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

Table of Contents

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've 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 Serves 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 typical business intelligence are built on, and how the same structure is leveraged to build the Power BI-only solution using whichever the features that 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:

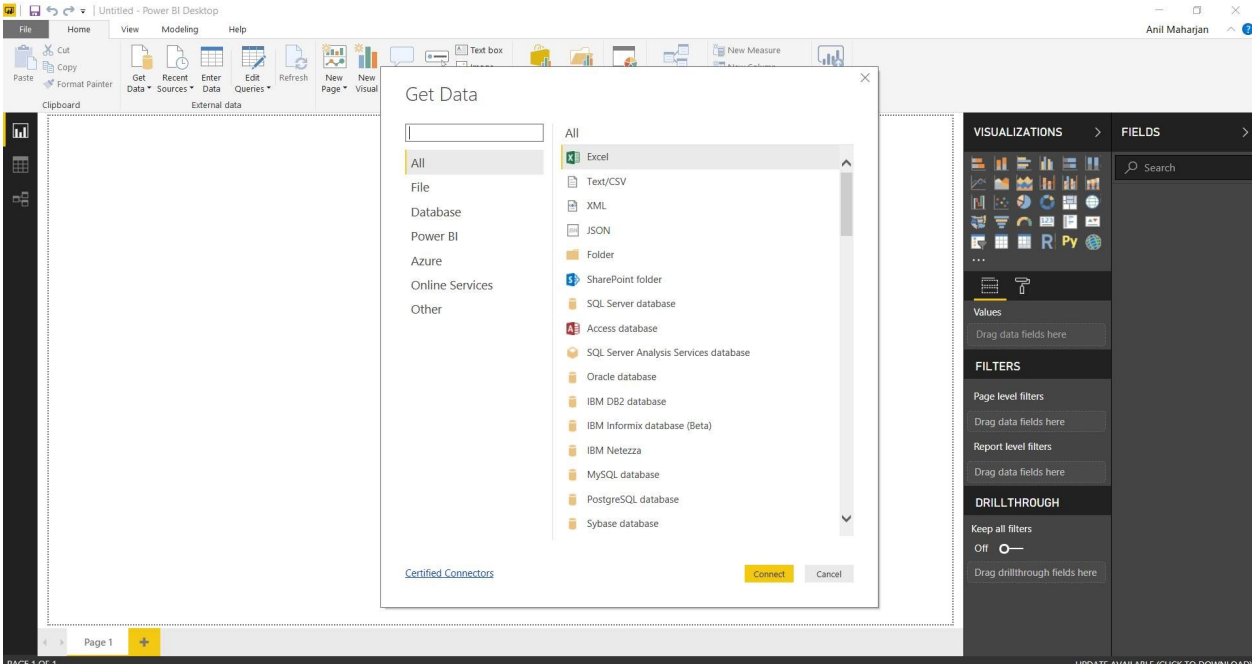


Figure 01-01: Launching the Power BI Desktop -Get Data tab

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.

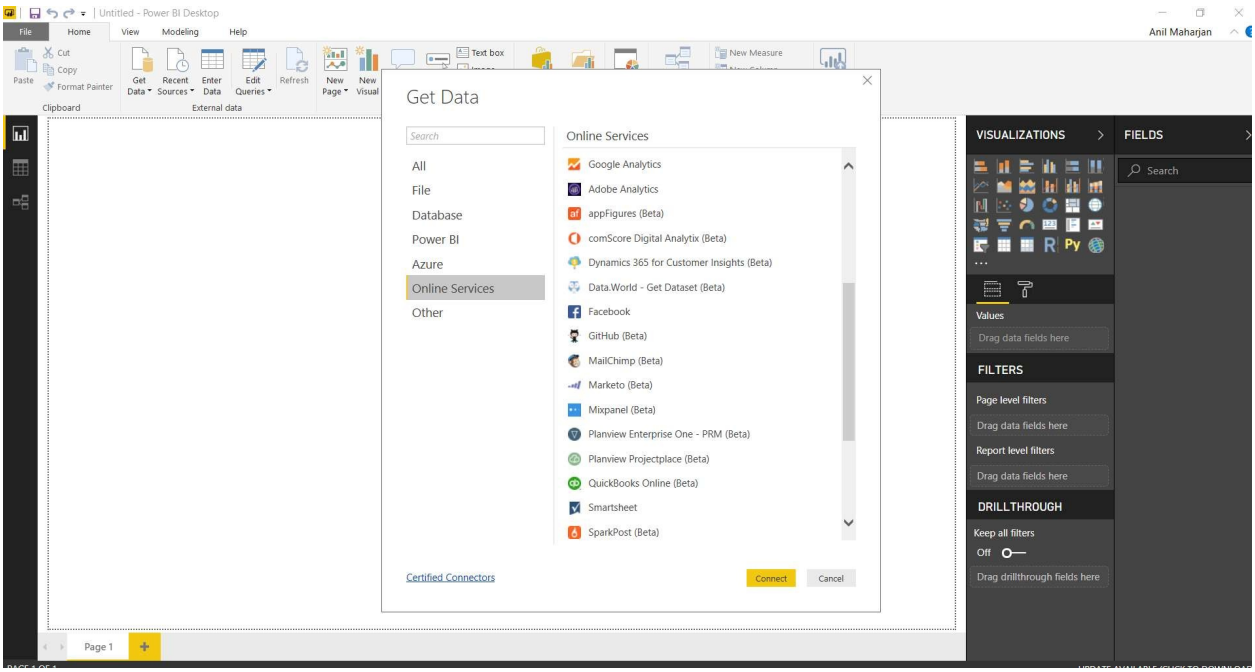


Figure 01-02: Selecting Online Services Facebook as a data source

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

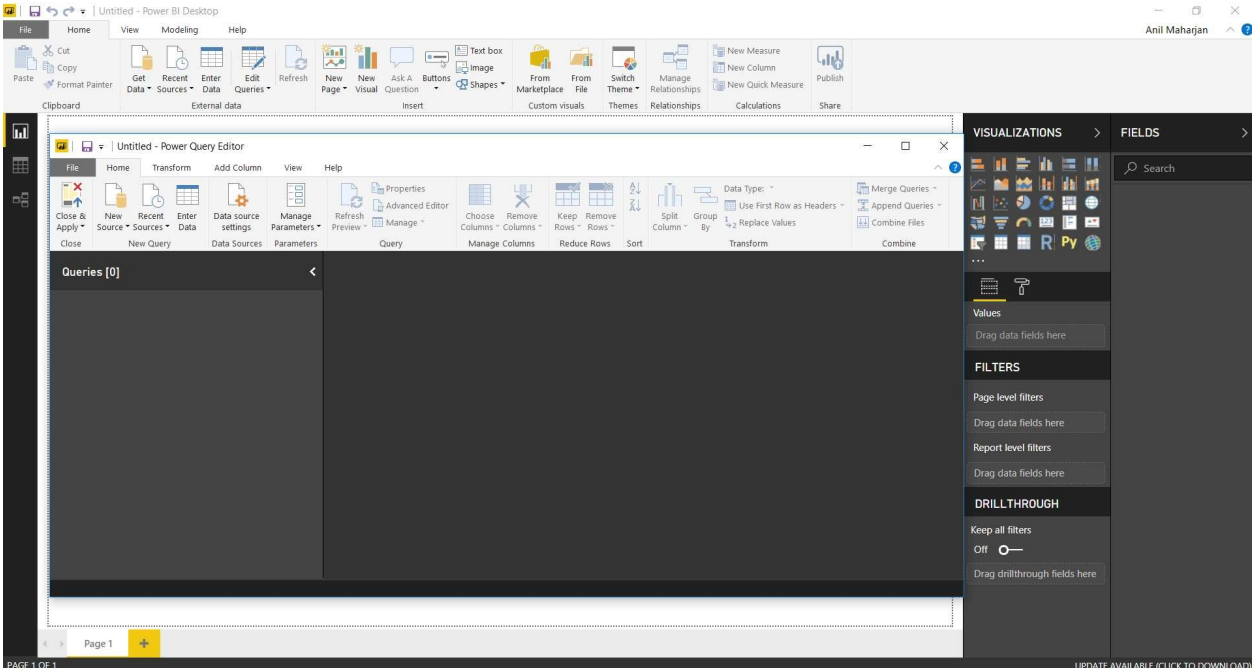


Figure 01-03: Editing Edit Queries section

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.

