Fischenko	22	F	1	0	0	0	0	0
814	RU	Ivan Fedotov	21	G	1	0	0	0	0	
815	RU	Maxim Tretiak	21	G	1	0	0	0	0	
816	RU	Nikolai Molkov	22	G	1	0	0	0	0	
817	RU	Sergei Alexeev	23	D	1	0	0	0	0	0

Figure 03-54: Complete Table in Excel

Hopefully, Microsoft will introduce a new built in feature to circumvent all this nasty coding, but in the meantime...

About the Author



Dr. Liam Bastick FCA FCMA MVP

Liam has over 30 years' experience in financial model development / auditing, valuations, M&A, strategy, training and consultancy. He is a Director in the Melbourne office of SumProduct, having previously run several other modelling consultancies.

An experienced accountant and a professional mathematician, Liam has worked around the world with many internationally recognised clients. He has been awarded Microsoft's Most Valuable Professional (MVP) award nine times for his expertise in Excel and financial modelling. He writes in various accounting magazines around the world and is author of *An Introduction to Financial Modelling*.

Check out free downloads, online training and Excel / Power BI articles at www.sumproduct.com.

Part II: Data Preparation

Chapter 4: Creating Calendar Dimensions with Power Query

Author: Ken Puls, FCPA, FCMA

In this chapter we'll look at how to create a dynamic calendar on the fly, then populate it with columns for the date formats you may need for your model. Whether you are building a standard 12-month calendar, a 12-month calendar with a non-standard year end, or a calendar that follows a 4-4-5, 4-5-4, 5-4-4 week setup, Power Query can create it for you and ensure it covers the entire date range contained in your data. All you need to know is how...

To Create or not to Create?

When performing any type of analysis or reporting that slices data by date, it is important to make sure that there is a calendar dimension in your Power BI model. This table is essential for bridging multiple fact tables with dates, and is also important if you plan to run measures like TOTALMTD(), as a gap in the date range will cause the measure to return incorrect values.

Dynamic Calendars vs the Corporate Database

Let's be honest, building a calendar on the fly with Power Query sounds like work, and is bound to take resources to refresh. You might be asking whether it wouldn't be better to just pull the calendar table directly from the corporate database. The answer is absolutely yes, and in fact, doing so will allow you to build either Import or DirectQuery Power BI models.

Building a dynamic calendar table using the techniques in this chapter only applies to an Import model. If you have access to the calendar from corporate, then you should use it. But what if you don't? What if you're served up a diet of Excel or text files, and don't have access to a calendar that is blessed by corporate IT? THAT is when the steps in this chapter open things up for you.

Doesn't Power BI Use Default Date Tables?

While Microsoft does provide default date functionality, I'll admit that I'm not a fan of it. Why? Because I don't have ultimate control over what gets added to my data model and how. Using this functionality adds a hidden table to your model for every column of each table that contains date values. That means that if you have three different tables with dates in them, and you don't declare a specific date table properly, you get three copies of a hidden date table in your model. And if one of those tables contains a "StartDate" and "EndDate" column, you actually get four hidden tables, as each column is treated differently! Each of these hidden tables holds columns for Date, Day, Month, MonthNo, Quarter, QuarterNo and Year fields, and takes memory to store the unique values. In addition, without being properly linked to the other fact tables in the models, they don't offer you the benefit of cross filtering, as a properly linked calendar dimension table does.

My advice? Turn the automatic date table feature off and build your calendar tables properly. To turn off this feature (both globally and for the current file):

- Go to File -> Options and Settings -> Options
- Go to Global -> Data Load -> uncheck 'Auto date/time for new files'

- Go to Current File -> Data Load -> Time intelligence -> uncheck 'Auto date/time'.

Sample Data

The sample data for this chapter is built assuming we have both a table of Transactions, and a table of Budget values. They are at a different level of granularity, with transactions being recorded on a daily basis, and budgets being recorded on a monthly basis. The data model (as it stands) looks like this:

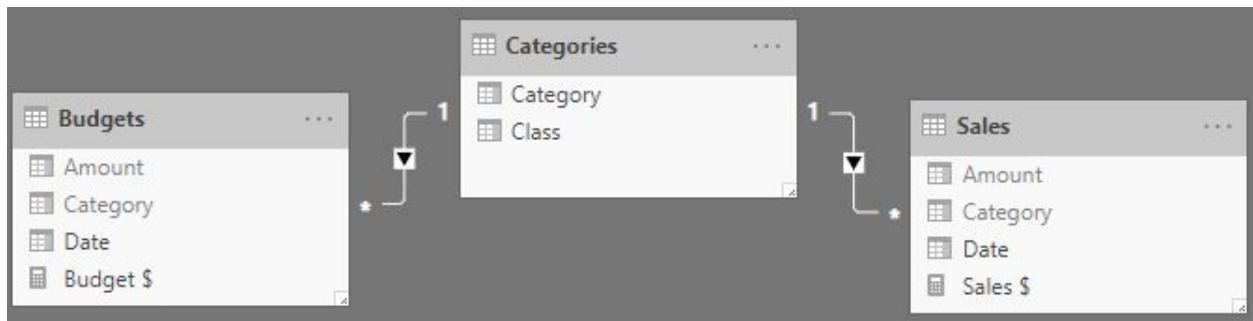


Figure 04-01: Sample data model

A sample of the data is shown below:

	A ^B _C Category	A ^B _C Class
1	Beer	Beverages
2	Wine	Beverages
3	Burgers	Food
4	Other	Other

	Date	A ^B _C Category	1 ² ₃ Amount
1	2018-01-01	Beer	175
2	2018-01-01	Beer	175
3	2018-01-01	Beer	175
4	2018-01-02	Beer	1785
5	2018-01-01	Wine	512
6	2018-01-01	Burgers	72
7	2018-01-02	Wine	873
8	2018-01-02	Wine	873
9	2018-01-02	Burgers	244

	Date	A ^B _C Category	1 ² ₃ Amount
1	2018-01-31	Beer	43683
2	2018-01-31	Wine	101558
3	2018-01-31	Burgers	13677
4	2018-02-28	Beer	104637
5	2018-02-28	Wine	21173
6	2018-02-28	Burgers	10884
7	2018-03-31	Beer	25203
8	2018-03-31	Wine	48346
9	2018-03-31	Burgers	11345
10	2018-04-30	Beer	83846

Figure 04-02: Sample data

While the data for this sample is contained entirely in the Power BI file, we'll pretend that the transactions have been sourced from a text file extract, and the budgets have been sourced from every accountant's favorite software: Excel.

The challenge here is that there is no calendar table provided at all, but we need to have one to act as a bridge between our fact tables, as well as provide the backbone for our date intelligence measures.

Creating a Dynamic Calendar Table

To create a fully dynamic calendar table, the first thing we need to figure out is the start date and the end date for the calendar. Once we have those, creating a complete calendar with a row for each day is actually very easy. The hard part is figuring out where to get the dates from.

Start dates and end dates can be hard coded in parameters, but the challenge here is that you need to update them when the data changes. Wouldn't it be better to just extract the earliest and latest dates right from the model data? The trick is which table to use for which date.

The key for the start date is picking up the data table that will always contain a record with the earliest date in all of your fact tables. A Sales or Transactions table is usually a good bet for this. For the end date, you may want the Sales table, but Budgets can often be better – at least – Budgets can be better providing that your company always budgets before sales happen. If you don't, then relying on your Sales table could be the better bet.

Regardless of the tables you choose, we'll force the boundaries to always cover the full fiscal year(s) for which we have data. As long as we pick the tables that will always cover the full date range, things will work flawlessly.

Recipe for StartDate and EndDate Queries

The dynamic calendar all begins with two queries: StartDate and EndDate. We'll walk through the process here of creating the StartDate query, then show the recipe as well as how it compares to the EndDate query.

To begin, we'll start with the Sales table, as it is the transactional table that will always hold the earliest date. Since it is sourced via Power Query, we can go to Edit Queries and open the Power Query Editor. From there:

- Right click the Sales query in the Queries pane (at left) -> Reference
- Right click the [Date] column -> Remove Other Columns
- Click the filter icon on the top of the [Date] column -> Date Filters -> Is Earliest
- Right click the [Date] column -> Remove Duplicates
- Select the [Date] column -> go to the Transform tab -> Date -> Year -> Start of Year
- Change the data type of the [Date] column to a date (Yes - even if it already is - as this will prevent a date from ever getting hard coded in the Power Query script!)

- Right click the date in row 1 (not the [Date] column header) -> Drill Down.

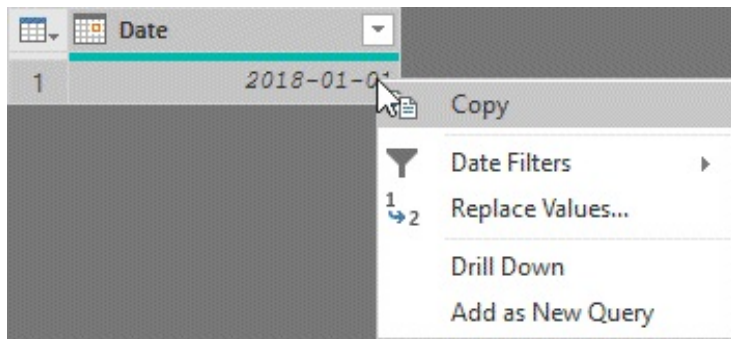


Figure 04-03: Right click the actual date, not the column header!

Provided you have done everything right, you’ve drilled down to a single date, forced it to the beginning of the year, and should now have a query that looks as shown here:

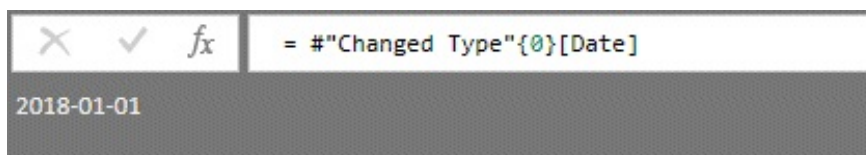


Figure 04-04: The contents of the StartDate query

A few key things to be aware of here:

1. You should not see a row number or column header of ‘List’ for your data point (if you do, you drilled down on the column header, not the data point itself)
2. The date will display in the format you use to display Jan 1, 20xx
3. You should not see a hard-coded date anywhere in the formula bar (if you do, you forgot to re-set the data type before drilling down).

Once you’re confident that everything is set up correctly, there are only two things left to do:

1. Rename the query to StartDate (notice there is no space between those words and watch the casing of those letters!)
2. Right click the StartDate query in the Queries pane -> un-check Enable Load.

And that’s it. You’ve got the first part of the calendar ready to go. Now it’s time to create the EndDate query. It follows virtually the same steps, with the exception of targeting the end of the year for the last date in the data set.

The table below shows a quick recipe summary for building both the StartDate and EndDate queries:

StartDate Recipe		EndDate Recipe
Reference the table with earliest date	●	Reference table with latest date
Remove all except the [Date] column	●	Remove all except the [Date] column
Filter to Dates -> Is Earliest	●	Filter to Dates -> Is Latest
Remove duplicates	●	Remove duplicates
Transform date to Year -> Start of Year	●	Transform date to Year - > End of Year
Change the data type to Date	●	Change the data type to Date
Right click the date cell -> Drill down	●	Right click the date cell -> Drill down
Name the query StartDate	●	Name the query EndDate
Disable the query load	●	Disable the query load

Figure 04-05: The recipe for creating StartDate and EndDate queries

Building the Base Calendar table

With the StartDate and EndDate queries set up, building the basic calendar table becomes very easy:

- Create a new blank query (right click in the Queries pane -> New Query - > Blank Query)
- Rename the query as Calendar
- Enter the following formula in the formula bar:
 - `= {Number.From(StartDate)..Number.From(EndDate)}`
- Go to List Tools -> Transform -> To Table -> OK
- Change [Column1] to a Date type
- Rename [Column1] to Date

CAUTION: *Your StartDate and EndDate queries must be capitalized consistently with the names of the queries you set up earlier. And if you put in spaces, you'll need to escape them with #" " like this: `= {Number.From("#"StartDate")..Number.From("#"End Date")}`*

In many cases, this single column Calendar table is all you really need. If you're comfortable here, then all that is left to do is:

- Go to Home -> Close & Apply
- Set the Calendar table as a date table:
 - Go to the Data view -> select the Calendar table
 - Go to the Modeling tab -> Mark as Date Table -> Date -> OK
- Go to the Model view and create relationships between
 - Calendar[Date] and Sales[Date]
 - Calendar[Date] and Budgets[Date].

It is also highly advisable to hide the [Date] fields on both the Sales and Budget tables, forcing users to pull the Date from the Calendar table, and avoid potential cross-filtering issues.

When complete, the model view will look as follows:

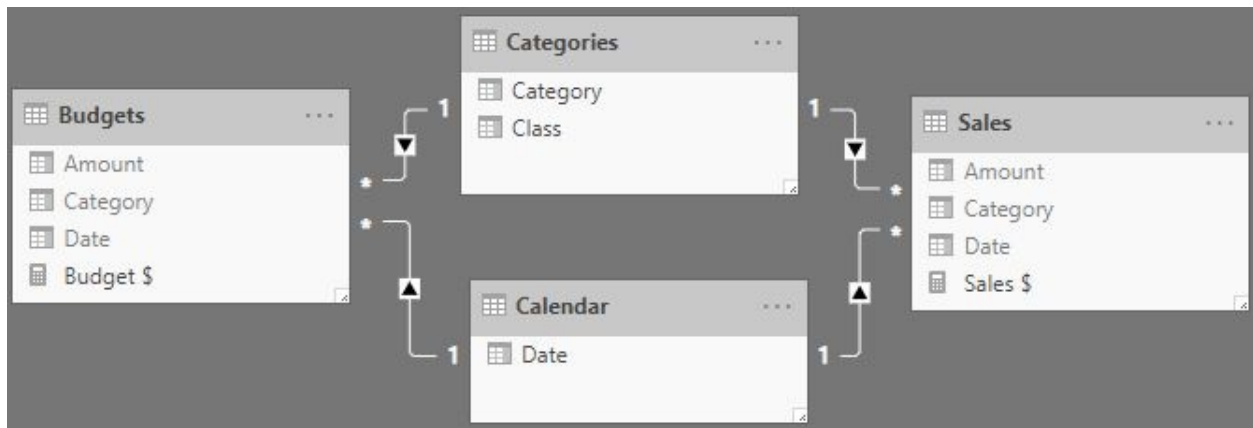


Figure 04-06: The contents of the StartDate query

Add any Additional Columns Needed

While Power BI does provide automatic date hierarchies for models, there are times where you'll prefer to take control of this yourself, or where you'll need to add date formats that fall outside the standard Year, Quarter, Month and Day provided by the hierarchy. Power Query provides many additional date transformations for you, which you can add to your table as follows:

1. Select Home -> Edit Queries
2. Edit the Calendar query
3. Select your Date column
4. Add Column -> Date -> select the format you wish to add
5. Repeat steps 2-3 as many times as necessary
6. Go to Home -> Close & Apply to load the new data into your model.

And that's it! You've got a fully dynamic calendar that will scale with your data!

Add Fiscal Year Ends to Your Calendar

The base Calendar is all well and good, but what if you don't work with a standard year end that ends on December 31 each year? This is not a problem at all, and we can even add columns for Fiscal Year, Fiscal Quarter and Fiscal Month.

The key is that you start with the regular Calendar table as described earlier. While it's not totally necessary in all cases, you may want to adjust your StartDate and EndDate queries so that they show your correct fiscal year start and end dates. But before you do, create a new query to hold the month number for the final month of your year end. The steps to do this are as follows:

- Create a new blank query
- Rename the query to YEMonth
- In the formula bar enter the numeric value of the final month of your year end (i.e. for a March 31 year end, enter 3, for a July 31 year end, enter 7)
- Right click the query in the Queries pane and uncheck Enable Load.

When complete, you should have a query that holds the month of your year end:

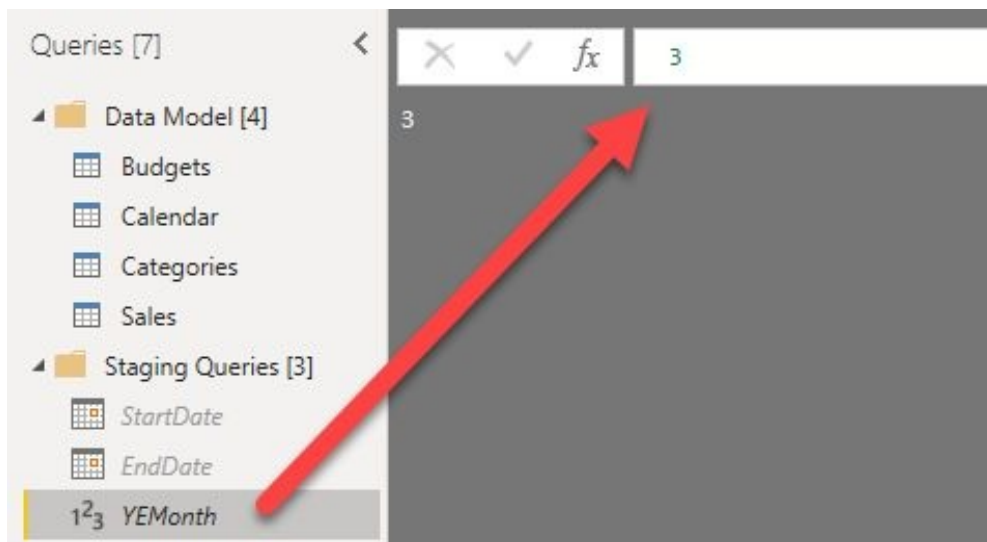


Figure 04-07: The YEMonth query set up for a March 31 year end

Next, we can adjust the StartDate and EndDate queries we built earlier following the recipe in the Table shown earlier. (Be aware that as you follow the next steps, you will be asked to confirm that you want to insert new steps. Say “Yes” each time you are prompted.)

- Edit each query
- Select the Calculated Start of Year (or Calculated End of Year) step

- Go to Add Column -> Custom Column
- Leave the column name as Custom, but enter the appropriate formula as follows:

Formula for adjusting the StartDate	Formula for adjusting the EndDate
=Date.AddMonths([Date], YEMonth - 12)	=Date.AddMonths([Date], YEMonth)

Figure 04-08: Modifying the StartDate and EndDate boundaries

- Click OK
- Right click the new [Custom] column -> Remove Other Columns
- Rename the [Custom] column to Date.

CAUTION: Depending on your data, this could end up padding your calendar with an extra year of dates on either end. Should you wish to change this, add or subtract 12 months immediately after the YEMonth in the formula.

A modified recipe for 12-month non-standard year ends is shown below:

StartDate Recipe	EndDate Recipe
Reference the table with earliest date	● Reference table with latest date
Remove all except the [Date] column	● Remove all except the [Date] column
Filter to Dates -> Is Earliest	● Filter to Dates -> Is Latest
Remove duplicates	● Remove duplicates
Transform date to Year -> Start of Year	● Transform date to Year - > End of Year
Add a custom column using this formula:	● Add a custom column using this formula:
=Date.AddMonths([Date], YEMonth - 12)	=Date.AddMonths([Date], YEMonth)
Remove all but the [Custom] column	● Remove all but the [Custom] column
Change the data type to Date	● Change the data type to Date
Right click the date cell -> Drill down	● Right click the date cell -> Drill down
Name the query StartDate	● Name the query EndDate
Disable the query load	● Disable the query load

Figure 04-09: StartDate and EndDate recipe for 12-month non-standard year ends

Remember, adjusting the year start and year end dates on your calendar is entirely optional. All of the patterns shown in the following section will work whether you do this or not!

Fiscal Periods

You may now wish to add further columns to enrich your Calendar table. A couple of good candidates might be Fiscal MonthOfYear and QuarterOfYear, since they won't match up to the standard 12-month calendar transforms available by default. We can easily do this, however, as the date transformations are not "standard" we'll need to do each with a Custom Column.

The following table shows the formulas required in order to create these columns

(assuming you have called your primary column [Date] and have created the YEMonth query described earlier in this chapter.)

Fiscal...	Required Columns	Formula
Month	[Date]	Date.Month(Date.AddMonths([Date] , - YEMonth))
Quarter	[Fiscal Month]	Number.RoundUp([Fiscal Month] / 3)
Month of Quarter	[Fiscal Month]	if Number.Mod([Fiscal Month], 3) = 0 then 3 else Number.Mod([Fiscal Month], 3)

Figure 04-10: Generating fiscal periods for a 12-month non-standard year end

Fiscal Year End

The most important field in the previous section is by far the [Fiscal Month] column. With that created, it becomes fairly easy to create our Fiscal Year End and Fiscal Quarter End columns:

Fiscal...	Required Columns	Formula
Year End	[Date] [Fiscal Month]	Date.EndOfMonth(Date.AddMonths([Date] , 12 - [Fiscal Month]))
Quarter End	[Date] [Fiscal Month of Quarter]	Date.EndOfMonth(Date.AddMonths([Date] , 3 - [Fiscal Month of Quarter]))

Table 09-11: Generating Fiscal Periods ends for a 12-month non-standard year end

The image shown below displays a 12-month calendar for a March 31 year end generated with the above formulas. Notice how the Fiscal Quarter End and Fiscal Year End periods change when moving from March 31 to April 1:

Date	1 ² ₃ Fiscal Month	1 ² ₃ Fiscal Quarter	1 ² ₃ Fiscal Month of Quarter	Fiscal Quarter End	Fiscal Year End
2018-03-27	12	4	3	2018-03-31	2018-03-31
2018-03-28	12	4	3	2018-03-31	2018-03-31
2018-03-29	12	4	3	2018-03-31	2018-03-31
2018-03-30	12	4	3	2018-03-31	2018-03-31
2018-03-31	12	4	3	2018-03-31	2018-03-31
2018-04-01	1	1	1	2018-06-30	2019-03-31
2018-04-02	1	1	1	2018-06-30	2019-03-31
2018-04-03	1	1	1	2018-06-30	2019-03-31
2018-04-04	1	1	1	2018-06-30	2019-03-31

Figure 04-12: Dynamically generated Calendar table for a March 31 year end

And remember, it's simple to update this to an April 30 year end: just change the YEMonth query value to four (4)!

Building a 4-4-5, 4-5-4 or 5-4-4 (ISO) Calendar

If you've never heard of one of these kinds of calendars, count yourself lucky, as they add a whole pile of complexity to your analysis. The main concept of a 4-4-5 calendar is to break the calendar down into four 13-week quarters per year which run in a 4-week, 4-week, 5-week pattern. Each quarter has 91 days, every year end is on a different day, and nothing follows the standard 12-month pattern. (The 4-5-4 and 5-4-4 are just different versions of the same concept with the 5-week period occurring in a different order.)

Why would anyone want to build a calendar like this? Simple: in retail, comparing May 12 from one year to the next isn't a good comparison, as it may be a Sunday one year (like 2019) and a Tuesday the next (like 2020). With its consistent pattern, the 4-4-5 calendar (and their variants), allow us to easily compare the third Sunday of this quarter against the third Sunday of the same quarter in the prior year, thereby drawing a much more accurate comparison.

As you can imagine, creating 4-4-5 calendars (and their variants) are a little tricky. Believe it or not, we still start with the base calendar pattern, but the secret is to make sure that the first day of your calendar isn't Jan 1, but rather the first day of one of your fiscal years.

From there, we need to add a "DayID" column which is then used to build up all the other PeriodID columns. These are special columns which always increase in value, never resetting at the end of a period, and their existence is important to allow writing DAX measures to compare values between periods. With the PeriodID columns in place, we can then build a variety of other date formats as well.

One caveat that we should acknowledge here: there are hundreds of ways that companies modify this style of calendar. They may add a 53rd week every five to six years to "reset" their year-end. They may set up a 53rd week if there are more than x days remaining in the week, or other manifestations as well. This pattern deals with a predictable cycle that doesn't cater to these issues. While these variations can be built, the patterns in this chapter will need to be adjusted in order to do so.

Creating StartDate and EndDate queries for 4-4-5 calendars

Let's assume that we need a 4-4-5 calendar. Our past few fiscal years started on 2017-01-01, 2017-12-31 and 2018-12-30 (using the ISO system of year-month-day). How can we adjust the calendar pattern to dynamically choose the correct

year start and end dates based on the data in the file?

The easiest way to do this is to create another new blank query in order to hold one of our specific start dates:

- Create a new blank query and rename it to **Start445**
- In the formula bar enter the starting date for one of your years (we'll use 2017-01-01)
- Right click the query in the query pane and uncheck Enable Load.

If you've done everything correctly, you should get a new query that displays with the Date icon in the queries pane:

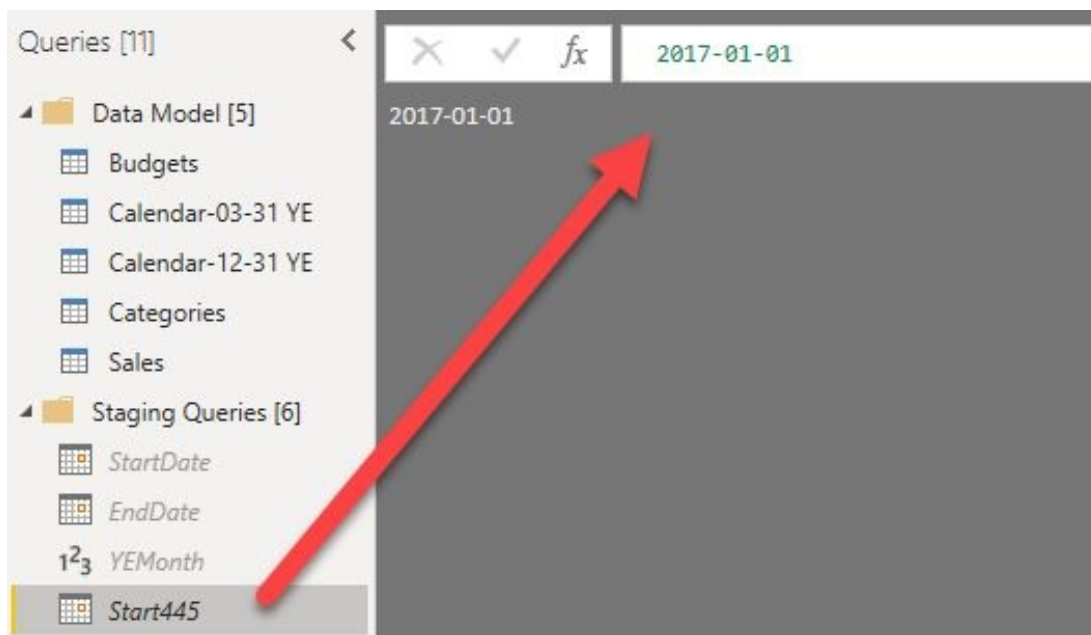


Figure 04-13: Recording the start of any fiscal year

While we could now modify the original StartDate and EndDate queries for the next steps, the following modified recipe will allow you to load a 4-4-5 calendar in the same file as the original calendar, thereby allowing you multiple ways to slice your data:

StartDate Recipe		EndDate Recipe
Reference the table with earliest date	●	Reference table with latest date
Remove all except the [Date] column	●	Remove all except the [Date] column
Filter to Dates -> Is Earliest	●	Filter to Dates -> Is Latest
Remove duplicates	●	Remove duplicates

Add a custom column using this formula:	•	Add a custom column using this formula:
<code>=Date.AddDays(Start445, 364 * Number.Round(Duration.Days(Duration.From([Date] - Start445)) / 364 , 0))</code>		<code>=Date.AddDays(StartDate445, 364 * Number.RoundUp(Duration.Days([Date] - StartDate445) / 364 , 0) - 1)</code>
Remove all but the [Custom] column	•	Remove all but the [Custom] column
Change the data type to Date	•	Change the data type to Date
Right click the date cell -> Drill down	•	Right click the date cell -> Drill down
Name the query StartDate445	•	Name the query EndDate445
Disable the query load	•	Disable the query load

Figure 04-14: The StartDate and EndDate recipe for 4-4-5 calendars and their variants

The formulas in the recipe are a little complicated, but the great thing about them is that they will automatically adjust to use the earliest fiscal year start and fiscal year ends required based on your data. This is fantastic, as it means that the calendar table will automatically span only the required years for the data model, without bloating the calendar table unnecessarily.

Notice in the image below, the StartDate445 query is not returning a date of 2017-01-01, but rather 2017-12-31. Since the first date in the Sales table is 2018-01-01, the most recent 445 year begins on 2017-12-31, and the formula generates that for us automatically!

--	--

Figure 04-15: The StartDate445 (left) and EndDate445(right) queries

With the StartDate445 and EndDate445 queries created, the base calendar can be created through the usual recipe steps:

- Create a new blank query (right click in the Queries pane -> New Query -> Blank Query)
- Rename the query as **Calendar445**
- Enter the following formula in the formula bar:
 - `= {Number.From(StartDate445)..Number.From(EndDate445)}`

- Go to List Tools -> Transform -> To Table -> OK
- Change [Column1] to a Date type
- Rename [Column1] to Date.

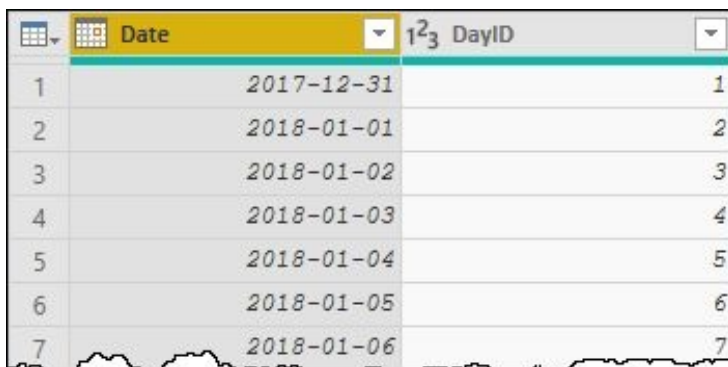
Creating the “DayID” column

It’s now time to add the second most critical column for the Calendar445 table: the DayID column. This column is used to drive every other PeriodID column, and therefore the rest of the calendar columns as well.

After ensuring that your calendar starts from the first day of one of your fiscal years, you simply need to:

- Go to Add Column -> Index Column -> From 1
- Rename the new column to DayID.

And that’s it! Now the calendar can be fleshed out with the rest of the PeriodID columns:



	Date	DayID
1	2017-12-31	1
2	2018-01-01	2
3	2018-01-02	3
4	2018-01-03	4
5	2018-01-04	5
6	2018-01-05	6
7	2018-01-06	7

Figure 04-16: The base Calendar445 table starting from the most recent fiscal year beginning

Creating the remaining PeriodID columns

Next, you’ll need to add the remaining Period ID columns in order to drive other date displays. These are all created via the Add Column -> Custom Column dialog using the following formulas:

Column	Required Columns	Formula
WeekID	[DayID]	Number.RoundUp([DayID]/7)
MonthID (for 4-4-5 calendars)	[DayID]	Number.RoundDown([DayID]/91)*3+ (if Number.Mod([DayID],91)=0 then 0 else if Number.Mod([DayID],91)<=

		28 then 1 else if Number.Mod([DayID],91)<= 56 then 2 else 3)
MonthID (for 4-5-4 calendars)	[DayID]	Number.RoundDown([DayID]/91)*3+ (if Number.Mod([DayID],91)=0 then 0 else if Number.Mod([DayID],91)<= 28 then 1 else if Number.Mod([DayID],91)<= 63 then 2 else 3)
MonthID (for 5-4-4 calendars)	[DayID]	Number.RoundDown([DayID]/91)*3+ (if Number.Mod([DayID],91)=0 then 0 else if Number.Mod([DayID],91)<= 35 then 1 else if Number.Mod([DayID],91)<= 63 then 2 else 3)
QuarterID	[DayID]	Number.RoundUp([DayID]/91)
YearID	[DayID]	Number.RoundUp([DayID]/364)

Figure 04-17: Formulas for PeriodID columns for 4-4-5 calendar variants

The image below shows all PeriodID columns for a 4-4-5 calendar which starts on 2017-12-31:

	Date	¹ ₂ DayID	¹ ₂ WeekID	¹ ₂ MonthID	¹ ₂ QuarterID	¹ ₂ YearID
362	2018-12-27	362	52	12	4	1
363	2018-12-28	363	52	12	4	1
364	2018-12-29	364	52	12	4	1
365	2018-12-30	365	53	13	5	2
366	2018-12-31	366	53	13	5	2
367	2019-01-01	367	53	13	5	2
368	2019-01-02	368	53	13	5	2

Figure 04-18: PeriodIDs created for a 4-4-5 Calendar

Adding Other Fiscal Periods

From this point forward, it is just a simple matter of deciding which date columns you want on your calendar table and adding them based on the formulas in the tables below.

Note that while the DayID column is required for each of the columns in the PeriodID section, the remainder of the formulas in this chapter require a variety of columns. Each required precedent column is included in the tables with the formula, so watch carefully to make sure you don't miss one that is essential for the format you're adding.

PRO TIP: *If you don't want the precedent column in your table... use it to create the field you do want, and then remove it afterwards. Power Query won't mind!*

Fiscal Year Columns

Column	Required Columns	Formula
Year *	StartDate445 [YearID]	Date.Year(Date.From(StartDate))+ [YearID]

Figure 04-19: Formula to generate the Fiscal Year

NOTE: *Depending on the first date used and the fiscal year you wish to represent for it, you may need to add or subtract 1 from the end result.*

X of Year Columns

... of Year	Required Columns	Formula
Quarter	[QuarterID]	Number.Mod([QuarterID]-1,4)+1
Month	[MonthID]	Number.Mod([MonthID]-1,12)+1
Week	[WeekID]	Number.Mod([WeekID]-1,52)+1
Day	[DayID]	Number.Mod([DayID]-1,364)+1

Figure 04-20: Formulas for generating QuarterOfYear, MonthOfYear, etc.

X of Quarter Columns

... of Quarter	Required Columns	Formula
Month	[MonthOfYear]	Number.Mod([MonthOfYear]-1,3)+1
Week	[WeekOfYear]	Number.Mod([WeekOfYear]-1,13)+1

Day	[DayOfYear]	Number.Mod([DayOfYear]-1,91)+1
-----	-------------	--------------------------------

Figure 04-21: Formulas for generating MonthOfQuarter, WeekOfQuarter, etc.

X of Month Columns

... of Month	Required Columns	Formula
Day (for 4-4-5 calendars)	[DayOfQuarter] [MonthOfQuarter]	if [MonthOfQuarter] = 1 then [DayOfQuarter] else if [MonthOfQuarter] = 2 then [DayOfQuarter] - 28 else [DayOfQuarter] - 35
Day (for 4-5-4 calendars)	[DayOfQuarter] [MonthOfQuarter]	if [MonthOfQuarter] = 1 then [DayOfQuarter] else if [MonthOfQuarter] = 2 then [DayOfQuarter] - 28 else [DayOfQuarter] - 63
Day (for 5-4-4 calendars)	[DayOfQuarter] [MonthOfQuarter]	if [MonthOfQuarter] = 1 then [DayOfQuarter] else if [MonthOfQuarter] = 2 then [DayOfQuarter] - 35 else [DayOfQuarter] - 63
Week *	[DayOfMonth]	Number.RoundUp([DayOfMonth]/7)

Figure 04-22: Formulas for generating DayOfMonth and WeekOfMonth

CAUTION: The appropriate [DayOfMonth] column must be created before the [WeekOfMonth] column can be created.

X of Week Columns

... of Week	Required Columns	Formula
Day	[DayOfYear]	Number.Mod([DayOfYear]-1,7)+1

Figure 04-23: Formula for building DayOfWeek

Days in X Columns

Days in...	Required	Formula
------------	----------	---------

	Columns	
Year	N/A	364
Quarter	N/A	91
Month (for 4-4-5 calendars)	[WeekOfQuarter]	if [WeekOfQuarter] > 8 then 35 else 28
Month (for 4-5-4 calendars)	[WeekOfQuarter]	if [WeekOfQuarter] > 4 and [WeekOfQuarter] <10 then 35 else 28
Month (for 5-4-4 calendars)	[WeekOfQuarter]	if [WeekOfQuarter] < 5 then 35 else 28
Week	N/A	7

Figure 04-24: Formulas for generating DaysInYear, DaysInQuarter, etc.

NOTE: Only DaysInMonth requires a formula in this case, as all the other versions have a consistent number of days in the given period.

Start of X Columns

Start of ...	Required Columns	Formula
Week	[Date], [DayOfWeek]	Date.AddDays([Date],[-([DayOfWeek]-1))
Month	[Date], [DayOfMonth]	Date.AddDays([Date],[-([DayOfMonth]-1))
Quarter	[Date], [DayOfQuarter]	Date.AddDays([Date],[-([DayOfQuarter]-1))
Year	[Date], [DayOfYear]	Date.AddDays([Date],[-([DayOfYear]-1))

Figure 04-25: Formulas for generating StartOfWeek, StartOfMonth, etc.

End of X Columns

End of ...	Required Columns	Formula
Week	[StartOfWeek]	Date.AddDays([StartOfWeek],6)
Month	[StartOfMonth], [DaysInMonth]	Date.AddDays([StartOfMonth], [DaysInMonth]-1)

Quarter	[StartOfQuarter]	Date.AddDays([StartOfQuarter],91-1)
Year	[StartOfYear]	Date.AddDays([StartOfYear],364-1)

Figure 04-26: Formulas for generating EndOfWeek, EndOfMonth, etc.

Summary

No matter how you want to slice up your data by time periods, Power BI can handle it. The secret is getting your hands on a proper calendar table in order to do the job. And if IT doesn't have one (or won't share), then no big deal, you can simply create one on the fly.

Even better than just being able to work around potential problems, however, is the ability to expand your intelligence to prototype comparisons that people in your company may not have been able to accomplish previously. Have you ever had to report using one calendar for head office, another for internal purposes, and yet another to make comparisons for reasonability? I have...

In the golf industry, we ran a season from May 1 to Apr 30. Our financial year end was December 31, and we issued payroll bi-weekly. And naturally, we did a fair amount of retail sales too, so wouldn't it be nice to have a 4-4-5 calendar for comparisons?

Well why not? Today, with Power Query at our fingertips, we can create multiple calendar tables for a single solution. Whether you have a single fact table or multiple fact tables, so long as each calendar dimension has a column of unique dates, there is no limit to how many dimension tables we can add in:

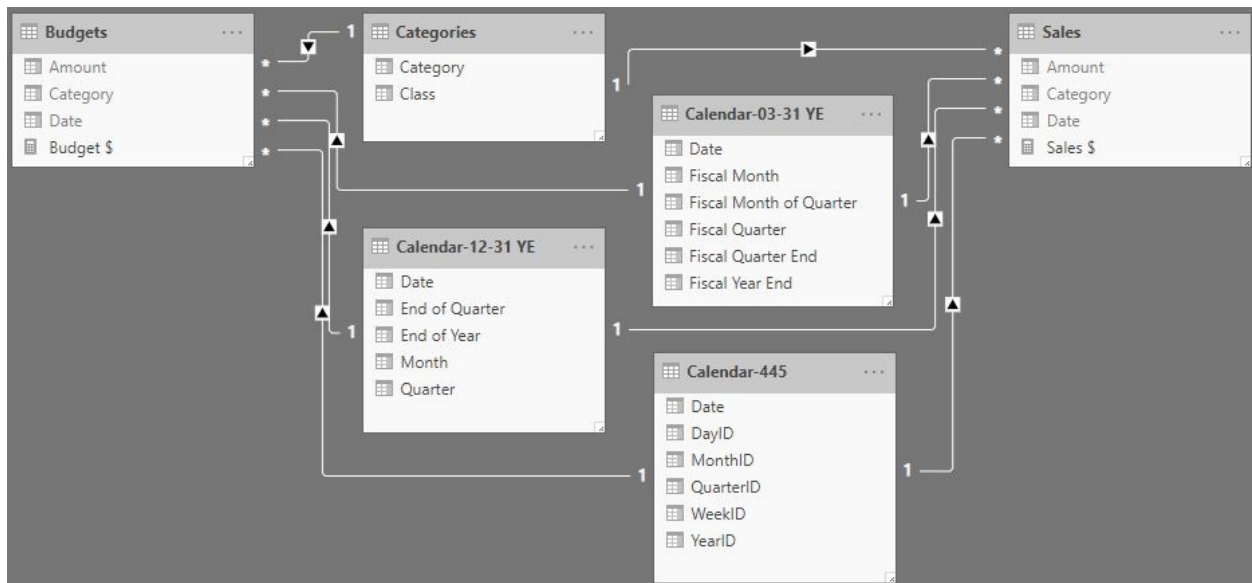


Figure 04-27: Multiple calendars linked into a single model

And just like that we can now create reports for each different audience, providing the date formats that they want to see for categorization and slicing:

12-Month Dec 31 Year End			12-Month Mar 31 Year End			4-4-5 Calendar		
End of Year	Sales \$	Budget \$	Fiscal Year End	Sales \$	Budget \$	EndOfYear	Sales \$	Budget \$
December 31, 2018	1,070,660	1,806,525	March 31, 2018	455,089	380,506	December 29, 2018	1,070,660	1,675,509
January 31, 2018	128,728	158,918	January 31, 2018	128,728	158,918	January 27, 2018	118,619	
February 28, 2018	143,930	136,694	February 28, 2018	143,930	136,694	February 24, 2018	132,276	158,918
March 31, 2018	182,431	84,894	March 31, 2018	182,431	84,894	March 10, 2018	204,194	221,588
April 30, 2018	131,234	147,794	March 31, 2019	615,571	1,426,019	April 28, 2018	117,020	
May 31, 2018	143,810	204,561	April 30, 2018	131,234	147,794	May 26, 2018	123,119	147,794
June 30, 2018	181,149	72,674	May 31, 2018	143,810	204,561	June 9, 2018	216,054	277,235
July 31, 2018	151,653	224,975	June 30, 2018	181,149	72,674	July 28, 2018	132,946	
August 31, 2018	7,725	95,685	July 31, 2018	151,653	224,975	August 25, 2018	26,432	224,975
September 30, 2018		245,954	August 31, 2018	7,725	95,685	September 8, 2018		95,685
October 31, 2018		96,039	September 30, 2018		245,954	October 27, 2018		245,954
November 30, 2018		207,321	October 31, 2018		96,039	November 24, 2018		96,039
December 31, 2018		131,016	November 30, 2018		207,321	December 8, 2018		207,321
Total	1,070,660	1,806,525	December 31, 2018		131,016	December 28, 2019		131,016
			Total	1,070,660	1,806,525	January 26, 2019		131,016
						Total	1,070,660	1,806,525

Figure 04-28: The same facts sliced up based on different calendars

About the Author



KEN PULS, FCPA, FCMA, is the president of Excelguru Consulting Inc, and has been a Microsoft MVP in various categories since 2006. Ken is a blogger, author and trainer with over 20 years of corporate accounting and IT experience. He is one of the world's leading experts in Microsoft Excel, Power Query and Power BI, and provides live and online training to clients around the world.

For self service BI training, look us up at www.excelguru.ca, our consulting division is www.cudas.com, and for online Power Query training you can't beat our academy at <https://academy.powerquery.training>.

Chapter 5: Transform and combine your data with ETL tool named Power Query

Author: Jesus Gil | Dr Rudo SQL

Since its birth Power Query has become an easy-to-use ETL tool. However, how can we exploit this ease of use? What is the best practice to do it? In this chapter we will talk about how and what to do to achieve it.

What is Power Query?

In Microsoft's own words, Power Query it's described as following:

Power Query is the Microsoft Data Connectivity and Data Preparation technology that enables business users to seamlessly access data stored in hundreds of data sources and reshape it to fit their needs, with an easy to use, engaging and no-code user experience.

Supported data sources include a wide range of file types, databases, Microsoft Azure services and many other third-party online services. Power Query also provides a Custom Connectors SDK so that third parties can create their own data connectors and seamlessly plug them into Power Query.

Where is used Power Query?

Power Query was born on 6th July 2013 previously known as codename “[Data Explorer](#)”, first as an add-in for Excel and then later incorporated as a full feature of Excel and Power BI.

To see the original post typing

<https://blogs.msdn.microsoft.com/dataexplorer/2013/07/06/data-explorer-is-now-microsoft-power-query-for-excel/>

In fact, do you remember the famous Power BI on Excel?

- Power Query
- Power Pivot
- Power Map
- Power View

All these add-ins were integrated on Excel 2013 and all under the same file “*.xlsx”

These add-ins evolved and became powerful tools or, as the case may be, part of the visualizations of Power BI (i.e. power map)

In our days you can choose where you’ll use Power Query because is natively integrated in several Microsoft Products as

- Microsoft Power BI
- Microsoft Excel
- Microsoft SQL Server Data Tools for Visual Studio
- Microsoft Common Data Service for Apps

For more info about it see <https://docs.microsoft.com/en-us/power-query/power-query-what-is-power-query#where-to-use-power-query>

Why do people love Power Query?

The answer from my viewpoint is because Power Query is a great tool, support you to pull data from many different sources to can see it all in the same place together. Let me explain:

- It has a simple interface based on ribbons with buttons,
- Any user can learn how upload data because the user interface was designed to understand the process through simple clicks,
- Power Query has a powerful language (named M) to code and perform complex data manipulations,
- It's easy to connect to multiple data sources and merge data, and (most important)
- You can work with both structured and unstructured data

ETL: The concept

ETL is an acronym that means Extract, Transform and Load.

The ETL process became a famous concept early in the 1970s. This is the basic concept to upload data to our data model. ETL processes are mostly used on Data Warehouse, Data Marts, ODS or in our case uploading data to Power BI.

With Power Query we will be building the ETL process to copy data from one or more sources into Power BI, as described in the steps below:

Extraction: will copy or extracting data from sources (homogenous or heterogeneous)

Transformation: will process the data and applying for example data cleansing, data types conversions, business rules, etcetera and finally transforming the data in a format that can be understood by Power BI

Load: will be the process responsible for inserting the data into our Power BI data model

Building the ETL with Power Query

With Power Query we can access data stored in several data sources and it offers the best no-code user experience: You can access file types such as csv, Excel, databases, third-party online services, Microsoft Azure, SQL Server On-premise services and many more.

Note: Power Query also provides the possibility of building your own connector using the Data Connector SDK. All the tools to do this are public and stored in a GitHub repo. For more information, see <https://github.com/Microsoft/DataConnectors>.

When you are using Power BI desktop, all your ETL functionality will be provided via the Power Query Editor. To launch it, click on Edit Queries from the Home tab.

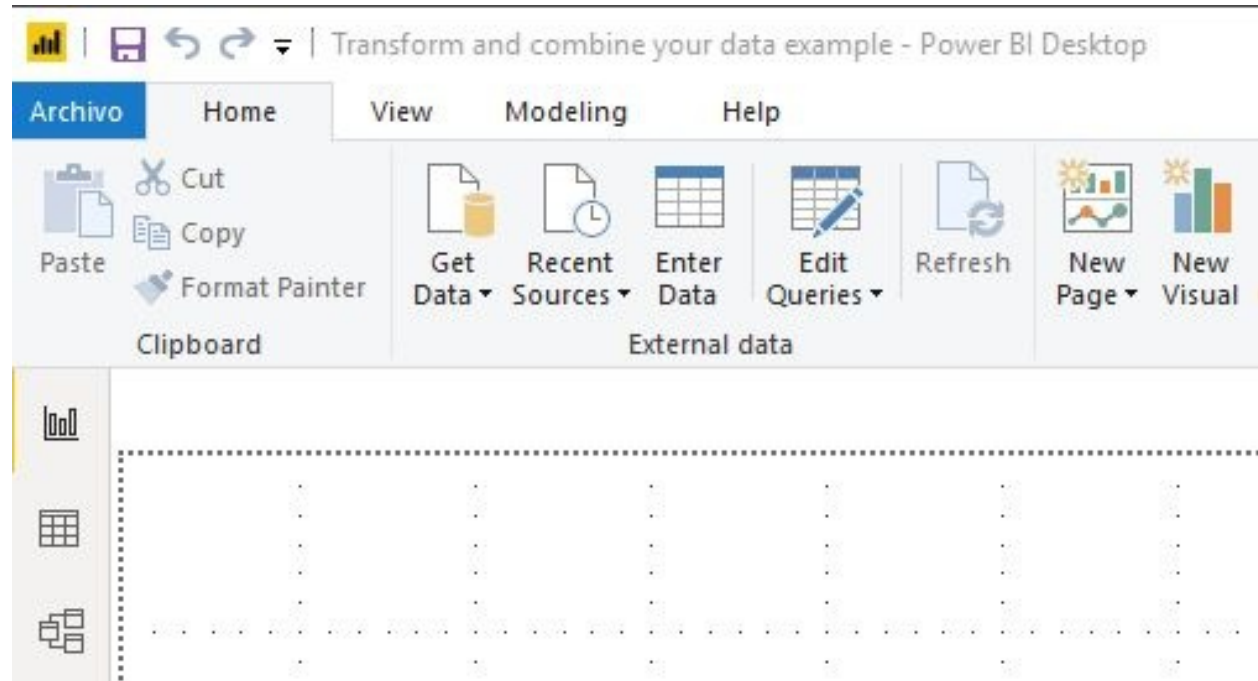


Figure 05-01: Launching the Power Query editor

Why do we get taken to a blank pane?

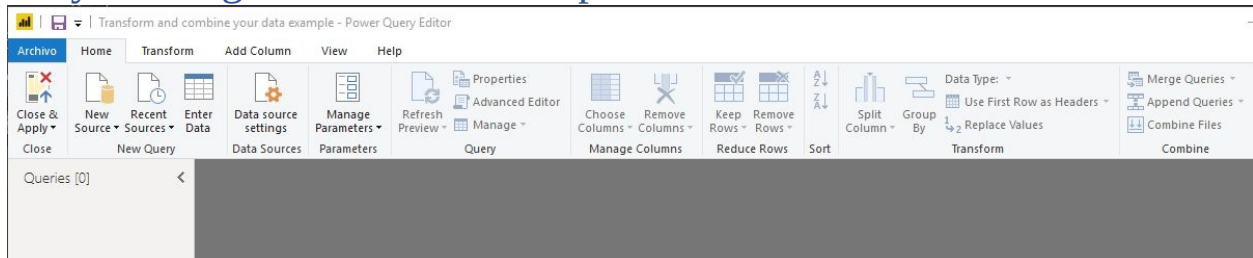


Figure 05-02: Black pane in Power Query editor

Because we not have data connections yet, so the Power Query Editor is waiting you import data and build data connections.

If you open a previous report, you'll see that have a Queries and data.

Now we will load information from the following Web data source and then you can begin to shape the data found at

https://en.wikipedia.org/wiki/UEFA_Champions_League.

Clicking on New Sources will show the Get Data window. Select Other sources, Web data source and finally click on Connect button.

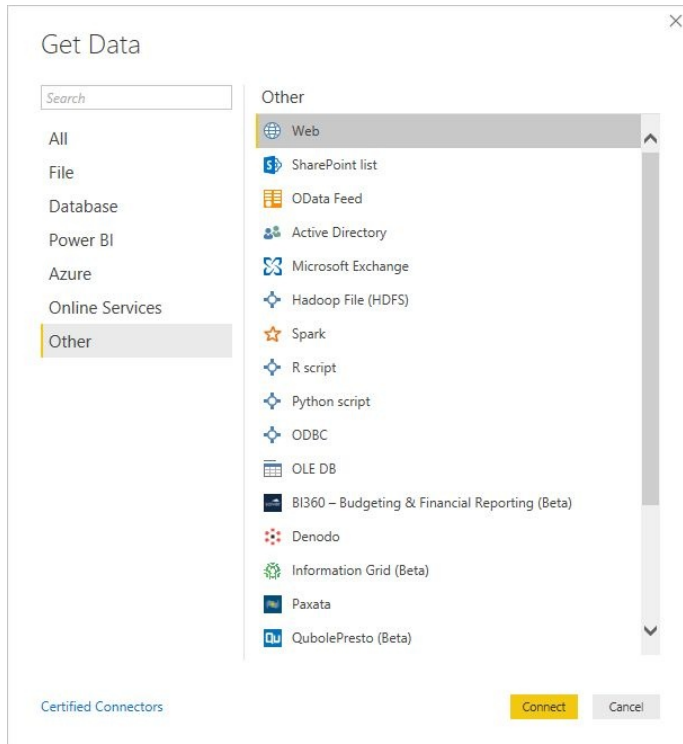


Figure 05-03: Get Data window, to select the data source

Type the URL source and click the OK button.

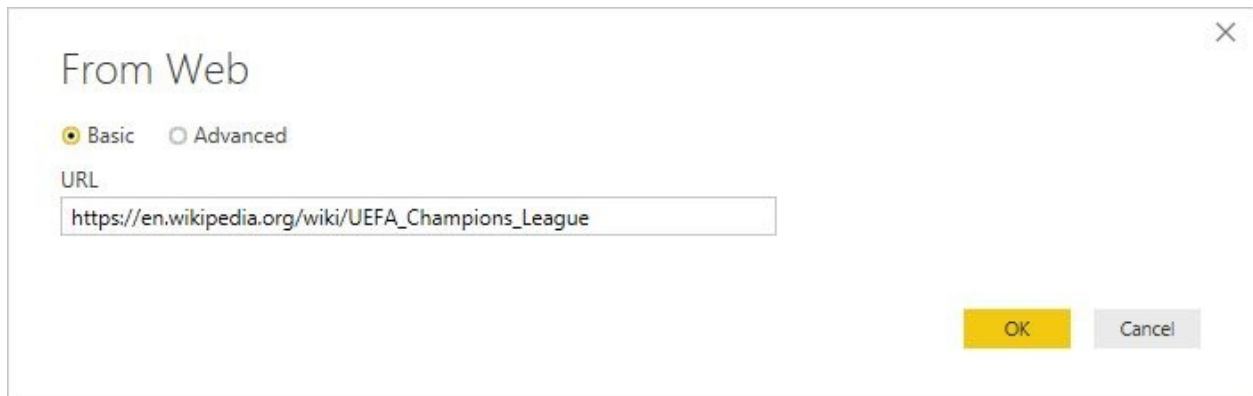


Figure 05-04: Typing the URL source

The Access Web Content window asks us the type of access we will have to the site. In our case we choose the Anonymous access (because we don't need any kind of credential to get the info) and then click on Connect button.

- Tip: the best practice is always verifying the data authenticity, use Windows or Organizational account over all when we will get sensitive data.

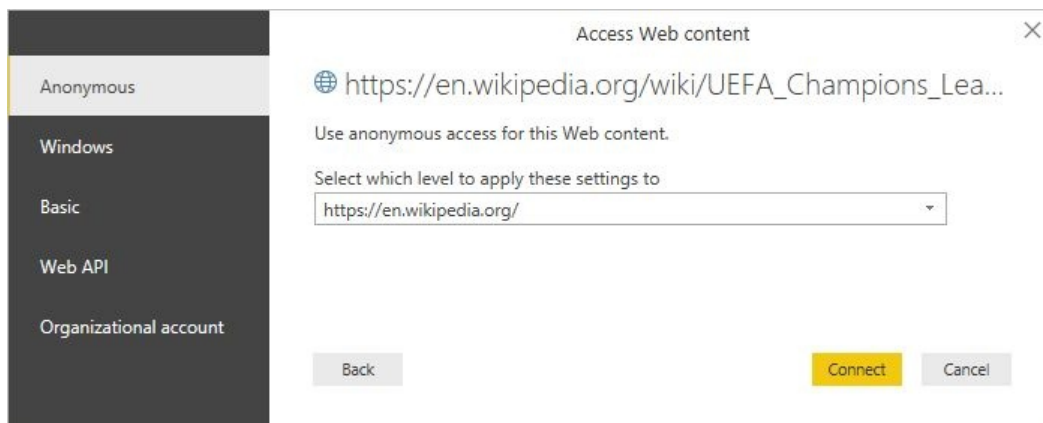


Figure 05-05: Asking us the type access

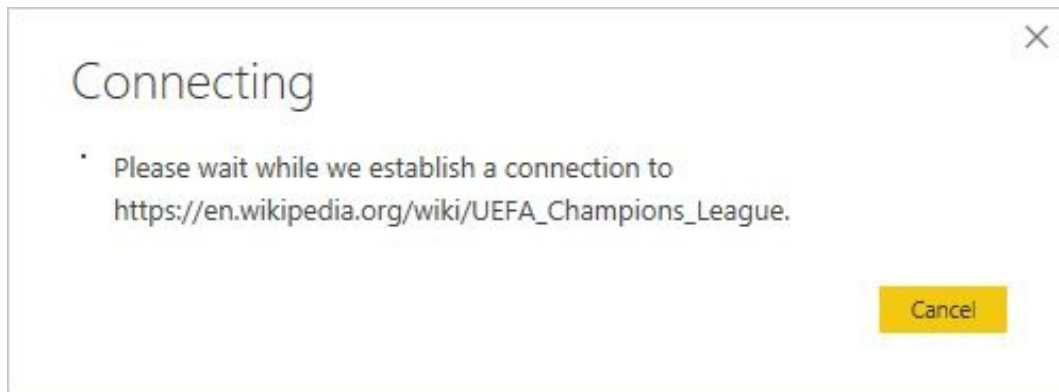


Figure 05-06: Establishing connection to the data source site

After the Connecting process finishes, we'll see the Navigator window showing all tables that could be converted from web data sources.

The left-side panel show the tables names and the right-side panel show the data rows preview. Please select the **Performances in the European Cup and UEFA Champions League by club** table and click OK to continue.

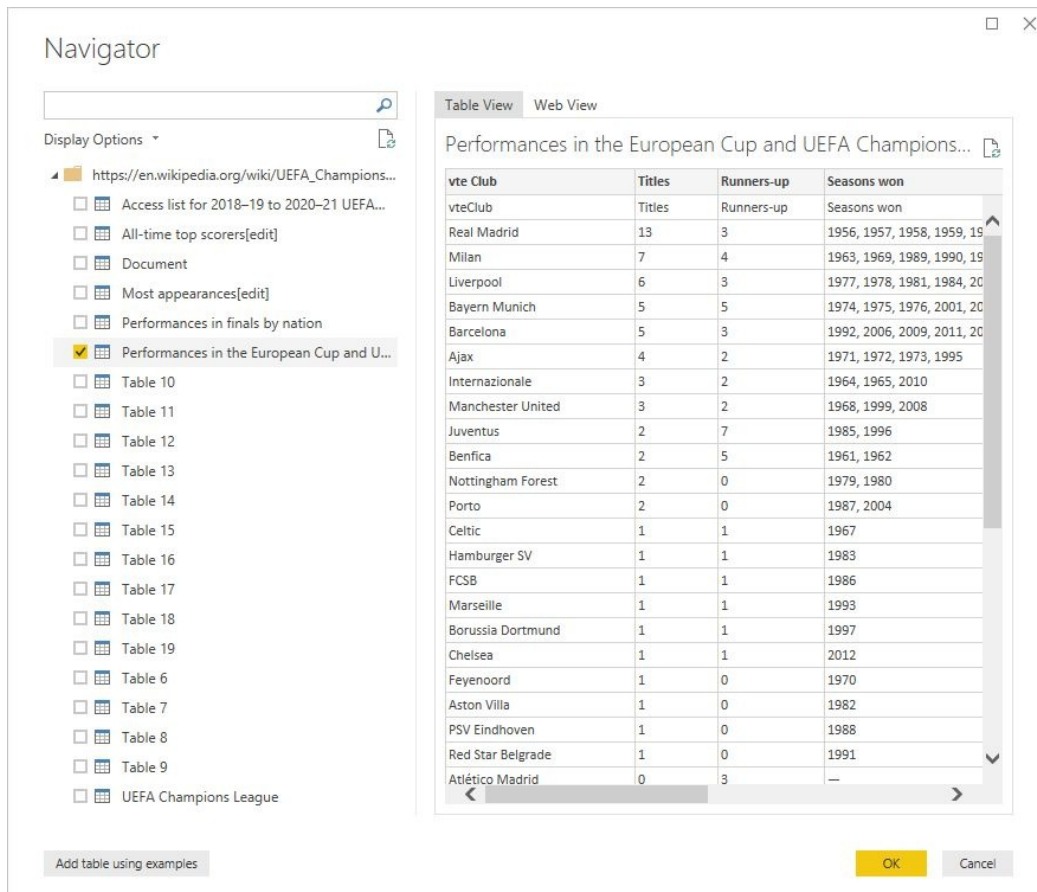


Figure 05-07: Navigator window allow the first look to the data uploaded

After clicking OK, Power Query will connect to and process some of the data, loading it into the Power Query Editor for you to review.

- In the ribbon you can view as the buttons are now active indicating that we can interact with them
- The Queries Pane (on the left side), shows all the queries you have, allowing you to select and view them
- In the center-pane you can view the data imported and available for shaping
- In the right-pane (Query Settings) you have the following windows:
 - Properties: This shows the query's name and allows us to rename and disable the load to report feature, or not include this query in the report refresh
 - Applied steps: is where Power Query shows each transformation that we are applying on the query

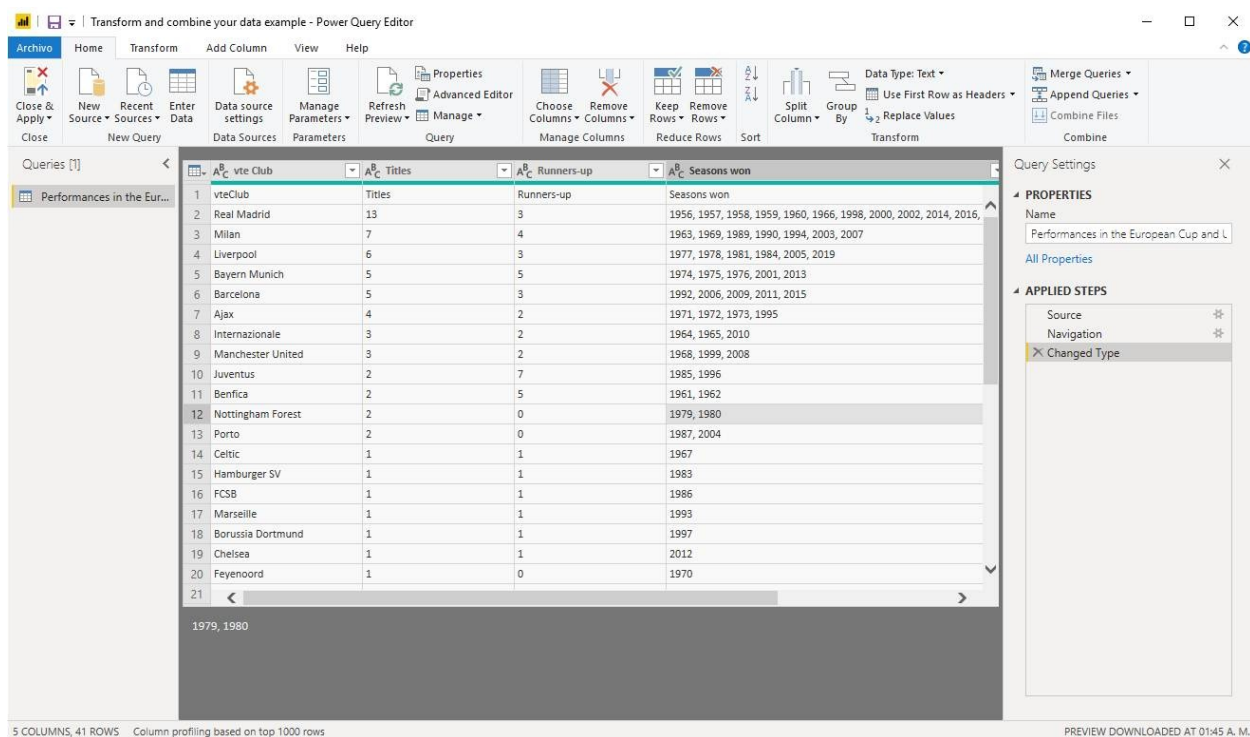


Figure 05-08: Power Query Editor window

Points to make note of:

- Number of columns and rows uploaded
- Preview Downloaded time

This info is very import to can doing a quickly audit on the newly loaded data.

Transform ribbon tab

This tab provides access to data transformation tasks, the most popular being Rename Column, Split Column, Remove Columns, change Data Type or Use first Row as Headers.

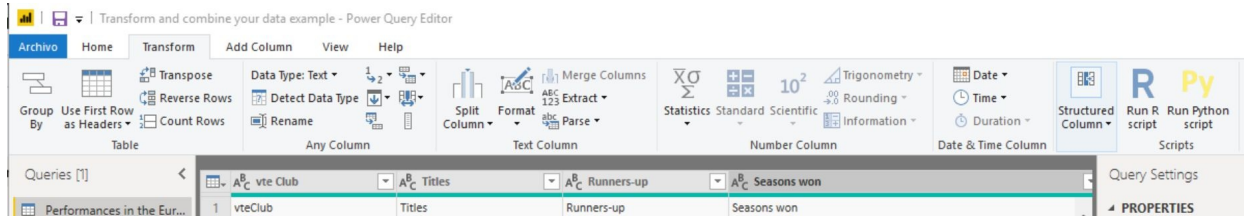


Figure 05-09: Transform ribbon tab, general look

Use first Row as Headers (Promoted Headers task)

One regular issue when we load web data is that the header name shows up as the first data row. To fix this issue we will promote this row as our header.

On the Transform tab click on Use first Row as Headers button appears two options: Use First Row as Headers & Use Headers as First Row, select the first option.

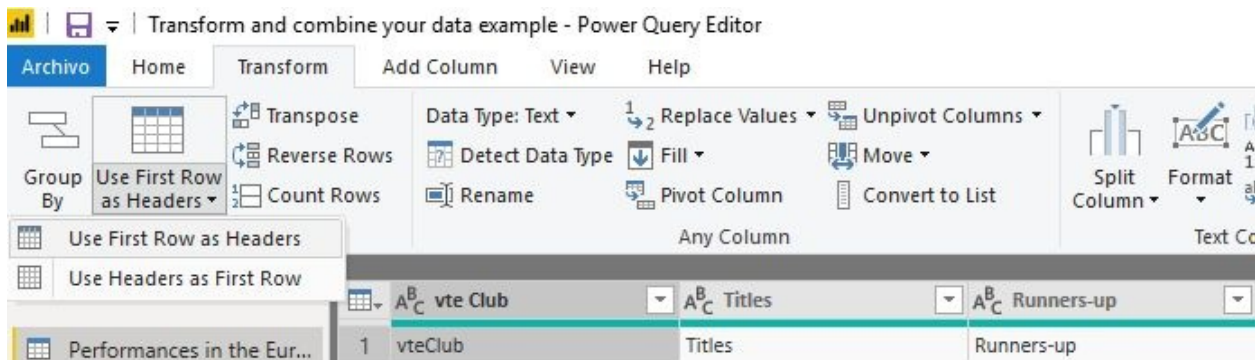


Figure 05-10: Launching the task: Use First Row as Headers

On the Applied Steps pane now, we have two new steps: Promoted Headers & Changed Type 1:

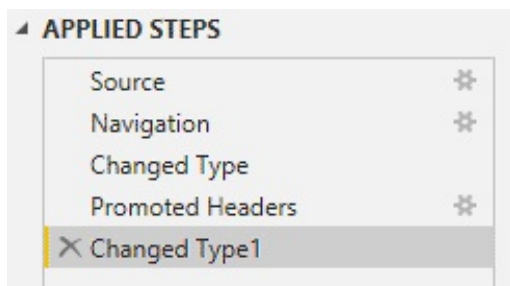


Figure 05-11: In the Query Settings pane, the APPLIED STEPS list the

query step that reshape your data

Remove the Changed Type1 step.

Changing Data Type

Select the Titles column then, on the Transform tab, click on the Data Type button. A popup menu appears. Click on the Whole Number option.

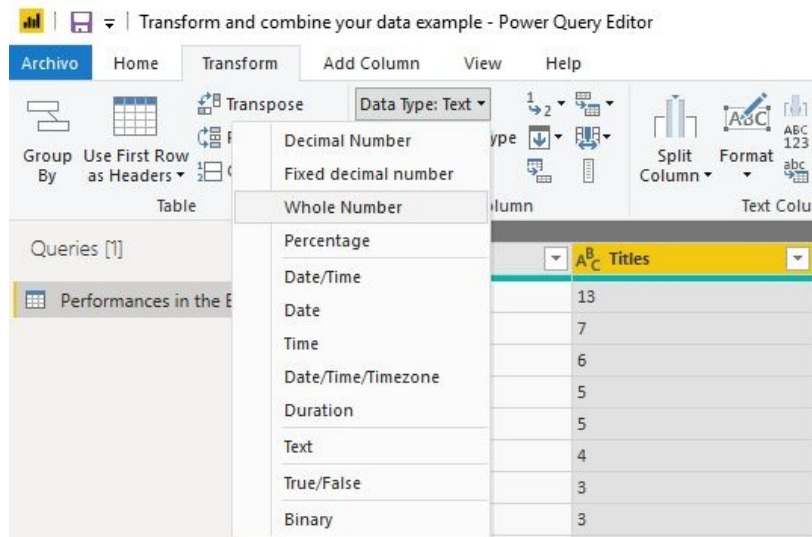


Figure 05-12: Changing data type from char to whole number

Repeat the same steps for Runners-up column and they will then both show as the Number Data Type.

Tips

- One easy way to change data type is using the Detect Data Type button, which will change the data type for you automatically. But please always validate the result and, if it is not right, change manually as explained before.
- Other way is to select the column, mouse right click and click Change Type on the popup menu, then select Whole Number.

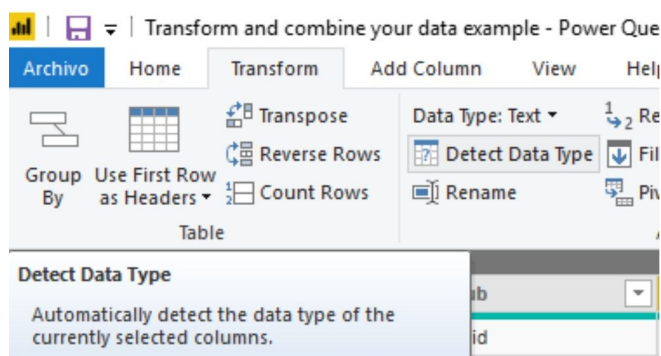


Figure 05-13: Using automatic detection button

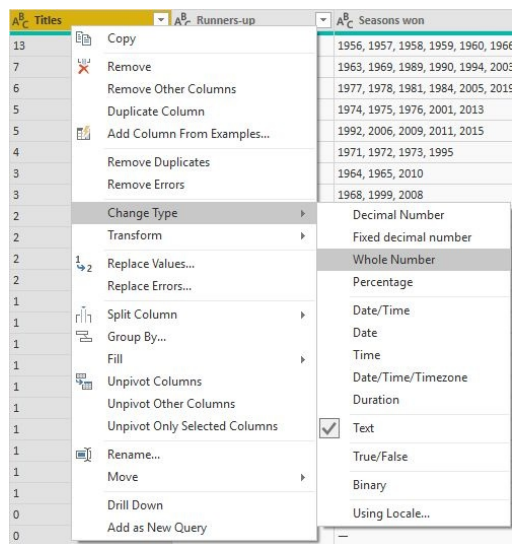


Figure 05-14: Using Change Type, on popup menu, from a column

Add Column tab

In this tab we can add, format or add a custom column with M formulas.

- The Power Query M formula language is a powerful query language, is optimized for building queries that mashup data and is highly flexible, functional and case sensitive similar to F# language

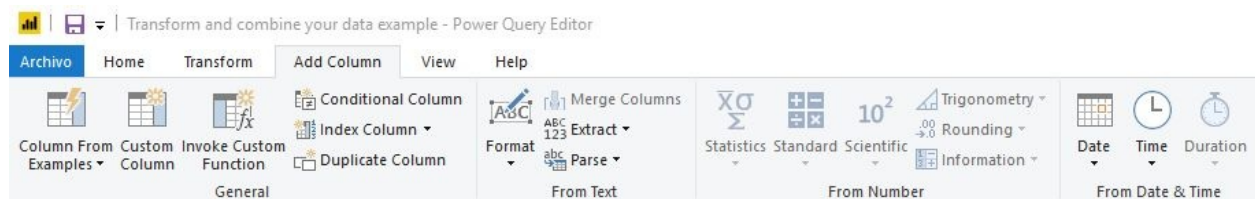


Figure 05-15: Transform Add Column tab, general look

Adding custom column with M Formula

In our example we'll add a custom column named "TotalTitles" and the formula will be [Titles] + [#"Runners-up"]

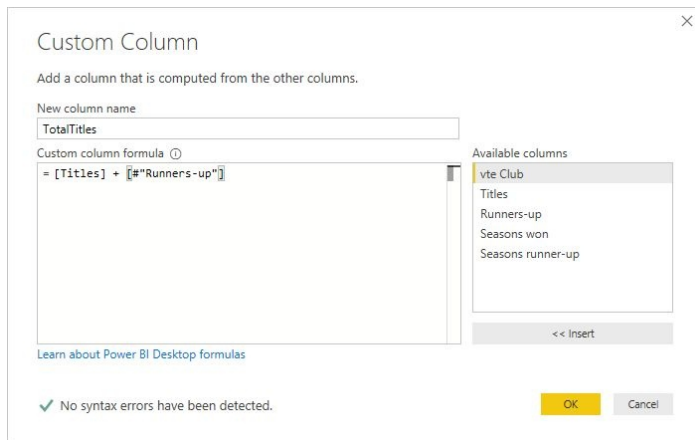


Figure 05-16: Building a Custom column with Power Query M language

After adding the column “TotalTitles”, in the Applied Steps panel you will see a new step named Added Custom:

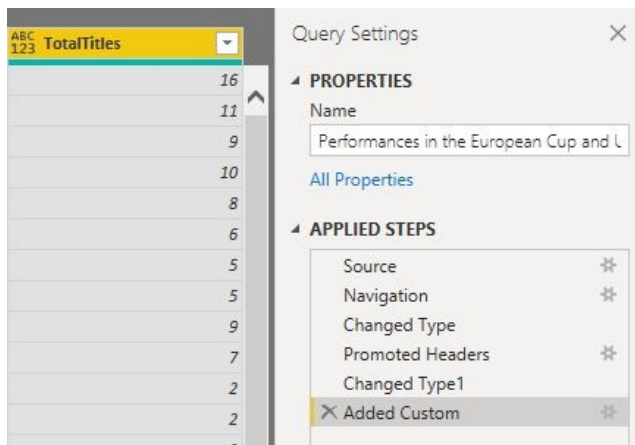


Figure 05-17: Added Custom appears as new query step at the APPLIED STEPS list

View tab

Have you ever wondered how you can get the quality of the data you have uploaded?

- Well, the answer is on the view tab ribbon. (This tab is used to toggle whether panes or windows are displayed or not)

This ribbon is very powerful as you have options to get the distribution, profile or quality of the data showing at the top of your columns. In addition, you can display the Advanced Editor if you want to read the M code that was generated during the button clicks.

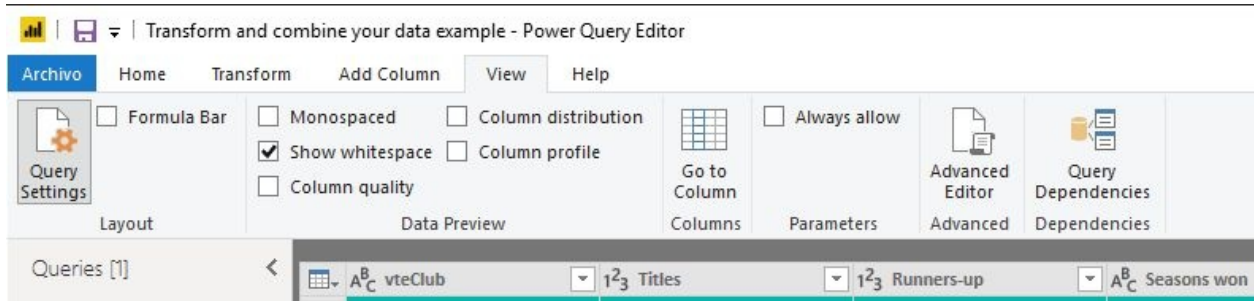


Figure 05-18: View tab, general look

Advanced Editor

On the view tab, click on Advanced Editor button. This launches the Advanced Editor window, where you can edit the query.

Following our example now we will remove the "Seasons runner-up" column.

- In the **let** body
 - type a comma at the line end “Added Custom”
 - type the following code:
 - `#"Removed Columns" = Table.RemoveColumns(#"Added Custom",{"Seasons runner-up"})`
- In the **in** body replace `#"Added Custom"` for `#"Removed Columns"`

You code should look similar to the image below:



Figure 05-19: Advanced Editor with M language

Click the Done button to return the Power Query Editor.

You can now observe two things:

- First, the column "Seasons runner-up" has been removed
- Second, one new step named "Removed Columns" was added in the Applied Steps window

Figure 05-20: Power Query Editor & Query Setting pane

A ^B C Seasons won	ABC 123 TotalTitles
1956, 1957, 1958, 1959, 1960, 1966, 1998, 2000, 2002, 2014, 2016, 20...	16
1963, 1969, 1989, 1990, 1994, 2003, 2007	11
1977, 1978, 1981, 1984, 2005, 2019	9
1974, 1975, 1976, 2001, 2013	10
1992, 2006, 2009, 2011, 2015	8
1971, 1972, 1973, 1995	6
1964, 1965, 2010	5
1968, 1999, 2008	5
1985, 1996	9
1961, 1962	7
1979, 1980	2
1987, 2004	2
1967	2

Tip:

The Advanced Editor is very powerful. You could create new columns, custom columns, remove columns, and more, all based in the Power Query Formula Language (aka the M Language). So, if you are a developer may by you feel more comfortable coding with M language for ETL task, but if you are like my friend Ken Puls a Power Query guy the way easier will be just click buttons.

For a complete reference about M language visit [Power Query M function reference](https://docs.microsoft.com/en-us/powerquery-m/power-query-m-function-reference) site at <https://docs.microsoft.com/en-us/powerquery-m/power-query-m-function-reference>.

Finally, to apply the changes and close Query Editor, change to Home tab and click on Close & Apply button.

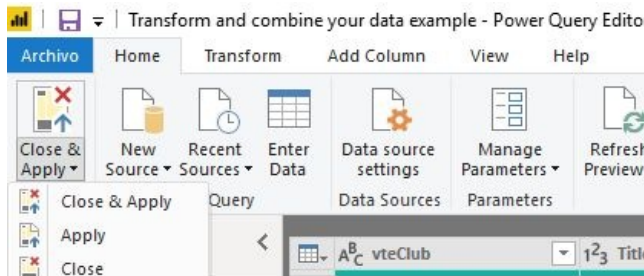


Figure 05-21: Close & Apply button

The transformed dataset will appear in Power BI Desktop, with the table(s) ready to be used for creating reports & dashboards.

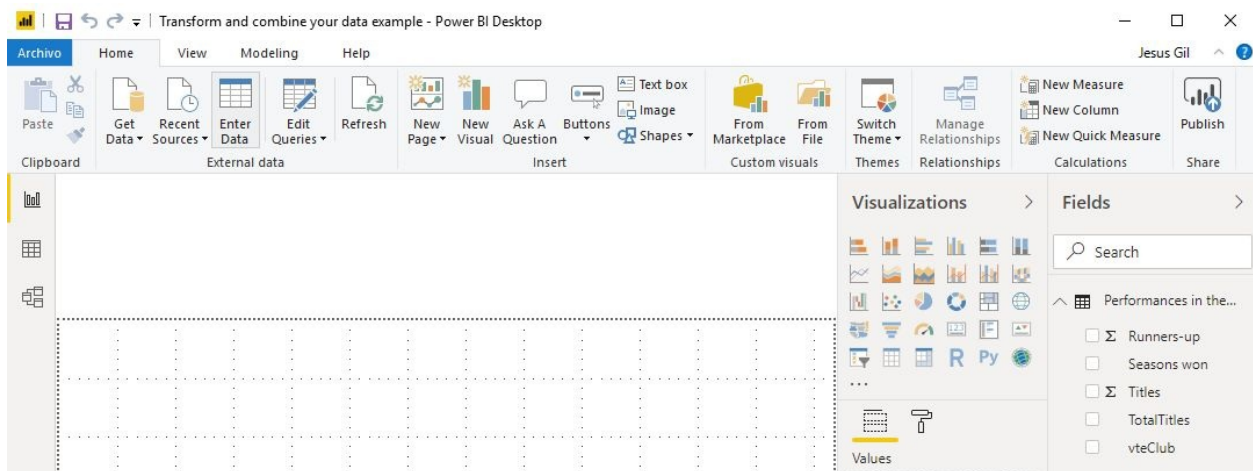


Figure 05-22: Dataset ready to use in the Power BI Desktop

Summary

As we have seen Power Query is a very easy tool to use, but also very powerful at the same time.

You can shape and combine data easily using Power Query Editor or if you are most dedicated to developing you can use the Advanced Editor.

Without a doubt, Power Query has never ceased to amaze us since 6th July 2013 to date.

I invite you to exploit its features and share their experience to help grow the Power BI community around the world.

About the Author



Jesus Gil aka Dr Rudo SQL is a Data Platform MVP since 2010, he is a co-author of SQL Server Upgrade Technical Guide for [2012](#) and [2014](#) versions. Jesus is a regular speaker of SQL Server and Microsoft events around Latin America, he is the SQL Saturday Mexico organizer. Early 2019 Jesus authored 4 courses for Microsoft LATAM talking about SQL Server 2008 & 2008R2 End of Support. His main motivation are: Pily, Monse and Mandy.

Chapter 6: Creating a Shared Dimension in Power BI Using Power Query: Basics and Foundations of Modeling

Author: Reza Rad

Data warehouse professionals understand the concept of shared dimensions and star schema designs always include these types of entities. However, many Power BI users are not coming from a data warehousing background. It is necessary to understand some of the concepts to design a good performing Power BI model. I have written about some of the concepts in an easy-to-understand way in some articles, one of them is this article: [What is a shared dimension, and Why do you need that in your Power BI model?](#) In this chapter, I will explain how the use of shared dimensions can prevent many issues, as well as the need for [both directional relationship](#). If you like to learn more about Power BI, read the [Power BI book from Rookie to Rock Star](#).

Sample Dataset

To follow the example in this chapter, download the custom Excel data source from the code of the book. In this sample dataset, there are three tables as shown below:

Table Inventory: shows the inventory details for each day in each warehouse.

Date	Product	Warehouse	Quantity
1/02/2019	PN 10	Warehouse X	23
2/02/2019	PN 20	Warehouse Y	12
3/02/2019	PN 13	Warehouse Z	43
4/02/2019	PN 50	Warehouse X	1
5/02/2019	PN 34	Warehouse Y	5
6/02/2019	PN 13	Warehouse Z	48
7/02/2019	PN 50	Warehouse X	65
8/02/2019	PN 34	Warehouse Y	91
19/02/2019	PN 13	Warehouse Z	23
10/02/2019	PN 60	Warehouse X	13

Figure 06-01: Inventory Table

Table Sales: shows the sales transactions.

Product Name	Date	Quantity	Revenue
PN 13	1/02/2019	10	1532
PN 12	9/02/2019	24	16541
PN 10	12/02/2019	35	5000
PN 20	4/02/2019	2	261
PN 12	5/02/2019	3	4165
PN 10	6/02/2019	12	123
PN 45	7/02/2019	523	41586
PN 20	8/02/2019	6	123
PN 12	9/02/2019	90	6584
PN 14	10/02/2019	100	321

Figure 06-02: Sales Table

Table Manufacturing: shows summarized information about the cost of producing each product based on date.


 Date	A_C^B Product	1_3^2 Cost
1/02/2019	PN 45	874
2/02/2019	PN 12	465
3/12/2019	PN 10	856
4/02/2019	PN 60	654
5/02/2019	PN 75	31
6/12/2019	PN 34	465
7/02/2019	PN 12	654
8/02/2019	PN 43	9541
9/02/2019	PN 60	123
10/02/2019	PN 50	654

Figure 06-03: Manufacturing Table

Design Challenge

When you load the three tables above into a model, you can see that they are not related to each other.

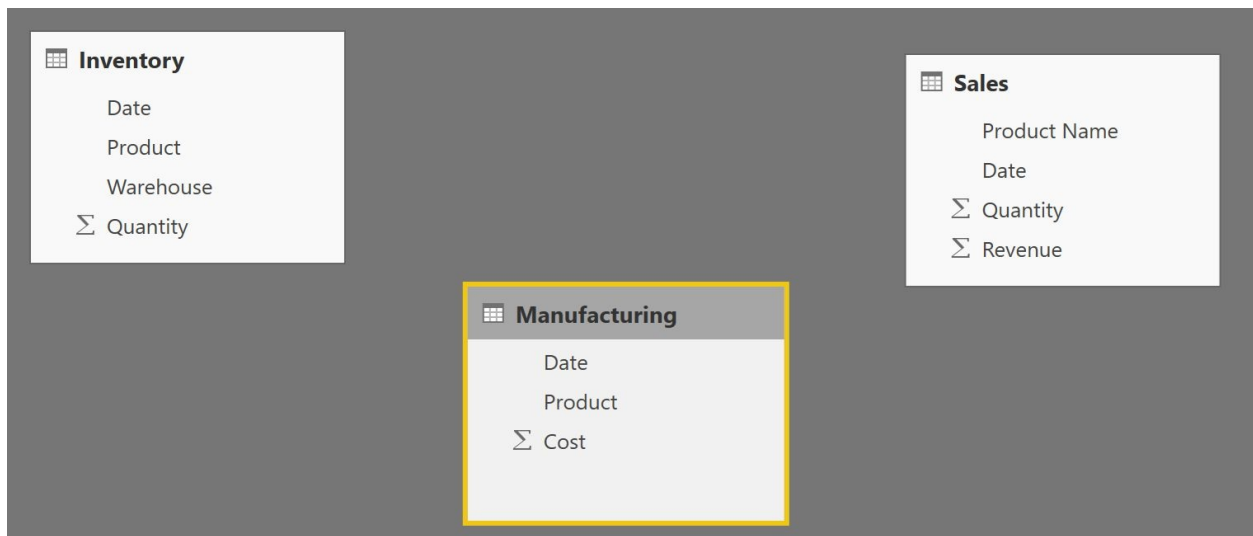


Figure 06-04: Design Challenge of the three tables

Many-to-many Relationship Issue

If you try to create the relationship yourself based on Product, for example, between the two tables Inventory and Manufacturing, you get the message pop up about the need for Many-to-Many relationship!