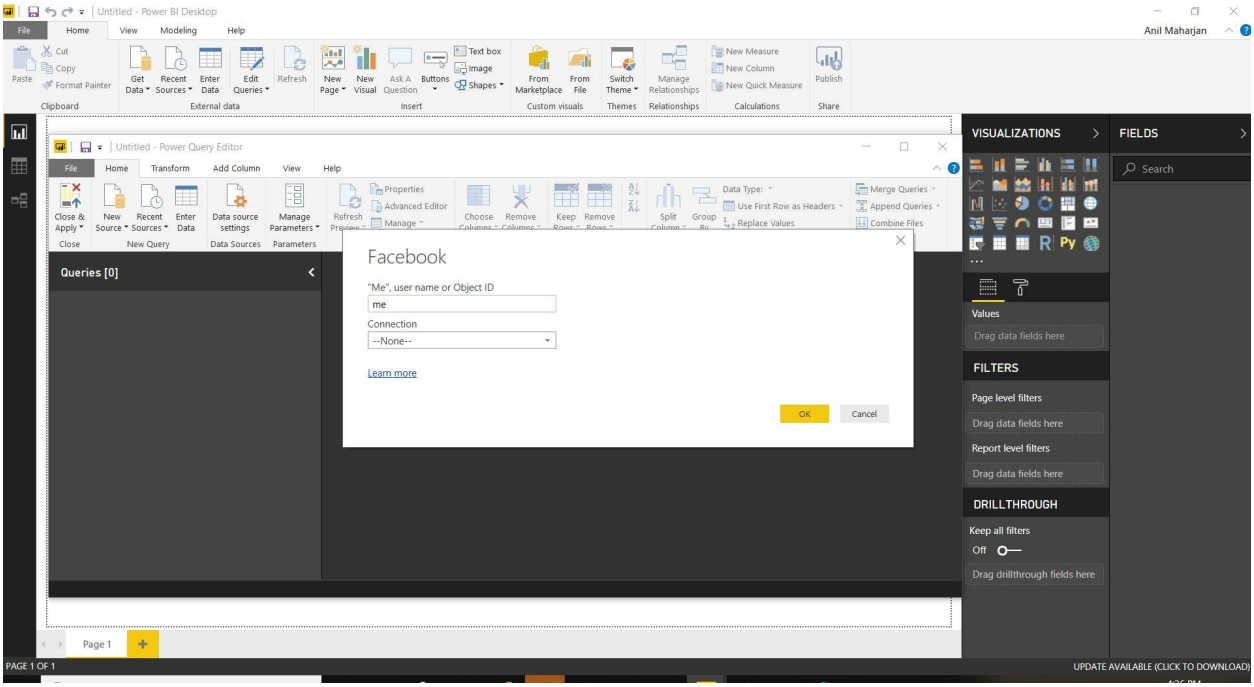


Figure 01-04: Connecting Facebook account credentials login in section

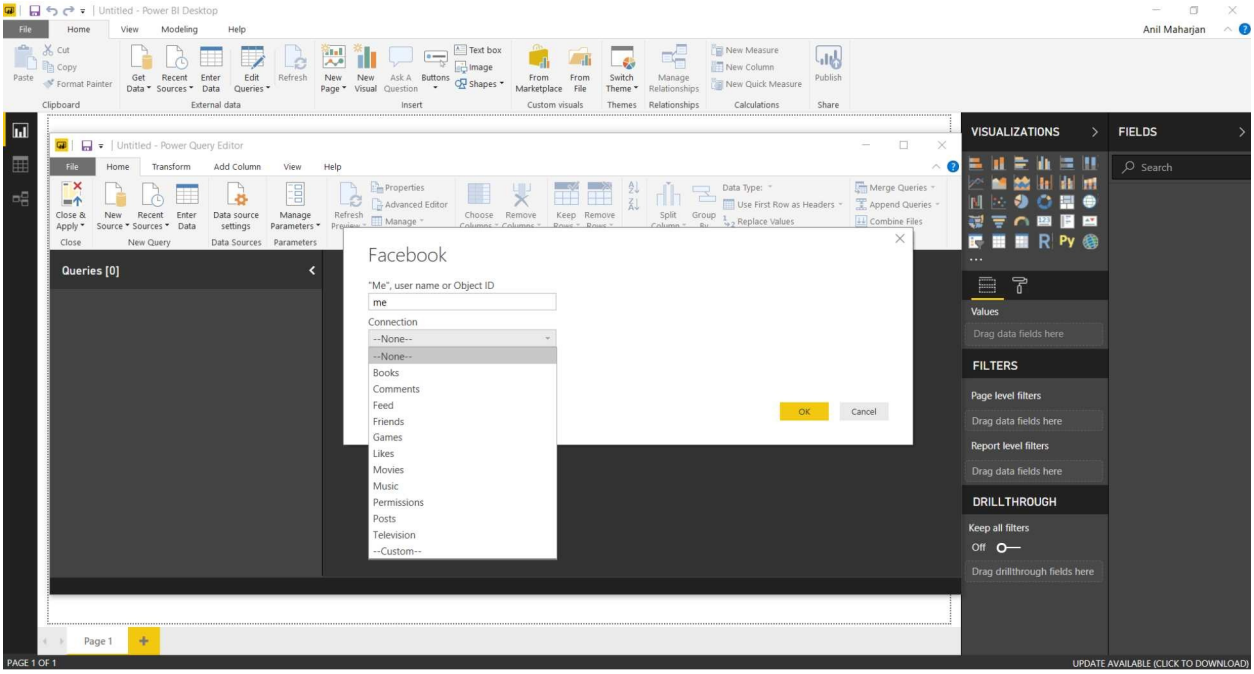


Figure 01-05: Different connection data list from Facebook

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:

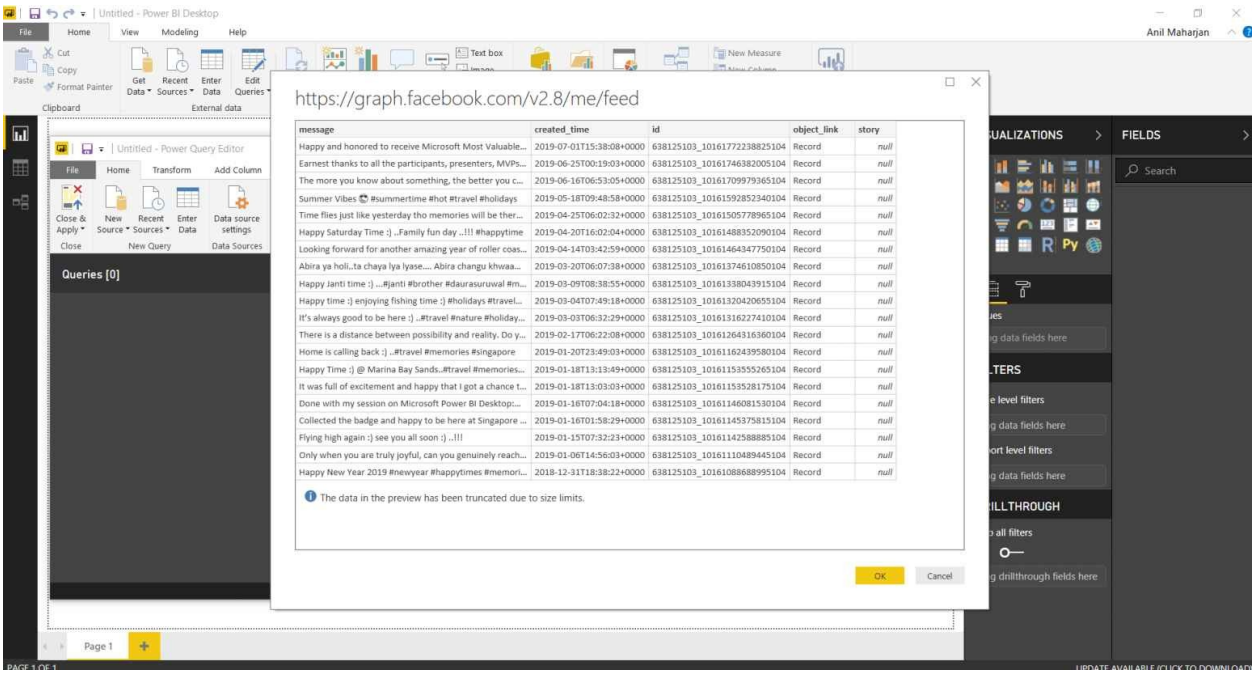


Figure 01-06: Getting Data Feed from Facebook account

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.

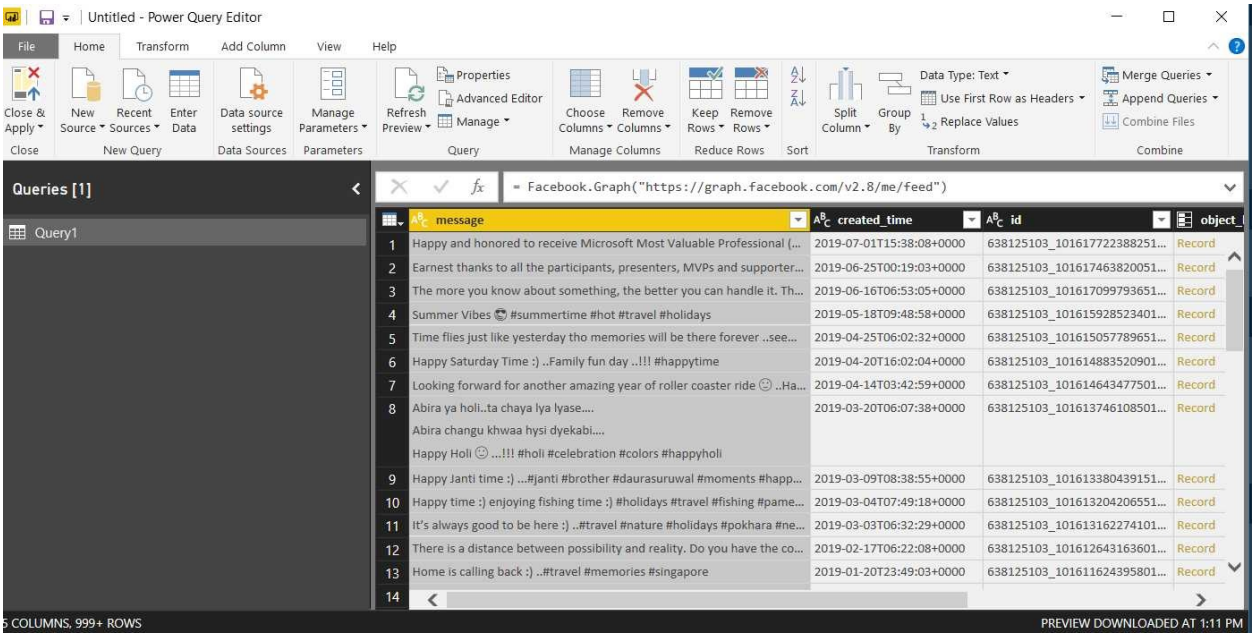


Figure 01-07: Power Query loading data from Facebook

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.

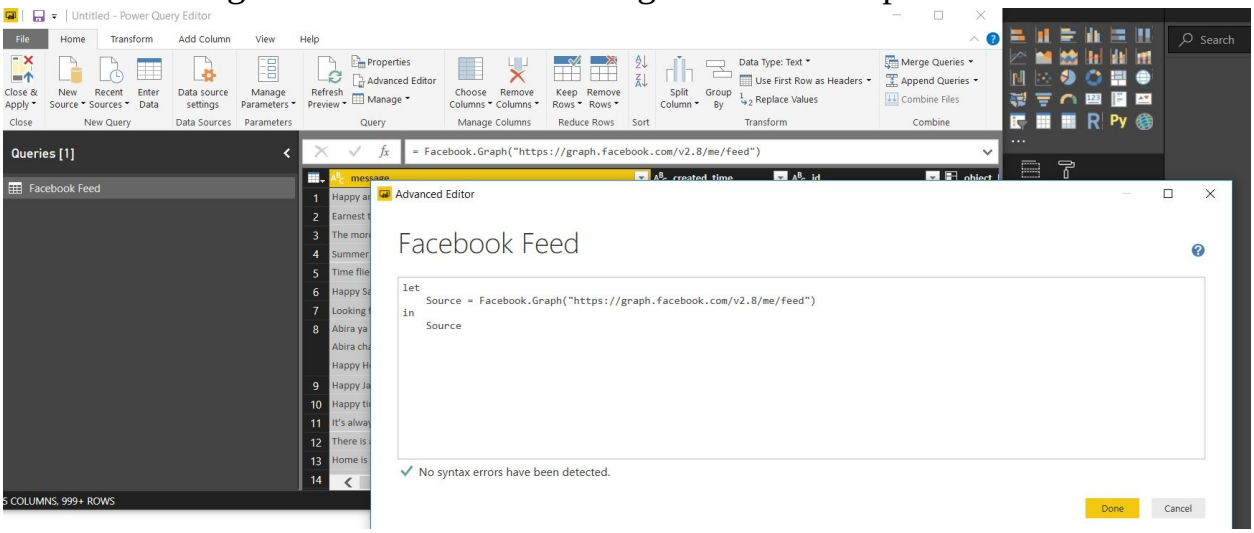


Figure 01-08: Power Query editor and Facebook Graph API connection

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"
```

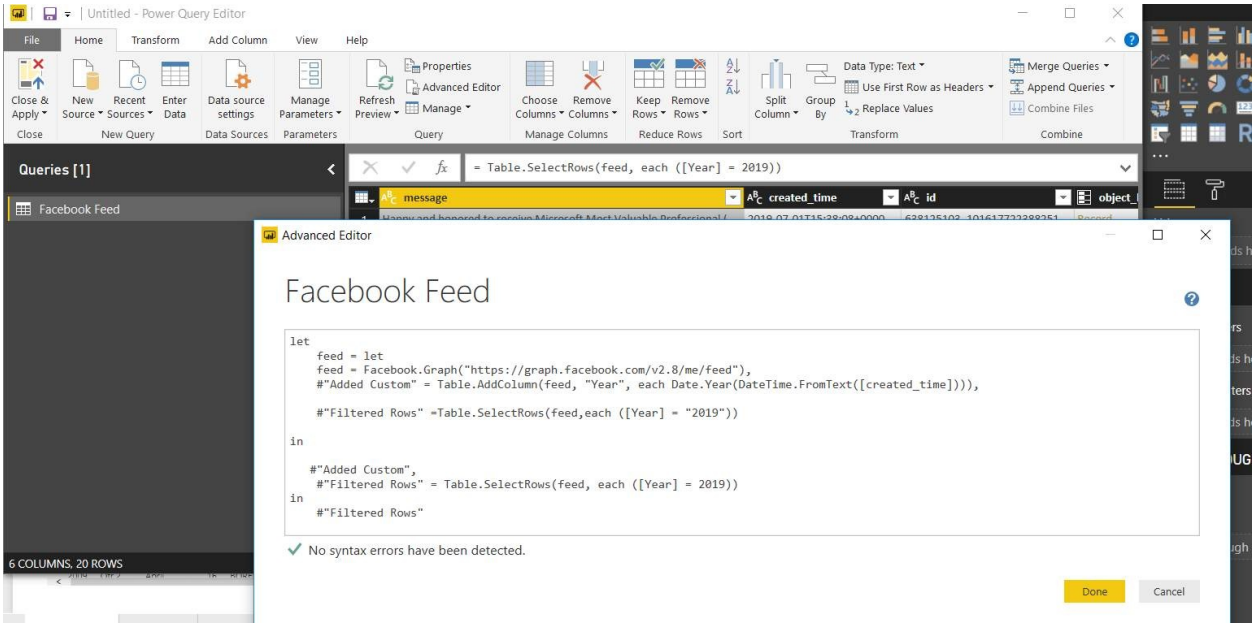


Figure 01-09: Custom Power Query with filter Date

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.

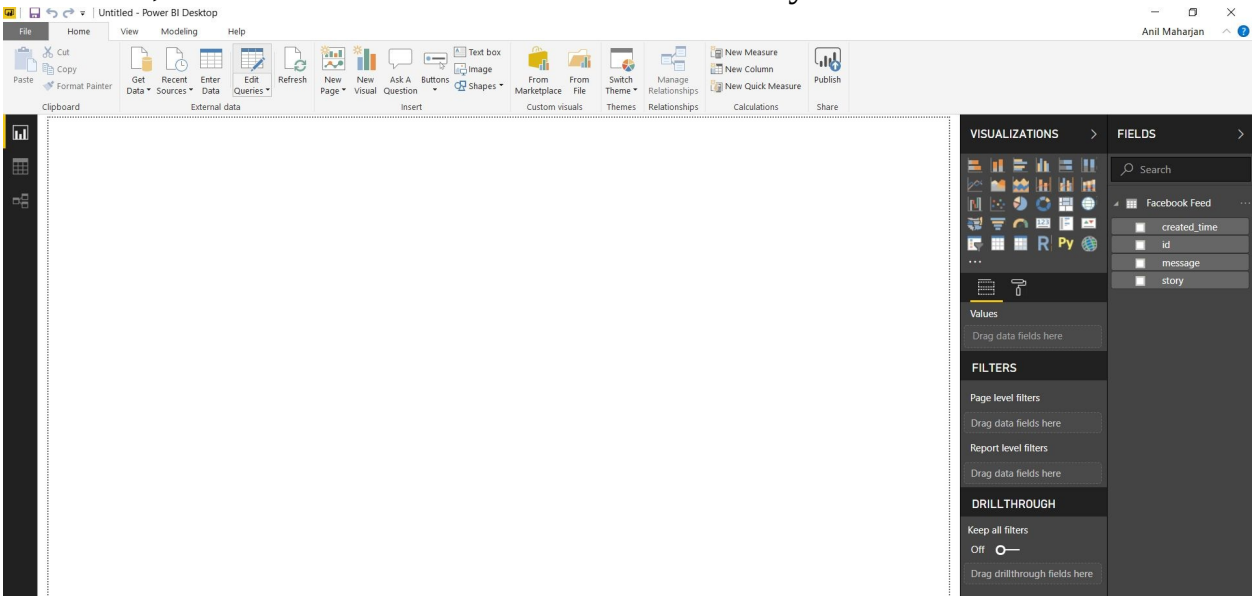


Figure 01-10: Loading data from Facebook

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:

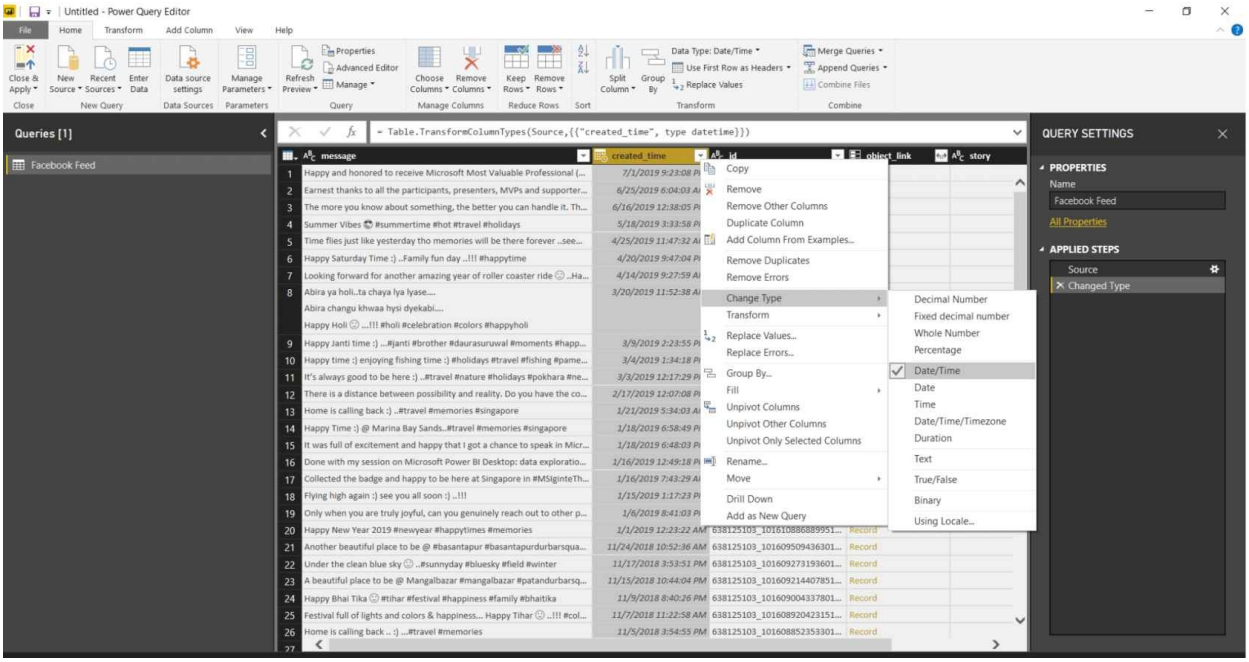


Figure 01-11: changing column format to Date/Time

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.