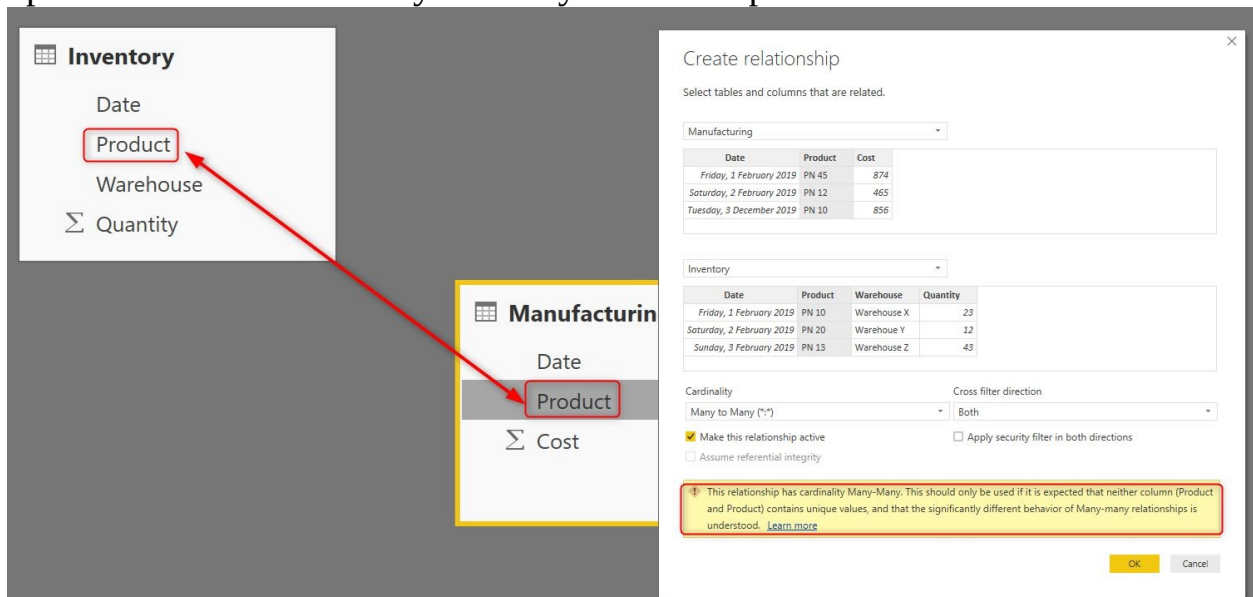


Figure 06-05: Many-to-many relationship Issue

Don't click on the option to create the relationship. A many-to-many relationship is not the best type of relationship to use here. The reason that the relationship

can be only created as a many-to-many relationship is that the Product column isn't unique in any of these tables. This means that no single table can be used as a source of one-to-many relationship.

When neither table in a relationship has unique values for the relationship field, then many-to-many will be suggested. However, I recommend that you don't use that. Read the rest of the chapter to learn how to fix it using a shared dimension.

Both-directional Relationship Issue

Another common issue happens when you have a unique list of values but you still want to slice and dice, based on both tables. Let's say you want to create the relationship between Inventory table and the Manufacturing table but this time based on the Date field.

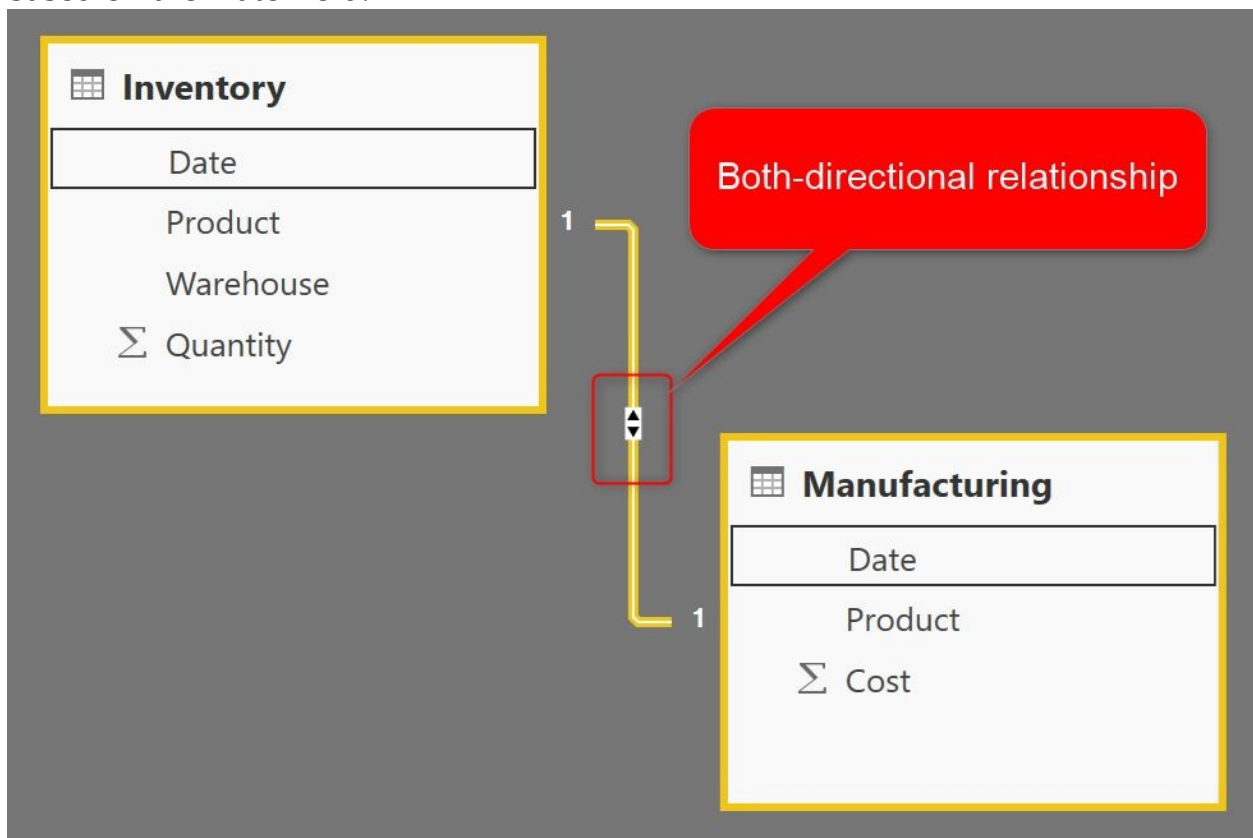


Figure 06-06: Both-directional Relationship Issue

In this scenario, we have unique values for the Date field in both tables; so that the relationship won't be many-to-many. However, because we want to slice and dice Inventory data by dates selected from the Manufacturing table, and vice versa, then the relationship needs to be both-directional.

A both-directional relationship usually happens when you want to use fields

from both tables for slicing and dicing. A both-directional relationship has a significant effect on performance and is not recommended. Read the rest of this chapter to learn how it can be fixed using a shared dimension.

Another issue with the both-directional relationships is that you cannot apply them on all relationships in your model, because it might create circular reference scenarios!

Master List Does Not Exist!

The third issue of design with the three tables above is that there is no master list! There are some products in each table, and we do not have necessarily all products in each table. Or there are some dates in each table, and we do not have necessarily all dates in each table (Power BI will create an auto date dimension which can resolve this issue only for date fields, but what about other fields such as Product?).

To explain the issue, I've created both directional relationships between all three tables to make sure that they are all filtering each other. All the relationships are based on date fields.

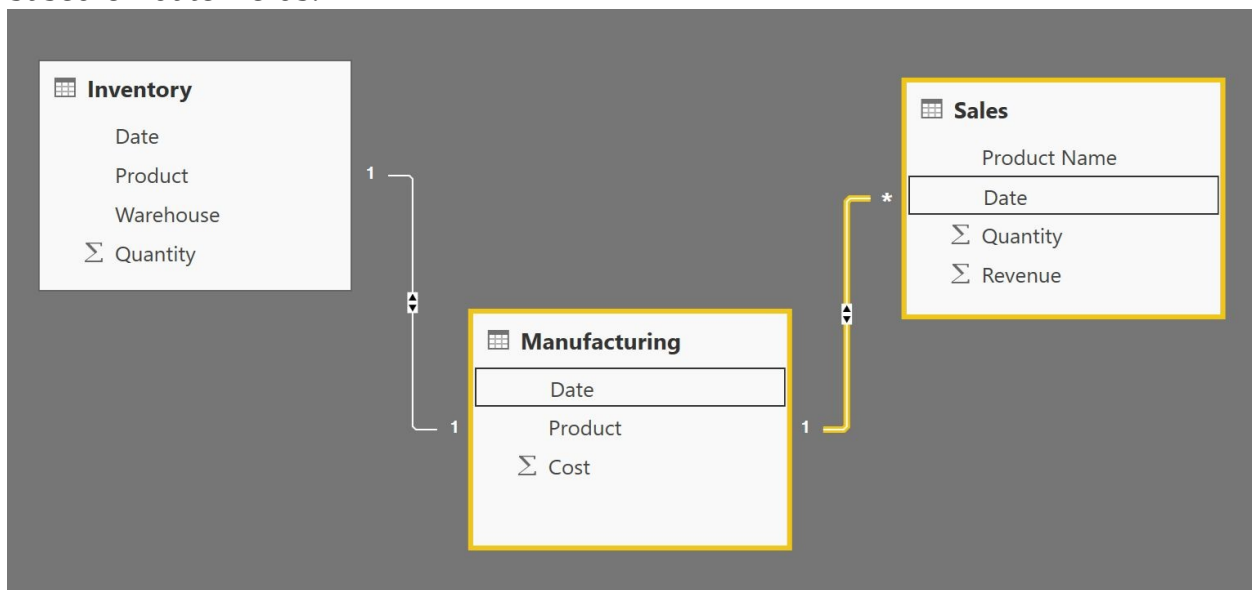


Figure 06-07: All both-directional relationship Issue

Then I created a table with the column Date from the Sales table as a slicer and three table visuals from each table. The date slicer should be able to filter all three tables based on the Date selection;

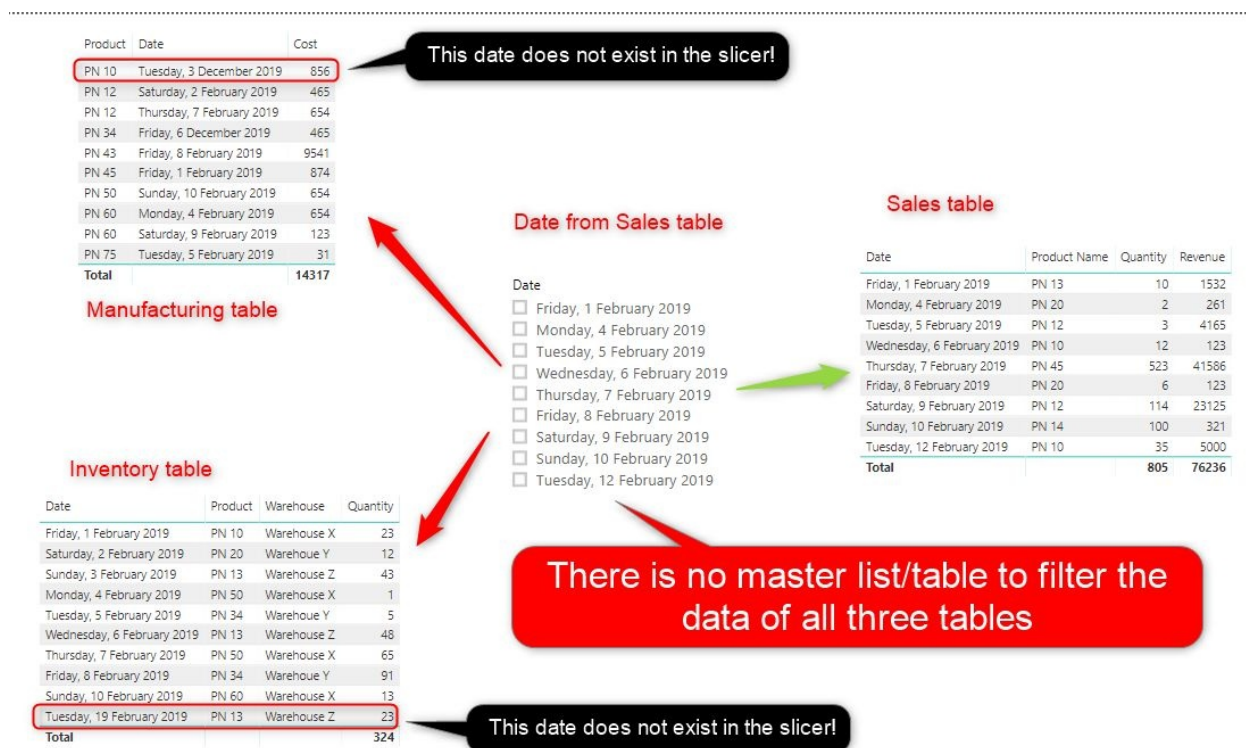


Figure 06-08: There is no master list/table to filter the data of all three tables

If I select a column in the date slicer, it will filter all three tables (because of both-directional relationships). However, if you look closely, the two dates mentioned in the above screenshot and some other dates in the Inventory and Manufacturing tables, don't exist in the slicer. Because the date slicer is coming from the Sales table, and the Sales table doesn't have those dates in it! We will have the same challenge with the Product slicer if we add it.

If you use a column as a slicer which doesn't have all possible values in it, then it cannot show the correct data from all tables. It is not recommended to design it this way, read the rest of the chapter to learn how shared dimension can fix this challenge.

Shared Dimension: Solution

I just mentioned three of the challenges you might have with a design like above. Usually, you don't have just three tables; you will have many more, and you will also have many more challenges with a design such as the one above. The best practice when designing such a model is to create a shared dimension. A shared dimension is a [dimension](#) that is shared between multiple [fact](#) tables. (In formal terminology, this is called a conformed dimension). [In this article](#), I have explained the purpose of fact and dimension tables. However, here is just a short summary:

- A dimension table is a table that has descriptive information, such as Product. Columns from the dimension table usually will be used to slice and dice the data from the fact table.
- A fact table is a table that has numeric and (usually) additive information, such as Sales. Columns from the fact table usually will be used as metrics and measures of our report and sliced and diced by dimension tables.

Well, given the design above, what are our dimensions? They are Date and Product. What are the fact tables? They are Sales, Inventory, and Manufacturing. The challenge is that there is no dimension table. In this case, the Dimensions are being stored as columns inside the fact tables, and this creates inconsistency in design approaches. What you should do is to build the two tables separately. Because the Date and Product tables will be tables that slice and dice all the fact tables and will be related to all fact tables, we call them ***shared dimensions***. Shared dimension is just a dimension which is shared between multiple fact tables.

A design sketch of tables above with shared dimensions would be like this:

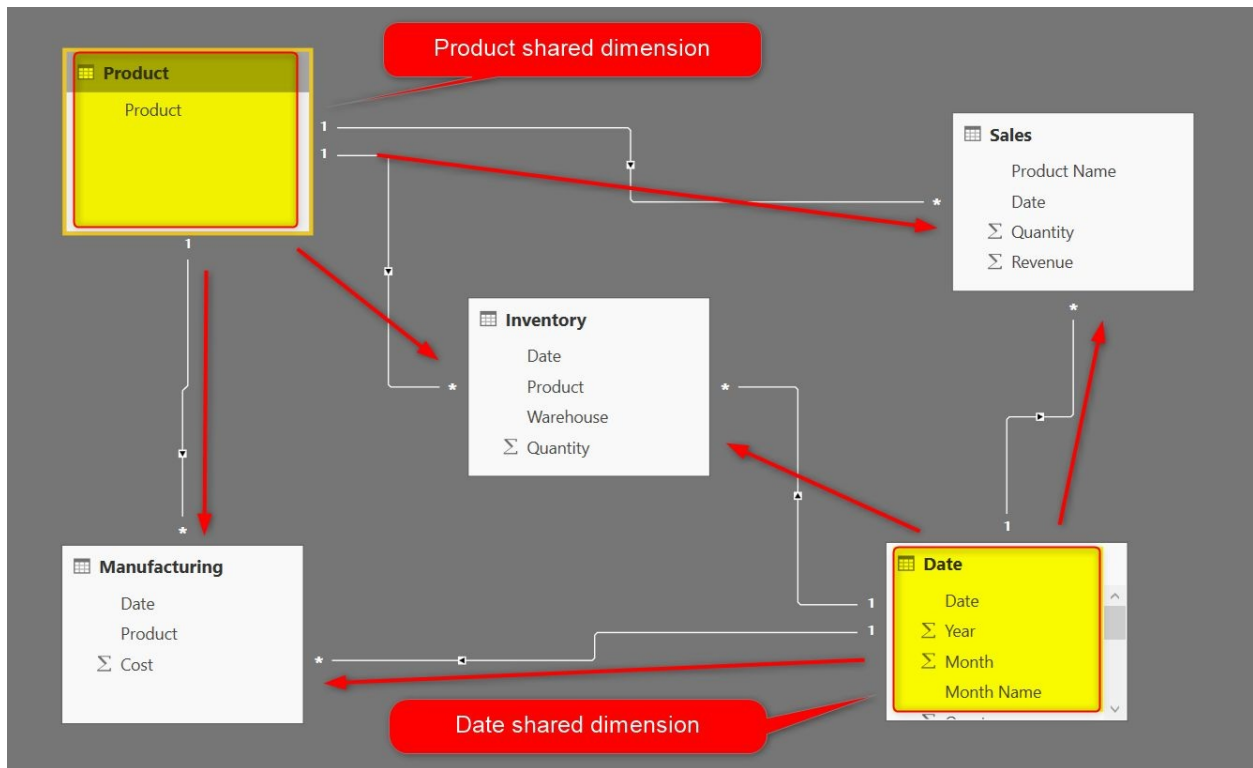


Figure 06-09: Solution Relationship Diagram

Creating Shared Dimension

Now that you know what a shared dimension is and how it can be helpful, let's see how we can add one to our design. You can build the shared dimension in many ways: using DAX calculated tables, using T-SQL (if sourced from database systems), or in Power Query. Because Power Query is applicable regardless of the data source you select, and because the data transformation step is better done in Power Query than in DAX, I am going to show you how to do it in Power Query.

Go to Edit Queries in the Power BI Desktop;

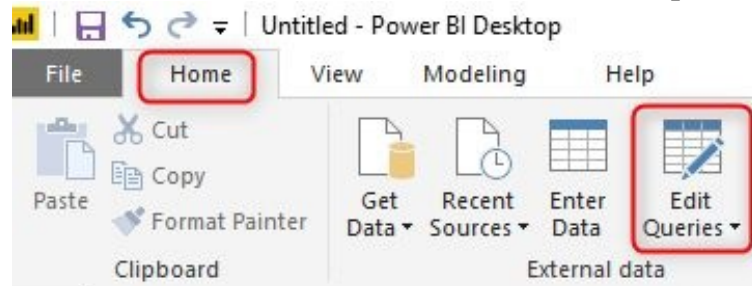


Figure 06-10: Edit Queries

Prepare sub-tables

In the Power Query Editor window, right click on Inventory table, and create a [reference](#) from the query:

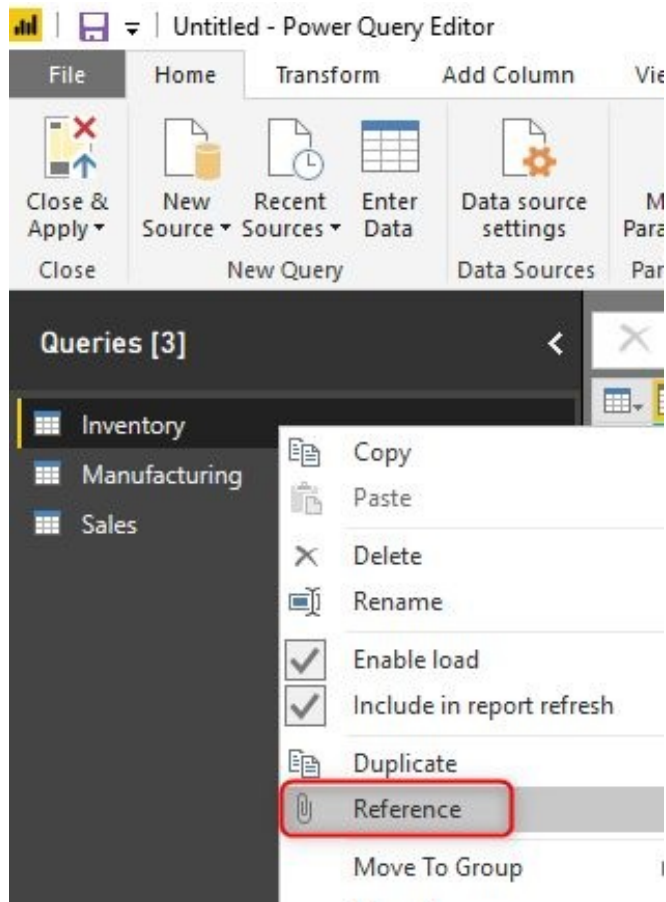


Figure 06-11: Reference from the existing query

If you like to learn more about using the Reference option, [read this article](#). Reference will create a copy of the existing query, with Reference to the existing query, which can now have extra steps in it. In the new query, right click on Product table and remove all other columns.

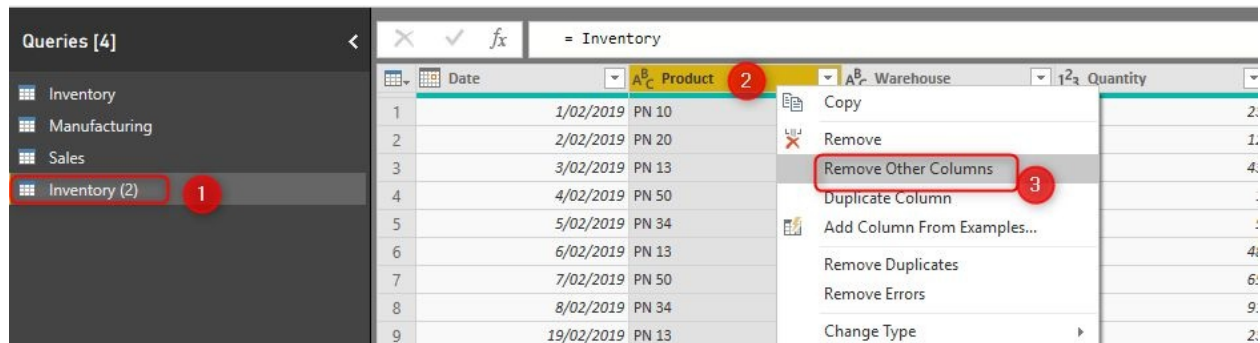


Figure 06-12: Removing Other Columns

The Inventory table should now look like this:

	Product
1	PN 10
2	PN 20
3	PN 13
4	PN 50
5	PN 34
6	PN 13
7	PN 50
8	PN 34
9	PN 13
10	PN 60

Figure 06-13: Inventory(2) Table with only Product Column

Right click on the Inventory (2) table and uncheck the [Enable load](#) option for it. This is to save performance and avoid loading extra tables into the memory of Power BI Desktop. Read more about this option [here](#).

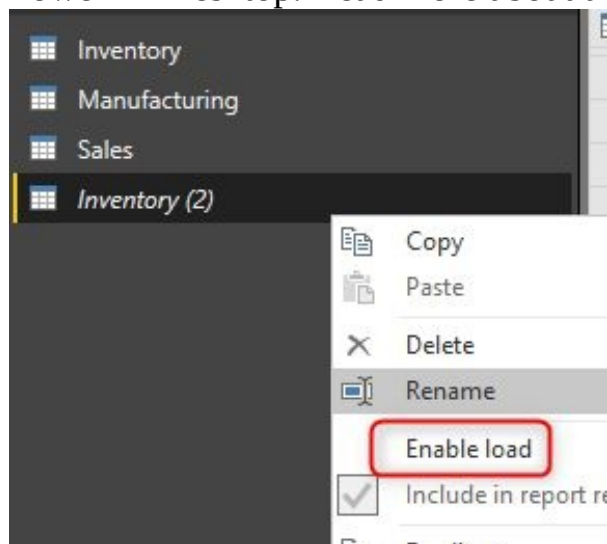


Figure 06-14: Disable Load for the Inventory(2) table

Do the same process now for the other two tables, Manufacturing and Sales:

- › create a reference from each table
- › only keep the Product table and remove other columns
- › uncheck the enable load in the new query

You should now have the new three tables with one Product column only in each:

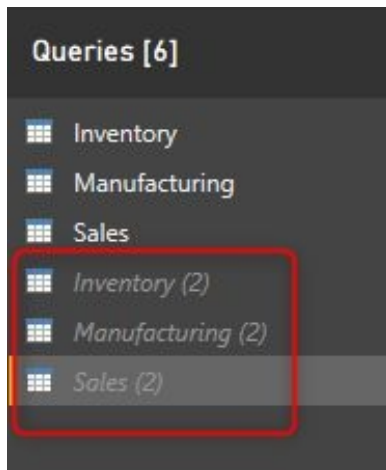


Figure 06-15: Three new tables all disabled load

Set all column names to be the same

The next step is to make sure the column names are the same. Because we are going to append the three tables, if we have different names, it would create extra columns. Names should be an exact match. Importantly, note that Power Query is a case-sensitive language i.e. product is different from Product in the Power Query world. In our sample model, the two tables Inventory and Manufacturing both have the column name Product, but in the Sales table, it is called Product Name, so we need to rename it to Product.

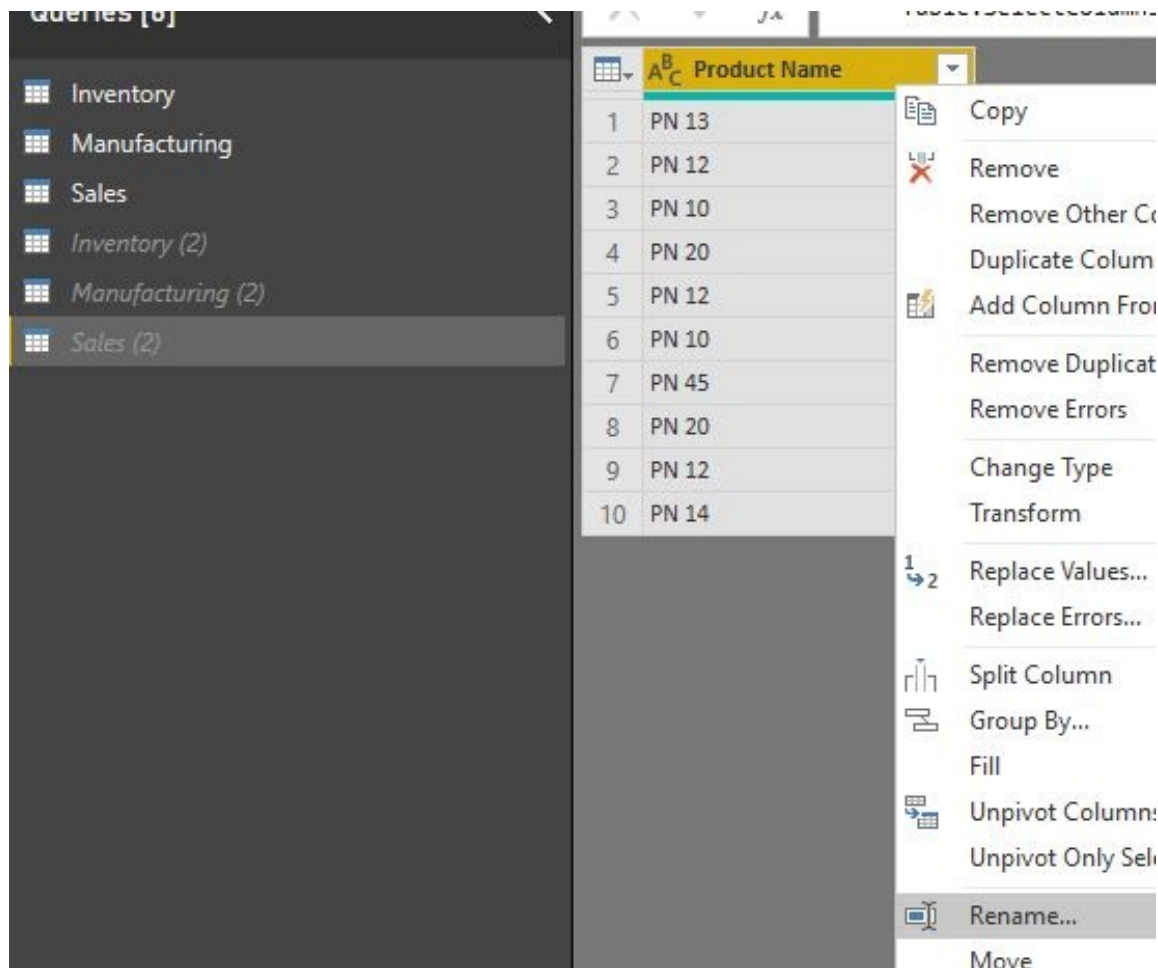


Figure 06-16: Renaming Product Name column in Sales (2) to be Product. All columns should be the same name.

Append all three tables

[Append](#) all three tables together to create one table with all Product values in it. If you like to learn more about Append, read my [article here](#).

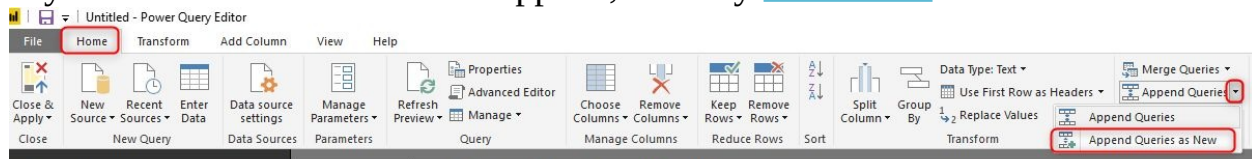


Figure 06-17: Append Queries as New

In the Append command window, select Three or more tables, and all the new tables in it;

Append

☐ Two tables

☒ Three or more tables

1

Available table(s)

Inventory
Manufacturing
Sales

Inventory (2)
Manufacturing (2)
Sales (2)

2

3

Add >>

Tables to append

Inventory (2)
Manufacturing (2)
Sales (2)



4

OK

Cancel

Figure 06-18: Append Three or more tables

The output of the append would be one table including all Product values. You can rename this query to Product.

Queries [7]		= Table.C	
Inventory		A ^B _C Product	
Manufacturing		1	PN 10
Sales		2	PN 20
Inventory (2)		3	PN 13
Manufacturing (2)		4	PN 50
Sales (2)		5	PN 34
Product		6	PN 13
		7	PN 50
		8	PN 34
		9	PN 13
		10	PN 60
		11	PN 45
		12	PN 12
		13	PN 10
		14	PN 60
		15	PN 75
		16	PN 34
		17	PN 12
		18	PN 43
		19	PN 60
		20	PN 50
		21	PN 13
		22	PN 12
		23	PN 10
		24	PN 20
		25	PN 12
		26	PN 10
		27	PN 45
		28	PN 20
		29	PN 12
		30	PN 14

Figure 06-19: Product table appended the result.

Because this is a table you want to load into Power BI Desktop, make sure that the Enable Load of this table is checked. This table is our master list, including all product values. We're nearly done but there are still duplicate values in the table, and they need to be removed.

Remove Duplicates

A dimension should have a unique list of values, so we need to remove duplicates for the key column here:

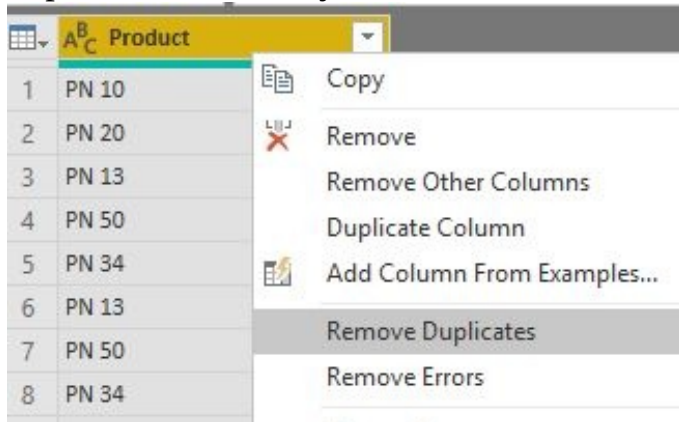


Figure 06-20: Remove Duplicates from the Product table

Before using Remove Duplicates, make sure to [read this article](#) about important tips you need to know before applying Remove Duplicates in Power Query. In a nutshell, because Power Query is case-sensitive, and because spaces can be present at the end of text values, and other special characters can be present, you still might end up with duplicate values, so this is how you would remove duplicates in few steps:

- Clean transformation
- Trim transformation
- Transform to Upper Case
- Remove Duplicates

full details of these steps are written [here](#).

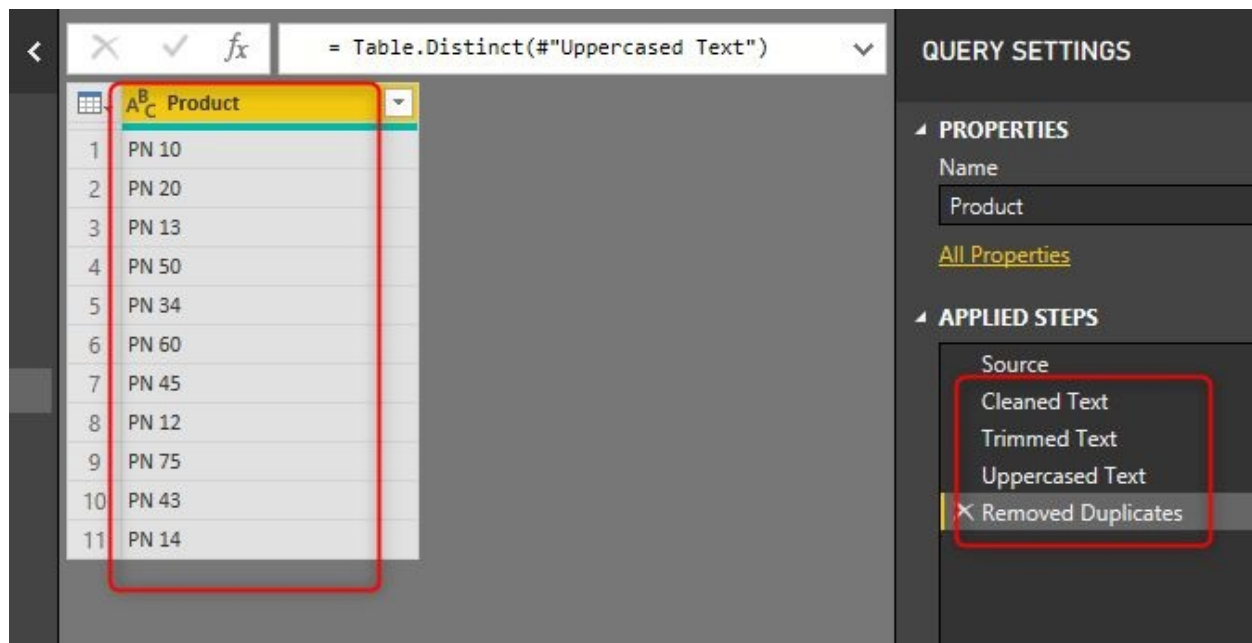


Figure 06-21: Product table with all the steps. Master List created.

Now you have your Product shared dimension ready! Repeat this process for any other shared dimensions with the relevant columns. However, for the Date table, we would do it differently.

Date Dimension

Because the date table is one of the most common tables, and it has one record per day, it really isn't related to which values happen to be present in Sales, Inventory, or other tables. There are general practices of how to create a date table. Here is my explanation about [creating a general purpose date dimension](#) for Power BI using Power Query.

Best Practice Design: Star Schema and Shared Dimensions

After loading the tables above, you can create one-to-many relationship single directional from Product and Date tables to all other fact tables. This is the final design based on our example;

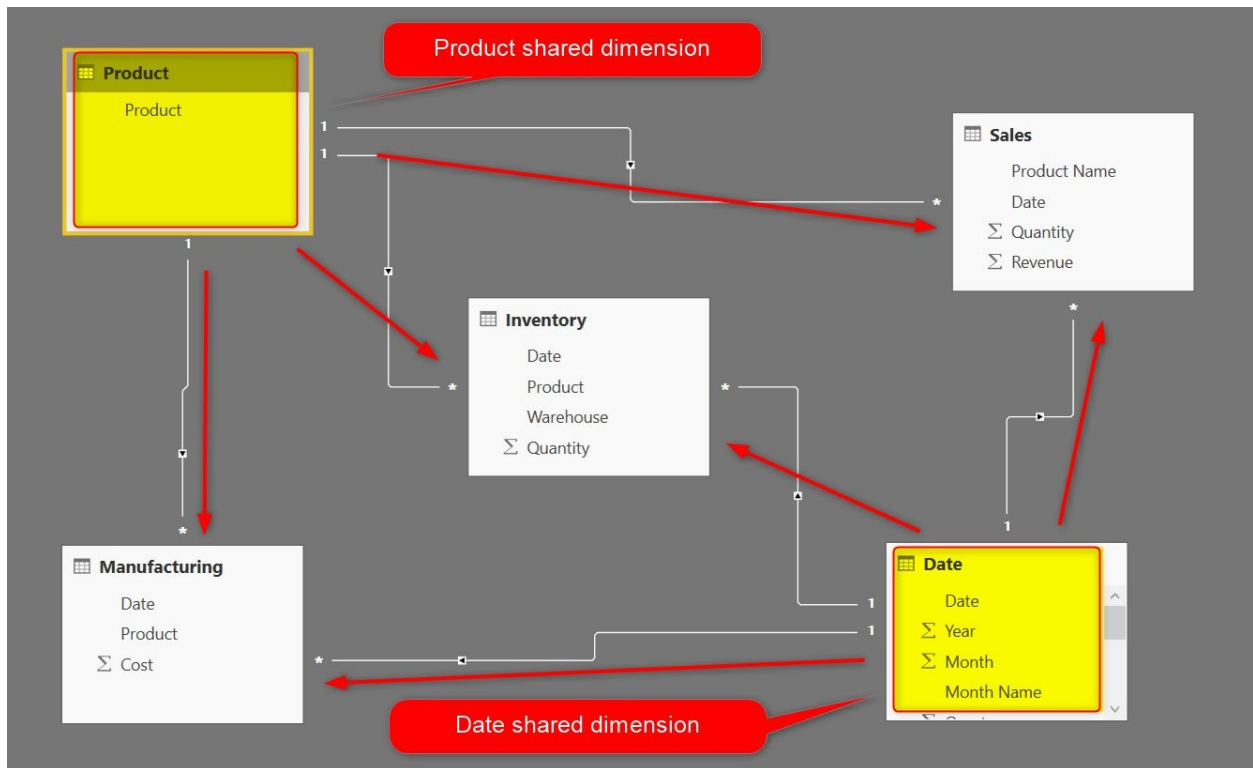


Figure 06-22: Solution Diagram with Date and Product Shared Dimensions

The above design uses two shared dimensions, and avoided all challenges mentioned:

- It doesn't need both-directional relationships
- It doesn't need many-to-many relationships
- Product and Date tables are master tables which can be the source of any slicing and dicing

To make sure that you won't use incorrect columns for slicing and dicing, make sure that you hide Date and Product columns in all the three fact tables as below:

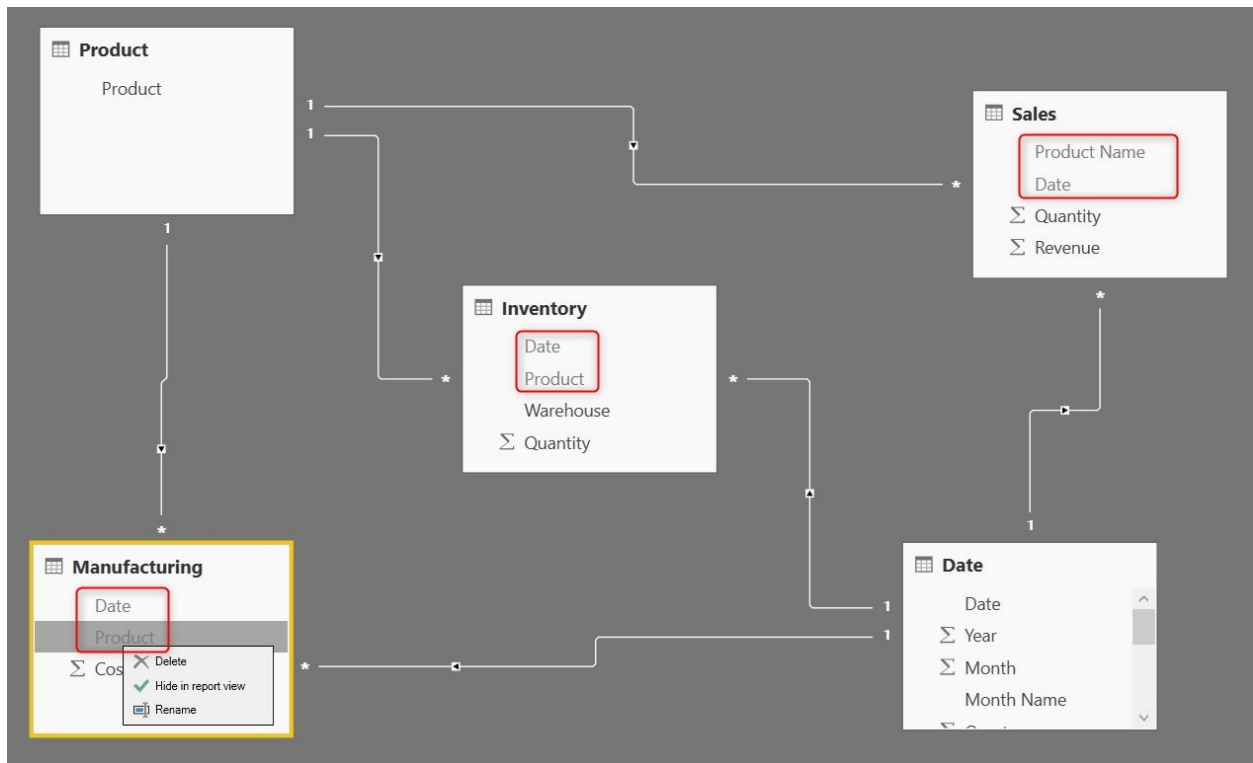


Figure 06-23: Hide Date and Product Column in Non-Dimension tables.

This solution can now provide correct reporting capabilities as below;

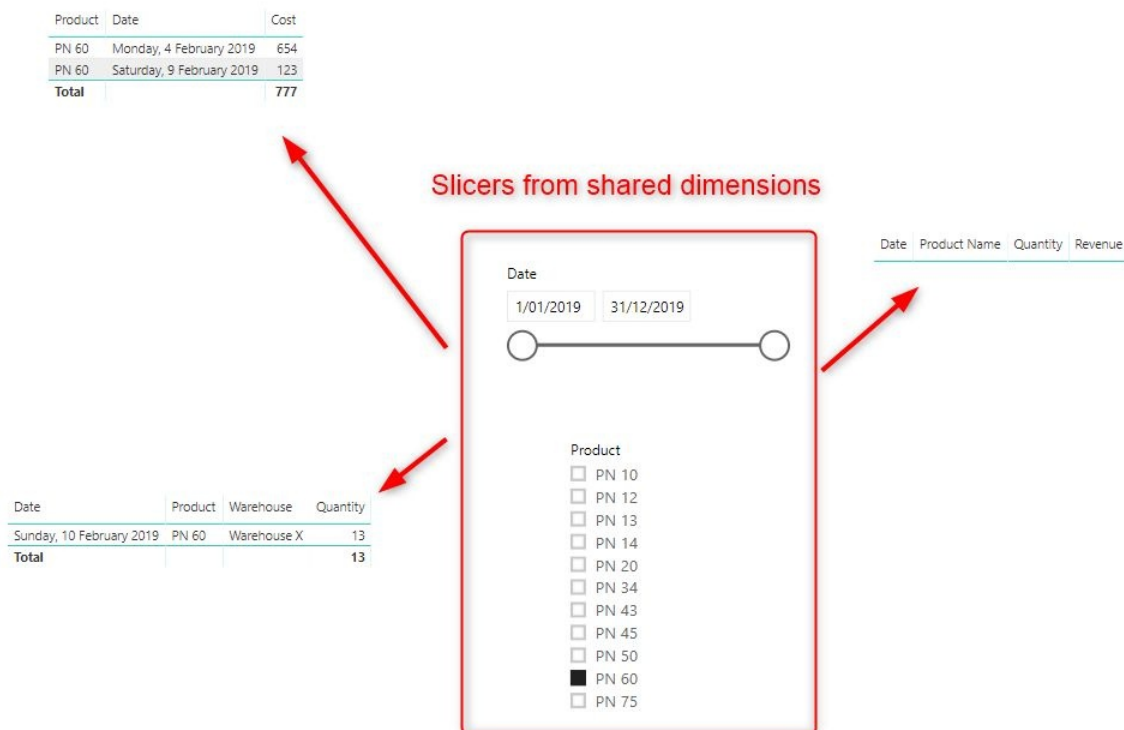


Figure 06-24: Testing the result

Summary

Nothing is worse than a bad data model design. These designs lead to poorly designed relationships that decrease the performance, and lead to a need to write a lot of unnecessary DAX expressions to cover for the poor design. At the end of the day, it performs slowly. In this example, you learned one basic but fundamental concept used in designing Power BI data models. Using a shared dimension in your model will avoid both-directional and many-to-many relationships. You've now seen how easy it is to create such a dimension. This same method can always be used in your Power BI data models.

If you would like to learn more about other basic tips and tricks for data modeling in Power BI, check my other articles at <http://radacad.com>. Here are related articles as a reference to study more:

- [What is the Relationship in Power BI?](#)
- [What is the Cardinality of the Relationship?](#)
- [What is the Direction of the Relationship?](#)
- [Data preparation; First and Foremost Important task](#)
- [What is a Dimension table and why say No to a single big table](#)
- [Basics of Modelling in Power BI: Fact Tables](#)
- [Combining Dimension Tables in Power BI using Power Query; Foundation of Modeling in Power BI](#)
- [Star Schema and How to Build It](#)
- [Build Your First Star Schema Model in Action](#)
- [Budget vs. Actual: Zero Complexity model in Power BI](#)

About the Author



Reza Rad is a member of the [Microsoft Regional Director program](#), an Author, Trainer, Speaker, and Consultant. He has a BSc in Computer engineering, more than 20 years' experience in data analysis, BI, databases, programming, and development, mostly on Microsoft technologies. He has been a [Microsoft Data Platform MVP](#) for eight continuous years (from 2011 till now) for his dedication in Microsoft BI. Reza is an active blogger and co-founder of [RADACAD](#). Reza is also co-founder and co-organizer of [Difinity](#) conference in New Zealand.

His articles on different aspects of technologies, especially on MS BI, can be found on his blog: <https://radacad.com/blog>. He has already written a number of books on MS SQL BI and is currently writing more. He was also an active member on online technical forums such as MSDN and Experts-Exchange, and was a moderator of MSDN SQL Server forums, and is an MCP, MCSE, and MCITP of BI. He is the leader of [the New Zealand Business Intelligence users group](#). He is also the author of very popular book [Power BI from Rookie to Rock Star](#), which is free with more than 1700 pages of content and the [Power BI Pro Architecture](#) published by Apress. He is an International Speaker in Microsoft Ignite, Microsoft Business Applications Summit, Data Insight Summit, PASS Summit, SQL Saturday and SQL, user groups. And he is a Microsoft Certified Trainer. Reza's passion is to help you find the best data solution; he is Data enthusiast.

Chapter 7: Data Modeling with Relational Databases

Author: Thomas LeBlanc

Data Modeling in Power BI is the way to go if you are working with an existing relational database. If not, you probably want to create a dimensional model with the data sources available. This step is an advancement from the single file or flat table. The skills required for this option include an understanding of relationships between data and the abilities to clean up the data before importing. If you do not have foreign keys in the relational database, Power BI will assist with finding relationships. Power BI will also try to select the right data types as well as default aggregations during import. The chapter will conclude with some interacts in the visuals for the data model, like the hierarchy in a dimension.

Data Modeling

Why is Data Modeling important? Power BI is a visualization tool that has an analytical database engine within it. The path forward for Microsoft is to use this engine for most forming of data for a visualization page. If someone is just starting to use Power BI, the usual path is to start importing a flat file or table. Once satisfied with the first Power BI deployment, a new one is created, and the same pattern is used. The developer discovers that the same modeling is being done in different Power BI pbix files. The thought occurs that it would be nice to have one model and connect multiple Power BI pbix files to the same model. This saves time and money.

Microsoft has now started to allow this to happen after deploying the model in a pbix to the Power BI service. The same idea has been available with SQL Server Analysis Service (SSAS). Some users do not realize they have the SSAS option, but it has been around for years with the Tabular Model and a decade or more with a Multidimensional Cube. The VertiPak Engine (Tabular) started in Power Pivot and migrated to Analysis Services before finally being introduced in Power BI.

If the user is not coming from a relational database background, there is a learning path for getting acclimated with Data Modeling. The first idea is to understand the relationships that can be established between data. The data usually comes from a table(s). The relational database may have foreign keys to relate tables automatically. The next idea is the data types. A numeric is different than a text and is different from a date. This lead to the data dimension used for Time Intelligence. There are some advance ideas like row level security and many-to-many relationships, but those have to be learned after the first three.

Relational Database

The diagram in Figure 07-01 will be used throughout this chapter. The underlying database is structured as a Dimensional Model. The fact table is Internet Sales, and related dimensions are Customer, Sales Territory, Date, and Product. This is the star schema of the data model, and it becomes a snowflake when the Subcategory table is related to the Product table with another relationship between Category and Subcategory.

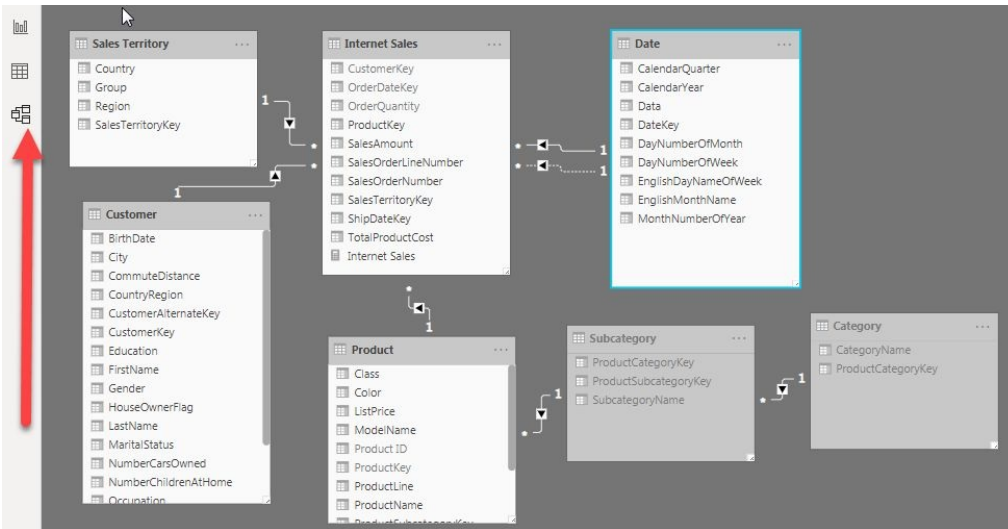


Figure 07-01: Internet Sales Data Mart Relationships

The opposite of this would be a flat table or file with the data columns from Customer, Date, Product, Sales Territory plus the Category/Subcategory data in the same row as the sales numbers. The problem with the flat file is someone will want one more column to be added to the flattened structure, and the data has to be repopulated all over again. The advantage a dimensional model database is the data is already in the tables that have been imported. The hidden aspect just has to be removed later. This is common because the flatten structure probable came from this relational structure as an export.

Tables and Relationships

The first step is to import the tables. Importing is the better option for performance than DirectQuery (Figure 07-02). DirectQuery relies on T-SQL while rendering data and visuals which will require a very fast system. DirectQuery also does not allow relating tables from different sources. Importing will bring the data into the analytical engine of Power BI. Data is structured in a column store architecture with compression. This structure is best for analytical aggregations.

SQL Server database

Server ⓘ
.

Database (optional)
SmallDWMSBI

Data Connectivity mode ⓘ
☒ Import
☐ DirectQuery

▸ Advanced options

Import For Modelling

Figure 07-02: Import for Data Modeling

Figure 07-03 shows the Import Wizard. The Fact Internet Sales is selected along with the related dimension tables. The dimension tables have a surrogate key (integer) column that is used in the fact table for foreign key relationships. This is common in dimensional model databases. In the lower left of the Navigator screen, there is a button labeled “Select Related Tables” to assist with finding tables related to the selected table by a foreign key. In this case, FactInternetSales is related to DimCustomer, DimDate, DimProductSF, and DimSalesTerritory. The table DimSubcategory is related to DimProductSF while DimCategory is related to DimSubcategory.

Navigator

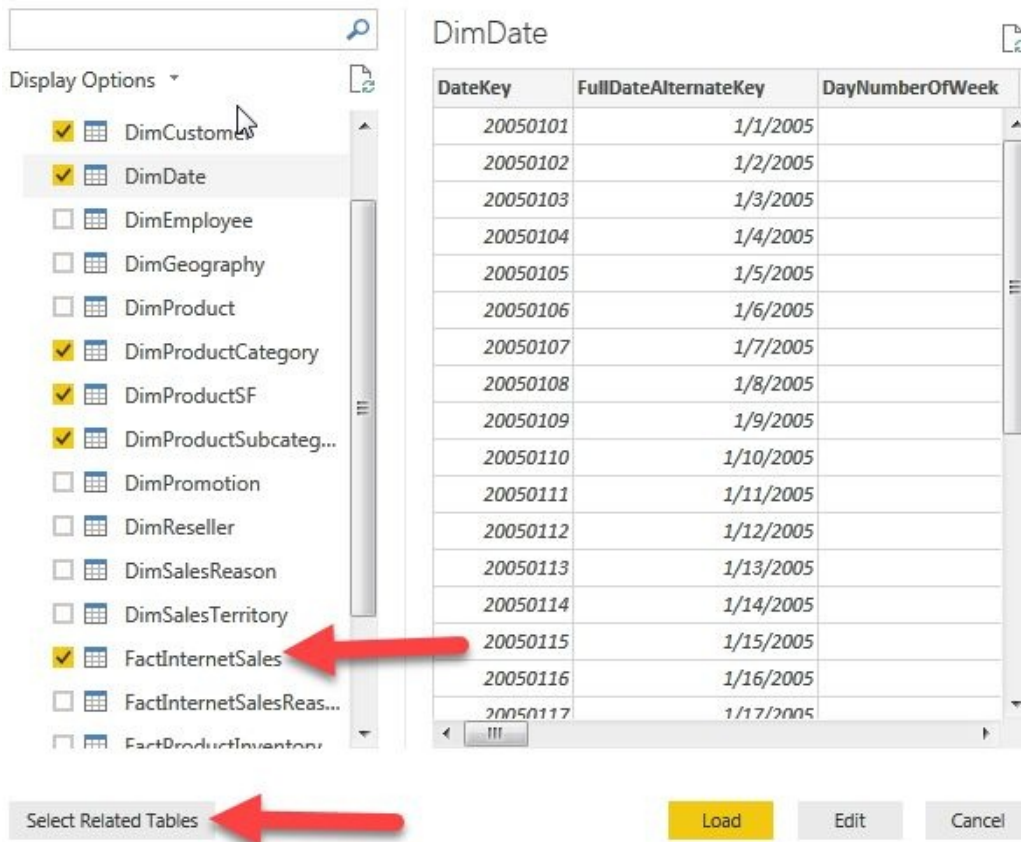


Figure 07-03: Importing Tables

Each table can be renamed after importing. Or, the developer can click the “Edit” button before loading and clean up the selected tables before importing. This edit button will use Power Query for the mashing. Table 10-01 shows the tables imported, relationships, and the columns used for the relationships.

Table	Related Table	Column(s)
FactInternetSales	DimCustomer	CustomerKey
FactInternetSales	DimSalesTerritory	SalesTerritoryKey
FactInternetSales	DimDate	OrderDateKey(DateKey)
FactInternetSales	DimProductSF	ProductKey
DimProductSF	DimSubcategory	SubcategoryKey
DimSubcategory	DimCategory	CategoryKey

Table 10-01: Related Tables and Keys

NOTE: If there is more than one relationship between two tables (Figure 07-04), one has to be selected as active. In this case, the OrderDateKey is the active relationship between FactInternetSales and DimDate. The inactive relationship can be used in DAX formulas, but all slicing and dicing from the columns in the Data table will default for measures as OrderDate.

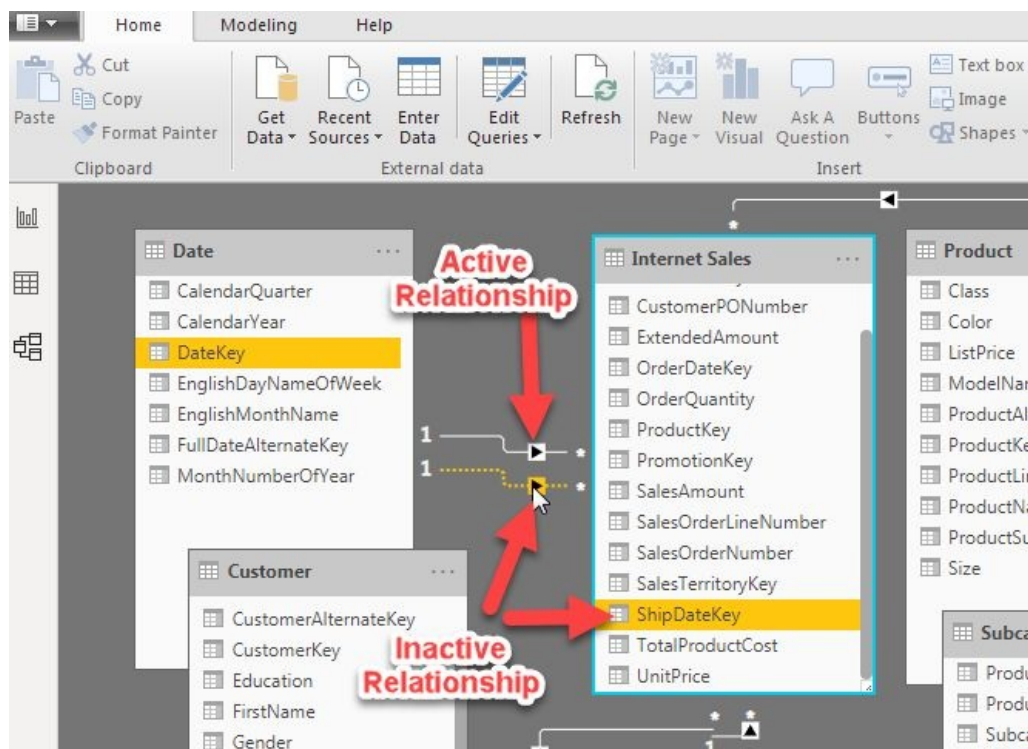


Figure 07-04: Multiple Relationships between Fact and Dimension

The Edit Relationship screen can be accessed by double-clicking the relationship or right-clicking the relationship line and select Edit Relationship... (Figure 07-05) from the menu. Here, the Cardinality and Cross Filter Direction can be changed. All dimension relationships to fact tables are one-to-many. There are very few changes to Cardinality needed unless you are an expert database modeller. There are some cases for one-to-one like a fact for Sales and a related dimension for Sales.

The Cross Filter Direction can be used for filtering between a dimension and a

dimension thru a fact table. The other case would be for many-to-many when there is a bridge table.

Edit relationship

Select tables and columns that are related.

Internet Sales

ProductKey	OrderDateKey	ShipDateKey	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrderNu
477	20130105	20130112	18958	1	4	SO51308
477	20130110	20130117	21469	1	4	SO51404
477	20130111	20130118	11281	1	4	SO51417

Customer

CustomerKey	CustomerAlternateKey	FirstName	LastName	MaritalStatus	Gender	YearlyIncome	T
11024	AW00011024	Russell	Xie	M	M	\$60,000	
11043	AW00011043	Nathan	Simmons	M	M	\$60,000	
11928	AW00011928	Isabella	Morris	M	F	\$60,000	

Cardinality

Many to one (*:1)

Cross filter direction

Single

☒ Make this relationship active

☐ Assume referential integrity

☐ Apply security filter in both directions

OK

Cancel

Figure 07-05: Edit Relationship

The relationships can be modified in another screen called Manage relationships, which can be accessed under the Modeling ribbon while the Data view is selected like Figure 07-06.

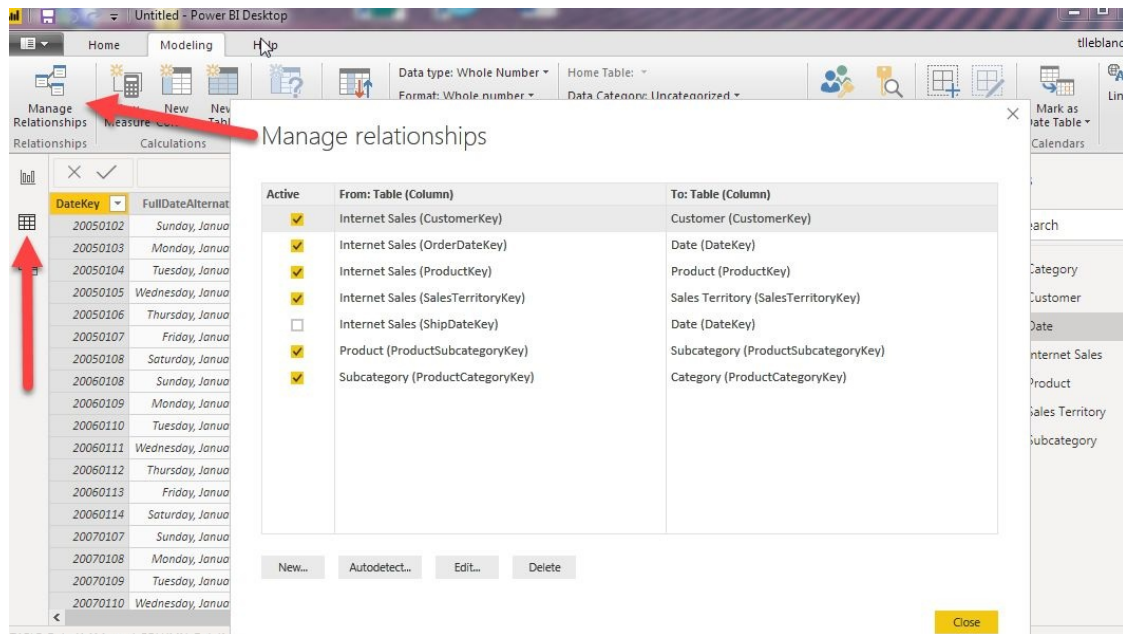


Figure 07-06: Manage relationships

Relationships can be added after the importing of tables. This is done in either the Manage relationship screen or the Model view. The easiest way is to drag and drop the column from the fact table to the dimension table (Figure 07-07).

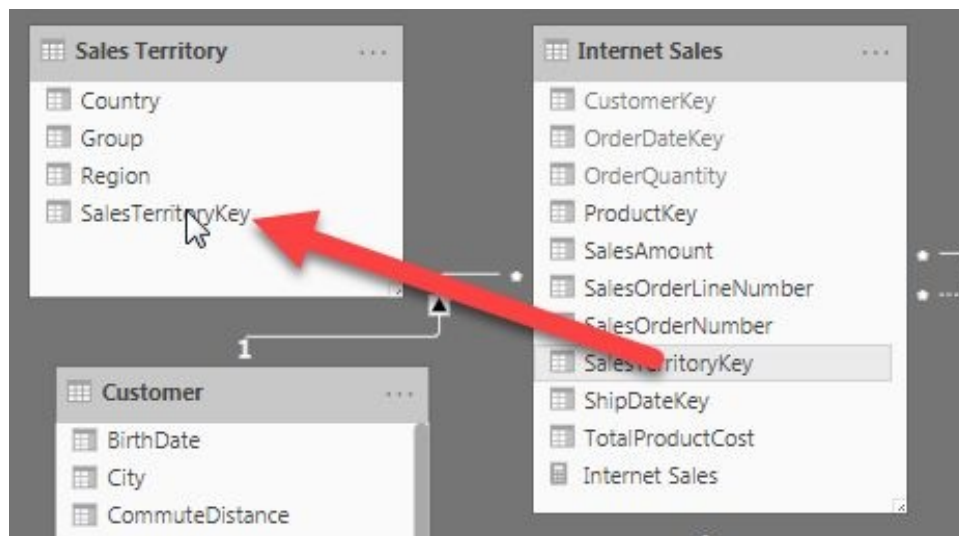


Figure 07-07: Create Sales Territory Relationship

The end result is a stacked bar chart showing a sum of Sales Amount being

sliced by Marital Status from the Customer table while diced by Color from the Product table. This is three different tables related through the modeling — no need to flatten the data in one file or table.

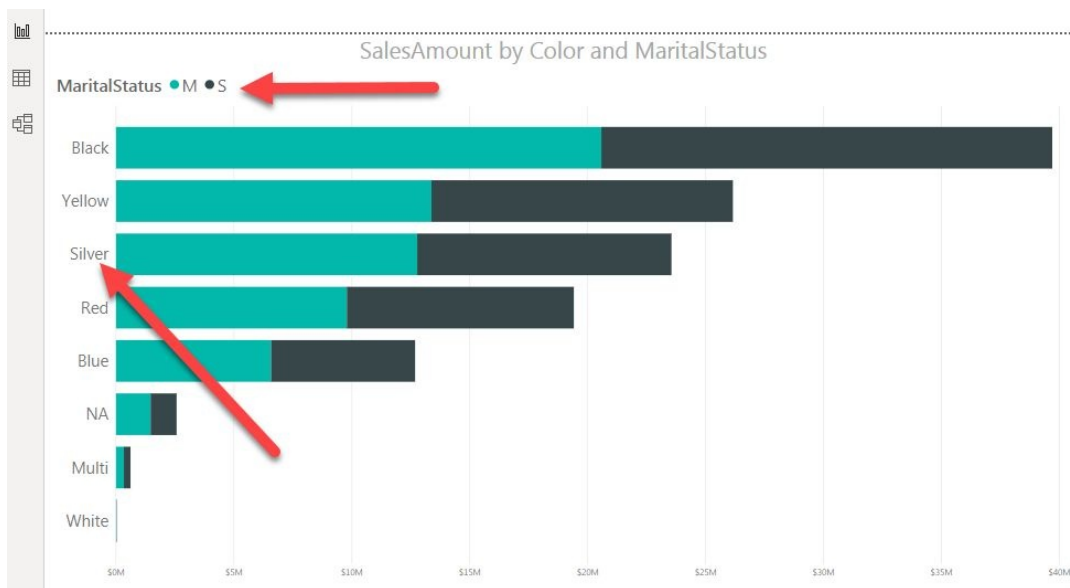


Figure 07-08: Slicing and Dicing Internet Sales Amount

Data Type

The data types pull over well with a relational database. There are a few things you should be aware that can happen. The first is Default Summarize. Next, the category might not be assigned for a column like a geography. The last item would be the assignment of the incorrect numeric data type.

The Default Summarize is assigned to numeric columns. Usually, this is Count for integers and Sum for numeric with decimals. This works fine for a simple summation of Sales. But, it does not do well for the integer columns for relationships. Figure 07-09 shows a default Count on the ShipDateKey in FactInternetSales. This needs to be set to “Don’t Summarize”. This needs to be repeated on all columns not summarized. The new Model view allows to select multiple columns and apply the No Summerization to the select columns.

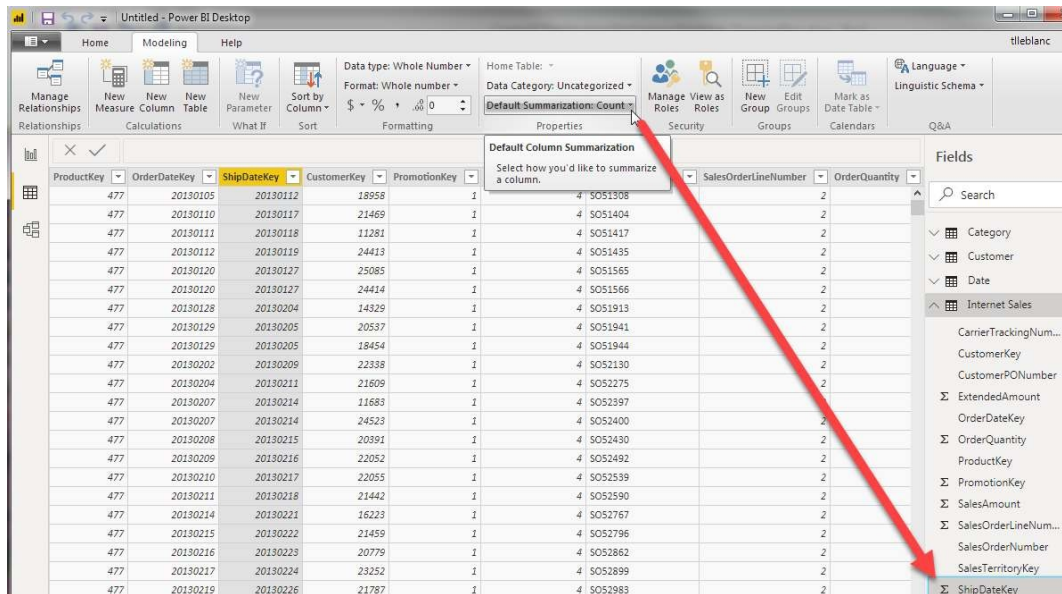


Figure 07-09: Count Default Summarization for Key Columns

Also, the Key columns in the fact and dimension tables should be hidden from the visualization canvas. This can be done in different ways, but the easiest is in the Model view and multi-selecting columns in a table by holding down the control key and clicking on all the columns to hide. Then, right-click and select the Hide in Report view from the menu.

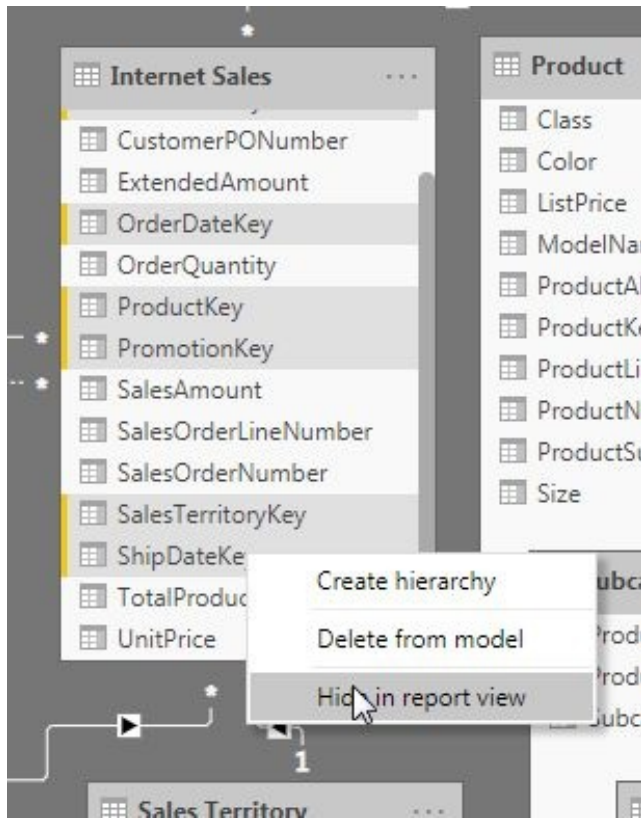


Figure 07-10: Hide Key Columns

Even though the Sales Amount is set as Sum for Default Summarization, this sum would be better managed in a Measure. Use DAX to create a measure by right-clicking on the table (Internet Sales) in Table view and select New Measure from the menu. The DAX would be like Figure 07-11. Then, change the Default Summarization and Hide the SalesAmount column.

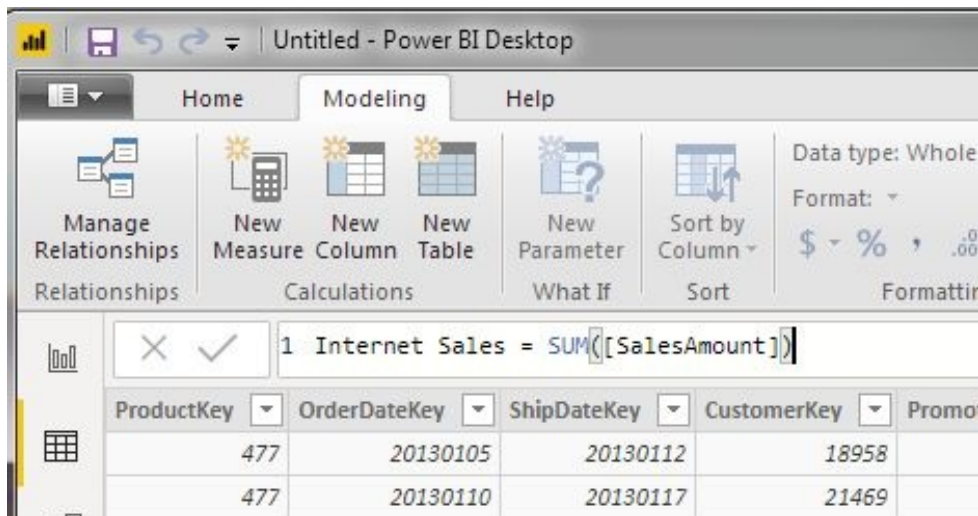


Figure 07-11: Internet Sales Measure

In order for some visualizations to work with a column, the column has to be categorized correctly. This is typical with a map visualization. Unless the visual has a categorized geography column, it cannot locate the place to place a sum or count. Even though the map visual sees a column named Country, it tries to auto match. Here, it is better to assign the Category to this column like Figure 07-12.

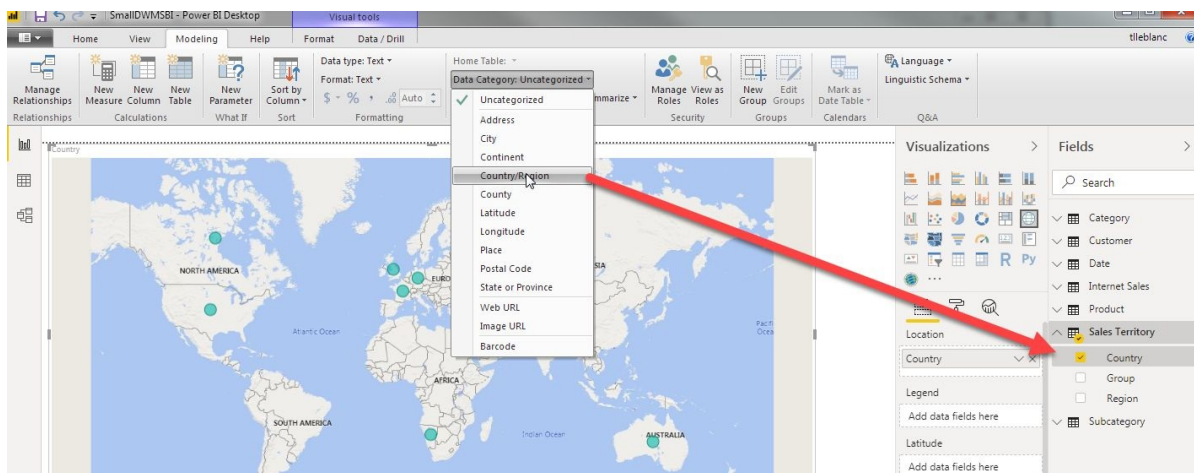


Figure 07-12: Country Category for Map Visual

Numeric Data Types

The last item is about numeric data types. Money data types can be formatted as currency, but data type should be fixed decimal number. Scientific numbers can

use the Decimal Number data type while all integer types should use the Whole Number. Figure 07-13 shows a list of data types in Power BI

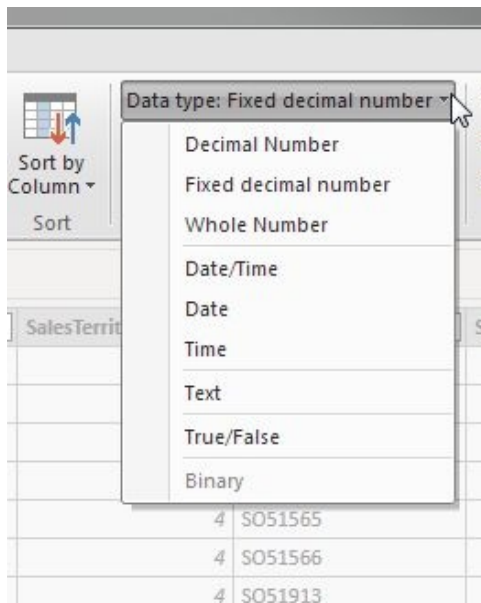


Figure 07-13: Data Types

Remember, there is a separate property for formatting (Figure 07-14). Use this property when making the data look like it falls in visualizations. The data type is for types of visuals, but formatting makes it look pretty or attractive.

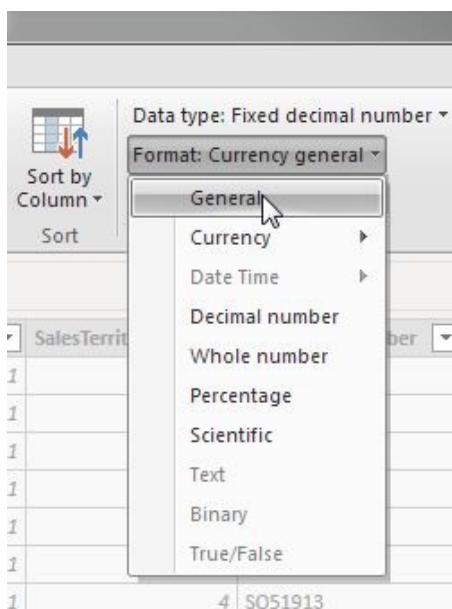


Figure 07-14: Column/Measure Formatting

Additional Fact Table

When modeling, multiple fact tables can be related to the same dimensions. This allows a slicer like Product Category to show a summation for sales related to internet sales as well as reseller sales. Figure 07-15 shows the relationship view with both fact tables in the model.

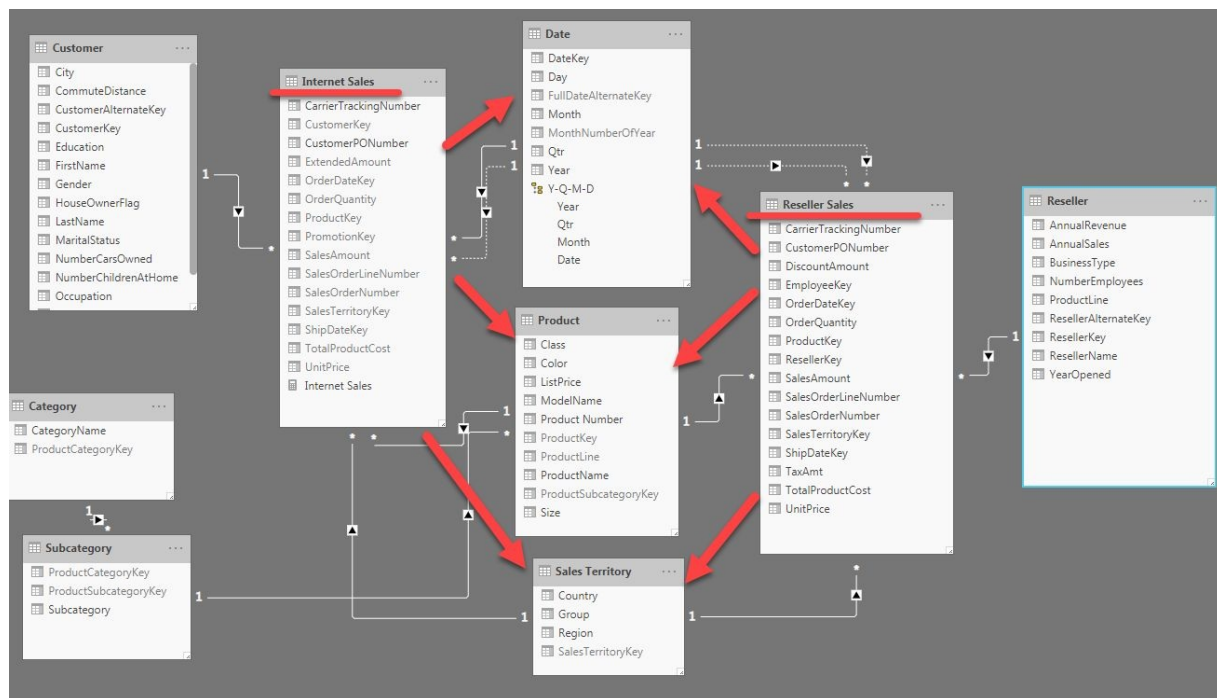


Figure 07-15: Multiple Fact Tables

The measures Reseller Sales and Internet Sales can be viewed by the same visual (Figure 07-16) and sliced or diced by columns in the Product, Sale Territory or Date dimension. Reseller Sales has an active relationship to the date table by OrderDateKey just like the Internet Sales. A measure could also be created to sum these sums together. One caution is that adding a dimension column to a visual from the Reseller table will not render Internet Sales correctly because there is no direct relationship between Reseller and Internet Sales. Only the dimensions with a relationship to both fact tables can be used. In this example, that is Customer, Product (Category and Subcategory) and Date.

Many-to-Many or Bi-Directional Filter

The bi-directional filtering can be used in some Many-To-Many scenarios. One used in this example will use the Sales Reason thru a bridge table to relate multiple possible sales reasons for each sales line item. The New Column modeling feature will be used in the FactInternetSalesReason and FactInternetSales table to concatenate Sales Order Number and Sales Line Item Number. This is in order to have one column for a relationship. It is called SalesNum in the example.

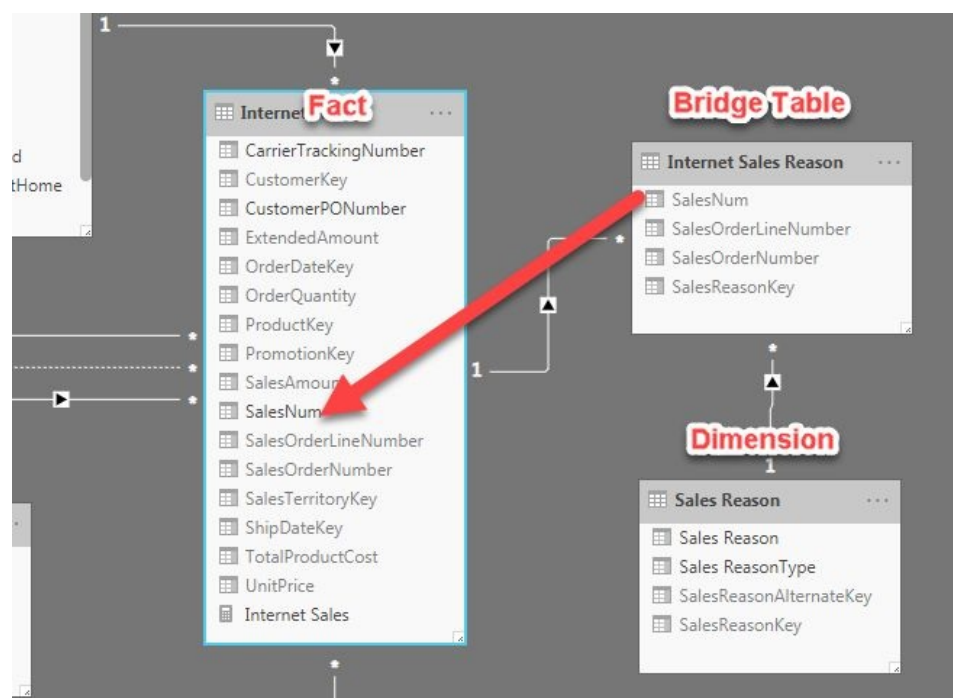


Figure 07-17: Many-To-Many Relationship

The last step would be to change the relationship to Both in the relationship property Directional cross filter. This will enable the Sales Reason table to filter Internet Sales thru the bridge table in Figure 07-17.

Hierarchies

Hierarchies can be modelled for dimension tables. The columns used in a hierarchy have to be in one table. In this example, the Category and Subcategory

are not in the Product table but are related thru keys (snowflake). Adding New Columns to the Product table can be done because the tables are related. The can be done with the RELATED() DAX function like Figure 07-18.

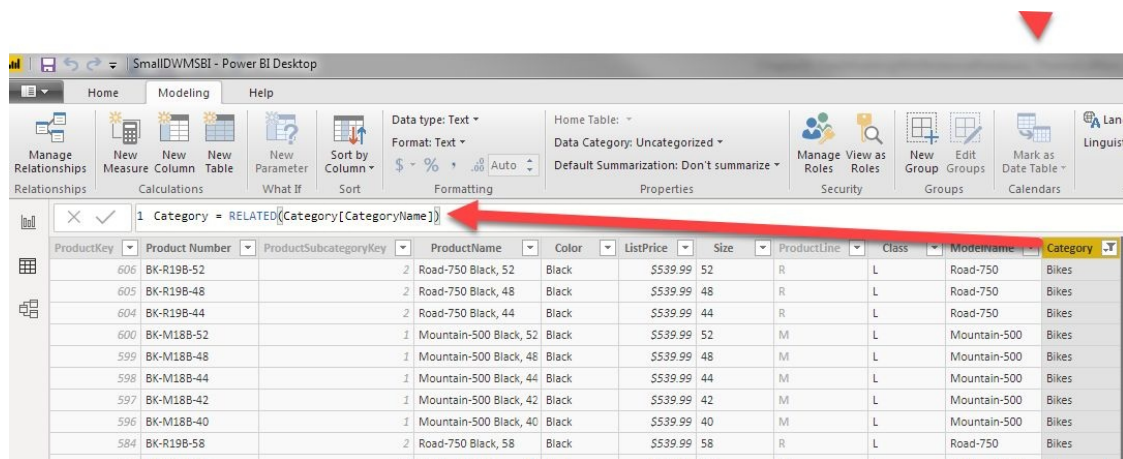


Figure 07-18: RELATED() DAX Function

Another example of a hierarchy is in the Sales Territory table. It is hard for an end user to remember the order of the columns, so creating a Sales Territory hierarchy solves this. Start by right-clicking on the topmost item Group and select New Hierarchy from the menu. Then add Country and Region in that order. The last step might be hiding the individual columns from the report view and leaving just the hierarchy. This hierarchy has been renamed to Sales Territory (Figure 19).



Figure 07-19: Y-Q-M-D and Sales Territory Hierarchy

Additional Comments

Multiple Data Sources

The Data Modeling in Power BI does allow multiple data sources. The tables or files can be related as previously shown only by using Import (cannot use DirectQuery). The tables do have to have a common column of the same data type. There is no support for multiple column relationships in the model. But, you can use Power Query to relate data from multiple columns. The output would be one merged table and not two tables with the relationships shown.

Sort by Different Columns

The ability to sort by a different column comes from the Modeling tab while in

the Data view. The Date table column EnglishMonthName can be sorted by MonthNumberOfYear rather than alphabetically like Figure 07-20. MonthNumberOfYear can be hidden from the report view if not needed for analysis.

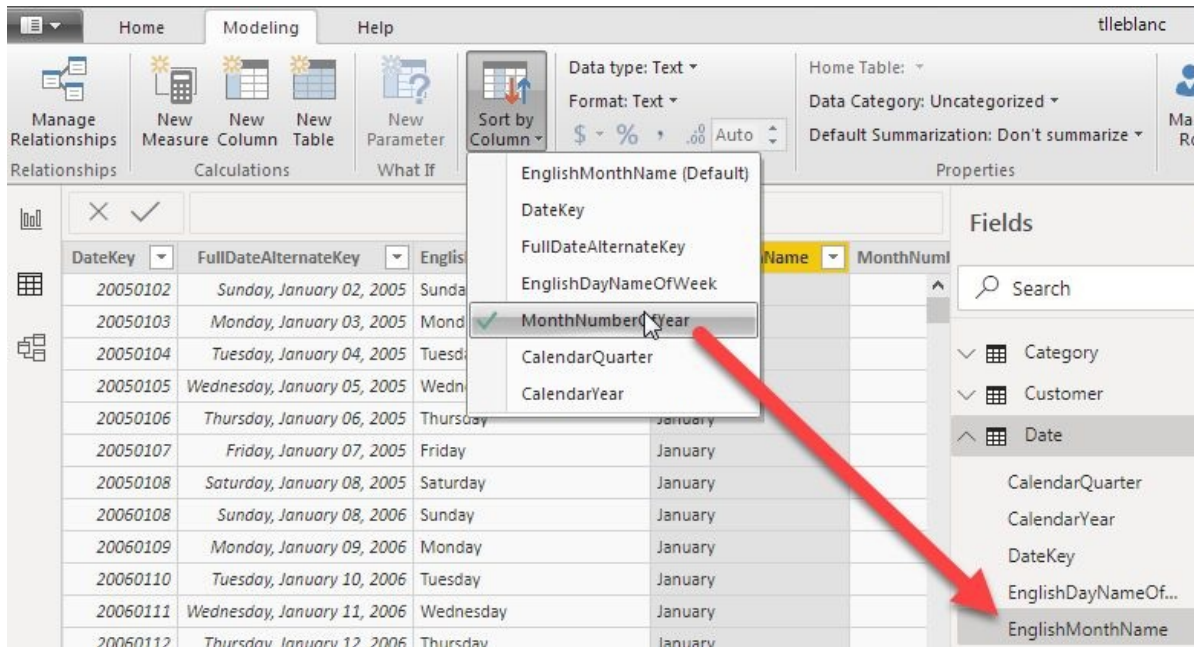


Figure 07-20: Sort By Column

Summary

Data Modeling is a skill that is very valuable for the Power BI Architect or Developer. Once mastered, it can be used in Power BI as well as Analysis Services and PowerPivot. The relational database design as a star/snowflake schema is best for modeling. Using a normalized database is fine; there are just more items to manage. Remember, proper formatting of data types leads to a cleaner design. The auto summations might not be what is needed for integer columns that are surrogate keys for relationships. The one-to-many relationship is the most common for a dimension table's join to a fact table. The many-to-many have to be occasionally used but use the bi-directional filtering sparingly. Loading the model to the service for use with other Power BI pbix files is the ultimate goal.

About the Author



Thomas LeBlanc (Microsoft Data Platform MVP) is a Data Warehouse and Business Intelligence Architect in Baton Rouge, LA and uses his 30+ years in IT to develop OLTP systems with normalized databases for high-performing T-SQL while migrating data for reporting to dimensional data marts. He is an expert in SSIS, SSAS, SSRS, Power BI, and Excel. As a PASS volunteer, he is past chair of Excel BI and Data Arch VCs and is active in the Baton Rouge Analytics and SQL Server User groups as well as SQLSaturday in Baton Rouge. Speak conferences include PASS Summit, VSLive, Live360, and SQLSaturday. Blogs - TheSmilingDBA.BlogSpot.com and Thomas-LeBlanc.com

Part III: DAX and Calculations in Power BI

Chapter 8: Up and Running with DAX as Quick as Possible

Author: Ike Ellis

In this chapter we'll get you started with the 80% of DAX that you need to know. You'll learn how to use measures and calculated columns to add amazing value to your Power BI reports. You'll start easy with SUM and AVERAGE and then finish with the harder topics like CALCULATE, Time Intelligence, FILTER, ALL, variables, debugging, and troubleshooting.

Introduction



Figure 08-01: Answering an interview question

Imagine sitting across from these people in a job interview being asked “Do you know DAX?” If you want to answer yes to that question, this is the chapter for you. What would you have to know in order to answer, “Yes, I know DAX”?

You need to know these functions: SUM, AVERAGE, MIN, MAX, COUNT, COUNTROWS, CALCULATE, and VARIABLES.

You need to know about the Row Context and the Filter Context.

You would need to learn best practices related to formatting, white space, time intelligence, X vs nonX functions (SUM vs SUMX), DAX Studio, and basic troubleshooting.

This chapter will get you started on this journey and will make other resources, like the online documentation, much easier to read.

Your First DAX Expression

1. Open up the sample notebook. Explore the data in it using the pane on the right.
2. Under Sales.OrderDetails, click the ellipsis button next to it and click “New Column.” When you do that, you should see the following image at the top of the page:



Figure 08-02: Default Calculated Column

3. Delete “Column = “ and replace it with the following code:

Order Line Total = 'Sales OrderDetails'[qty] * 'Sales OrderDetails'[unitprice]

4. Now add the new column you created (Order Line Total) to a Table visual, along with orderid, productid, qty, and unitprice.

Your table should look like this:

orderid	productid	qty	unitprice	Order Line Total
10248	11	12	\$14	\$168
10248	42	10	\$9.8	\$98
10248	72	5	\$34.8	\$174
10249	14	9	\$18.6	\$167.4
10249	51	40	\$42.4	\$1,696
10250	41	10	\$7.7	\$77
10250	51	25	\$42.4	\$1,100

Figure 08-03: Creating a table with qty, unitprice, and Order Line Total

You did it! You just created your first DAX expression! And you can see that it’s working. For instance, for orderid 10248, you’ll see that you ordered 12 of productid 11. Each unit is \$14. $12 * 14 = 168$. The calculated column is working!

Calculated columns create one value per record in the source tables. Calculated columns can only reference columns that exist in a single table. They cannot reference columns that exist in other tables.

Your Second DAX Expression

Now let's create a measure. A measure is a calculation that is created when it is requested in a visual.

1. Create a new page in the Power BI workbook by clicking on the New Page Button. We will use this page shortly.



Figure 08-04: The New Page Button

2. In the Measures table on the right, click the ellipsis button and then click "New Measure."

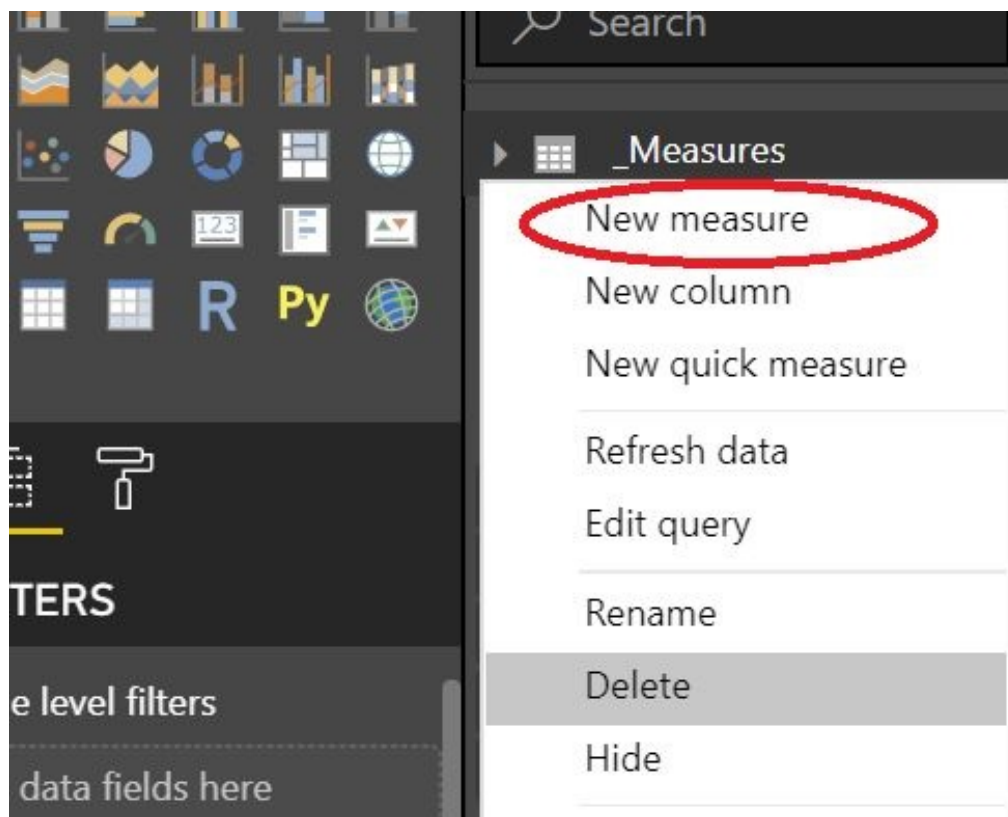


Figure 08-05: The New Measure Button

3. Replace the text that appears in the formula bar with this DAX expression:

Total Sales = SUM('Sales OrderDetails'[Order Line Total])

This measure is named Total Sales. It will sum the Order Line Total column that

you created in the last exercise. The measure will create that sum whenever it is placed in a visual.

4. On the new page that you created in the Power BI Project, click on a stacked bar chart from the Visuals pane to add it to the canvas.
5. From the Date table, drag year to the Axis portion of the chart. Drag Total Sales to the Value portion of the chart. Your fields pane should look like the image below:

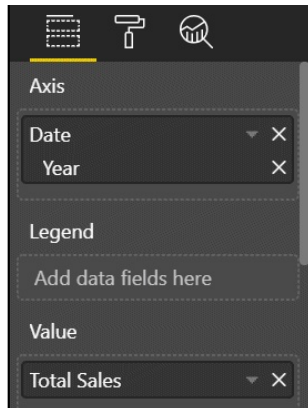


Figure 08-06: The New Fields Pane

6. Your stacked column chart should look like this:

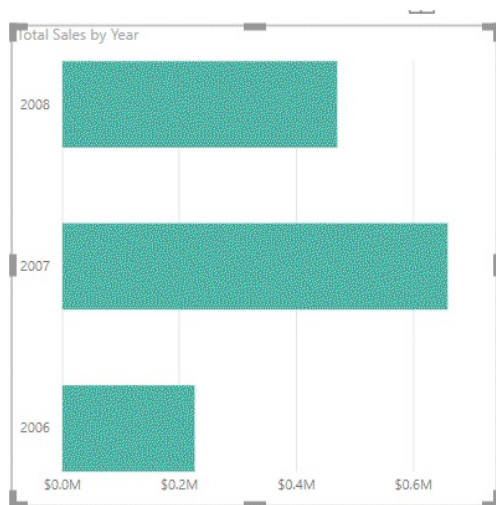


Figure 08-07: The Stacked Column Chart with Total Sales

7. Play with this chart. Keep Total Sales as the Value, but change the field in the Axis portion. You might try Production Category.categoryname, Sales Customers.region, Production Products.productname.
8. Notice that no matter which field you put in the Axis, the breakdown always appears and calculates for you. Also notice that it always totals

the same: \$1,354,458.59. The calculation is always consistent, no matter how you choose to break it down.

You did it! You just created your first measure.

DAX Expressions always breakdown like the following diagram:

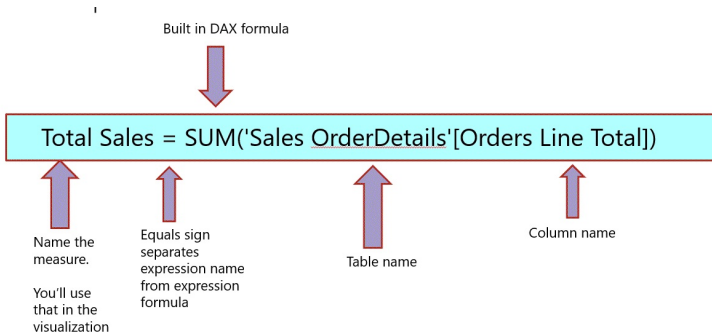


Figure 08-08: The DAX Expression Breakdown

No matter how complicated a DAX expression can be, they will have the parts broken apart above. Sometimes it's helpful to separate those parts in your head as you try to parse difficult to read expressions.

There are all kinds of easy to use expressions, some of which will be familiar to you if you already know Microsoft Excel or T-SQL. Here are a few examples:

- SUM
- AVERAGE
- MIN
- MAX
- COUNT
- COUNTROWS
- DATEDIFF
- DATEADD

Another Example

Let's try it again with a different example. I will give you fewer instructions this time and no visuals so you can get some practice in.

1. Create a calculated column on the Sales Orders table. Use the following expression:

`Days To Ship = DATEDIFF('Sales Orders'[orderdate], 'Sales Orders'[shippeddate], DAY)`

2. This calculated column takes the date between orderdate and shippeddate in day increments. It calculates how long it took to ship something.
3. Create a table and place the orderid, orderdate, shippeddate, and Days To Ship columns in it. Does the data look correct to you?

Now that we created a calculated column, let's figure out our average days to ship something.

1. In the Measures table, create a new measure. Use the following code:

`Average Days to Ship = AVERAGE('Sales Orders'[Days To Ship])`

2. Add this measure to a stacked bar chart. Put it in the Values section.
3. Add different fields to the Axis section of the visual. Can you see how long it takes to ship something to particular regions? How about by Year and Month?

By now you're getting all kinds of experience with calculated columns and measures. You should feel accomplished. You are well on your way to understanding DAX.

Calculated Tables

You can also use DAX to create calculated tables. Look at the definition of the Dates table. You should see a DAX expression like the following:

```
Dates = CALENDAR("1/1/2000", "12/31/2016")
```

This DAX expression creates a dates table from 1/1/2000 to 12/31/2016. It will create a row for every date between those two dates. It will also break down the date into a hierarchy of Year, Quarter, Month, and Day. It will also make time intelligence operations like Year to Date and Month to Date calculations much easier. We'll see that a little later.

The CALCULATE Function

This function is one of the most important DAX functions to learn. It can be complicated to understand for people who are learning Power BI and DAX. It is sometimes easier to talk about what DAX is doing when it calculates a measure for you, and then show how CALCULATE can manipulate the natural behaviour of DAX and Power BI. There is a different chapter in this book about Contexts. I would encourage you to read that chapter. This is a brief summary of much of the information in that chapter.

1. In the Power BI Workbook, examine the three pages called Filter Context 1, Filter Context 2, and Filter Context 3.
2. All three pages have different visuals break down the exact same total: the sum total for freight charges over the lifetime of the company. Notice that the total freight charged by this company is \$64,942.69.
3. On the page Filter Context One, Click on 2008 in Column Chart. Notice how the values in the table change to only include the totals from 2008.
4. In the bottom visual, click on the values for USA. Notice how the other two visuals change what they display. These visuals are interacting with each other.
5. On the page titled Filter Context 3, you will see a Map visual, next to a Column visual. Click on the bubble for USA and notice how the Column visual changes. Click on one of the columns in the Column visual and notice how the map is changing.

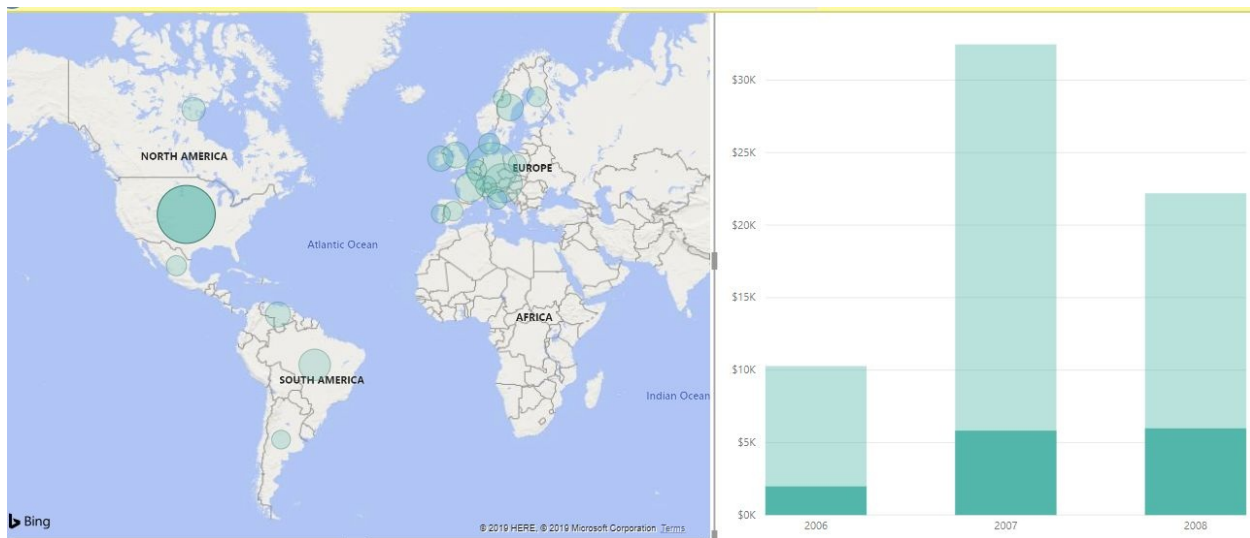


Figure 08-09: The Column Chart affecting the Map visual

What is going on here? Power BI is calculating the total freight for you based on

how you are interacting with these visuals. It is calculating the total on the fly and right in front of you. It is very quick! When the user interacts with visuals, Power BI is telling DAX what the filter context is.

The CALCULATE function overrides the filter context and makes the visual display exactly what data you want it to display. Follow these steps to see it in action:

1. In the measures table, create a new measure.
2. Replace the code with the following:

```
Total Sales (Beverages) = CALCULATE  
(  
    SUM('Sales OrderDetails'[Order Line Total])  
    , 'Production Categories'[categoryname] = "Beverages"  
)
```

3. The measure overrides the filter context as it relates to Production Categories.categoryname. No matter what category we click in the filter context, we will always get the sum of Sales OrderDetails.Order Line Total for beverages. Let's see this in a visual.
4. Add a table visual to a new page. Add Year, Total Sales, and Beverages Total Sales to the table. It should look like the image below

Year	Total Sales	Total Sales - Beverages
2006	\$226,298.5	\$53,879.2
2007	\$658,388.75	\$110,424
2008	\$469,771.34	\$122,223.75
Total	\$1,354,458.59	\$286,526.95

Figure 08-10: The Total Sales – Beverages measure compared to the Total Sales measure

5. Notice how the Total Sales – Beverages measure is a smaller amount than Total Sales. This is because the CALCULATE function is creating a new the filter context for our new measure where the categoryname is always “Beverages”.
6. Now create a new table visual. Add categoryname (found in the Production Categories table), Total Sales – Beverages, and Total Sales. Your visual should look like the image below:

Figure 08-11: The Total Sales – Beverages measure compared to the Total Sales measure by categoryname

7. Notice how no matter which category is specified in the filter context, the total amount is always the same for Total Sales – Beverages. This is because the CALCULATE function is overriding the filter context.
8. Now add Year as the first value in Values section. Your visual will look like this:



The screenshot shows a table with four columns: Year, categoryname, Total Sales - Beverages, and Total Sales. The data is grouped by Year (2006, 2007, 2008) and then by categoryname (Beverages, Condiments, Confections, Dairy Products). The 'Total Sales - Beverages' column shows a constant value for each year across all categories, while the 'Total Sales' column shows the sum of sales for each category within each year. A 'Total' row is at the bottom.

Year	categoryname	Total Sales - Beverages	Total Sales
2006	Beverages	\$53,879.2	\$53,879.2
2007	Beverages	\$110,424	\$110,424
2008	Beverages	\$122,223.75	\$122,223.75
2006	Condiments	\$53,879.2	\$19,458.3
2007	Condiments	\$110,424	\$59,679
2008	Condiments	\$122,223.75	\$34,557.45
2006	Confections	\$53,879.2	\$31,511.6
2007	Confections	\$110,424	\$87,227.77
2008	Confections	\$122,223.75	\$58,359.73
2006	Dairy Products	\$53,879.2	\$11,615.8
	Total	\$286,526.95	\$1,354,458.59

Figure 08-12: Adding Year

9. You can see that though we are overriding the filter context as it relates to the categoryname, we are not overriding the filter context for date or for any other column. We are only overriding it for one particular column and value.

If we were to look at the online documentation for the CALCULATE function, we would immediately see a syntax description that looks like this:

```
CALCULATE(<expression>,<filter1>,<filter2>...)
```

Figure 08-13: CALCULATE in books online

The online documentation, as of this writing, is found here:

<https://docs.microsoft.com/en-us/dax/calculate-function-dax>

Notice how the CALCULATE function can take more than one filter, ie filter1 and filter2. It also has a “...” at the end of the definition. This is the

documentation telling you that you can apply more than one filter override in the CALCULATE function. You can override a lot of different column values. Let's see what it looks like to override two columns.

1. In the Measure table, create a new measure.
2. Replace the code with the code below

```
Total Sales (Beverages in USA) = CALCULATE(  
    sum('Sales OrderDetails'[Order Line Total])  
    , 'Production Categories'[categoryname]= "Beverages"  
    , 'Sales Customers'[country] = "USA"  
)
```

3. This measure uses CALCULATE to override both the category name (forcing Beverages) and country (forcing USA).
4. We name the measure Total Sales – Beverages in USA to tell the user of the report that we are overriding their wishes here.
5. Now add a table to a page and add year, categoryname, Total Sales, Total Sales – Beverages, and Total Sales – Beverages in the USA
6. Play with the filter context and see how different filters affect these three measures differently. You should notice similar things with how CALCULATE is overriding the filter context.

I know this can be difficult to understand. With practice and patience, you can learn exactly what Power BI is doing here. Often it just takes trying to apply it with real world problems and working through it to find a solution. As you practice with this, you will understand this better and better. We will use CALCULATE a few times more before this chapter is over.

Variables and Return

DAX calculations can be multiple lines. They do not have to be a single line. Using multiple lines can make DAX far easier to read and troubleshoot. When using variables and the RETURN statement, you must use multiple lines.

1. In the measures table, create a new measure.
2. Replace the code with the following:

```
Total Sales For Customers with Minimum Order Count =  
VAR MinimumOrderCount = 5  
VAR CustomersWithMinimumOrders = CALCULATE  
(  
    sum('Sales OrderDetails'[Order Line Total])  
    , FILTER('Sales Customers', [Number of Orders] > MinimumOrderCount)  
)  
RETURN CustomersWithMinimumOrders
```

3. This measure is multiple lines. It declares two variables. The first variable is titled MinimumOrderCount. It has an assigned value of 5.
4. The second variable is titled CustomersWithMinimumOrders. It uses the CALCULATE function to only aggregate Order Line Total for customers that meet the minimum order count.
 - a. This can be hard to read if you're new to DAX and CALCULATE. In this case, the second argument of the CALCULATE function has a nested function inside of it. That nested function, FILTER, is filtering the Sales Customers table. The FILTER function is filtering out all customers that have less than 5 orders. Then CALCULATE is totaling the Order Line Total for just these customers.
 - b. The net effect of this is that we are getting a Total Sales calculation for only our repeating customers that have given us the minimum number of five orders. For this business, repeat customers defined like this are interesting to the stake holders. They are categorized separately.
5. Add this measure to a table visual next to the Total Sales measure to see how the number are different.

Total Sales For Customers with Minimum Order Count	Total Sales
\$1,284,359.38	\$1,354,458.59

Figure 08-14: Multiple line DAX calculation in a table visual

6. Having a multiline DAX calculation can help with troubleshooting. Change the value of the MinimumOrderCount variable from 5 to 2. And then change it to 7. Then change it to 10.
 - a. You can see the value change as you change the definition. This can help you discover the real order count that you care about. The variable title also helps you document why the number 5 is important to you. Of course, you can just embed the number 5 inside the CALCULATE statement, but then you've lost intent. Why are you putting 5 there? Variables make code easier to read and maintain over time.
7. The RETURN statement does not need to return the last variable. Change the RETURN statement to return MinimumOrderCount instead of CustomersWithMinimumOrders. The code would look like this:

```
Total Sales For Customers with Minimum Order Count =  
VAR MinimumOrderCount = 5  
VAR CustomersWithMinimumOrders = CALCULATE  
    (  
        sum('Sales OrderDetails'[Order Line Total])  
        , FILTER('Sales Customers', [Number of Orders] > MinimumOrderCount)  
    )  
RETURN MinimumOrderCount
```

8. If you have a complicated, multi-step formula, you can use RETURN to verify that each step is performing the way you expect it to.

Time Intelligence – YTD

1. Create a new measure in the measures table.
2. Replace the code with the following:

YTD Total Sales = TOTALYTD

```
(  
    SUM('Sales OrderDetails'[Order Line Total])  
    , Dates[Date].[Date]  
)
```

3. Similar to the CALCULATE function, the TOTALYTD function also manipulates the filter context. It creates a running total for the year, no matter which dates, months, quarters, or years have been selected.
4. Add a matrix visual with Month on the rows and Year on the columns. Add YTD Total Sales as the values. It should look like this:

Figure 08-15: TOTALYTD in a measure

5. You can clearly see the total is accumulating as the months progress. You can see it reset when the year resets. The DAX here is deceptively simple, but very powerful! If you wrote something similar in other languages, it would involve a lot of complicated looping and tallying, but DAX does this for you as long as you have a Dates table. The Dates table is necessary for all functions that use time intelligence. There are similar functions for month to date and quarter to date.

Time Intelligence – PREVIOUSMONTH

1. Create a new measure in the measure table.
2. Replace the code to the following:

Total Sales Previous Month = CALCULATE

```
(
    sum('Sales OrderDetails'[Order Line Total])
    , PREVIOUSMONTH(Dates[Date])
)
```

3. This function overrides the filter context to say that no matter which month is selected, give the value for the previous month.
4. Add a Matrix visual with Month as the Row and Year as the Column. Put Total Sales and Total Sales Previous Month in the values. Your visual should look like this:

Year Month	2006		2007		2008		Total	1
	Total Sales	Total Sales Previous Month	Total Sales	Total Sales Previous Month	Total Sales	Total Sales Previous Month	Total Sales	
January			\$66,692.8	\$50,953.4	\$100,854.72	\$77,476.26	\$167,547.52	
February			\$41,207.2	\$66,692.8	\$104,561.95	\$100,854.72	\$145,769.15	
March			\$39,979.9	\$41,207.2	\$109,825.45	\$104,561.95	\$149,805.35	
April			\$55,699.39	\$39,979.9	\$134,630.56	\$109,825.45	\$190,329.95	
May			\$56,823.7	\$55,699.39	\$19,898.66	\$134,630.56	\$76,722.36	
June			\$39,088	\$56,823.7		\$19,898.66	\$39,088	
July	\$30,192.1		\$55,464.93	\$39,088			\$85,657.03	
August	\$26,609.4	\$30,192.1	\$49,981.69	\$55,464.93			\$76,591.09	
September	\$27,636	\$26,609.4	\$59,733.02	\$49,981.69			\$87,369.02	
October	\$41,203.6	\$27,636	\$70,328.5	\$59,733.02			\$111,532.1	
November	\$49,704	\$41,203.6	\$45,913.36	\$70,328.5			\$95,617.36	
December	\$50,953.4	\$49,704	\$77,476.26	\$45,913.36			\$128,429.66	
Total	\$226,298.5		\$658,388.75	\$50,953.4	\$469,771.34	\$77,476.26	\$1,354,458.59	

Figure 08-16: PREVIOUSMONTH in a measure

5. CALCULATE and PREVIOUSMONTH overrides the filter context to display the value for the previous month selected. For instance, in August 2006, you can see that the current value is \$26,609.40, while the previous month value is \$30,192.10.

X vs Non-X Functions

You may have noticed that there are a lot of functions that have the same name, but are only different because one has an X at the end of it. Some examples are SUM, SUMX, COUNT, COUNTX, MIN, MINX, and so forth. What are the differences between these functions?

The X functions are what are called iterator functions. They are important where the value is important in the context of a row. It works row by row. SUMX has awareness of rows in a table, hence can reference the intersection of each row with any columns in the table. SUMX will add values as it relates to the entire row that is iterating.

SUM is an aggregator function. It works like a measure, calculating based on the current filter context.

The following is an example of a SUMX function:

```
Total Sales SUMX = SUMX(  
    'Sales OrderDetails'  
    , 'Sales OrderDetails'[qty] * 'Sales OrderDetails'[unitprice]  
)
```

Notice that it multiplies qty * unitprice, row by row, and then sum the results. If you added this code to a measure and put it in a visual, you would notice that the total is the expected \$1,354,458.59. If we attempted to create this function using SUM, we would get a syntax error. SUM can only operate on one column.

After fiddling with it, we might come up with something like this:

```
BadMeasure = (SUM('Sales OrderDetails'[unitprice]) *  
Sum('Sales OrderDetails'[qty]))
```

This measure takes the total number of qty in OrderDetails and multiplies it with the total UnitPrice. This is very bad logic and results in \$2,899,457,198.47.

This result is very wrong.

SUMX is necessary because it preserves the row value and ties the correct qty with the correct unitprice.

Best Practice: Organize your code

This author recommends keeping measures together in one table. You may want to create more than one table for your measures. In my experience, you will end up having dozens of measures in your Power BI model. It might be difficult to find a measure if you don't practice organization.

Best Practice: Naming Columns & Measures

When you name columns and measures, obey the following rules:

- Feel free to use spaces
- Avoid acronyms
- Make names terse, but descriptive
 - Makes Q & A easier to use
- In formulas, reference table names for calculated columns and do not reference table names for measures, so you'll know the difference

These rules will make it easier to maintain your Power BI project, data model, and reports.

Best Practice: Formatting

Power BI and DAX ignore white space. That means you can freely use whitespace to make your code easy to read and easy to navigate.

```
Days To Ship = DATEDIFF
(
    'Sales Orders'[orderdate]
    , 'Sales Orders'[shippeddate]
    , DAY
)
```

Figure 08-17: DAX ignores white space

DAX Expressions can have lots of parentheses and square brackets. Please use white space to control help this. Above is a good example of a properly formatted DAX expression. You can also use the website DaxFormatter.com to help you format your DAX expressions.

Other Resources

The following books can help you along your DAX journey:

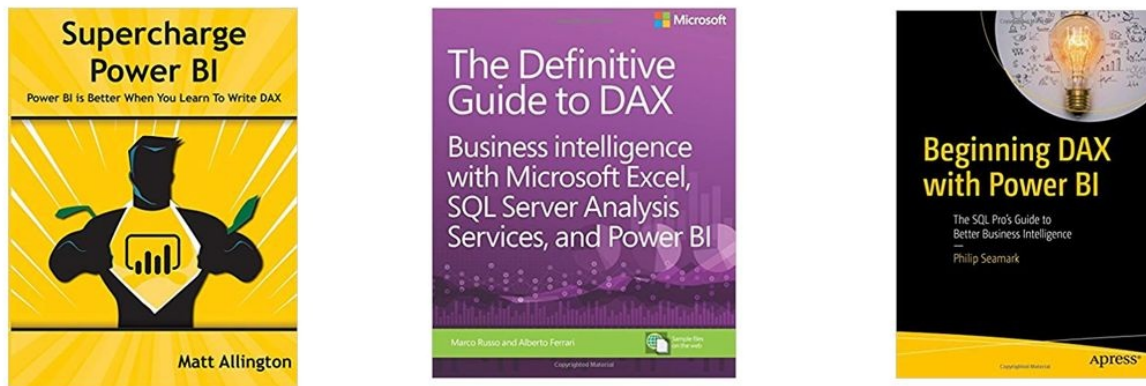


Figure 08-18: Great DAX books

DAX Studio is a great tool for learning DAX and writing DAX expressions. It was the brainchild of one of the authors of one of the books above, Darren Gosbell, Marco Russo and Alberto Ferrari. Learning this tool is essential for the budding DAX developer

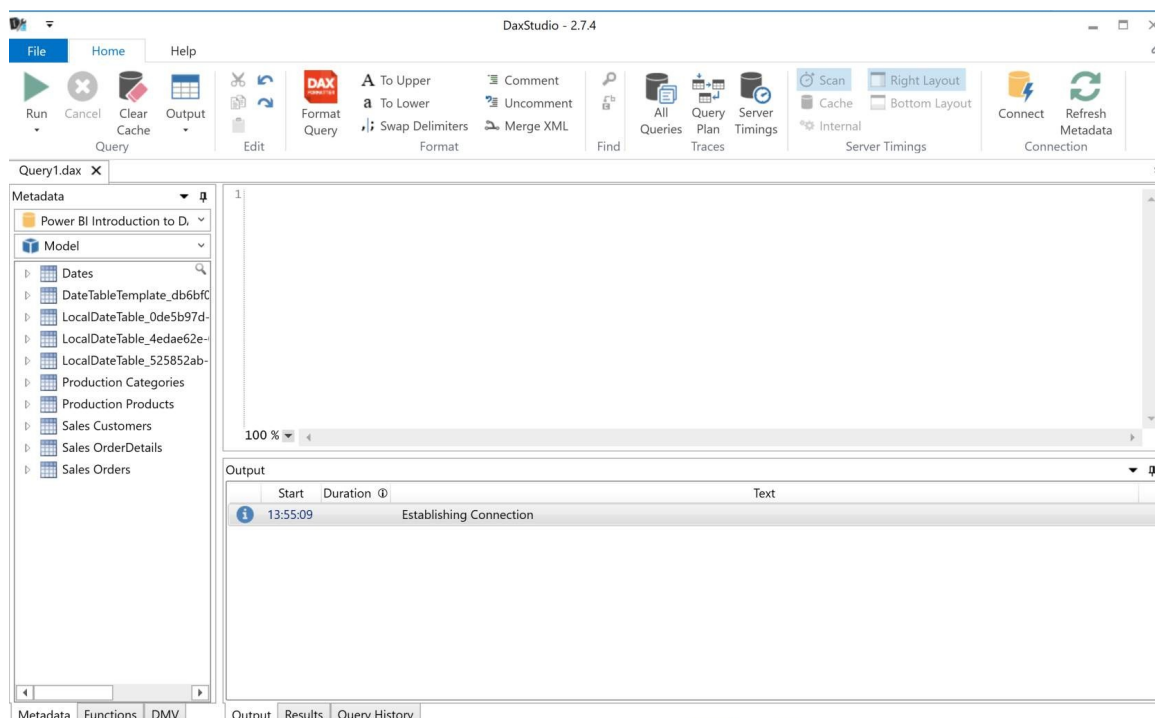


Figure 08-19: DAX Studio

Those two, affectionately called The Italians by the Power BI community, also

created a website called [DAX.guide](#). This website, along with the DAX documentation will also accelerate your journey to DAX mastery.

Summary

This chapter helped you begin your journey to answer the job interview question “Do you know DAX?” It covered the essential elements of DAX. From here you can practice and read your way to knowing all that you need to know to answer “Yes! I know DAX really well!”

About the Author



With over 18 years of experience in databases and a current Microsoft MVP, Ike has been Microsoft certified since the beginning, currently holding an MCDBA, MCSE, MCSD, and MCT. Ike is the General Manager of Data & AI for Solliance. We have a full team of cloud data engineers and data scientists. We specialize in building highly scalable data solutions for any size of organization. Ike is a partner in Crafting Bytes, a San Diego software studio and Data Engineering group. We build software and BI solutions for companies all around the country.

In 2010, Ike founded the San Diego Tech Immersion Group (SDTIG). It has grown to be the largest user group in San Diego with over 125 active members including three other Microsoft MVPs. It is a technical book club that reads a book on a significant technical topic. Recent topics include Linux on Microsoft Azure, Angular 2/TypeScript, Data on Azure, Python for Data Scientists, and Docker/DevOps. In July 2018, SDTIG started a track on Docker/Kubernetes. We will start a new track on Databricks/Spark in November 2018. You can join virtually and watch the youtube live stream. www.sdtig.com.

Ike leads the San Diego Power BI and PowerApps user group that meets monthly. Ike is also the leader for San Diego Nerd Beers, a monthly software development social group and the co-chairman of the San Diego Software Architecture Group, a round-table of software leaders in San Diego. He also co-chairs the San Diego Software Architecture Group with Azure MVP Alumni Scott Reed.

In 2015, he co-wrote Developing Azure Solutions for Microsoft Press. He contributed all sections that related to the Microsoft Data Platform. The second edition will be released in 2018.

For more information, see <http://www.ikeellis.com>

Chapter 9: Power BI is Not the Same as Excel

Author: Matt Allington

Power BI and Excel are similar in what they can be used for, but they are not the same type of tool. It can be deceptive because both tools seem to do similar things using a similar approach. For example, Excel has a functional language that is used for calculations, and Power BI also has a functional language (called DAX). But that doesn't mean Excel formulas are exactly the same as DAX formulas (although they can sometimes be similar). In this chapter I will cover the differences you need to be aware of to start your Power BI journey, particularly when you come from an Excel background.

Introduction

I have been teaching people how to use Power BI, Power Pivot and Power Query since 2014 (more than 5 years at this writing). Most of the people I have taught (although not all) have come from an Excel background. That is to say that they have spent most of their business life using Microsoft Excel as their data analytics tool of choice. If you are reading this chapter, then chances are you are just the same. You love Excel but also see potential to take your reporting and analytics to the next level using Power BI.

Over my time teaching thousands of students, I have developed a deep understanding of the things that people struggle with when trying to make the transition from Excel to Power BI. I am not saying that they (or you) can't make the transition; I strongly believe that any competent Excel user can make the transition from using Excel to using Power BI. But what you must understand is that Power BI is not the same type of tool as Excel.

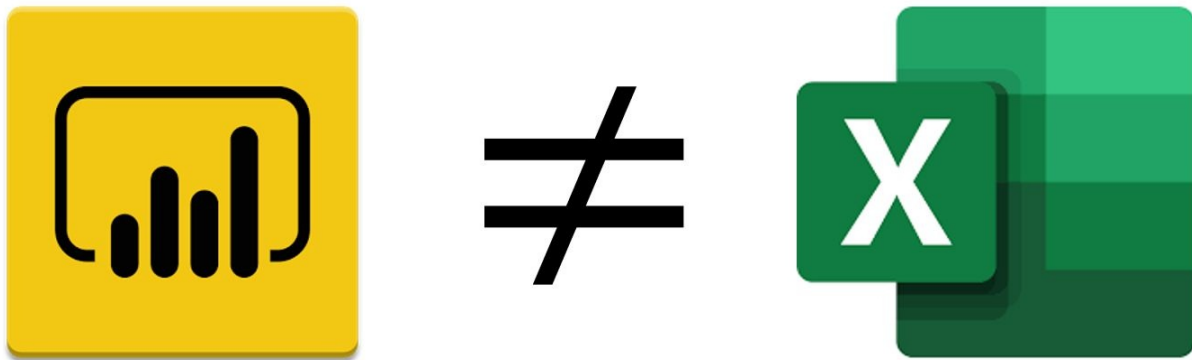


Figure 09-01: Power BI is not the same type of tool as Excel

Power BI is an umbrella term referring to a very broad ecosystem including Power BI Desktop, the Power BI Service (Powerbi.com), on premise report servers, and paginated reports to name just a subset of the total ecosystem. In this chapter I will refer to just the Power BI authoring tool called Power BI Desktop. Power BI Desktop and the PC version of Excel are the tools that are potential substitutes for each other in the world of self-service business intelligence reporting. I will be comparing Power BI Desktop to Microsoft Excel for PC, although the same comparison also applies also to Excel for Mac. At this writing, there is no Power BI Desktop for Mac.

Some Things Are the Same in Power BI and Excel

Built with You in Mind

Let me start out by talking about the similarities between Power BI Desktop and Microsoft Excel for PC. Firstly, and most importantly, Power BI Desktop was built with Excel users front of mind. Microsoft did this for a very important reason – it wanted to build a tool that was easy for Excel users to use. Microsoft corralled key people from the Excel and Analysis Services teams and set them to the task of building a new self-service BI tool. Initially the team built Power Pivot for Excel. Then a team was created to build Power Query for Excel and another team worked on Power View for Excel. Once these three “products” were underway, Microsoft had started to build out all the components needed to create the new tool that we know of today as Power BI.

DAX is a Functional Language

At the core of Power BI is the Vertipaq Engine and the DAX Language. DAX (Data Analysis Expressions) is the formula language of Power BI. Some of the older business intelligence and relational database tools do not have a functional language at all and instead have a scripted query language. Traditional SQL Server Analysis Services has a language called MDX (Multidimensional Expressions). SQL Server itself uses a scripted query language called T-SQL (Transactional-Structured Query Language). T-SQL is relatively easy to learn and MDX is quite hard to learn, but the point is that neither of these database tools have a functional language like in Excel.

A functional language is one that uses functions to perform a task. I like to think of a function like a black box – you give it some inputs and it gives you some outputs. You don’t need to know “how” it comes up with the result – that is the job of the function. Consider the function SUM() in Excel. This function takes one or more parameters as input(s) and returns the addition of the values as its output. Power BI also has a functional language (DAX) and this makes it is relatively easy for Excel users to make the transition from Excel functions to DAX functions.

DAX has Many Common Functions with Excel

There are many functions that have the same or similar syntax to DAX. The example SUM() given above is a good case in point. The function has the same name and similar syntax. In Excel the SUM() function accepts ranges as inputs, and then the values in those ranges are added together. DAX is slightly different

– it accepts a single column from a single table as the input and adds the values in that column. More on this later.

Here is another example: both DAX and Excel have a function called TODAY(). In both tools, this function does not take any parameters and will return the current date from the PC.

The list of common functions is very long. You can download my DAX Quick Reference Guide PDF from <http://xbi.com.au/drg>

Sometimes Functions Are Similar, But Have Small Differences

There are many common functions that have the same name in both DAX and Excel but have a slightly different syntax. Take the OR() and AND() functions as 2 examples. In Excel, both of these functions can take as many input parameters as needed to solve the problem at hand.

Excel Syntax OR(logical1,[logical2],...[logicalN])

Power BI Syntax OR(logical1,logical2)

Do you spot the difference? The Excel syntax will accept 1, 2, 3, or as many parameters as you need. The Power BI syntax must have 2 and exactly 2 parameters. For this reason, I am not a big fan of the OR() and AND() syntax in Power BI. Instead I tend to use the in-line syntax as follows:

Power BI OR Inline Syntax logical1 || logical2 || logicalN

The double pipe is the “in-line” way of saying OR in DAX. The pipe symbol is the vertical bar somewhere on your keyboard. The exact location depends on the locale of your keyboard.

Power BI AND inline syntax logical1 && logical 2 && logicalN

Many Things Are Very Different Between Power BI and Excel

Although there are many similarities between Power BI and Excel, there are many more things that are very different. The rest of this chapter is dedicated to explaining these differences so that you understand them and know how to approach Power BI differently to Excel.

Power BI is a Database, Excel is a Spreadsheet

Let me start off by defining what I mean by a database and a spreadsheet.

Database

A database is a tool that is built to efficiently store and retrieve data. There are different types of databases including transactional databases (like SQL Server) and reporting databases (like SQL Server Analysis Services SSAS). Power BI is more like SSAS than it is like SQL Server.

The Power BI database is a reporting database. It is not designed to allow you to edit or alter the data that is loaded, but instead to faithfully and efficiently produce summary reports of the actual data loaded.

Spreadsheet

A spreadsheet is a 2-dimensional page that contains multiple cells in rows and columns. It is an unconstrained “canvas like” space not unlike a canvas used by an artist to create a picture. One of the greatest strengths of a spreadsheet is that you can do anything you want with the space. One of the greatest weakness of a spreadsheet is, you guessed it, you can do anything you want with the space. A spreadsheet is a doubled edged sword. The freedom and flexibility to do what ever you want is its greatest strength and its greatest weakness.

A Spreadsheet Used as a Database

While not wanting to confuse matters any more than needed, the simple fact is that many business people tend to use Excel as a database even though it is actually a spreadsheet. I am not saying it is wrong to do this, it is just that Excel lacks the checks and balances that exist in a database tool to prevent accidental errors being created. Have you ever wondered why a 500,000 row spreadsheet can take 20 mins to update when you make a change? This is why – it is a spreadsheet and was not designed for its primary purpose to be a database.

One of the key things that Microsoft has set out to change with Power BI is to provide a more robust reporting database tool designed to be used and maintained by business users. Actually, Power BI is also an enterprise strength BI tool. Power BI is designed to serve both self-service and enterprise BI users.

Differences Between a Spreadsheet and a Reporting Database

There are quite a few differences between these things; here are just a few of them.

Excel

Power BI

Can change one or more data points by just typing over the top.

Data needs to be in a single table before you can summarize the data using a Pivot Table or Pivot Chart.

Excel formulas are created and then copied. Each cell therefore has its own unique copy of a formula (except array formulas of course). It is possible that formulas across cells can get out of sync causing potential errors.

Each copy of the formula can be edited as needed. You can write formulas like SUMIF() to create on the fly filters and summaries.

The data must be changed prior to (or during) load. Once the data is loaded, you can't change it.

Data doesn't need to be consolidated into a single table prior to summarizing using a Matrix or any other visual. In fact you are encouraged to keep the data in separate tables as a star schema.

DAX formulas (measures) are written once. The same measure is then used over and over again in different visuals. There is only one single definition of the measure, so there can be no exceptions and discrepancies. The results of each calculation are not permanently stored but only used when needed to be displayed in a visual.

DAX formulas (calculated columns) can be written in the data view to extend a loaded table of data. This looks very similar to Excel, but you should choose to use measures instead of calculated columns where possible (so the data is not permanently stored).

A single measure is re-used with different "filters" applied to get different results from the same measure.

Figure 09-02: Differences between a spreadsheet and a reporting database

Tips to Get You Started as You Move to Power BI

The rest of this chapter will focus on my key tips to help get you started on your Power BI journey. What you do with this information is important, and it is related to your teachability index. The Teachability Index is defined as:

$$\text{Ability to Learn} \times \text{Willingness to Change}$$

There is no point learning something new if you are not willing to change. There is no point being willing to change if you don't know what needs to be done differently. What is your teachability index?

DAX Calculated Columns vs Measures vs Tables

Both Power BI and Excel have a functional language that a business user can use to extract insights from the underlying data. But unlike Excel, the DAX language can be used in three very different ways within Power BI Desktop:

- DAX Measures
- DAX Calculated Columns
- DAX Tables

Each of these approaches to writing DAX are valid, and they all use the same DAX language. All approaches have their place and purpose in Power BI, however in my experience, most beginners make some simple mistakes and select the wrong approach.

Excel Users Write Too Many Calculated Columns

The most common problem I see Excel users make is to write too many calculated columns. I am not saying it is wrong to write a calculated column – what I am saying is that most Excel users tend to default to writing calculated columns when they would be better off writing measures instead.

There are a few reasons why Excel users tend to default to writing calculated columns rather than measures.

- New Power BI users are often self-taught, and they don't know the difference between a calculated column and a measure. Indeed, often they don't even know that measures even exist.
- When you switch to the data view in Power BI, it looks a lot like an Excel spreadsheet. Excel users feel very comfortable when adding a new column to a table that looks like this, because it feels like home.

ExtendedAmount	TotalProductCost	TaxAmt	Freight	RegionMonthID	DiscountSell
564.99	308.2179	45.1992	14.1248	Australia6	451.99
564.99	308.2179	45.1992	14.1248	Australia6	463.29
24.49	9.1593	1.9592	0.6123	Australia6	23.02
8.99	3.3623	0.7192	0.2248	Australia6	7.82
8.99	3.3623	0.7192	0.2248	Australia6	7.19
564.99	308.2179	45.1992	14.1248	Australia6	474.59
53.99	41.5723	4.3192	1.3498	Australia6	40.49
53.99	41.5723	4.3192	1.3498	Australia6	42.65
120	44.88	9.6	3	Australia6	102.00
7.95	2.9733	0.636	0.1988	Australia6	6.60
24.49	9.1593	1.9592	0.6123	Australia6	20.33

Figure 09-03: Data view in Power BI looks like an Excel spreadsheet

- The DAX syntax is normally easier in a calculated column, and it is less common to get strange error messages that you don't understand (as is often the case when writing measures for the first time).

SQL Users Write Too Many DAX Table Queries

The most common problem I see SQL Server professionals make is to write too many table queries. This is understandable I guess, because SQL professionals are comfortable writing queries and DAX is used as a query language when creating tables. But as the saying goes, “just because you can, doesn't mean you should”. There is a time and a place for DAX queries and DAX tables, but it is often not the best approach in Power BI (sometimes, but not often).

Everyone Should Be Writing Lots of Measures

The new shiny thing that sets Power BI apart from Excel is the introduction of Measures. Measures are not a new concept – they have been available in SQL Server Analysis Services for many years. But measures are new to most Excel users, and they are not always the first thing that comes to mind to SQL professionals. Unless someone tells you that you should be using more measures, how would you even know? I'm telling you now – so now you know.

Measures have a lot of benefits over Calculated Columns and Tables. I like to think of the data you load into Power BI being a bit like raw cooking ingredients, and the measures you write being like a “recipe” to “cook” something up using the raw data. The recipe is repeatable – you can use it over and over again against the same ingredients to get the same result.

Benefits of measures include:

- Measures do not create duplications of your data. Conversely, both calculated columns and tables duplicate the data in Power BI.

- You write a measure once, and then use it many times.
- Each measure that references one or more columns can be given its own unique name making it easy for end users to find and use the business insights they need.

g.:

Total Sales = SUM(Sales[Value])

Avg Transaction Value = AVERAGE(Sales[Value])

Note how the above 2 measures reference the same column, but they deliver a different business insight. The name clearly communicates what the measure is providing.

- Each measure has its own bespoke formatting that is fit for purpose. e. g. even though Total Sales and Avg Transaction Value above are calculated from the same column of data, both of these measures can have their own data format

g.:

Total Sales = \$29.36 M

Avg Transaction Value = \$486.09

- Measures can use data from multiple columns and tables within the same formula. This cannot be done with a simple drag and drop of a column into a visual in Power BI.

Using Visuals to Structure your Output

When you first open a new report in Power BI, you are faced with a daunting sight. The Power BI Report canvas looks a lot more like Power Point than it does Excel.

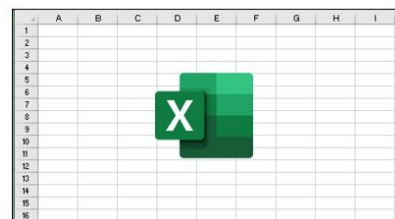
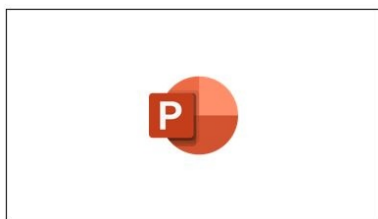


Figure 09-04: PowerPoint's, Power BI's and Excel's canvas

Power BI intentionally has a blank reporting canvas. When you want to visualize your data, you need to add one or more visualisations to the reporting canvas to create the layout structure for your report.

Creating Structure in Excel

Excel has a 2 x 2 grid to store your data. You can put any bespoke number, text,

formula etc, in each and every cell on the sheet. There is little or no constraint on how you must proceed to structure your report. It is possible that you may end up with a nice, orderly, tabular layout in your Excel spreadsheet, but of course it may not end up that way if you are not careful.

Creating Structure in T-SQL

Writing queries using T-SQL uses a different approach to a spreadsheet. You write code using the T-SQL query language. The structure of the tabular output and all the aggregations are configured inside the T-SQL script itself (see the example below).

T-SQL Script	Table Result										
<pre>SELECT P.Category , SUM([ExtendedAmount]) AS TotalSales FROM [AdventureWorks].[dbo].[Sales] AS S LEFT JOIN Products AS P ON S.ProductKey = P.ProductKey GROUP BY P.Category ORDER BY P.Category</pre>	<table><tr><th>Results</th><th>Messages</th></tr><tr><th>Category</th><th>TotalSales</th></tr><tr><td>1 Accessories</td><td>700759.959999784</td></tr><tr><td>2 Bikes</td><td>28318144.650696</td></tr><tr><td>3 Clothing</td><td>339772.609999958</td></tr></table>	Results	Messages	Category	TotalSales	1 Accessories	700759.959999784	2 Bikes	28318144.650696	3 Clothing	339772.609999958
Results	Messages										
Category	TotalSales										
1 Accessories	700759.959999784										
2 Bikes	28318144.650696										
3 Clothing	339772.609999958										

Figure 09-05: Creating structure in T-SQL

Creating Structure with DAX Queries

When you write DAX Queries in Power BI to produce a table, the process is quite similar to T-SQL above. This is why many SQL professionals like to write DAX queries. Here is a DAX query that could be considered equivalent to the T-SQL Script above.

1	EVALUATE
2	SUMMARIZECOLUMNS (
3	Products[Category],
4	"TotalSales", SUM (Sales[ExtendedAmount])
5)
6	ORDER BY Products[Category] ASC

Figure 09-06: Creating structure with DAX queries

Despite being able to write DAX queries as shown above, Power BI is not designed to use this approach as the primary way to create summary data. Instead, you should prefer to use Power BI visuals to create the structure.

Creating Structure in Power BI with Visuals

Both the Excel approach and T-SQL approach outlined above are very different to the Power BI approach. When using Power BI, the structure of the output is first selected from one of the available visuals.

Each visual will implicitly shape and control the structure of the final output. If you don't like the result, it is very easy to change it to a different visual (try doing that in Excel).

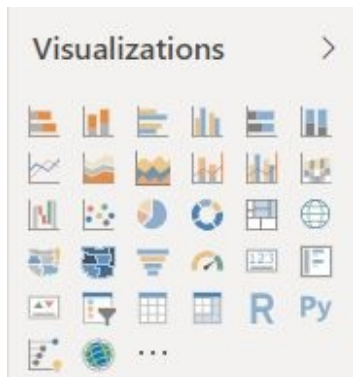


Figure 09-07: Standard visuals in Power BI

I always teach Excel users to start with a Matrix, as this is the closest match to an Excel pivot table and a sheet. Using a Matrix as the starting visual is a good way to get used to using Power BI. Another good thing about using the Matrix visual is you get to “see” the numbers. This makes the experience more “Excel like”, and it is an important visual confirmation that you are doing something right.

After adding the visual to the canvas (as can be seen in 1 below), data can be added to the matrix against Rows (2), Columns (3) and Values (4). This process is very similar to how you would build a pivot table in Excel.

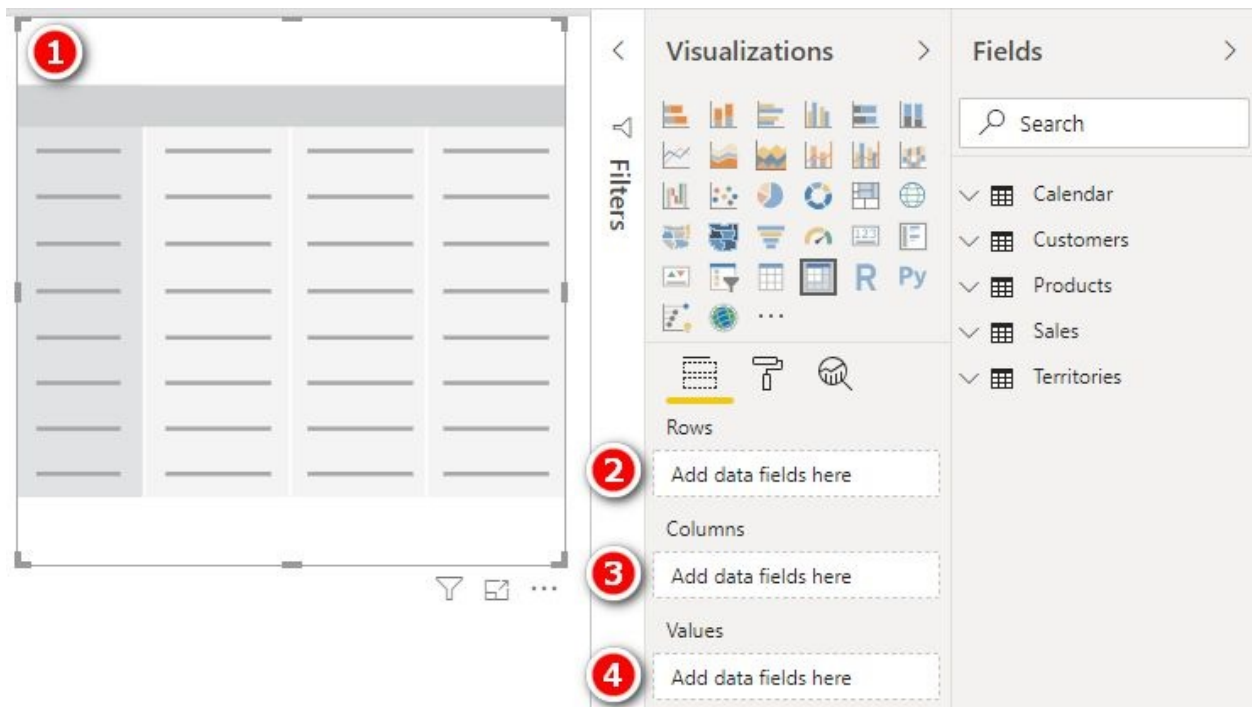


Figure 09-08: Steps to create a visual in Power BI

In the image below, I have built the same table created earlier using T-SQL but I have used the Matrix visual to create the structure of the result. The big difference is that the table below was built without writing a single line of code. I simply added the matrix visual, dragged the Product[Category] column into the Rows and the Sales[ExtendedAmount] column into the Values – Power BI did the rest.

Category	ExtendedAmount
Accessories	700,759.96
Bikes	28,318,144.65
Clothing	339,772.61
Total	29,358,677.22

Figure 09-09: A Matrix visual with columns Category and ExtendedAmount

Note: Sales[ExtendedAmount] is a column of numbers, not a measure. When I added the column to the Values above, Power BI was smart enough to add the numbers up for me without writing a measure.

Why Write Measures at All, Then?

OK, so now you are asking why bother to write measures at all when you can just drag a column of numbers into a visual. Well, the short answer is that you don't have to write measures if you don't need to. The basic concepts such as adding up numbers in a column, finding the average in a column, etc can all be achieved by dragging a column of data to the Values section of the visual, and then setting the aggregation behaviour for that data in the column. If you don't like the default aggregation for the column, you can change it by selecting the drop-down menu (shown in 1 below) and then selecting a different aggregation from the available options (2 below).

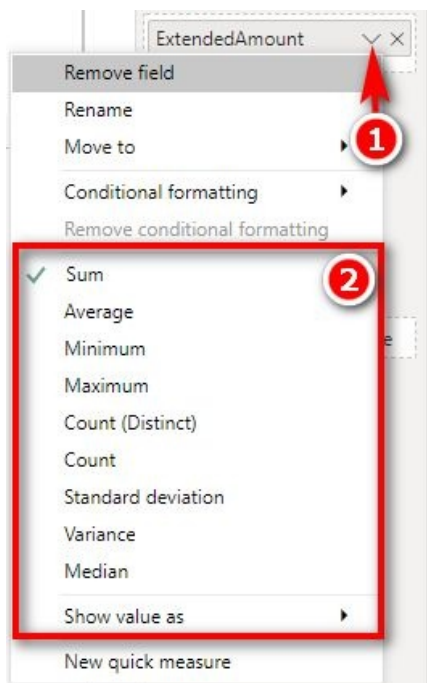


Figure 09-10: How to select the aggregation for column ExtendedAmount

But while the above approach will work, you will very quickly hit the limits of capability. Take the following example:

- You can drag a column to work out the total number of invoices
- You can drag a column to work out the total number of items sold

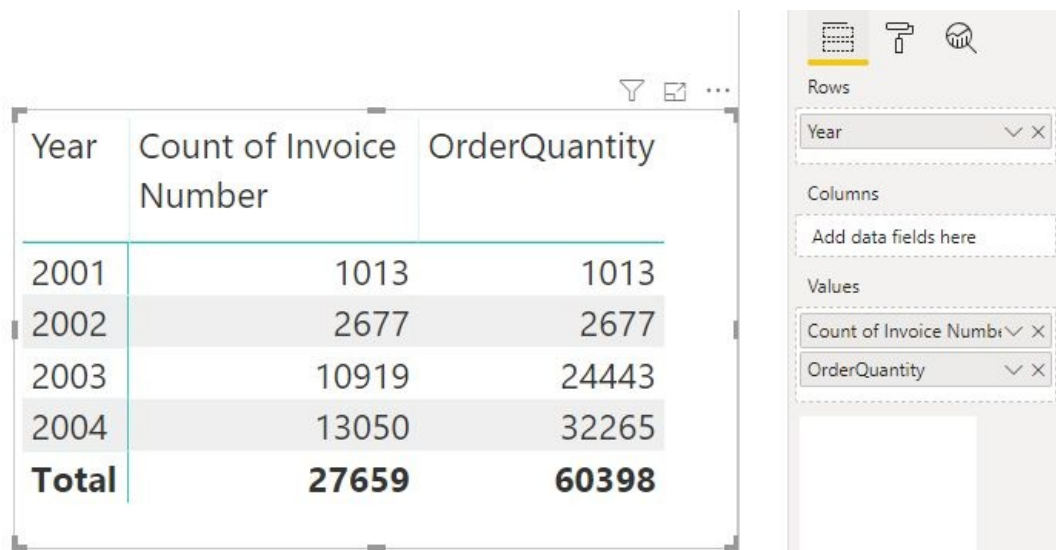


Figure 09-11: A matrix visual over Year, Invoice Number, and Order Quantity

But you can't drag a column to find out the average number of items per invoice. You need data from two columns to complete this calculation, and for that you will need to write a DAX Measure.

I have written the DAX measure to calculate the average number of line items per invoice, and you can see it in the formula bar in the image below.

```
1 Avg Line Items per Invoice = DIVIDE(SUM(Sales[OrderQuantity]), DISTINCTCOUNT(Sales[Invoice Number]))
```

Year	Count of Invoice Number	OrderQuantity	Avg Line Items per Invoice
2001	1013	1013	1.00
2002	2677	2677	1.00
2003	10919	24443	2.24
2004	13050	32265	2.47
Total	27659	60398	2.18

Figure 09-12: A matrix visual by Year, Invoice Number, Order Qty, and Avg Line Items per Invoice

In addition to needing a DAX measure when you are referencing multiple columns of data, don't forget all the benefits of having better business names for

your measures and also being able to apply suitable formatting for each individual business concept. Compare the table above with the DAX version below and note the changes/improvements in the names and the formatting.

Figure 09-13: A matrix visual over Total Line Items, Total Invoices, and Avg Line Items per Invoice

Total Line Items = SUM(Sales[OrderQuantity])

Total Invoices = DISTINCTCOUNT(Sales[Invoice Number])

Avg Line Items per Invoice = DIVIDE([Total Line Items] , [Total Invoices])

Figure 09-14: DAX formulas for Total Line Items, Total Invoices, and Avg Line Items per Invoice

I hope you agree the good business names, the suitable formatting, and the reuse of measures inside other measures all make the small extra effort of writing measures worthwhile. Believe me, writing measures and learning to write DAX is the secret sauce that will set the Power BI superheros apart from the rest of the pack.

Filter First, Calculate Second

In Excel, you can write bespoke, one off formulas in any cell you like. That is not how it works in Power BI. In Power BI, you first take ALL the data you are given in one or more columns or tables. You write generic formulas (Measures) using those column and table inputs and use the visuals to filter the data before returning the results.

Use the Visuals to Filter the Data Before Returning the Results

I want you to read that heading again “use the visuals to filter the data before returning results”. This is one of the most important concepts you must understand about Power BI. Let me refer back to the Matrix I created earlier (shown again below for convenience).

Year	Total Invoices	Total Line Items	Avg Line Items per Invoice
▲			
2001	1,013	1,013	1.0
2002	2,677	2,677	1.0
2003	10,919	24,443	2.2
2004	13,050	32,265	2.5
Total	27,659	60,398	2.2

Figure 09-15: A matrix visual over Total Line Items, Total Invoices, and Avg Line Items per Invoice

As you can see in the visual above, there are 5 different numbers in the Matrix for [Total Invoices] despite the fact that I only wrote one DAX Measure. The DAX measures is as follows:

Total Invoices = DISTINCTCOUNT(Sales[Invoice Number])

Despite only writing a single formula, you can see I have 5 different results. How can that be? The answer is that the **Visuals in Power BI filter your data** – that's what they do.

I am not suggesting you would be surprised to see 5 different numbers for Total Invoices in the Matrix above. What I am saying is that you need to understand HOW those 5 different numbers were created.

The process is as follows.

- The Row section of the visual is displaying the Calendar[Year] column.
- The visual takes the first year (2001 in this case) and filters the Calendar table.
- The filter on the Calendar table flows through the relationship in the direction of the arrow (shown in 1 below in the Model view) onto the sales table (no VLOOKUP required here).

Figure 09-16: Flow of filters from one table to another

- Now the sales table is filtered for all sales in 2001 as a result of the filter flowing from the Calendar table to the sales table. We say that the filter propagates from the Calendar table to the Sales table.
- The DAX formula then is calculated for just those sales that remain after the filter is applied to all the tables.
- Every row, column, bar, line, pie slice, total, sub-total etc in Power BI are ALL calculated this way.

All DAX formulas are always evaluated using this approach. Filter first, evaluate the result second after all filters have been applied. If you remember nothing else from this chapter, this is it.

Filter First, Evaluate Second

Understanding filter propagation is essential to writing good DAX.



About the Author

Matt Allington is a Power BI Consultant, Trainer and MVP specialising in helping business users learn and apply Power BI to solve business problems. Matt has over 35 years' commercial and IT experience using data to get things done. Matt is the author of the bestselling book "Supercharge Power BI – Power BI is Better When You Learn to Write DAX" <http://xbi.com.au/scpbib> and he runs regular live online training courses teaching people how to use Power BI <http://xbi.com.au/scpbi>

Part IV: AI and Power BI

Chapter 10: AI for Business Users in Dataflow and Power BI Desktop

Author: Leila Etaati

In this chapter, an overview of how to use AI in Power BI service and Power BI desktop will be discussed. For business users, always using AI is about easy access to the tools without writing any codes and activate AI capability with a couple of clicks. In this chapter two different possibility of consuming AI, will be discussed. First how as a business user we are able to analyse the text in Power BI service will be shown. In the next part, how using some AI powered visuals such as Key Influencer in Power BI desktop to analyse the data without knowing the machine learning concepts.

Cognitive Service in Power BI

Cognitive Services are an asset of APIs, SDKs and services that can be used by developers to add AI to their applications. Cognitive services in Azure has five major categories as Language, Vision, Speech, Search and Decision. Each of those has a selection of different services. These services have been used by many developers in web, and mobile applications. Moreover, there is a way to use them in Power BI reports [1].

One of the interesting services is about language analysis or Text Analytics. Text analytics can range from detecting the main keywords in a sentence, identify how much a customer is happy, detecting the main entity and structure of a sentence, detect the language of a sentence and so forth.

As a developer in Power BI Desktop, there is a way to use the Text Analytics API in Power Query [2]. However, this process is a bit hard for Business users or people who are not familiar with Power Query. In this section, First, how a business user is able to use cognitive services for the aim of text analytics and extract the main keywords of the text.

AI in Dataflow

There is an announcement about the availability of using Cognitive Services in Power BI Service, Dataflow. Dataflow is the new feature in Power BI service that allows people to transform their data in the cloud and do the ETL in the Power BI service.

To access the AI in Dataflow you need to have a premium account.

To start, you need to login to your Power BI Service account, and you need to create a new workspace (not “My Workspace”). You will not be able to create Dataflow in “My workspace” (Figure 10-01).

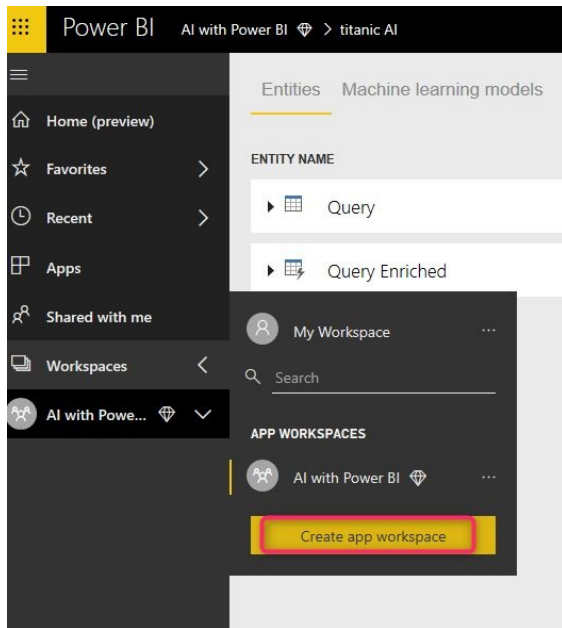


Figure 10-01: Create New Workspace in Power BI Service

To create a new workspace, click on the “Workspaces”, then click on the “Create app workspace”. Now, in the new page, you are able to create a new workspace (pro or premium). Now you can create a new one by putting the name, description and image for the workspace. So, choose the proper image, name, and description and click on the Save (Figure 10-02).

Create an app workspace

The screenshot shows a form titled "Create an app workspace". It includes an "Image" section with an "Upload image" button (annotated with a red circle 1) and a "Delete image" button. Below this is a "Name your workspace" text input field containing "Text Analytics with Dataflow" (annotated with a red circle 2). There is an "Available" checkbox which is checked. A "Description" text area contains the text "this an example for the using cognitive service in Power BI" (annotated with a red circle 3). A link "Learn more about workspace settings" is below the description. At the bottom, there is an "Advanced" dropdown menu and a "Save" button (annotated with a red circle 4) next to a "Cancel" button.

Figure 10-02: Create New Workspace by providing some Details

In the new page, you can see your new workspace and the welcome page, just skip the welcome page to navigate to the main workspace.

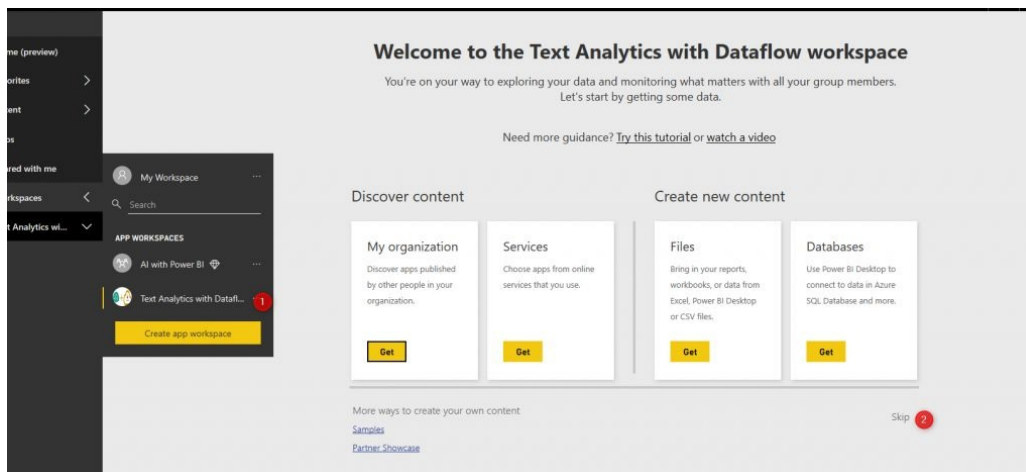


Figure 10-03: Workspace Welcome Page

In the main workspace page, beside Dashboard, Reports, Workbooks, and Datasets, we have a new tab named Dataflow. Click on Create option at the top

right of the page and choose the Dataflow.

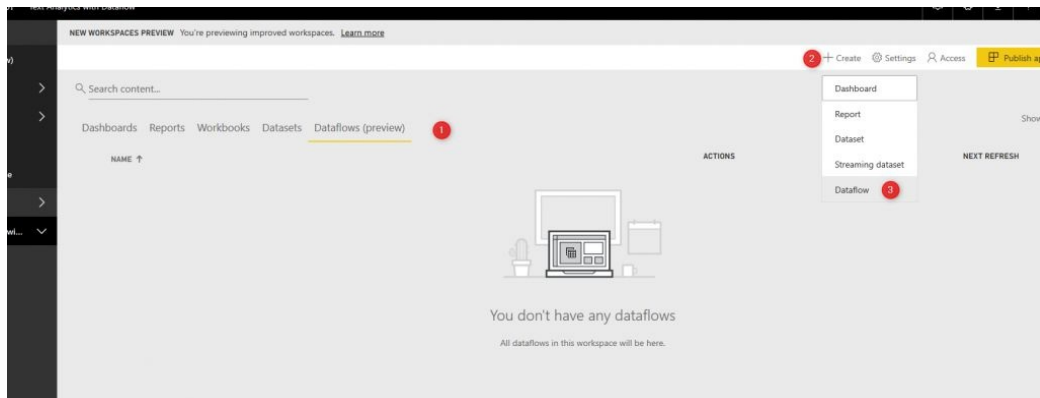


Figure 10-04: Create New Dataflow in Power BI Service

By creating a new Dataflow, Power BI Service navigates you to a new page that asks you about adding entities (data source) click on the “Define new entities” (Figure 10-05).

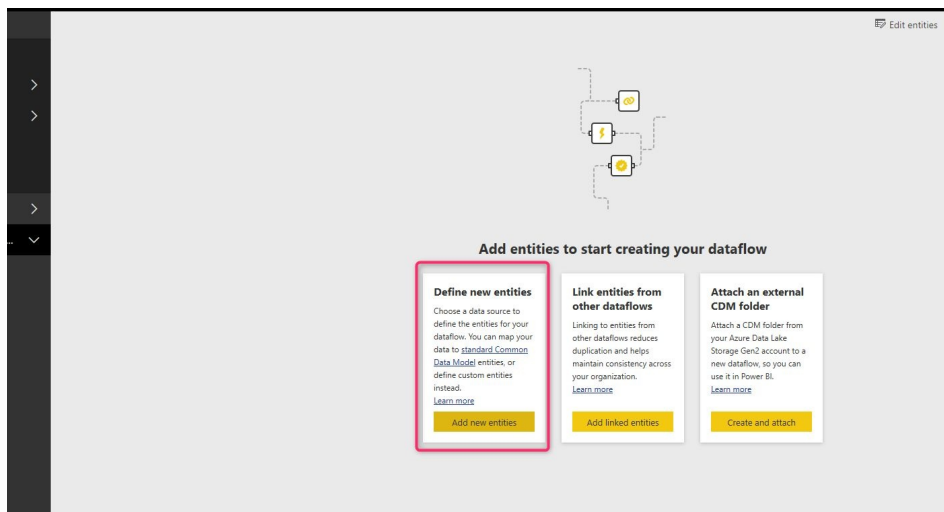


Figure 10-05: Define New entity page

In the next page, you can connect to different resources from cloud to on-premises. For this example, I just put some comments people put for some products, to access the data navigate to the files folder, data for chapter 17. Click on the Blank Table.

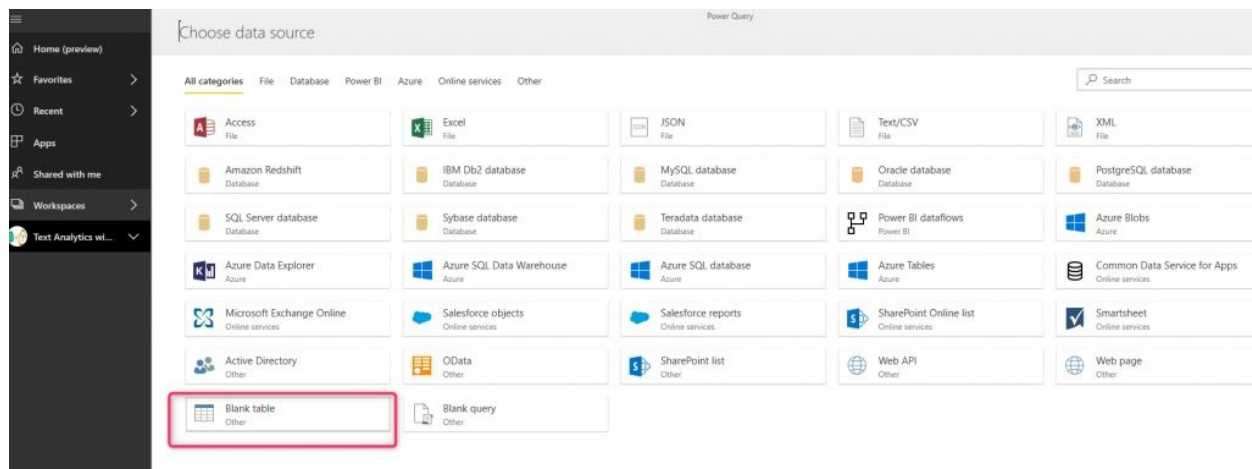


Figure 10-06: Create a Blank Table

Then change the title of the column to Comments, put the text for each row (row by row). then put a name for the table and click on the Next.

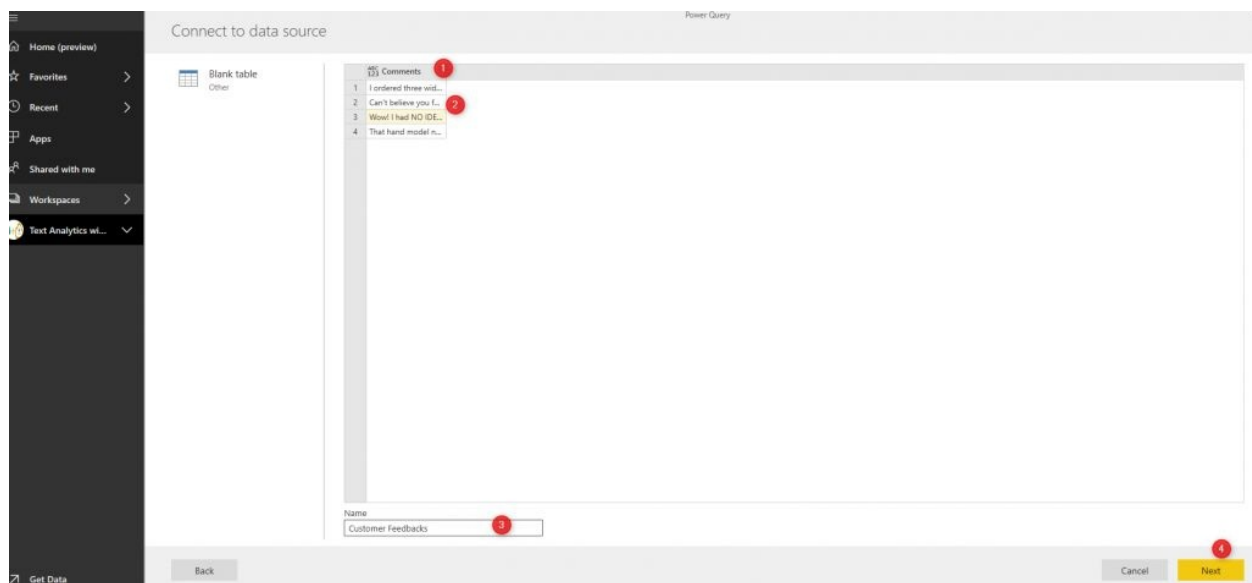


Figure 10-07: Create a Blank Table and Put the text

The new query will be generated and navigate into a new page as Power Query editor in Power BI web service. As you can see in the below Figure 10-08, It has some of the features of the Power Query Editor from transforming the columns, to combine and so forth. For the aim of text analytics, there is an icon on the top of the page as AI Insight, click on it.

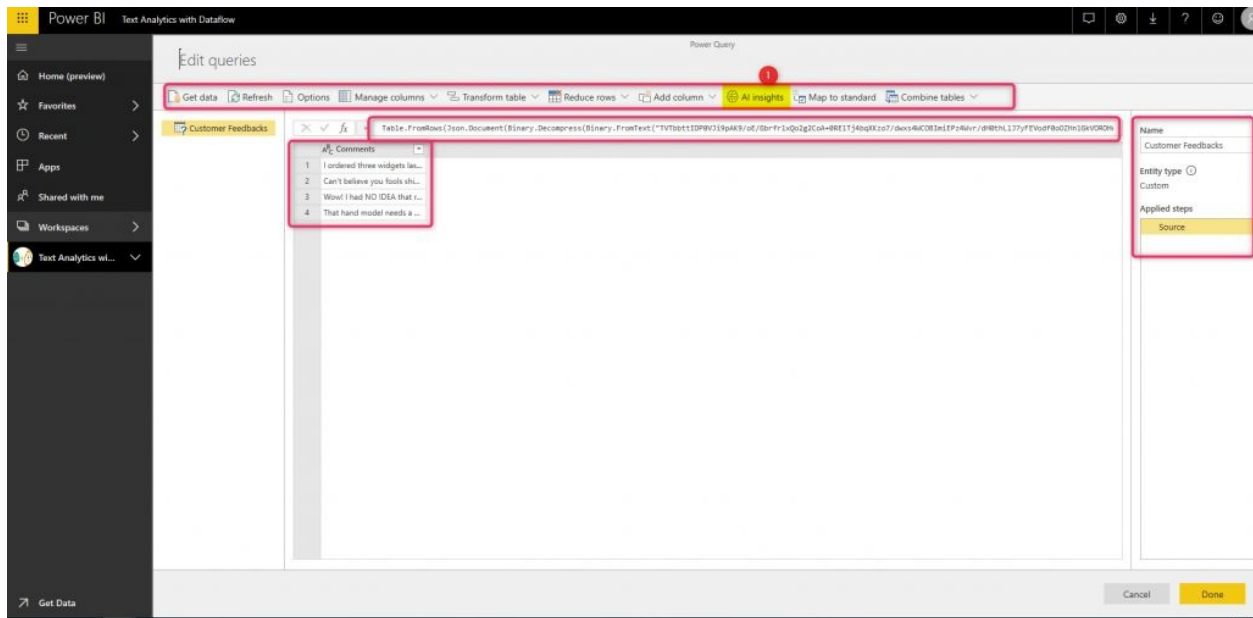


Figure 10-08: Edit Data in Dataflow

Then a new page will show up that you will see the cognitive service folder (if you already create one for Azure ML there should be a folder for Azure ML as well). Expand the cognitive services folder and you will be able to see four different cognitive services

- Sentiment Analysis
- Keyword Extraction
- Language Detection and
- Image Tag

Click on the last one that is *Sentiment Analysis*

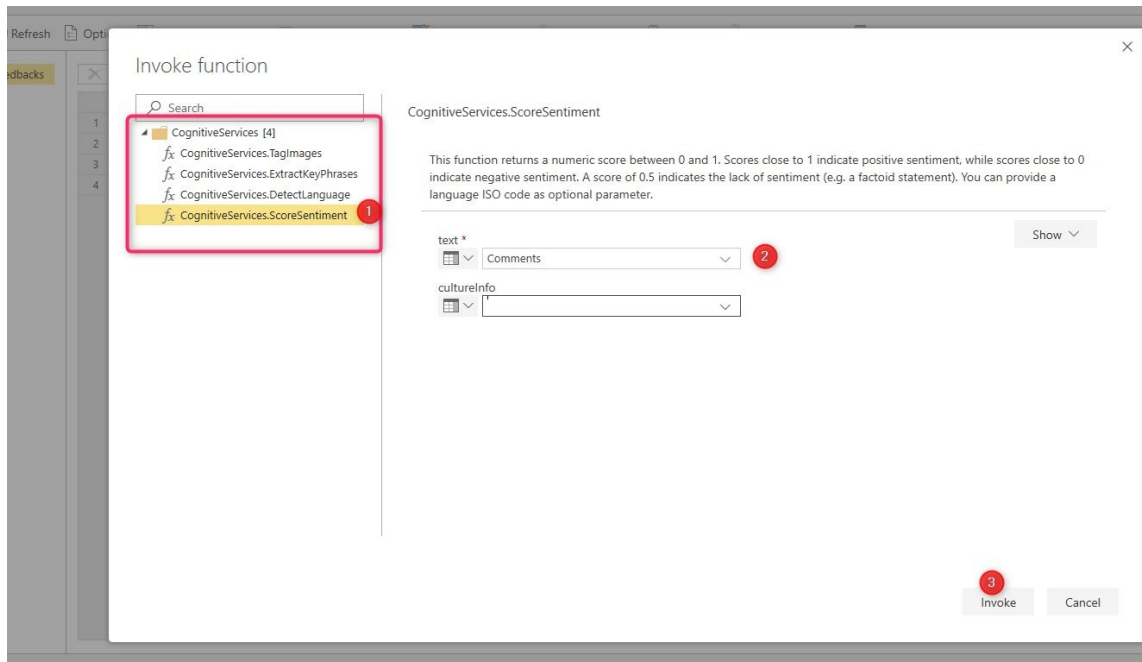


Figure 10-09: Cognitive Service in Dataflow

Sentiment Analysis is for identifying how much a customer is happy with products. The value will range from 0 to 1.

If the value is close to 1 that means the customer is happy about the product, otherwise if the value is close to 0 that means the customer is not happy about the product.

Choose the sentiment analysis service, it will ask about the target column as *Text*, then you need to choose the column you want to apply this service. In this example the *Comment* should be chosen as the target column.

Then, just click on *Invoke*. A new page will be shown that shows the original column (Comment) plus a newly added column name *Cognitive Service*. This new column has a value from 0 to 1 for each row.

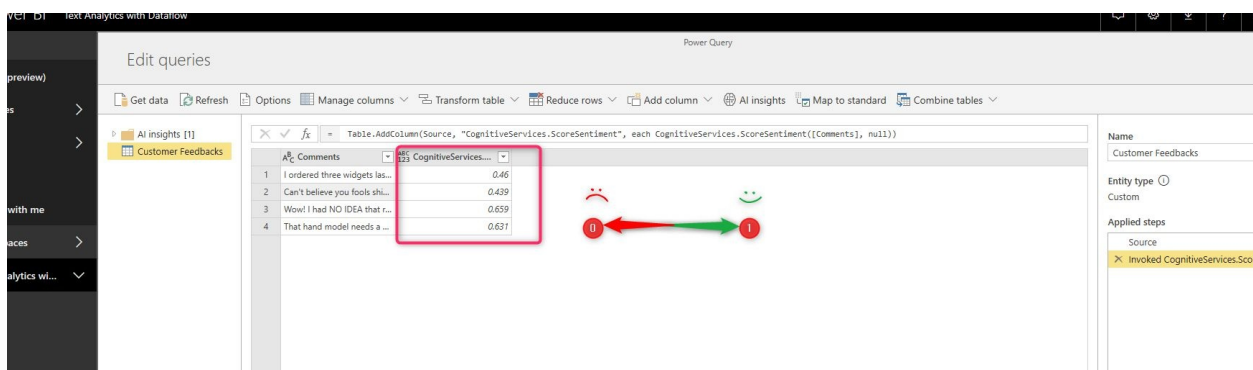


Figure 10-10: Invoke Sentiment Analysis in Power BI Service

Now we got the result, we need to save the result so we can use it in Power BI Service report or Power BI Desktop. Click on “done” at the bottom of the page to apply all changes. Then, you need to save the query by clicking on the top right of the page and put a name for the query.

Then, under the Dataflow you can see your new Dataflow. You just need to click on the Refresh Button. In this example, first I am going to get the data in Power BI desktop.

To do that, open your Power BI Desktop, and sign in with the account you used for the Dataflow. Click on Get Data, choose the Power BI dataflow, you may need to sign into your power BI service that you created the dataflow.

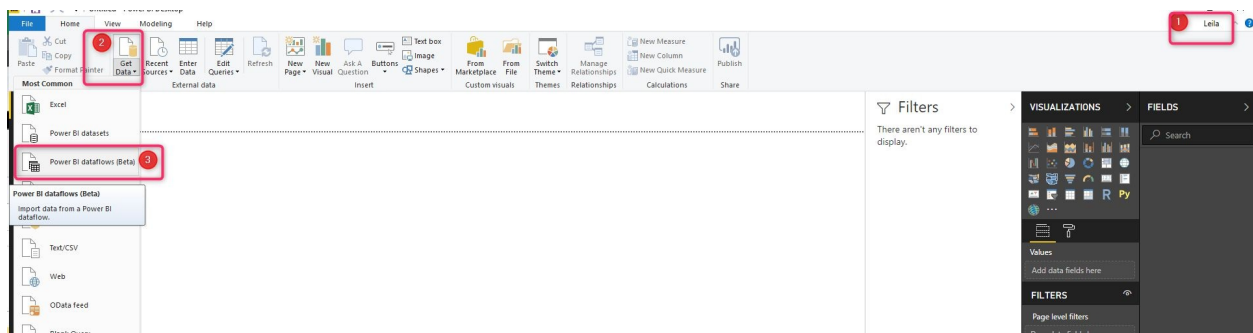


Figure 10-11: Connect to Dataflow from Power BI Desktop

After signing in, now you can see the list of Dataflow you already created. Expand the list and see an overview of the query with Comments and sentiment score columns. Now by loading the data you are able to create a report in Power BI Desktop.