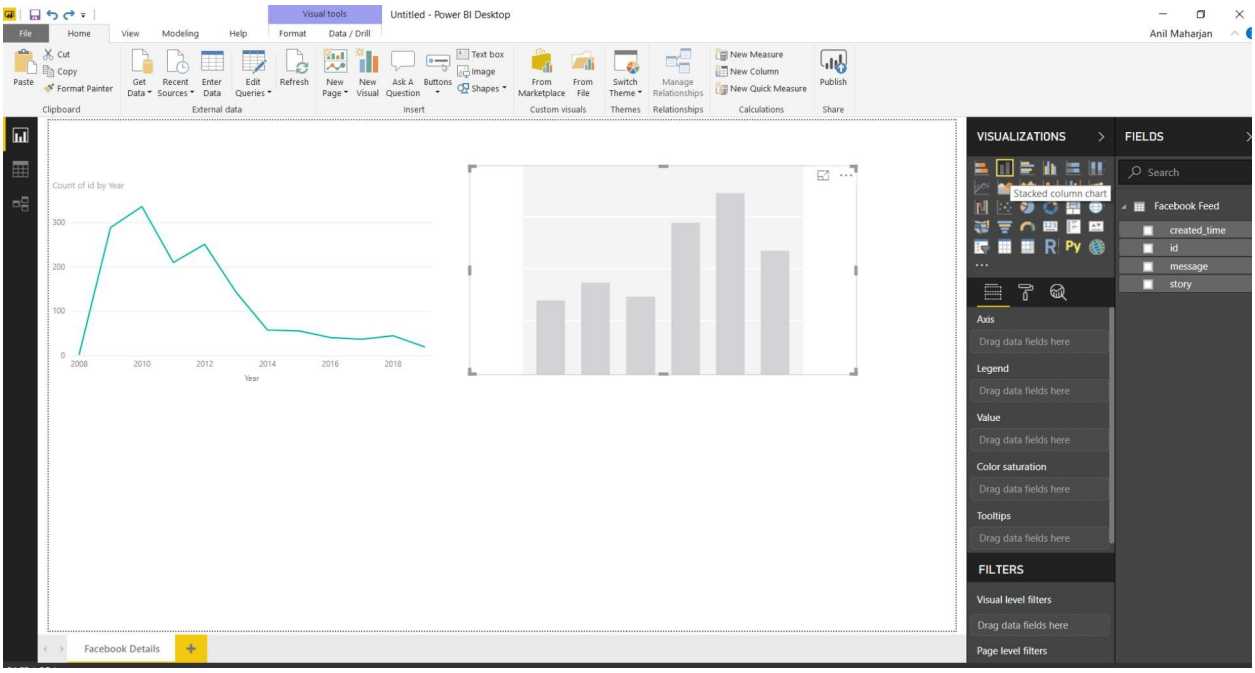


Figure 01-12: Visualization using Power BI Desktop

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.

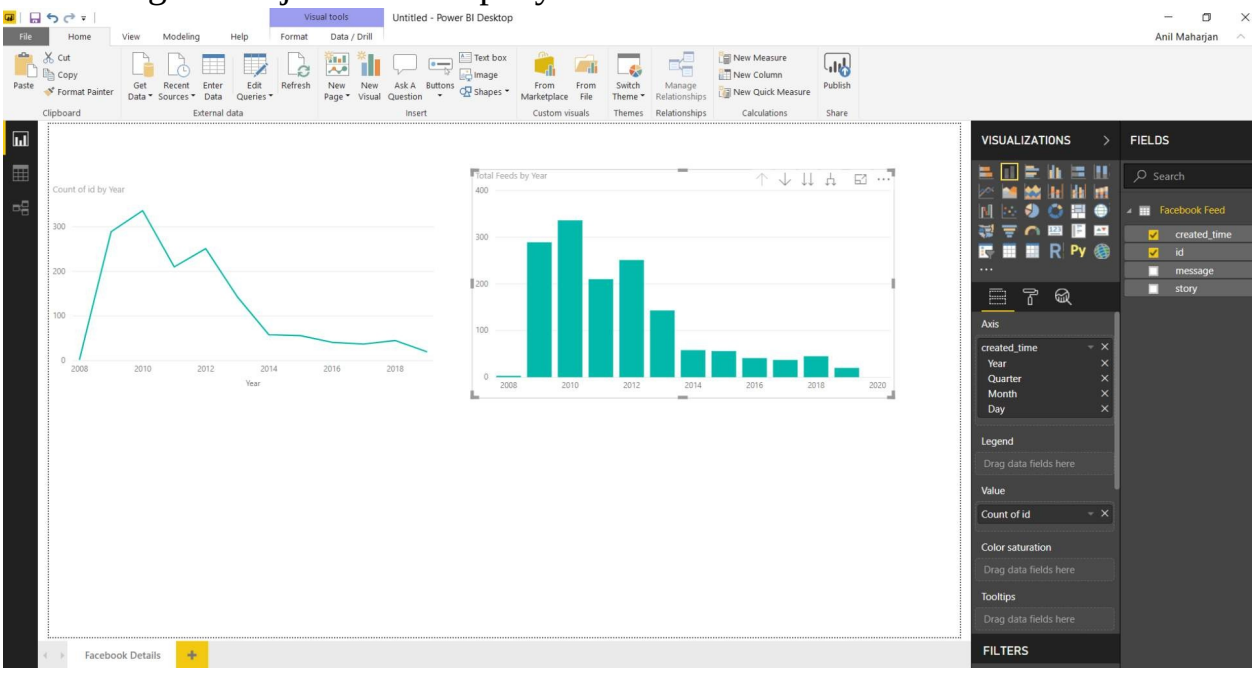


Figure 01-13: Visualization using Power BI Desktop

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.

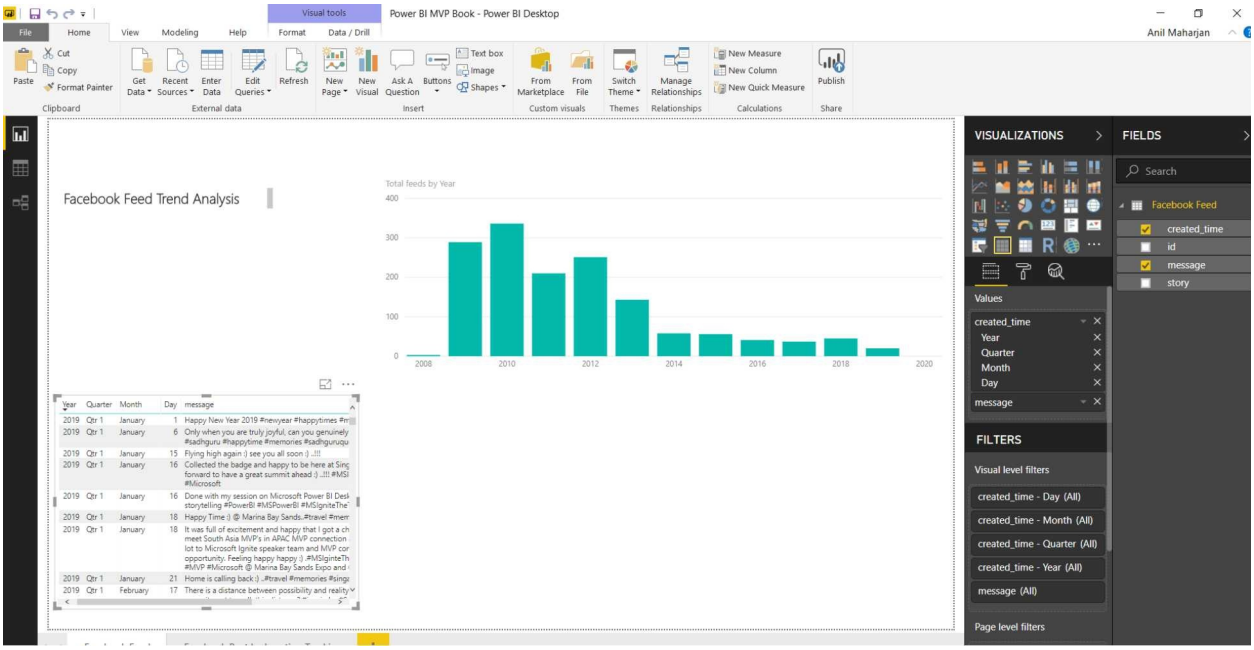


Figure 01-14: Visualization using Power BI Desktop

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.

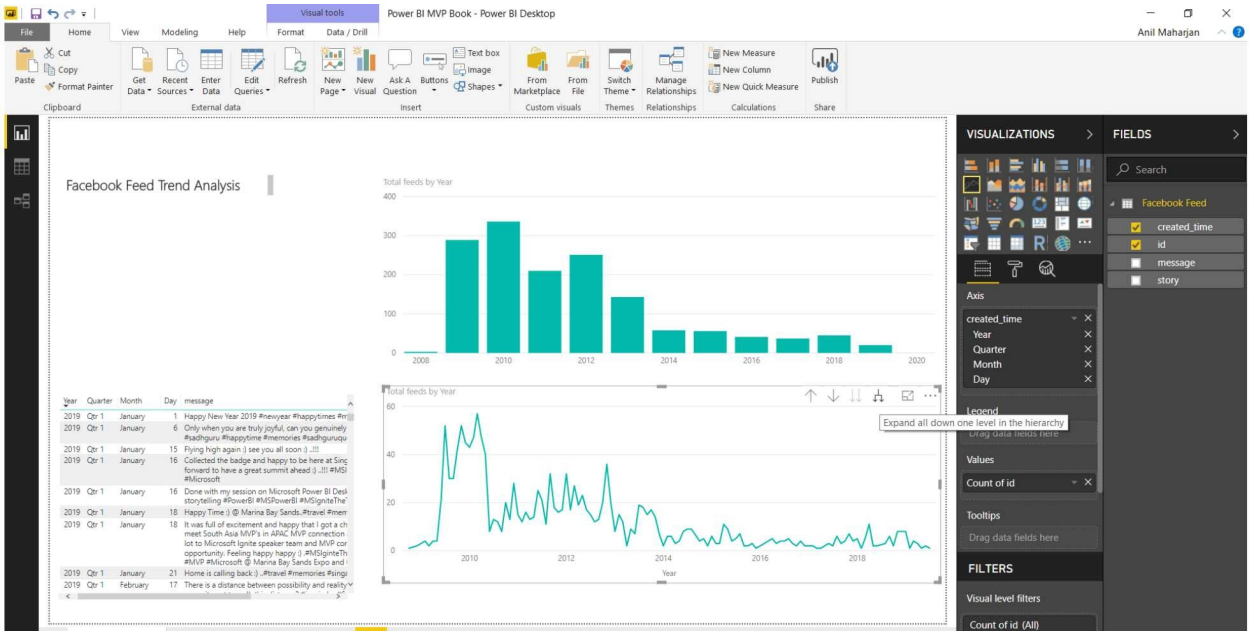


Figure 01-15: Visualization using Power BI Desktop

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.

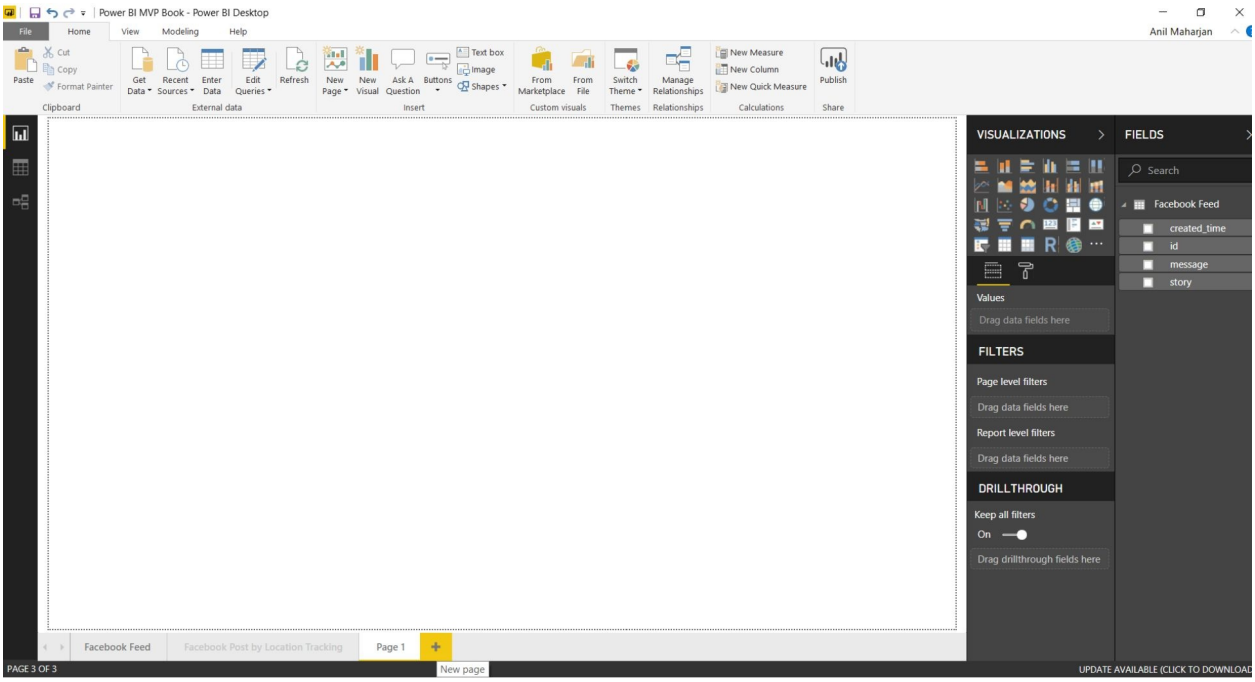