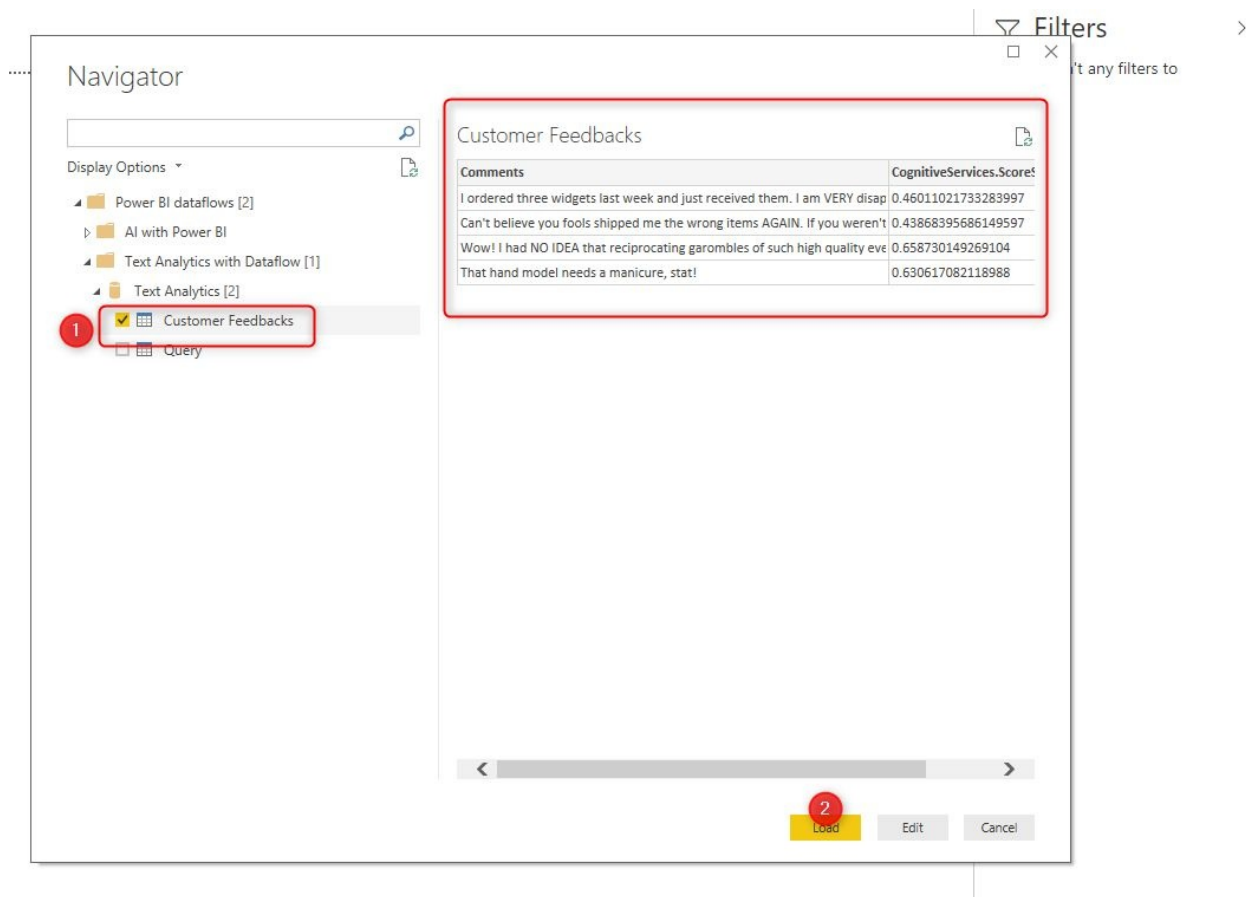


Figure 10-12: Dataflow in Power BI Desktop

You can also extract the Keyword and the language of the text following the same process.

However, for the image tagging you may need to follow some more steps. In the next section, I will explain the process.

Image Tag in Power BI

Image tag is another service in Cognitive Service that allows to extract the object in the image and for each detected object, it provides the confidence level.

To see a demo on this service you need to navigate to the

<https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/#analyze>

Then just import a picture and see the result. For example, I uploaded a picture of my dog and I got the below results.

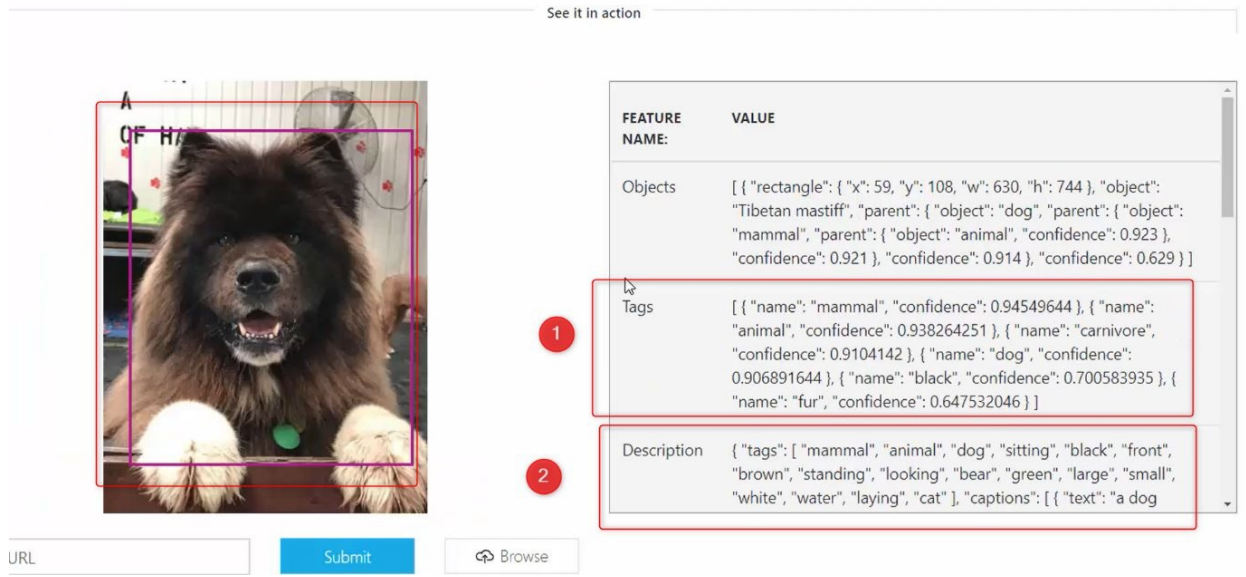


Figure 10-13: Image Tag in Cognitive Service

This service is available in Dataflow alongside with other language service. I already uploaded some of the pictures into my Azure Blob storage and in this section, I will show how to apply image tag on them using Cognitive Service function available in AI Insight.

I have some links to my pictures, also I need to write some code to extract them from the blob storage (Figure 10-14).

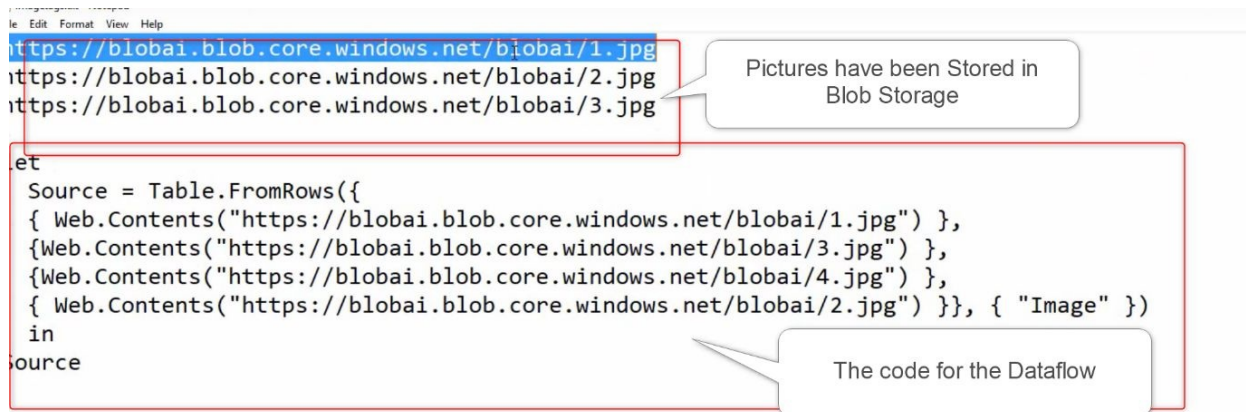


Figure 10-14: Image Links in Blob Storage and the Required Code to extract them.

Next, get data in Dataflow (same as the previous section), this time instead of choosing the get data from text, choose the get data from Blank Query.

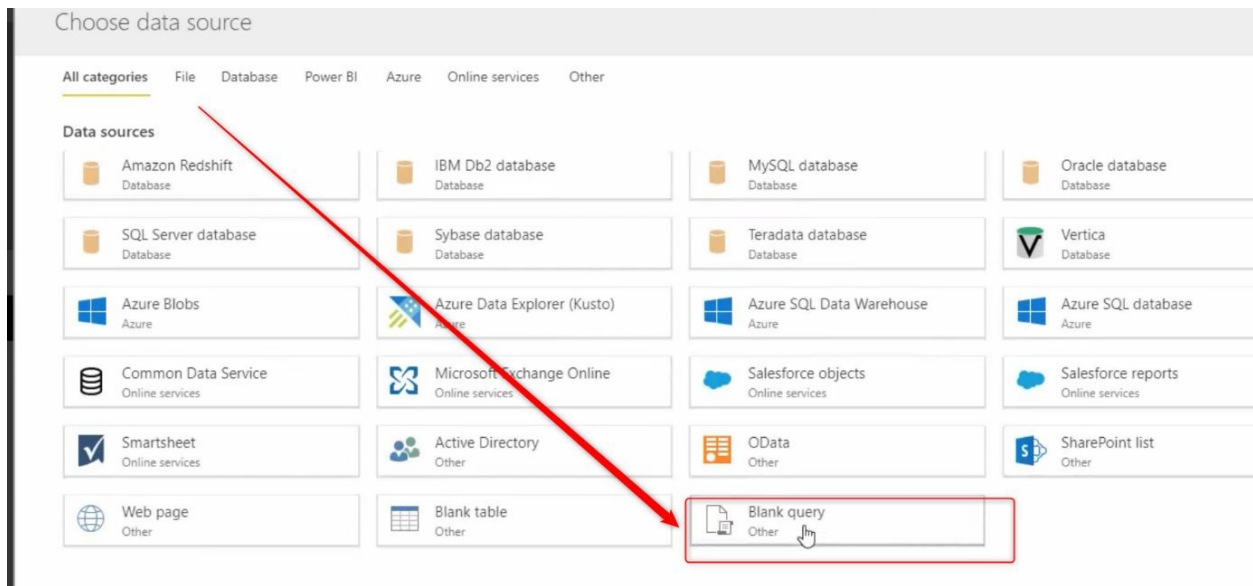


Figure 10-15: Get Data from Blank Query

In the new page delete the existing code and replace it with below code (you should have your own image link).

let

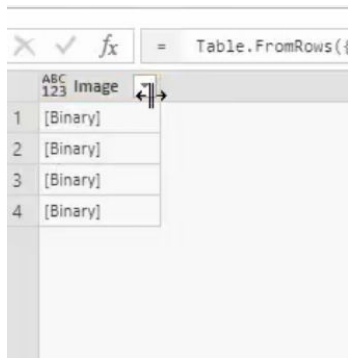
```
Source = Table.FromRows({ { Web.Contents("image address 1") },
{Web.Contents("image address 2") }, {Web.Contents("image address 3") }, {
Web.Contents("image address 4") }}, { "Image" })
```

in Source



Figure 10-16: Get Data Using Blank Query

After loading the data, you should see the imported image as a column with binary format.



The screenshot shows a data table with a header row and four data rows. The header row has columns labeled 'ABC' and 'Image'. The data rows show the value '[Binary]' in the 'Image' column for each row. The table is part of a larger application window with a formula bar at the top showing '= Table.FromRows({'. A small icon is visible next to the 'Image' column header.

ABC	Image
1	[Binary]
2	[Binary]
3	[Binary]
4	[Binary]

Figure 10-17: Loaded Image with Binary Format

Next, click on the AI Insight at the top of the page, and this time choose the Image Tag function.

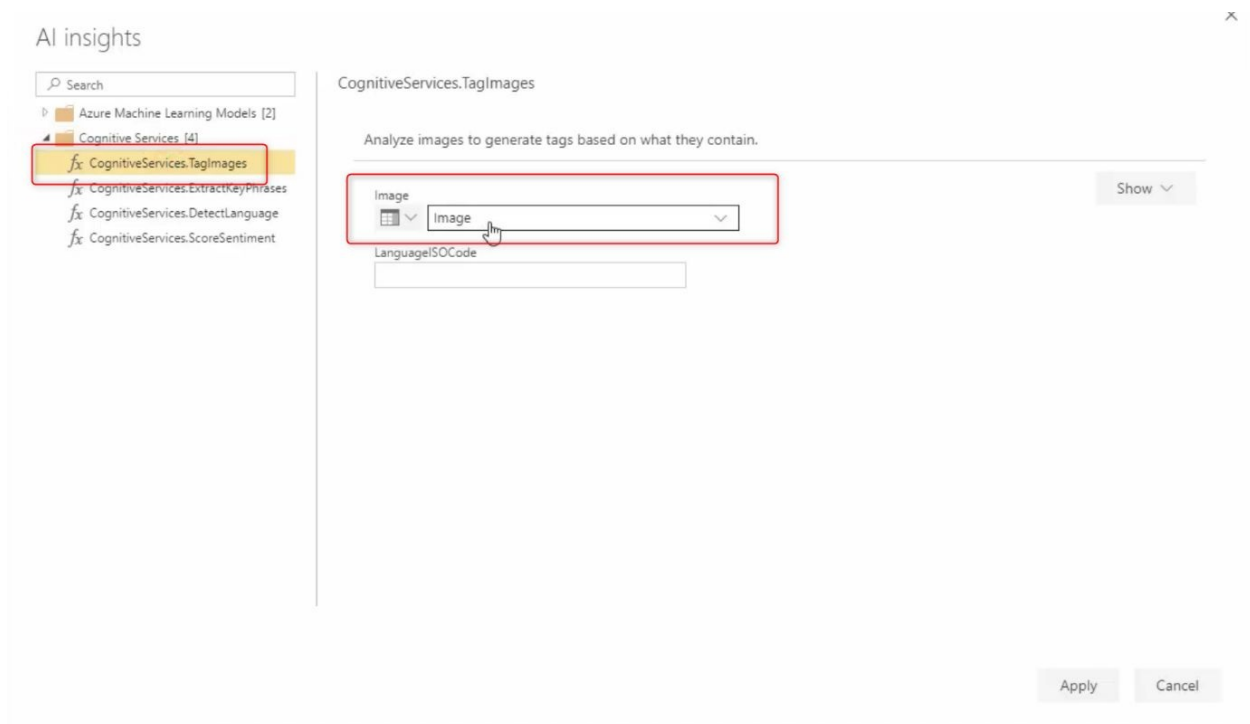


Figure 10-18: Image Tag Function

Select the Image column and click on “Apply” to see the results. Then you need to expand the result to see two columns: one is the image tag and another one has more details such as image tags and related confidence.

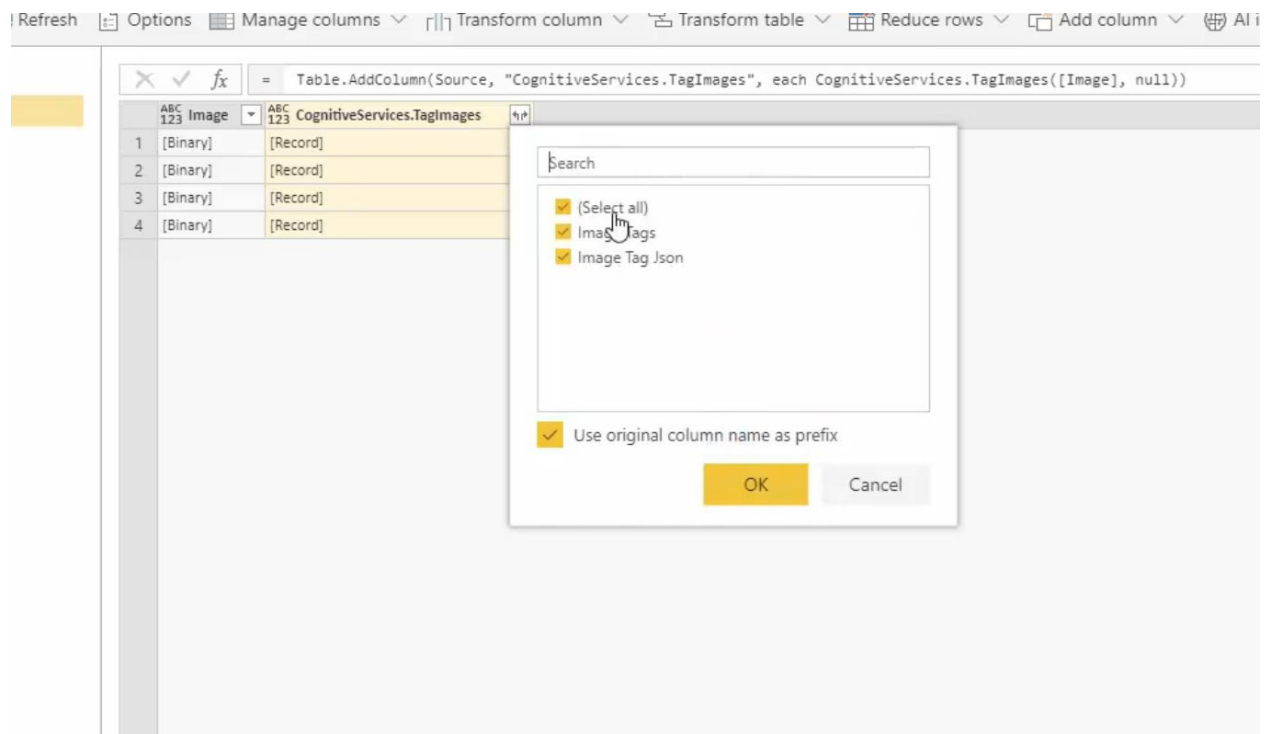


Figure 10-19: Image Tag Results in Dataflow

Just expand the results to see the tags and confidence. Same as text analytics, you can see the query in Power BI Desktop. You need to put a name and refresh the dataset in Dataflow, then open your Power BI Desktop and from “get data” choose the Dataflow. The result is like a JSON format, so you need to do some data transformation to see the confidence, but the Image tag column is already created.



Figure 10-20: Image Tag Results in Power BI Desktop

Key Influencer

In this section, I am going to demonstrate a new visualization that has been released by the AI team in Power BI in recent months. Before showing off this nice feature, there are some key points about this visual.

- It can be used by Data scientist, Data Engineer and End users
- It is easy enough to use and interpret
- It consumes lots of algorithms behind the scene to identify the main factor
- It can be used to align with other customer visuals to create a better visualization

Business Users



Data Engineer



Data Scientist



Figure 10-21: Who can use Key Influencers Visual

Moreover, this visual:

- Interpret both Categorical and Numeric variables
- Provides a great clustering approach: find the natural grouping on data, and then visualize the main top cluster (segment) and also, it shows how a combination of factors affects the metric that you're analysing.
- Explain the results: the visual provides a brief description of how it works [1]
- As mentioned before, this visual employs a combination of algorithms. In categorical and regression analysis different algorithms are used.



Figure 10-22: Analysing categorical and Numeric Values

In this section, I am going to use a dataset about concrete. Concrete is used in building the bridge, buildings and so forth.

The main elements for creating concrete is

- **Cement:** A cement is a binder, a substance used for construction that sets, hardens, and adheres to other materials to bind them together.
- **Blast Furnace Slag:** stony waste matter separated from metals during the smelting or refining of ore.
- **Fly Ash:** Fly ash or flue ash, also known as pulverized fuel ash in the United Kingdom, is a coal combustion product that is composed of the particulates that are driven out of coal-fired boilers together with the flue gases.
- **Water:** The amount of water in concrete controls many fresh and hardened properties in concrete including workability, compressive strengths, permeability and water tightness, durability and weathering, drying shrinkage and potential for cracking [2].
- **Superplasticizer:** Superplasticizers, also known as high range water reducers, are chemical admixtures used where well-dispersed particle suspension is required. These polymers are used as dispersants to avoid particle segregation and to improve the flow characteristics of suspensions such as in concrete applications [3].
- **Coarse Aggregate:** Coarse aggregate is the portion of the concrete which is made up of the larger stones embedded in the mix. Concrete contains three ingredients; Water, cement, and aggregate. That aggregate is made of fine sand and coarse gravel.

- Fine Aggregate
- Age: how many days

The dataset available from [here](http://archive.ics.uci.edu/ml/machine-learning-databases/concrete/compressive/Concrete_Data.xls): http://archive.ics.uci.edu/ml/machine-learning-databases/concrete/compressive/Concrete_Data.xls

So, let's start to predict what will be the strength of the concrete regarding other elements such as ashes, water, and so forth.

List of Questions

I want to answer the following questions:

- What factors have more impact on the strength of the concrete to decrease or increase and how much?
- I am interested in seeing the natural classification of my data.
- Also interested in seeing some rules like if the amount of Cement is ... and age is ... then what is the strength

Let's Answer these three questions using a brand-new visualization named **Key Influencers**.

Get it!

The key influencer is a preview feature, to access it you need to follow below steps

Click on File → Option and Settings → Options → then click under Global, click Preview Feature, and you should find the Key Influencer Visual at the bottom

You need to restart the Power BI (close and open again)

Just notice, it is a preview feature, some enhancement will be applied on it soon

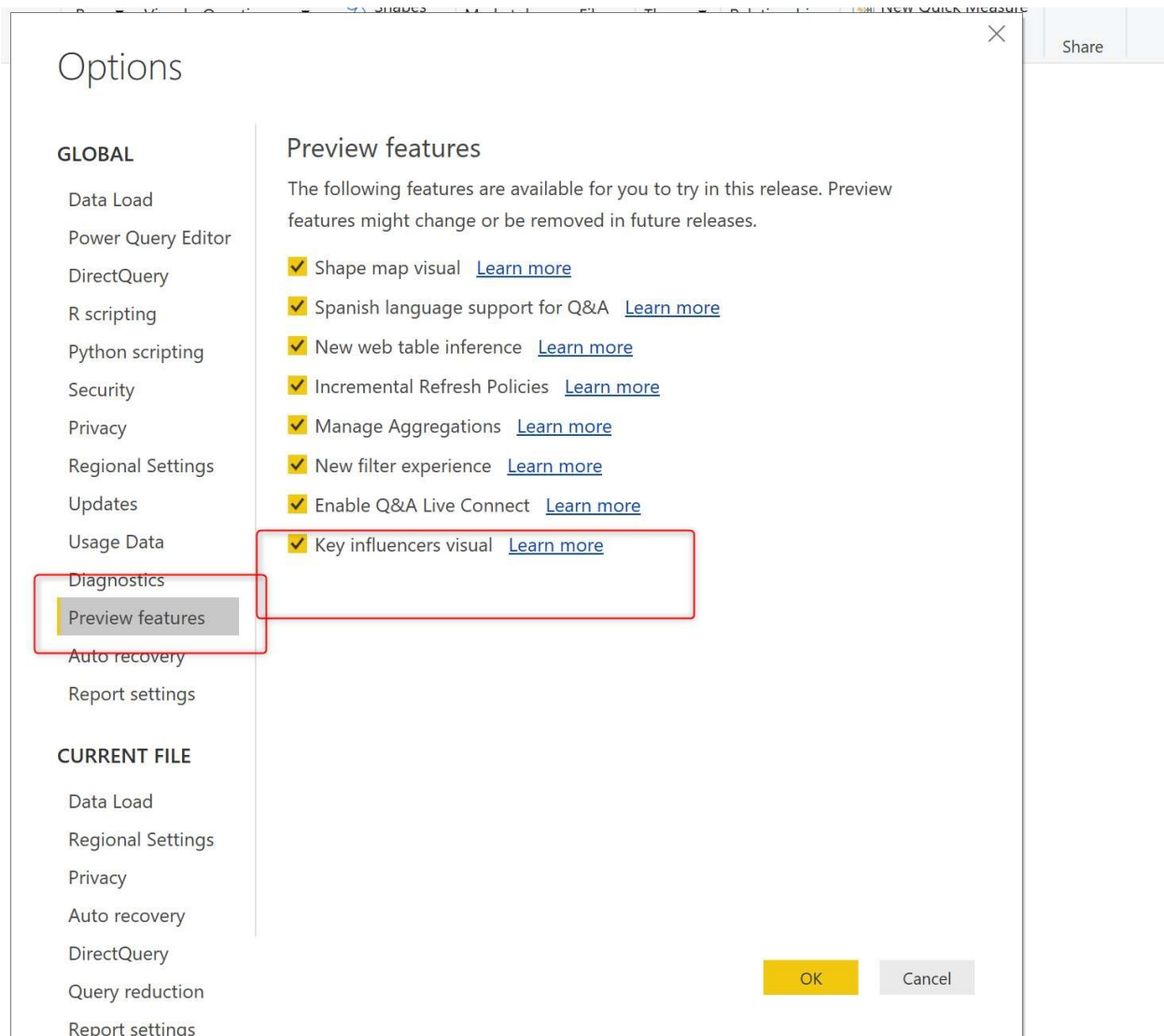


Figure 10-23: Enable Key Influencers Visual

Use It!

Now you need to import the concrete dataset into Power BI Desktop

Get Data → CSV → Load

Now Our plan is to analyse the strength of the concrete, hence click on visual that has been added to the Visualization panel, and for the analyse choose **Strength** field from **Concrete** dataset. You can see in the Figure 10-24.

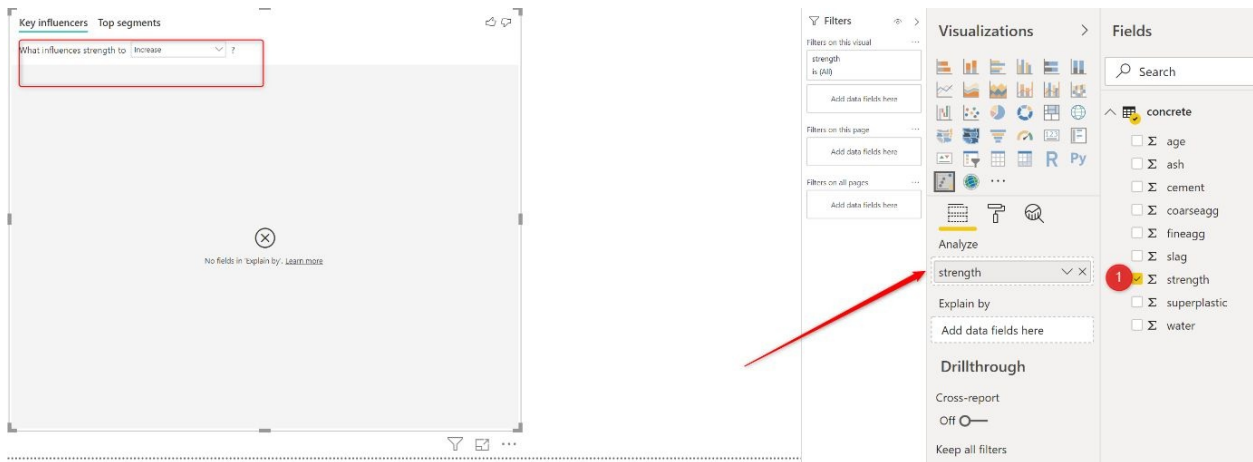


Figure 10-24: Load Data and Select which Column to Analyse

Now, you drop other columns into “Explained By” data field and see the visual show some analytics about what will impact on the strength of the concrete.

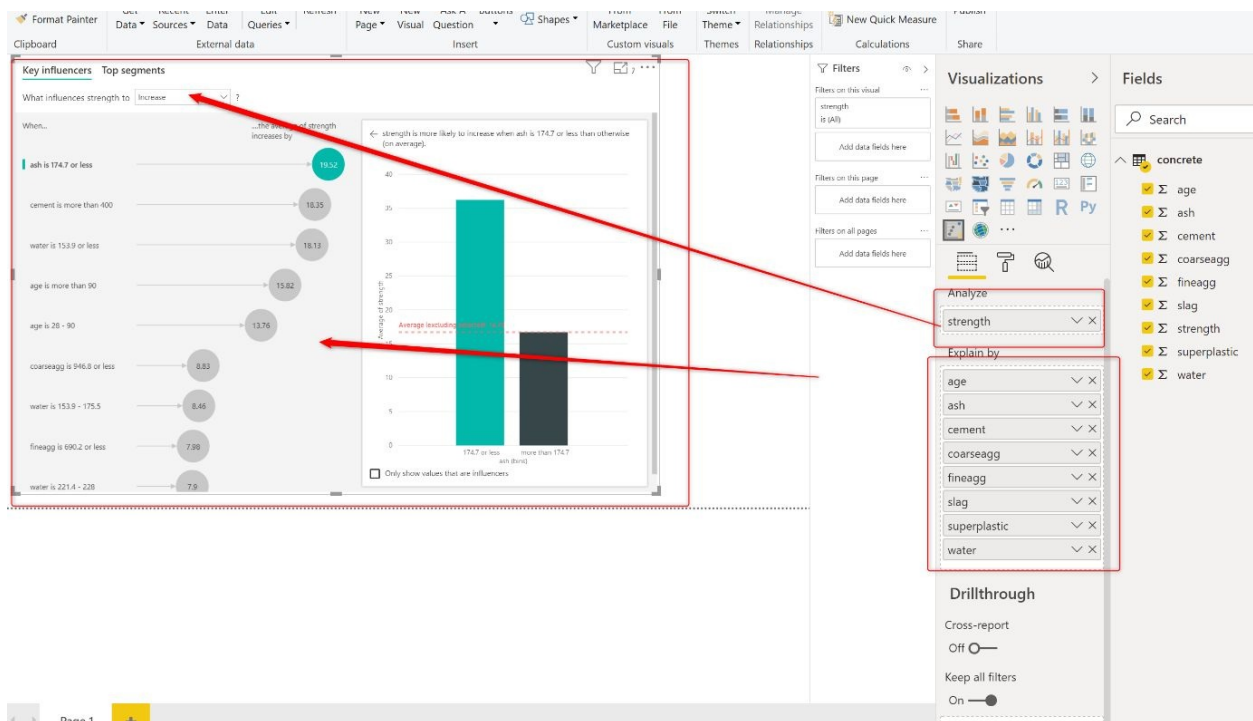


Figure 10-25: Fill out the Explained by Data Fields

The first important point, the list of the factors are displayed in the left panel with specific order. The one at the top has most impact on the strength of concrete than others. As you can see in the Figure 10-26, If the cement increases 400 then the strength will increase by 18.35 in another word it impacts by 4.5%.

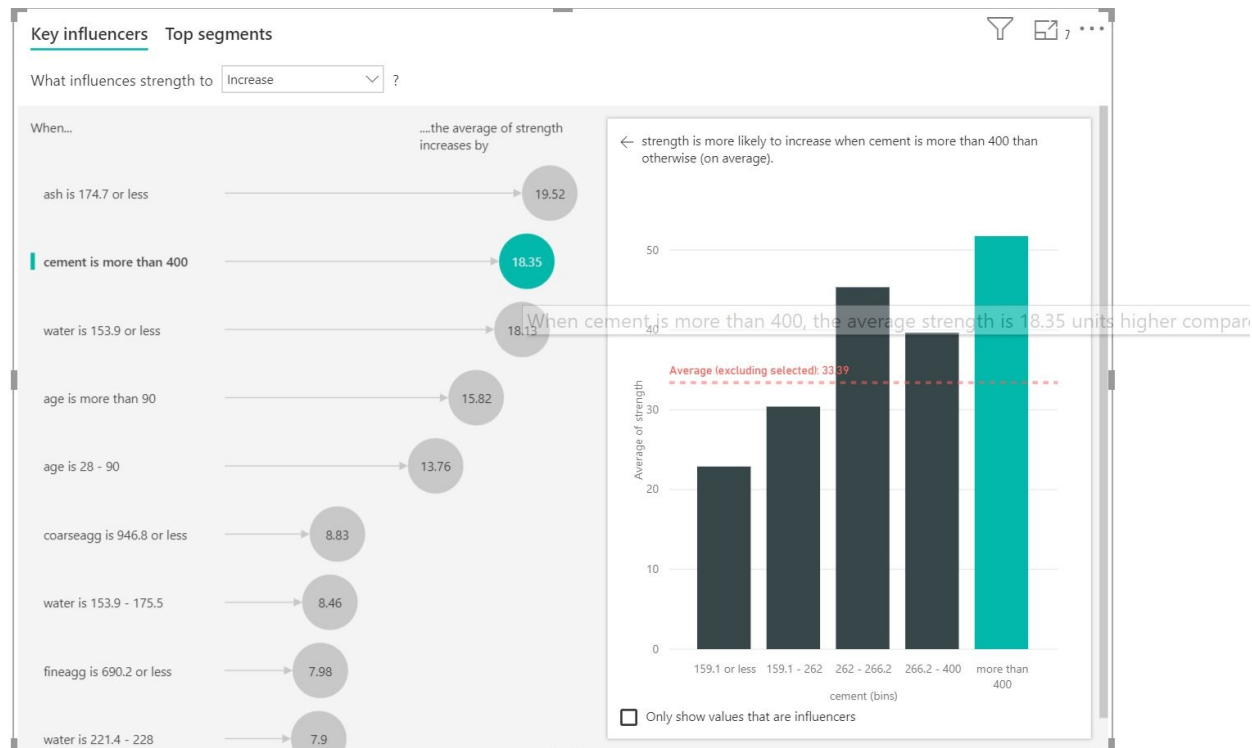


Figure 10-26: Concrete Strength Analysis by Key Influencers Visual

There is another analysis tab named Top Segments. Click on the Top segment at the top and choose the first segment with 61.12. As you can see in the Figure 10-27, in Segment 1, the average strength is 61.12 and the average age of cement is more than 3 days and amount of cement is between 159.1 and 262.

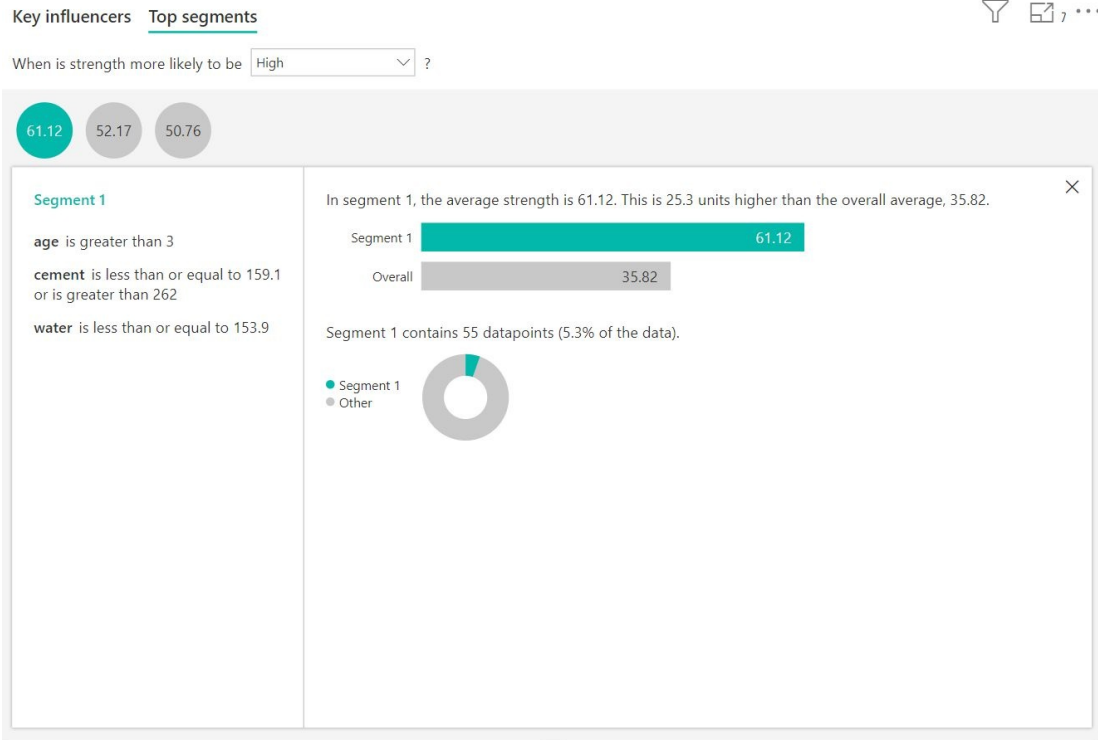


Figure 10-27: Top Segment Analysis

This visual uses the decision tree algorithms to extract the rules about which attribute will impact on the strength of cement. As a business user, you are not running any algorithm, and by using the Key Influencers visual the analysis will be automatically shown to you. This example was about a continues value (strength of cement) but you can also use it for the categorical variable analysis as well.

Summary

In this chapter, some of the features for AI that can be used by the business users without writing any code in Power BI service and Desktop have been explained. First, an explanation on how to apply text analytics on the customer feedback was discussed. How users can see the customer feedback in a numeric scale has been shown. Next, how we do image tagging has been presented. Finally, the analytical Key Influencers visual that provides more insight out of data has been presented.

All these features presented in this chapter can be done by Business Users, Developers and Data Scientists.

About the Author



[Leila](#) is the first [Microsoft AI MVP](#) in New Zealand and Australia. She has a PhD in Information System from the University of Auckland. She is the Co-director and data scientist in [RADACAD](#) Company with more than 100 clients around the world. She is the co-organizer of [Microsoft Business Intelligence and Power BI Use group \(meetup\)](#) in Auckland with more than 1200 members, She is the co-organizer of three main conferences in Auckland: [SQL Saturday Auckland](#) (2015 till now), [Difinity](#) (2017 till now) and [Global AI Bootcamp](#) 2-18. She is a Data Scientist, BI Consultant, Trainer and Speaker. She is a well-known International Speakers to many conferences such as Microsoft Ignite, SQL PASS, Data Platform Summit, SQL Saturday, Power BI World Tour and so forth in Europe, USA, Asia, Australia and New Zealand. She has over ten years' experience working with databases and software systems. She was involved in many large-scale projects for big-sized companies. Leila is an active [Technical Microsoft AI blogger](#) for RADACAD.

Chapter 11: AI in Power BI Desktop

Author: Markus Ehrenmüller-Jensen

AI is everywhere – and now even included in Power BI Desktop. Sometimes AI might be very apparent when you enrich your data with predictions by explicitly calling an Azure Machine Learning web service in Power Query. Sometimes it might be hidden in an analytic feature offered by Power BI's interface.

No matter if you are a business user, analyst or data scientist – Power BI has AI capabilities tailored to you. This chapter will cover how you can integrate and leverage the use of languages DAX and R, and how to integrate Azure Cognitive Services and Azure Machine Learning Services to make more out of your data.

Introduction

Microsoft is currently investing heavily in making Artificial Intelligence (AI) available for everybody. After Self-service BI we are now facing Self-service AI. In this chapter we will concentrate on the capabilities of Power BI Desktop (not the Power BI Service, as in the previous chapter). While the possibilities are almost endless, I will concentrate on two AI functionalities, linear regression and text mining, and walk you through the steps necessary to implement both with different features within Power BI Desktop: Visuals & Analytics, DAX, R, Azure Cognitive Services, and Azure Machine Learning Services. While the chapter's limitations in term of pages do not allow for a comprehensive introduction to neither Linear Regression and Text Mining, nor to DAX, R, Azure Cognitive Services and Azure Machine Learning Services, it gives you an impression of what steps you need to take and of how to get started. Take the examples as templates from which you can derive solutions for problems of your own organisation.

Linear Regression

Linear regression is a quite simple mathematical formula to create e. g. a trend line for a measure over time. In our example I will use simple linear regression for sales amount over order date, like the straight blue line in this screenshot:

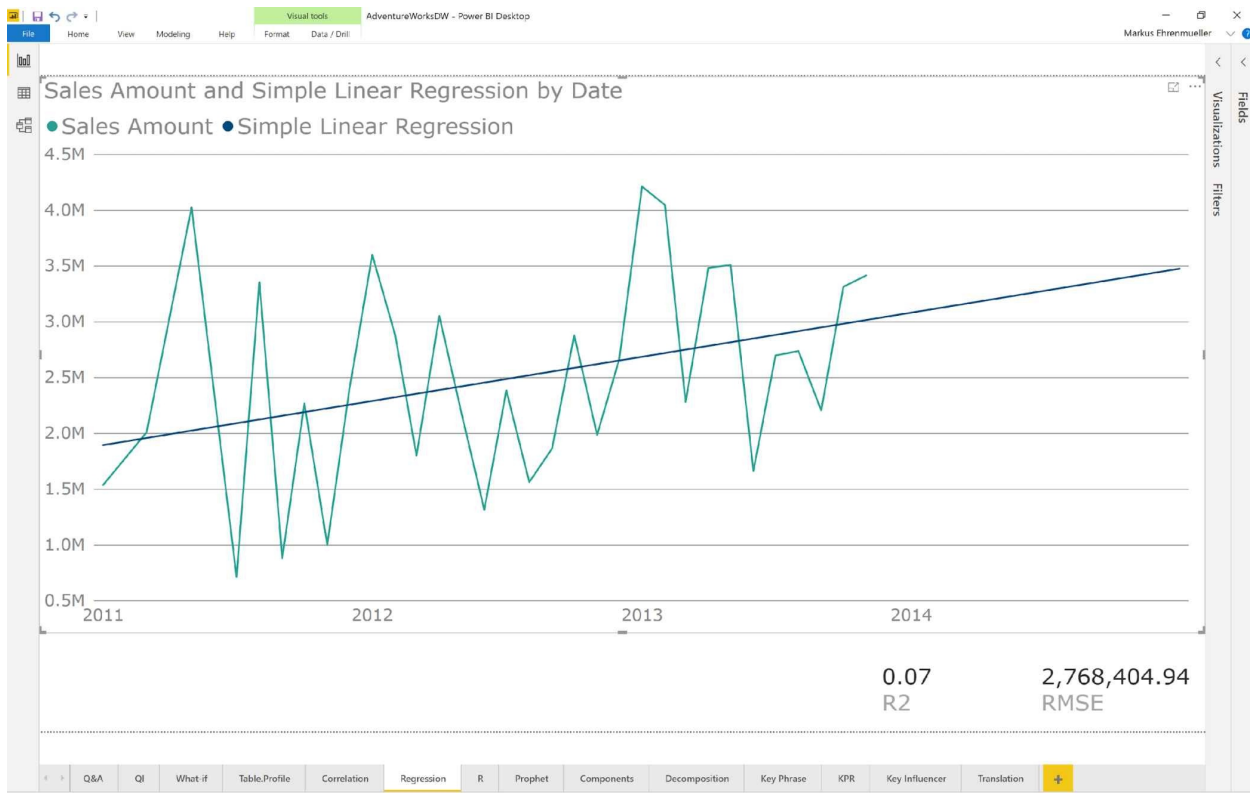


Figure 11-01: Sales Amount and Simple Linear Regression by Date

While Simple Linear Regression has its limitations and assumptions, it is simple enough to understand it with mere basic mathematical knowledge and allows you to learn about the data (in our case: the correlation between sales amount and the order date). The formula is:

$$Y = \text{Intercept} + \text{Slope} * X$$

While: **Intercept** is the amount for Y where the line crosses the Y -axis. And **Slope** states how much Y increases for every increase of X (for positive values of **Slope**; a negative **Slope** would mean a decreasing value of Y for every increase of X).

In plain English: When X is zero, Y is calculated as the **Intercept**. This might not be an appropriate assumption in all cases (e. g. because there is not a real-

world zero-value for our order date). And **Slope** tells us, how much **Y** increases (or decreases for that matter) every single day over the order date.

The actual values for **Intercept** and **Slope** can be calculated via the following formulas:

$$\text{Slope} = \frac{N * \sum (x * y) - \sum x * \sum y}{N * \sum x^2 - (\sum x)^2}$$

$$\text{Intercept} = \frac{\sum y}{N} - \text{Slope} * \frac{\sum x}{N}$$

Now let's look how we can apply this in a useful way:

- Analytic Line
- DAX
- R Visual

Analytic Line

For certain visualisations Power BI offers us to apply Analytics. In the case of a line chart we can apply the following:

- Trend line
- Constant line
- Min line
- Max line
- Average line
- Median line
- Percentile line

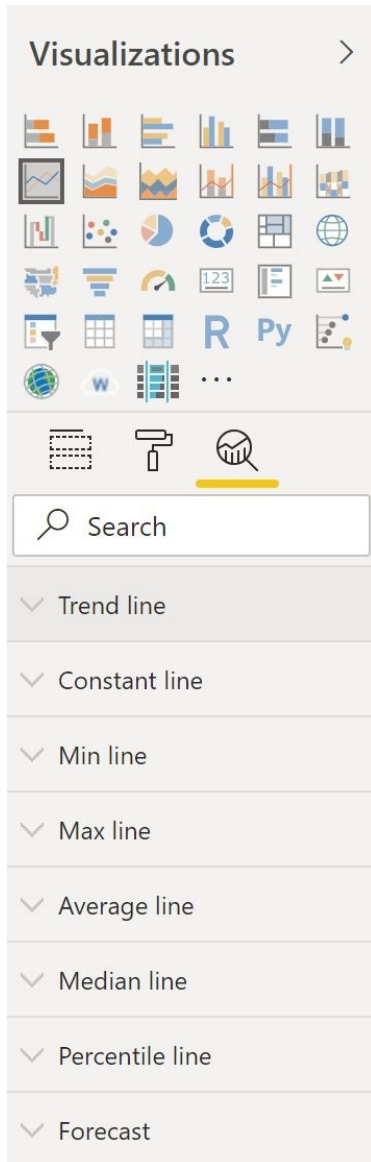


Figure 11-02: Analytic Trend Line

The first entry, trend line, automatically calculates and displays a simple linear regression line, as discussed above and as we can see in the following screenshot as a straight, dotted black line.

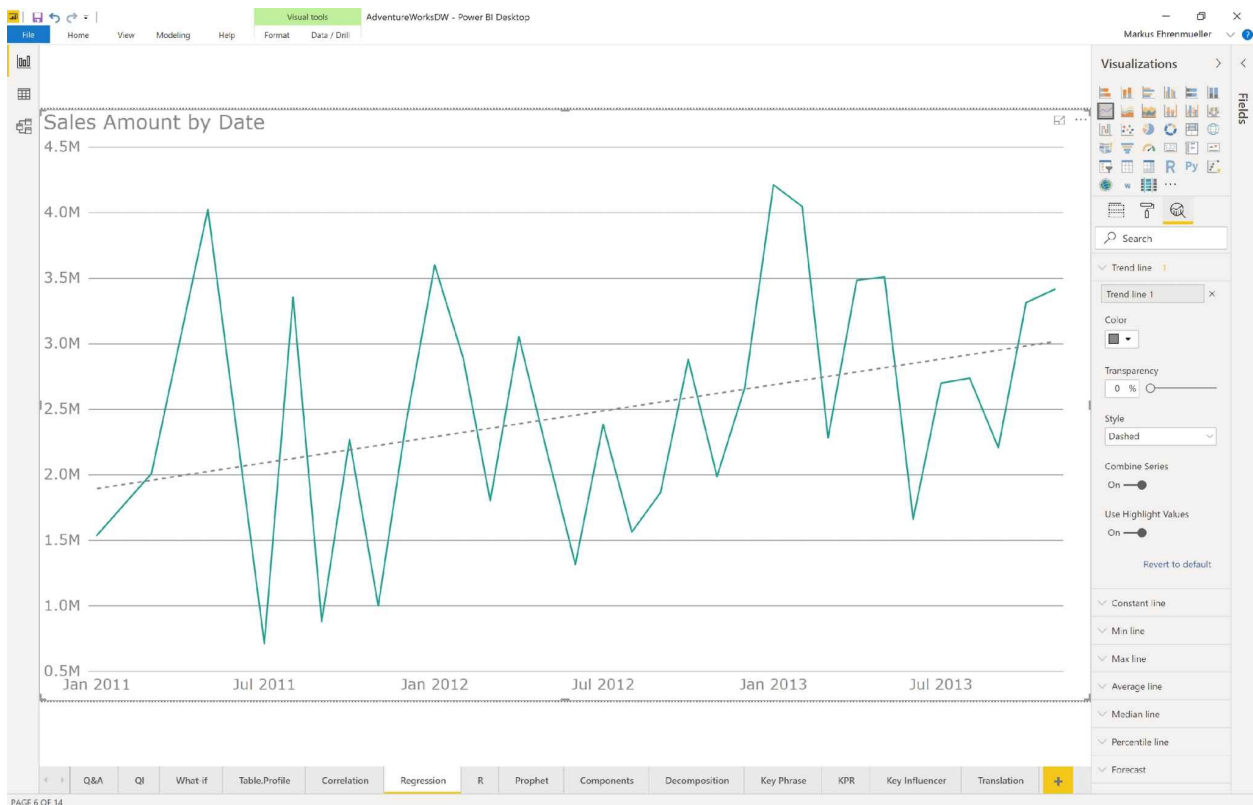


Figure 11-03: Sales Amount by Date and Analytic Trend Line

To apply the trend line, we don't even have to know what simple linear regression is or how to calculate it. That's good for that matter. The only disadvantage is: Even when we know how to calculate it, we cannot influence the calculation of the trend line in any way. If you are satisfied with the trend line how it is, then go for it. If you are not satisfied with the calculation, you can't do anything here, but make use of DAX or R, which is explained below.

Exercise: Try out the other available analytic lines as well. In what use cases they could be of help for your existing reports?

DAX

Daniil Maslyuk describes an implementation of simple linear regression in DAX in his blog (<https://xxlbi.com/blog/simple-linear-regression-in-dax/>). The DAX formula is quite lengthy, text-wise, as Daniil made use of variables, which helps to understand the single elements of the formula:

Simple Linear Regression =

```
/* Simple linear regression via DAX by Daniil Maslyuk
* https://xxlbi.com/blog/simple-linear-regression-in-dax/
*/
```

```

VAR Known =
    FILTER (
        SELECTCOLUMNS (
            ALLSELECTED ( 'Date'[Date] ),
            "Known[X]", 'Date'[Date],
            "Known[Y]", [Sales Amount]
        ),
        AND (
            NOT ( ISBLANK ( Known[X] ) ),
            NOT ( ISBLANK ( Known[Y] ) )
        )
    )
VAR Count_Items =
    COUNTROWS ( Known )
VAR Sum_X =
    SUMX ( Known, Known[X] )
VAR Sum_X2 =
    SUMX ( Known, Known[X] ^ 2 )
VAR Sum_Y =
    SUMX ( Known, Known[Y] )
VAR Sum_XY =
    SUMX ( Known, Known[X] * Known[Y] )
VAR Average_X =
    AVERAGEX ( Known, Known[X] )
VAR Average_Y =
    AVERAGEX ( Known, Known[Y] )
VAR Slope =
    DIVIDE (
        Count_Items * Sum_XY - Sum_X * Sum_Y,
        Count_Items * Sum_X2 - Sum_X ^ 2
    )
VAR Intercept =
    Average_Y - Slope * Average_X
RETURN
    SUMX (
        DISTINCT ( 'Date'[Date] ),
        Intercept + Slope * 'Date'[Date]
    )

```

)

Figure 11-04: Simple Linear Regression in DAX

When you use the DAX measure as it is, you will end up with the very same trend line as with the analytic line in the example above, but with two exceptions. One, the trend line continues over the whole of the dates available in your date-table (even if there are not actual values available for that time-frame). In the screenshot (Figure 11-1) the blue trend line is continued until end of 2014: Second, we can freely influence the trend line by pushing the right buttons. In the following example I introduced the following:

a) A filter for the calculation of variable *Known*. Therefore the calculation of the trend line is not based on all of the available actual sales, but limited to sales from 2013 only. As the trend for year 2013 is negative, the trend line now points down.

And b), another filter in the *RETURN* expression, as well. This filter returns values for dates after January 1 2013, only. Therefore, the trend line starts not anymore at the very left of the chart, but in year 2013. I marked the changes in bold font:

Simple Linear Regression 2 =

/* Simple linear regression 2 via DAX

* based on formula by Daniil Maslyuk

* <https://xxlbi.com/blog/simple-linear-regression-in-dax/>

*/

VAR Known =

FILTER (

SELECTCOLUMNS (

FILTER (ALLSELECTED ('Date'[Date]); 'Date'[Date] >=

DATE(2013; 01; 01));

"Known[X]", 'Date'[Date],

"Known[Y]", [Sales Amount]

),

AND (

NOT (ISBLANK (Known[X])),

NOT (ISBLANK (Known[Y]))

)

)


```

VAR Count_Items =
    COUNTROWS ( Known )
VAR Sum_X =
    SUMX ( Known, Known[X] )
VAR Sum_X2 =
    SUMX ( Known, Known[X] ^ 2 )
VAR Sum_Y =
    SUMX ( Known, Known[Y] )
VAR Sum_XY =
    SUMX ( Known, Known[X] * Known[Y] )
VAR Average_X =
    AVERAGEX ( Known, Known[X] )
VAR Average_Y =
    AVERAGEX ( Known, Known[Y] )
VAR Slope =
    DIVIDE (
        Count_Items * Sum_XY - Sum_X * Sum_Y,
        Count_Items * Sum_X2 - Sum_X ^ 2
    )
VAR Intercept =
    Average_Y - Slope * Average_X
RETURN
    CALCULATE (
        SUMX (
            DISTINCT ( 'Date'[Date] ),
            Intercept + Slope * 'Date'[Date]
        );
        FILTER ( 'Date'; 'Date'[Date] >= DATE(2013; 01; 01) )
    )

```

Figure 11-05: Simple Linear Regression in DAX with filters applied

Exercise: Create a DAX measure for each of the other available analytic lines (e. g. min line, max line, average line, etc.). Change the formula a) to adopt the calculation to only a certain time frame (e. g. calculating the min line for year 2013 only) and b) to adopt the length of the line to only a certain time frame (e. g. showing the min line only along 2013).

R Visual

Note: While the R visual is a standard visual in Power BI and automatically available, you need a local instance of R installed on your computer. Please follow instructions at <https://mran.microsoft.com/> on how to install (a free copy) of R. As I make use of libraries `ggplot2`, `scales` and (later) `tm` in my examples, make sure to install those libraries as well. I would recommend to install an R IDE (e. g. R studio) and to run the following statement:
“`install.packages("ggplot2", "scales", "tm")`”.

In the following example, I created a *R visual*, which runs the script listed below. I make use of the `ggplot2` library to visualize the actual sales amount over date (green line). Via function `stat_smooth` I let `ggplot()` calculate a simple regression line (“`method=lm`”), shown as blue line. By default, `ggplot` is showing a confidence interval (grey area along the blue line) as well, which can be turned off (by adding parameter “`se=FALSE`” to `stat_smooth`).



Figure 11-06: Simple Linear Regression in an R Visual

Simple Linear Regression in R Visual

data cleaning

FactResellerSales <- dataset

```
colnames(FactResellerSales) <- c("OrderDate", "SalesAmount")
FactResellerSales <- FactResellerSales[FactResellerSales$OrderDate!="",]
FactResellerSales$OrderDate <- as.POSIXct(FactResellerSales$OrderDate)
```

```
# adding a regression line
```

```
library(ggplot2)
```

```
library(scales) #generic plot scaling methods
```

```
ggplot(FactResellerSales,
       aes(x = OrderDate,
           y = SalesAmount)) +
  geom_line(color = 3) +
  theme(text = element_text(size = 18)) +
  scale_y_continuous(limits = c(0, 5000000), labels = comma) +
  theme(legend.position = "none") +
  labs(x = "Order Date",
       y = "Sales Amount") +
  stat_smooth(method=lm) # linear model / linear regression
```

Figure 11-07: R script to display sales amount over date and a simple linear regression line

Exercise: With the search engine of your choice find out which other methods `stat_smooth` allows. Try them out and evaluate if those methods give a better fitting line.

Summary

In this chapter we looked into different implementations of simple linear regression. We added a trend line to our line chart via Power BI's analytic features. This is easy to apply, but does not offer much flexibility, according to the calculation and display of the line. To achieve flexibility, we then calculated simple regression via the help of both, DAX and R. The implementation in DAX is straight forward, if you are used to writing DAX. The implementation in R was done with `stat_smooth` function available as part of the `ggplot` library.

Text Mining

In this example we are going to analyse the product description column. The description consists of shorter and longer texts. I combined the product's *EnglishProductName* and *EnglishProductDescription* into a new column *EnglishProductNameAndDescription*, as the description column might be empty for some of the columns, while there is always a product name.

If you add the description to an ordinary table visual, it is hard to compare different products or get an idea how those descriptions differ between different product categories. Therefore, we are going to “mine” the content of the column.

Word Cloud

I assume, that the more important a term in a text is, the more often it will appear in the text. A word cloud visual counts how often a term is mentioned. Terms with a higher count are printed in a bigger font size and appears more towards the centre than the others. I've included a screenshot below:

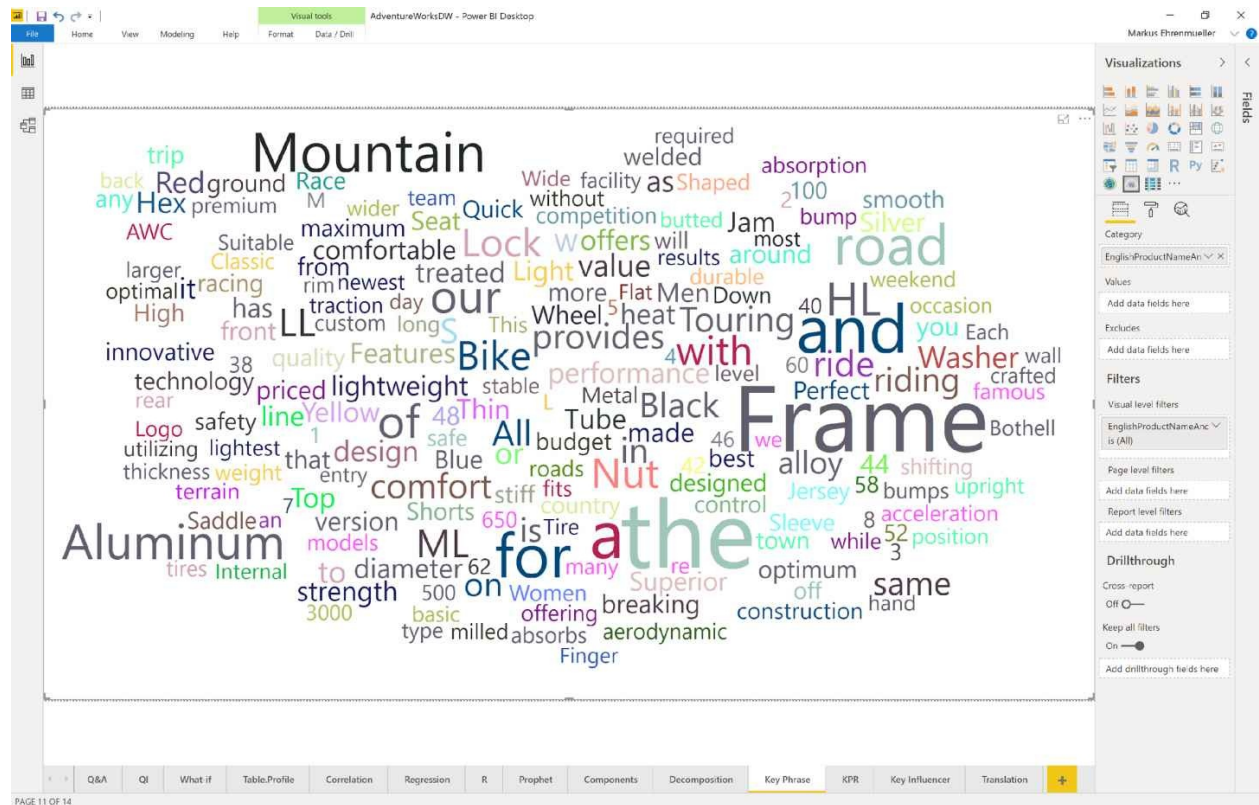


Figure 11-08: Word cloud of column EnglishProductNameAndDescription

This is easy to apply but comes with strings attached: In the English language certain words, like “and”, “the”, “a”, “for”, “on”, “of”, or “with” are very common. Therefore, they are also more common in the product descriptions. But

as they are not special for a certain product category, we would rather like to remove them from the word cloud. That's what we are going to do with the help of the following technologies:

- Azure Cognitive Services
- Azure Machine Learning
- R Visual

Azure Cognitive Services

Azure Cognitive Services is Microsoft's offer for so-called pre-trained models. Microsoft's data science team is developing and publishing an ever-growing amount of different services, which are used internally as well (e. g. to localize the online documentation at docs.microsoft.com). "Pre-trained" means, that Microsoft took care of tuning the models. This guarantees a high level of quality for lot of use cases. Currently Azure Cognitive Services offers different features like decision-making, vision, speech, search, and language. The latter offers services like Text Analytics, Translator Text, QnA Maker, Language Understanding, and Immersive Reader. For our use case we will leverage "key phrase", which is part of the "Text Analytics" offer.

To make use of Azure Cognitive Services you need to go through the following steps:

- Sign-in to portal.azure.com and add "Cognitive Services" to one of your subscriptions.
- For later reference we will need one of the both generated keys and the API endpoint. Both can be obtained by opening the newly created Cognitive Services.
- I created a parameter in Power Query for both: *apikey* and *endpoint*. This makes it easier later in case you have several references to both and want to change them.

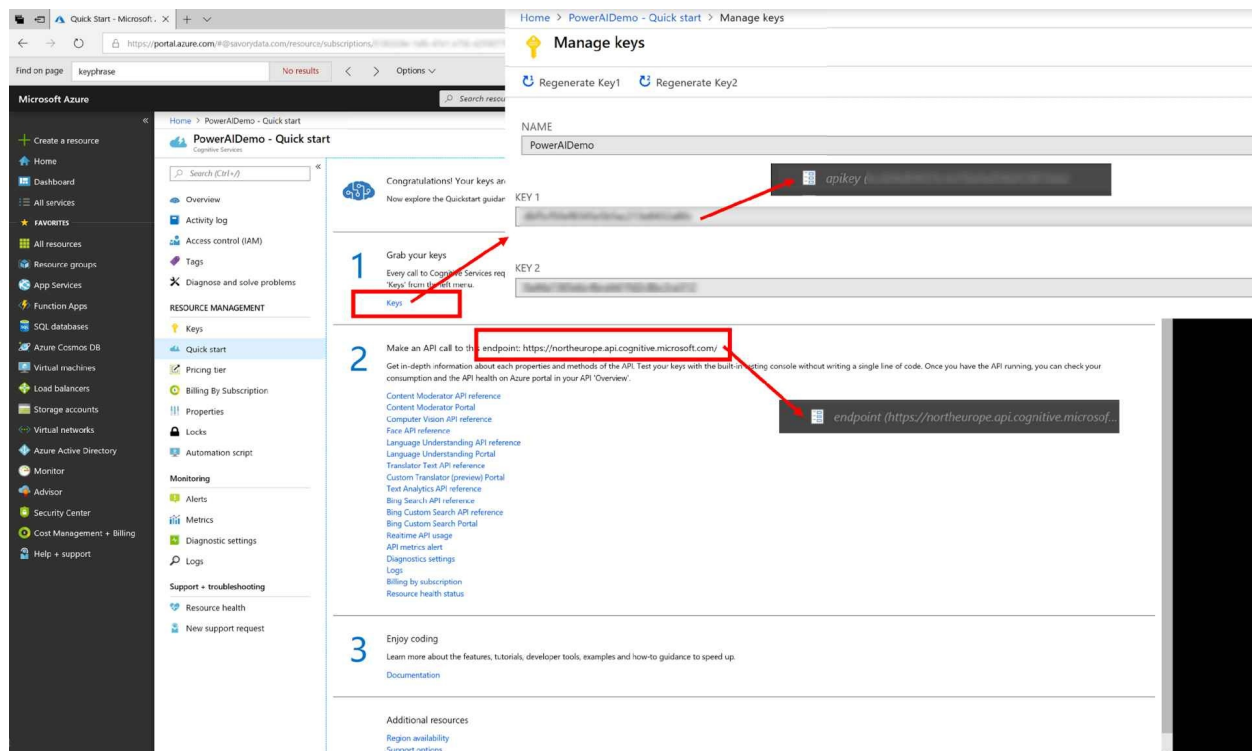


Figure 11-09: Where to find API key and endpoint of Cognitive Services in Azure's portal

Create a function in Power Query by creating a new query and choosing “Blank query”. Then you insert the following script in the *Advanced editor*. I took this script from Microsoft’s online documentation and changed the assignment of `apikey` and `endpoint` to use the Power Query parameters, created in the step above. The script is accepting one parameter (`text`), converting it into a JSON format, sending it to Cognitive Services by using `apikey` and `endpoint` and converting the returned JSON value back into a textfield. Make sure to rename the query to “Keyphrase API”.

[// https://docs.microsoft.com/en-us/azure/cognitive-services/text-analytics/tutorials/tutorial-power-bi-key-phrases](https://docs.microsoft.com/en-us/azure/cognitive-services/text-analytics/tutorials/tutorial-power-bi-key-phrases)

[// Returns key phrases from the text in a comma-separated list](#)

(text) => `let`

`apikey = apikey,`

`endpoint = endpoint & "/keyPhrases",`

`jsonText = Text.FromBinary(Json.FromValue(Text.Start(Text.Trim(text), 5000))),`

`jsonbody = "{ documents: [{ language: ""en"", id: ""0"", text: " & jsonText & " }] }",`

```

bytesbody = Text.ToBinary(jsonbody),
headers   = [{"Ocp-Apim-Subscription-Key" = apikey},
bytesresp = Web.Contents(endpoint, [Headers=headers,
Content=bytesbody]),
jsonresp  = Json.Document(bytesresp),
keyphrases = Text.Lower(Text.Combine(jsonresp[documents][0]
[keyPhrases], ", "))
in keyphrases

```

Figure 11-10: Power Query function which accepts a text parameter and returns the key phrases

This Power Query function we can now apply on any column in any Power Query. Select a column of type text and then select “Add column” in the menu and “Invoke Custom Function” in the ribbon. Give the resulting column a useful name (e. g. “keyphrases”) and select “Keyphrase API” as “Function query”. As soon as you click OK every single row of your Power Query will be sent over to Cognitive Services and the parsed key phrases are returned back to Power Query.

×

Invoke Custom Function

Invoke a custom function defined in this file for each row.

New column name

Function query

text (optional)

Figure 11-11: Invoke custom function “Keyphrases API” on column EnglishProductNameAndDescription

Please be aware of two facts: One, costs will be charged to your Azure subscription every time you refresh Power Query. Please refer to “pricing tier” to find out about how much. In my subscription I will currently be charged EUR 0.84 per 1000 calls to the API, but there are trial subscriptions available which

give you a limited amount of credits free of charge.

Second, the content of the column you are invoking Cognitive Services on is sent over the internet to Azure. In case this would violate any privacy restrictions, please consider running Cognitive Services in a container on your premises instead.

In cases where you want to build upon a pre-trained model or come up with a model on your own, you must consider one of the following options: Azure Machine Learning or R.

Azure Machine Learning

Azure Machine Learning allows you to build your own experiments by using different building blocks and deploying the experiment as a web service which can be called by your application. We will build a Machine Learning experiment, deploy it and then call it in Power Query now.

In the first example we will make use of a building block “Extract Key Phrases from Text”. This is based on the same pre-trained model as we used above in Cognitive Services. The extracted key phrases will therefore be the same. Azure Machine Learning gives though the option to clean and transform the data before and after the building block, which I will not do in this first example to keep it simple. (In the next chapter we will build our own algorithm to extract key phrases.)

As you can see in the screenshot below, I used “Book reviews from Amazon” as a basis for my experiment. This dataset, and many more, are available by default so you can easily start experimenting. Then I used the step “Edit metadata” to rename the columns of the dataset to “ProductKey” and “EnglishProductNameAndDescription” to match the column names of my dataset I will later use in Power Query. “Partition and Sample” allows me to combine both, “Book reviews from Amazon” and “Web service input” into one set of rows. Then I “Extract Key Phrases from Text” and combine the discovered keyphrases with the original two columns “ProductKey” and “EnglishProdctNameAndDescription”. Finally, “Web services output” makes sure to send all three columns back to the caller of the web service.

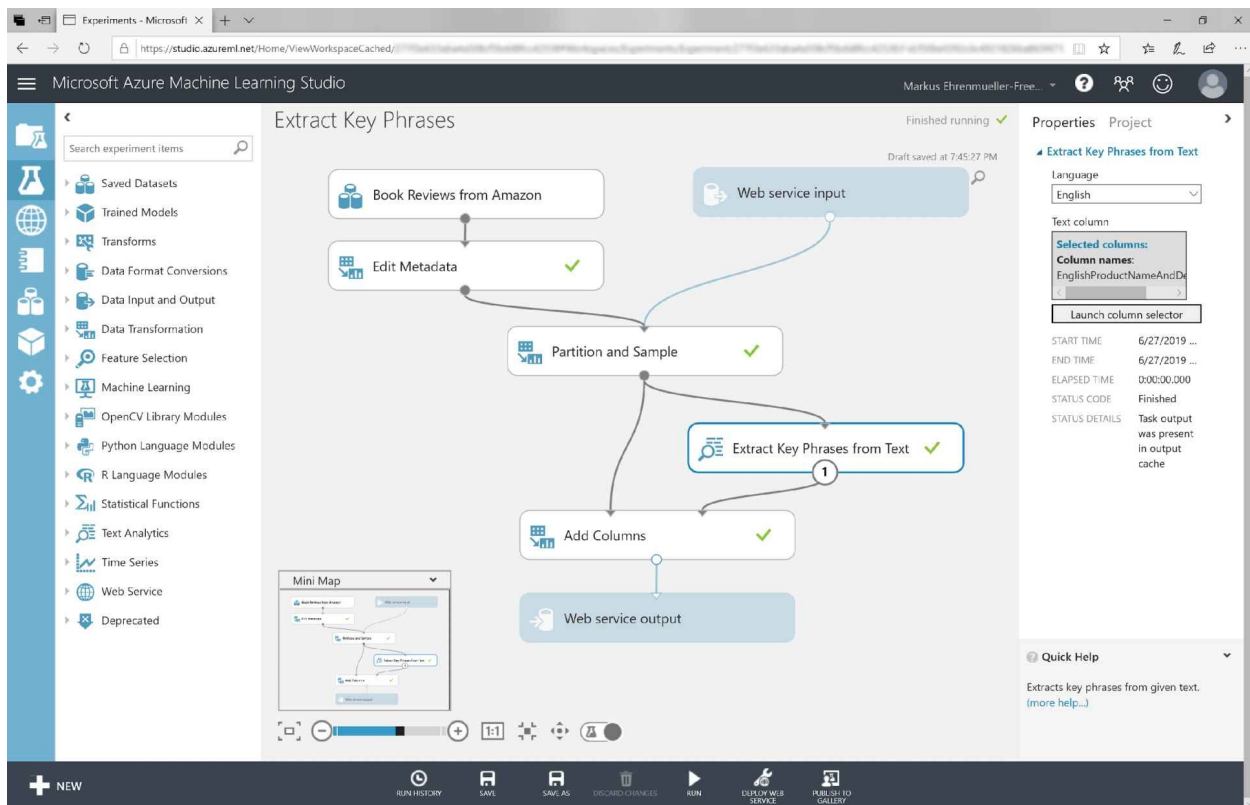


Figure 11-12: An Experiment in Azure Machine Learning Studio to extract key phrases

Before you can finally press the button “deploy web service”, you first have to press button “run” to run the experiment once.

When in “experiment view” (🔍) the blue lines and the web service component are ignored, so the experiment is not waiting for a web service call or returning data. When in “web service view” (🌐), on the other hand, the “Book reviews from Amazon” and the “Edit Metadata” is ignored, so the web service result is not influenced from the data of the experiment.

Before we can incorporate a call to our new web service in Power Query with the method I will describe in a minute, we must gather two important parameters of the web service: API key of the web service and the request URI of the API. I created Power Query parameters for both: *WebServiceAPIKey* and *RequestURI*.

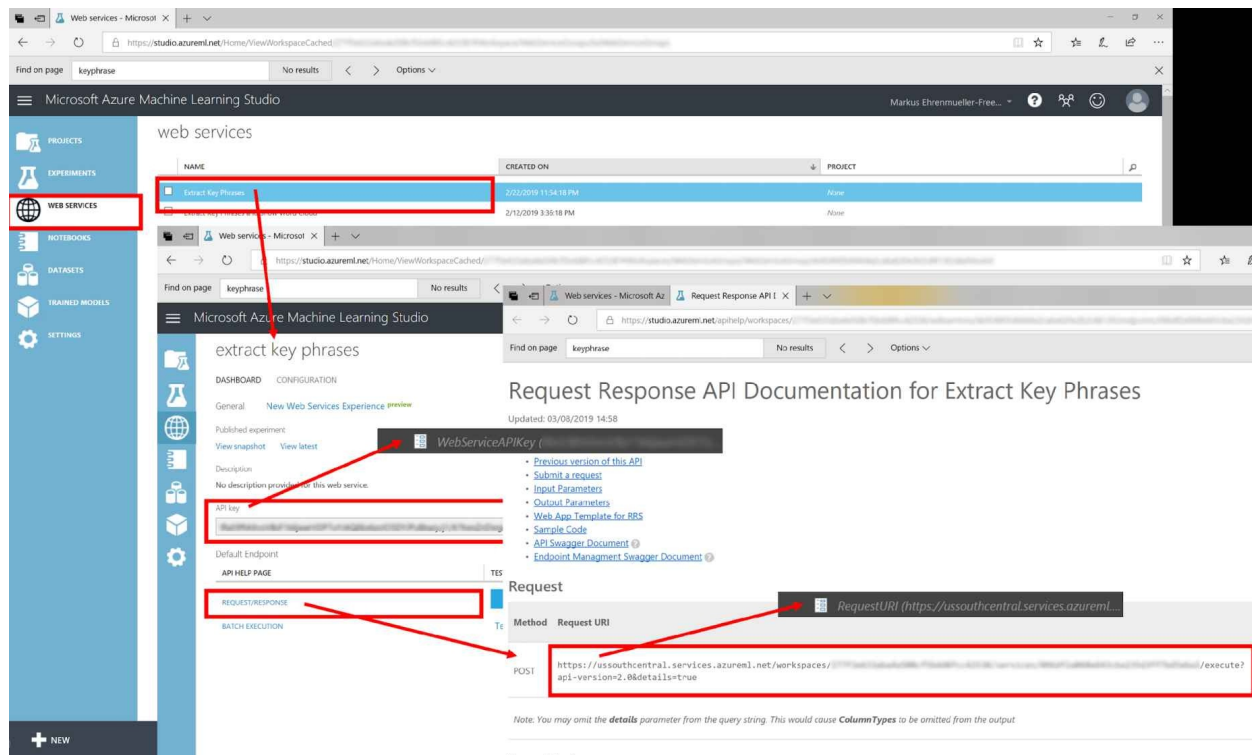


Figure 11-13: Where to find API key and request URI of Azure Machine Learning web service

While different possibilities of calling a Azure ML web service from Power Query are available, I prefer Gerhard Brückl's approach (<https://blog.gbrueckl.at/2016/06/score-powerbi-datasets-dynamically-azure-ml/>). Basically, he wrote three Power Query functions: One to convert Power Query data into JSON (*ToAzureMLJson*), one to convert a JSON into a Power Query table (*AzureMLJsonToTable*) and one to call Azure ML's API and making use of the first two functions (*CallAzureMLService*). You can read more details about that in his excellent blog post.

The rest of the steps are like those described in the previous chapter of applying a function on a column in Power Query. The function *CallAzureMLService* takes the web service's API key and the request URI as a parameter (just select the Power Query Parameter for that). Furthermore, we must pass in the name of a table, which must consist of exactly those two columns we defined for the input of the web service, with exactly the same name and data type. The function then returns the two columns plus an extra one, containing the key phrases.

Please be aware of two facts: Costs will be charged to your Azure subscription every time you refresh Power Query. Please refer to "pricing tier" to find out

about how much. I used a free subscription of Azure ML, which allows for only a limited amount of calls, which was sufficient for my sample.

Second the content of the whole table selected in the third parameter of the Power Query function *CallAzureMLService* is sent over the internet to Azure. Currently there is no on-premise alternative (e. g. containers) available for Azure ML.

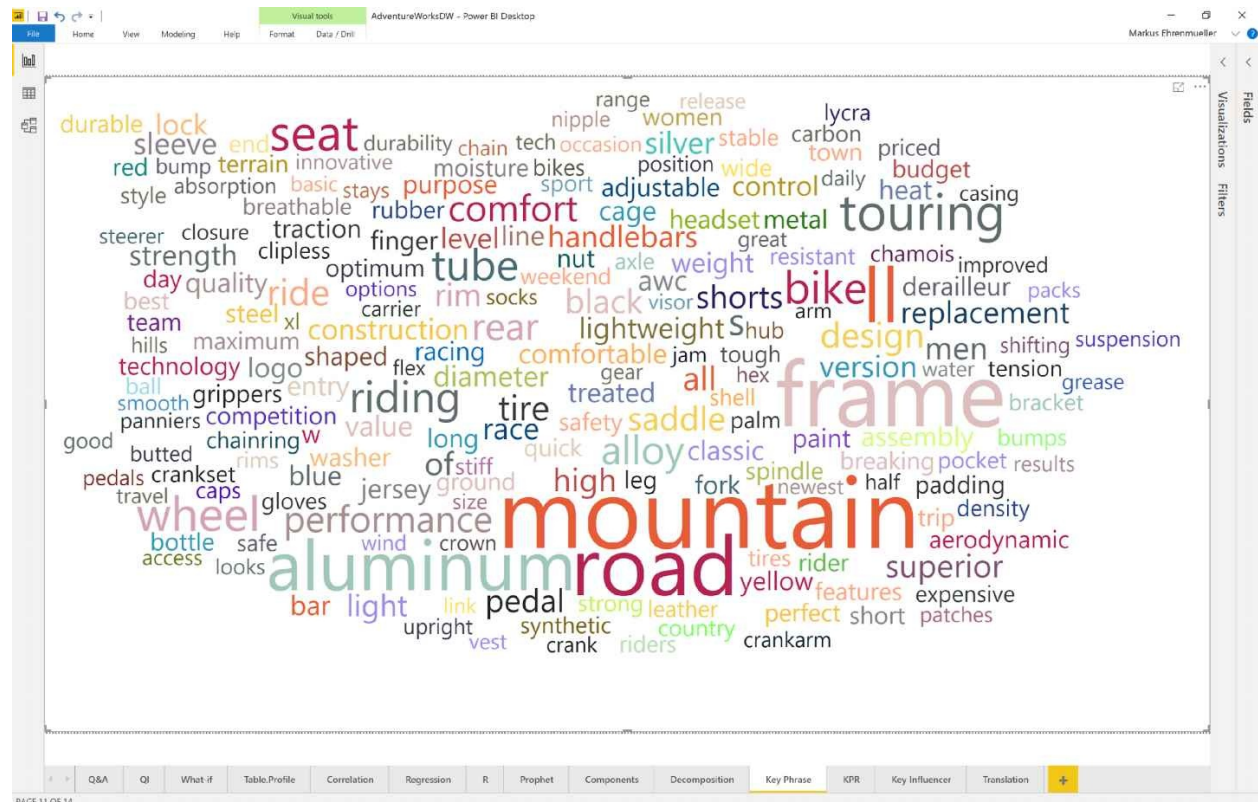


Figure 11-14: Word cloud based on Cognitive Services' key phrases

R in Azure Machine Learning

In cases the available machine learning algorithms in Azure Machine Learning are not sufficient for your use case you can always plug in R code into it. You can use R to create both, a model, and to run a R script in Azure Machine Learning. The following example will show you, how you can extract key phrases via R. I copied the experiment from above and exchanged the “Extract Key Phrases from Text” with “Execute R Script”. You can find the R script here:

```
# Map 1-based optional input ports to variables
dataset <- maml.mapInputPort(1) # class: data.frame
```

```
library(tm)
```

```

corpus_clean <-
VCorpus(VectorSource(dataset$EnglishProductNameAndDescription))
corpus_clean <- tm_map(corpus_clean, content_transformer(tolower)) # apply r-
standard function tolower() to content
corpus_clean <- tm_map(corpus_clean, stemDocument)
corpus_clean <- tm_map(corpus_clean, stripWhitespace) # eliminate unneeded
whitespace
corpus_clean <- tm_map(corpus_clean, removeNumbers) # apply tm's
removeNumbers to content
corpus_clean <- tm_map(corpus_clean, removeWords, stopwords()) # remove
stop words
corpus_clean <- tm_map(corpus_clean, removePunctuation) # remove
punctuation
corpus_clean <- tm_map(corpus_clean, stemDocument)
corpus_clean <- tm_map(corpus_clean, stripWhitespace) # eliminate unneeded
whitespace

# Select data.frame to be sent to the output Dataset port
df <- data.frame(text = sapply(corpus_clean, paste, collapse = " "),
stringsAsFactors = FALSE)
maml.mapOutputPort("df")

```

Figure 11-15: R script to fetch Azure ML's input, find key phrases and return them as output

After running the experiment and deploying it as a web service you have to then grab the API key of the web service and the request URI of the API and apply Power Query function *CallAzureMLService* exactly as described in the previous chapter.

Before you write your own R scripts to run in Azure Machine Learning, please check if the libraries you want to use are available in Azure. Find a list of available libraries here: <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/r-packages-supported-by-azure-machine-learning>.

Again: This is an easy way to share a functionality across different Power BI files and can be applied in Excel or your own application as well. Costs apply for every call of the web service and data is exposed over the internet. If you want to avoid those costs and the exposure over the internet you can incorporate the R

script directly as a Power Query transformation step, as described in the following chapter.

R as a Power Query Transformation

We've already seen in the chapter about linear regression that we can write R code to run in an R visual. Power BI offers two more possibilities, to directly inject R code: R source and R transformation in Power Query. Let's check out R transformation in the next example.

In the Power Query window find the table containing the column "EnglishProductNameAndDescription" and select "Transform" in the menu and select "Run R script" in the Ribbon. Then copy & paste the script from above, where I only removed the first and last line of code, which controlled the communication between Azure ML and R. In R transformation there is no need for calling special functions (like `maml.mapInputPort` or `maml.mapOutputPort`), as all the columns and rows available in the current Power Query are injected as a data frame with name "dataset" into the script:

```
library(tm)
corpus <-
VCorpus(VectorSource(dataset$EnglishProductNameAndDescription))
corpus_clean <- tm_map(corpus, content_transformer(tolower)) # apply r-
standard function tolower() to content
corpus_clean <- tm_map(corpus_clean, stemDocument)
corpus_clean <- tm_map(corpus_clean, stripWhitespace) # eliminate unneeded
whitespace
corpus_clean <- tm_map(corpus_clean, removeNumbers) # apply tm's
removeNumbers to content
corpus_clean <- tm_map(corpus_clean, removeWords, stopwords()) # remove
stop words
corpus_clean <- tm_map(corpus_clean, removePunctuation) # remove
punctuation
corpus_clean <- tm_map(corpus_clean, stemDocument)
corpus_clean <- tm_map(corpus_clean, stripWhitespace) # eliminate unneeded
whitespace
```

Figure 11-16: R script to find key phrases

This solution runs completely locally – there are no additional costs during refresh and no data leaves your laptop. But this comes on the price of less

reusability, as you have to duplicate the code in every Power BI file. Applying changes to the code can be challenging, as you must maintain different copies of the code.

The R code is executed only once during refresh and cannot be influenced by filters. If the R script needs to be parametrized by user selection, we can use an R visual instead.

R Visual

You've already seen an R visual above, when we used `ggplot` to calculate and draw the regression line. We can do the same for our key phrases and the word cloud.

The code below to extract the key phrases is the same as above, except that it has lines of code added to create a word cloud visual:

```
library(tm)
corpus <-
VCorpus(VectorSource(dataset$EnglishProductNameAndDescription))
corpus_clean <- tm_map(corpus, content_transformer(tolower)) # apply r-
standard function tolower() to content
corpus_clean <- tm_map(corpus_clean, stemDocument)
corpus_clean <- tm_map(corpus_clean, stripWhitespace) # eliminate unneeded
whitespace
corpus_clean <- tm_map(corpus_clean, removeNumbers) # apply tm's
removeNumbers to content
corpus_clean <- tm_map(corpus_clean, removeWords, stopwords()) # remove
stop words
corpus_clean <- tm_map(corpus_clean, removePunctuation) # remove
punctuation
corpus_clean <- tm_map(corpus_clean, stemDocument)
corpus_clean <- tm_map(corpus_clean, stripWhitespace) # eliminate unneeded
whitespace

library(wordcloud)
library(RColorBrewer)
wordcloud(corpus_clean, min.freq = 1, random.order = FALSE,
colors=brewer.pal(9, "BuGn"))
```

Figure 11-17: R script to find key phrases and plot them in a word cloud

Summary

In this chapter we have seen different possibilities to extract key phrases out of a free text column (the concatenated product name and description). Every solution had its own strengths and weaknesses, as shown in the following table:

	Cognitive Services	Azure ML Key Phrase	Azure ML R	Power Query R Transformation	Power BI R visual
Additional Costs	Apply	Apply	Apply	No	No
Data is sent over the internet	Yes *)	Yes	Yes	No	No
Reuseability	Yes	Yes	Yes	No	No
Calculation	Data refresh	Data refresh	Data refresh	Data refresh	Visual refresh
User filter	No	No	No	No	Apply
Pre-trained vs. self-trained Model	Pre-trained	Pre-trained	Self-trained	Self-trained	Self-trained

*) You can run Cognitive Services in a container on premises to avoid exposure of data.

About the Author



Markus Ehrenmüller-Jensen is the founder of Savory Data and works as a project leader, trainer & consultant for data engineering, business intelligence, and data science since 1994. He is an educated software-engineer, graduated business educator and professor for databases and project engineering at HTL Leonding (technical college) and certified as MCSE Data Platform, MCSE Business Intelligence, and MCT. Markus speaks regularly on international conferences (eg. PASS Summit, SQLBits, SQL Saturdays, SQL Days, Power BI World Tour, ...) and writes articles for well-known journals. In 2013 he co-founded SQL PASS Austria and in 2015 Austria PUG and organizes SQL Saturdays in Austria since 2014. For his technical leadership in the community he was awarded as a Microsoft Data Platform MVP since 2017.

Chapter 12: Automated Machine Learning in Power BI

Author: Ashraf Ghonaim

Microsoft is committed to democratizing AI through its products by making automated machine learning accessible through Power BI, so that Data Analysts and BI professionals can also take advantage of machine learning and become Citizen Data Scientists.

With Automated Machine Learning (AutoML), the data science behind the creation of ML models is automated by Power BI, with precautions to ensure model quality, and visibility to have full insight into the steps used to create the ML model.

What is Machine Learning (ML)?