Figure 01-16: Adding new worksheet or page in Power BI Desktop

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

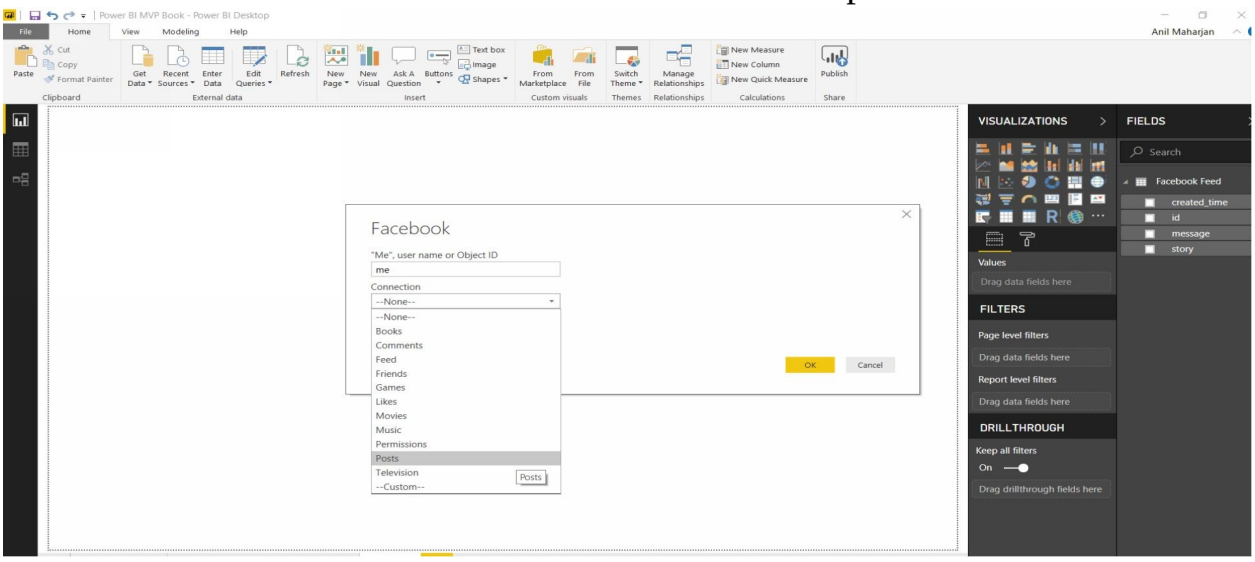


Figure 01-17: Adding new data feed from Facebook in Power BI Desktop

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

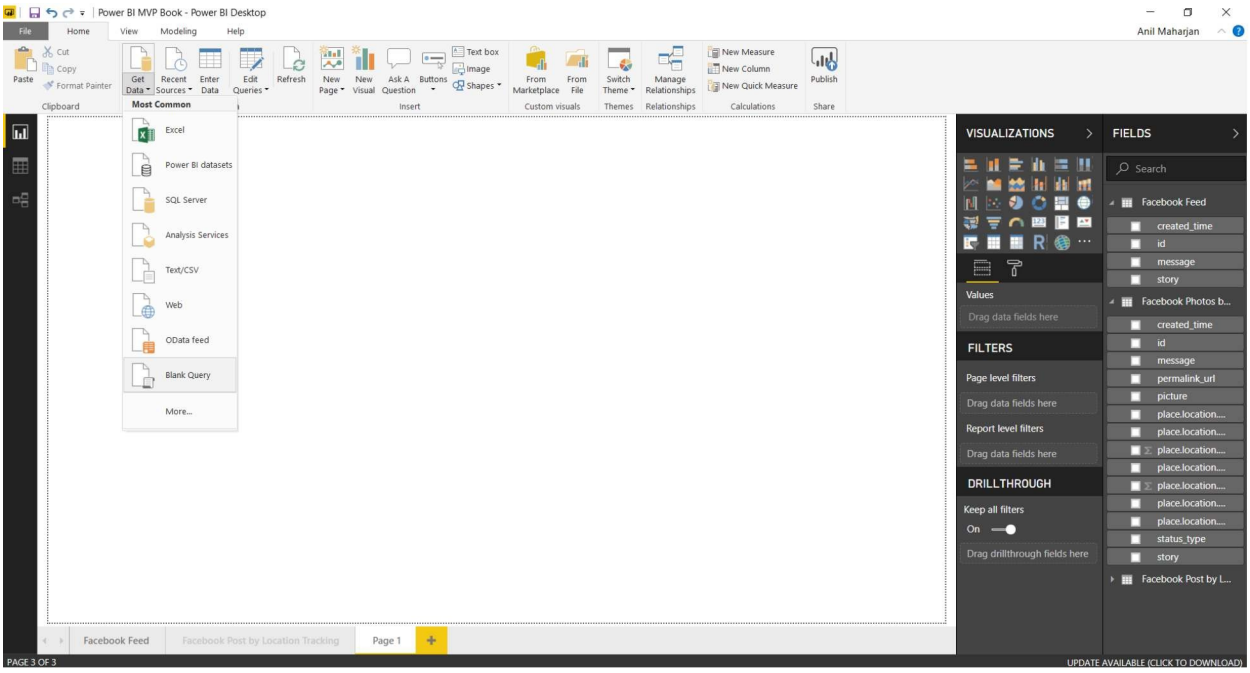


Figure 01-18: Using Blank Query in Power BI Desktop

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

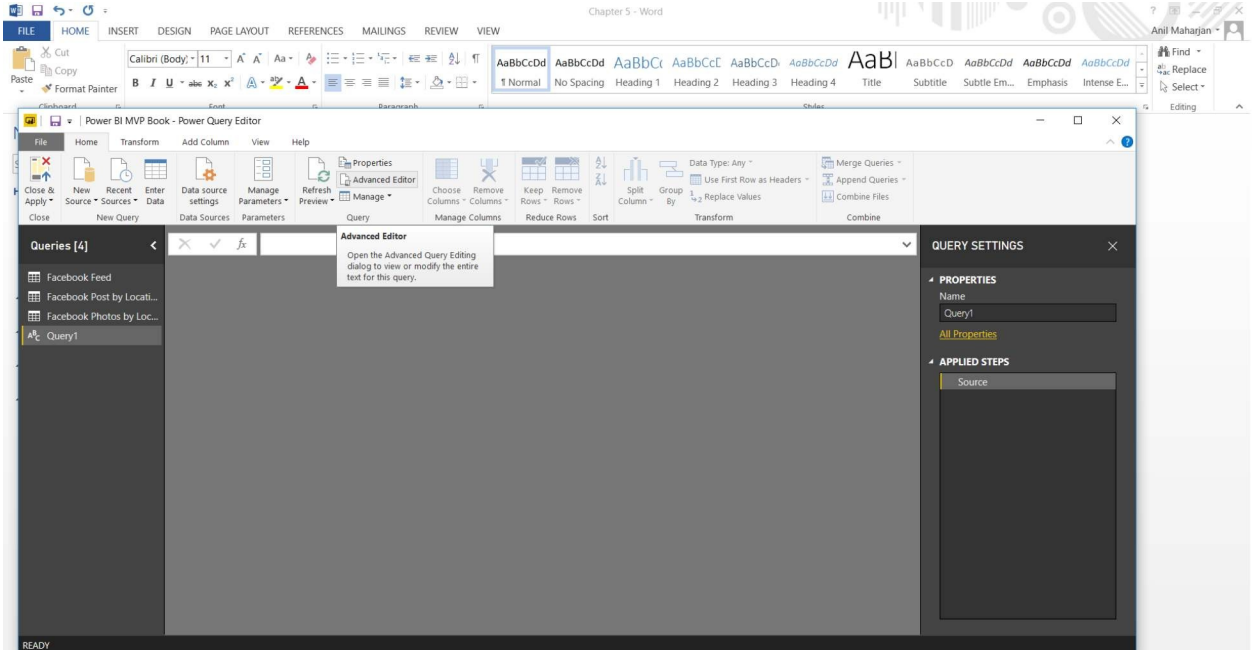


Figure 01-19: Using Blank Query in Power BI Desktop

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.

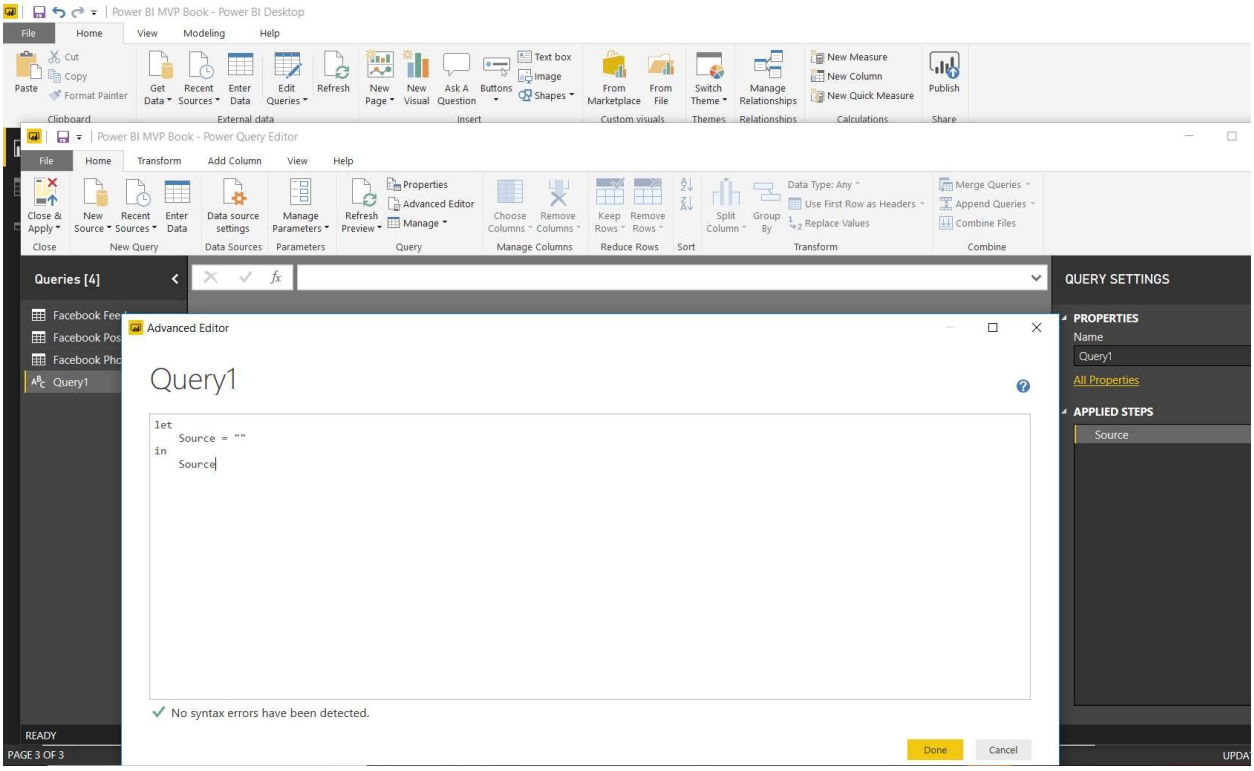


Figure 01-20: Using Blank Query in Power BI Desktop

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.

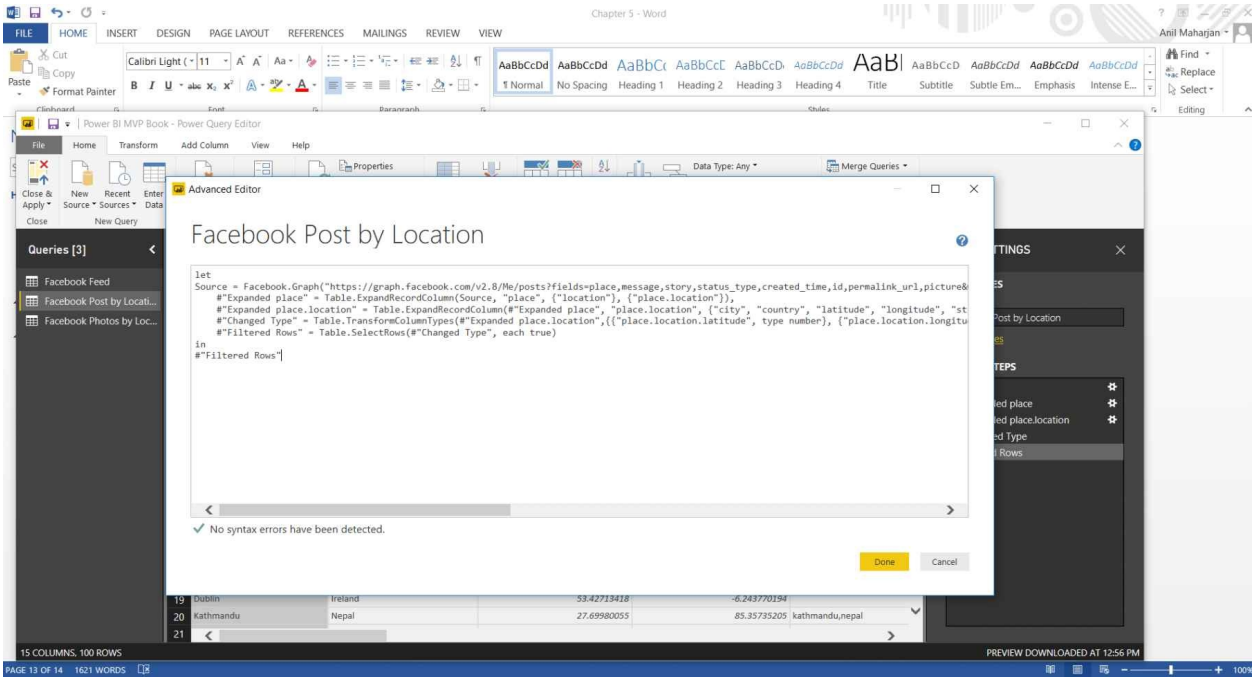


Figure 01-21: Using Blank Query in Power BI Desktop

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**
picture	*Image URL*
permalink_url	*Web URL*
city	*City*
country	*Country/Region*
latitude	*Latitude*
longitude	*Longitude*

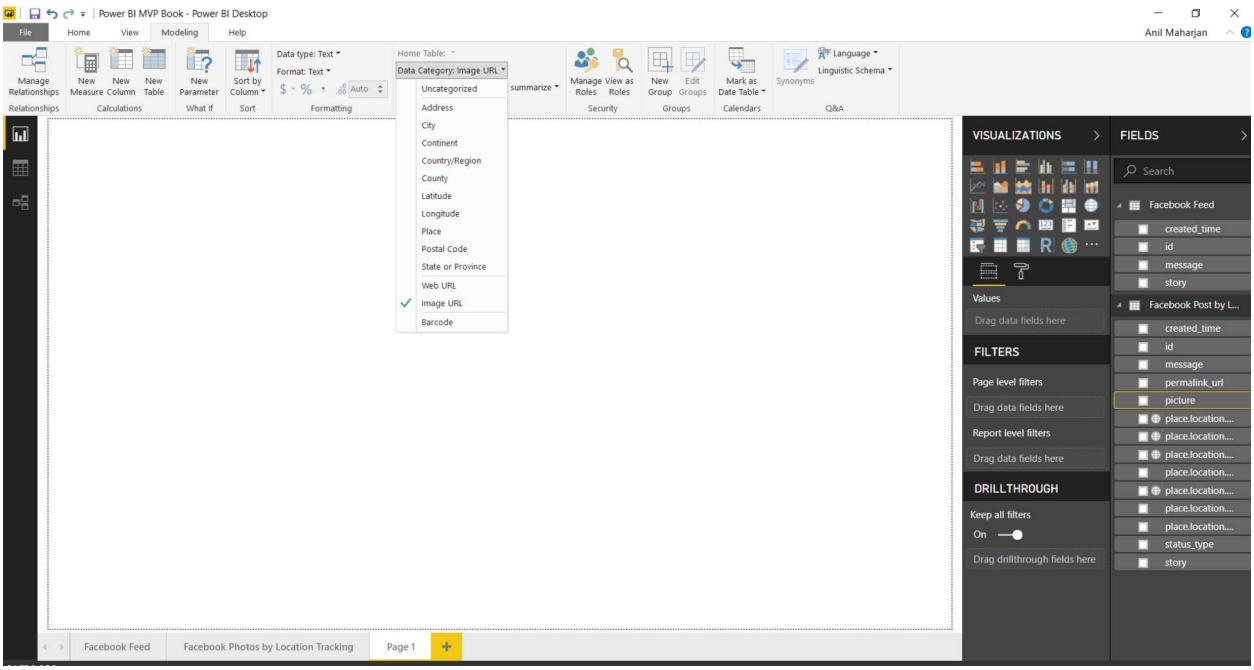


Figure 01-22: Modeling Data Category in Power BI Desktop

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.

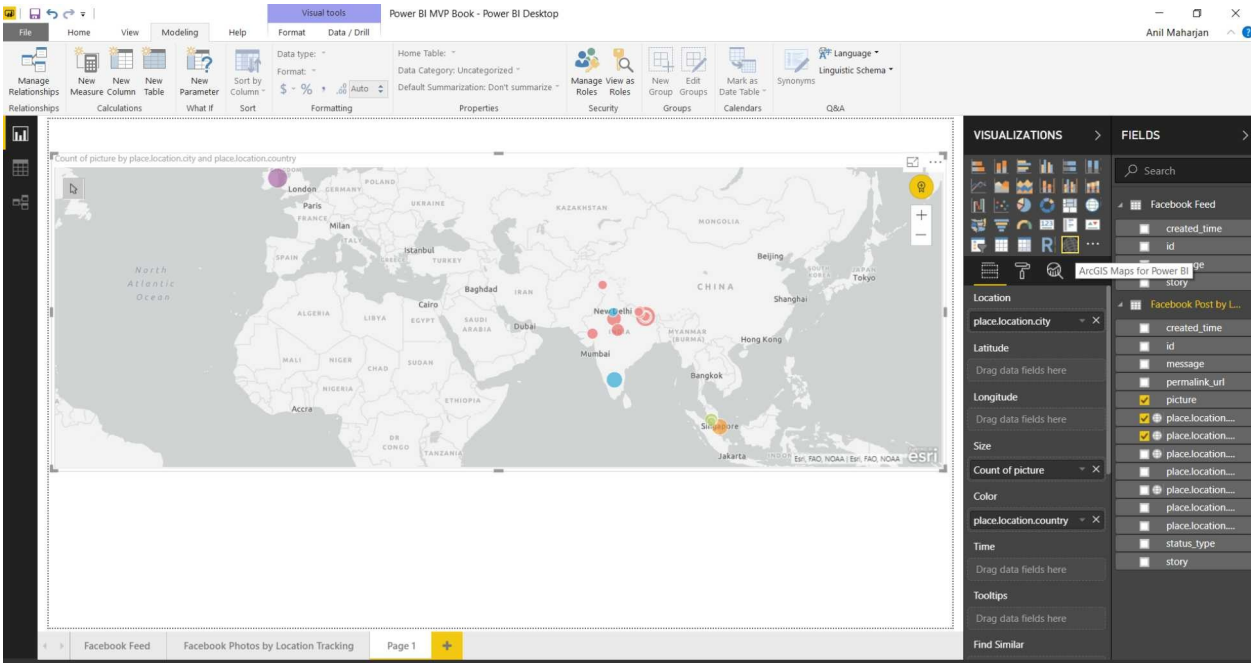


Figure 01-23: ArcGIS Maps Visualization in Power BI Desktop