Among so many definitions, Machine Learning can be defined as: "The field of study that gives computers the ability to learn without being explicitly programmed" - Arthur Samuel (1959)

ML algorithms learn from the historical data to build models that predict the future. ML at its heart is a probabilistic not a deterministic science because of generalization based on probabilities. That means any ML model has a margin of error and accuracy level.

The role of a data scientist is to come up with the model that has the highest level of possible accuracy of predicting the outcome of the new unseen data.

To be able to achieve that, the data scientist has to split the data to training dataset and testing dataset. Then, try several ML algorithms and tweak so many parameters based on their human expertise to train multiple models. The data scientist compares the results of different trained models using the testing dataset that includes unseen labeled data and finally picks the model that achieved the highest level of accuracy.

What are the challenges of Traditional ML?

Traditional ML model development process is resource-intensive, requiring significant domain knowledge and time to produce and compare dozens of models.

Data scientists and developers face a series of sequential and interconnected decisions along the way to achieving "magic" machine learning solutions. For example, should they transform the input data, and if so, how – by removing nulls, rescaling, or something else entirely? What machine learning algorithm would be best - a support vector machine (SVM), logistic regression, or a tree-based classifier? What parameter values should they use for the chosen classifier – including decisions such as what the max depth and min split count should be for a tree-based classifier? And many more.

Ultimately, all these decisions will determine the accuracy of the machine learning pipeline - the combination of data pre-processing steps, learning algorithms, and hyperparameter settings that go into each machine learning solution.

What is Automated Machine Learning (AutoML)?

Automated machine learning, also referred to as AutoML, is the process of automating the time consuming, iterative tasks of ML model development process. It allows data scientists, analysts, and developers to build ML models with high scale, efficiency, and productivity all while sustaining model quality.

AutoML democratizes the ML model development process, and empowers its users, no matter their data science expertise, to identify an end-to-end machine learning pipeline for any problem.

Data scientists, analysts and developers across industries can use AutoML to:

- Implement machine learning solutions without extensive programming knowledge
- Save time and resources
- Leverage data science best practices
- Provide agile problem-solving

AutoML empowers Power BI users, with or without data science expertise, to identify an end-to-end machine learning pipeline for any problem, achieving higher accuracy while spending far less of their time. It enables a significantly larger number of experiments to be run, resulting in faster iteration towards production-ready intelligent experiences.

AutoML is based on a [breakthrough from Microsoft Research](#). The approach combines ideas from collaborative filtering and Bayesian optimization to search an enormous space of possible machine learning pipelines intelligently and efficiently. It's essentially a recommender system for machine learning pipelines. Similar to how streaming services recommend movies for users, AutoML recommends machine learning pipelines for data sets.

Automated Machine Learning (AutoML) in Power BI

There are several options to do Machine Learning in Power BI:

- 1- Python/R scripts
- 2- Azure Machine Learning Studio (Web Service API)
- 3- Azure Machine Learning Services Integration
- 4- Built-in AutoML (which is the scope of this chapter)

AutoML is done in Power BI through dataflows in Power BI Service in the cloud. Dataflows offers self-serve data prep for big data. AutoML enables you to leverage your data prep effort for building ML models, right within Power BI. Dataflows is a simple and powerful ETL tool that enables analysts to prepare data for further analytics. You invest significant effort in data cleansing and preparation, creating datasets that can be used across your organization. AutoML enables you to leverage your data prep effort for building ML models directly in Power BI.

AutoML in Power BI dataflow allows to build ML models with clicks, not code, using just their Power BI skills. With AutoML, the data science behind the creation of ML models is automated by Power BI, with precautions to ensure model quality, and visibility to have full insight into the steps used to create your ML model.

AutoML for dataflows enables data analysts to train, validate and invoke ML models directly in Power BI. It includes a simple experience for creating a new ML model where analysts can use their dataflows to specify the input data for training the model. The service automatically extracts the most relevant features, selects an appropriate algorithm and tunes and validates the ML model. After a model is trained, Power BI automatically generates a report that includes the results of validation that explains the performance and results to analysts. The model can then be invoked on any new or updated data within the dataflow.

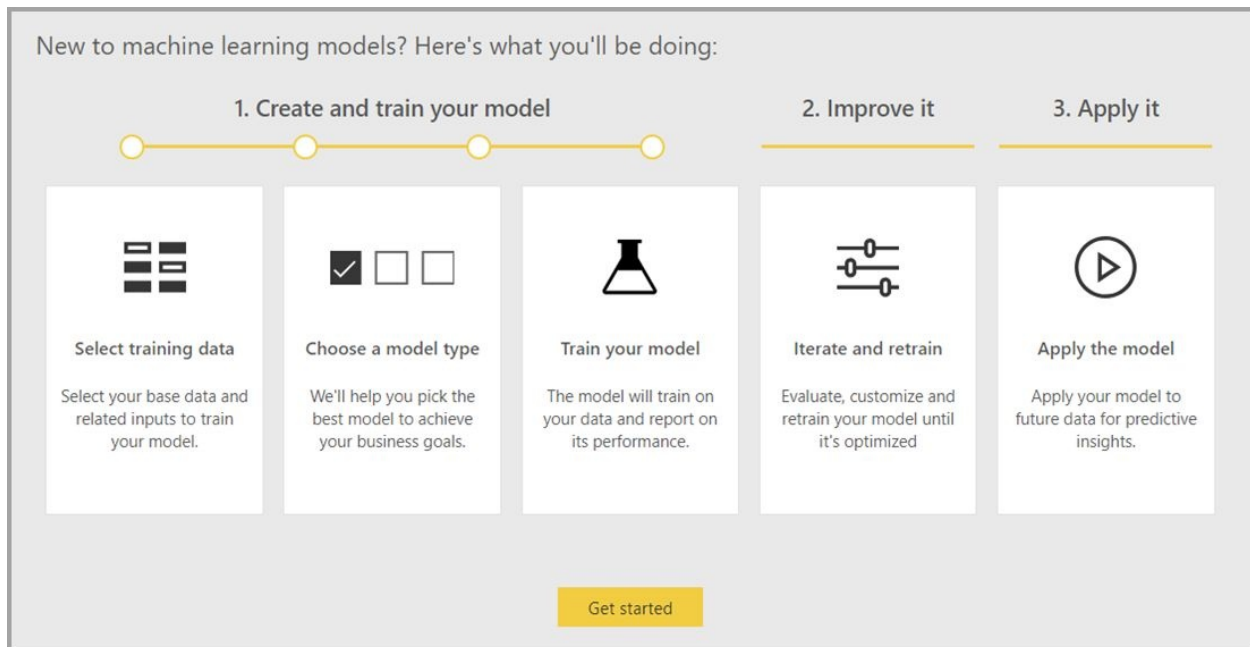


Figure 12-01: AutoML Getting Started

AutoML in Power BI integrates AutoML from the Azure Machine Learning service to create your ML models. However, you don't need an Azure subscription to use AutoML in Power BI. The process of training and hosting the ML models is managed entirely by the Power BI service.

AutoML is available for dataflows in workspaces hosted on Power BI Premium and Embedded capacities only. To be able to use it, you have to turn it on first in the premium capacity.

Enabling AutoML in your Power BI Premium Subscription

One of the goals of premium capacity is to give customers a fine-grained control of exactly what is running in each capacity. Each capacity has dedicated resources assigned, is isolated from others, and can only run generally available workloads by default. So, if there is a mission critical app running in a capacity, administrators are able to constrain other functionality to get the performance they need. For workloads in preview to run in a capacity, the capacity administrator needs to enable them and configure the maximum memory percentage available to it.

Power BI AI is a *capacity workload* and must be enabled before the features can be used. Note that this capacity workload includes Cognitive Services and AutoML. All other AI features such as Key Influencers and Quick Insights do not require a premium subscription.

You must be the Power BI Administrator or Capacity Administrator to be able to enable the Power BI AI capacity workload. Select the gear in the top right panel and click on “Admin Portal”:

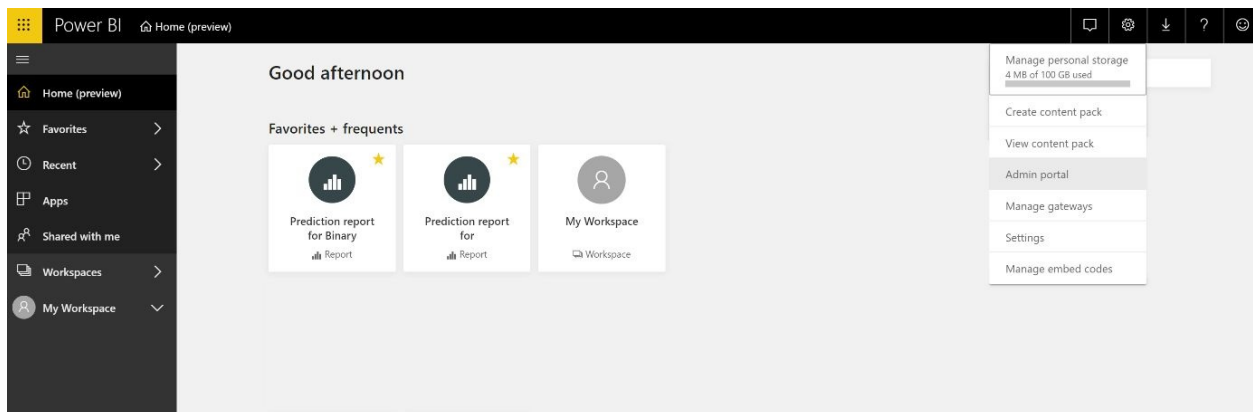


Figure 12-02: Admin Portal

Then click on the Capacity Settings on the left-hand panel on the subsequent page:

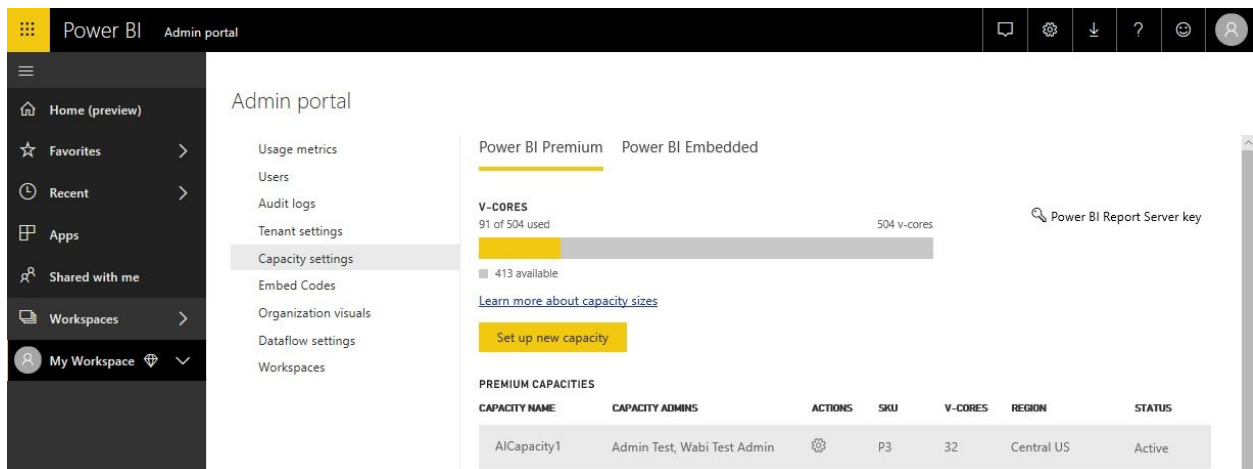


Figure 12-03: Capacity Settings

Click on the capacity where the workload is to be enabled. In the page above, it is “AICapacity1”:

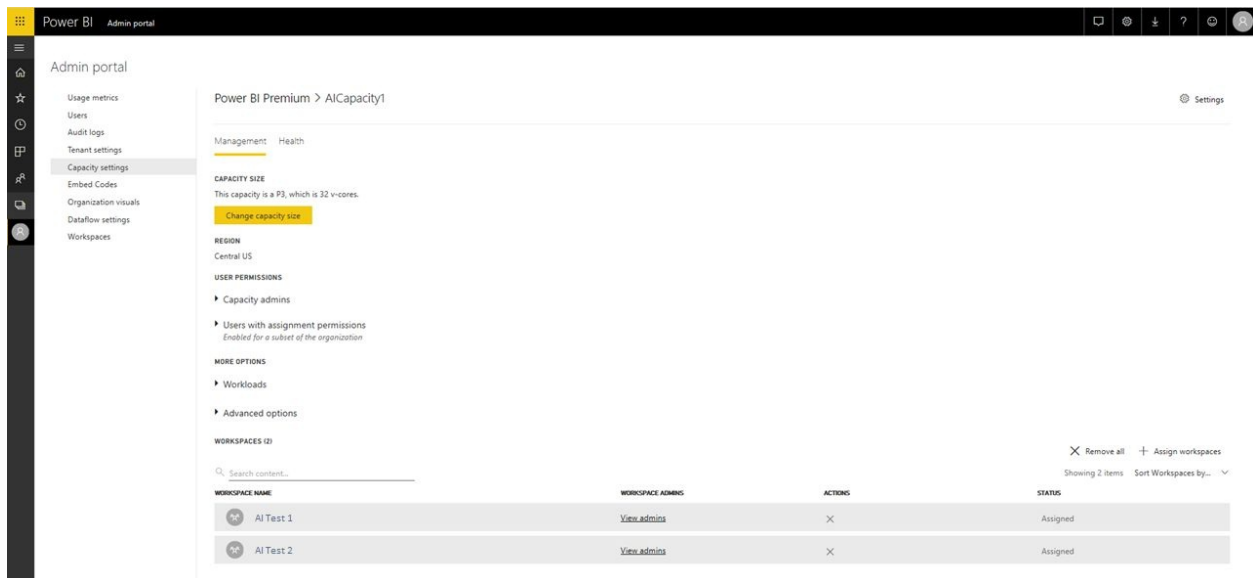


Figure 12-04: Capacity Workloads

Note that you can assign workspaces to this capacity after it is configured the way you want.

Then click “Workloads” under “More Options” and turn both Dataflows and AI (Preview) to On and set the maximum memory:

Workloads

AI (PREVIEW) - Active

Your workload is ready to use.

☒ On

Max Memory (%)

40

Allow usage from Power BI Desktop

☒ On

DATASETS (PREVIEW) - Active

Your workload is ready to use.

☐ On

XMLA Endpoint

1

DATAFLOWS - Active

Your workload is ready to use.

☒ On

Max Memory (%)

30

Container Size (Mb)

700

PAGINATED REPORTS (PREVIEW) - Active

Your workload is ready to use.

☒ On

Max Memory (%)

20

Apply

Cancel

Figure 12-05: Dataflows and AI (Preview)

AutoML performs numerous AI calculations in memory. While the memory needed depends on data size and content, it is a good idea to give a liberal

amount of memory for the AI workload. Our own testing was on 5GB max memory or higher (or 100% on an A2 node).

The impact of enabling each workload is that capacity is now shared – memory used for one workload may not be available for others.

After this is applied, Cognitive Services and AutoML will be available in all of the workspaces in the selected premium capacity.

Creating an AutoML Model in Power BI

In the current preview, AutoML enables you to train ML models for Binary Prediction, Classification, Regression and Forecasting (coming soon).

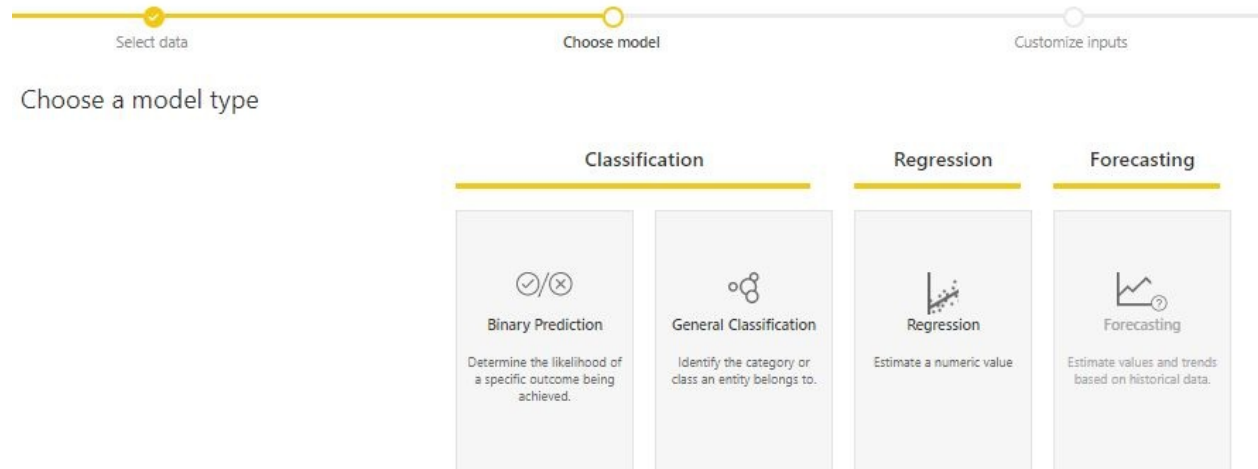


Figure 12-06: Model Types

Binary Prediction, Classification, and Regression models for dataflows are types of supervised ML models, which means that they learn from the known outcomes of past observations to predict the outcomes of other new unseen observations. The input dataset for training an AutoML model is a set of records that are labeled with the known outcomes.

After an ML model is trained, AutoML automatically generates a Power BI report that explains the likely performance of your ML model. AutoML emphasizes explainability, by highlighting the key influencers among your inputs that influence the predictions returned by your model. The report also includes key metrics for the model, depending on the ML model type.

Other pages of the generated report show the statistical summary of the model and the training details. The statistical summary is of interest to users who would like to see the standard data science measures of performance for the model. The training details summarize all the iterations that were run to create your model, with the associated modeling parameters. It also describes how each input was used to create the ML model.

You can then apply your ML model to your data for scoring. When the dataflow is refreshed, the predictions from your ML model are automatically applied to your data. Power BI also includes an individualized explanation for each specific prediction score that the ML model produces.

Creating an AutoML Model Step by Step

1- Data prep for creating ML Model:

Create a dataflow for the data with the historical outcome information, which is used for training the ML model.

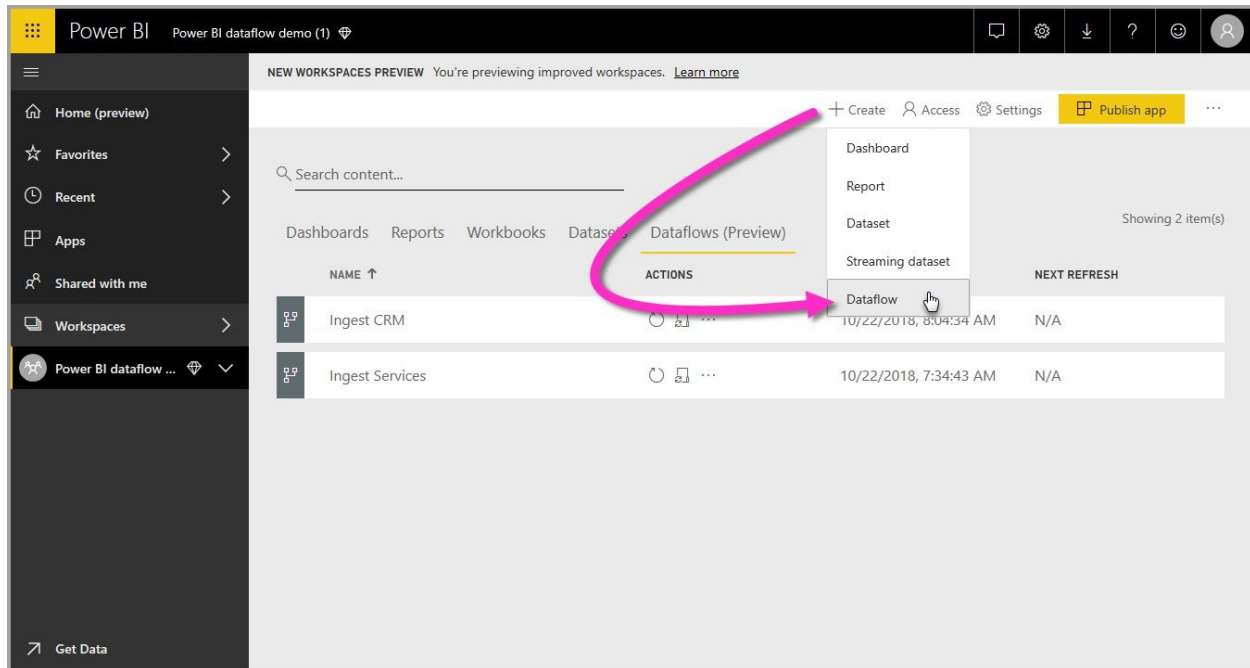


Figure 12-07: Create Dataflow

In the current release, Power BI uses data from only a single entity to train the ML model. So, if your historical data consists of multiple entities, you must manually join the data into a single dataflow entity. You should also add calculated columns for any business metrics that may be strong predictors for the outcome you're trying to predict.

2- Configuring the ML Model Inputs

AutoML has specific data requirements for training a ML model based on respective model types.

To create an AutoML model, select the ML icon in the "Actions" column of the dataflow entity with the historical data, and select "Add a ML model".



Figure 12-08: Add ML model

A simplified experience is launched, consisting of a wizard that guides you through the process of creating the ML model. The wizard includes the following simple steps.

- 1- Select the entity with the historical outcome data, and the field for which you want a prediction
- 2- Choose a model type based on the type of prediction you'd like to see
- 3- Select the inputs you want the model to use as predictive signals
- 4- Name your model and save your configuration

The historical outcome field identifies the label attribute for training the ML model, shown in the following image.

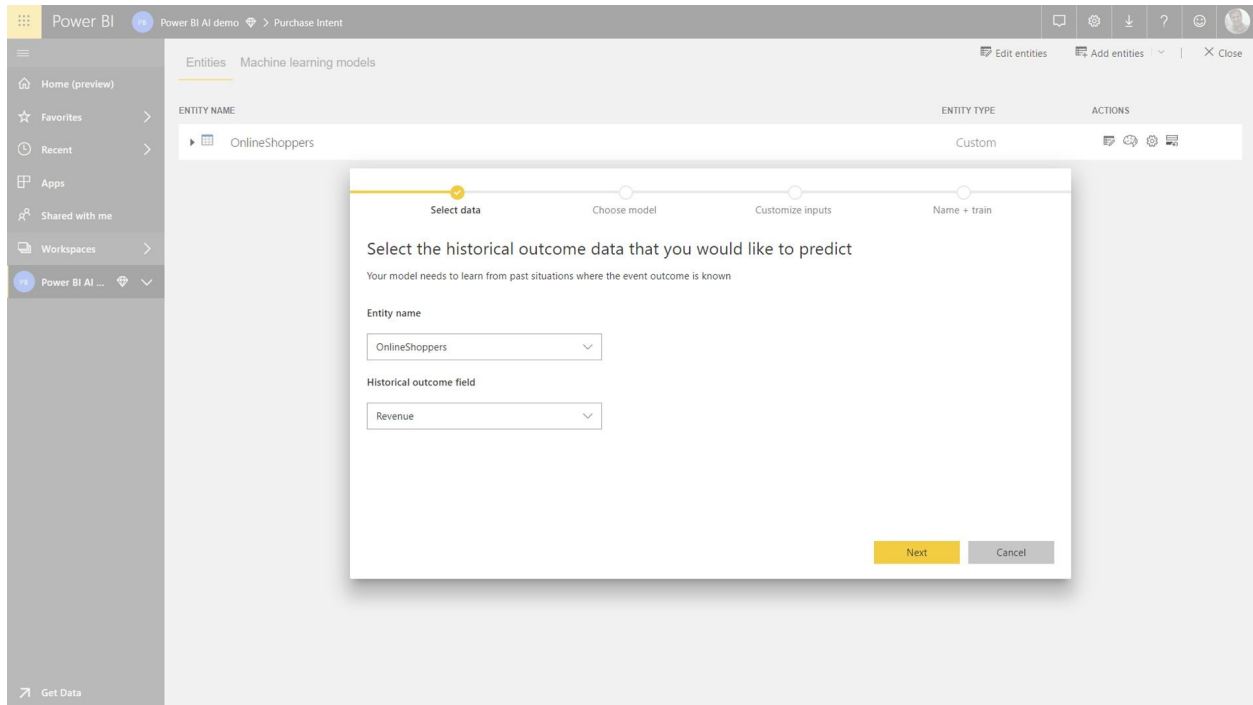


Figure 12-09: Select Outcome

When you specify the historical outcome field, AutoML analyzes the label data to identify the types of ML models that can be trained for that data and suggests the most likely ML model type that can be trained. Some model types may not be supported for the data that you have selected.

AutoML also analyzes all the fields in the selected entity to suggest the inputs that can be used for training the ML model. This process is approximate and is based on statistical analysis, so you should review the inputs used. Any inputs that are directly dependent on the historical outcome field (or the label field) should not be used for training the ML model, since they will affect its performance.

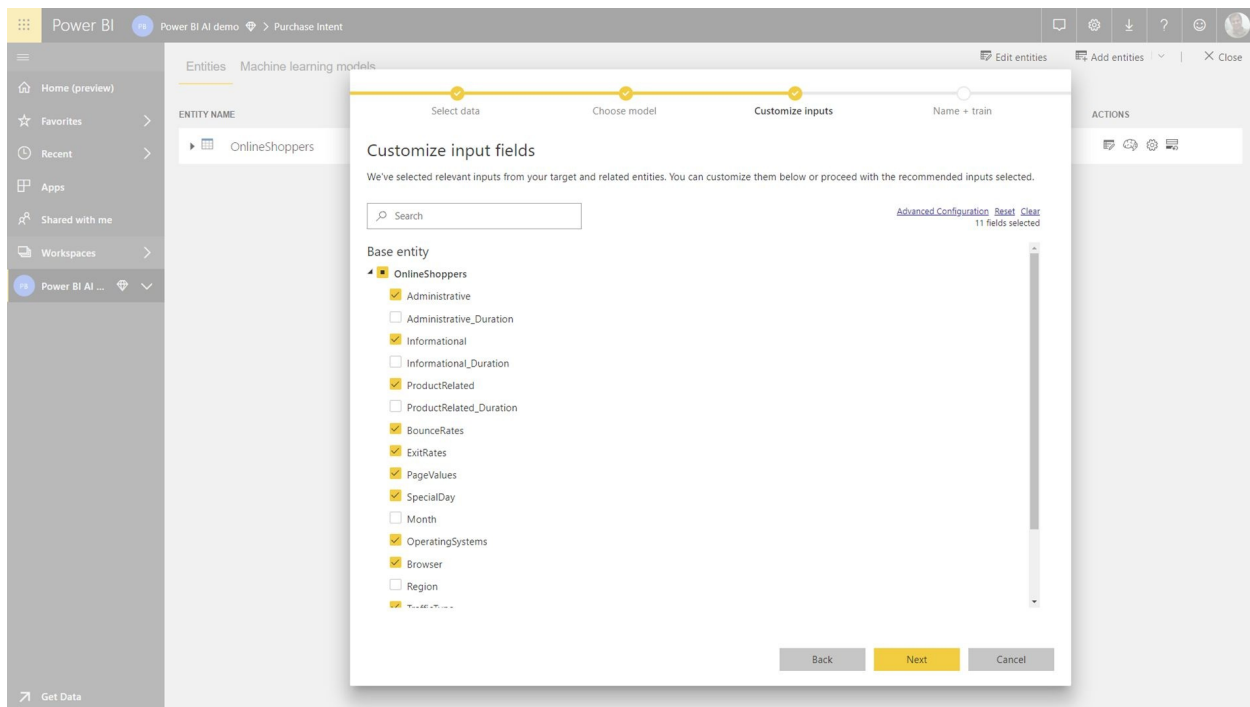


Figure 12-10: Select Inputs

In the final step, you can name the model and save its settings. At this stage, you are prompted to refresh the dataflow, which begins the training process for the ML model.

3- ML Model Training

Training of AutoML models is a part of the dataflow refresh. AutoML first prepares your data for training. AutoML splits the historical data you provide into a training and testing datasets. The test dataset is a holdout set that is used for validating the model performance after training. These are realized as Training and Testing entities in the dataflow. AutoML uses cross-validation for the model validation.

Next, each input field is analyzed and imputation is applied, which replaces any missing values with substituted values. A couple of different imputation strategies are used by AutoML. Then, any required sampling and normalization are applied to your data. AutoML applies several transformations at each selected input field based on its data type, and its statistical properties. AutoML uses these transformations to extract features for use in training your ML model.

The training process for AutoML models consists of up to 50 iterations with different modeling algorithms and hyperparameter settings to find the model with the best performance. The performance of each of these models is assessed

by validation with the holdout test dataset. During this training step, AutoML creates several pipelines for training and validation of these iterations. The process of assessing the performance of the models can take time, anywhere from several minutes to a couple of hours, depending on the size of your dataset and the dedicated capacity resources available.

In some cases, the final model generated may use ensemble learning, where multiple models are used to deliver better predictive performance.

4- AutoML Model Explainability

After the model has been trained, AutoML analyzes the relationship between the input features and the model output. It assesses the magnitude and direction of change to the model output for the holdout test dataset for each input feature. This is known as the *feature importance*.

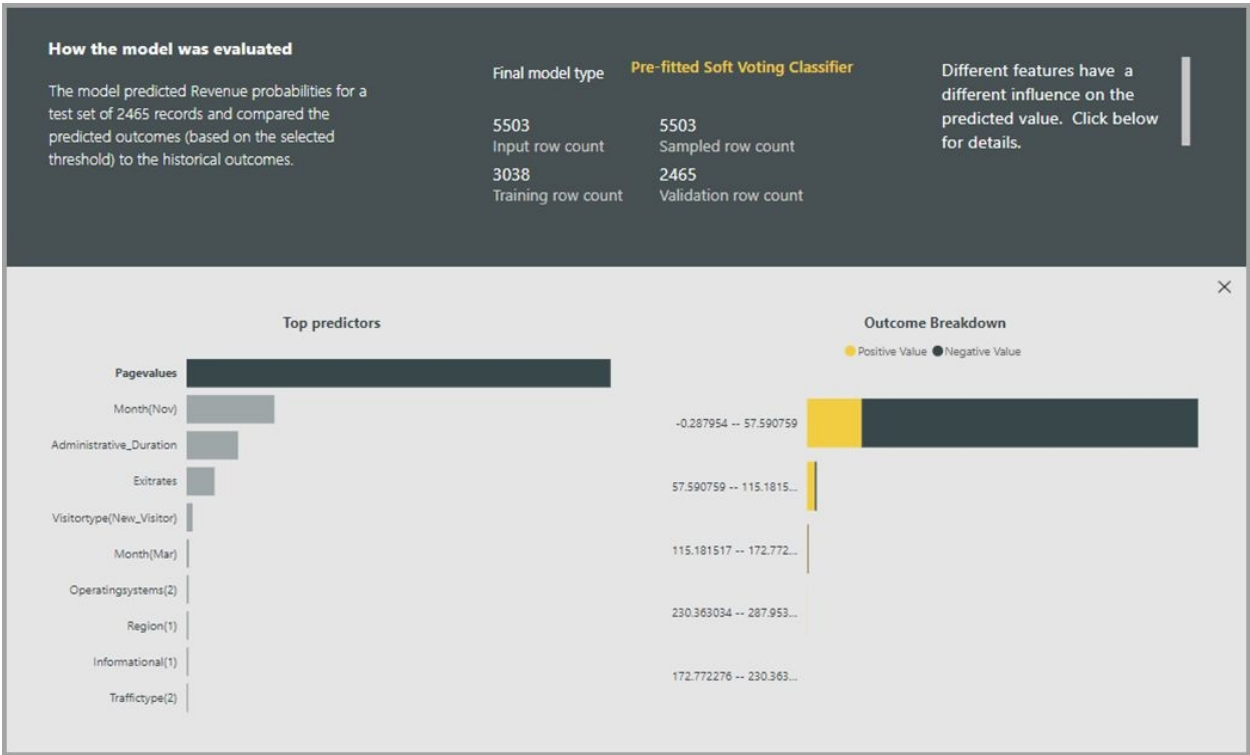


Figure 12-11: Model Report

5- AutoML Model Report

AutoML generates a Power BI report that summarizes the performance of the model during validation, along with the global feature importance. The report summarizes the results from applying the ML model to the holdout test data and comparing the predictions with the known outcome values.

You can review the model report to understand its performance. You can also

validate that the key influencers of the model align with the business insights about the known outcomes.

The charts and measures used to describe the model performance in the report depend on the model type. These performance charts and measures are described in the following sections.

Additional pages in the report may describe statistical measures about the model from a data science perspective. For instance, the **Binary Prediction** report includes a gain chart and the ROC curve for the model.

The reports also include a **Training Details** page that includes a description of how the model was trained, and includes a chart describing the model performance over each of the iterations runs.

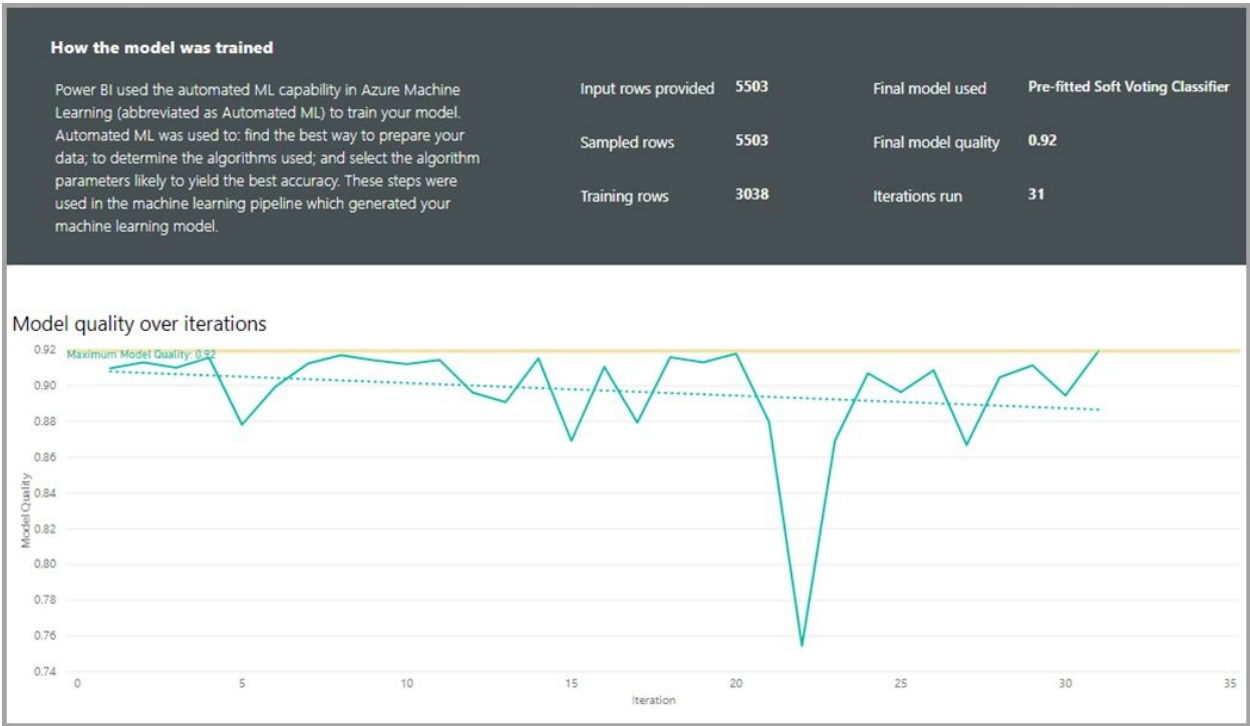


Figure 12-12: Model Performance

Another section on this page describes how the imputation method used for filling missing values for the input fields, as well as how each input field was transformed to extract the features used in the model. It also includes the parameters used by the final model.

Your machine learning model

The tables below contain the list of features extracted from the inputs you provided, and the final set of parameters that were used to create your machine learning model. This information can be used to recreate the machine learning model outside Power BI.

Data Featurization

Feature	Detected Column Type	Imputation	Details
Administrative	Categorical		Details
Browser	Categorical		Details
Informational	Categorical		Details
Month	Categorical		Details
OperatingSystems	Categorical		Details
Region	Categorical		Details
TrafficType	Categorical		Details
VisitorType	Categorical		Details
Administrative_Duration	Numeric	Mean	Details
BounceRates	Numeric	Mean	Details
ExitRates	Numeric	Mean	Details
Informational_Duration	Numeric	Mean	Details
PageValues	Numeric	Mean	Details
ProductRelated	Numeric	Mean	Details
ProductRelated_Duration	Numeric	Mean	Details
SpecialDay	Numeric	Mean	Details
Weekend	Categorical	Mode	Details

Pre-fitted Soft Voting Classifier final parameters

Parameter Name	Parameter Value
model_seed_threshold	0.05
min_models	1
max_models	15

Figure 12-13: Data Featurization and Parameters

If the model produced uses ensemble learning, then the Training Details page also includes a section describing the weight of each constituent model in the ensemble, as well as its parameters.

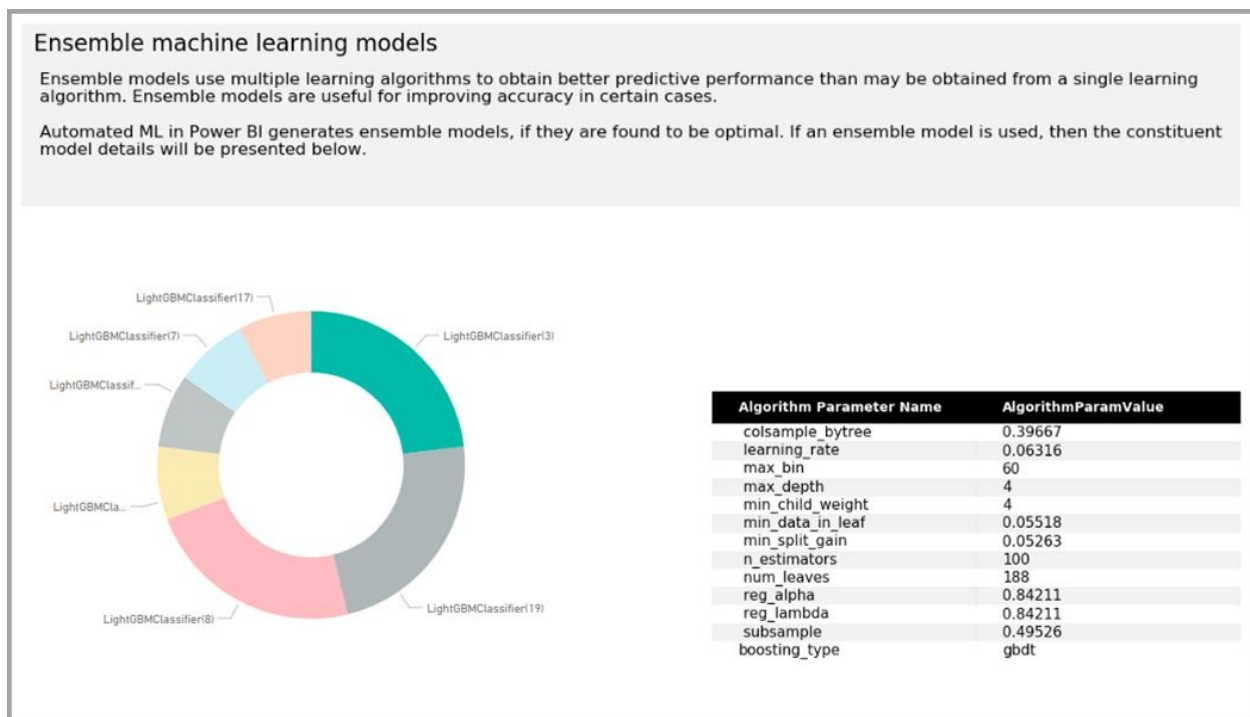


Figure 12-14: Ensemble ML

6- Applying the AutoML Model

If you're satisfied with the performance of the ML model created, you can apply it to new or updated data when your dataflow is refreshed. You can do this from the model report, by selecting the Apply button in the top-right corner.

To apply the ML model, you must specify the name of the entity to which it must be applied, and a prefix for the columns that will be added to this entity for the model output. The default prefix for the column names is the model name. The *Apply* function may include additional parameters specific to the model type.

Applying the ML model creates a new dataflow entity with the suffix *enriched <model_name>*. For instance, if you apply the *PurchaseIntent* model to the *OnlineShoppers* entity, the output will generate the *OnlineShoppers enriched PurchaseIntent*.

Currently, the output entity cannot be used to preview the ML model results in the Power Query editor. The output columns always show null as the result. To view the results, a second output entity with the suffix *enriched <model_name> Preview* is created when the model is applied.

You must refresh the dataflow, to preview the results in the Query Editor.

Power BI | Power BI AI demo > Purchase Intent

Edit queries

Get data | Refresh | Options | Manage columns | Transform columns | Transform table | Reduce rows | Add column | AI insights | Map to standard | Combine tables

Dataflow[Entity = "OnlineShoppers enriched Purchase Intent Prediction"][[Data]]

	y3 TrafficType	y3 VisitorType	y3 Weekend	y3 Revenue	y3 Purchase Intent Pr...	y3 Purchase Intent Pr...	y3 Purchase Intent Pr...	y3 Purchase Intent Pr...
1	1	2 Returning_Visitor	FALSE	(null)	FALSE	65	["VisitorType", "Categori...	
2	1	2 Returning_Visitor	TRUE	(null)	TRUE	77	["Month", "Categorical", ...	
3	1	2 Returning_Visitor	FALSE	(null)	TRUE	89	["PageValues", "Numeric", ...	
4	4	1 Returning_Visitor	TRUE	(null)	FALSE	6	["PageValues", "Numeric", ...	
5	1	11 Returning_Visitor	TRUE	(null)	FALSE	20	["PageValues", "Numeric", ...	
6	3	1 Returning_Visitor	FALSE	(null)	FALSE	9	["Administrative_Duration...	
7	3	2 New_Visitor	FALSE	(null)	FALSE	24	["Administrative_Duration...	
8	3	10 Returning_Visitor	TRUE	(null)	FALSE	10	["PageValues", "Numeric", ...	
9	1	2 Returning_Visitor	FALSE	(null)	TRUE	80	["Administrative", "Catego...	
10	3	1 Returning_Visitor	FALSE	(null)	FALSE	14	["PageValues", "Numeric", ...	
11	7	2 Returning_Visitor	FALSE	(null)	TRUE	84	["PageValues", "Numeric", ...	
12	2	1 Returning_Visitor	FALSE	(null)	TRUE	92	["Month", "Categorical", ...	
13	1	1 Returning_Visitor	FALSE	(null)	FALSE	9	["VisitorType", "Categori...	
14	1	7 New_Visitor	FALSE	(null)	FALSE	26	["Month", "Categorical", ...	
15	1	2 Returning_Visitor	FALSE	(null)	FALSE	8	["Administrative", "Catego...	
16	1	9 Returning_Visitor	TRUE	(null)	TRUE	85	["PageValues", "Numeric", ...	
17	6	6 Returning_Visitor	FALSE	(null)	FALSE	18	["Month", "Categorical", ...	
18	1	10 Returning_Visitor	TRUE	(null)	FALSE	33	["PageValues", "Numeric", ...	
19	8	2 Returning_Visitor	TRUE	(null)	FALSE	24	["ProductRelated_Duration...	
20	3	10 Returning_Visitor	FALSE	(null)	FALSE	15	["PageValues", "Numeric", ...	
21	1	3 Returning_Visitor	TRUE	(null)	TRUE	88	["VisitorType", "Categori...	
22	2	1 Returning_Visitor	TRUE	(null)	FALSE	9	["VisitorType", "Categori...	
23	1	2 Returning_Visitor	FALSE	(null)	FALSE	16	["VisitorType", "Categori...	
24	6	2 Returning_Visitor	TRUE	(null)	FALSE	25	["ProductRelated_Duration...	
25	1	2 Returning_Visitor	FALSE	(null)	TRUE	86	["VisitorType", "Categori...	
26	1	3 Returning_Visitor	TRUE	(null)	TRUE	84	["ProductRelated_Duration...	
27	1	2 Returning_Visitor	TRUE	(null)	TRUE	87	["VisitorType", "Categori...	
28	2	10 Returning_Visitor	TRUE	(null)	FALSE	15	["PageValues", "Numeric", ...	
29	1	2 Returning_Visitor	FALSE	(null)	TRUE	79	["PageValues", "Numeric", ...	
30	1	1 Returning_Visitor	FALSE	FALSE	FALSE	6	["Month", "Categorical", ...	
31	1	2 Returning_Visitor	FALSE	FALSE	FALSE	9	["VisitorType", "Categori...	

Entity type: OnlineShoppers enriche...
Custom
Applied steps: Source, Workspace, Dataflow, Preview

Cancel Save & close

Figure 12-15: Output Entity

When you apply the model, AutoML always keeps your predictions up-to-date when the dataflow is refreshed.

AutoML also includes an individualized explanation for each row that it scores in the output entity.

To use the insights and predictions from the ML model in a Power BI report, you can connect to the output entity from Power BI Desktop using the dataflows connector.

Deep dive into the 3 types of ML Models

1- Binary Prediction Models

Binary Prediction models, more formally known as binary classification models, are used to classify a dataset into two groups. They're used to predict events that can have a binary outcome, such as whether a sales opportunity will convert, whether an account will churn, whether an invoice will be paid on time; whether a transaction is fraudulent, and so on.

Since the outcome is binary, Power BI expects the label for a binary prediction model to be a Boolean, with known outcomes being labeled true or false. For instance, in a sales opportunity conversion model, sales opportunities that have been won are labeled true, those that have been lost are labeled false, and the open sales opportunities are labeled null.

The output of a Binary Prediction model is a probability score, which identifies the likelihood that the outcome corresponding to the label value being true will be achieved.

Training a Binary Prediction Model

To create a Binary Prediction model, the input entity containing your training data must have a Boolean field as the historical outcome field to identify the past known outcomes.

Pre-requisites:

- A Boolean field must be used as the historical outcome field
- A minimum of 50 rows of historical data is required for each class of outcomes

In general, if the past outcomes are identified by fields of a different data type, you can add a calculated column to transform these into a Boolean using Power Query.

The process of creation for a Binary Prediction model follows the same steps as other AutoML models, described in the section Configuring the ML model inputs above.

Binary Prediction Model Report

The Binary Prediction model produces as an output a probability that a record will achieve the outcome defined by the Boolean label value as True. The report includes a slicer for the probability threshold, which influences how the scores above and below the probability threshold are interpreted.

The report describes the performance of the model in terms of *True Positives*, *False Positives*, *True Negatives* and *False Negatives*. True Positives and True Negatives are correctly predicted outcomes for the two classes in the outcome data. False Positives are outcomes that had the actual Boolean label of value False but were predicted as True. Conversely, False Negatives are outcomes where the actual Boolean label value was True but were predicted as False.

Measures, such as Precision and Recall, describe the effect of the probability threshold on the predicted outcomes. You can use the probability threshold slicer to select a threshold that achieves a balanced compromise between Precision and Recall.

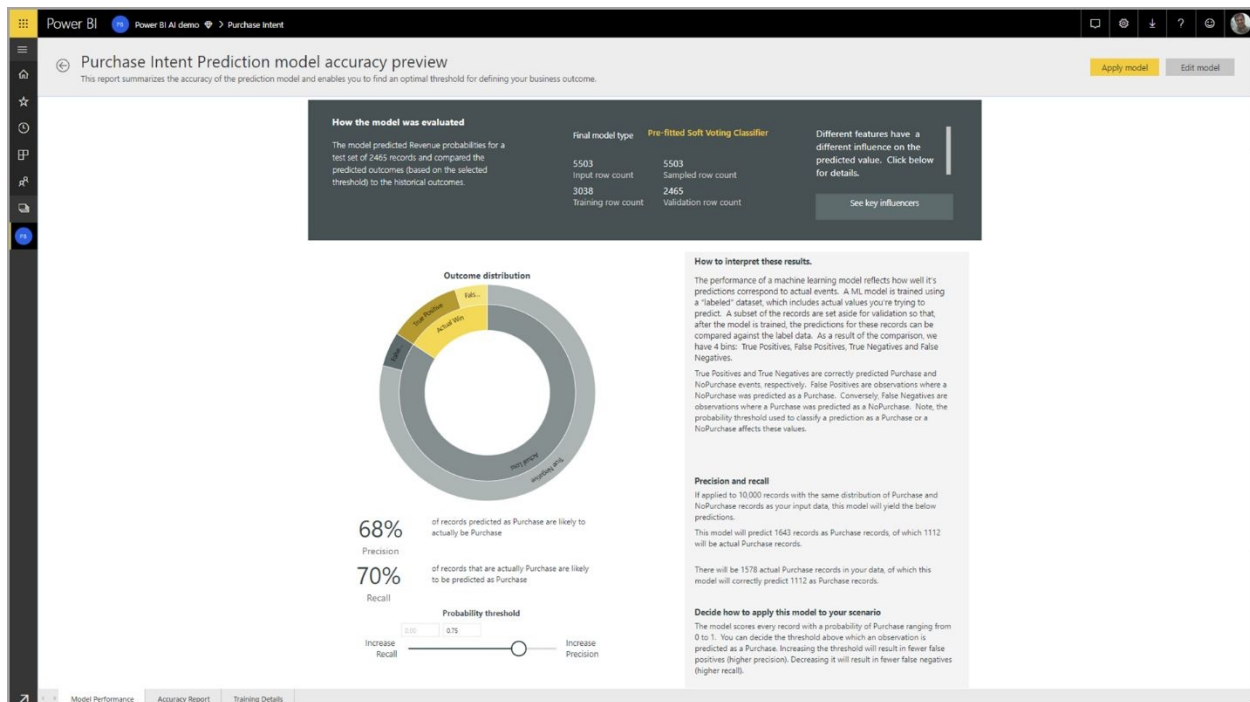


Figure 12-16: Binary Prediction Report

The Accuracy Report page of the model report includes the *Cumulative Gains* chart and the ROC curve for the model. These are statistical measures of model performance. The reports include descriptions of the charts shown.

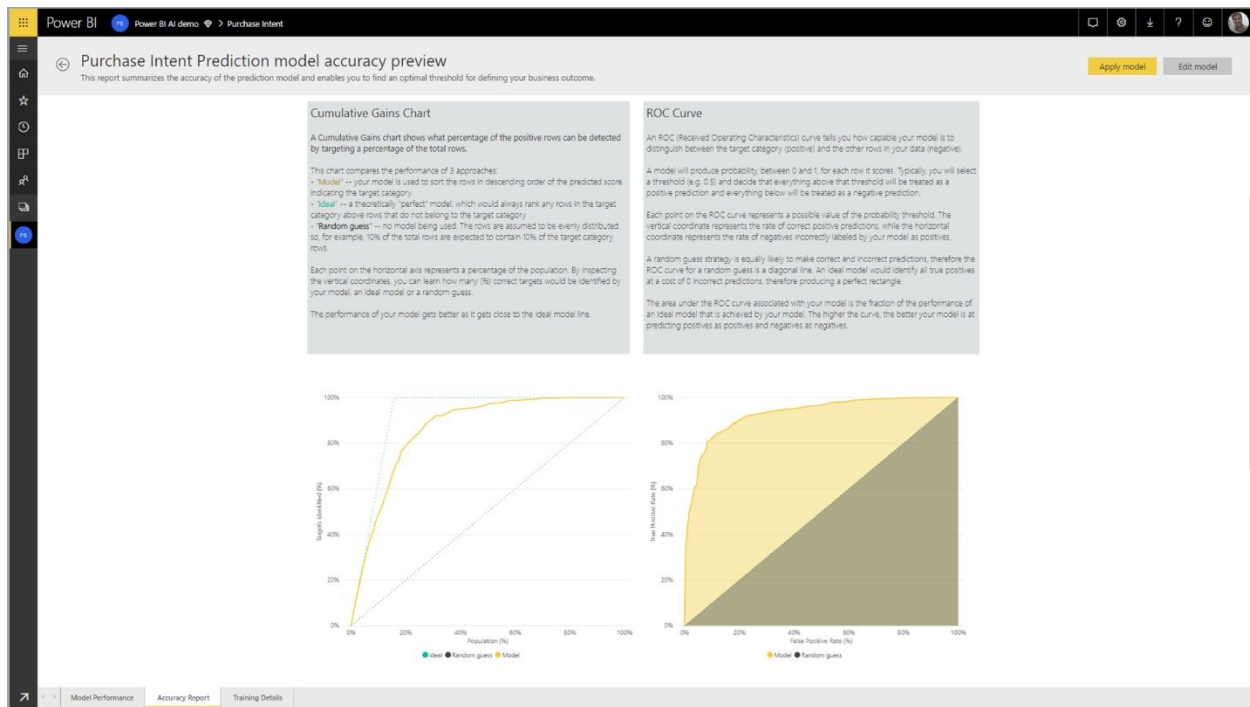
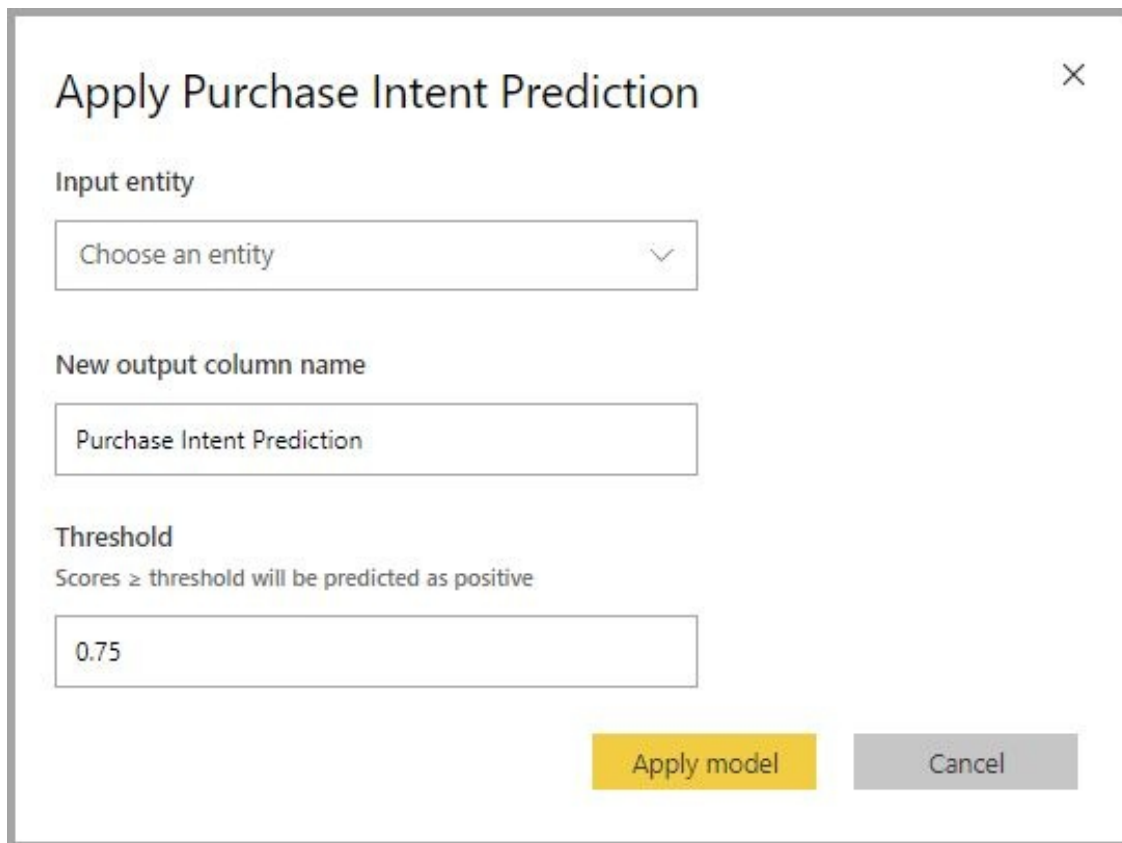


Figure 12-17: Binary Prediction Model Accuracy

Applying a Binary Prediction Model

To apply a Binary Prediction model, you must specify the entity with the data to which you want to apply the predictions from the ML model. Other parameters include the output column name prefix and the probability threshold for classifying the predicted outcome.



Apply Purchase Intent Prediction

Input entity

Choose an entity

New output column name

Purchase Intent Prediction

Threshold

Scores \geq threshold will be predicted as positive

0.75

Apply model Cancel

Figure 12-18: Applying Binary Prediction Model

When a Binary Prediction model is applied, it adds three output columns to the enriched output entity. These are the *PredictionScore*, *PredictionOutcome* and *PredictionExplanation*. The column names in the entity have the prefix specified when the model is applied.

The *PredictionOutcome* column contains the predicted outcome label. Records with probabilities exceeding the threshold are predicted as likely to achieve the outcome, and those below are predicted as unlikely to achieve the outcome.

The *PredictionExplanation* column contains an explanation with the specific influence that the input features had on the *PredictionScore*. This is a JSON formatted collection of weights of the input features for the prediction.

2- Classification Models

Classification models are used to classify a dataset into multiple groups or classes. They are used to predict events that can have one of multiple possible outcomes, such as whether a customer is likely to have a very high, high, medium, or low Lifetime Value; whether the risk for default is High, Moderate, Low or Very Low; and so on. The output of a Classification model is a

probability score, which identifies the likelihood that a record will achieve the criteria for a given class.

Training a Classification Model

The input entity containing your training data for a Classification model must have a string or numeric field as the historical outcome field, which identifies the past known outcomes.

Pre-requisites:

- A minimum of 50 rows of historical data is required for each class of outcomes

The process of creation for a Classification model follows the same steps as other AutoML models, described in the section Configuring the ML model inputs above.

Classification Model Report

The Classification model report is produced by applying the ML model to the holdout test data and comparing the predicted class for a record with the actual known class.

The model report includes a chart that includes the breakdown of the correctly and incorrectly classified records for each known class.

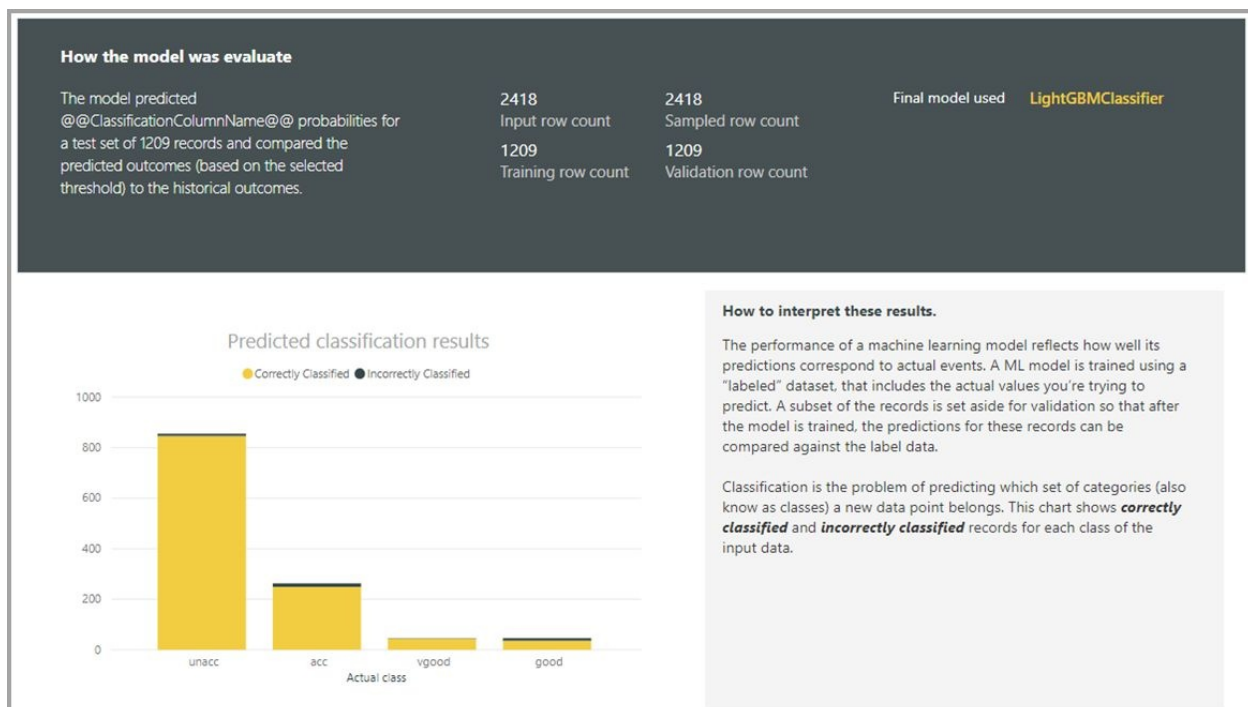


Figure 12-19: Classification Model Report

A further class-specific drilldown enables an analysis of how the predictions for a known class are distributed. This includes the other classes in which records of that known class are likely to be misclassified.



Figure 12-20: Classification Model Accuracy

The model explanation in the report also includes the top predictors for each class.

The Classification model report also includes a Training Details page similar to the pages for other model types, as described in the section AutoML model report earlier in this article.

Applying a classification model

To apply a Classification ML model, you must specify the entity with the input data and the output column name prefix.

When a Classification model is applied, it adds three output columns to the enriched output entity. These are the *PredictionScore*, *PredictionClass* and *PredictionExplanation*. The column names in the entity have the prefix specified when the model is applied.

The *PredictionClass* column contains the most likely predicted class for the record. The *PredictionScore* column contains the list of probability scores for the record for each possible class.

The *PredictionExplanation* column contains an explanation with the specific influence that the input features had on the *PredictionScore*. This is a JSON formatted collection of weights of the input features for the prediction.

3- Regression Models

Regression models are used to predict a value, such as the revenue likely to be realized from a sales deal, the lifetime value of an account, the amount of a receivable invoice that is likely to be paid, the date on which an invoice may be paid, and so on. The output of a Regression model is the predicted value.

Training a Regression Model

The input entity containing the training data for a Regression model must have a numeric field as the historical outcome field, which identifies the past known outcome values.

Pre-requisites:

- A minimum of 100 rows of historical data is required for a Regression model

The process of creation for a Regression model follows the same steps as other AutoML models, described in the section *Configuring the ML model inputs* above.

Regression Model Report

Like the other AutoML model reports, the Regression report is based on the results from applying the model to the holdout test data.

The model report includes a chart that compares the predicted values to the actual value. In this chart, the distance from the diagonal indicates the error in the prediction.

The residual error chart shows the distribution of the percentage of average error for different values in the holdout test dataset. The horizontal axis represents the mean of the actual value for the group, with the size of the bubble showing the frequency or count of values in that range. The vertical axis is the average residual error.

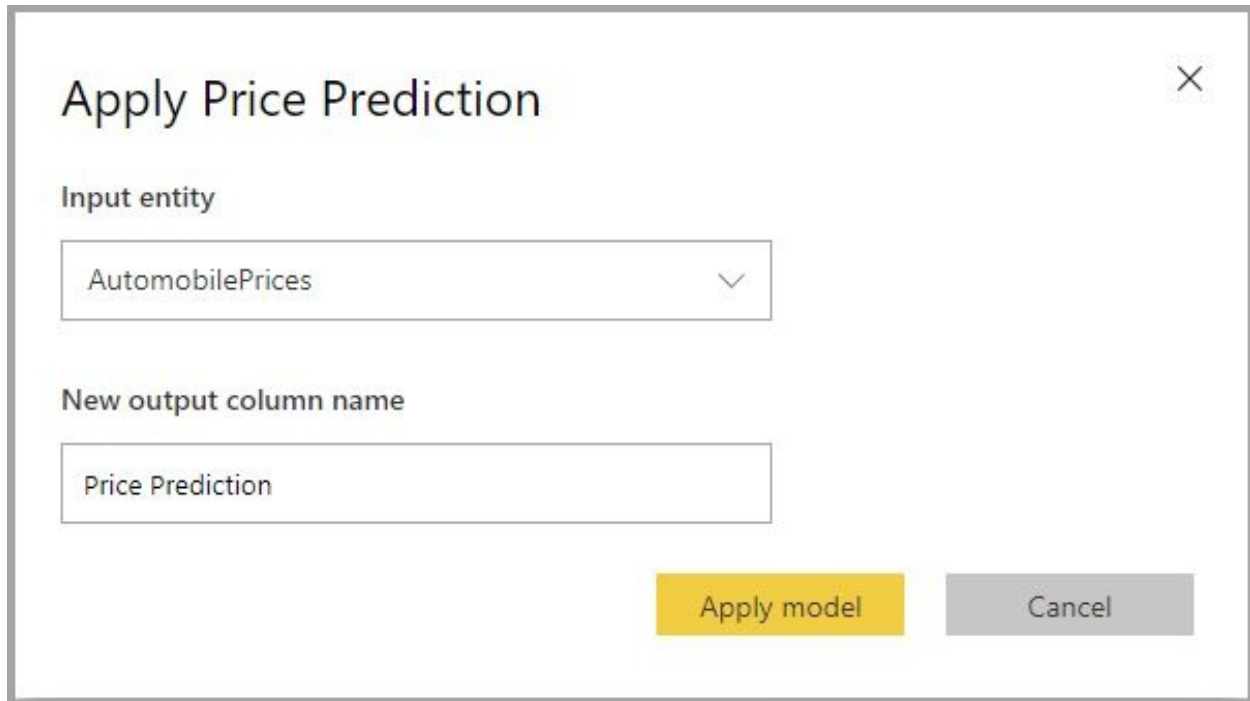


Figure 12-21: Regression Model Performance

The Regression model report also includes a Training Details page like the reports for other model types, as described in the section AutoML model report above.

Applying a Regression Model

To apply a Regression ML model, you must specify the entity with the input data and the output column name prefix.

A dialog box titled "Apply Price Prediction" with a close button (X) in the top right corner. It contains two input fields: "Input entity" with a dropdown menu showing "AutomobilePrices" and a downward arrow, and "New output column name" with a text box containing "Price Prediction". At the bottom right, there are two buttons: "Apply model" (yellow) and "Cancel" (gray).

Apply Price Prediction

Input entity

AutomobilePrices

New output column name

Price Prediction

Apply model Cancel

Figure 12-22: Apply Regression Model

When a Regression model is applied, it adds two output columns to the enriched output entity. These are the *PredictionValue*, and *PredictionExplanation*. The column names in the entity have the prefix specified when the model is applied.

The *PredictionValue* column contains the predicted value for the record based on the input fields. The *PredictionExplanation* column contains an explanation with the specific influence that the input features had on the *PredictionValue*. This is a JSON formatted collection of weights of the input features.

Summary

Automated machine learning, also referred to as AutoML, is the process of automating the time consuming, iterative tasks of ML model development. It allows data scientists, analysts, and developers to build ML models with high scale, efficiency, and productivity all while sustaining model quality.

Traditional ML model development is resource-intensive, requiring significant domain knowledge and time to produce and compare dozens of models. AutoML in Power BI empowers Citizen Data Scientists to accelerate the time it takes to get production-ready ML models with great ease and efficiency.

AutoML is available for dataflows in workspaces hosted on Power BI Premium and Embedded capacities only. If you don't have those options, you still can do Machine Learning with Power BI using Python/R, Integration with Azure Machine Learning Studio or Integration with Azure Machine Learning Services.

About the Author



Ashraf Ghonaim, Strategic Management Consultant at the City of Toronto. He holds a Computer Engineering degree and an MBA degree in Strategic Management with special emphasis on IT-Business Strategic Alignment using Balanced Scorecard. He is also a certified Balanced Scorecard Professional™ from the co-creators Drs. Kaplan and Norton, a Lean Six Sigma Black Belt and a Project Management Professional (PMP). Ashraf's areas of expertise are; Strategy Management, Performance Measurement, Process Improvement, Analytics and Visualization.

Ashraf is the leader of the Power Platform user group community in Toronto that has more than 1,400 active members. He is also the organizer of and speaker at the Power Platform World Tour events in Toronto and Montreal and also a frequent speaker at other Microsoft's events like Global AI Bootcamp, Azure Bootcamp, D365, SQL Saturdays, Sharepoint Saturdays, etc.

Ashraf also volunteers in several Open Data and Data for Public Good initiatives. Ashraf is very active in participating in datathons and hackathons as a mentor and a judge. He recently got his Microsoft MVP award in recognition of his contribution to the data and analytics community. Ashraf is an [MVP in Data Platform](#) (Power BI) with special focus on leveraging Machine Learning, AI Cognitive Services and Advanced Analytics Capabilities in Power BI to deliver impactful results.

Part V: Integration of Other Applications with Power BI

Chapter 13: Power BI REST API

Author: Eduardo Castro

Power BI service components can be configured and administered using its user interface on the web available at <http://app.powerbi.com>, however sometimes automation is needed in order to perform repetitive tasks or integration.

In this chapter we will explore how to use the Power BI REST API to administer and integrate Power BI with other applications. We will use C# to manage workspaces, permissions and other administration related task using Power BI REST API.

Getting ready to use Power BI REST API

The Power BI REST API is available to developers to take advantage of when you need to perform repetitive administration, including configuring and integrating Power BI with external applications. However, before you can use the REST API you need to have the proper permissions and configuration.

The first step is to register your application. This is a requirement because any application that wants to interact with the Power BI Service and with Azure must be registered and authorized in the Power BI and Azure tenants.

Register your developer application

To start the registration process, navigate to <https://dev.powerbi.com/apps>. In this first step you must first login using your corporate Power BI account and must follow the instructions to register and authorize your application.

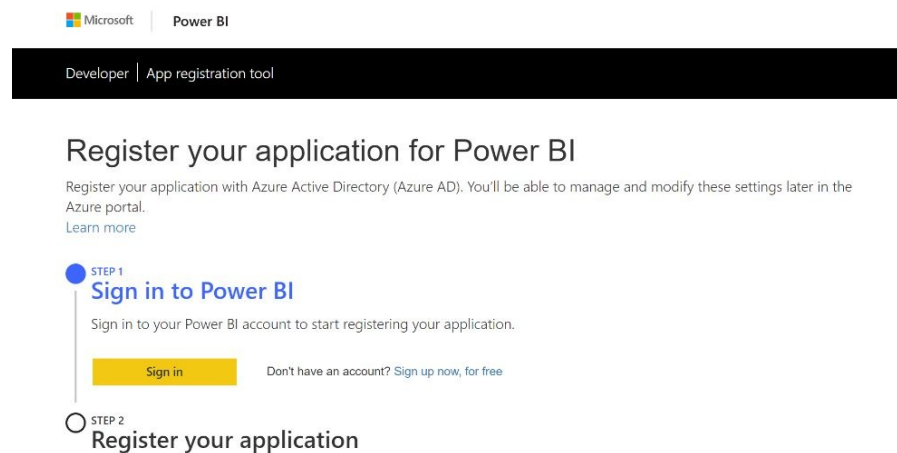


Figure 13-01: Beginning the Power BI Application Registration

Click on the “Sign In” button and login with your credentials.

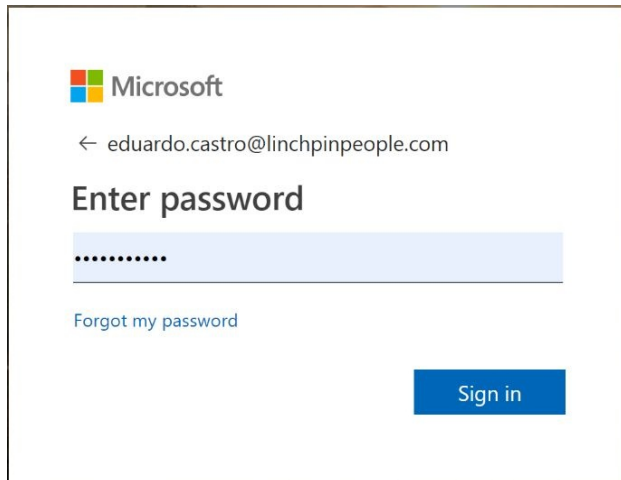


Figure 13-02: Logging with your credentials

After logging in you can continue with your app registration.

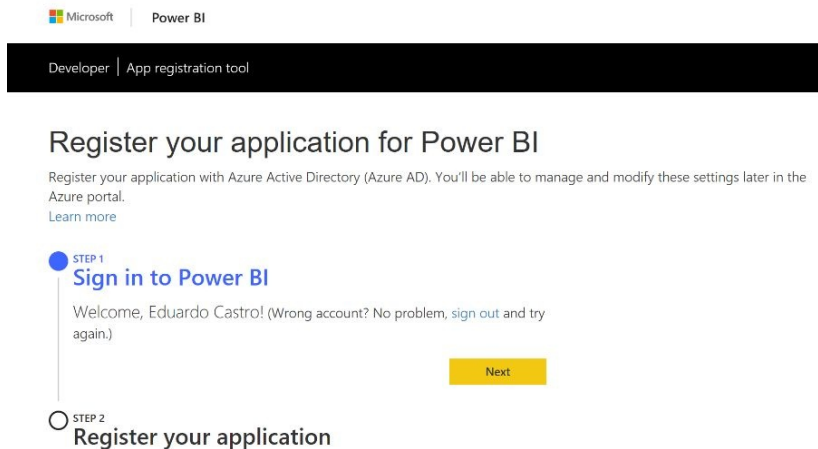


Figure 13-03: Begin App registration

The App Registration page requires the following information: your application name and the application type. For the application type you have two options:

- **Server-side Web App:** This use case is for applications that resides on a server. In this case it can be web applications or mobile applications.
- **Native App:** This use case is for applications that will run on a specific environment, for example a console application.

During the App Registration you must provide your application name and application URL and redirect URL.

STEP 1
Sign in to Power BI

STEP 2
Register your application

Register your application with Azure AD to allow your application to access the Power BI REST APIs and to set resource permissions for your application. You can change this later in the Microsoft Azure portal. [Learn more](#)

Application Name
Enter a display name to identify your application in Azure

Eduardo Power BI API

Application Type
Choose the type of application you are developing

Server-side web application (for web apps or web APIs)

Home Page URL
Enter your application's homepage URL

http://www.linchpinpeople.com

Redirect URL
Enter a URL where users will be redirected upon sign in so your application can receive an authorization code.

http://www.linchpinpeople.com

Figure 13-04: Basic app information

The application type is related with the integration of Power BI with your applications, this is because there are two ways to integrate Power BI into your application:

- **Integration with a token:** In this case your application is the one that authenticates with the Power BI Service, all the authentication process is done by the application, and end users do not even need a Power BI Account.
- **Integration without a token:** In this case, when the users want to access your application content, they will be prompted for a Power BI Account for authentication.

If your application is going to use integration with a token you must create the application as Native Application Type, it doesn't matter if the application you are creating is web-based, console or a mobile application.

The next step is to choose the proper permissions you want for your application, you must choose only the required permissions based on the operations you plan to use with the Power BI API. Your application can have access only to the "Read Only APIs" this is useful when you are creating an application that is going to use to generate reports about your Power BI Objects, but if your application needs to modify the content then you will need to give your application "Read and write APIs" permissions.

API access

Select the APIs and the level of access your application needs. You can change these settings later in the Azure portal.

[Learn more](#)

☒ Select all

Read only APIs [ⓘ]	Read and write APIs [ⓘ]	Create APIs [ⓘ]
<input checked="" type="checkbox"/> Read all datasets	<input checked="" type="checkbox"/> Read and write all datasets	<input checked="" type="checkbox"/> Create APIs
<input checked="" type="checkbox"/> Read all dashboards	<input checked="" type="checkbox"/> Read and write all dashboards	
<input checked="" type="checkbox"/> Read all reports	<input checked="" type="checkbox"/> Read and write all reports	
<input checked="" type="checkbox"/> Read all workspaces	<input checked="" type="checkbox"/> Read and write all workspaces	
<input checked="" type="checkbox"/> Read all capacities	<input checked="" type="checkbox"/> Read and write all capacities	
<input checked="" type="checkbox"/> Read all storage accounts	<input checked="" type="checkbox"/> Read and write all storage accounts	
<input checked="" type="checkbox"/> Read all dataflows	<input checked="" type="checkbox"/> Read and write all dataflows	
<input checked="" type="checkbox"/> Read all gateways	<input checked="" type="checkbox"/> Read and write all gateways	
<input checked="" type="checkbox"/> Read all Power BI apps		

By clicking Register, you agree to the [terms of use](#)

Note: An application registered here can't be used as a service principal. [Learn how to register a service principal](#)

Figure 13-05: Choosing the API access permissions

The last step is the register your application, once you have selected all the properties and permissions of your applications, you click the “Register” button and if everything is ok, you will receive a Client Secret and Client ID, copy & paste this information and save it, because you will need it in your code in order to interact with Power BI Service.

Success!

Your application has been registered.

Note: You can retrieve your client ID from the Azure portal, if needed. If you lose your client secret, you'll need to create a new one in the Azure portal

Application ID:

Application secret:

Figure 13-06: Power BI Application successfully registered

Register your application in Azure Portal

Another method to register your application is using the Azure Portal, to use this method you must go to <http://portal.azure.com> using this method gives your more control on your application and if you want you can always go to the Azure Portal and change the permissions of your application.

The first step is to login in <http://portal.azure.com>, once you are authenticated in the Portal you must select the Azure Active Directory Configuration, as shown below.

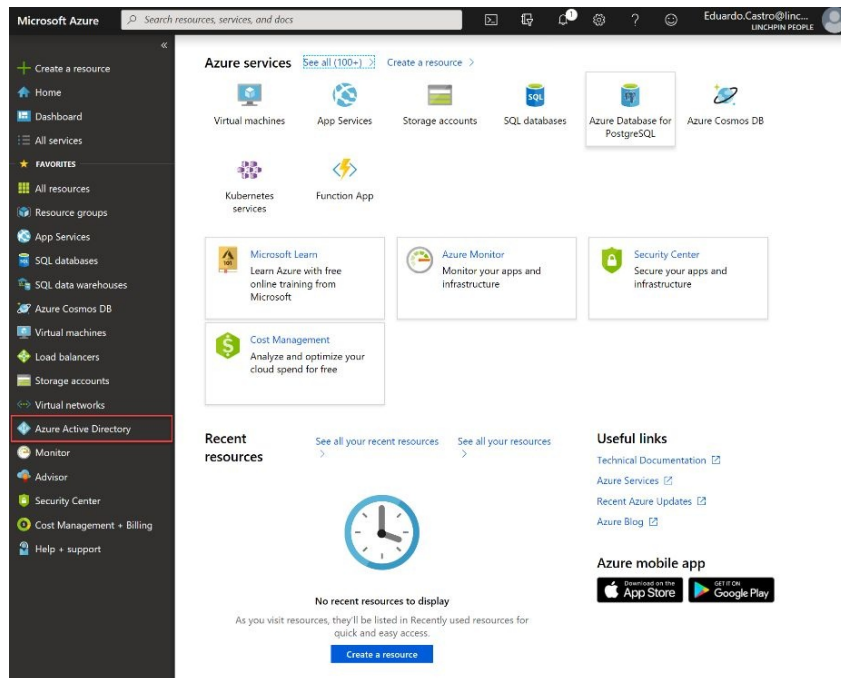


Figure 13-07: Power BI Application registration using the Azure Portal

The next step is to choose the App Registration Option.

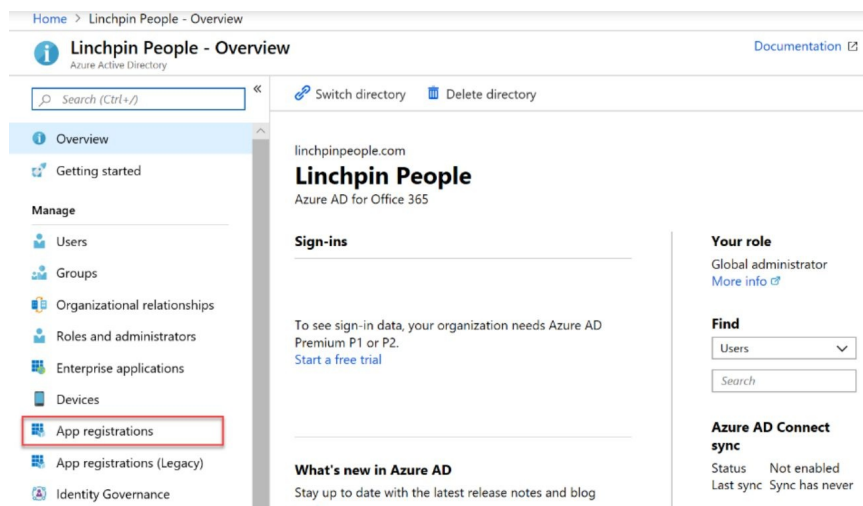


Figure 13-08: Application registration using the Azure Portal

In the App registrations page, you will see all your previously created applications and you can create new ones. In our case, we will create a new application registration.

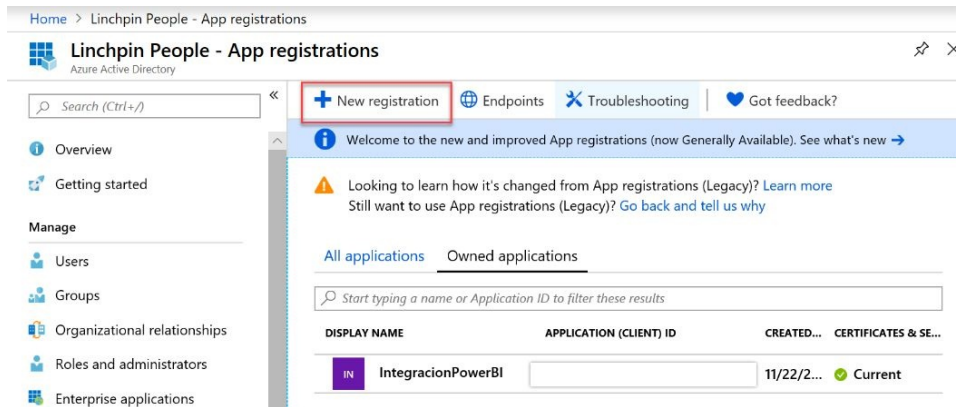


Figure 13-09: New application registration using the Azure Portal

Once you click New Registration, you must assign a name to your application and must choose one of the supported account types related to who can use this application or access this API.

- Accounts in this organizational directory only (Linchpin People). This option is the default if you are creating a line-of-business (LOB) application. This option maps your application to a single-tenant Azure AD.
- Accounts in any organizational directory
- Accounts in any organizational directory and personal Microsoft accounts (e.g. Skype, Xbox, Outlook.com)

If you want to restrict the use of the Application to only your internal users you must choose the first option. Once created you must assign the proper permissions to your application, to do so, click on API Permissions.

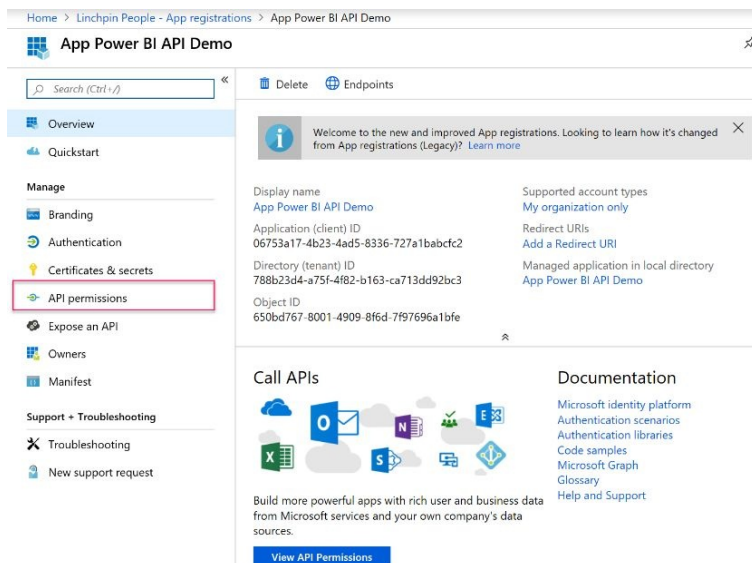


Figure 13-10: New application API Permissions

In the Request API permissions page, you must select Power BI Service to get the permission configuration related to Power BI.

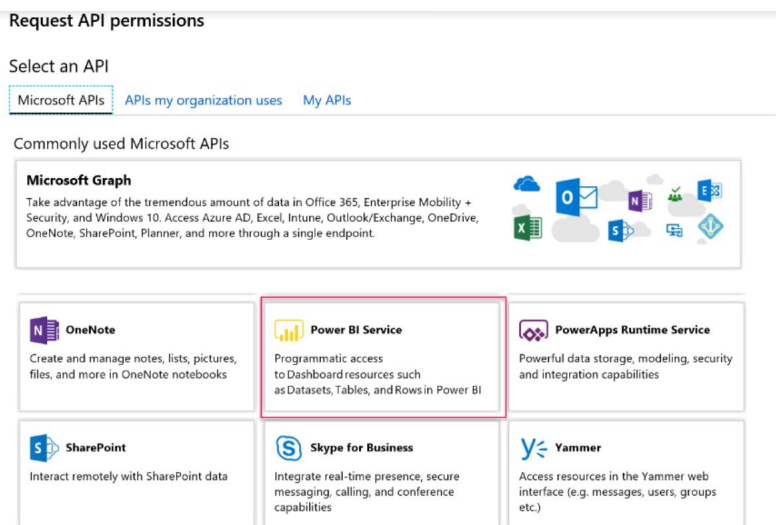


Figure 13-11: Power BI Service selection in permissions page

In the permission page you can assign or delegate your application one or more permission depending on the operations and interaction that your application will be doing using the REST API, in our case we will select all the permissions, as shown below.

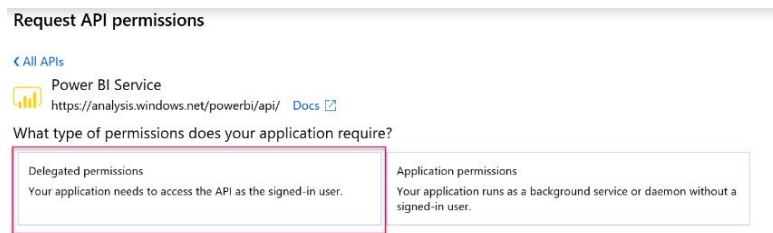


Figure 13-12: Power BI Service selection in permissions delegation

In the Delegated permissions page, you must select the permissions for your application and the click on Add permissions.



Figure 13-12: Power BI Service adding permissions

After you have added permissions a warning message is shown indicating that a tenant administrator must grant consent the permissions for your application.

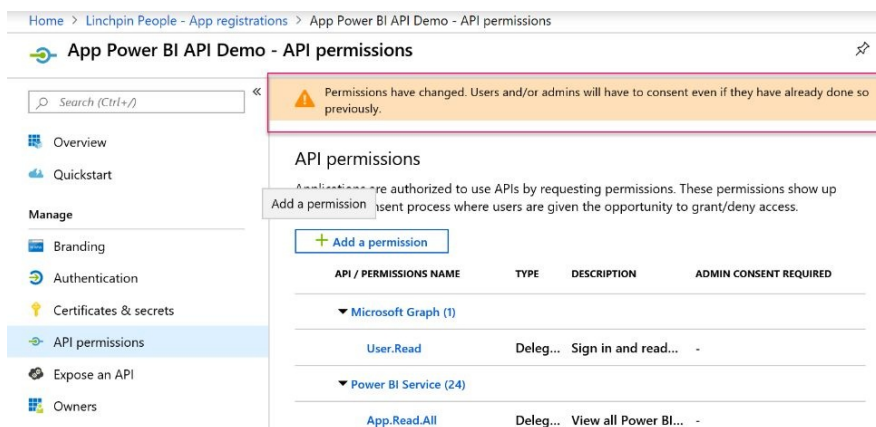


Figure 13-13: Power BI Service warns that administrator consent is required

If your are a tenant administrator you can grant consent yourself if not you will have to ask your tenant administrator to grant permissions consent to your application, as show below.