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.

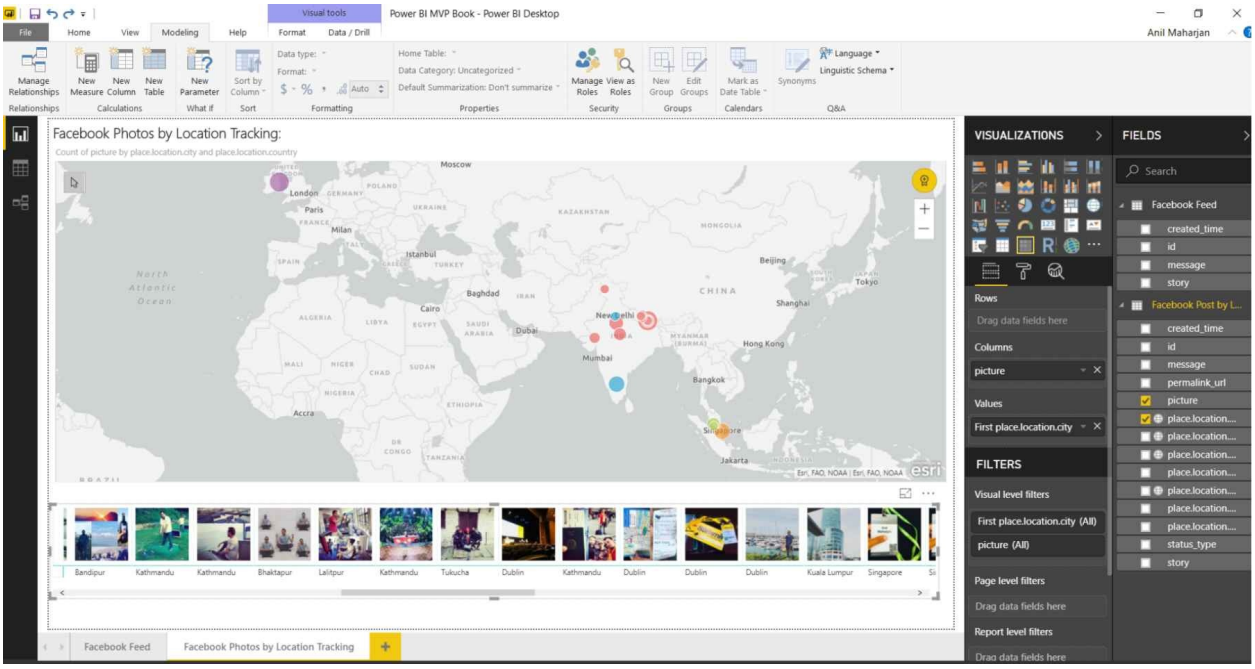


Figure 01-24: ArcGIS Maps Visualization and Photos Image in Power BI Desktop

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.

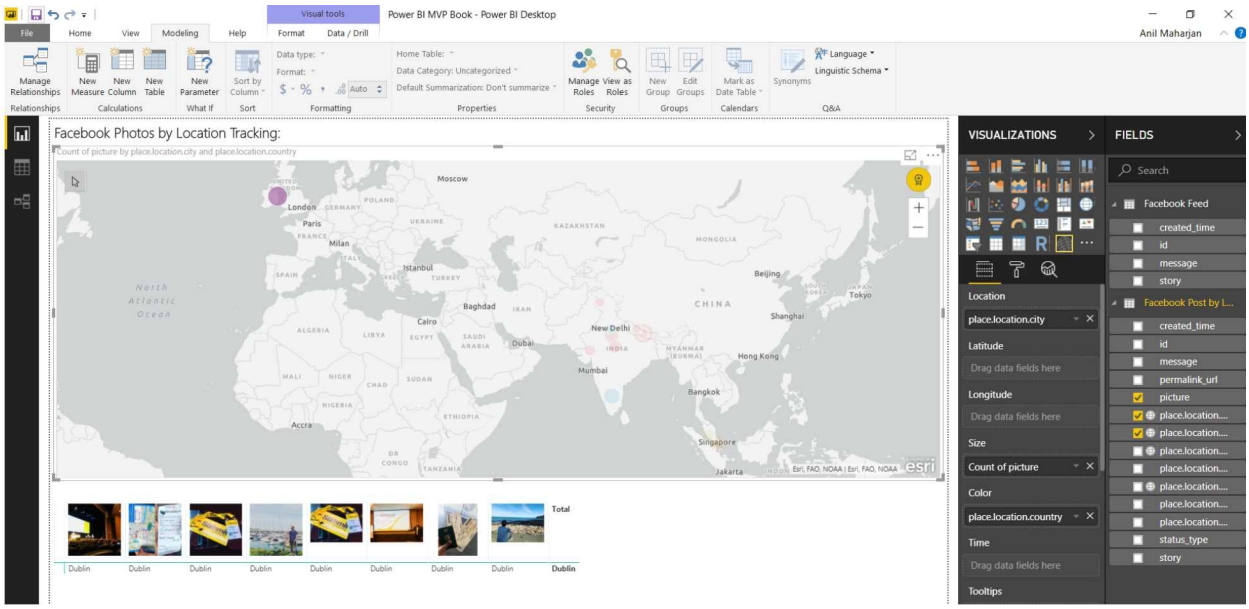


Figure 01-25: Country level filter Photos Image in Power BI Desktop

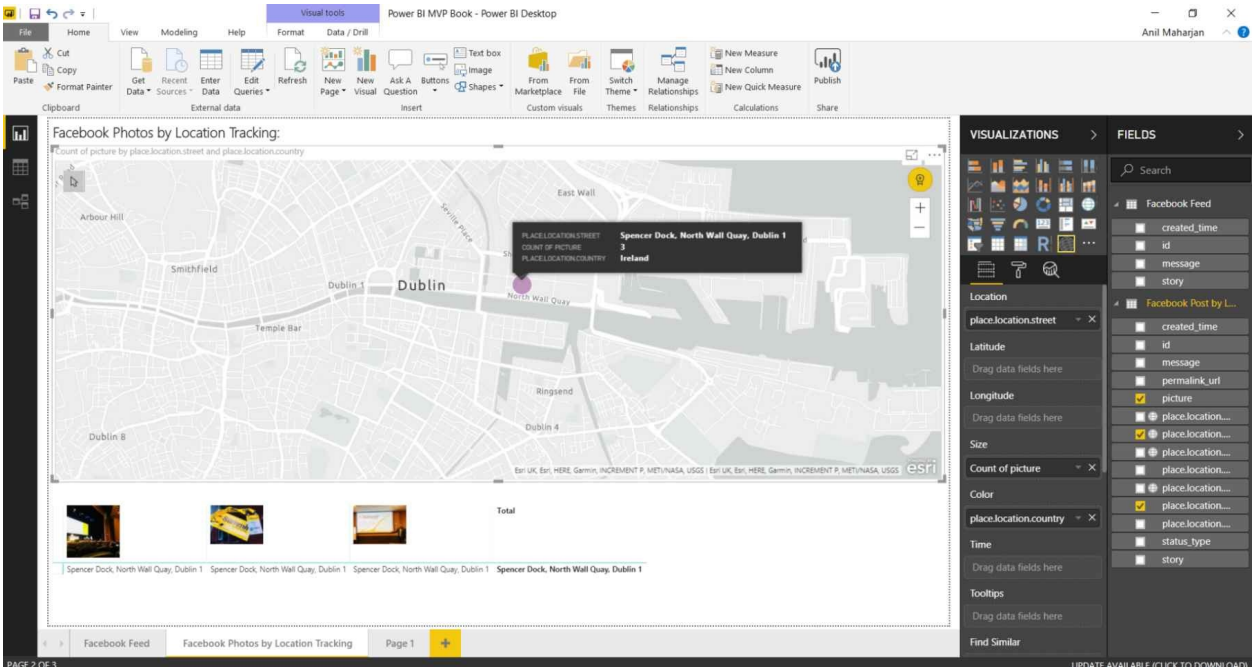


Figure 01-26: City level Drill Photos Image in Power BI Desktop

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/

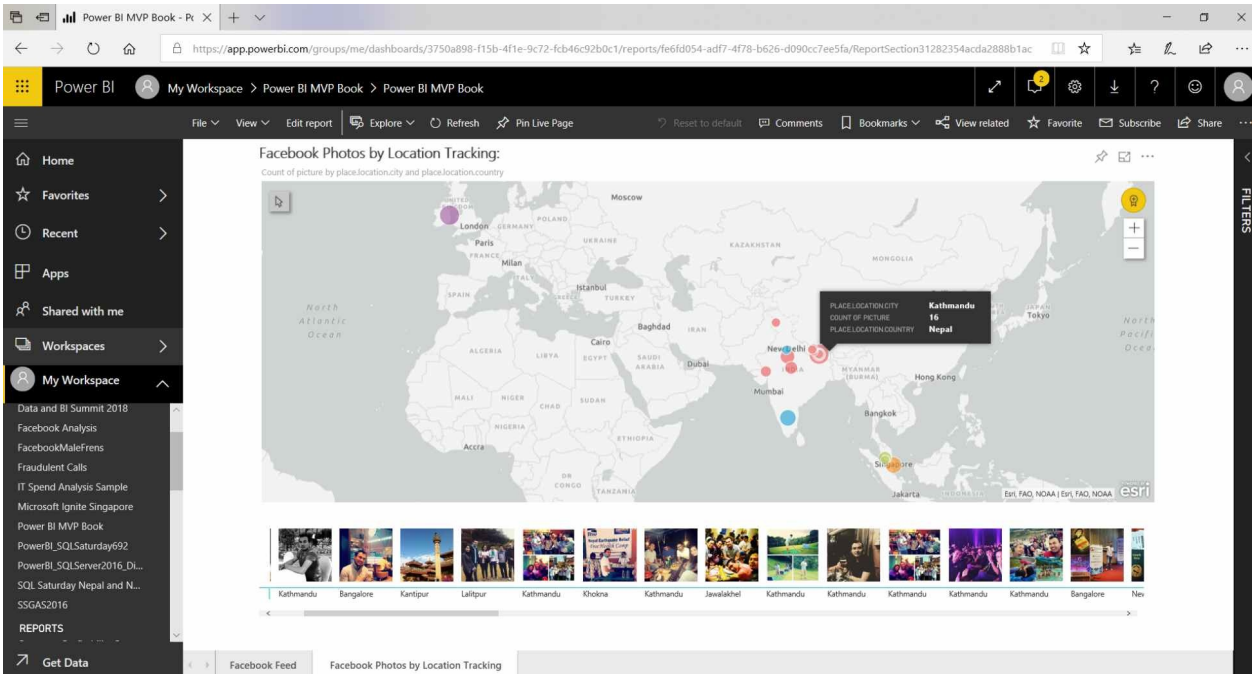


Figure 01-27: City level Drill Photos Dashboard in Power BI Service