Grant consent

As an administrator, you can grant consent on behalf of all users in this directory. Granting admin consent for all users means that end users will not be shown a consent screen when using the application.



Figure 13-14: Administrator consent for the required permissions

After the permissions are granted the page will all the permissions that the application can use.

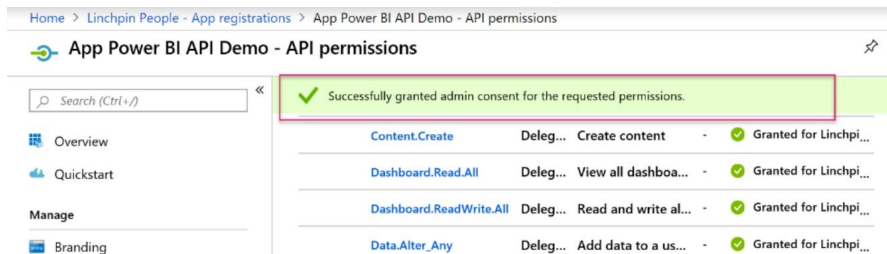


Figure 13-15: Administrator has granted consent for the required permissions

Before you can use the Power BI Rest API you must get the application secret. To do it click on “Certificates & secrets”

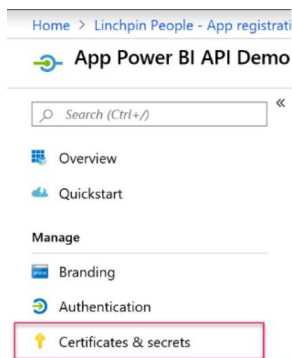


Figure 13-16: The application secret must be created before using the REST API

The next step is to create the Client Secret to be used in our code, select “Certificates & secrets” and then “New client secret”.

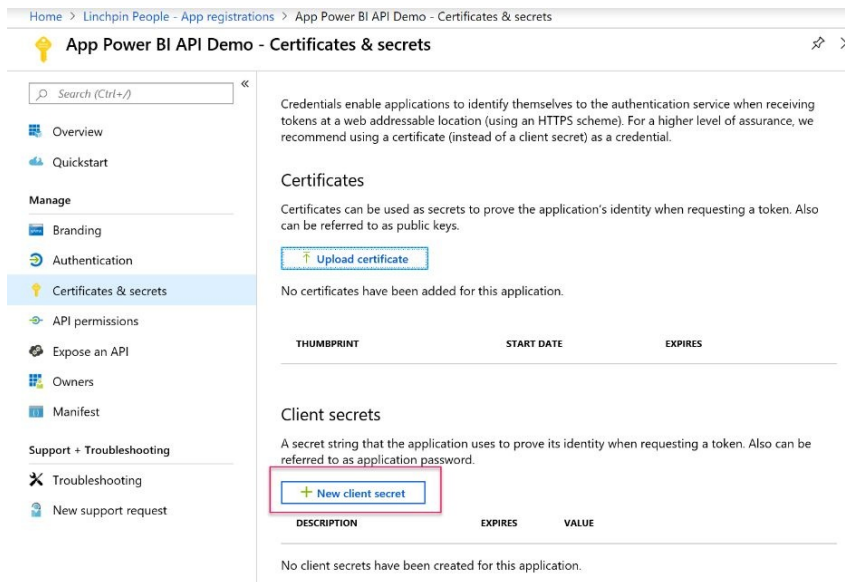


Figure 13-17: Creating of the client secret to be used with the REST API

In the client secret creation page, you must select the duration of the client secret, it can be 1 year, 2 years or without expiration.

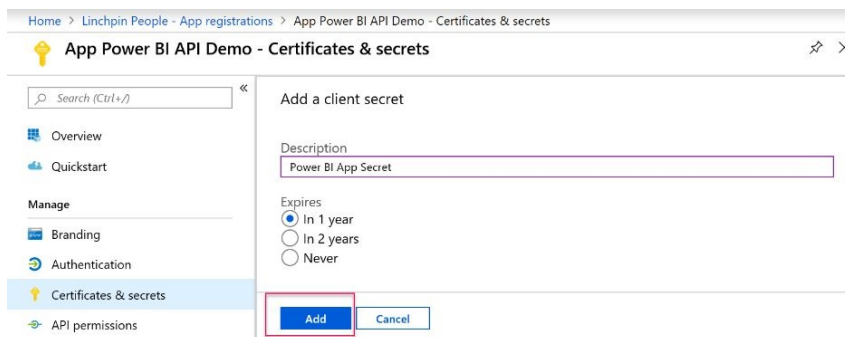


Figure 13-18: Client secret expiration configuration

After the client secret is created, it will be displayed next to the description field, be sure to write down the client secret because it will not be show after you navigate off this web page.

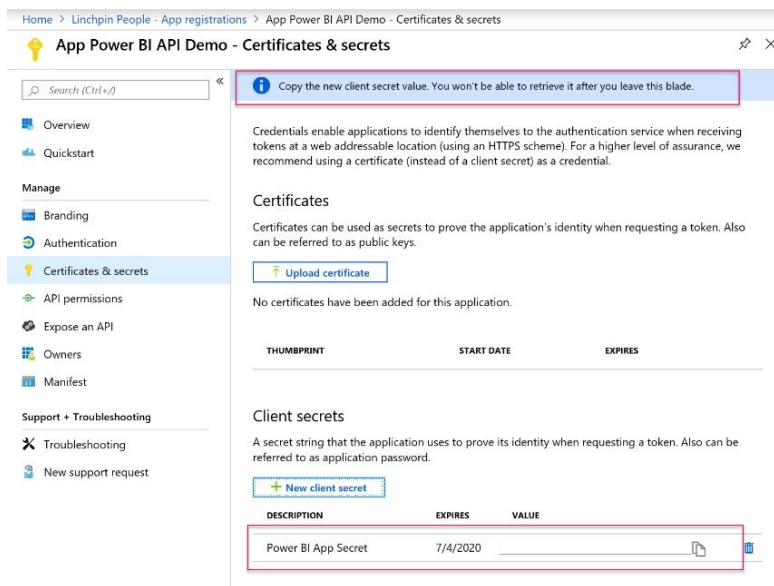


Figure 13-19: Client secret was created successfully

The final step is to configure our application to not use redirect URI, to do it go to Authentication section in the application configuration.

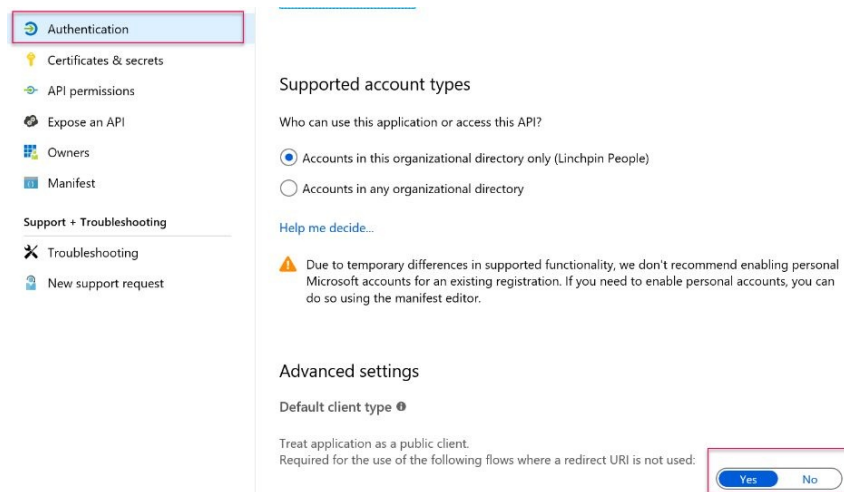


Figure 13-20: Configure your application to not use redirect URI

Now you are ready to start creating your C# application and to use the Power BI API.

Preparing Visual Studio to use the Power BI REST API

In this chapter we will use the sample code developed by Microsoft and that is available in GitHub at <https://github.com/Microsoft/PowerBI-Developer-Samples/tree/master/App%20Owns%20Data>, the code provided by Microsoft is free to use, and it will be base for our sample here.

Start by downloading or cloning the GitHub repository to your local machine, in our case we will download it.

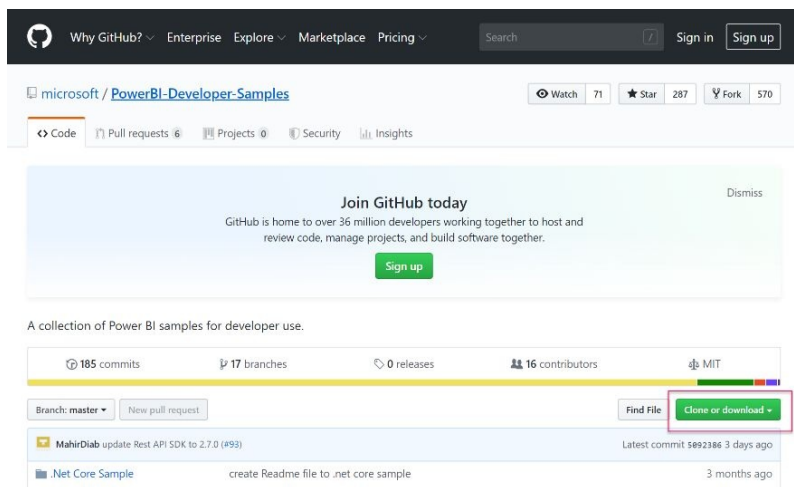


Figure 13-21: Download the sample code to your computer

Open the PowerBIEmbedded_AppOwnsData solution, this will be the base code we will be using.

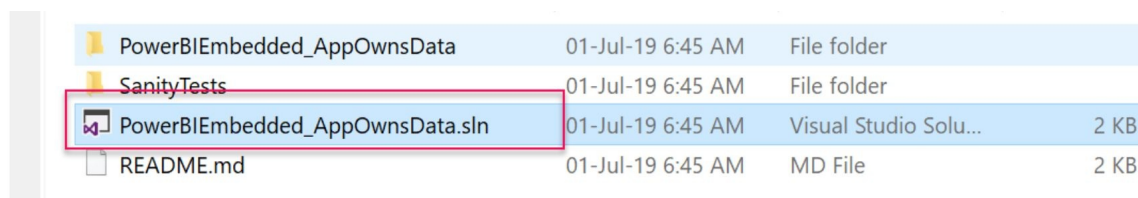


Figure 13-22: Sample code for using Power BI REST API

Before your application can use the Power BI REST API you must provide the proper configuration in the web.config file, the data you must fill are: [applicationId](#), [workspaceId](#), [pbiUsername](#), [pbiPassword](#), [applicationSecret](#) and [tenant](#)

```

<add key="AuthenticationType" value="MasterUser"/>
<!-- Common configuration properties for both authentication types -->
<add key="applicationId" value=""/>
<add key="workspaceId" value=""/>
<!-- The id of the report to embed. If empty, will use the first report in group -->
<add key="reportId" value=""/>

<!-- Fill Tenant ID in authorityUrl-->
<add key="authorityUrl" value="https://login.microsoftonline.com/common/">
<add key="resourceUrl" value="https://analysis.windows.net/powerbi/api"/>
<add key="apiUrl" value="https://api.powerbi.com/">
<add key="embedUrlBase" value="https://app.powerbi.com/">
</appSettings>

<MasterUser>
<!-- Note: Do NOT leave your credentials on code. Save them in secure place. -->
<add key="pbiUsername" value=""/>
<add key="pbiPassword" value=""/>
</MasterUser>

<ServicePrincipal>
<!-- Note: Do NOT leave your app secret on code. Save it in secure place. -->
<add key="applicationSecret" value=""/>
<add key="tenant" value=""/>
</ServicePrincipal>

```

Figure 13-23: Web.config configuration to use Power BI REST API

To get these values please get back to the Azure Portal and click on the application we had just created, the information we need will be displayed.

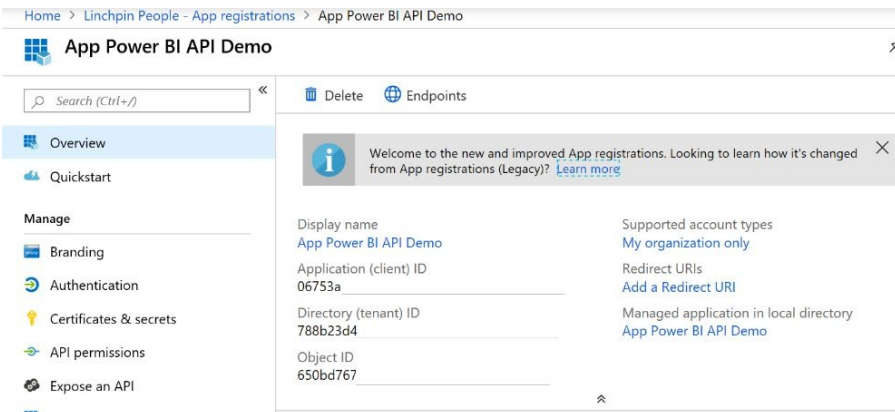


Figure 13-24: Information needed to be included in the web.config of your application

The following code invokes the Authentication Method and as result you will get valid token credentials that will be used in the following methods to invoke the REST API.

```

// Get token credentials for user
var getCredentialsResult = await GetTokenCredentials();
if (!getCredentialsResult)
{
    // The error message set in GetTokenCredentials
    return false;
}

```



```
}
```

To begin using the REST API we use the PowerBIClient class, this object has the methods to call the Power BI REST API. The following code uses the PowerBIClient class and list the reports in the workspace.

```
// Create a Power BI Client object. It will be used to call Power BI APIs.
using (var client = new PowerBIClient(new Uri(ApiUrl), m_tokenCredentials))
{
    // Get a list of reports.
    var reports = await client.Reports.GetReportsInGroupAsync(WorkspaceId);

    // No reports retrieved for the given workspace.
    if (reports.Value.Count() == 0)
    {
        m_embedConfig.ErrorMessage = "No reports were found in the workspace";
        return false;
    }

    Report report;
    if (string.IsNullOrEmpty(ReportId))
    {
        // Get the first report in the workspace.
        report = reports.Value.FirstOrDefault();
    }
}
```

We can continue using the REST API depending on our needs, for example the following code lists the datasets inside a specific workspace and gets information about roles.

```
var datasets = await client.Datasets.GetDatasetByIdInGroupAsync(WorkspaceId, report.DatasetId);
m_embedConfig.IsEffectiveIdentityRequired = datasets.IsEffectiveIdentityRequired;
m_embedConfig.IsEffectiveIdentityRolesRequired = datasets.IsEffectiveIdentityRolesRequired;
```

The following code uses the Power BI REST API to get the list of dashboards.

```
// Create a Power BI Client object. It will be used to call Power BI APIs.
using (var client = new PowerBIClient(new Uri(ApiUrl), m_tokenCredentials))
{
    // Get a list of dashboards.
    var dashboards = await
        client.Dashboards.GetDashboardsInGroupAsync(WorkspaceId);

    // Get the first report in the workspace.
```



```
var dashboard = dashboards.Value.FirstOrDefault();

if (dashboard == null)
{
    m_embedConfig.ErrorMessage = "Workspace has no dashboards.";
    return false;
}
}
```

Depending on your needs you can continue using the REST API using the methods implemented in the PowerBIClient class.

Summary

Power BI Service gives you the user interface to administrative tasks but you can use the REST API in order to integrate Power BI with your applications. In this chapter we gave you the steps on how to do configuration to be able to use the Power BI REST API and also, we gave some starting examples on how to use it inside C#.

About the Author



Eduardo Castro is an Enterprise Chief Architect, Microsoft Regional Director and Microsoft Data Platform MVP, living in San José Province, Costa Rica.

With more than 20 years of experience in IT projects in public sector and large companies, Eduardo has helped the companies to achieve their best using technology based on Windows or Linux. A fan of new technology, entrepreneurship, and travel, he enjoys drinking good cup of coffee. Eduardo is a regular speaker in several Microsoft Conferences and Community Events in USA and Latin America.

Chapter 14: Real-Time Streaming Datasets

Author: Manohar Punna

Power BI helps you build appealing visualizations of your data. With an ever-growing footprint of data, it is essential to get real-time insights into your data. These insights can be as simple as monitoring a single metric or as complex as viewing real-time sales performance across multiple locations. Power BI real-time streaming datasets enable you to stream data and update dashboards in real-time. Any time-sensitive data can be a source of streaming datasets such as IoT sensor devices, social media sources, and service usage metrics.

In this chapter, I will introduce different types of streaming datasets and step-by-step implementation of these datasets using various examples. By the end of this chapter, you will be able to create streaming datasets, push data into them, and visualize data from them in Power BI.

Introduction

Real-time analytics has become the norm for most businesses. The application of real-time streaming is useful in monitoring sensor data, social media trends, and metrics from any time-sensitive analysis on data captured in real-time.

With this requirement, there is a need to address the complexity of implementing real-time streaming solutions. Microsoft Power BI provides real-time streaming datasets as a solution to this requirement. Streaming datasets can be created to collect the transmitted data and analyze them using real-time streaming visuals. These visuals can be real-time on dashboards and can also be used in building reports.

The definition of “real-time” varies among industries and businesses. For example, in a manufacturing factory, detecting an anomaly in the product manufactured should be done before the product is packaged and sent. Whereas in the same factory, monitoring pressure levels are critical and need to be fixed within a few minutes. As an architect designing a real-time solution, it is crucial to identify and record these requirements and service level agreements to design a suitable real-time solution.

Real-Time Datasets

In Power BI, there are three types of real-time datasets:

- Push dataset
- Streaming dataset
- PubNub streaming dataset

These datasets are designed to display real-time streaming data on dashboards. First, let us see how these datasets work. Later we can look at pushing data into these datasets and building visuals from these datasets.

Push Datasets

Push dataset, as the name suggests, allows to push data into the Power BI service. When you create this dataset, Power BI provisions a database to host the data pushed into this dataset. This database is only accessible as a dataset in Power BI. There is no way to connect to this database directly.

The data stored in the dataset can be used to build visuals in reports. These visuals can then be pinned to dashboards to display real-time data.

Note: Visuals created using real-time streaming datasets refresh in real-time on dashboards only. The Power BI service triggers the refresh of a dashboard tile when the data refreshes on the real-time dataset.

Once you pin a visual created on a push dataset to a dashboard, you can also perform Q&A in natural language on the dataset. You can pin the resulting visual as a live tile on the dashboard.

It is important to note that when you pin the live page that hosts the visuals created on real-time datasets to a dashboard, the data does not refresh in real-time on the live page.

Advantages and Disadvantages

The main advantage of push datasets is to build visuals in reports similar to standard datasets. The data is saved forever in the provisioned database. Hence, the reports can be built to analyze historical data for in-depth analysis in addition to real-time analysis.

The disadvantage of having to push data into a database is that the data refresh is not instantaneous. You can expect to see a latency of 3-5 seconds for the data to appear on the visual on the dashboard from the time data is pushed from the source.

Data ingestion is at the rate of 1 request per second with up to 16 MB per request. The throughput is limited to 1 million rows of data per hour. If your data is more extensive than these limits, it is recommended to push aggregated data from the source.

Streaming Datasets

A streaming dataset is the pure flavor of a real-time dataset. The latency is minimal and is built only for real-time analysis. The dataset is provisioned in Power BI using a temporary cache. This mechanism helps reduce the latency and provides near real-time data access in Power BI. As this is a temporary cache, the data is hosted up to one hour, hence providing a transient real-time trend for visuals like a line chart.

The visuals can be created only on a dashboard using the *Add Tile* functionality. You cannot create a visual using this dataset in reports. All the report-specific functions like filtering and custom visuals are not available when using the streaming dataset.

Streaming datasets are available under the **custom streaming data** option as a data source and is visible when you create a live tile on the dashboard.

Advantages and Disadvantages

The advantage of using streaming dataset is that it provides very little latency. There is no database to host the data, which reduces the latency of writing and reading from a database.

The disadvantages are that there is no access to historical data, and there are limited visuals that you can create on a dashboard using live tiles. The data is saved temporarily for up to one hour.

Data ingestion is at the rate of 5 requests per second with up to 15 KB per request. There is no throughput limit for streaming dataset, while the size of the request is less than that of push dataset. This limit is set because it targets scenarios where you would want to use streaming datasets to view data which is meaningful for real-time analysis as-is, like temperature readings or other sensor data to detect any spikes.

PubNub Datasets

PubNub is a streaming service provided by PubNub Inc. These datasets are useful if you are already using PubNub for your real-time solutions and would like to integrate it into your reporting in the Power BI service. The Power BI service uses the PubNub SDK to connect to the PubNub data stream. PubNub

hosts the data and pushes to Power BI through the PubNub data stream.

PubNub datasets are streaming datasets where the data does not reside on Power BI. So, you cannot build reports using this dataset. The only available option to use this dataset is to use live tiles in the dashboard.

The refresh of live tiles is almost instantaneous as the service connects directly to the data stream. As the data is not hosted on Power BI service, there are no limits defined on these datasets. The limits are defined in PubNub service.

Creating Real-Time Datasets

Real-time datasets can be created on the Power BI service using the following methods:

- Power BI Service UI
- Power BI REST API
- Azure Stream Analytics
- PubNub

Power BI Service UI

Real-time datasets can be created using the UI on the Power BI Service. You need to follow the below steps to create a new real-time dataset:

1. Click on any workspace in the Power BI portal.
2. Click on **Create** and select **Streaming dataset**.

Figure 14-01: Create – Streaming dataset

3. Select API to create a **Push** or **Streaming** dataset using UI in Power BI service.

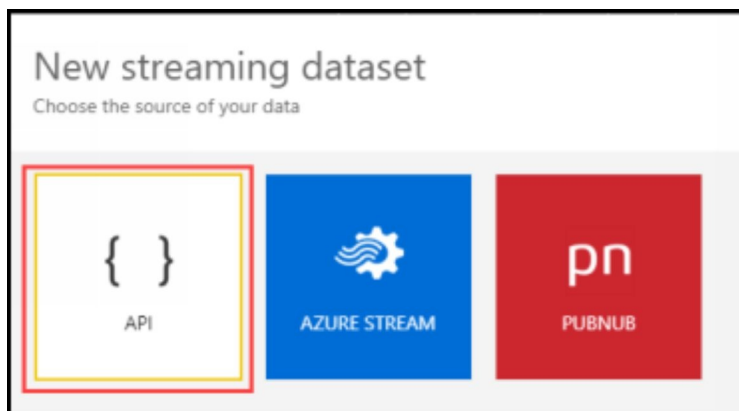


Figure 14-02: New streaming dataset – Choose API

4. In the **New streaming dataset** screen, provide the **Dataset name**. Under **Values from stream**, provide the columns in the dataset with an appropriate data type.

Figure 14-03: New streaming dataset

sample JSON string shows up as you type in the columns and the data type. This JSON string is the format for the data that you pass as payload to push data

to the dataset.

The option **Historic data analysis** selection decides if the dataset is push or streaming. If this option is selected, the data is saved forever in a database, making it a **push dataset**. If you do not opt to select this option the dataset is saved temporarily in a cache, making it a **streaming dataset**.

Click **Create** button to create the new streaming dataset.

5. On the **Streaming dataset created** screen, the URL to use for pushing the data into the dataset is provided. Also, sample scripts are provided in **cURL** and **PowerShell** syntaxes to push data to this dataset.

Figure 14-04: Streaming dataset created

Once the dataset is created, we can start creating visuals in dashboards (and reports in case of push dataset).

Power BI REST API

The Power BI REST API provides the functionality to create datasets and post data to the datasets. The dataset can be created by using PostDataset API using the following URLs for the POST request. The body passed with the POST request contains the structure of the dataset that is created.

My Workspace: <https://api.powerbi.com/v1.0/myorg/datasets>

App Workspace: <https://api.powerbi.com/v1.0/myorg/groups/{groupid}/datasets>

Request Body:

```
{
  "name": "Streaming_MVPDD",
  "defaultMode": "PushStreaming"
  "tables": [
    {
      "name":
      "RESTStreaming_MVPDD",
      "columns": [
        {
```

```
        "name": "state",  
        "dataType": "string"  
    },  
    {  
        "name": "value",  
        "dataType": "Int64"  
    }  
]  
}  
]
```

Azure Stream Analytics

In the first two methods of creating a streaming dataset, you need to create the dataset knowing the structure of the dataset. When pushing data from Azure stream analytics job, the dataset is created based on the data passed from the query output. If a dataset exists in Power BI with the same name, the dataset is recreated, and any structure that exists in Power BI is overwritten with the new structure from stream analytics job.

To create a streaming dataset using Azure Stream Analytics, you need to follow the steps below.

1. Setup an Event Hub to which you can push data using the steps provided [here](#)
2. *Create an Event hub – <https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-create>*
3. *Send and receive events - <https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-dotnet-standard-getstarted-send>*
4. Setup a Stream Analytics job with Event Hub as the source and Power BI as a target for the query.

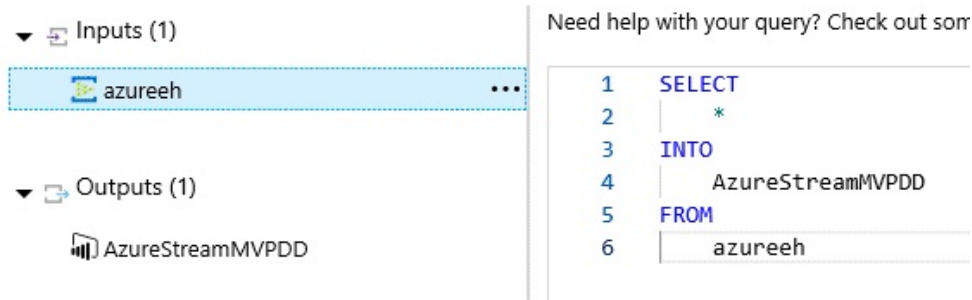


Figure 14-05: Stream analytics job – Query

5. Start the Stream Analytics job and start pushing the data into the event hub.
6. The Power BI service creates the new dataset when the first set of data arrives into the Power BI service.

Figure 14-06: Edit streaming dataset

PubNub

You can create Power BI streaming datasets that connect to a PubNub data stream by selecting **PUBNUB** when creating a new streaming dataset.

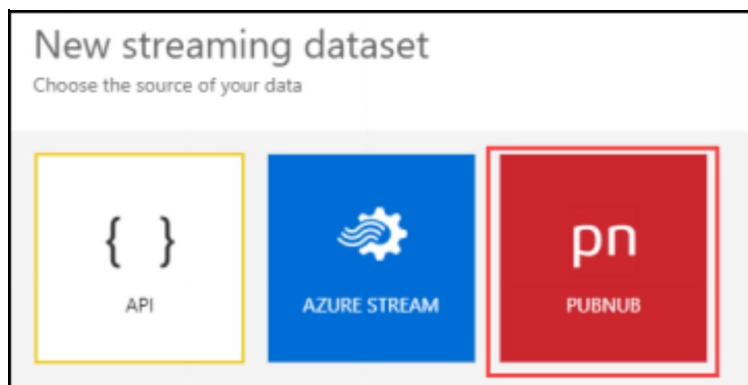


Figure 14-07: New streaming dataset – Choose PUBNUB

To connect to a PubNub data stream, you need to provide the **Sub-key** and **Channel** name from PubNub data stream.

New streaming dataset

For customers of the PubNub data stream network, subscribe to a channel to display data on your dashboard. [Learn more about PubNub.](#)

Dataset name *

Sub-key *

Channel name *

PAM Auth Key

Figure 14-08: New streaming dataset – PubNub dataset

A sample data stream is available to test the PubNub data stream.

Sub-key - *sub-c-5f1b7c8e-fbee-11e3-aa40-02ee2ddab7fe*

Channel - *pubnub-sensor-network*

Push Data to Streaming Datasets

Once the push or streaming dataset is created using one of the above methods, data can be pushed using HTTP post requests using Power BI REST API calls. For the scope of this chapter, let us detail the steps to push data using PowerShell scripts with an example.


The sample script provided when creating a dataset using Power BI service UI can be used as a reference to write the post requests. The sample scripts are accessible after creating the streaming dataset by clicking the  icon on the dataset.



Figure 14-09: Dataset properties

```
$endpoint = "https://api.powerbi.com/beta/00029a0b-28a4-4b60-aed5-b8b504506e77/datasets/f77(
ec9b2241ab18/rows?
key=Ljvs8eMHX%2F1WvGp4Hq2ejKoCfe4uXqmadPJjIewHdThq0rI1Ch%2B5m7L1AXEj3YdS
$payload = @{
    "state" = "AAAAA55555"
    "value" = 98.6
}
Invoke-RestMethod -Method Post -Uri "$endpoint" -Body (ConvertTo-Json @($payload))
```

For simulating a real-time workload, I have a database where I insert the data in a loop. I also have a procedure which reads the data from the table and marks the data as “read” once it has been read. The SQL script for the schema is as follows.

```
--Tables
CREATE TABLE [dbo].[StreamTest](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [EventTime] [datetime] NULL,
    [StateName] [nvarchar](30) NULL,
    [Value] [int] NULL,
    [ReadStatus] [bit] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[States](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [StateName] [nvarchar](3) NULL
) ON [PRIMARY]
```

```

INSERT INTO dbo.States
VALUES
('NSW'),('VIC'),('QLD'),('SA'),('TAS'),('WA'),('ACT'),('NT')

-- Proc to insert data into StreamTest table
CREATE PROCEDURE [dbo].[StreamInsert_prc]
    (@value INT = NULL
    ,@StateName NVARCHAR(3) = NULL)
AS
BEGIN
    --DECLARE @value INT = NULL, @StateName NVARCHAR(3) = NULL
    IF OBJECT_ID('StreamTest') IS NULL
        CREATE TABLE StreamTest
            (id INT IDENTITY
            ,EventTime DATETIME
            ,StateName NVARCHAR(30)
            ,Value INT
            ,ReadStatus BIT DEFAULT 0)

    INSERT INTO StreamTest (EventTime, Value, StateName)
    SELECT --GETDATE()
        CAST((GETDATE() AT TIME ZONE 'UTC') AT TIME ZONE 'AUS Eastern
        Standard Time' AS DATETIME)
        , ISNULL(@value,CAST(RAND()*100 AS INT))
        , ISNULL(@StateName, (SELECT StateName FROM States WHERE id =
        CAST(RAND()*100 AS INT)%8+1))
END
GO

-- Proc to read data for push dataset

CREATE PROCEDURE [dbo].[StreamRead_prc]
AS
BEGIN
    DECLARE @id INT

    SELECT @id = MAX(id)
    FROM StreamTest
    WHERE ReadStatus = 0

    SELECT EventTime, Value, StateName AS State
    FROM StreamTest
    WHERE id <= @id AND ReadStatus = 0

```

```

ORDER BY EventTime
FOR JSON AUTO

UPDATE StreamTest
SET ReadStatus = 1
WHERE id <= @id

END
GO

```

Run the following script to insert data in a loop.

```

WHILE (1=1)
BEGIN
    EXEC StreamInsert_prc
    WAITFOR DELAY
    '00:00:01'
END

```

To push the data in increments, modify the sample PowerShell script to run in a loop. This script picks the incremental data from the database and pushes it to the streaming dataset in Power BI. This method is the same for both push and streaming datasets.

You can start creating visuals in Power BI, once this data appears on the dataset.

```

#Copy endpoint for push dataset
$endpoint = "https://api.powerbi.com/beta/00029a0b-28a4-4b60-aed5-b8b504506e77/datasets/f77/
ec9b2241ab18/rows?
key=Ljvs8eMHX%2F1WvGp4Hq2ejKoCfe4uXqmadPJjIewHdThq0rI1Ch%2B5m7L1AXEj3YdS

#Run in a loop for 100 iterations
For ($i=0; $i -le 100; $i++)
{
    $conn = New-Object System.Data.SqlClient.SqlConnection
    $conn.ConnectionString = "Data Source=<database server>;Initial Catalog=<database>;user=<

#connect to the database to run the read proc
$conn.Open()
$table = new-object "System.Data.DataTable"
$tableAgg = new-object "System.Data.DataTable"

$cmd = $conn.CreateCommand()
$cmd.CommandText = "EXEC StreamRead_prc"
$result = $cmd.ExecuteReader()

#Push the json result form proc to streaming dataset
if ($result.HasRows -eq "true")

```



```
{
    $table.Load($result)

    $payload = $table[0].Rows[0][0]
    $payload.substring(1,$payload.Length-2)

    Invoke-RestMethod -Method Post -Uri "$endpoint" -Body $payload
}
$result.Close()

start-sleep -s 3
echo "iteration $i completed"
$conn.Close()
}
```

The above PowerShell code or code using other supported languages can be integrated into your existing application to push data into streaming datasets in Power BI.

In case of using an Azure Stream Analytics job or using PubNub as the data source, there are no additional steps required to push data into streaming datasets. The Stream Analytics job pushes data to the streaming datasets. The PubNub SDK that connects to the PubNub data stream triggers data refresh on live tiles as the data refreshes in the PubNub data stream.

Visualizing Real-Time Datasets

There are two ways you can visualize streaming data on a Power BI dashboard:

- The first method is using streaming datasets that are created using the UI, Azure Stream Analytics, or PubNub directly on the dashboard to create a tile.
- The second method is to create reports using the push dataset and pin the visuals to the dashboard.

Streaming datasets

1. On a dashboard where you want to visualize streaming data, click on + **Add Tile**.

Figure 14-10: Dashboard – Add tile

2. On **Select source** screen, under **REAL_TIME DATA**, select **Custom Streaming Data** and click **Next**.

Figure 14-11: Dashboard – Add tile – Select source

3. On the **Choose a streaming dataset** screen, under **YOUR DATASETS**, select any push or streaming dataset that exists. In this example, let us choose the sample PubNub dataset.

Figure 14-12: Dashboard – Add tile – Choose streaming dataset

4. On the **Visualisation Design** screen, select the **Visualization Type**, **Axis**, **Legend**, and **Values** as applicable based on the visualization type.

Figure 14-13: Dashboard – Add tile – Visualization design

The time window can be adjusted to display data for up to one hour.

The types of visuals that can be created on a dashboard using the **add tile** function is limited. You can create the following types of visualizations:

- Card
- Line chart
- Clustered bar chart
- Clustered column chart
- Gauge

5. On the **Tile details** screen, change the **Title** and **Subtitle** as needed. You can also link the visual to an URL or redirect to a report or dashboard within the same workspace.

Figure 14-14: Dashboard – Add tile – Tile design

After adding the visual, the data on the visual refreshes in real-time as the PubNub data stream refreshes.

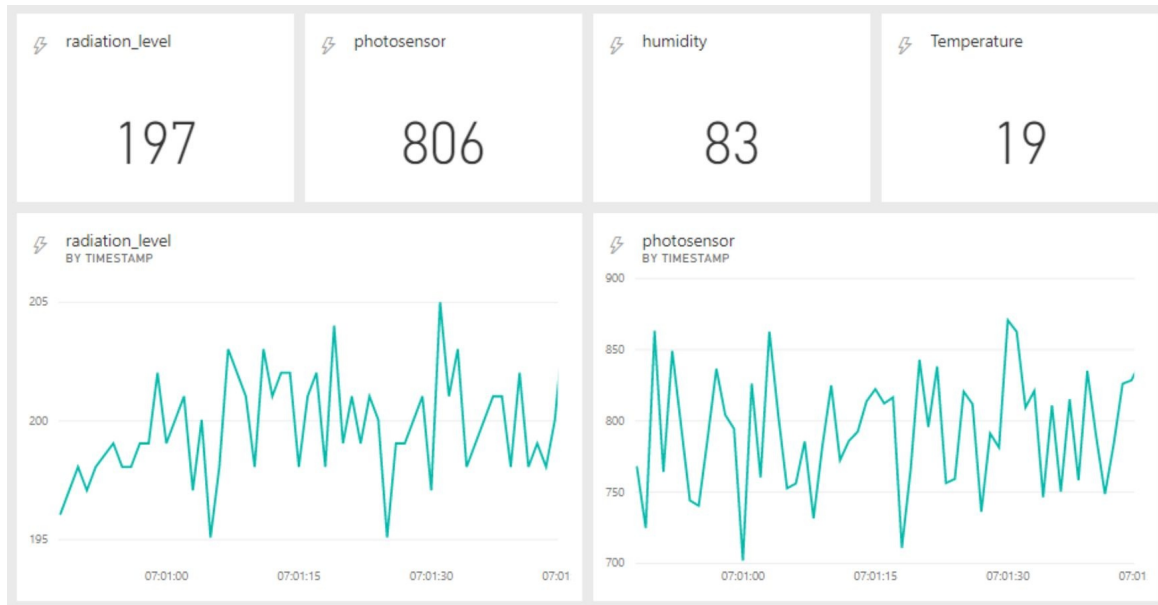


Figure 14-15: Real-Time Dashboard

Push datasets

This method gives you access to more visuals that are not available when creating live tiles on the dashboard. As mentioned earlier, there is a small latency in the data displayed using a push dataset.

1. Click on the **Create report** icon on the push dataset.

Figure 14-16: Push dataset – Create report

Note: Only the push dataset has this option, which shows that you can start creating reports using this dataset. Streaming datasets cannot be used to create reports.

2. Create a visual on the report of the type that is not available in the live tile option. For this example, a 100% Stacked column chart is used.
3. Pin the visual to a dashboard.

Figure 14-17: Pin visual to dashboard

4. When you push the data into the push dataset using the PowerShell script provided in the earlier section, the data refreshes with a small latency on the dashboard tile.

Figure 14-18: Real-time streaming - dashboard vs report

Note – The data does not refresh on the report page. You need to refresh the report page to see the new data.

Summary

In this chapter, you have seen the different types of real-time streaming datasets available in Power BI. You have learned in detail the usage of these datasets, which will help you in choosing the right option for your requirement. You have gained the knowledge of creating datasets and pushing data into these datasets. Finally, you have seen how you can visualize different real-time streaming datasets on a Power BI dashboard.

About the Author



Manohar Punna is a Data Platform Geek and Vice President of DataPlatformGeeks.com by passion and a Data Platform Consultant by profession. He is a speaker at various events around the world like PASS Summit, SQLBits, SSGAS / DPS conferences, SQLSaturdays, SQL Server Day events, and many user groups. He has a wide variety of experience working on code to manage physical hardware, database administration and development, architecting databases for enterprises, designing and building end-to-end BI solutions and building business solutions using PowerApps and Flow. He has authored over 150 blogs and has written One DMV a Day series which is the most extended one-day series on any topic on SQL Server till date.

Part VI: Power BI Usage in Enterprise Environment

Chapter 15: Introduction to Conversation-Centric Design TM

Author: Treb Gatte

Many organizations are challenged when attempting to design business intelligence content for the first time. Where do you begin? How can you ensure your content is adopted? The Conversation-Centric Design approach ensures you design BI content that is aligned to the business need, structured where you can manage scope and ensures that it is clear to the consumer where the content should be used.

“Spreadsheet on a Web Page”

Steve plopped in his office chair and let out a sigh. Dipti, his team manager, seeing him return, walked over to his desk. “Hey Steve, how did the dashboard design meeting go?”, she asked. “It was awful, really.” Steve shook his head and explained. “All the users wanted was a spreadsheet on a web page so that they could see all the details.” “What about the ability to show data graphically in Power BI?” she asked. He replied, “They simply couldn’t see how it applied to them nor could they explain the questions they were looking to answer from the one big spreadsheet.” Dipti frowned. “We will need to try again with this group. If we don’t get them excited by the visualization capabilities, this highly visible project is going to fail and take our jobs with it.”

What's really going on here?

This scene happens over and over in many workplaces as they start their Power BI journey. We approach Business Intelligence (BI) content development as a blank canvas exercise. Unfortunately, most people are not Picasso and struggle to visualize a wonderful solution to their BI needs from a blank canvas. They need help to understand what decisions are important to make and how does that map to their BI content so that they know where to use their brand-new reports and dashboards.



Figure 15-01: Artist, Jackson Pollack at work

In the end, many design their BI content the way the artist Jackson Pollack created art. Pollack would throw paint at the canvas and a masterpiece would emerge. Unfortunately, using the same approach with data ensures that you deliver many reports and dashboards, but few answers.

Conversation-Centric Design™ Overview

The Conversation-Centric Design™ (CCD) process uses the user's social interactions to provide context and direction for the development of Business Intelligence content.

All work is inherently social within the workplace. We interact socially when:

- We create work
- We are assigned work
- We have questions about the work
- We report status on the work
- We deliver the outcomes.

These interactions are called client calls, meetings, projects, hallway conversations, etc. Many of these interactions are formal in nature. They occur regularly, have a set attendee list and a set agenda. Hence, the CCD process leverages formal interactions as a starting point for BI content design.

Why formal interactions?

Successful adoption of new BI content is key to your project's success. Projects that change the way a person works fail when it is unclear to a person where and how to use the new content in their work. Using these formal interactions makes it clear to the impacted person where to apply the new functionality.

Process Overview

The Conversation-Centric Design™ process has four distinct phases as shown in the figure below. The intent is to take you from a position of endless possibilities to a starting point with clear priorities and steps to deliver them.

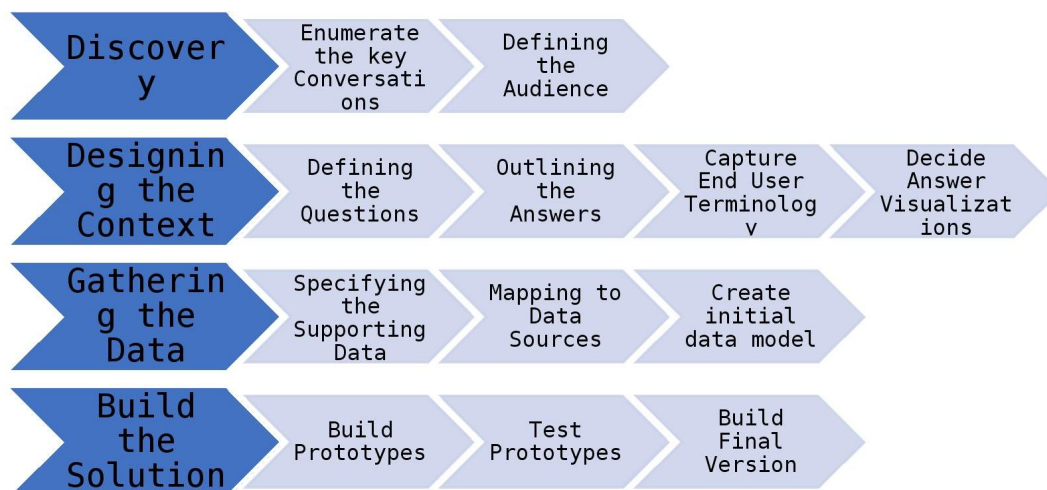


Figure 15-02: Detailed Conversation Centric Design process

Discovery

The discovery phase provides the steps necessary to focus your efforts on the key conversations. First, we start by cataloguing the key conversations. This can be standing meetings or common interactions like status reports or client meetings. The goals are to create a recognizable list of conversations and a list of audience members for each conversation.

The conversation list is used for four purposes.

Scoping

This use enables you to decide which conversations to invest in from a business intelligence perspective. One of the hardest decisions for business decision maker (BDM) to make is around scoping as many data investments are opaque to them. Providing a known event as the scoping mechanism means the BDM understands what the data investment impacts and the relative importance of the event. This context enables the BDM to make an informed scoping decision.

Prioritization

This use enables the BDM to decide which conversations to address first. This aspect enables the BDM to get the biggest impact for the investment. If they are hoping to win support in the executive suite, prioritizing the executive conversations first ensures the audience's needs are met.

Status

The business event list provides the context for the current activity and status. BDMs are usually not data professionals so providing status on data models, measures, and Extract-Transform-Load (ETL) processes without context, results in glassy eyed looks, nods of the head, and zero understanding of what's really happening. By providing updates relative to the business event, it helps the BDM in understanding where the impact of a given topic will lie. We should still take steps to make the process as clear as possible for the BDM.

Audience

Capturing the audience per conversation forms the basis for a security model and for testing groups later in the process. It also provides a list of stakeholders for reporting status.

Many business intelligence projects only consider the direct requirements of the requestor. However, once implemented more broadly, a flurry of activity results as other stakeholder needs are discovered and were not accommodated in the design. This can lead to expensive rework.

Using the STAR model below, you can consider or purposely eliminate the needs of various potential stakeholders. Doing so ensures you are correctly considering a broader audience if needed.



Figure 15-03: STAR model for potential audience discovery

Example of Discovery outcome

Conversation	Frequency	Audience	Priority
VP meeting	Monthly, usually first Wednesday of Month	Joe, Pradeep, Karthik, Amy, Mason	1
Project Status Meeting	Weekly on Monday	PM and PM Team	4
Finance Review	Monthly on first Tuesday of Month	Bill, Susan - Accounting, Joe, Pradeep, Karthik, Amy, Mason	2
3 + 9 Finance Exercise	Quarterly	Bill, Susan - Accounting, Joe, Pradeep, Karthik, Amy, Mason	3

Designing the context

Once we have our prioritized list of conversations from Discovery, we start to break down the activity into our four areas. These areas are:

- Defining the questions
- Outlining the answers
- Capture end user terminology
- Decide answer visualizations

Developing Key Questions

Conversation-Centric Design starts with a key conversation. We'll discuss how to decide what those will be shortly. For example, let's use the Monthly VP Meeting as an example.

Using a meeting like this, we gain a lot of useful information quickly. First, we know what the conversation generally is about as there is an agenda for this meeting. We know the audience that attends the meeting. Looking at the figure below, we can see that two items are already captured.

When you drill into the meeting agenda, rephrase the items as a series of key

questions. For example, your list could look like this:

Original agenda item	Question version
Budget Update	Where are we with regards to our overall budget? What are our top five deviations from budget?
Key Projects Update	What projects are running late? What projects are finishing this month? What projects are starting this month?
Personnel Update	What roles are short-staffed over the next six months? What roles are being filled primarily via vendors?

The next step is to group, prioritize, and triage the questions as it may not be possible to address all questions due to resource constraints or other factors. Optimally, this should be done by the attendees. They should rank the questions in importance and the input should be used to stack rank the questions.

Developing the Answers

Developing the answers for each question requires deciding several aspects to ensure you have what you need. The goal is to uncover mismatches between requirements and process.

For example, one VP has determined that the Budget reports need to be refreshed daily. However, the process that delivers the budget data only runs monthly. Therefore, do you as the creator want to face a VP who has been looking at the reports daily expecting changes or do you want to make them aware of this issue upfront?

Data

In examining the questions, a key need is identifying the source and data elements needed to address the question. For example, the question “Where are we with regards to our overall budget?” implies that the budget numbers are in a system that can be accessed and can be matched with expenditures in some fashion.

Definition

When formulating the answers, it is very important to clearly define the conditions that influence or are used in filters, KPIs, measures, conditional columns, and other constructs. As you will see later in the chapter, this may be the most important aspect in ensuring expectations are met and all data consumers have the same shared understanding of the data represented.

Granularity

Once we determine that the data exists and can be accessed, the next question is one of granularity. How granular does the answer need to ultimately be? This will create the basis for groupings, pivots, and what's the lowest level of data available to view.

Process

We also need to understand the process by which the data is created and delivered. In some cases, direct access to the data source enables faster access. However, high business impact data such as financial information may only be provided on a recurring basis, in a summary format like a CSV file. The timing and process by which the data is delivered can significantly impact delivered functionality. It also creates a data requirement to include the last modified date of the data to the data consumer so that they are aware of the data's age.

Terminology

Power BI has a powerful ad hoc querying tool called Q&A. Q&A depends on a detailed linguistic schema to map data consumer terminology to data elements in the Power BI Model so that it knows what to data to query when asked a question. As you are developing and designing questions and answers, it is important to capture the language used by the data consumer when referring to a data element. This can be incorporated into the model's linguistic schema, making Q&A much more useful. For example, if I have a column called ProjectName in the data source but my consumers call them Ganttts, schedules, and plans, I'd incorporate those three terms in the linguistic schema so that Q&A would understand the consumer request.

Tool

Microsoft provides three core tools for Business Intelligence: Power BI, Power BI Paginated Reports, and Excel. It is imperative to pick the right tool for the need to prevent brittle, workaround solutions. For example, if there's a need to print a report to provide to field personnel, it may be a better fit for Power BI Paginated Reports rather than attempting to engineer a solution in Power BI

itself.

Visualization

Once you have the question and the answer with permutations decided, you can classify your answer to determine the best set of visualizations for the need. By permutations, the answer may require different groups, levels of rollup, etc.

There are seven types of visualizations.

- Nominal comparison
 - Comparison of quantitative values grouped by some categorization value.
 - Example: Headcount by Department
- Time-Series
 - Changes in one or more metrics over a defined time period
 - Example: Year to date expenditures
- Ranking
 - Comparison of two or more values to illustrate relative contribution in a most to least or least to most pattern
 - Example: Top 5 projects by total budget
- Part to Whole
 - Comparison of a data subset to the whole data set
 - Example: Percentage of riders of a specific bus line compared to overall riders
- Deviation
 - Comparison of data points against expected data values
 - Example: Monthly actual sales versus projected sales
- Distribution
 - Visualization of data distribution, either around a mean, timeframe or some other nominal value
 - Example: Product demand by age group
- Correlation
 - Changes between two or more metrics that change in relation to each other in a positive or negative way
 - Example: Job performance ratings versus social likeability scores

Visualization types are as follows:

Type of Visual	Example	Use Lines to	Use Bars to	Use Points to	Suggested Power BI Visual

Nominal Comparison	Number of hours logged this week toward a project versus planned hours for the week	Show deltas between values and overall trend	Compare between individual values	To show individual values. Add a line to highlight the trend	Column Chart Line Chart Table Matrix
Time-Series	Project Costs by Month	Show trends in the data	Compare between individual values	To show individual values. Add a line to highlight the trend	Column Chart Line Chart
Ranking	Top 5 Projects by ROI		Show ranking, which is the most common use of bars	To show ranking using a nonzero based scale	Funnel Bar
Part to Whole	Annual Project Investment by Strategic Objective		In place of Pie Charts to improve usability		Waterfall Tree Map Stacked Bar Donut Area
Deviation		Show overall trend	Highlight magnitude of deviation	Denote values and use with a line to show trend	Column Chart Gauge Chart Scatter Plot
Distribution		Visualize if the data shape is most important to communicate	Visualize if the individual values are the most important to communicate		Column Chart Area Chart
Correlation		Use with scatter plot to denote trend		Use to denote values	Scatter Plot Bubble Plot

Gathering the data

This phase involves the creation of the initial data model in Power BI or could be used to modify an existing data model. The goal is to ensure

- The list of data sources is known so that security access can be requested

- The requisite data is available from the identified data sources
- Create the initial data model

Specifying the supporting data

A review of the answers outlined above provide an opportunity to determine the source of the answer's data.

If your group owns the data, then follow your normal process to connect to the data.

If the data is coming from an external system source that is managed by another entity, access to the data source should be requested. This should also include the creation of data access accounts with requisite licensing. It is a good idea to be engaged with the data source's change management process so that as they upgrade and maintain their systems, your process is not broken by inadvertent changes.

If the data is being sourced from a manual creation process, there is a need to standardize the location and name of the requisite data files. A change management process for the data structures should also be considered for ongoing management. Doing so ensures that inadvertent changes don't break the data refresh going forward.

Mapping to data sources

Once all data sources have been identified, three aspects must be rationalized to ensure use of the data will lead to actionable data.

In many cases, you will need to combine data from the various sources to create a model. Doing so requires a set of common key values between tables. For example, if you are tracking a rolling total of expenditures and need to compare them to the budget, you'll need a key such as budget code on both the budget and expenditure records to tie them together.

In some cases, a multi-part key, involving multiple column values, must be used to denote a unique record. To ensure Power BI can work with this effectively, you'll need to synthesize a compound key, where all the values of the multiple columns are concatenated into a single column value.

Differences in data granularity is another difficulty encountered. Pulling data from various sources are usually at different levels and types of granularity. This creates issues rationalizing the data. For example, your budget information is only available by cost center and your expenditures are at the transaction level. How would you tie these together to create a project level summary?

Mismatches in data refresh cycles can also lead to integration issues. Some data may be real-time, others refreshed once a month. As you bring the data together, a time scale minimum granularity should be defined.

Create initial data model

A data model is ideally structured in a star schema pattern. By this, a model generally has facts and dimensions. Facts are the events that have occurred. Examples of this can be transactions, timesheet entries, status updates, etc. Dimensions are those entities to which the fact applies. For example, for a transaction, dimensions could include billing company, cost center, project, etc.

Once the core data is in place and structured correctly, the initial measures, summary tables and other transformations should be put in place. In many cases, a company may have standard elements that are created for a dimension and/or a fact table. These could include a date table, for example. This creates the basis for the next phase, where iterative development is used to finalize the model and requisite reports.

Build the solution

The building of the solution starts once you have a solid design. This should reduce the time to develop the solution and reduce any rework. Note, the following describes an iterative approach to developing business intelligence content. Many data consumers do not have the training or experience to formulate the final visualizations. The iterative process gives them a voice in the process as well as builds champions for the final implemented product.

Note, the process below is designed for large business intelligence projects with high visibility in the organization. The process can be streamlined for smaller projects as needed. The key learning is to drive the test process rather than simply reacting to it. Simply reacting leads to scope creep, unmet expectations and generally longer timelines to deliver.

Build Prototypes

The initial report prototypes validate the design process findings and are the first opportunity for the data consumers to be hands on. A focus group of data consumers should be identified from the earlier audience identification and recruited to help with this validation.

In most cases, the data author should be listening only, with the session facilitated by another with no vested interest in the feedback. Any feedback that is gathered, should be reviewed, triaged as to whether it is of value to address,

and then work assigned to incorporate feedback.

To prevent runaway scope, there should be a set number of review rounds initially and a data consumer representative should be part of the feedback triage process.

Test Prototypes

Once you've reached a specific level of quality with the model and reports, an extended pilot group should be given access to it to test the new content.

It is imperative that you provide clear guidelines to the new pilot group on how to file feedback, what is considered a bug, what is considered a change, and what will be done if it's simply product behaviour. There should also be a task plan with clearly defined assignments. Otherwise, you'll have a new group of people wandering around in the new content with no context and focusing on product behaviour. This will yield feedback of little value.

Typically, a few rounds are necessary to get everyone's feedback and to have the feedback triaged and prioritized.

Build Final Version

Once the test phase is over, the final build out of the content occurs. The work plan of remaining items should be tracked. The content should be packaged as an app and released according to internal procedures to the target audience defined earlier.

A Real World Example

Steve, a business analyst in the IT group, received the following request from Donna, an IT Director.

“Steve, we need a dashboard for our IT Managers and Finance personnel to use in the Monthly Review meeting. This will be used to see project related data. How soon do you think you can get that done?”

Steve sighed and thought, “It’s time to put this CCD training to work.”

First step is to enumerate the conversation. He highlighted the phrase as follows, as this gave him the conversation and the timing.

*Steve, we need a dashboard for our IT Managers and Finance personnel to use in the **Monthly Review meeting**. This will be used to see project related data. How soon do you think you can get that done?”*

Next step was to determine the audiences. Steve highlighted the following.

*Steve, we need a dashboard for our **IT Managers and Finance** personnel to use in the Monthly Review meeting. This will be used to see project related data. How soon do you think you can get that done?”*

Steve frowned, “What I don’t see is any actual questions in this request.” He replied to Donna with some questions about what questions needed to be answered for the IT Managers and for Finance. This dialog continued for a bit. As he learned in his training, he needed short questions that started with Who, What, When, Where, Which or How much. At the end of the exchange, he had the following questions.

Audience	Monthly Review Conversation
IT Management	Which projects are behind? Which projects are finishing this month? What are the upcoming key milestones this month?
Finance for IT	What projects are over budget? What are the anticipated capital expenses for this month?

Steve decided to focus on the first question to train Donna in what he needed, as this was her first time asking for content. Once she understood what he needed,

the process would go quickly.

Since this was an extension to an existing model, he knew the data would be coming from Project Online. However, he still needed to identify the data fields once this request was clearly defined. He looked at Project and saw there was many projects. Also, what did they mean by behind?

He replied with the following questions.

With regards to the What projects are behind question,

- *What kind of projects should be included in the report? Capital? All?*
- *Define the criteria for “behind.”*
 - *Is it Planned > Baseline?*
 - *Is it some percentage variance?*
 - *Is it schedule or cost?*

After several iterations, they finally arrived at the following criteria.

- Show all capital projects over \$250,000 in total spend where the planned finish date is greater than 30 days past their baseline finish date.

Steve gave some thought as to how this might be visualized. He decided that three levels of data were appropriate. This would provide several pivots without being overwhelming. This would also set up the report to enable drill through to other detailed reports.

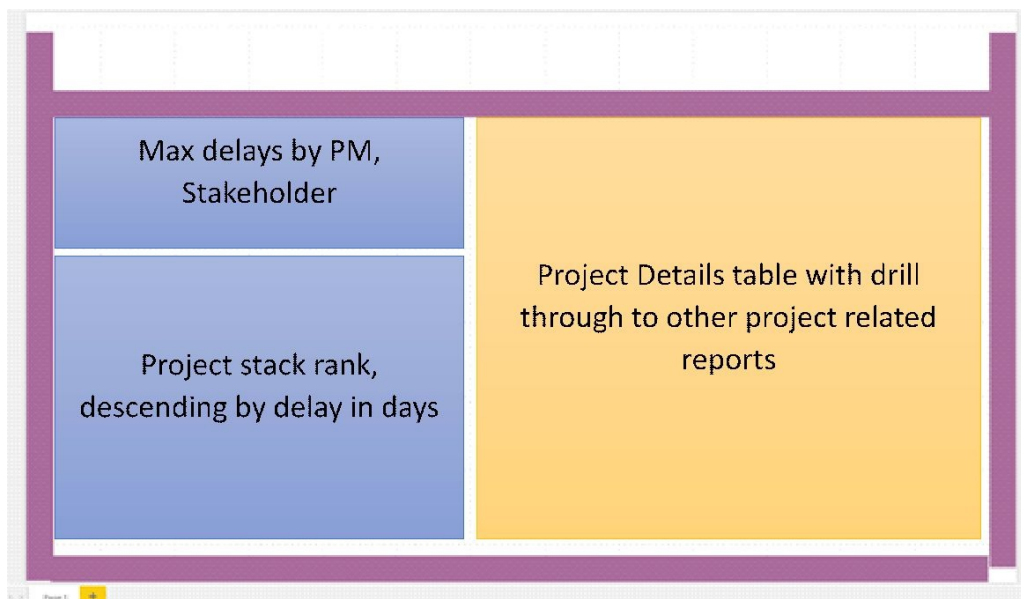


Figure 15-04: Page layout for new project report

He decides to use cards for the max delay information since it's an overall number. He'll use the Stacked bar chart visual to do the project stack rank by delay visual. Finally, he's using the table visual to show key information for each individual project.

His first prototype looks like the following. The entire page is structured around a single question and the number of visuals is kept to a minimum. He repeats this process for each of the questions.

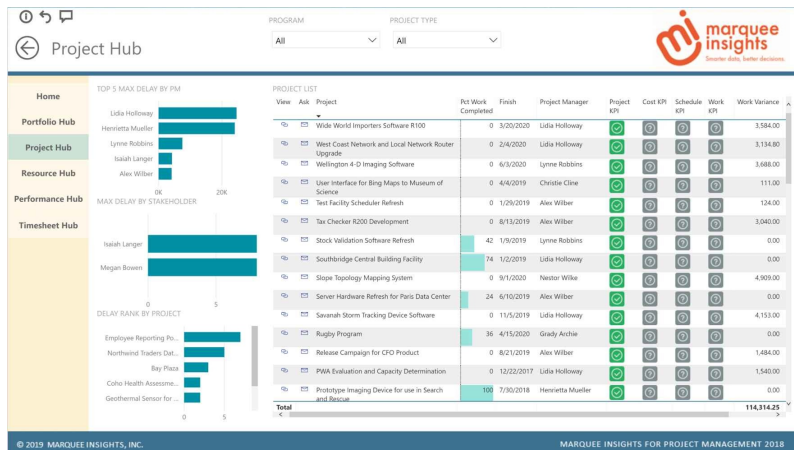


Figure 15-05: Screenshot of prototype

Steve reviews the prototype with Donna. She sees how the reports are question focused and understands how these can be easily used and explained. She asks for specific drill throughs to other reports which Steve can do. She also appreciates the ability to add new pages as new questions arise, without having to re-architect every report.

Once Steve is done with the changes, Donna signs off on it. He publishes the final version to a new workspace in PowerBI.com. This is used to create the **IT Monthly Meeting** app, which Steve publishes and assigns security rights to the IT Managers and Finance.

Summary

The Conversation-Centric Design™ process streamlines the development of business intelligence data by providing a framework that helps you anticipate and address issues before they arise in the development process. Once a solid base of content is in place, it also makes it easier to maintain that content over time as the requisite documentation is created as the content is developed.

Real world use also shows it increases adoption of content since the delivered apps are aligned with the business activity that drove the need for the content. It also reduces the risk of changing reports since the exposure of a given report collection is limited to a single business activity.

For questions about Conversation-Centric Design™, contact us at hello@marqueeinsights.com.

About the Author



Trebuel Gatte, CEO, MarqueeInsights.com

Treb Gatte is a business intelligence expert with 24 years of experience.

Prior to becoming CEO, he worked in leadership positions at Microsoft, Starbucks, Wachovia (now Wells Fargo). He has been recognized as Data Platform MVP from Microsoft.

He lives in the Seattle area with his wife, children and corgi and holds an MBA from Wake Forest University.

This chapter is an approved excerpt from the book, Introduction to Conversation-Centric Design™, ISBN: 978-1-7333418-0-6, © 2019 Trebuel Gatte, Marquee Insights. All rights reserved.

Chapter 16: Understanding when to move to Power BI Premium

Author: Gilbert Quevauvilliers

When looking at Power BI Premium there are a lot of options available to you with Power BI Premium. At times when there are too many options and it can potentially be a challenge to understand which features are applicable for your situation. In this chapter you will gain a better understanding of what these options are. By having a deeper understanding of the Power BI Premium features, it will allow you to make a more informed decision on looking to move to Power BI Premium.

Whilst the list below is not comprehensive it does overview the standard set of features that are currently available in Power BI Premium as at July 2019 (Please refer to <https://docs.microsoft.com/en-us/power-bi/service-premium-what-is>)

Dedicated Performance

When looking to move to Power BI Premium, one of the advantages is having dedicated performance. What currently happens when you are using the Power BI Service you are using a shared environment.

If I had to put this into an example, it would be you are sitting in an aeroplane as you to today. There are a lot of people in the aeroplane, but you are sharing not only the costs of having to fly the aeroplane, but you too are sharing all the components (engines, fuel, entertainment, pilots, air hosts, etc.) that make up the aeroplane.

Whilst when you move to Power BI Premium you are now flying on your private jet. Here you do not share anything with anyone else; the entire aeroplane is available for you to use as you so wish.

The same applies to Power BI Premium, where you now have dedicated CPU and Memory.

At the time of writing this book (July 2019), there are the following dedicated capacities for Power BI Premium, which indicates how much memory, CPU, and Storage per capacity that you purchase.

The details below have been taken from the Power BI Whitepaper Deploying and Managing Power BI Premium Capacities (<https://docs.microsoft.com/en-us/power-bi/whitepaper-powerbi-premium-deployment>)

Capacity Nodes	Total v-cores	Backend v-cores	RAM (GB)	Frontend v-cores	DQ/LC (per sec)	Model Refresh Parallelism
EM1/A1	1	0.5	2.5	0.5	3.75	1
EM2/A2	2	1	5	1	7.5	2
EM3/A3	4	2	10	2	15	3
P1/A4	8	4	25	4	30	6
P2/A5	16	8	50	8	60	12
P3/A6	32	16	100	16	120	24

Figure 16-01: Table showing different Power BI Capacities

A quick overview of what each of the Capacities is:

- EM – This is for Embedding Power BI into an application (An

application is your custom application that is not part of the Power BI Service).

- A – This is also for embedding Power BI into an application (An application is your custom application that is not part of the Power BI Service). The difference here is that it is run from Azure. The advantage of running it from Azure is that you get the Azure functionality to start and pause the embedded capacity as is required. You can also scale up and scale down the capacity as required.
- P – This is the Premium capacity where it can be used within the Power BI Service and assigned to specific app workspaces, my workspace (users' workspaces) or for the entire organization. One additional thing to note is that the P Capacity can also be used for embedding into an application.

The current way to purchase the Power BI EM or P Capacities is through the Office 365 portal. This is because Power BI currently falls under the Office 365 stable of products.

If you are looking to use the A Capacity that can be created through the Azure Portal.

Here are some of the reasons you might be looking to move to Power BI Premium for the dedicated performance.

Free Users

What happens is that very often, there is a dataset or a report that can assist a large number of users in your organization. You do not want to license each user individually to simply view and interact with a report.

By using Power BI Premium, you can use the feature of free users who can view the content in an App Workspace that has been assigned to Power BI Premium (This only applies on the P SKU)

What could also be considered is if there is a pricing breakpoint between the number of users who need to view dashboards and reports when compared to purchasing Power BI Pro licenses. It is a good exercise to investigate as there might be some great cost savings.

XMLA End Points

XMLA endpoints is fancy terminology for meaning that you can connect to your Power BI Premium dataset with other tools such as Excel.

The way I would explain XMLA endpoints from the standpoint of reporting is

that any client tool that can currently connect to SQL Server Analysis Services (SSAS) Multidimensional will also be able to connect to Power BI Premium capacities.

As it stands today July 2019, the XMLA endpoints are read-only. In the future are plans for the XMLA endpoints will change to read-write, which will allow for a lot greater management of your Power BI Premium capacities.

If you are looking to leverage other tools which have connectivity to SSAS cubes such as Excel, Tableau, or others, then this is an option to consider.

Higher Refreshes

Currently, when using Power BI Pro, you are limited to 8 refreshes a day. When you have an App Workspace assigned to Power BI Premium you can then have up to 48 scheduled refreshes per day.

This means that during the core business hours (being 8 hours a day), you could refresh the dataset every 15 minutes.

One thing to note is that you would need to ensure that your dataset will have completed refreshing within the 15 minutes time frame, for the next refresh to start.

With higher refreshes, you can refresh the data as often as you like when using the Refresh API.

There are some other considerations to think about when refreshing data, which will be covered next when using Incremental Refreshing.

If you have data that is frequently updating this enables you to do this with Power BI Premium.

Incremental Refresh

Before I get into incremental refreshing, I think it will be good to over off what incremental refreshing is.

The best way to describe it is by using another example.

When you import sales data into Power BI, it all gets imported into one big table as shown below.



Figure 16-02: Complete Sales Data table

If we had to take a closer look at your sales data, it will typically always have a date for when the sales took place.

If we had to look at the same sales data table in another view, it looks like this too.

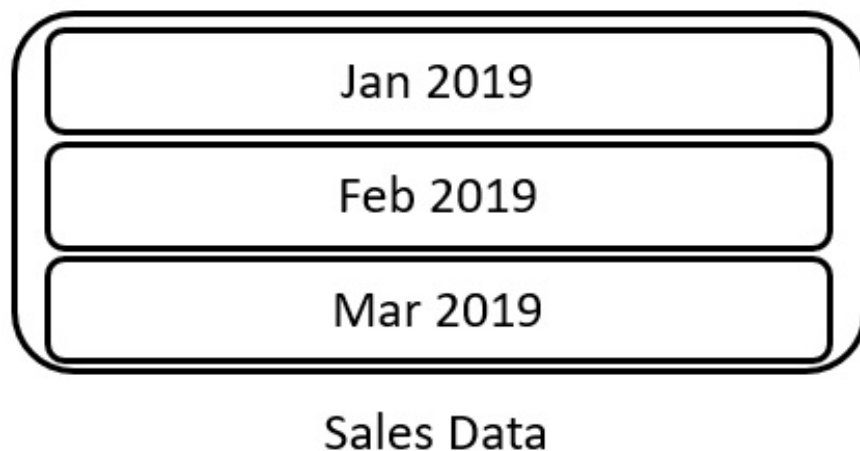


Figure 16-03: Sales Data table with months (partitions)

What you see above is that even though the sales data is still in one table, there are three segments of data. One for each month where there is data, being Jan – Mar 2019.

What incremental refreshing is, it will NOT refresh all the data, but rather

refresh the latest data? Which means that it will incrementally refresh the new or updated data. And not refresh the entire sales data table.

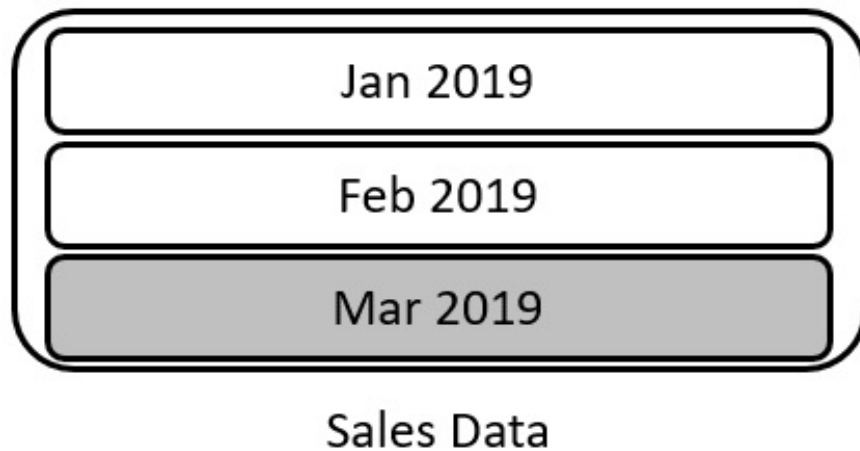


Figure 16-04: Sales Data table highlighting the month of Mar 2019

Using the above example if our current month we were in was Mar 2019, when configuring the incremental refreshing, it would only refresh the Mar 2019 portion of the sales data table.

From the above image, each of the months is known as partitions when using incremental refresh.

There are some distinct advantages of using incremental refresh

Refreshes are quicker

Because you now do not have to refresh the entire dataset, but only a subset of the data it will refresh a lot faster.

This is because if you had a table with three years' worth of data, it would have to refresh 360 000 rows (The number is derived by assuming that each month has got 10 000 rows).

Whilst if you only had to refresh the current months' worth of data, that would mean only refreshing 10 000 rows (if it was a complete month, or even fewer rows if it was during the month).

If we had to put this into a comparative chart, you could see how much less data there is to refresh.

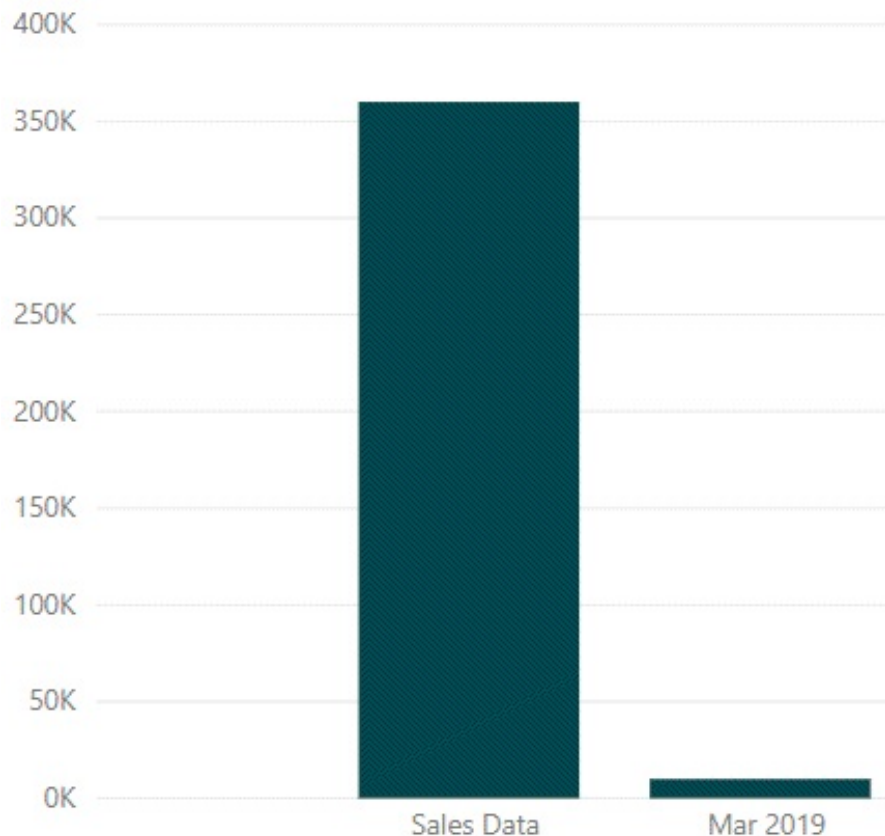


Figure 16-05: Comparison for refreshing the entire Sales data table vs. refreshing the Mar 2019 partition

Fewer resources are required

One of the things to always be aware of, as is typically the case when using dedicated resources is that you only have a certain amount of resources that you can utilize.

When using incremental refreshing, it will use a lot fewer resources if you had to compare it to refreshing the entire sales dataset.

If we have a look at the chart above and assume that it represents resources, you can very quickly see that March 2019 will use a lot less resources.

One thing to note is that as with all things in Power BI, it will depend on a lot of factors in how much resourcing saving you will get, but without a doubt, you will use a lot less resources when using incremental refreshing.

The image below you can see how the incremental refresh uses a lot less memory compared to a full dataset refresh

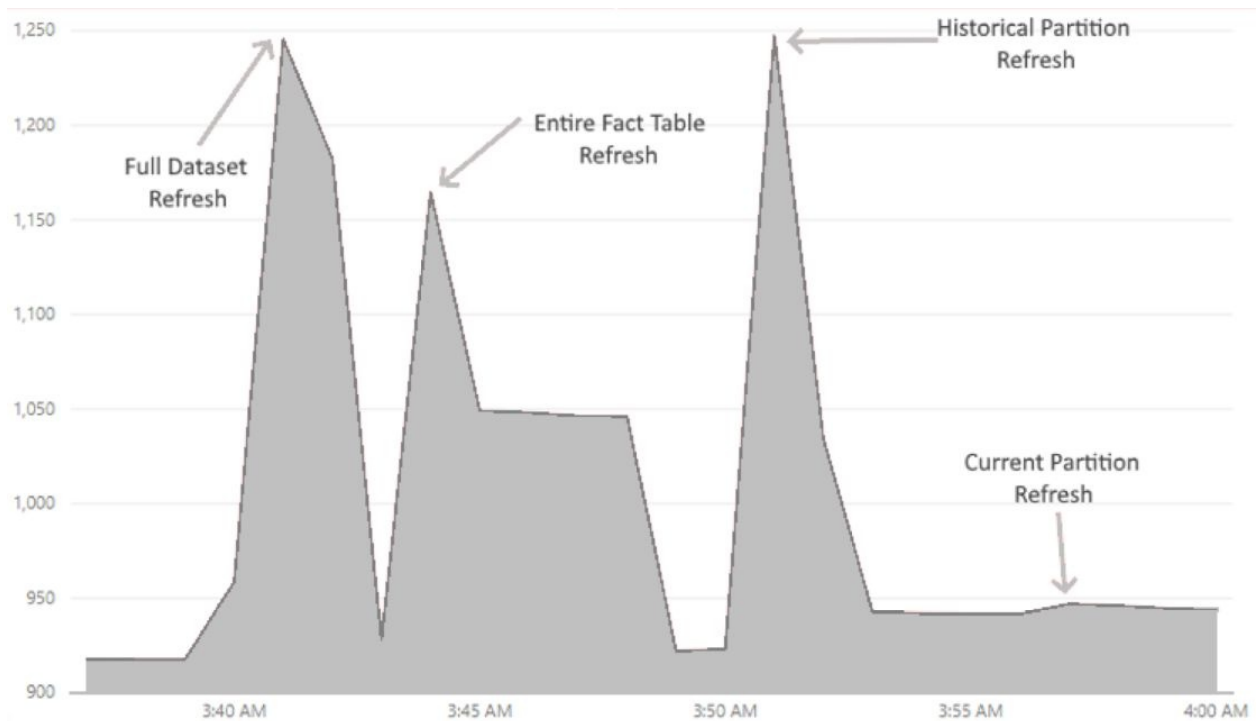


Figure 16-06: Chart showing how different dataset refreshes affect memory consumption

If you are using larger datasets and looking to get the data refreshed as quickly as possible incremental refreshing could be a solution.

Another reason to decide to use Incremental refreshing is if you want the refresh to consume fewer resources, mainly memory and CPU, this could be a viable option to allow that to happen in a very efficient manner.

Dataset size larger than 1GB

When looking to use Power BI Premium another reason to use it, is that it allows you to use datasets that are larger than the 1GB file size limit when using Power BI.

Whilst the Vertipaq engine (this is the compression engine that runs Power BI) is very good at compressing data a lot of organizations have a lot of data, and it simply will not fit into the 1GB PBIX file size, which is a limitation in the Power BI Service not on the actual PBIX. This is where Power BI Premium could be a solution for you.

As shown previously and shown again below, there are different capacities that you can buy based on how large your dataset is.

Note, the image below only shows the Power BI Premium Capacities for the P capacities. This is because these are the only ones that can be used within the Power BI Service.

Capacity Nodes	Total v-cores	Backend v-cores	RAM (GB)	Frontend v-cores	DQ/LC (per sec)	Model Refresh Parallelism
P1	8	4	25	4	30	6
P2	16	8	50	8	60	12
P3	32	16	100	16	120	24

Figure 16-07: Power BI Premium P Capacities

Another thing which you must take note of is when moving to Power BI Premium it is not the size of the PBIX that determines how many resources you consume, but rather how much memory the PBIX consumes once it is loaded and people are using the reports or dashboards used in queries when interacting with the data.

To determine how much memory your PBIX is going to consume, there are two ways to determine this.

As a working example, the size of my PBIX is shown below


 WWI - Power BI - Completed.pbix 57,336 KB

Figure 16-08: File size of the PBIX when stored on disk

Actual memory consumption

A better way to see how much memory the PBIX will consume is to find the

msmdsrv.exe

You can complete the following steps below.

- Make sure you only have got **one Power BI Desktop File Open**.
- Now right-click on the taskbar and select Task Manager

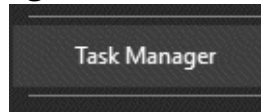


Figure 16-09: Task Manager selection

- Next click on the Details tab.
- The easiest way to find the msmdsrv.exe is I click on the Name to sort it alphabetically

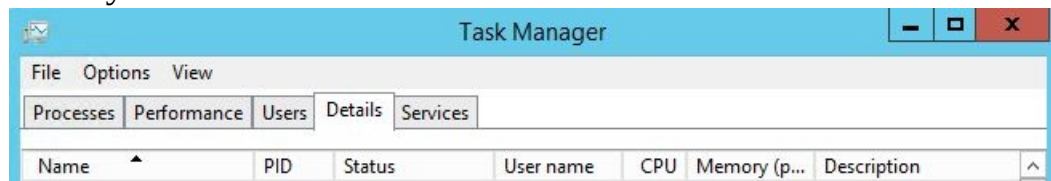


Figure 16-10: Task Manager Window sorting by Name

- I then click on any item under Name
- I then type “ms” which allows me to then go down to a name that starts with “ms”
- I can then **find** my **msmdsrv.exe**
- As you can see below my current PBIX is using about 318MB of memory
 - msmdsrv.exe 79704 Running Gilbert 00 319,720 K Microsoft SQL Server An...
 - **Figure 16-11: msmdsrv.exe memory usage when using Task Manager Window**
- NOTE: Even though it is currently using 318MB of memory I am currently not interacting with the report, which would very likely increase the amount of memory consumed. Not by a significant amount, but just something to be aware of.

A quick tip is if you want to understand what is consuming all the memory in your PBIX, you can use the Vertipaq Analyzer.

This is a tool that has been created by the great people from SQLBI.COM

Here is the link to where you can find their blog post:

<https://www.sqlbi.com/articles/data-model-size-with-vertipaq-analyzer/>

Part of this blog post also has a link to the Excel file that you can download and allow you to see which columns are consuming a lot of memory. Another use

case is you could use the XMLA endpoints to see how much space is being used. There are times when either certain columns can be removed (because they are not needed) or they can be changed, which can drastically affect the amount of memory consumed.

The Microsoft Power BI team have indicated that they will be removing the limit on dataset sizes which is currently set as at July 2019 to 1GB for Power BI Pro Users and 10GB for Power BI Premium.

There is also an upcoming feature where they are going to allow you to potentially store dataset sizes of up to 4TB. More on those details will be available on the Power BI Roadmap.

When looking into larger dataset sizes as a reason to move to Power BI Premium it is often a use case that not only do you need the larger dataset size, but it almost goes hand in hand with incremental refreshing.

Paginated Reports

And currently, a lot of organizations have invested a significant amount of time in the creation and development of SQL Server Reporting Services (SSRS).

These were originally the first type of reports that allowed the business users to run their reports, change the parameters, and self-serv.

The SSRS reports were also very good at creating pixel-perfect reports, which allowed for the printing of the reports for a lot of business users.

Not only that, but the business users could also export the data in the reports into multiple formats.

And finally, the SSRS reports could be set up with a schedule to allow the SSRS reports to be emailed to you. This meant that the business users could get the SSRS reports either in the Inbox or from a folder location.

While this is great for On-Premise solutions, there is also the requirement to have all the above functionality within the Power BI Service.

It was called Paginated Reports when moving to the Power BI Service; this was first announced on 11 July 2018 at the Business Applications Summit.

Paginated then went Generally available on 09 June 2019.

As with all things being migrated from an On-Premise solution to the cloud, they do take time to get there. And as has been the approach with Power BI is that they release new products or features with limited functionality and then build new features as defined by the customers, idea's and their roadmap.

As it stands today, paginated reports do not have full feature parity with SSRS. The paginated reports team has confirmed that their goal is to have all the features in SSRS available in paginated reports.

When looking to move SSRS reports to paginated reports, you would have to first consider if paginated reports have the features that you are looking for. After which you would then need to plan to move your SSRS reports to paginated reports.

One of the major advantages of having paginated reports in the Power BI Service is that you now have one central location for all the reporting requirements in your organization. You could also look at using Power BI Report Server which is the On-Premise Version of Power BI and SQL Server Reporting Services (Paginated reports)

This simplifies where and how users can get the data that they need.

Paginated reports also use existing features and shared and certified datasets, which can be consumed by paginated reports. By having a shared dataset, it means that both Power BI reports and paginated reports will be able to use a single data source for multiple reports. This goes a long way to ensure that when multiple people are looking at data or making decisions, no matter what format they are using the numbers remain the same.

Geographic Distribution

Another interesting feature that is only available to Power BI Premium is the geographic distribution of your data.

What geographic distribution means is that you can physically host your data in a Power BI Premium App workspace in another geographic location.

Typically, there are two reasons why you would be looking to using the geographic distribution feature.

Data Sovereignty

Data sovereignty is another word for ensuring that your data is hosted in the country or location which has been defined by a company or country regulations.

This often happens when certain countries have defined that all data must reside within their borders.

If you are a global organization and have users around the globe and have transactions from multiple countries where the data must reside in the country where the data was transacted, this is another great option. This is where having the geographic distribution feature is valuable.

Performance

Along the same lines as the example above with having a global organization, there could be users who want to access their reports with the best performance possible.

The business users could be on the other side of the world, so when one part of the business has finished their working day, the other part of the business is just starting their day. Or it could be where one user is in Europe and another user is in Southern Africa.

By having the capability to be able to have the data hosted in the same country or close by a location means that interacting with the reports is a lot faster. This is because potentially the data does not have to travel half way around the world to be executed, and then travel half way around the world getting back to see the results.

As shown below for data to travel from the USA (New York) to the UK (London) is roughly 5,500km and would take an estimated 200 milliseconds (ms) to travel there.

Whilst if the data had to travel within the UK, it would travel possibly less than 100km and would take less than an estimated 40ms

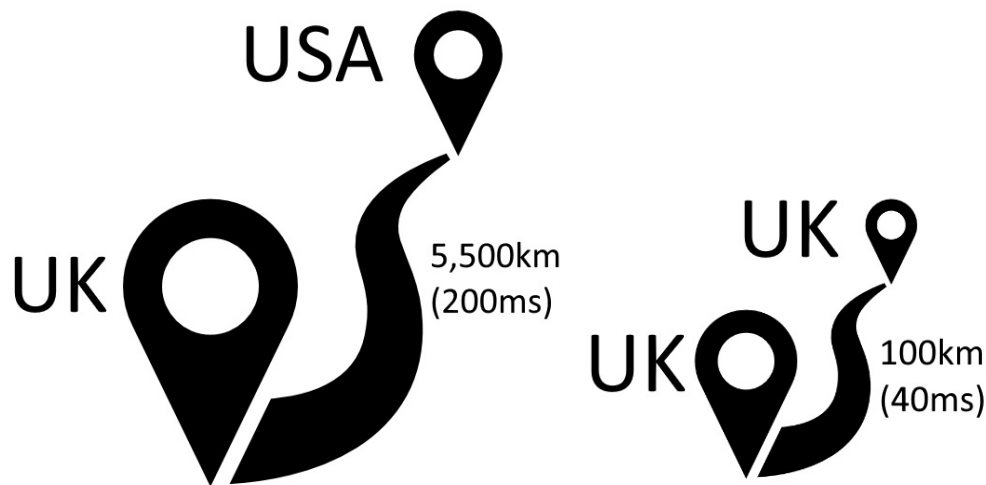


Figure 16-11: Distance between 2 offices.

Another advantage of looking to use geographic distribution is that if there is any maintenance or updates required to reports, they could be done in an App Workspace whilst those users are not at work. This allows for additional testing and ensuring that the reports and dashboards are updated and ready for those users when they get into the office the following day.

Dataflows

Dataflows in Power BI is a way to unify data from different sources using Power Query Online. All the data is stored within Power BI (Azure Data Lake Gen2) and allows other users to consume dataflows within their reporting.

Below are some considerations where using dataflows in Power BI premium might help assist you in moving some capabilities to Power BI Premium.

Linked Entities

It is best to explain what linked entries are in dataflows.

Let me take a step back and explain when you import data using dataflows, the data is taken from the source and then put into a dataflow.

Now each time you refresh data in a dataflow, it then goes back to the source and refreshes the data from the source.

Now when the linked entity is refreshed, it does not have to go back to the source data, but rather it gets the data from the existing dataflow. This allows for a much faster refresh of the data because it simply gets the data from the dataflow that is already stored in the Power BI Service.

One of the key features of a linked entity is that as a business user, you do not have to understand how the linked entities pieced (how they relate to each other) together, and in what order they need to be refreshed.

All of this is handled by Power BI Premium, and if there is an underlying dataflow that must be refreshed for your linked entity to be refreshed, Power BI Premium is aware of this and will do it for you automatically.

Computed Entities

Another feature that can assist an organization is computed entities.

Where computed entities it is different is that you created a linked entity off an existing dataflow. It also allows for you to create in-storage computations. In-Storage computation means that it does not have to go back to the source data to complete the computations, but rather, it is done within the existing dataflow.

It could also be described as when using the Reference, Duplication, Append or Merge functions in Power Query.

These computed entities can then be linked to other entities that you have already created.

Computed entities also mean that when the data is refreshed, it is then taken

from data that already exists in the dataflow. This speeds up the dataflow refresh performance significantly.

Incremental refresh

As explained previously on how incremental refresh works and what it does.

There is also the capability to use incremental refreshing in dataflows. This has the same advantages in that it does not have to refresh the entire dataflow.

As well as also allow for consuming fewer resources and a faster refresh of the data.

Parallel Execution of transformations

Another reason to potentially investigate using dataflows in Power BI Premium is that it allows you to have your transformations run in parallel.

This means if you have multiple dataflows refreshes happening at once, more than one can run at the same time. Which once again increases the performance of the dataflow refresh.

Enhanced Compute Engine

One of the new features that are coming to Power BI Premium is an enhanced compute engine.

This allows you to be able to ingest data up to 20x quicker than is currently possible. This is done by using a new compute layer which can ingest the data in a very efficient manner. It could be described as almost SQL like when ingesting the data. This makes it extremely efficient when ingesting data that requires joins or merges.

Ideally, you would import your data using the new compute engine and then use the computed entities to complete the in-storage computations as explained in the section above on how to leverage computed entities.

Monitoring for Power BI Premium

Whilst this does not cover when to move to Power BI Premium specifically, ensuring that you monitor your Power BI Premium capacity will enable that you are using all the resources available to you.

Along with this, if you start running into performance issues, having the ability to use the Power BI Premium monitoring will allow you to quickly and easily troubleshoot where there is a performance issue.

There are the following workloads that are currently available in Power BI Premium (as at July 2019)

AI (PREVIEW) - Active

Your workload is ready to use.

☒ On

Max Memory (%)

40

Allow usage from Power BI Desktop

☒ On

Allow building machine learning models

☒ On

Enable Parallelism for AI Requests

☒ On

DATASETS - Active

Your workload is ready to use.

☐ On

DATASETS - Active

Your workload is ready to use.

☐ On

XMLA Endpoint

1

DATAFLOWS - Active

Your workload is ready to use.

☒ On

Max Memory (%)

20

Enhanced Dataflows Compute Engine (Preview)

☒ On

Container Size (Mb)

700

PAGINATED REPORTS (PREVIEW) - Active

Your workload is ready to use.

☒ On

Max Memory (%)

20

Figure 16-13: Power BI Premium Capacity Settings

There is also an insightful and detailed Power BI Premium performance

whitepaper that I would recommend any user who is administering Power BI Premium read.

The link to Power BI Premium can be found here: <https://docs.microsoft.com/en-us/power-bi/whitepaper-powerbi-premium-deployment>

Summary

There are multiple considerations that you must investigate when moving to Power BI Premium.

In this chapter, I have gone through the different Power BI capabilities and by gaining a better understanding of how each of the capabilities works.

AI is becoming essential in a lot of organizations, and when looking to use Power BI Premium, there are more AI features being released, which allows for easier integration with your existing data assets, which allows organizations to quickly use AI for better data-driven decisions.

With this knowledge of the capabilities, you can then leverage the Power BI Premium capabilities to enhance further or solve future or current reporting requirements in Power BI Premium.

About the Author



I am a Power BI & Data Analytics Consultant, having over 12 years' experience working in Business Intelligence or Data Analytic solutions on the Microsoft Platform.

I have successfully implemented solutions for small to large enterprise customers.

Recently I have focused on Power BI, with its rapidly changing features and incredible uptake it has allowed me to consult in a variety of different and challenging solutions.

I was awarded the Microsoft MVP award for Power BI since 2017.

I have spoken at the Microsoft Business Applications Summit (Seattle & Atlanta), Power BI World Tour & SQL Saturdays.

Proven competencies in the implementation of data analytic solutions from the ground up. Which included developing data warehouses, SSAS Cubes, and most recently Power BI solutions for customers in various business sectors.

I have worked within teams, managed teams as well as worked alone on various successful data analytics projects.

Chapter 17: Incremental refresh

Author: Michael Johnson

Abstract: When it comes to managing loading data, Power BI does not provide many options as to how data is loaded into the model, and this causes issues in larger models where full refreshed as time-consuming. Incremental refresh offers a simplified approach to loading only the most recent changes to the data making data loads faster and more reliable while consuming fewer resources. In this chapter, we cover how to set up and maintain Incremental-refresh for your power BI model.

Introduction

Working with large volumes of data within Power BI can often be very time consuming, especially when it comes to refreshing datasets and it is not uncommon for large datasets refreshes to take several hours. In a world that demands near real-time insights, these loads have become an impediment to the business.

Incremental Refresh was added to the Power BI service to reduce the time taken to refresh a dataset by merely refreshing only the most recent data and not the entire dataset. By reducing the amount of data that needs to be re-loaded, refreshes take not only less time but also consume fewer resources doing it.

What is incremental refresh and how does it work

Incremental Refresh implements partitioning where a partition is a small part of a larger table, each partition is assigned a range of values usually by the date that gets stored in that partition. The power of partitioning is that the Power BI service can refresh a single partition or even a group of partitions instead of the entire table as we see in Power BI service currently. It is generally true that our older data never changes such as last year's sales and then new data is continuously changing, such as today's sales. The ability to refresh only today's sales and not last year's sales results in a lot of saving of time and effort. Thus, benefits of this incremental refresh are:

Faster Refreshes:

As each partition contains only a small percentage of the total rows, we reduce the number of rows that need to be imported.

Fewer resource required

As only a small portion of the data is loaded the number of resources required from both the source system and the power BI service is reduced including locks on source systems, CPU and network resources.

More reliable processes

As Power BI datasets often use core business systems as their source, long running queries may affect their performance. By reducing the number of rows, we improve not only the speed of the load but also reduce the risk of query timeouts and other network related issues.

Requirements

Before setting up Incremental Refresh there are a few requirements that must be met

Premium Workspace

Incremental refresh is currently only available to datasets published to a premium workspace, while the Incremental Refresh policy is set up in Power BI Desktop the feature is not activated until the report is published to a premium Workspace. The report can also not be published to a non-premium workspace such as 'My Workspace' once it has an Incremental Refresh policy applied.

Query Folding data source

While not absolutely necessary, it is recommended that Incremental Refresh only be set up on a data-sources that support query folding. Query folding is the ability of Power Query to adapt some of the transformations that are usually done in the mashup engine and push them down to the source system. Many sources, such as text files and websites, are unable to do this. However, data sources such as relational database and some OData sources can do this.

Transaction dates

For Incremental refresh to work, it requires a date of transaction (such as a sales or event date). This record should also be non-volatile meaning that details of this event should not change after a period of time as these changes will not be loaded if they occur outside of the refresh window. Another requirement for this date column is that there are no future dated transactions as these will not be loaded as part of the incremental refresh.

Enabled Incremental Refresh feature

At the time of writing Incremental refresh is still a preview feature. Therefore, you need to enable it in the Power BI preview features tab which can be found at File ?? Option and Settings ?? Options ?? Preview features

Options



GLOBAL

- Data Load
- Power Query Editor
- DirectQuery
- R scripting
- Python scripting
- Security
- Privacy
- Regional Settings
- Updates
- Usage Data
- Diagnostics
- Preview features
- Auto recovery
- Report settings

Preview features

The following features are available for you to try in this release. Preview features might change or be removed in future releases.

- ☒ Shape map visual [Learn more](#)
- ☒ Spanish language support for Q&A [Learn more](#)
- ☐ New web table inference [Learn more](#)
- ☒ Incremental Refresh Policies [Learn more](#)
- ☒ Enable Q&A Live Connect [Learn more](#)
- ☒ Key influencers visual [Learn more](#)
- ☐ Personalized visualizations pane [Learn more](#)

Figure 17-01: Enabling the Incremental Refresh preview feature

Once these requirements have been met, we can setup incremental refresh.

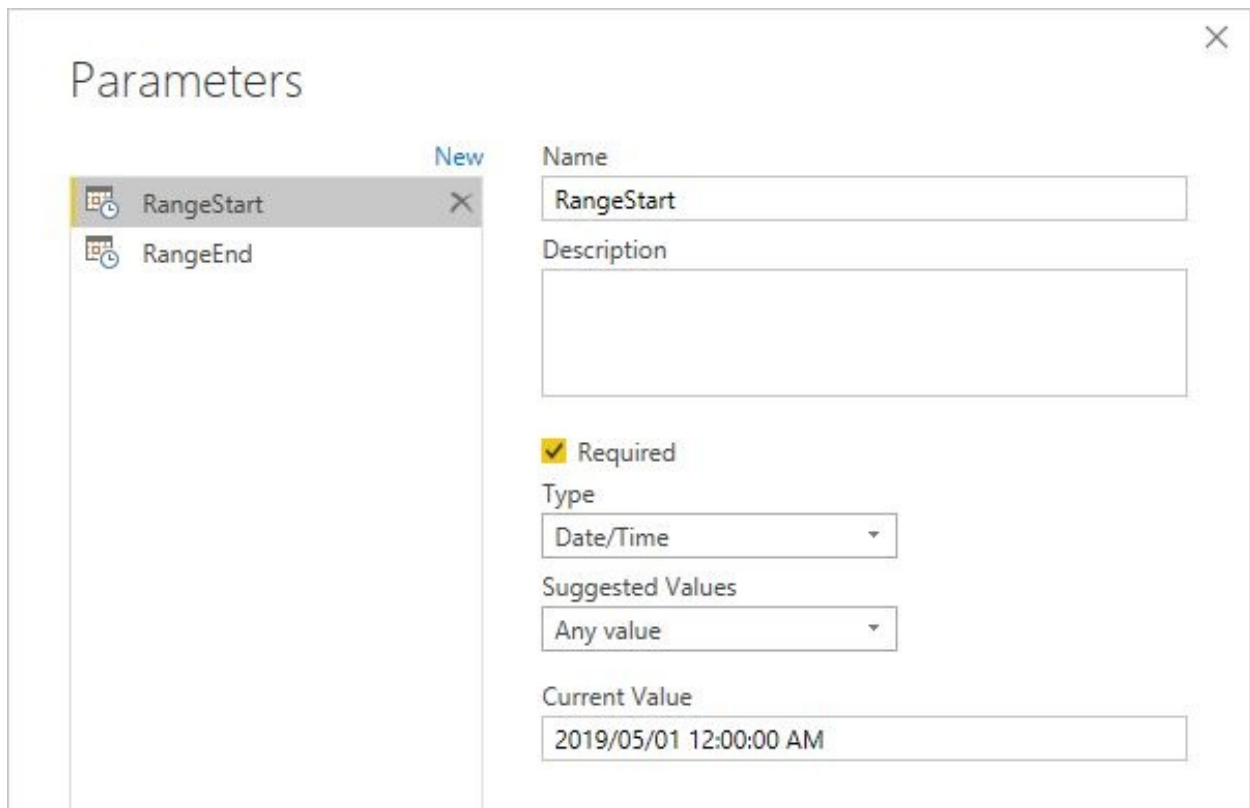
Setting up Incremental refresh

Setting up incremental refresh can be done in 4 easy steps

- **Step 1:** Create range parameters
- **Step 2:** Filter dataset using range parameters
- **Step 3:** Configure incremental refresh in Power BI desktop
- **Step 4:** Deploy and publish report

Step 1 – Create range parameters

For Incremental Refresh to be configured first need to create two specific parameters within the report called **RangeStart** and **RangeEnd**, these parameters must be defined as the Date\Time data type. The two parameters are used to determine the upper and lower boundary values of each partition, **RangeStart** being the first date in the partition and **RangeEnd** being the first date of next partition i.e. Up to but not including the **RangeEnd**. The selection of dates for these parameters is not critical and dates should be chosen to support the report development process such as a range of 13 months.



The screenshot shows the 'Parameters' dialog box in Power BI. On the left, there is a list of parameters: 'RangeStart' and 'RangeEnd'. 'RangeStart' is selected. On the right, the configuration for 'RangeStart' is shown. The 'Name' field contains 'RangeStart'. The 'Description' field is empty. The 'Required' checkbox is checked. The 'Type' dropdown is set to 'Date/Time'. The 'Suggested Values' dropdown is set to 'Any value'. The 'Current Value' field contains '2019/05/01 12:00:00 AM'.

Figure 17-02: Adding the new RangeStart and RangeEnd parameters to report

Step 2: Filter dataset using parameters

Once the two required parameters have been created, the next step is to filter the large table using those parameters by limiting the range of data to only dates between the two values. Once published Power BI dynamically replaces these original parameter values with new upper and lower bound values for each partition.

The date filter can easily be applied by using the between operator to the date

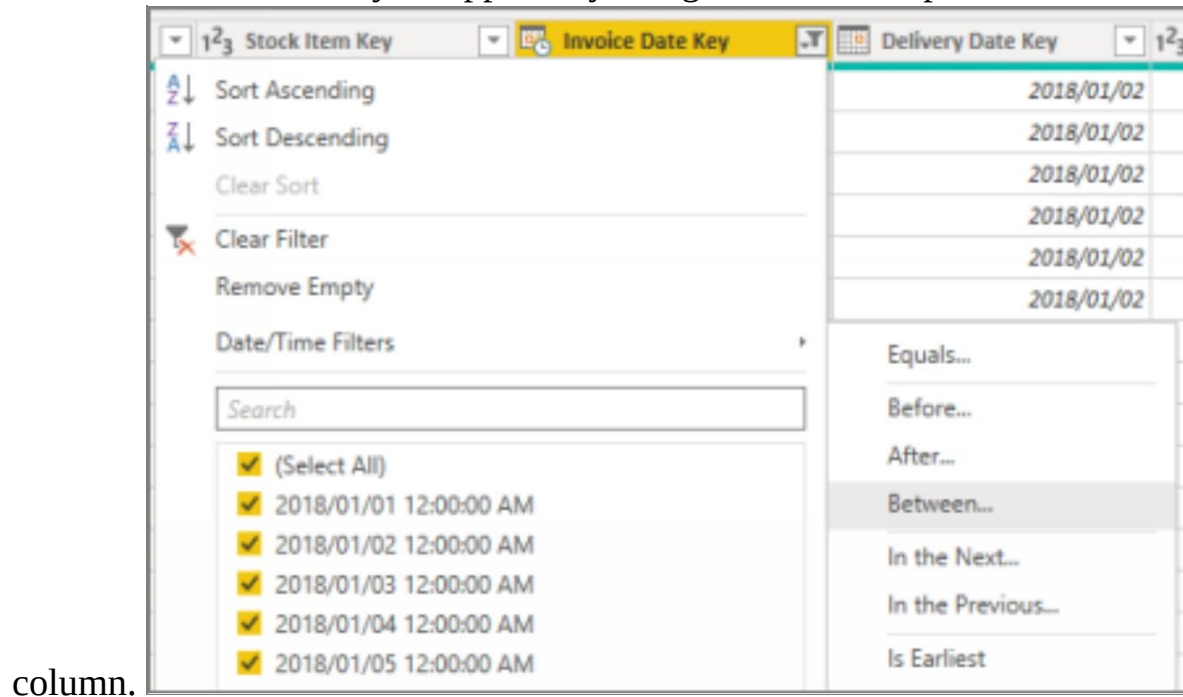


Figure 17-03: Adding a filter to the date column

Then the filter needs to be configured. Note: in order to avoid missing or duplicated records it is important that the 'is after or equal to' is selected for the first filter step. The next step is to assign this the parameter value by selecting the parameter option then selecting the **RangeStart** parameter, which is in the list of available parameters. The second filter is applied in the same way this time using the 'is Before' option and the **RangeEnd** value.



Figure 17-04: Assigning parameters to table filter

Alternatively, the filter can also be added by using M, The M code for this operation would look something like this

```
= Table.SelectRows("#"Changed Type", each [Invoice Date Key] >= RangeStart and [Invoice Date Key] < RangeEnd)
```

Step 3: Configure incremental refresh in Power BI desktop

Once the table has been correctly filtered using Power Query the next step is to set up the Incremental Refresh policy. This is done in the Power BI editor (You should select Close and Apply from the Power Query editor to return). The incremental policy can be found on the context menu (Right click on the table name) for the for the table.

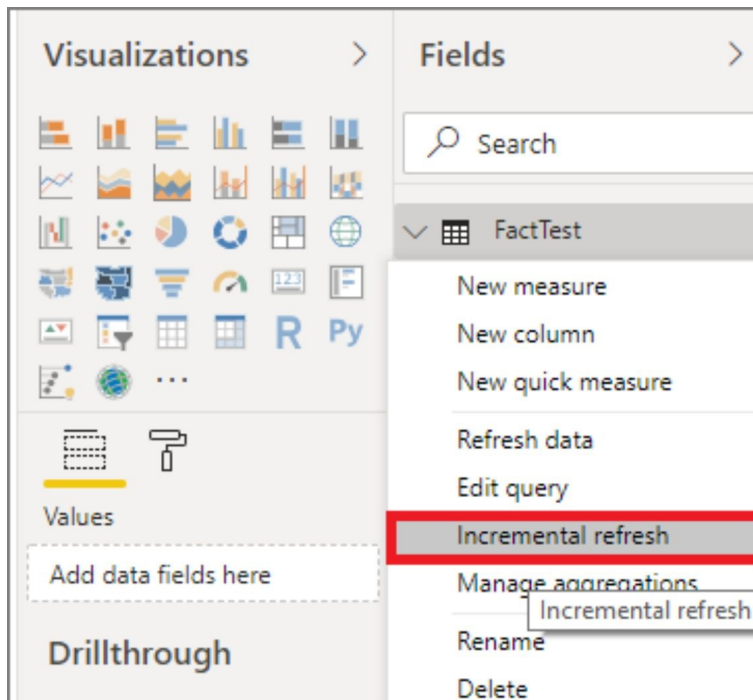


Figure 17-05: Adding Incremental refresh policy

Before the Incremental Refresh policy is set up, Power BI validates that the first two steps have been completed correctly, if there are any issues an error message such as the one below is displayed, this will need to be corrected before proceeding.

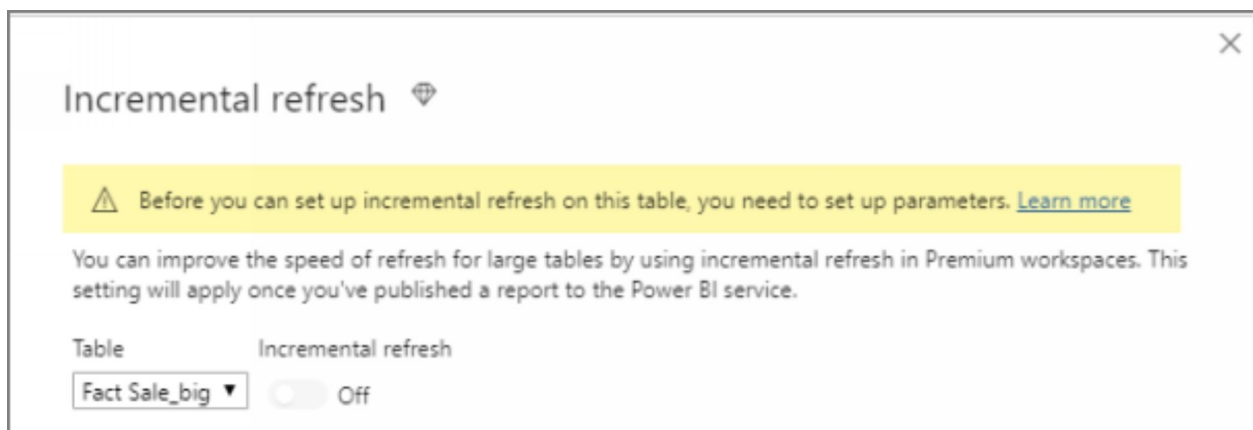


Figure 17-06: Incremental refresh unable to be setup due to error

If all the prior step were successfully completed, then the option to enable Incremental Refresh is provided by toggling the incremental tab alongside the name of the table that you want to configure incremental refresh.

Incremental refresh

You can improve the speed of refresh for large tables by using incremental refresh in Premium workspaces. This setting will apply once you've published a report to the Power BI service.

Once you've deployed this table to the Power BI service, you won't be able to download it back to Power BI Desktop. [Learn more](#)

Table Incremental refresh

Big SaleTable On

Store rows where column "Invoice Date Key" is in the last:

2 Years

Refresh rows where column "Invoice Date Key" is in the last:

10 Days

☐ Detect data changes [Learn more](#)

☐ Only refresh complete days [Learn more](#)

Apply all Cancel

Figure 17-07: Configuring the Incremental refresh policy

With the incremental refresh option enabled there are 4 configurations that need to be set

Store rows where “Date column” is in the last: The first configuration defines how much data must be kept in the model. In the case of the policy above at least 2 years of data are stored.

Refresh rows where the column “Date column” is in the last: This configuration option sets the number of periods that will be loaded each time the

model is refreshed. A balance must be found between catching all rows that have changed, and data volume, a range too big takes longer to load, and a range too small may miss late arriving data.

Detect data changes: Power BI can also be configured to only load partitions within the process window that has changed data changes. This is only possible if the data has a last changed date column, also note it that important to note that the timestamp column in SQL server does not work for this purpose.

Only refresh completed Days: The final configuration option is only to import completed periods; this is useful for reports then do not support partial periods.

Step 4: Deploy and publish report

Once the Incremental Refresh policy has been set up the next step is to publish the report. There are no particular actions required during this step; however, it should be noted that reports with an Incremental Refresh policy can only be published to a workspace backed by a premium capacity. There is little risk of saving to a non-premium workspace as the publish dialogue only lists Premium workspaces with non-premium workspaces lowlighted.

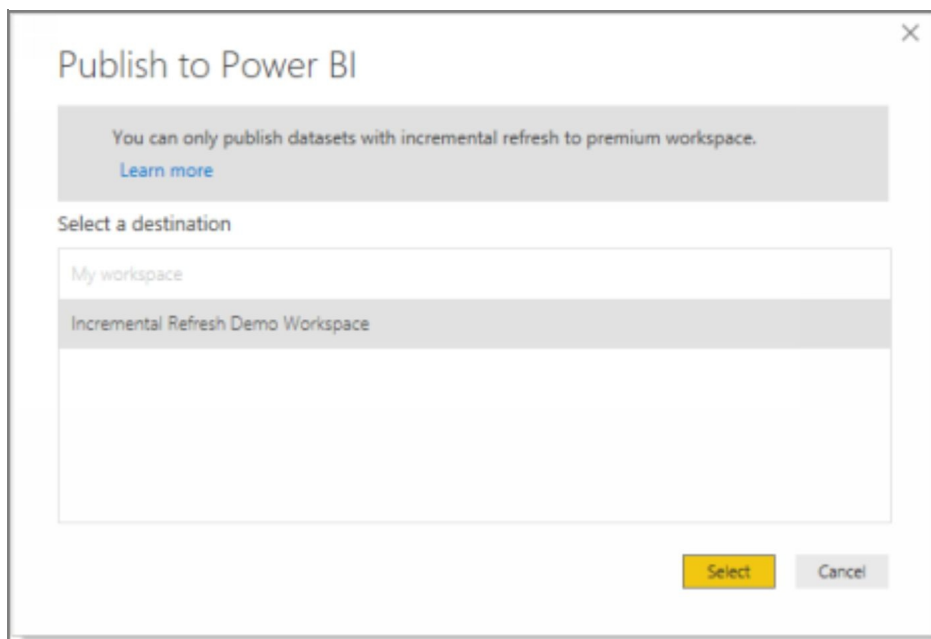
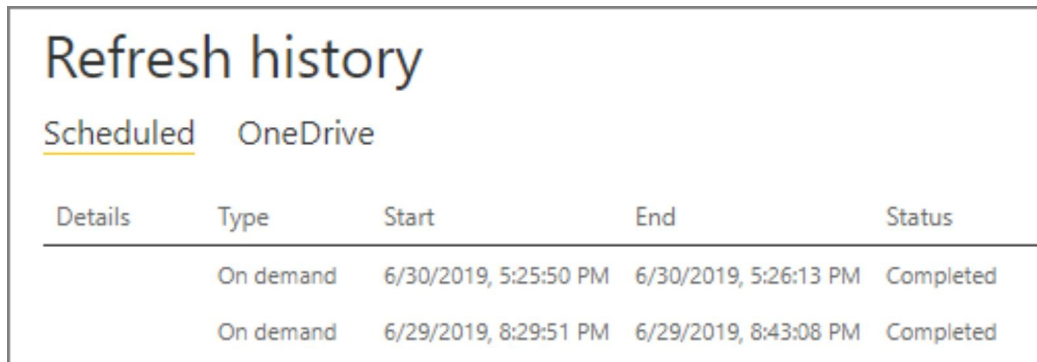


Figure 17-08: Publishing report with Incremental Refresh to a premium workspace

After the report has been published, you need to refresh the dataset as it still contains only the data from our filtering applied in the Power BI desktop, the Incremental Refresh policy has not yet been applied.

During the first data refresh Power BI removes the single partition of data created in Power BI desktop and replace it with a dynamic number of partitions required to support the Incremental Refresh policy. The first refresh takes approximately the same amount of time as a regular full load did as all the partitions need to be loaded. It is only on subsequent refreshes that we see the load time improve



<u>Scheduled</u> OneDrive				
Details	Type	Start	End	Status
	On demand	6/30/2019, 5:25:50 PM	6/30/2019, 5:26:13 PM	Completed
	On demand	6/29/2019, 8:29:51 PM	6/29/2019, 8:43:08 PM	Completed

Figure 17-09: Difference in processing times of first and second refreshes.

Note: That first refresh took approximately 14 minutes while the second refresh took only 23 seconds

This is all that needs to be configured when setting up Incremental Refresh; the Power BI service manages all subsequent administration tasks such as creating new, merging existing and deleting old partitions. There is no further action required by the report creator. The remainder of this chapter explores the inner working of how Incremental Refresh is implemented.

Looking under the covers

Under the hood of Power BI lies Analysis Services and more specifically SSAS Tabular, which is the engine that is responsible for both storing data and executing queries against that data. For years Business Intelligence practitioners have used partitioning to manage data loads into and sometimes out of the model. Creating an effective partitioning strategy was time-consuming and occasionally error-prone process. In the Incremental Refresh, the Power BI implementation of partitioning all the work of creating, loading, merging and deleting partitions is handled for you keeping in line with the easy to start self-service goals of Power BI.

However, what do partitions look like under the hood, in this section we look at how Power BI implements its partitioning strategy. While it is possible to modify the policy to get different behaviours we will use the Incremental Refresh policy created in Figure 25.7 where the data retention period is set to 2 years, data refresh is set to 10 days and the other to option are not enabled.

To achieve this Power BI will create several partitions. Conceptually the simplest would be to create one partition per day. However, it becomes inefficient to have many smaller partitions. Power BI begins by creating daily partitions, when there are more than a calendar months' worth of daily partitions and those daily partitions are outside of the 10-day refresh window then those daily partitions will be merged into a single monthly partition. When there are three consecutive monthly partitions those are merged into a quarterly partition, which will then be merged into an annual or year partition when there are 4 of them all in the same year, that was certainly a mouthful so the graphic below should make this easier to understand.

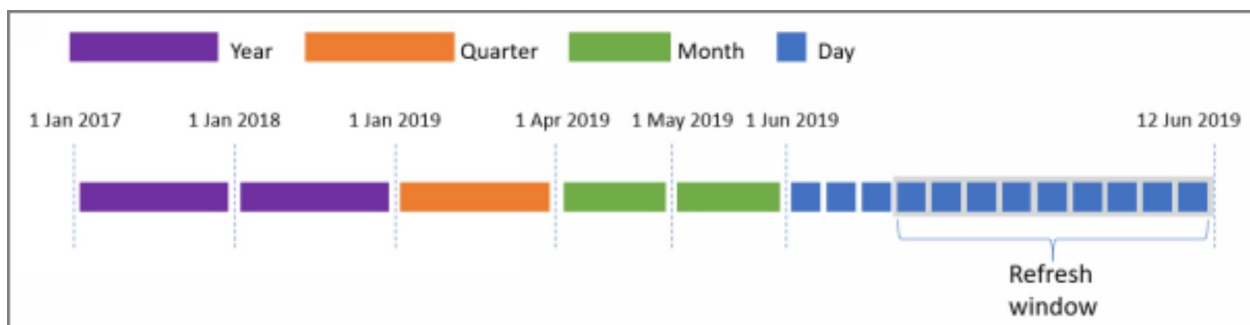


Figure 17-10: Conceptual view of the partitions created by incremental refresh

Assuming the date was 12 June 2019 then there would be 17 partitions (2 Year, 1

Quarter, 2 Month and 12 Day) this can be verified by using the new XMLA endpoint available in Power BI premium, XMLA is another set of tools in Analysis Services that allow us to query and eventually modify (on roadmap but not available) the structure of the Power BI dataset. The Path for the endpoint is found in the Workspace setting

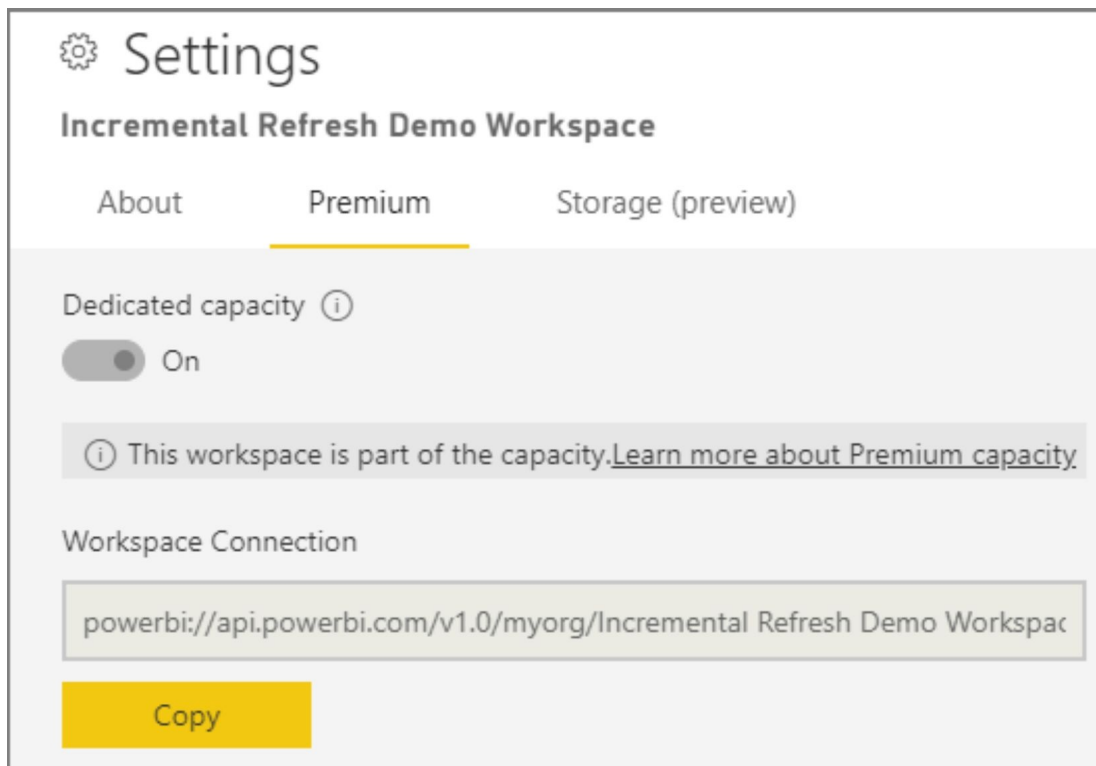


Figure 17-11: Physical view of partitions

Then using a tool such as SQL Management studio, a connection to the Power BI datasets can be made.

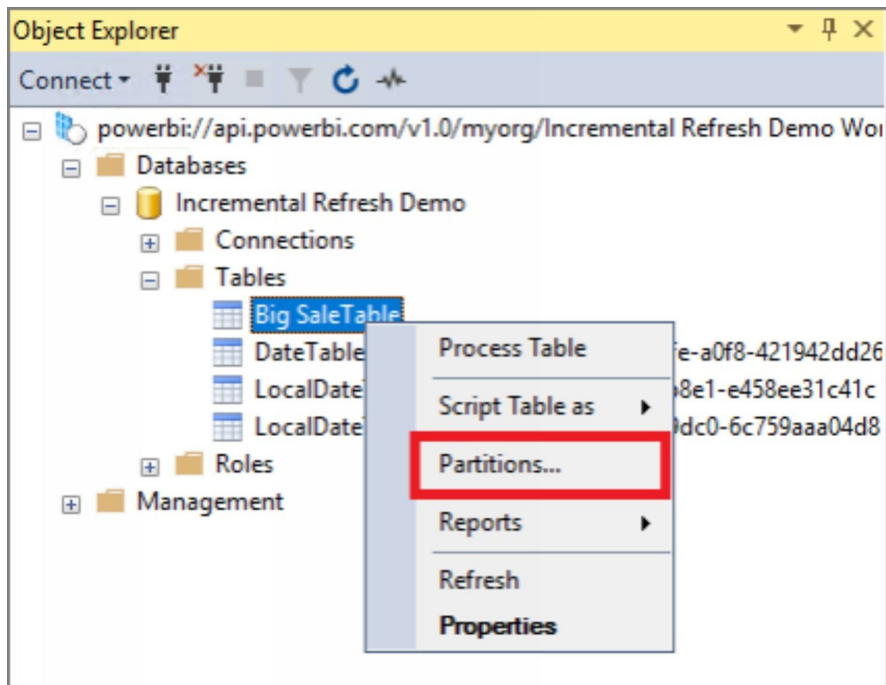


Figure 17-12: Connecting to Power BI datasets using the XMLA endpoints

When we query the partitions in the model we get, as we expected 17 partitions back

Partitions	
Partition Name	# Rows
2017	324630
2018	359205
2019Q1	85455
2019Q204	30370
2019Q205	29615
2019Q20601	0
2019Q20602	685
2019Q20603	1260
2019Q20604	1975
2019Q20605	1715
2019Q20606	1680
2019Q20607	635
2019Q20608	0
2019Q20609	1240
2019Q20610	680
2019Q20611	1535
2019Q20612	1235

Figure 17-13: Physical view of partitions

In this view, we see the 17 partitions we expected (2 Year, 1 Quarter, 2 Month and 12 Day) we also see some additional data such as the number of rows in each partition. The next thing to look at is the definition for the partitions themselves, we briefly mentioned XMLA however Power BI and SSAS Tabular (as of compatibility level 1200) now use Tabular Object Model (TOM), which is a JSON based format for representing the metadata of the data set. Scripting out the definition for the partition named '2019Q205' which is May 2019 we get.


```

{
  "createOrReplace": {
    "object": {
      "database": "Incremental Refresh Demo",
      "table": "Big SaleTable",
      "partition": "2019Q205"
    },
    "partition": {
      "name": "2019Q205",
      "mode": "import",
      "source": {
        "type": "policyRange",
        "start": "2019-05-1T00:00:00",
        "end": "2019-06-01T00:00:00",
        "granularity": "month"
      }
    }
  }
}

```

Figure 17-14: TOM for the May 2019 partition

Here we can see several useful bits of information starting with the names of the database, table and partition. We also see more detailed information about the partition itself

Type: This tells us what the Incremental refresh policy is being used, only Policy Range is available at this time, other options include M for tables that are derived from Power Query and Calculated for calculated tables created with DAX.

Start & End: The values for start and end correspond the first (Included) and last (not included) values in the partition, these map to the **StartRange** and **EndRange** parameters that we created in Step 1.

Granularity: This denotes the partition type and can be year, quarter, month or day.

The final task that we will look at is to examine the query that is sent to the data source, to get an idea of what the query will look like we can use the View Native Query option in the applied steps in Power Query, this shows us what the effect of query folding will look like

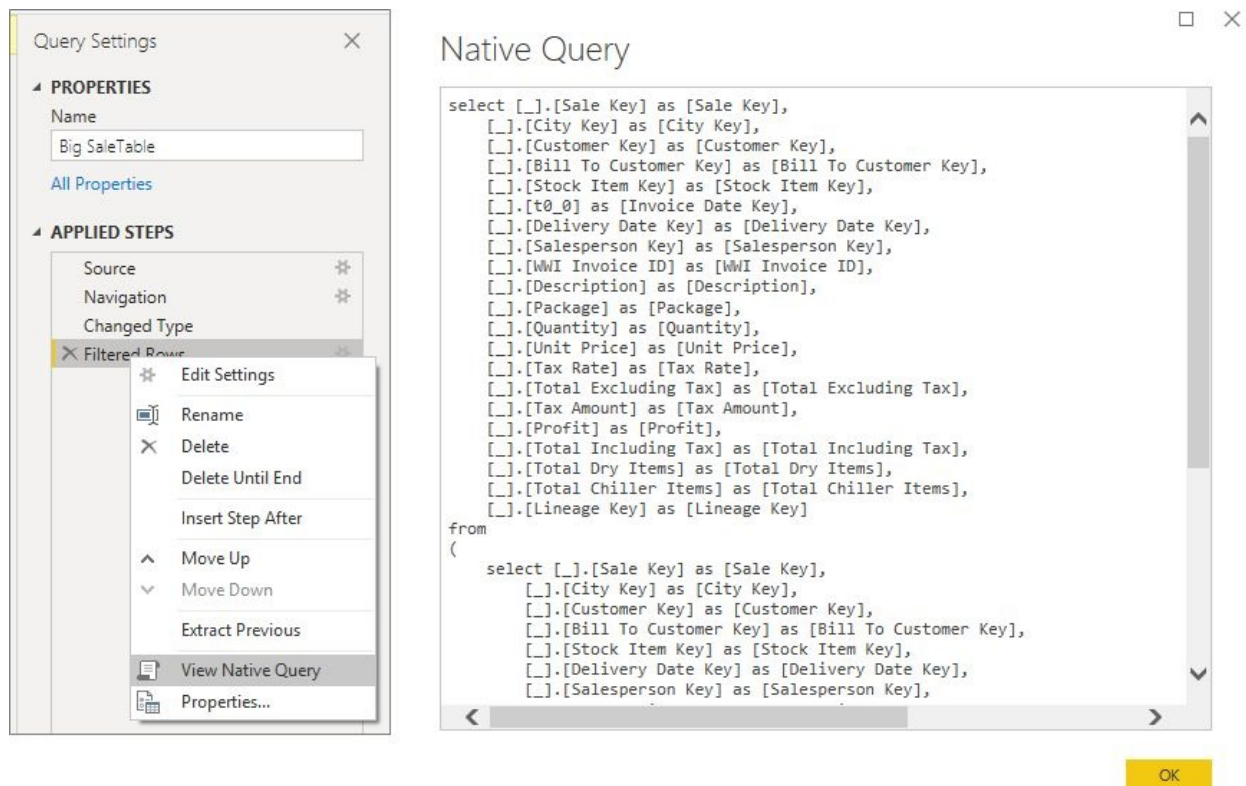


Figure 17-15: Query generated by power Query after Query folding

An alternative approach or one to validate this approach is to use extended events, or SQL profiler capture all incoming queries, below is the query generated.

```

execute sp_executesql N'
select [_].[Sale Key] as [Sale Key],
    [_].[City Key] as [City Key],
    [_].[Customer Key] as [Customer Key],
    [_].[Bill To Customer Key] as [Bill To Customer Key],
    [_].[Stock Item Key] as [Stock Item Key],
    [_].[t0_0] as [Invoice Date Key],
    [_].[Delivery Date Key] as [Delivery Date Key],
    [_].[Salesperson Key] as [Salesperson Key],
    [_].[WWI Invoice ID] as [WWI Invoice ID],
    [_].[Description] as [Description],
    [_].[Package] as [Package],
    [_].[Quantity] as [Quantity],
    [_].[Unit Price] as [Unit Price],
    [_].[Tax Rate] as [Tax Rate],
    [_].[Total Excluding Tax] as [Total Excluding Tax],
    [_].[Tax Amount] as [Tax Amount],
    [_].[Profit] as [Profit],
    [_].[Total Including Tax] as [Total Including Tax],
    [_].[Total Dry Items] as [Total Dry Items],
    [_].[Total Chiller Items] as [Total Chiller Items],
    [_].[Lineage Key] as [Lineage Key]
from
(
    select [_].[Sale Key] as [Sale Key],
        [_].[City Key] as [City Key],
        [_].[Customer Key] as [Customer Key],
        [_].[Bill To Customer Key] as [Bill To Customer Key],
        [_].[Stock Item Key] as [Stock Item Key],
        [_].[Delivery Date Key] as [Delivery Date Key],
        [_].[Salesperson Key] as [Salesperson Key],
        [_].[WWI Invoice ID] as [WWI Invoice ID],
        [_].[Description] as [Description],
        [_].[Package] as [Package],
        [_].[Quantity] as [Quantity],
        [_].[Unit Price] as [Unit Price],
        [_].[Tax Rate] as [Tax Rate],
        [_].[Total Excluding Tax] as [Total Excluding Tax],
        [_].[Tax Amount] as [Tax Amount],
        [_].[Profit] as [Profit],
        [_].[Total Including Tax] as [Total Including Tax],
        [_].[Total Dry Items] as [Total Dry Items],
        [_].[Total Chiller Items] as [Total Chiller Items],
        [_].[Lineage Key] as [Lineage Key],
        convert(datetime2, [_].[Invoice Date Key]) as [t0_0]
    from [Fact].[Sale_big] as [_]
) as [_]
where [_].[t0_0] >= Convert(datetime2, ''2019-05-01 00:00:00'') and [_].[t0_0] < convert(datetime2, ''2019-06-01 00:00:00'')

```

Figure 17-16: Query generated by power Query after Query folding

Looking at the T-SQL generated in the statement above it can be seen how the power query have used query folding to pass the filter expression to SQL server where it is more effectively implemented, and the partitions have been used to dynamically replaced the date values in each query to only return the data required.

Limitations and things to look out for

While Incremental Refresh has made this process amazingly simple there are a few things to be aware of.

Incremental refresh does not support future dated transactions and there does not seem to be an option for this. This is usually not a problem for most systems but if you do have future dated transactions then this may be an issue.

There does not seem to be a way to force a full load after the initial publish load, e.g. if we wanted to reload as a once off exercise possible due to some correction in the business process, this is not currently possible, you will need to re-import the PIDB file to the service (if changes have been made to the report through the online report editor then these changes will be lost). Remember that the PIBX cannot be downloaded to the desktop once it has been published as the desktop does not know how to handle partitions.

If you are using a relational database as a source, make sure that you have indexes that support incremental refresh, the optimal would be to have the partition date used as the cluster index.

Summary

Incremental Refresh has been well implemented and will meet the needs of most Power BI users, and its use should be encouraged in all reports that import a large number of rows. The limitation of requiring a premium workspace impedes this, and we would love to see this removed as more efficient report loads are good for both the report creator and the Power BI service as a whole

To learn more about incremental refresh you can view the official Microsoft documentation at <https://docs.microsoft.com/en-us/power-bi/service-premium-incremental-refresh>

About the Author



Michael Johnson is a Business Intelligence architect and Microsoft data platform MVP living in Johannesburg, South Africa. Michael has been working with data for the last 15 years and has run the local SQL Server User Group and SQL Saturday conference event for the last few years.

He enjoys showing people new tools and technologies that allow them to work more effectively with their data.

Chapter 18: Report Server Administration

Author: Shree P Khanal

This Chapter will help to bridge the gap of understandings required for those organizations that are not using cloud infrastructure for Power BI reporting. A non-cloud Power BI architecture can be planned using a Power BI feature called Power BI Report Server. Though this feature gives an added flexibility, it has to manage and administer a new service.

Power BI Report Server

Today, Power BI is much more than a cloud-based reporting service. Along with the enhancement in its services and the tremendous increase of users, many businesses now demand of having data and reporting solutions in on-premises. Therefore, Microsoft has introduced an option to fully deploy Power BI on-premises. This version of the Power BI on-premise is called the Power BI Report Server.

Power BI is much more than a cloud-based reporting technology. With the increasing demand of having data and reporting solutions on-premises by many businesses, Microsoft introduced an option to fully deploy Power BI on-premises. The version of Power BI on-premise is called Power BI Report Server.

In this chapter, you will learn everything you need to know about Power BI Report Server, a complete on-premise solution. Including, how to install and configure it as well as will highlight the pros and cons as we get along it. At the end of this chapter, you will be able to decide whether Power BI on-premise is the right choice for you or not and how can you implement and configure this feature.

What Is Power BI Report Server?

Power BI Report Server is an edition of SQL Server Reporting Services that can host Power BI reports. To install the Power BI Report Server, you don't need to have a SQL Server installation media; it comes with its setup files. Power BI Report Server can host Power BI reports as well as Reporting Services like SSRS Reports.

Along with Power BI Report Server, there will be an instance of Power BI Desktop installation. The Power BI Desktop edition which comes along with the report server should be used to create the Power BI reports else reports cannot be hosted on the report server. Power BI Report Server also regularly gets updates like Power BI Desktop giving an essence as using a Power BI desktop version

Power BI Report Server Can be installed on the following Microsoft operating systems

Server - Operating System: Windows Server 2012 or higher versions

Workstation - Operating System: Windows 8 or higher versions

Web browser:

- Microsoft Edge,
- Microsoft Internet Explorer 11,
- Google Chrome,
- Mozilla Firefox

Note: The Power BI Report Server installation is only supported on 64-bit OS and requires .Net framework 4.6 installed in it.

For more refer to this link:

Hardware and software requirements for installing Power BI Report Server:

<https://docs.microsoft.com/en-us/power-bi/report-server/system-requirements>

Download Power BI Report Server

Before you begin installation, download the latest edition of Power BI Report Server from following link:

<https://powerbi.microsoft.com/en-us/report-server/>

Download following items on the server as well as client/desktop respectively.

- Power BI Report Server and
- Power BI Desktop Report Server edition (available in X86 and X64 versions).

Installing Power BI Report Server

Installation of Power BI Report Server is simple. The user just needs to run the setup file and follow the instructions. Figure 18-01 shows the very first screen of the installation process.

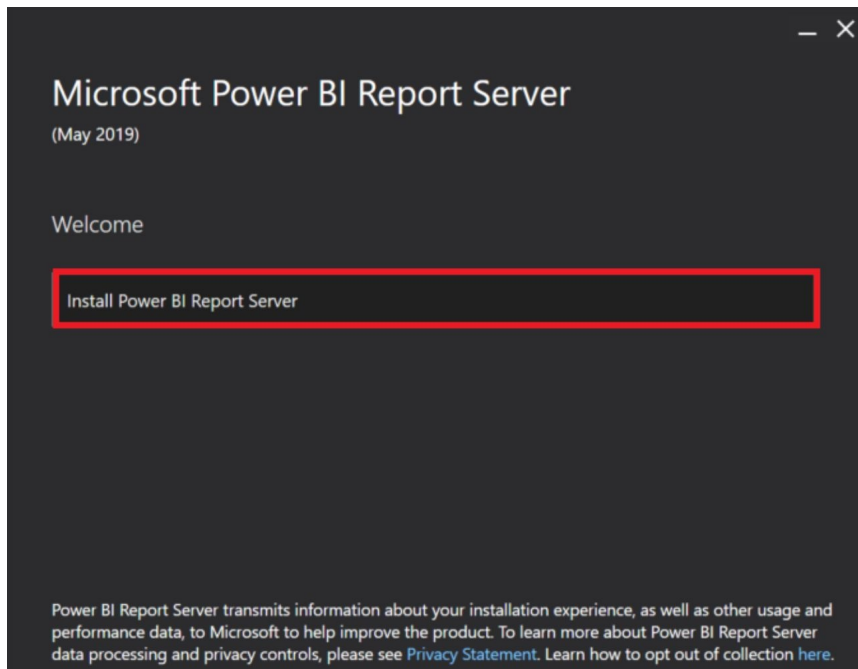


Figure 18-01: Installing Power BI Report Server

You can choose between the evaluation edition (180 days) and the licensed version of Power BI Report Server. Besides, the development edition is also available free of cost.

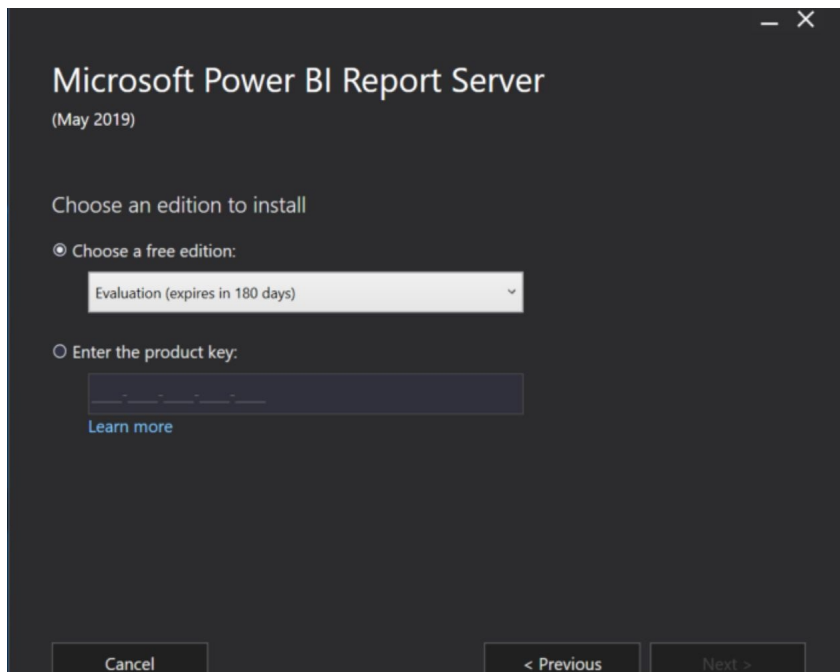


Figure 18-02: Choose the edition

Accept the license terms before moving to the next screen.

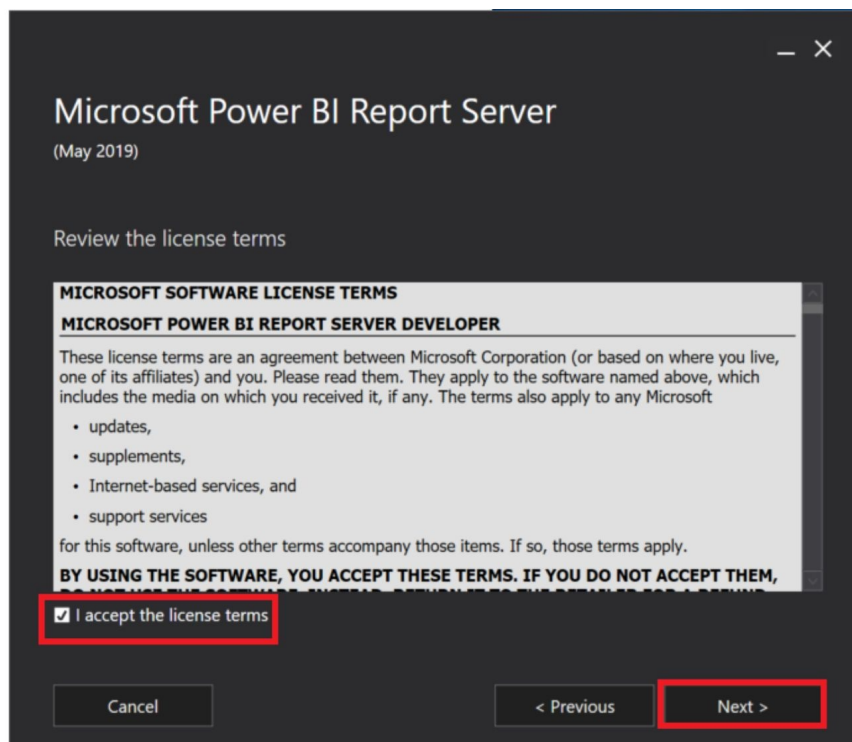


Figure 18-03: License terms for BI Report Server

Earlier, it was mentioned that you don't need to have SQL Server installed to get the Power BI Report Server. However, the SQL Server database engine is a

prerequisite for the report server to run. If you do not already have SQL Server installed, the Report Server setup process will install the database engine for you, as shown in Figure 18-04.

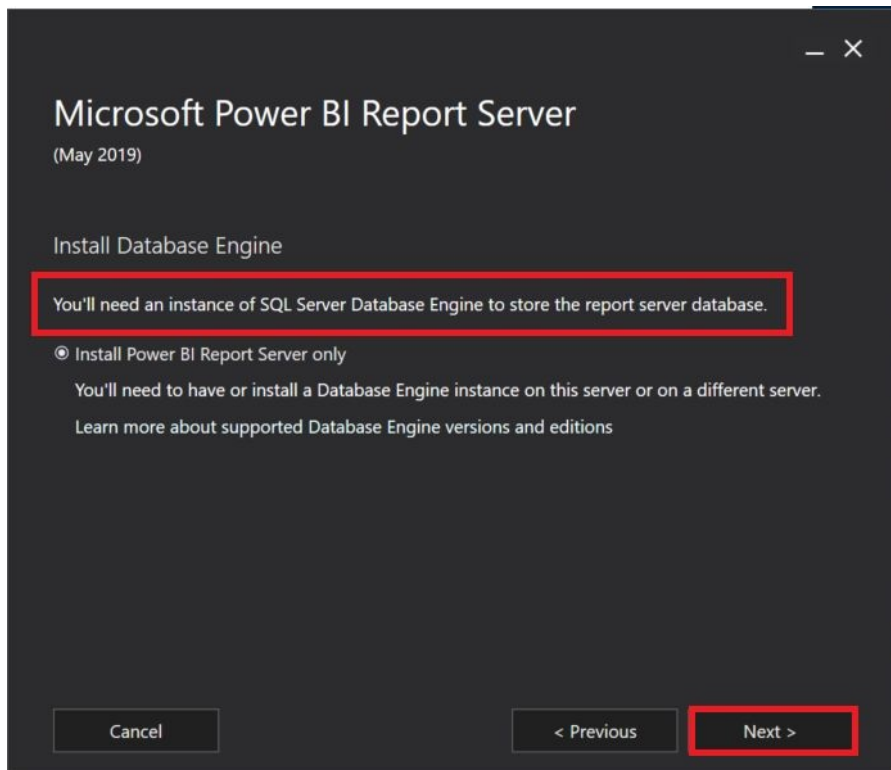


Figure 18-04: SQL Server database engine is required for the Power BI Report Server

Follow the on-screen instructions for the next steps.

Finally, restart the Report Server when the installation is successful. Upon restart, you will need to configure it.

Configuring the Power BI Report Server

After the installation is successful, open the configuration section by clicking on “Configure Report Server” button as shown in Figure 18-05. The instructions may ask you to restart the server. Once you restart, access the **Report Server Configuration Manager** from **Start | Microsoft Power BI Report Server**.

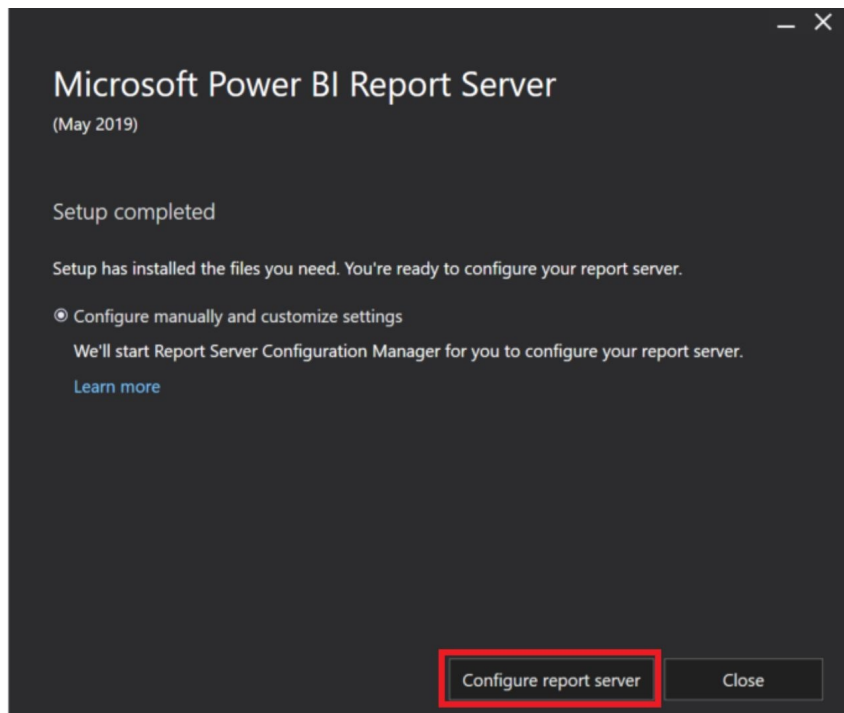


Figure 18-05: Configure report server

In the “**Connect** to a Power BI report server instance” dialogue box, make sure that your local report server instance (for example **PBIR**) is selected and click **Connect**.

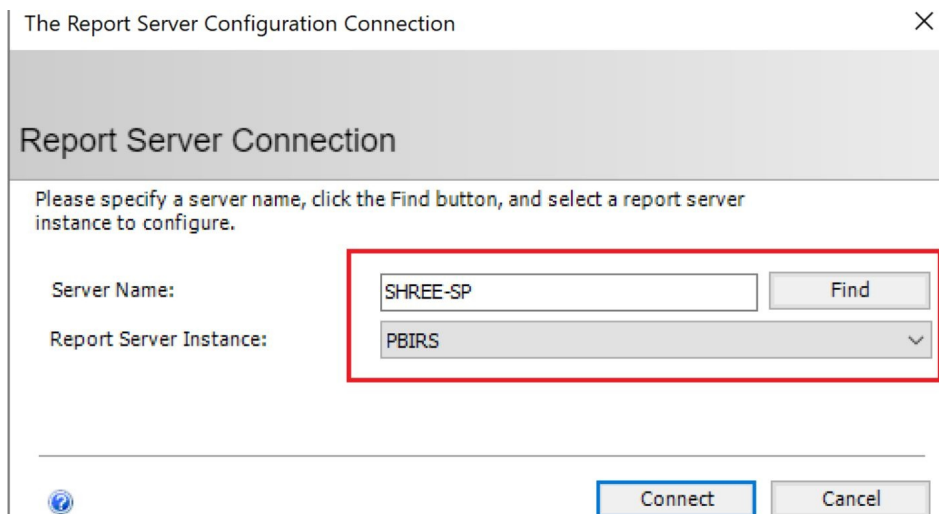


Figure 18-06: Report Server Configuration Manager connection

› setup a service account for the Report Server, access the ‘Service Account’ tab.

You can leave it to default or setup a new account.

Web Setup

Report Server needs Web setup for it to work. There are two URLs that you must configure. The first is for the web service and the other is for the web portal. These are accessed from their own tabs.

Web Service Setup:

To configure the web service, click on the **Web Service URL** (🌐) tab, make sure that the **Virtual Directory** is set to **ReportServer** by default and the **TCP Port** is set to **80**. Alternatively, you can use your Report Server Name and a different port number.

Note: The virtual directory name identifies which application receives the request. Because an IP address and port can be shared by multiple applications, the virtual directory name specifies which application receives the request.

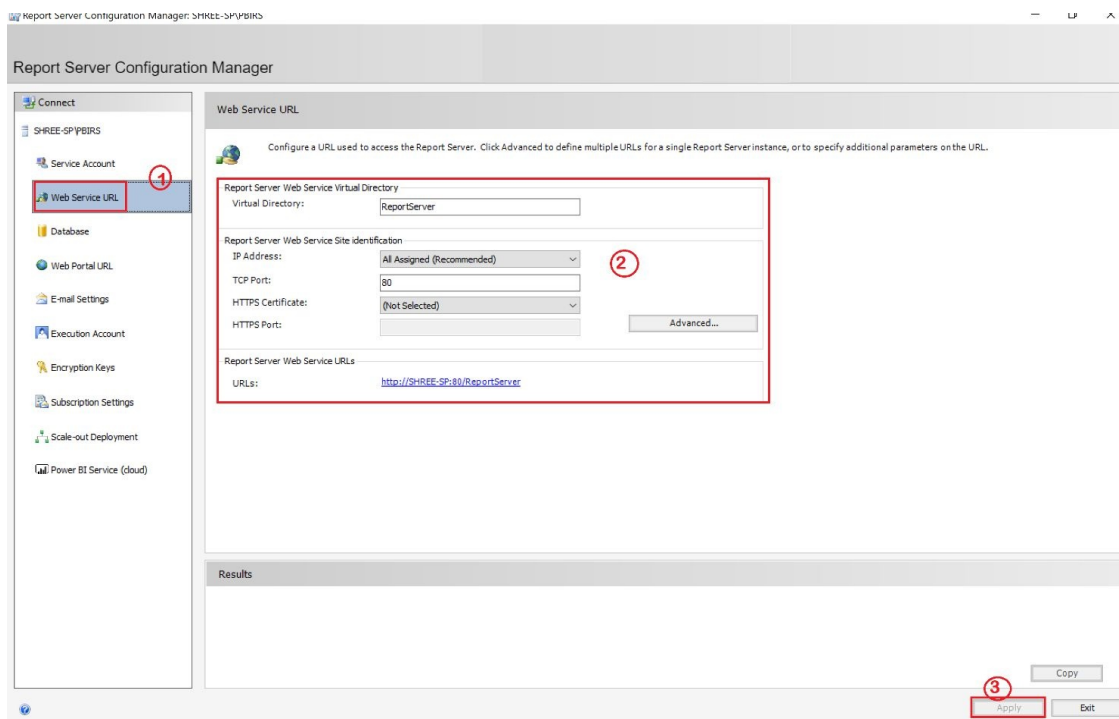


Figure 18-07: Web Service URL Configuration

Click on **'Apply'** after you have provided the necessary details. You will see success messages and a URL that you can click to open the report server's web service. For example, notice the highlighted URL in Figure 18-08.

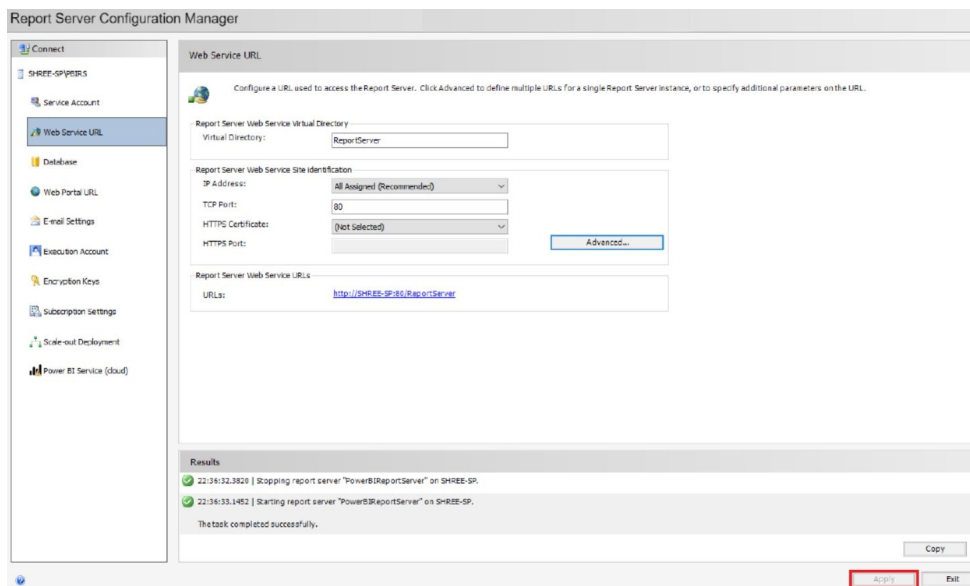


Figure 18-08: Web Service URL configuration

When you click on the URL, you should be able to open the web service's page without any issues.

Figure 18-9 shows an example of what that service page will look like.

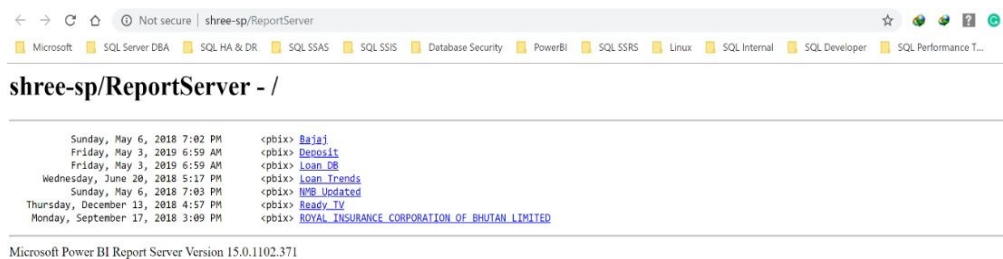



Figure 18-09: Web Service URL browsed in a web browser

In the Report Server page, initially, you will only see the version of the Report Server along with its name. However, later when you upload Power BI files, you'll be able to see the contents.

Database Setup:

On the **Database** tab() , make sure that **the SQL Server Name** points to your local SQL Server instance & **Report Server Mode** is set to **Native**.

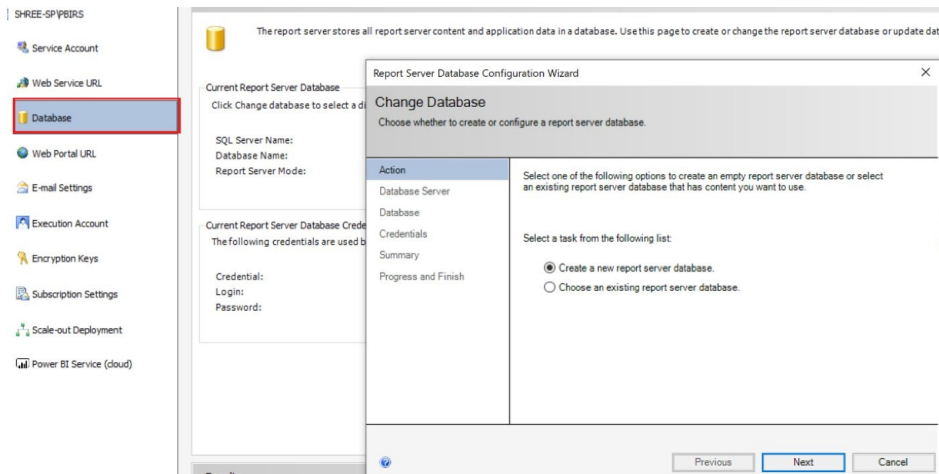


Figure 18-10: Database Configuration

To setup, follow these steps:

Click **Create Database** to open the Report Server Database Configuration Wizard.

Later, you can update the database setting by clicking on the **Change Database** button. On the screen that follows, provide new configuration values for Server Name, Authentication Type, Username and Password as shown in Figure 18-11.

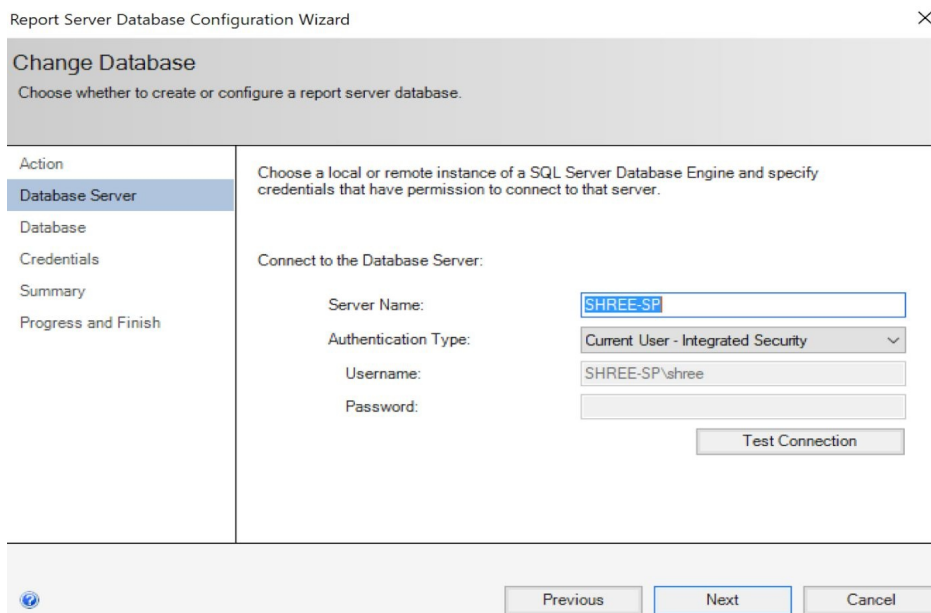


Figure 18-11: Database Server Configuration

In the **Database** page, type **ReportServer** as the default **Database Name**, select **Native Mode** for the **Report Server Mode**, then click **Next to continue**.

Report Server Database Configuration Wizard

Change Database
Choose whether to create or configure a report server database.

Action
Database Server
Database
Credentials
Summary
Progress and Finish

Enter a database name, select the language to use for running SQL scripts, and specify whether to create the database in native or SharePoint mode.

Database Name: ReportServer
Temp Database Name: ReportServerTemp
Language: English (United States)
Report Server Mode: ☒ Native Mode ☐ SharePoint Integrated Mode

Previous Next Cancel

Figure 18-12: Database Configuration

In the **Credentials** page, select **Service Credentials** for the **Authentication Type**, then Click **Next** and finally **Finish** to complete the wizard.

Report Server Database Configuration Wizard

Change Database
Choose whether to create or configure a report server database.

Action
Database Server
Database
Credentials
Summary
Progress and Finish

The following information will be used to connect to an existing report server database. Verify this information is correct before you continue.

SQL Server Instance: SHREE-SP
Report Server Database: ReportServer
Report Server Mode: Native
Authentication Type: Service Account
Username: NT Service\PowerBIReportServer
Password: *****

Previous Next Cancel

Figure 18-13: Database Configuration steps

Web Portal Setup

To set up the web portal, click on Web Portal URL tab. You can then configure the service as needed using the interface in Figure 18-14. When you're done, click on **Apply**. If the configuration process completes successfully, you will see a success message as shown in Figure 18-15. You can then click on the Web Portal URL to open it in a browser window, as in Figure 18-14.

Figure 18-14: Web Portal URL configuration

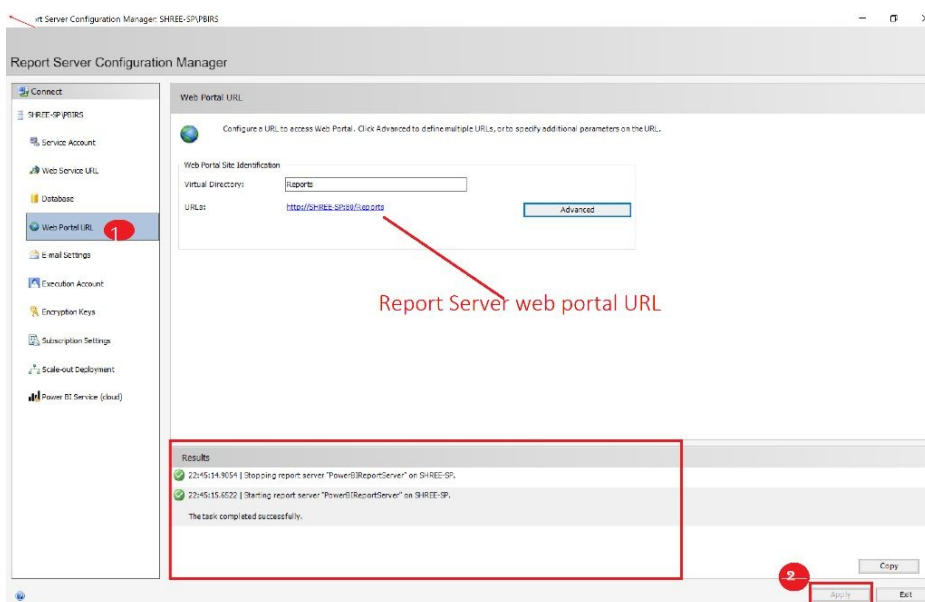


Figure 18-15: Web portal created

The web portal should show you the environment of Power BI Report Server's admin view.

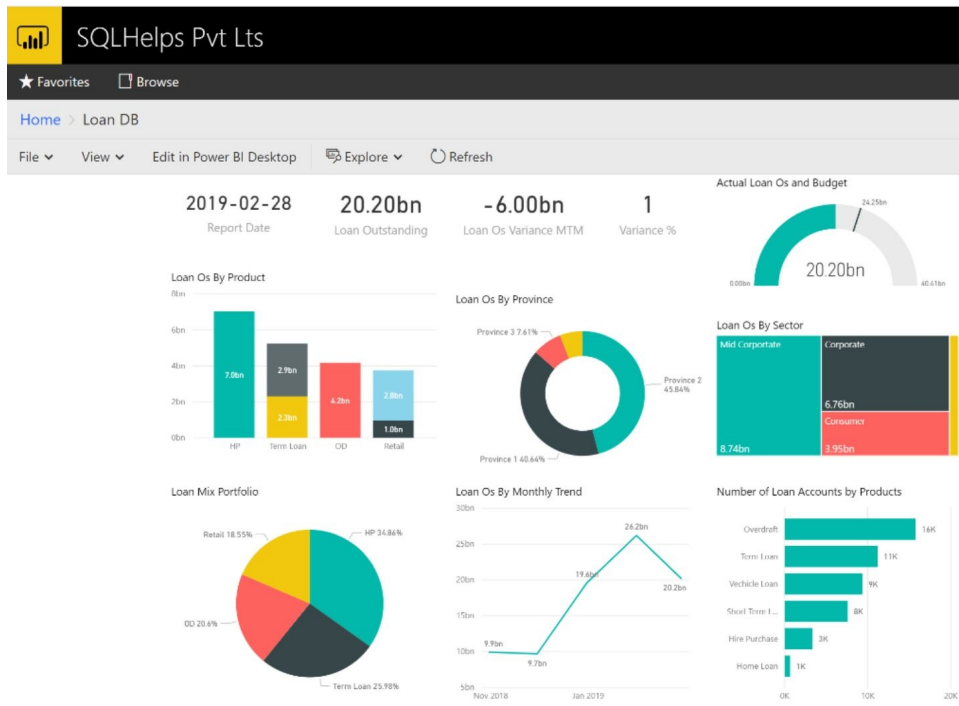


Figure 18-16: Power BI Report Server Web Portal

When installation and configuration of the Power BI Report Server are completed, you can close the Report Server Configuration Manager.

Installing Power BI Desktop Report Server

Power BI reports are developed with a specific tool called Power BI Desktop Report Server.

Download it from <https://www.microsoft.com/en-us/download/details.aspx?id=56722> then Start the installation wizard and complete the installation.



Figure 18-17: Installing Power BI Desktop for the Report Server

After a successful installation, you can open the Power BI Desktop Report Server which appears as in Figure 18-18. This interface looks similar to the regular Power BI Desktop interface.

Figure 18-18: Power BI Desktop Report Server

Developing Reports with Power BI Report Server

You can create a report in the Power BI Desktop Report Server by following the same steps as you would follow when creating a report in normal Power BI Desktop. You can even open a report developed with normal Power BI Desktop in the Power BI Desktop Report Server. Figure 18-19 shows a report in Power BI Desktop Report Server.

Figure 18-19: A report from Power BI Desktop opened in Power BI Report Server

To deploy reports, you need to connect to a report server as shown in Figure 18-20.

Figure 18-20: Connecting to Power BI Report Server from Power BI Desktop

Then, enter the Web Portal URL. Use the same URL that you had used when configuring the Report Server. Figure 18-21 shows such an address at the bottom of the page.

Figure 18-21: Report Server URL is needed to connect to the Power BI Report Server

After successful deployment, you will see a message with a link to the report. Figure 18-22 shows an example.



Figure 18-22: Publishing to the Power BI Report Server

Figure 18-23 shows an example of a report hosted on Power BI Report Server. This report is fully interactive and is similar to the report hosted in Microsoft's Power BI web service.

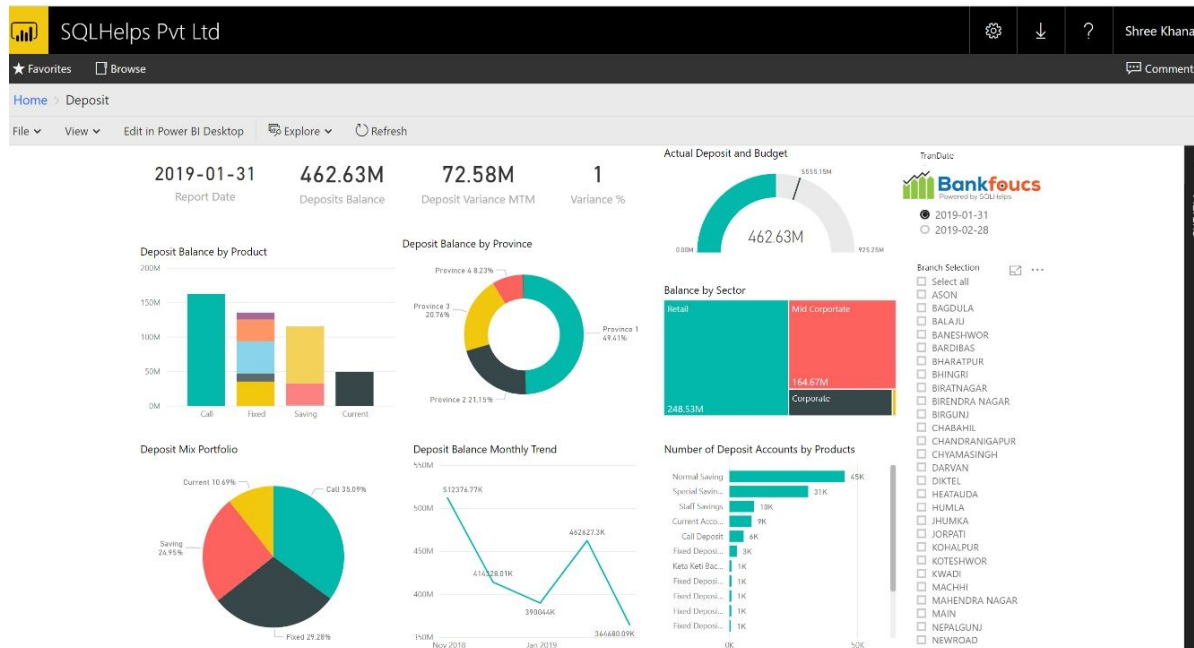


Figure 18-23: Reports are fully interactive in the Power BI Report Server

Figure 18-24 shows another way to publish a Power BI report to the report server.

You can do so by choosing the Upload option from the web portal's toolbar.



Figure 18-24: Uploading Power BI reports to the Report Server

Managing Datasets on the Report Server

A Power BI report published to the report server can be configured for scheduled data refresh. To configure, open the report server web portal, and click on the more options of the Power BI report. Figure 18-25 shows an example.

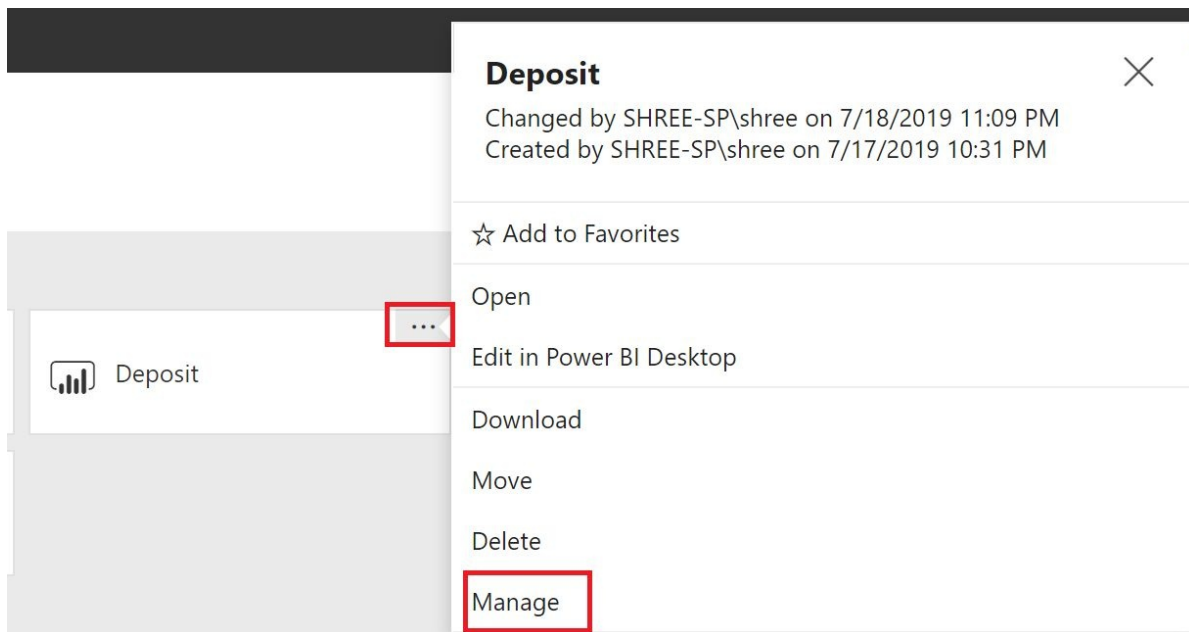


Figure 18-25: Manage the dataset of the report on the Report Server

Click on ‘Manage’ for data source configuration, connection to the data source, and to schedule a data refresh as required. Figure 18-26 shows the Manage tab for a report named Daily **deposit**.

Figure 18-26: Dataset properties configuration in Power BI Report Server

Schedule Refresh Requirement

If your report is sourced from a file, then you may need to schedule the report to refresh. At first, you need to specify file location from a network, as shown in the Figure 18-27.

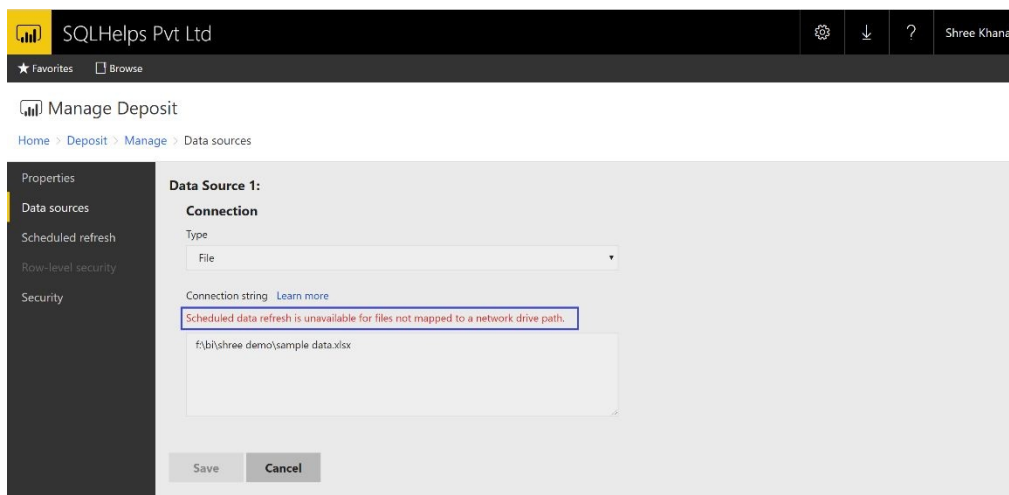


Figure 18-27: Scheduled data refresh is only available for files with a shared network drive path

If you use a network shared path to access the source file, you can set up the connection to the file as shown in Figure 18-28.

SQLHelps Pvt Ltd

★ Favorites Browse

Manage Deposit

Home > Deposit > Manage > Data sources

Properties

Data sources

Scheduled refresh

Row-level security

Security

\\shree-sp\demobank\deposit data.xlsx

Credentials

Log into the data source

Authentication Type

Windows Authentication

User name

shree.khanal@outlook.com

Password

.....

Test connection

Connected successfully

Figure 18-28: Setting up credentials for the data source connection and testing the connection

Make sure to save after this step. Otherwise, you won't be able to schedule the refresh process. Then click on the Scheduled refresh option on the left sidebar to create a refresh plan.

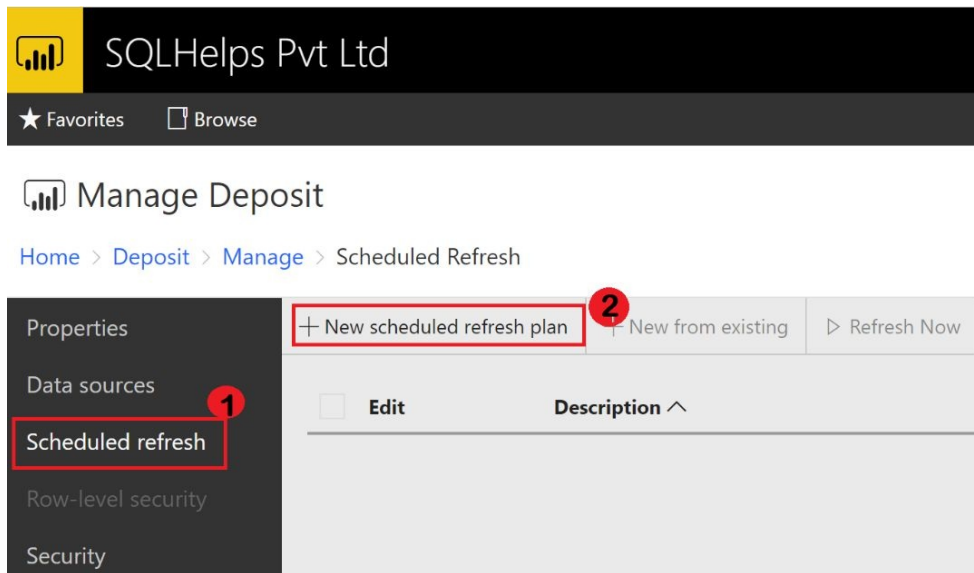


Figure 18-29: Creating a scheduled refresh plan for the dataset

The scheduled refresh configuration of the report server has more options than the Power BI Service. You can choose to schedule hourly, daily, weekly, monthly, or any custom period. Additionally, you can choose the start and end dates and many other configurations. Figure 18-30 shows some of the scheduling options that are available to you.

SQLHelps Pvt Ltd

★ Favorites □ Browse

Edit Schedule

Home > Deposit > Manage > Scheduled Refresh > Edit Scheduled Refresh Plan > Edit Schedule

Schedule details

Choose whether to run the report on an hourly, daily, weekly, monthly, or one time basis.

① All times are expressed in (UTC+05:45) Nepal Standard Time.

☐ Hour
 ☒ Day
 ☐ Week
 ☐ Month
 ☐ Once

Daily schedule

☐ On the following days:

☒ Sun ☒ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☒ Sat

☐ Every weekday

☒ Repeat after this number of days: 1

Start time: 02 : 00 AM

Start and end dates

Specify the date to start and optionally end this schedule.

Begin running this schedule on:

Jul 17, 2019

Figure 18-30: More detailed scheduling options are available with Power BI Report Server

In the Scheduled Refresh section, you can see a list of configurations along with their status.

Figure 18-31 shows such a list, with just one entry named Daily Refresh.

SQLHelps Pvt Ltd

★ Favorites □ Browse

Manage Deposit

Home > Deposit > Manage > Scheduled Refresh

Properties

Data sources

Scheduled refresh

Row-level security

Security

+ New scheduled refresh plan + New from existing > Refresh Now □ Delete Search...

<input type="checkbox"/> Edit	Description ^	Schedule	Last run	Status
<input type="checkbox"/> Edit	Daily Refresh	At 2:00 AM every day, starting 7/18/2019		New Scheduled Refresh Plan

Figure 18-31: Monitoring scheduled refresh plans

Resources/ References

<https://powerbi.microsoft.com/en-us/documentation/powerbi-desktop-use-directquery/>

<https://docs.microsoft.com/en-us/power-bi/report-server/admin-handbook-overview>

<https://radacad.com/power-bi-report-server-power-bi-in-on-premises-world>

Summary

The Power BI Report Server is an on-premises reporting solution covering all aspects of report development, deployment and hosting. With the Power BI Report Server, interactive reports are now available on on-premises servers, not just on Power BI service.

Power BI Report Server needs a specific licensing, which comes either with the Power BI Premium or through the SQL Server Enterprise License with Software Assurance.

Power BI Report Server is a viable option for companies who are not yet ready to move to cloud-based technologies.

About the Author



Shree Khanal is a Database Architect, Speaker and Trainer having 15+ years of professional experience in database solution development and optimization. Mr. Khanal is a Data Platform MVP & Microsoft Certified Database Professional since 2000 to till date. Starting with SQL Server 7, he has been working with all the version of SQL Server up to 2017 in the production environment. In the context of OLAP cube development, he started working on it from SQL Server 2000 to latest SSAS 2017. He has worked several years providing SQL BI solutions to various enterprise clients most of them from the BI sector. Being a founder, he has been leading the Himalayan SQL Server User Group (www.sqlpassnepal.org) based in Kathmandu, Nepal since 2010.

Part VII: Architecture

Chapter 19: Governance

Author: Ásgeir Gunnarsson

The Business Dictionary defines governance as “Establishment of policies, and continuous monitoring of their proper implementation, by the members of the governing body of an organization.” [\[1\]](#) With this in mind, this chapter, in the context of Power BI, is going to focus on 4 pillars of governance, processes, training, monitoring and roles.

Introduction

Power BI, like many other self-service BI tools suffers for its dual purpose of being self-service but also used as an enterprise BI tool. Power BI started out as a pure self-service tool but has increasingly been moving to be more of an enterprise tool and can rightly be called a hybrid BI tool. No matter if you use Power BI as a self-service tool, as an enterprise BI tool or both, it's important to include governance into your implementation. Far too many organizations start using Power BI without thinking about governance and then have the problem of trying to get their users to stop doing things as they are used to and to start using process they are not used to and often feel will hinder their progress. No matter if your organization is starting its Power BI journey or has already ventured in the Power BI “Wild West”, governance is an important and necessary part of any Power BI implementation.

This chapter will focus on the 4 pillars of a good governance strategy:

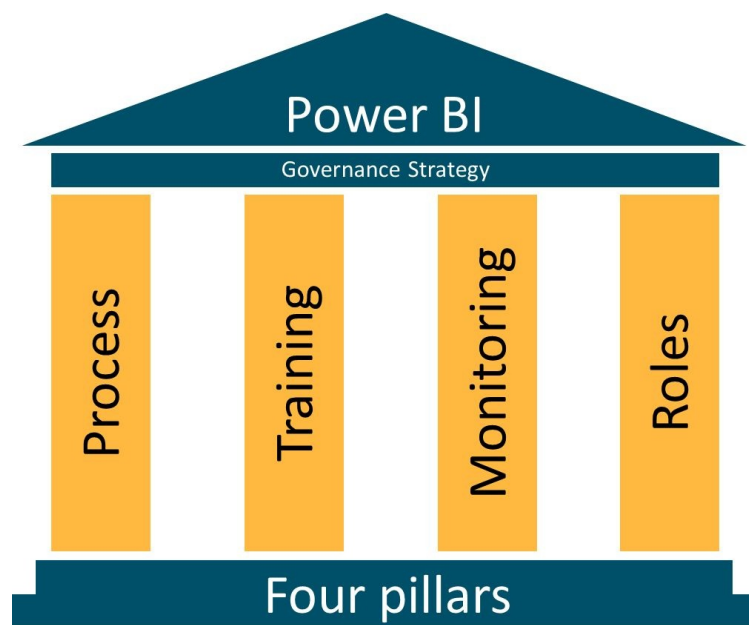


Figure 19-01: The Four Pillars of Power BI Governance Strategy

One of the following sections is devoted to each pillar.

Process

It's important to have a formal governance process in place. This process is often broken down into smaller processes and usually contains processes for Development, Publishing, Sharing, Security, Naming standards, Support and Tenant Settings. In this section we will see examples of each process and how they can be tied together and exposed to end users.

At the heart of a governance plan are processes. There can be many smaller processes or few bigger ones but without them there is not much governance.

Most often these processes describe how to work with Power BI and sometimes they describe how to support Power BI. It's vital that the processes are easily discoverable and are setup as a part of a whole so that users will know how each process ties into the whole governance strategy. One way is to have one master process document with links to all the process documents. Another way is to store all the process documents in the same library and categorize them so it's easy to navigate between them and they are logically grouped.

We will look at the seven most common (in the authors opinion) processes and see examples on what they might contain.

1-Development process

A development process most often describes how a report, datasets or both are developed. They describe where you develop the Power BI content and how you store and version your files.

Developing Power BI content

It is recommended to always develop Power BI content in Power BI Desktop rather than in the Power BI Service. The main reasons for developing in Power BI Desktop are the following:

Power BI Desktop has full functionality while the Power BI Service does not have full parity. The Power BI Service has very limited data manipulation although Dataflows for Power BI do have some. It's not possible to version the content in the Power BI Service (see next section)

Storing and versioning Power BI content

When you develop in Power BI Desktop you can store the original file in a secure place and version it. When creating reports in the Power BI Service you don't have a source file and cannot store a master copy of it. If someone changes the report or even deletes it, you cannot go back to the original. Another

advantage of using Power BI Desktop is that you can put the file into version or source control. This enables you to revert to older versions without storing multiple copies of the report under different names. One such place that many organizations use is OneDrive for Business which has built in version control. Many BI teams have source control systems they use but they are often cumbersome for business users that don't normally use such systems whereas OneDrive for Business is merely a folder on the computer if synchronized. Working on reports in Power BI Desktop also allows for the developer to have reports as work in progress without interfering with reports in the Power BI Service.

A development process will describe the above in detail, fitting to the needs of the organization

2-Publishing Process

The publishing process usually describes how to set up multiple environments and how to promote Power BI content between them.

Since the Power BI Service does not have built in functionality for multiple environments it's important to describe how that should be done (if relevant to your organization). The most common way this can be achieved is through the use of multiple workspaces. You would create one workspace for each environment and move the datasets and reports between them as they go between development stages such as development, test, pre-production and production. A Publishing process normally describes how this should be achieved and how you go about promoting datasets between environments. This can be done using simple publish in Power BI Desktop or more elaborately using the Power BI REST API. The process will normally reference the Naming Standards and Security processes for the workspace naming and access control.

3-Sharing Process

Generally speaking, there are four ways of sharing Power BI report or a dashboard with an end-user:

- Share the Power BI Desktop file
- Direct sharing of reports and dashboards
- Give access to a Power BI Workspace where the report and/or dashboard reside
- Share a Power BI Workspace App

Sharing a Power BI Desktop file is not recommended as the user will have full control over the report and can change whatever they want and can access all the data if the data is imported. The user can also make their own copy of the report and change things the original author does not have any control over.

Directly sharing a Power BI report or dashboard is a method that can be used in certain circumstances. It's not recommended unless the receivers are few or if you need to share one or few reports/dashboards out of a workspace with many reports/dashboards. The main drawbacks of direct sharing is that it is difficult to keep track of who has access and which reports/dashboards have been shared and which have not.

Giving access to a Power BI Workspace can be dangerous as the user can in most cases change or delete the report as well as access all the data in the report. With a recent change it is now possible to add users in a Viewer role in the new type of workspaces. I would still argue that only contributors should have access to a Power BI Workspace and the viewer role used for users that don't need editing rights but will still contribute such as tester. See next section 4-Security Process for more details.

The best way to share Power BI reports and dashboards is through Power BI Workspace Apps. The Workspace Apps are made to share to a larger audience and include functionalities such as custom navigation, multiple reports/dashboards at once and the ability to share the finished report in the app while the report is being developed in the workspace. Microsoft recommends using workspace apps to share Power BI content with end users.

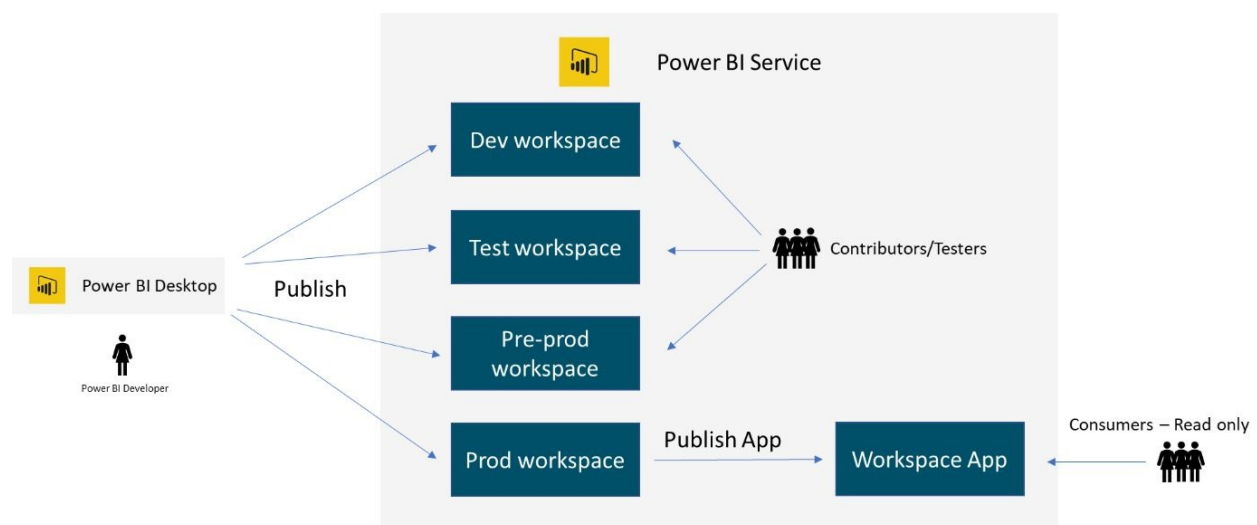


Figure 19-02: Publishing and Sharing in Power BI

The Sharing process describes how to share reports, dashboards and datasets and links to the security process for more details.

4-Security Process

A security process describes how to secure Power BI content. This is usually split into two categories: Object level security and data security.

Object level security

There are several types of objects that get created in the Power BI ecosystem and need to be secured. These objects are: Power BI Desktop files, Workspaces, Workspace apps, datasets, Power BI reports and dashboards in the service. Besides that, who can publish against certain data sources in the Power BI On-Premise Gateway needs to be decided.

It can be important not to store Power BI Desktop files where unauthorized people can get access to them. If the dataset uses import mode all the data is available to the person who has access to the Power BI Desktop file. The security process will normally point this out and advice on how to store the Power BI Desktop file.

As of when this is written everyone who has access to a workspace can see all the data, can change all reports and download a Power BI Desktop file of all reports in a Power BI Workspace. You therefore need to be careful of who get access to Power BI Workspaces. The golden rule is that only people that need to contribute to a report or dataset should have access to a workspace. In other words, if you need to WORK on the report or dataset you have access to a WORKspace. A security process will also describe if individuals, Active Directory Security groups or Active Directory Distribution List should be used to give access to workspaces.

The recommended way of sharing Power BI objects is via a Power BI Workspace App (see previous section 3-Sharing Process). Workspace apps can be considered a read only version of a workspace and as such will never give the user any modification rights. That said it's important to give only authorized people access to workspace apps. A security process will normally describe how to give access to workspace apps as well as if individuals, Active Directory Security groups or Active Directory Distribution List should be used to give access.

Sharing reports/dashboards directly with end users is not recommended in most

cases. If you do, you can keep track of who the report/dashboard has been shared with using the Share dialog window. A security process will describe how and when you should use direct sharing and how to (manually) monitor who has access.

Data security

Sometimes it's not enough to just control access on an object level. An example of this is when a single cost report serves the whole organization, but department managers can only see the data for their own departments. If that kind of security should be done using object level security the author would need to create as many copies of the report as there are departments. Instead it's more efficient to secure the data instead of the objects. Securing data in that way is often referred to as row-level security. Power BI allows for row-level security where the report author can do static or dynamic row level security allowing or denying access to whole tables or the content of certain columns. When to do data security and how to implement it is often dependent on other security and privacy processes as well as rules and regulation such as GDPR.

Another aspect of data security is tightly tied to development processes as it's best practice to connect to development versions of the source system until the report is ready and has been properly secured. The main reason for this is that development versions of source system don't usually have real up to date data in them thus preventing the real data from getting into the wrong hands. After the author has secured the data and objects as prescribed in the security process the data source is switched to other environments ending with the production environment when ready to be released to users.

Deciding how users get added to the Power BI On-Premise Gateway is very important. The tendency is to just add all developers to the needed data source in the gateway but from a governance perspective it's important to only add users that have been approved. Usually the process is that the data or data source owner needs to approve who can publish against the data source. This is often an excellent opportunity to engage the developer and make sure their reports/dashboards are following best practices and are approved.

A security process will describe how to secure access to objects as well as to data and will often refer to other security and process documents that exist within the organization.

5-Naming standard process

One of the most undervalued process is the Naming Standard process. Having this process early in the Power BI implementation will greatly improve the usability of the Power BI environment. Finding workspaces, reports, dashboards and datasets can be very tricky when you have hundreds of workspaces with no clear naming convention. When each project has development, test, pre-production and production workspaces (see 1-Development process) the number of workspaces can increase fast. Having a clear naming convention describing how to name workspaces as well as how to identify different environment is very important. Another common issue is that the user only sees certain amount of characters in the workspace name. Many users don't realize this and put the environment name at the end but when browsing the workspaces in the Power BI Service you sometimes don't see the end of the name and therefore need to hover over the workspace name to see which one is the dev or production workspace. A Naming Standard process will often be linked to a more general naming standard process within an organization.

6-Support process

Many organizations neglect to create a proper support organization when implementing Power BI (see Roles section). Having a good support process will enable your current support organization or dedicated Power BI support people to more easily assist users when needed. A support process will help non-Power BI supporters to know when to dig in and try to solve a problem and when to refer the problem to the report owner or Power BI Support people.

Typically, a support process will describe common scenarios like access requests, no data incidents, wrong data incidents and change requests to Power BI content and guide the supporter on how to react. Depending on how established your support process is, your support organization might do none, some or all of the support on Power BI content. If you have a support organization, they will get support requests on Power BI because users are used to getting support there. Therefore it's important that you integrate your Power BI support into your current support organization even though it's only for them to pass it on to the "real" Power BI supporters.

If your access is generally done via AD security groups or AD distribution lists a non-Power BI support organization will be able to support Power BI access requests as long as there is a separate support document for each Power BI Workspace App. This will greatly reduce the manual work by the Power BI developers or administrators and often reduce the time to get the request

resolved.

Your supports should also have a process section on how to handle dataset refresh failures and a way to redirect support requests to the Power BI Application owner or data owner if there is a data issue.

7-Tenant Settings process

There are several settings in the Power BI admin portal that are important when it comes to governance. Publish to web, Sharing outside of organization, Export data, Internal support page to name few are all very important for different reasons. It's very important that the organization has a process in place defining how each setting in the Power BI Admin portal should be set, who the setting should apply to and describe why it's important. Unfortunately, there is no way of automatically monitoring changes in the Admin portal which makes it even more important to have the settings well documented.

Training

If you want to have a successful Power BI implementation training is very important. You want to train everyone who touches Power BI but in a different way depending on their role. You want to make sure you get to everyone and deliver the right training based on their needs. In this section we will see examples of what categories of training you can use, the content of each training and the most effective delivery method. It's not only governance training that is important. Training users in properly using Power BI and using best practices will deliver value faster and will make report and dataset developers more compliant.

One of the things we will explore is how to automate the training offer to users by using Microsoft Flow in combination with Office 365 (who has license) and the Power BI audit log (what are they doing)

Training categories

The most common training categories are Consumer, Report Developer and Report and Dataset Developer. For each category there is a definition of who belongs to it as well as what training content is appropriate and how it should be delivered.

Consumer training

Consumers are normally defined as Power BI users than will only consume reports or dashboards created by others. They will never create their own or modify any content. This is normally the biggest group of Power BI users and is often overlooked when it comes to training. At the same time this group is the easiest to train.

The content of the consumer training is usually general training in navigating Power BI. How to log in and find content, how to open a report or dashboard, how to use the "toolbar" above a Power BI report/dashboard and the most common ways to interact with a Power BI report/dashboard such as navigating between pages, using buttons and bookmarks and using slicers and filters. The appropriate delivery method for the consumer training is videos or training manuals. Live training is not needed and normally too expensive, due to the number of people. The exception from this is if you want to train a subset of consumers on a particular report or dashboard. That is often best done with live classroom training as then training will also involve how to analyze the data.

Report Developer

A report developer is a person that will create Power BI reports on top of readymade datasets created by others. They don't create their own datasets but will use datasets such as Power BI datasets or SQL Server Analysis Services models.

The training for this group of people will often contain topics such as Visualization, Storytelling and Publishing and sharing. Below is an example from a real training course for report developers.

1. Introduction to Power BI
 - a. What is Power BI
 - b. Components of Power BI
2. Data visualization
 - a. Introduction to visualization best practices
 - b. The canvas and properties
 - c. Theming reports
 - d. Basic Charts and Visuals in Power BI
 - e. Working with visuals in Power BI Desktop
 - f. Custom Visuals in Power BI Desktop
 - g. Practice: Create report pages and theming them
3. Power BI Service
 - a. Introducing workspaces
 - b. Sharing options in Power BI Service
 - c. Dashboard vs. Report
 - d. Practice: Publish reports to My workspace and share it with a colleague
4. Advanced scenarios
 - a. Bookmarks and selection pane
 - b. Report page tooltip
 - c. Designing for Mobile devices
5. Introduction to Power BI development process
 - a. Where is the document
 - b. Content of the document
6. Governance
 - a. Where is the documentation
 - b. How do I make sure I'm compliant
 - c. Best practices and common sense

The report developer training is best done in a live classroom training but can be

done via live online if required. It's also possible to create an on-demand video course but it's the authors opinion that the users will get the fastest start and biggest benefit from a in person live classroom training.

Report and Dataset developers

Report and Dataset developers are the most advanced group of users that will both create their own datasets and reports.

The training for this group of users will often contain all the topics from the report developer course plus data topics such as Getting and cleaning data, Data modelling, Security, refreshing data and DAX. Below is an example from a real training course for report and dataset developers.

1. Introduction to Power BI
 - a. What is Power BI
 - b. Components of Power BI
2. Getting and cleaning data
 - a. Connecting to data
 - b. Direct/Live query vs. import
 - c. Query Editor
 - d. Transformation GUI
 - e. Practice: Getting data from a database, Excel file, text file and a webpage
3. Data Modelling
 - a. Why is data modelling important?
 - b. Relationships in Power BI
 - c. Hierarchies, formatting and hiding columns
 - d. Sorting by other columns
 - e. Date Table
 - f. Practice: Create relationships and data modelling
4. DAX
 - a. Introduction to DAX
 - b. Calculated Columns and Measures
 - c. Practice: Creating calculated columns and measures
5. Data visualization
 - a. Introduction to visualization best practices
 - b. The canvas and properties
 - c. Theming reports
 - d. Basic Charts and Visuals in Power BI

- e. Working with visuals in Power BI Desktop
- f. Custom Visuals in Power BI Desktop
- g. Practice: Create report pages and theming them
- 6. Power BI Service
 - a. Introducing workspaces
 - b. Sharing options in Power BI Service
 - c. Dashboard vs. Report
 - d. Row Level Security
 - e. Schedule Refresh vs. Other types of connections
 - f. Gateway's Role in the Service
 - g. Practice: Publish reports to app workspace and share it as a workspace app with a colleague
- 7. Advanced scenarios
 - a. Bookmarks and selection pane
 - b. Report page tooltip
 - c. Designing for Mobile devices
- 8. Governance
 - a. Where is the documentation
 - b. How do I make sure I'm compliant
 - c. Best practices and common sense

The report and dataset developer training is best done in a live classroom training. It's also possible to create an on-demand video course but it's the author's opinion that the users will get the fastest start and biggest benefit from a in person live classroom training. In the experience of the author there is usually need for a person in the room to help people out as the dataset part of the training is often quite tricky.

Who to train and how to prioritize

One of the things the author has seen with many organizations is that they struggle to know who to train and how to prioritize when to train them. Here is a suggested method of discovering who to train and how to prioritize their training.

The first issue, who to train, can be automated in part. First of all, you can discover who has gotten a Power BI license (Pro or free) and send them an invitation to go through the consumer training, which is in the form of online videos or training manuals, so the users can do it in their own time. To discover who has a license one could build a call to the Office 365 API that has a method

which will return everyone with the type of license (Power BI) the query filters on. This needs to be stored somewhere and then the next time the call is made a comparison is made to the existing dataset and those that are not in the existing dataset are new licenses and should be invited to the training. The whole thing can be automated using Microsoft Flow or Azure Logic Apps. To decide if people need report developer or report and dataset developer training it is possible, in a similar way, to gather data from the Power BI Audit log in Office 365. The Audit log contains information about all Power BI interactions including creation of content. By looking at who is already creating content it can be used to prioritize training as those who are already creating content should get training straight away, especially on the governance.

Monitoring

One of the cornerstones of governance is monitoring. Monitoring what users are doing and monitoring what users are creating. From a governance perspective monitoring creation, access, usage, changes, deletion and data exports are the most important.

Artefact inventory

The Power BI Rest API can tell you what artefacts exists and who has access to what. Besides that, the Rest API has powerful administration endpoints that allow you to get information about various administration objects as well as allow you to perform admin tasks. To access the Power BI Rest API, you can either create your own web application and call the API or you can use PowerShell to call it. Microsoft has put some effort into wrapping many of the endpoint in the API into PowerShell cmdlets and they have also created a cmdlet to wrap the call to the Rest API. You can read more about the PowerShell cmdlets at <https://docs.microsoft.com/en-us/powershell/power-bi/overview?view=powerbi-ps> and you can read more about the Power BI Rest API at <https://docs.microsoft.com/en-us/rest/api/power-bi/>. The user will have access to some of the endpoints through normal Power BI workspace access, but a lot of the endpoints require the user to be a Power BI administrator, at least if they want tenant level information.

Monitoring usage

The Power BI Audit log can tell you who accessed what and who changed or deleted what. The audit log is part of the Office 365 Security and Compliance Centre. The Power BI audit log is turned off by default but can be turned on in the Power BI Admin Portal. To get access to the audit log in the Office 365 Security and Compliance Centre you need to have the View-Only Audit Logs or Audit Logs role in Exchange Online or be an Office 365 admin. It is possible to fetch data from the audit log in two ways. One is to log into the Office 365 Security and Compliance Centre, run the log query and either view the results on the screen or download the results as a CSV file. Another way is to use the Office 365 Rest API which is the preferred way if you want to automate the collection of the log information. Note that the log is only stored in the Office 365 Security and Compliance Centre for 90 days so if you want to keep it for a longer time you will need to collect it and store it in a different place such as data warehouse. More information about the audit log and how to collect the data can be found here: <https://docs.microsoft.com/en-us/power-bi/service->

[admin-auditing](#)

The author recommends that both the audit log and artefact inventory is collected and stored in a database. Partly because of governance issues as described before but partly because there is valuable information in there about adoption, development over time and user behavior which could be beneficial for the organization at a later time.

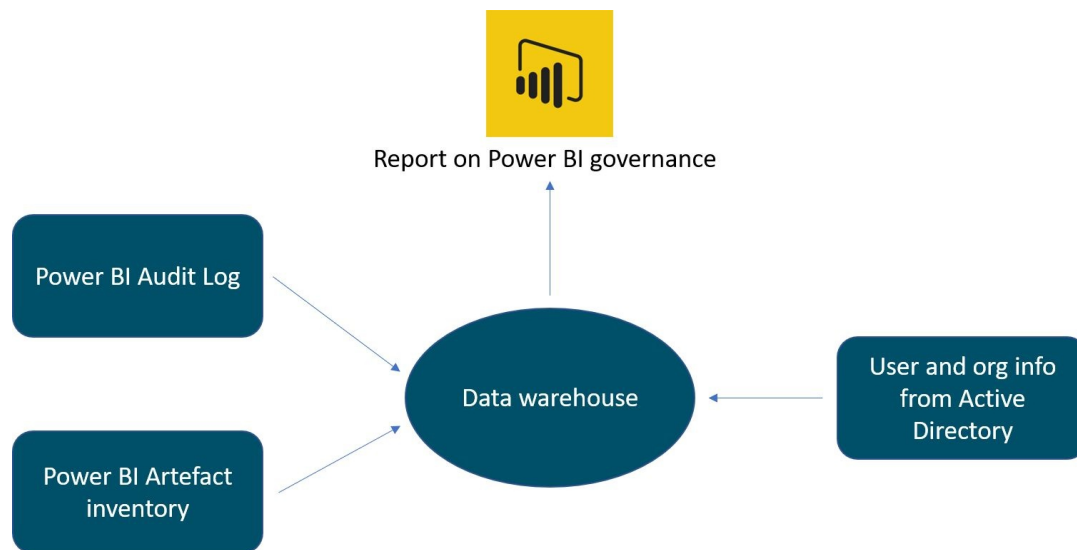


Figure 19-03: Monitoring Power BI and reporting the results

At the time of this writing there are 91 events that are monitored in the Power BI audit log. If your organization does not want to store all that data you should consider taking all events that are about viewing, editing (including deleting) and exporting. When you have started the collection of the data you might want to join it to further information from the artefact inventory discussed in the previous section as well as information about the organization employees and the organization structure.

Monitoring the Power BI On-Premise Gateway

The Power BI On-Premise Gateway is a Windows service running on an on-premise server. The gateway needs to be monitored as other Windows services. The main things to monitor are the service uptime and server performance. Normally monitoring is in the hands of an infrastructure team (if one exists).

Roles

To be successful with a Power BI implantation in the long run it's important to have well defined roles. This is most likely different from organization to organization and in some cases the same person might have more than one role. The most common roles are Power BI Administrator, Power BI Gateway Administrator, Data steward, Power BI Auditor and Power BI Supporter(s). In this section we will look at each role, it's responsibilities and what type of person should have this role.

Power BI Administrator

The Power BI Administrator has two main responsibilities, Power BI tenant settings and capacity administration if the organization has Power BI Premium.

Power BI tenant settings, set in the Power BI Admin portal, are very important when it comes to governance. The Power BI Administrator can change all settings within the Power BI Admin portal and therefore has the power to allow for all, allow for certain groups or deny users access to certain functionality.

A Power BI administrator can also administer Power BI Premium capacities and assign workspaces to those capacities.

Far too often the Power BI administrator is the first developer that used Power BI within an organization or the "best" developer. We often refer to these administrators as accidental administrators. In some instances, this is fine, but the preferred way is to look for the best suited person that can fulfil the role within the organization. Typically, this person knows Power BI and it's terms but has strong understanding of governance principles and the importance of the Power BI tenant settings. This is usually not a fulltime role, but some small allocation is needed if the person is to be able to do a good job.

Power BI Gateway Administrator

The Power BI Gateway Administrator is responsible for ensuring the Power BI On-Premise Gateway is running, updated and has its performance monitored. The administrator also has the encryption keys to the gateway.

The Power BI On-premise Gateway is a central component in the Power BI ecosystem and requires someone to administer it. The person might be part of an infrastructure team as the tasks are mostly about monitoring and updating which is often an integrated part of infrastructure teams.

Data steward

The data steward role is not Power BI specific but usually focused on master data. The reason it mentioned here is that it's often a central part of a governance strategy as master data is a very important part of it. Due to it not being Power BI specific I won't get into any specifics about the role but instead leave it up to the reader to gather information about it.

Power BI Auditor

The Power BI Auditor is the one responsible for monitoring the Power BI Audit log in Office 365. The main responsibilities of the role are to make sure the audit data is gathered and stored and most importantly that Power BI users are acting in accordance with the organization's governance processes. This means reporting on top of the audit data and flagging non-compliance actions. They will often make parts of the audit log or reports on top of the audit log available to other people within the organization so they can fulfil their role. The data will often be anonymized when made available to others.

This role is often in the hands of an internal auditing function, governance team or security team.

Power BI Supporter

For a successful Power BI implementation there is a great need for someone to support Power BI developers. The main reason for this is that as Power BI is a self-service tool, developers are often business people with different backgrounds that are not necessarily used to developing integrations, data models or visualizations. Sadly, the author finds this role is often overlooked or the ones that are appointed to the role are supposed to do that AND their normal full-time job. This diminishes the success of a Power BI implementation.

The main responsibilities of the Power BI Supporter vary but often are to assist users with Power BI related problems, assure best practices are followed, champion new functionalities and generally be the go-to person for the business.

The person(s) selected for this role are often the internal super users or part of the business intelligence function in the organization. The most important thing, in the opinion of the author, is that with the role comes allocated time because without it, the quality of the organizations Power BI work might suffer.

Summary

This chapter suggest that a good Power BI governance strategy has 4 pillars, Processes, Training, Monitoring and Roles. Organizations need to define processes so that their users do Power BI right, train them to follow the processes as well as best practices when it comes to Power BI, Monitor the Power BI environment and have defined roles and responsibilities. Each pillar has equal importance and for a successful Power BI implantation you want to make sure you think about them all.

About the Author



Ásgeir is a Data Platform MVP and Chief Consultant at Datheos. He works on Business Intelligence solutions using the whole of the MS BI stack. Ásgeir has been working in BI since 2007 both as a consultant and internal employee. Before turning to BI Ásgeir worked as a technical trainer and currently teaches BI courses at the Continuing Education Department of the University of Iceland.

Ásgeir speaks regularly at events both domestic and internationally and is the group leader of the Icelandic PASS Group as well as the Icelandic Power BI user group.

Ásgeir is passionate about data and loves solving problem with BI

Chapter 20: Architecture of a Power BI Solution in an Enterprise Environment

Author: Dr Greg Low, SQL Down Under

Most of my consulting work is in large enterprises, particularly large financial organizations like superannuation (retirement fund) companies and banks. These organizations often ask me for a “big picture” of how Power BI should integrate with an enterprise architecture. Time and again, what they really are asking me for is how to do the following:

- Keep core organizational data on-premises
- Provide secure access to reports and dashboards in Power BI
- Control who sees which parts of the data

In this chapter, I describe how to do this.

The Big Picture

They say that a picture paints a thousand words, so let's start with a picture of the typical outcome that I'm looking to achieve:

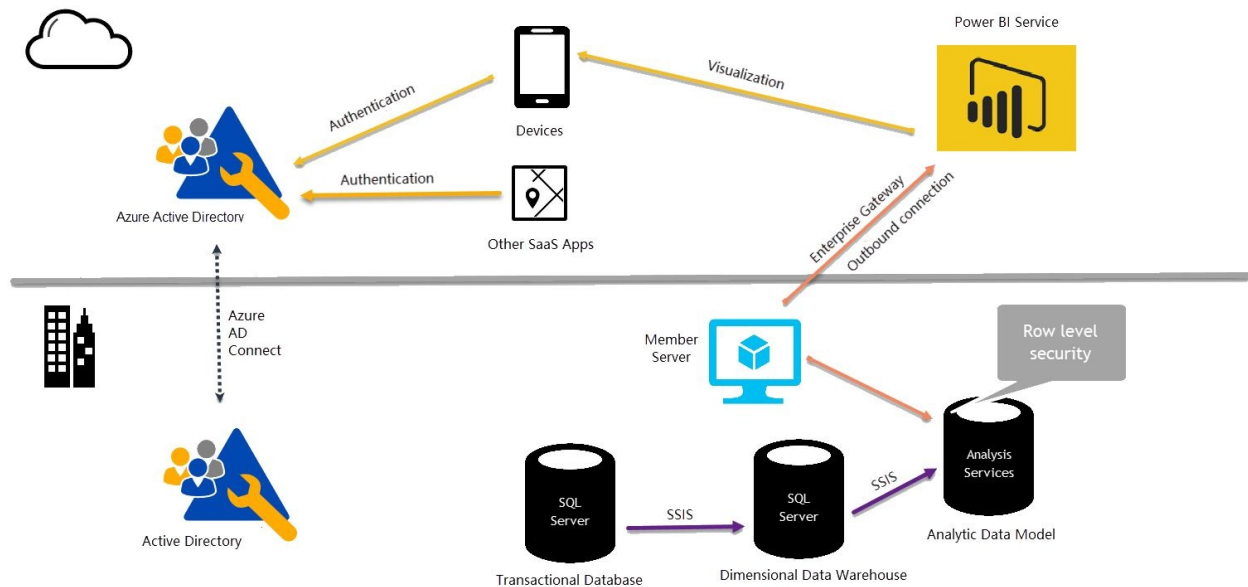


Figure 20-01: Big picture of an enterprise architecture involving Power BI

The diagram in Figure 20-01 shows the cloud-based infrastructure above the dividing gray line, and the on-premises infrastructure below that line. Let's now look at each of the key sections of this diagram separately.

Identity Management

I always tell customers that identity management is the most important aspect of planning this type of architecture. Figure 20-02 shows the core components that are involved.

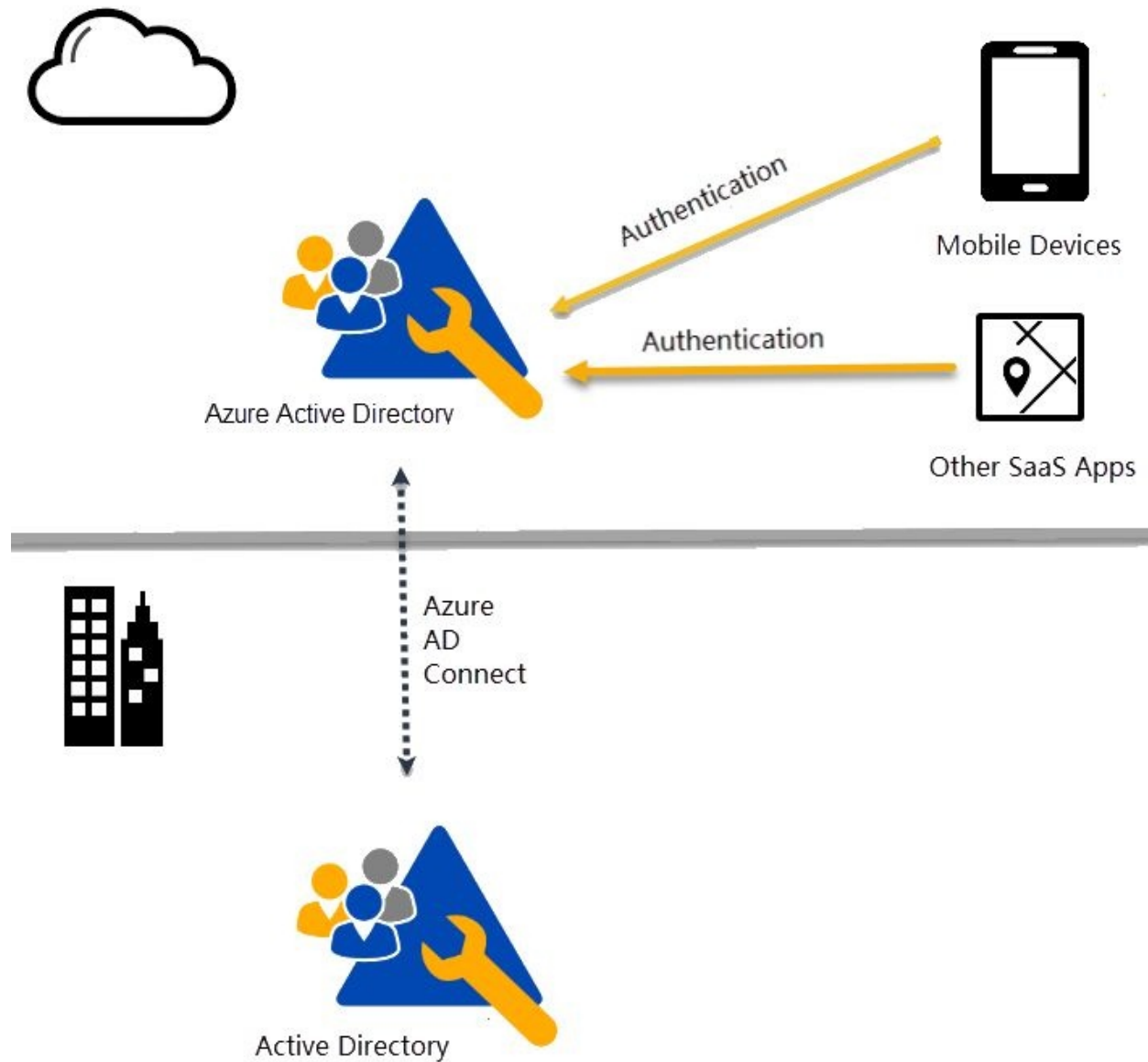


Figure 20-02: Core components for Hybrid Identity
On-Premises Active Directory

Organizations that use Windows-based PCs today, generally use Windows

Active Directory to provide identity management.

Windows Active Directory (AD) provides several services:

- ADDS - Active Directory Domain Services
- AD LDS - Active Directory Lightweight Directory Services
- ADFS - Active Directory Federation Services
- ADCS - Active Directory Certificate Services
- ADRMS - Active Directory Rights Management Services

Credential Spread Issues

Most enterprises will no doubt move into scenarios that involve cloud-based systems and require cloud-based authentication. If this process is not managed well, it quickly leads to a problem often referred to as YAUP (yet another username and password) where overall identity management becomes a significant mess.

Even if tooling like Office 365 is not on the immediate horizon, early investment in hybrid identity provision via AAD is important, so that this identity spread can be avoided right from the start. Later, Office 365 and other tools can then utilize the same hybrid identity configurations.

Hybrid Identity

Azure Active Directory (AAD) provides a cloud-based identity service that can help in avoiding these issues. Importantly, AAD can be directly integrated with the existing on-premises AD, to avoid users needing to have separate identities and passwords for the cloud services that are being used.

It is important to note that this hybrid identity can also be used in a wide variety

of other applications. I checked the marketplace gallery today and over 2800 applications are advertised as directly supporting AAD for authentication. These can be seen here:

<https://azuremarketplace.microsoft.com/en-us/marketplace/apps/category/azure-active-directory-apps>

As well as the standard Microsoft offerings like Office 365, SharePoint Online, Exchange Online, Lync Online, etc. all of which use AAD for identity, many organizations are adopting best-of-breed applications for different aspects of their businesses. They might use Zoom for meetings, ZenDesk for a helpdesk, or Drip for an email marketing engine, DropBox for cloud-based storage of shared documents, etc. These applications can also use this same form of hybrid identity that is provided by AAD.

Integration Applications

One case where this form of identity management is crucial is where integration applications are used. For example, a retirement fund might decide that when a new member is added to their core systems, that a new entry is made in Salesforce, and when that happens, a new email sequence is enabled in Drip, and so on. Often, applications like Zapier or Microsoft Flow are used to provide the logic behind these integrations. Once again, having a single form of identity across all these applications is highly desirable.

Azure Active Directory Offerings

AAD currently provides three main services:

- AAD (Azure Active Directory) – this provides identity services - exposing a graph API along with OAuth, SAML-P, WS-FEDERATION, and more
- AACS (Azure Access Control Services) – this federates identities from

external providers – generally on-premises AD but could also potentially include Google, Yahoo, Facebook, which could be important if member access is ever provided to resources

- AADDS (Azure Active Directory Domain Services) – this is basically a domain controller as a service. It is a scalable, high-performance, managed domain service for domain-join, LDAP, Kerberos, Windows Integrated authentication, and group policy

A major advantage of AAD is the ease of enabling options like multi-factor authentication (MFA), and cloud-based password reset. The implementation of these options alone often brings significant savings to IT support teams.

An added advantage is that member/customer/partner identities can also be stored and managed in the directory tenant.

Enhanced Security on Terminations

In terms of overall security, it is worth noting that this approach to identity also provides a single point where access can then be removed when required. If an employee leaves, and their identity is disabled in the local AD, the federation and hybrid identity means they will also no longer have access to any of these associated applications.

If separate identities had been used, disabling even a single employee can become an onerous task, and one that potentially could leave gaps open after the employee has been terminated, until they are all cleaned up.

Implementing Hybrid Identity

Azure AD Connect is the synchronization mechanism for on-premises AD and cloud-based AAD. It provides a choice of:

- Password hash synchronization (PHS)
- Pass-through authentication (PTA)
- Federation (AD FS)

My usual recommendation is to implement Federation as it provides the highest level of interoperability and ensures that not even hashed passwords are sent to the cloud.

Application Integration

It is likely that many new applications will be cloud-based. Integrating these with AAD is now straightforward.

For .NET development, developers can include the WIF (Windows Identity Framework) in their projects. The application is registered with AAD to create a Service Principal (which includes an identifier called an AppPrincipalID, and a secret).

When logon to an application starts, control is transferred to AAD for logon. AAD eventually calls the application back and passes a signed token that is then used to verify the user.

Authentication for External Users

Azure AD has two forms of authentication for external users:

- Azure AD B2B (business to business) is used to assign permissions on internal resources to external users. It is generally used with partner organizations and allows collaboration with internal users. The pricing structure for Azure AD currently allows a 1:5 ratio for internal users to B2B partner users.
- Azure AD B2C (business to consumer/customer) is used to manage authentication to corporate applications by external users. It is often used for customers or members who need access to corporate websites. These users do not need to be part of the internal domain's user list. There is no practical limit to the number of these external users that can be attached. The pricing is based upon the number of authentications performed each month. The first 50,000 authentications are currently free.

B2B allows the use of MFA (multi-factor authentication) if the Azure AD tier is one that supports it.

B2C allows the use of MFA at a very low additional cost.

Password-less Implementation

I have mentioned this topic for completeness although it might be a component of a longer-term strategy. Passwords are fast becoming considered as inappropriate for authentication. Many industry leaders consider password authentication as fundamentally broken.

The costs associated with using passwords are fast outweighing the benefits. Although many organizations have strict password policies, even the strongest passwords are easily phishable.

For enterprise IT departments, password and identity related support and maintenance is expensive. And as it becomes common for IT teams to require ever stronger password complexity and demanding more frequent password changes, the costs rise ever more quickly.

Details on approaches to password-less implementations can be found here:

<https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RE2KEup>

On-Premises Source Data

The data that Power BI operates can come from on-premises data sources. My preference is for the following structure, as shown in Figure 20-03.

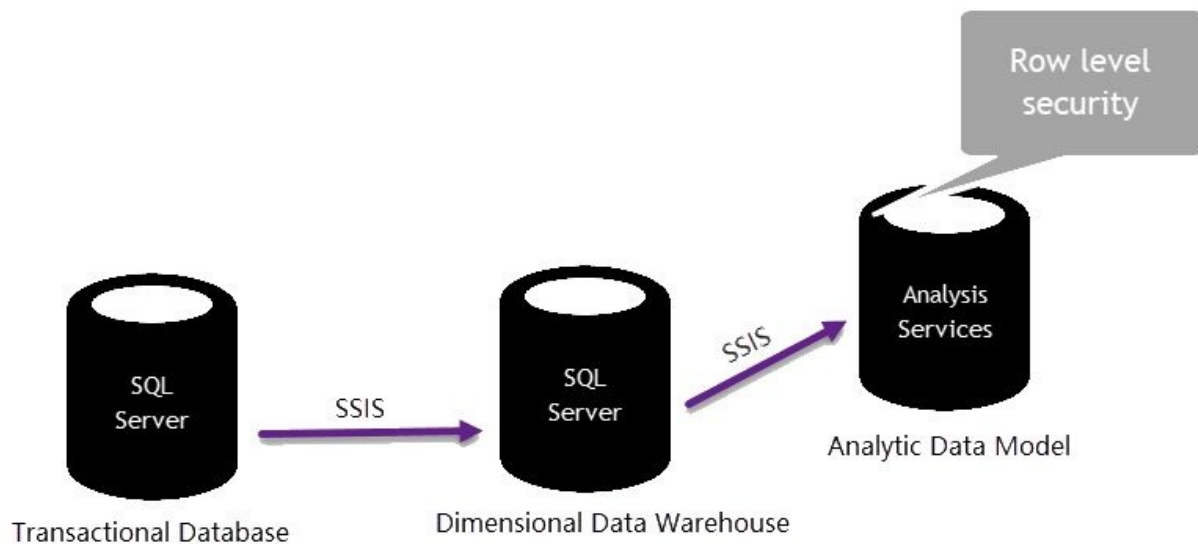


Figure 20-03: On-premises databases

Dimensional Data Models

Reporting directly from source systems (particularly for strategic reporting) is generally undesirable for several reasons such as:

- Transactional source systems are designed to support their applications, and are generally highly normalised
- The relationships between the tables are often complex and complex joins are required in queries
- Names of columns and tables are not directly usable in reports and charts
- Most source systems do not support versioning – they only show a single current state of the data
- Reporting can place a significant processing load on the source systems

Instead, we recommend the creation of a dimensional data warehouse that, while it is still a relational database in SQL Server, is structured to support analysis and reporting rather than transactional processing.

Some key advantages of this type of model are:

- Provides a single cleaned up model of the data
- Avoids the need for data cleansing in all reports and analytics by centralising it
- Has simplified (star schema based) relationships
- Directly supports both reporting and analytics
- Easy for end users to navigate when required
- Names are directly reusable in reports, charts, and dashboards
- Can support versioning if needed
- Can provide a single view of data from multiple source systems

Customers often want to build all reporting against the source system data, but this is not appropriate. It generally impacts the system too greatly; the source system data is usually not in a form that is designed for reporting and analytics; it is common to need to combine data from multiple source systems; and it is also often useful to add versioning of the data that is not provided by the source systems.

For example, if I have customers allocated to sales territories, and I rearrange the territories, what happens when I view last year's sales report? Does all the data automatically jump into the new territories? I might want that, but more commonly, I want the data to still look the same as it did last year. Versioning can help to fix that.

One of the challenges with any BI implementation though, is that if a separate dimensional data warehouse is created, latency will be involved in the transfer of source system data across to the data warehouse.

It's important to separate strategic reporting from operational reporting.

I recommend using SQL Server Reporting Services (SSRS), connected directly to the transactional database, for operational reporting that must be up to date. For reporting that is more strategic in nature, and can generally tolerate some small latency, I recommend using Power BI, Excel, and SSRS, with all of these supported by SQL Server Analysis Services (SSAS).

Strategic reporting is best delivered from an analytic data model (using SSAS), that was constructed over a dimensional data model in a separate SQL Server database. I know that you'll read information that tells you that you can just connect Power BI or other dashboarding tools, directly to your transactional databases, without worrying about creating a dimensional data model first. And while that's strictly true, it's not how you get a great outcome.

Transactional systems make poor direct data sources for analytic systems and for most dashboards. They also rarely hold versioned data. A dimensional data model can help with that.

Staging and Cleansing Data

Your reporting and analytics will need data from source systems. I recommend implementing staging databases to be used for import and cleansing of source system data that is needed for reporting and analytics. Generally, I try to achieve this while meeting the following aims:

- Minimising the impact on the source system
- Not missing any data
- Not duplicating any data
- Cleansing the data before reuse

Wherever possible, I try to import data incrementally. This is possible for most systems, but I do come across tables where this is not possible, and I need to retrieve entire tables. But I aim to do the data retrieval in a way that avoids excessive load on the source system.

With incremental loading, it is critical to avoid missing data, and duplication of

data.

Source system data generally also needs to be cleansed i.e. correcting/aligning data values, supplying missing values, correcting invalid values, fixing relationships between data elements.

I use staging databases in the dimensional data warehouse to hold the data where this work is being performed.

Analytic Data Models

SSAS allows you to use advanced mashup and modelling features to combine data from multiple data sources, define metrics, and secure your data in a single, trusted tabular semantic data model.

The data model provides an easy and fast way for users to browse large amounts of data for ad hoc analysis. Queries on the data model will often be dramatically faster than the equivalent queries run directly on the dimensional data warehouse.

Power BI, Excel, and other applications can be used as the visualisation layer for the data in SSAS.

SSAS provides the ability to create a semantic layer above your dimensional data model, by creating an analytic data model. There are two flavors of SSAS: Multi-dimensional and Tabular. At this point, I would almost always recommend that you use a tabular data model. There are a handful of exceptions, but you should just start with a tabular data model in SSAS.

Power BI can support multi-dimensional models somewhat, but it works cleanly and efficiently with tabular data models because that's basically the same data model that it uses internally within Power BI.

As well as loading tables from your underlying dimensional database, the analytic model allows you to create:

- Relationships between the tables
- Control over query filtering directions
- Computed columns to ease navigation and simplify the data for the user
- Measures such as totals or averages over time

Importantly, an analytic data model in SSAS can also implement RLS (row level security) based on the users and groups in Active Directory. RLS lets you make sure, for example, that only New Zealand users see New Zealand data, Queensland users see Queensland data, and so on.

While it is possible to create row level security within the Power BI service, it will be tied to identities within Azure Active Directory.

Moving the data between databases

If I've convinced you that you need a dimensional database, and an analytic data model between your source systems and Power BI, you also need to be able to move data around. The tool that I typically use for this is SSIS (SQL Server Integration Services).

SQL Server Integration Services (SSIS) provides graphical tooling for the creation of enterprise-level data integration and data transformations solutions that involve SQL Server. In fact, it can be used for even broader purposes than SQL Server-related integration. It can copy or download files, load data warehouses, cleanse and mine data, and manage SQL Server objects and data.

SSIS can work with a wide variety of data sources such as XML data files, flat files, and relational data sources, and then load the data into one or more destinations. It includes a rich set of built-in tasks and transformations, graphical tools for building packages, and its own Catalog database, where you store, run, and manage packages.

Most SQL Server installations utilize SSIS for a wide variety of scheduled data movement tasks. The packages that are created with it can be scheduled by either the inbuilt SQL Server Agent, or by external schedulers such as Control-M.

Enterprises can use SSIS for controlling the cleansing of data in the staging databases, for loading data into the dimensional data model, and for processing (i.e. reloading) the analytic data model.

Paginated Reporting

Apart from Excel, SSRS is the most popular Microsoft BI tool. Since 2016, the server aspects of SSRS are shipped in two forms:

- Report Server (RS)
- Power BI Report Server (PBIRS)

Paginated reports can also now be deployed to the Power BI service.

I recommend using Reporting Services reports for both operational and strategic paginated reporting.

SSRS provides the ability to create, publish, and manage reports, then deliver them in several ways such as in a web browser, embedded within an internal application, on a mobile device, or emailed to the user. Reports can be viewed on-demand, or they can be scheduled.

The web portal can be branded to provide an organizational look and feel.

PBIRS is a superset of RS but also allows deploying some Power BI reports directly to an on-premises server. PBIRS is licensed by either SQL Server Enterprise with SA, or via Power BI Premium. Most organizations have SQL Server Enterprise Edition with SA, so it is possible to take advantage of the PBIRS based version. Generally, though, I recommend that Power BI reports are delivered from the Power BI service instead.

Providing Power BI Access to the On-Premises Data

Power BI needs access to the on-premises data sources. It does that by using the Enterprise Gateway. I recommend that enterprises implement the Microsoft Enterprise Gateway to support development of applications such as Power BI, Power Apps, Microsoft Flow, and others. The gateway is shown in Figure 20-04.

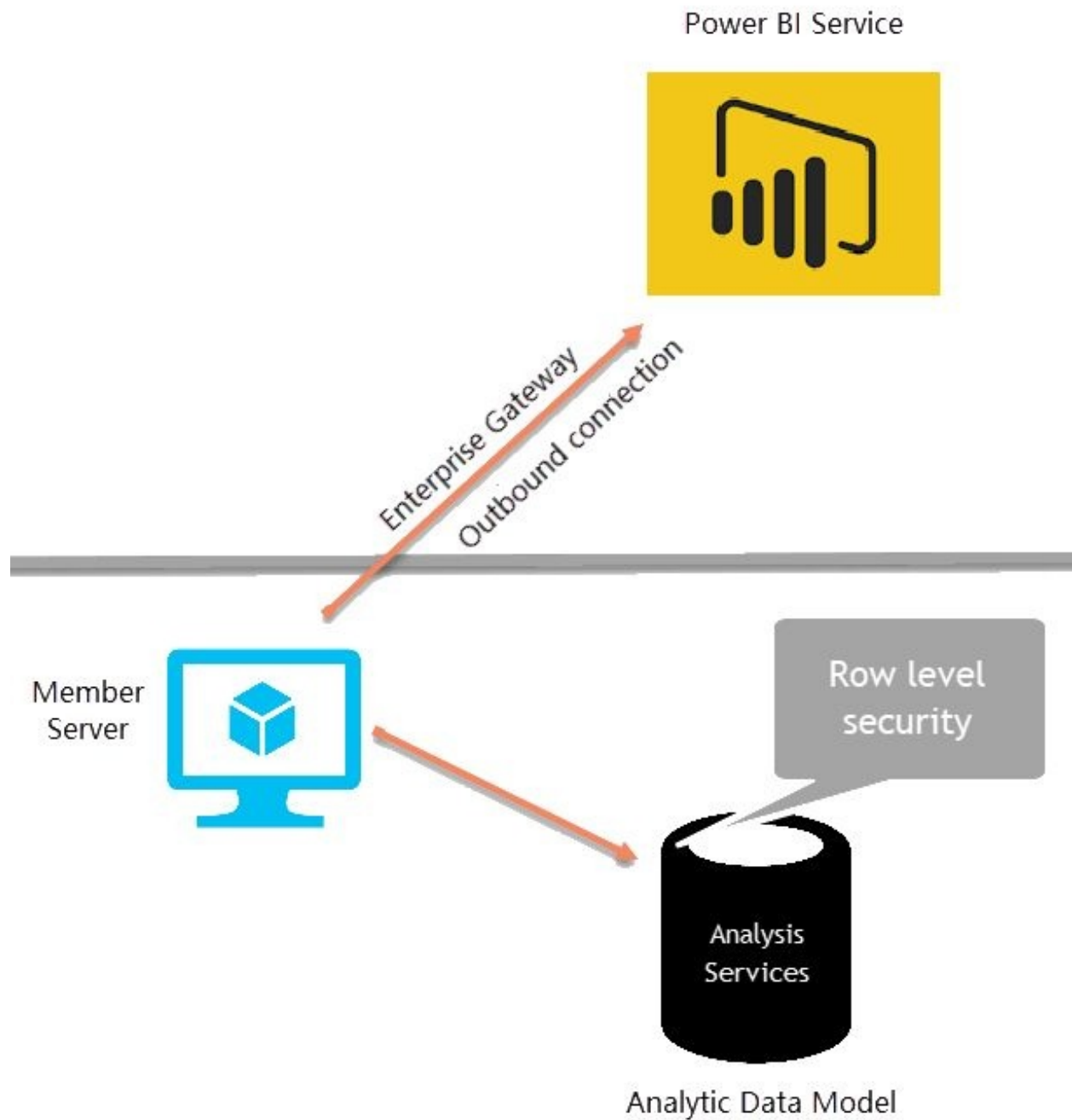


Figure 20-04: Enterprise Gateway providing access to on-premises data

Enterprise Gateway

Tools such as Flow, Power BI, Power Apps, etc. need the Microsoft Enterprise Gateway to be deployed before they can access any on-premises source data. It should be deployed before any of those applications or related applications

require it. This will be needed when starting to test or use Power BI based reports and dashboards, and if Microsoft Flow becomes part of the integration solution.

The Enterprise Gateway provides a secure communication service. It avoids the need for applications to connect into the on-premises systems, and for incoming ports in the corporate firewall. The gateway connects outbound to Microsoft infrastructure, and performs certificate exchanges for authentication. Secrets held by the Microsoft service are encrypted by a key that is supplied by the local gateway via the certificate exchange.

The gateway can be installed on almost any server in the network. Although it can be installed (and commonly is installed) on the same server as SQL Server Analysis Services or the SQL Server database engine, it does not need to be. If you were using import mode instead of Live Query mode, it would be better supported on a dedicated server.

In this case though, when it connects to SSAS using Live Query mode, it impersonates the user logged into Power BI.

If, for some reason, the domain in Azure (and Power BI) is different to the on-premises domain, and directory sync isn't being used, the gateway also provides the ability to map the UPNs (user principal names) so that users in the Azure AD domain can be mapped to users in the on-premises domain.

You can read more about the Enterprise Gateway in the following article:

<https://docs.microsoft.com/en-us/power-bi/service-gateway-onprem>

Power BI Service for Dashboards

I recommend implementing Power BI for the deployment of dashboards and analytic reports. It can support both users within the corporate network and mobile users, as shown in Figure 20-05.

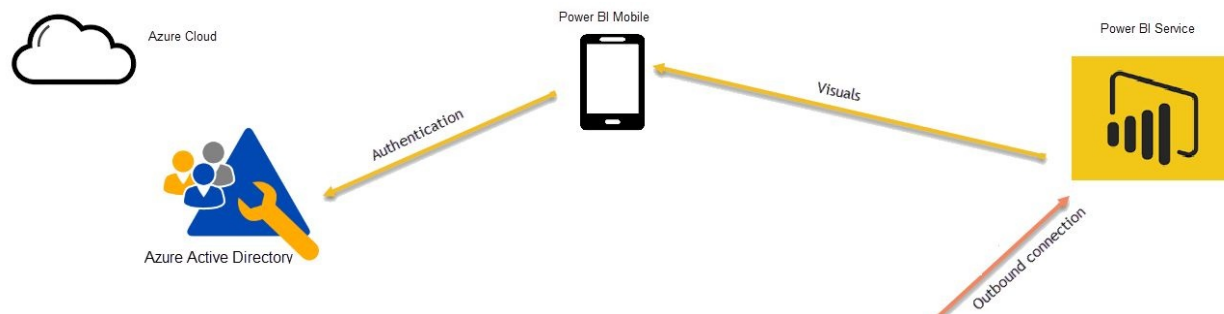


Figure 20-05: Authenticating mobile Power BI to Azure AD

Power BI Service

Power BI is one of the most popular applications that Microsoft has ever released. It is made up of several components:

- Power BI Service is an online (cloud-based) service that is used to provide dashboards and analytic reports to users. Data for the reports can be sourced from a very wide range of services, applications, and locations.
- While users could connect directly to the Power BI website, many will choose to use native applications to access this information. There are downloadable native Power BI applications (sometimes called Power BI Mobile) for Windows 8 and later (from the Microsoft Store), for iOS (from the Apple Store), and for Android (from the Google Play Store). It is also possible to embed Power BI directly into other applications with minimal code changes.
- Power BI Desktop is a free tool that can be used to create datasets, and reports based on those datasets. It can be used as a standalone “thick” client, or more commonly, these datasets and reports are published to the Power BI Service for consumption.
- Power BI Report Builder is a tool for building paginated reports to be deployed to the Power BI Service. This capability only works on Premium editions of the Power BI Service.

Power BI uses the hybrid AD (that I have also recommended) for authentication.

Power BI Workspaces

The Power BI Service has a concept of workspaces. These are collections of related Power BI objects. Workspaces are a security boundary. I generally use them for two purposes:

- To separate items being worked on by development teams
- As separate environments for deployment (i.e. UAT, Staging, Production)

Connecting Other Applications

Tableau

Many organizations already have an existing investment in Tableau reporting. Generally, I find that companies migrate from Tableau to Power BI when they have the opportunity, based on licensing. However, while it would be possible to continue to use Tableau to connect to its current data sources (often direct connections to source systems), if use of Tableau is continued, it is likely that it should move to using the recommended dimensional or analytic data models as a data source, or once the XMLA endpoints for the Power BI service are fully released, Tableau could connect to those interfaces.

Excel

Most organisations have many people who spend a great deal of time using Excel and are very familiar with it. Excel is a very capable tool and it could be used for further reporting and analytics via:

- Connection to the dimensional data model
- Direct connection to the analytic data model in SSAS (Excel has very capable options for doing this)

Summary

Power BI can be a key part of an enterprise BI strategy that keeps sensitive data on-premises while still providing secure access to dashboards.

About the Author



Greg is one of the better-known data consultants in the world. He is a long-term Microsoft Data Platform MVP, and a member of the Microsoft RD (Regional Director) program. Microsoft describe the RD program as consisting of “150 of the world's top technology visionaries chosen specifically for their proven cross-platform expertise, community leadership, and commitment to business results”.

Greg heads up a boutique data consultancy called SQL Down Under in Australia. He is best known for his SQL Down Under podcast, SDU Tools, and his online training school. Greg regularly presents at conferences world-wide.

Chapter 21: A Power BI-only Solution for Small Organizations

Author: Author: Gogula Aryalingam

Power BI plays many roles when it comes to analytics: A visualization and dashboarding tool in an enterprise solution, a prototyping tool to evaluate a solution to a business problem, a data profiling tool before undertaking an ETL implementation, and more. Now, there is this universe where organizational budget overrules many things; mainly information technology spending, including analytics. There are also other reasons, such as end users not seeing the entire picture of how analytics will help the organization. Hence, there are many cases where business intelligence is required to be implemented on a tight budget but showcasing quick value. This is where Power BI shines. In this chapter, I will talk about how, using just Power BI, you could design a complete business intelligence solution. Of course, there will be limitations and workarounds, but that is expected.

Background

Traditionally, a data warehouse has been the central component around which business intelligence is built. It worked on the principle of *the more data you bring, the more analytics you get*. Of course, data had to be of quality and formulated and structured to suit the business before any analysis was done. At present, however, analytics and the way analytics is done has changed immensely. There are various types of analytics that can be performed, which is enabled by the various types of data and numerous ways that these data are processed. Yet the old principle of a data warehouse remains, but with bells and whistles, and is implemented at a much grander scale where the traditional (relational) data warehouse becomes just part of the entire (enterprise) analytics platform or *modern data warehouse*.

A business intelligence tool such as Power BI fits in very nicely with a modern data warehouse. It also fits in very nicely in an enterprise data warehouse environment that is built along the traditional lines. However, there is also another arena that Power BI works well and thrives. In this arena, you have small organizations, departments of medium to large organizations, and in some case, specific large organizations themselves.

These institutions have one or more of the following in short supply; foremost is budget. Then comes other restrictions: the workforce to build a business intelligence solution (with the right skills, the right size, and the recommendation of the right technology), stakeholder buy-in (with the right conviction), and of course a champion who sees the value of analytics.

When such things are in short supply, the justification to invest in a business intelligence solution is naturally bleak, unless those in these institutions understand the value that analytics brings, and initiate a program that starts small, showcases value, then expands across departments or verticals of the organization, and finally up the ranks for the entire organization. The keyword, of course, is *budget* and to help come around this hurdle we have Power BI.

So, the premise of this chapter, and what it will build upon throughout its course will be the following: how with Power BI, small organizations, departments of medium to large organizations, or in some cases even large organizations themselves can start, evolve and build a complete business intelligence solution with the least investment. Power BI goes beyond being a cost-effective tool by providing the features and functionality to develop such a solution without the need for any other technologies, hence this chapter will deal with how all of this

can be achieved. Of course, there will be some exceptions and limitations, but that will be dealt with as we progress.

Putting a Solution Together with Power BI

A cautious approach to start a business intelligence solution is to start small with something tangible that provides value to a stakeholder, then get feedback and incorporate changes. This, in most cases, would be a process of 3-4 cycles. A tangible deliverable ideally would revolve around a business process or activity, such as sales performance monitoring, or an executive dashboard.

Once the value is proven, you move on to the next piece, and so forth. When the time comes, of course, an enterprise solution will be the natural course. While by that time much value will be seen by stakeholders, where budget may not be a question anymore.

Let us first look at a conceptual solution to determine who the actors are, and the processes that will be put forth for these actors to enable analytics, and to consume analytics for business intelligence. All of what you will see in this solution needs to be implemented at the beginning but can be included as the solution gains acceptance, and new requirements come in.

The Actors

The most important actor is the business user, also called a business analyst. Why the most important? Because this actor, performs a variety of activities in a business intelligence initiative such as this. From understanding the business needs, then going back to the various business systems to identify the right data sources, connecting to them, extracting, transforming and integrating the data, building business models on top of the data, and finally crafting the reports against the business models to answer specific business questions.

Now, it does not mean that such an individual, nor that the business analyst role should be carrying out all these tasks. Depending on the complexity and workforce that is available, the data wrangling and modeling task can be separated out to an individual with technical skills.

The end user becomes the ultimate consumer. They are the folk who would make or break the initiative. If the value is delivered to them, and that too in good time, they are the ones who would eventually become sponsors and stakeholders for the initiative.

Power users are actors who would come into the picture later in the initiative. These are users who go beyond what a business analyst would do to provide analytics to end users. Data scientists also make up this group.

The Solution

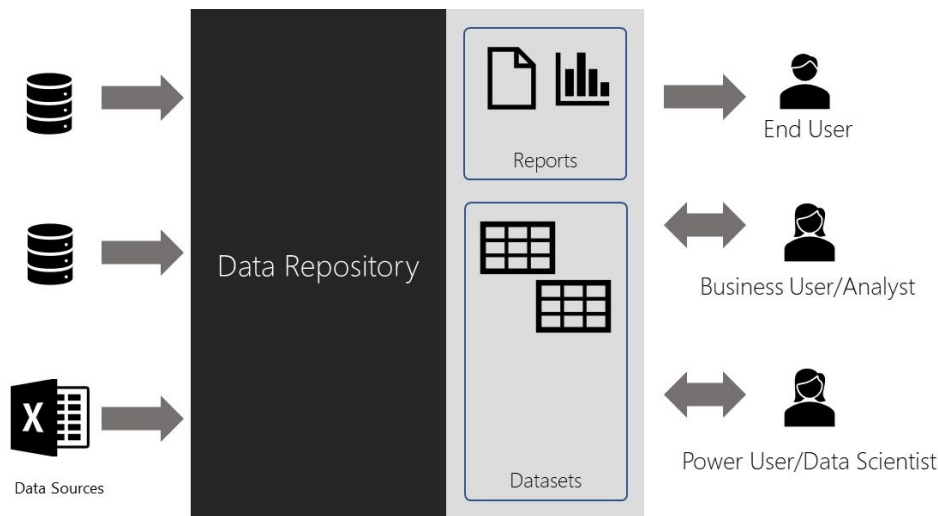


Figure 21-01: The Solution

In principle, the solution can be depicted as what you see in Figure 21-01. Information that is present in multiple data sources needs to be brought into the system on to a data repository. The three types of users would want to access and manipulate information in multiple ways. The end user needs to access and read report and dashboards to act upon the insights. To facilitate that, the business users will create perform the data cleansing, structuring, and formulating of data and load them to datasets, and then create reports and dashboards. The power users will go beyond what the business users do to create complex datasets and analyses to provide to the end users. In many cases, the business user will play the same role as the power user.

As simple as the above narrative explains the solution, the process of doing this can quickly become a nightmare. Power BI has evolved over the years and months into a tool that does wonders with data. Many an insightful report and dashboard complete with beautiful layouts have been built, and those who use its insights to make informed decisions have seen value in these “solutions”.

The issue, however, occurs when many such overzealous solutions start cluttering the environment that it becomes quite hard to maintain. Moreover, you would not know if all these solutions indeed possessed the single version of the truth. How one set of users define a metric, or a KPI differs from how others define them. When these two sets of users come together at a quarterly sales summit and start presenting their insights, everything but fisticuffs break loose. It becomes evident that when self-service business intelligence starts hitting the

brinks of madness, a method needs to be put into place to contain it

The solution that we are going to focus on in this chapter will be limited to Power BI Pro functionality. Power BI Premium features are deliberately left out since, as suggested in the Background, we will be focusing on small organizations.

Data Movement and Processing

Data Movement and Processing in Typical Cases

The movement of data from its sources, all the way to the reports follows a specific path. The following diagram outlines such a path that is similar to what traditional, enterprise, or even modern big data solutions make use of:

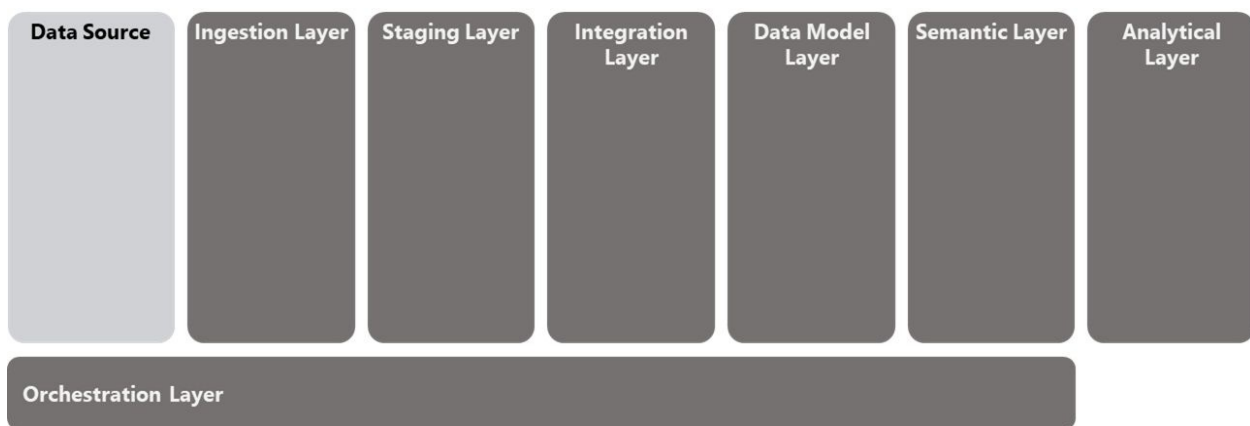


Figure 21-02: Typical Data Integration and Modelling Architecture

In a typical data integration architecture, the **Ingestion** layer pulls in data from the source systems and dumps them as-is on the **Staging** layer. Of course, the staging layer will be structured for effective dumping. This is the extract and load mechanism that we talk about in big data scenarios where data is ingested to a data lake, or in traditional cases when data is extracted and staged from source systems, to avoid performance issues at the source.

Then, when the need for analysis and reporting arises, relevant data will be cleansed, transformed and integrated via the **Integration** layer into the **Data Model** layer (which essentially will be transforming data onto a relational data warehouse or data mart, for example). The data model will evolve as more and more requirements are fulfilled but will remain the base organization-wide.

The **Semantic** layer provides data modelled for the business at a business function-specific level, be it descriptive, diagnostic, or even prescriptive. All that the users must do is use it in the **Analytical** layer in various ways; an example is

a self-service analytical report.

The complete data movement pipeline is facilitated and coordinated by the **Orchestration** layer.

Data Movement and Integration in Power BI

Now, in the Power BI-based scenario, not all the above layers can be segregated as they are, and they need not be either. For one, the above is ideal when analytics is designed and implemented for an enterprise since multiple technologies are used for specialized tasks. Hence, in our case, let's try looking to replicate the above as best as we could, but at the same time combine layers to overcome technical hurdles imposed by the single technology that is in focus here.

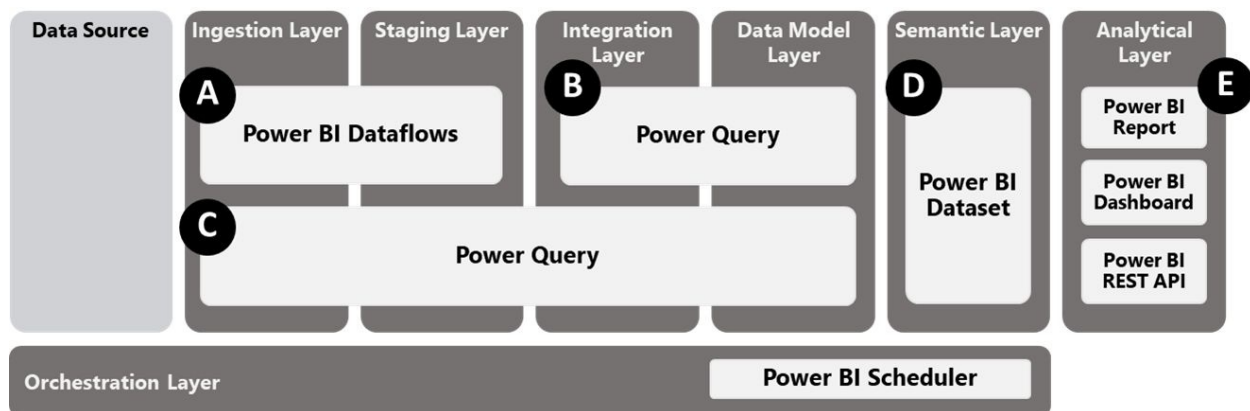


Figure 21:03: Data Integration and Modelling on Power BI

In the Power BI-only universe, one of the disadvantages that we have is the lack of a staging ground. However, this need not be a hindrance. We will utilize Power BI Dataflows to extract and prep entities that will be used as the basis for dimensions across multiple business units. Think of Dataflows entities as the catchall dimension table of a data warehouse. It will possess all the possible attributes of every dimension.

Take a look at Figure 21-04, which is the solution architecture; it shows three layers, the **curated staging layer**, the **business function layer**, and the **end user layer**. To understand how the *data integration and modelling architecture* (Figure 21-03) correlates with the *solution architecture*, you will need to refer to both diagrams when you read the rationalization below.

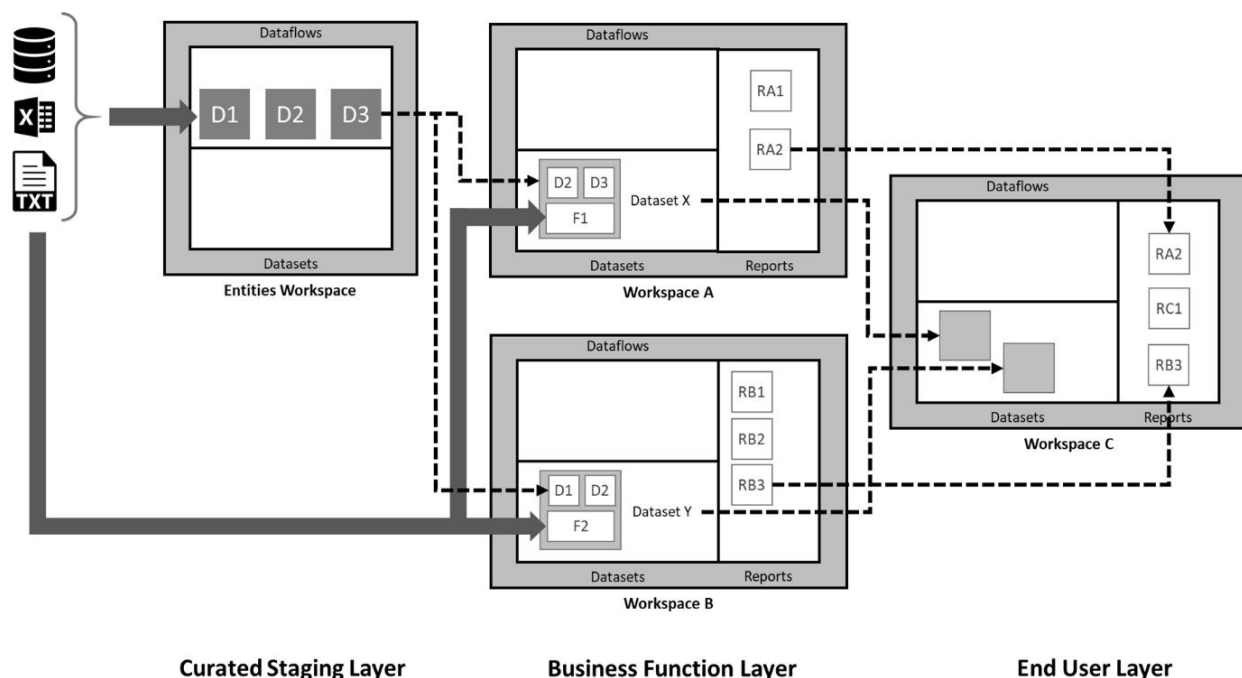


Figure 21-04: Solution Architecture

Curated Staging Layer

First up, to stage data, we do not have the luxury of a dedicated storage area. Hence, staging data as-is will not make much sense. Hence, we will build a set of entities in the **curated staging layer**. This layer will serve as a stage, but with cleansed data, with each entity containing all the information that will be required across business functions.

Now, the type of data that is used for analysis is categorized as dimensions and facts. Dimensions are used across the organization, some more than others. Hence, if a sales analyst from the sales department creates and uses the Product dimension for their sales analysis reports and dashboards, it would only make sense that the same dimension is used for marketing campaign analysis. Of course, the attributes that make-up product in sales would defer from those that make-up product in marketing. Hence, it is important that all possible attributes are available when creating a dimension so that it would cater to various types of analyses and business units.

Therefore, one of the first things that will have to be defined are entities. Entities are objects that are singular and identifiable within an organization. It is the entities that morph into dimensions based on need. Information that makes up an entity may come from multiple sources, and they all need to be combined and structured appropriately before being used across the business.

Therefore, the first step would involve setting up a workspace that will only house entities. These entities will be created on Power BI using Dataflows. This is what will make up the curated staging layer: Data extracted, cleansed, and stored as Dataflows. The layer will consist of a workspace dedicated to maintaining Dataflows of entities for the organization (*Entities workspace*). Think of this as the entity vault, and would only contain dataflows of entities, and no datasets, reports nor dashboards. Dataflows, once created, can be accessed from other workspaces. This will allow users from across multiple disciplines to access the Dataflows and build dimensions that are unique to their business context. Typically, it would be the business analyst role that performs the task of identifying, designing, and creating dataflows for each required entity.

This is **Process A** depicted in Figure 21-03.

Business Function Layer

The next step will be the exercise of creating packages of business process or business function-specific content, for example, sales or finance. These packages will be workspaces that belong to the *business function layer*, and one workspace per business function will be built. Hence, you will have a *Sales* workspace, a *Finance* workspace, and more.

Here, the analyst will create dimensions based on the entities customized for the current business function. If it is the sales business unit that we are looking at for example, and the Product dimension is one of those being built, then this dimension will be customized to suit the needs of sales personnel, for instance, including a product hierarchy, skipping specific columns, and modifying other columns. If the marketing business unit also needs the product dimension for their analysis, they will be provided with their own one customized to their specific needs. Each of these dimensions will be derived from one place: The Entities dataflow from the *curated staging layer* and built separately within a workspace of their own context, inside a dataset in the form of a Query, using Power BI Desktop.

This is **Process B** depicted in Figure 21-03.

Next up, we have facts (or the transactions) that will be pulled out of one or more systems. The data for facts will naturally be more significant in record size, and sometimes more extensive in terms of columns than dimensions.

Transactional data are usually business function-specific, hence staging them as

entities will not make sense. The fact data will follow the long path of being ingested, transformed and integrated straight to the data model layer within a Power BI dataset. The fact tables will meet the dimension tables within this dataset as queries on Power BI Desktop. So, unlike on a data warehouse, where data can be staged, and stored temporarily, in the world of Power BI data preparation is mostly dealt with dynamically.

This is **Process C** from Figure 21-03.

The datasets are finalized when semantic modelling is performed on top of the queries. This is done by creating measures, formatting them, creating hierarchies on the dimensions to further enhance them, hiding columns that have no business purpose, creating relationships among the dimensions and facts, and more. This is when you will have the entire business function-specific semantic model.

This is **Process D** from Figure 21-03.

The next step, within the business function layer, is to build the reports and then dashboards off them (if required). These reports will be built off the dataset. However, when building the reports, the best practice would be to first publish the dataset to the Power BI service onto the business function-specific workspace and then build reports off a fresh instance of Power BI Desktop by connecting to the published dataset. Reports can also be created off the Power BI portal via a web browser, if necessary.

This is **Process E** depicted in Figure 21-03.

The business function-specific workspaces will be owned and maintained by a set of business analysts who belong to that business function, i.e. sales analysts will be responsible for the creation, maintenance and governance of content of the sales workspace, while market analysts will take care of the marketing workspace.

What you will ultimately have are one or more datasets relating to the business function, reports built off these datasets, and dashboards if necessary. All of these can be packaged if necessary and published as an app within the organization (Workspaces A & B). Each of these workspaces will be managed by analysts from the related business function, i.e., finance will put out a finance workspace/app, sales would do the same, and so forth.

So now that we have departmental analytics packaged into workspaces. These will become standard across the organization, and users and analysts from across

the organization can tap into these for their analytical needs, apart from having their own analytics. However, it is a good idea that these business function-specific datasets be promoted for use by others, thus indicating that the dataset is indeed something that has been relatively used and tested. This would give confidence to users of the organization that the dataset can be used to answer crucial business questions. These datasets can be further endorsed by going through a certification process that will allow a specified select set of users, ideally, those who are subject matter experts of the business function domain to endorse the dataset, thus enabling wider adoption and trust across the organization.

The screenshot shows the 'Datasets' tab selected in a navigation bar. The main content area is titled 'Settings'. Below the title, it states 'This dataset has been configured by' followed by 'Last refresh succeeded: Wed Jul 03 2019 00:57:18 GMT+0530 (India Standard Time)' and a link to 'Refresh history'. A list of settings is shown with expandable sections: Gateway connection, Data source credentials, Parameters, Scheduled refresh, and Endorsement. The 'Endorsement' section is expanded, showing three options: 'Default' (selected), 'Promoted', and 'Certified'. The 'Promoted' option is highlighted with a red box and includes the text 'Promote this dataset with a badge to show it's ready to be used by others.' The 'Certified' option is also highlighted with a red box and includes the text 'Request certification from experts in your org to get a badge that shows it's recommended for use by others. Learn more'. Below the endorsement options is a 'Description' section with a text area for describing the dataset's contents and a character count of '500 characters left'.

Figure 21-05: Options to promote and certify a dataset

End User Layer

Business users from across the organization can build upon the packaged business function-specific datasets and reports by bringing them in, by building reports on top of these datasets and creating their own datasets and reports within their workspace (Workspace C). Management dashboards that showcase aspects of the entire organization, drilling through to individuals reports from

various business functions is an excellent use case to illustrate this.

Security

The containers that we work within this architecture are workspaces, and as such, the Entities workspace needs one or more contributors who will perform the task of creating entities using dataflows. All users who will need access to consume the entities can be assigned the viewer role, so that they may not modify the contents of this workspace. A good idea will be to create AD security groups for the viewers of the entities, add the viewers to the group, and then set up the group as a viewer on the Entities workspace.

The viewers will be added as contributors on their respective business function workspaces, and here too these contributors can be bundled up inside an AD security group. Users from outside of the business functions may be included as viewers, and this time via another AD security group.

Development life-cycle

Source control

Unlike a solution where a development team completes building the solution, setting it up, and leaving the users just to use the system, a solution such as these involves the users' active participation. Hence when source-controlling the code that was crafted for the dataflows, datasets and reports need to be versioned and stored.

The easiest and most straightforward method for source control will be to use OneDrive or SharePoint. These technologies have a Check in/Check out function that allows you to keep track of a file's history of changes, which makes it ideal to double up for source control. Azure DevOps can also be used with a five-user-free deal, is purpose-built and is loaded with functionality. However, it can be an overkill.

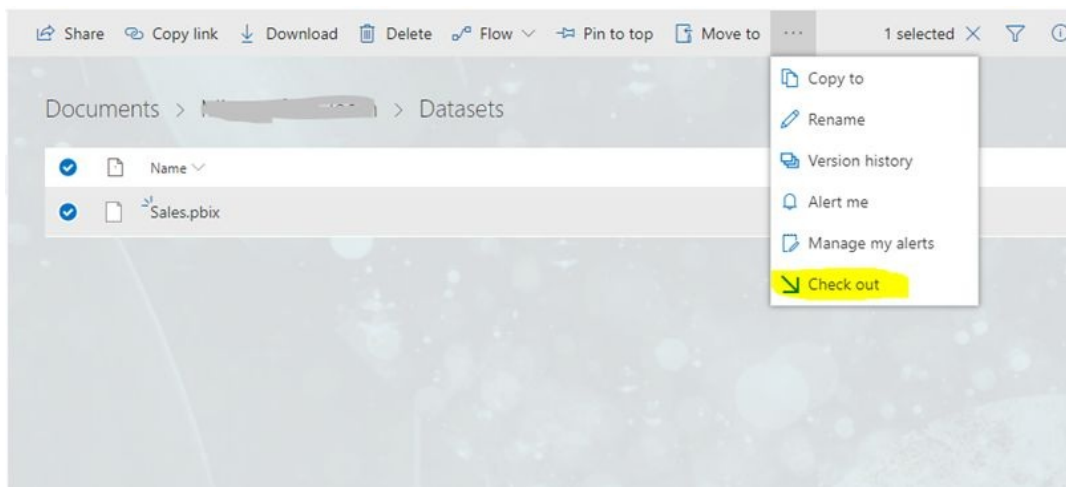


Figure 21-06: Check out function on SharePoint

Regardless of the approach that we utilize, we need to be cognizant to the fact that not all components of the solution can be source controlled in a straightforward manner. When listed down, this is what it looks like:

Component	Is source-control straight forward?	Method
Dataflows	No	Copy Power Query code behind from the Advanced editor, paste to a text editor, before being saved.
Datasets	Yes	Save the PBI desktop

		file used to create the dataset.
Reports	Yes	Save the PBI desktop file used to create the report
Dashboards	No	No source control mechanism available

Figure 21-07: Source control likelihood of components

Deployment

Like any solutions development practice, a solution such as this will have to follow a specific deployment process in conjunction with the source control process. An easy approach to take will be through the utilization of a sandbox workspace, where business users can build and test their data structures and reports first.

The process of development will start within the sandbox workspaces where the business analysts will start by building the dataflows to create the entities within a dataflow called Entities. As they are completed the Power Query source of each of the entities will be copied to a text file, named with the name of the entity and put into source control. The next step of creating the datasets will take place on the desktops of the business analysts, where they would create datasets on Power BI Desktop and then published to the business function-specific workspaces. Once published, these *.pbix files will be source controlled in appropriately-named folders. Reports, however, will not be created on the same file that houses the datasets. Instead, reports will be created in separate *.pbix files connecting to the published datasets on the Power BI service. These files, too, will be published to the appropriate workspaces before being source controlled. One thing that cannot be source controlled, however, are the dashboards that you build off report parts.

Once all the development is completed and tested on your sandbox, you will have to “push” your development to the “live” or “production” environment. This is where you will have to “replicate” your development by pulling out the code from source control and applying it on the new environment. If each of the components that made up the solution were able to be saved as files, then the application would be quite straightforward. However, in the case components such as dataflows, you will have to re-create the steps, with the only “easy part” being copying the code over from source control.

One thing that you would need to keep in mind while pushing your development to the live environment is that data source connectivity and credentials may have to be altered at the dataflow, dataset, and report levels.

Going through this process of sandbox-first, and then going live will increase your implementation effort, and will be too much of an overkill for specific organizations. In cases like that the sandbox may be skipped as long as a similar process of source control is set up as a process for developing the primary solution.

Summing it up

To conclude, Power BI is a versatile tool, which can work on the sidelines as a prototyping tool, and at the same time measure up to provide a complete business intelligence solution. Solutioning, however, is not just about building data models and reports to solve business problems. It is to provide a streamlined and structured process to build data models and reports to solve business problems, so that business users can be efficient, data is ensured to be relevant and reliable, while end users will know where to go for which piece of information. This can be achieved quite well with Power BI by architecting and designing the solution along with the ideas presented in this chapter.

About the Author



Gogula is a 12-time Data Platform MVP hailing from Sri Lanka. He currently provides technical leadership for Intellint, the business intelligence and analytics arm of Fortude. 14 of his 19 years in technology has been on data and analytics on the Microsoft space, now mainly focused on the cloud. His experience is complemented with a passion for data and analytics through his involvement in technical communities, writing, speaking, mentoring, speaking, and serving as subject matter expert for Microsoft certifications. He is the community leader of the Sri Lankan Data Community and is also PASS regional mentor for South Asia.

^[1] <http://www.businessdictionary.com/definition/governance.html>