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.

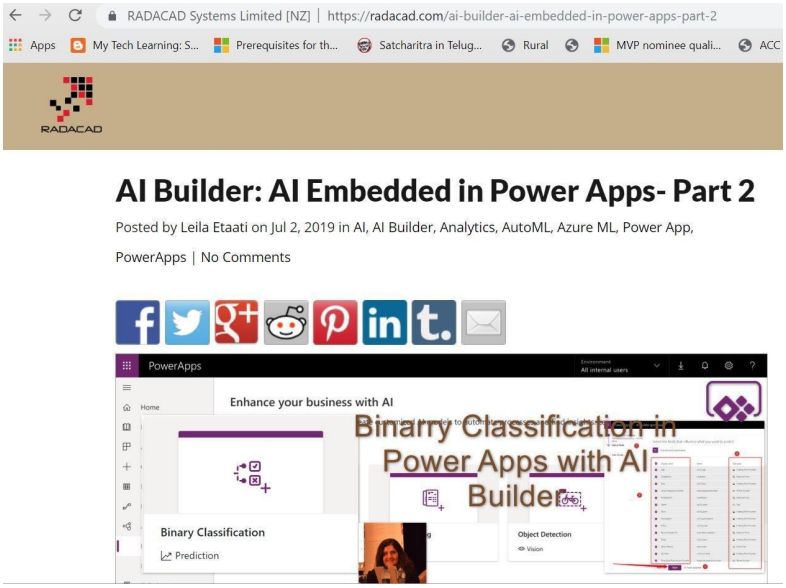


Figure 02-01: Blog URL

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:

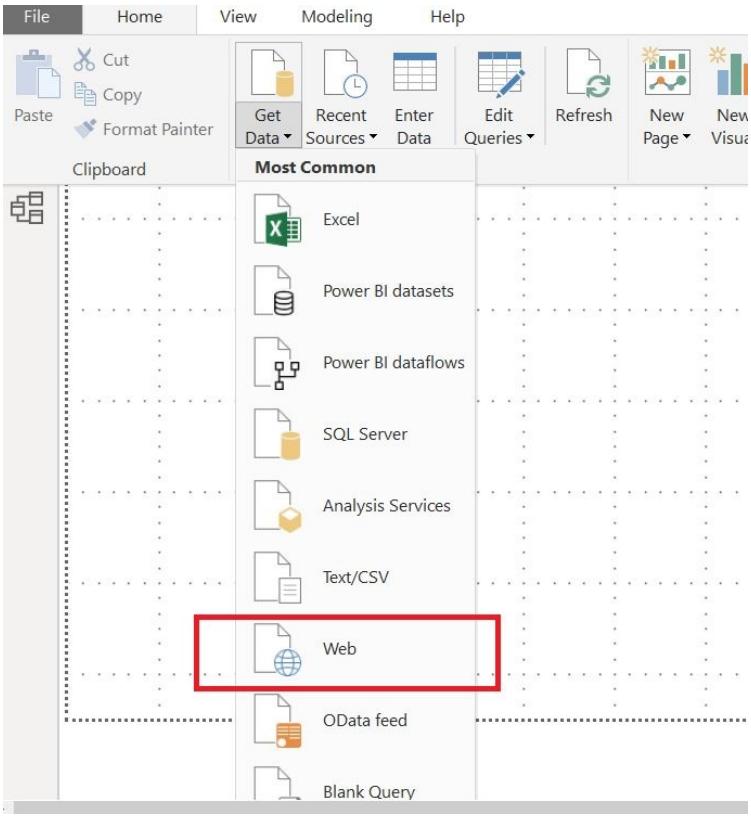


Figure 02-02: Get data from web button

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.

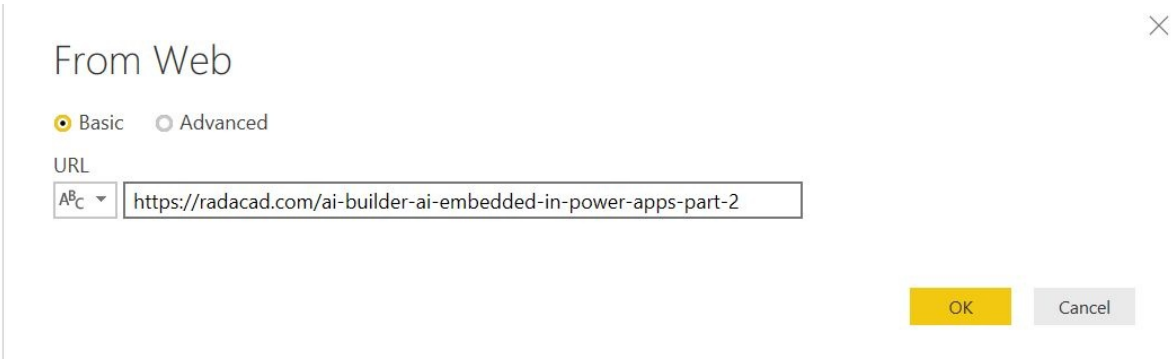


Figure 02-03: Get data from web -- provide URL

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).

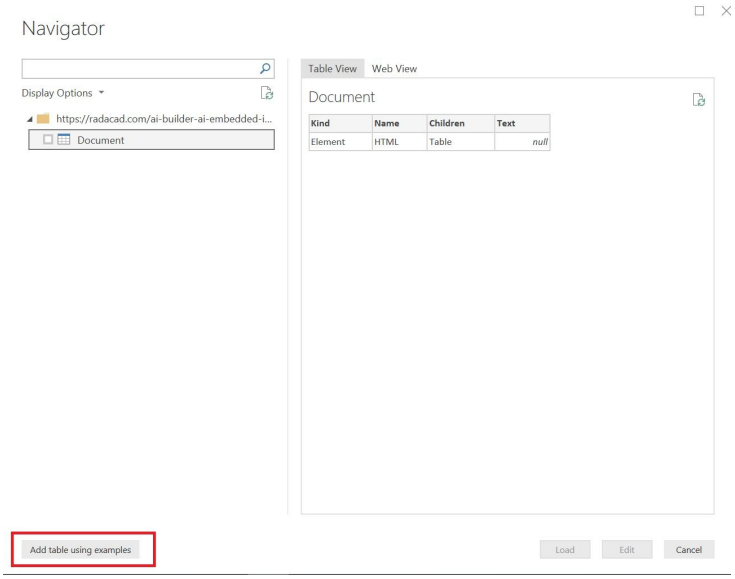


Figure 02-04: Add table using examples

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.

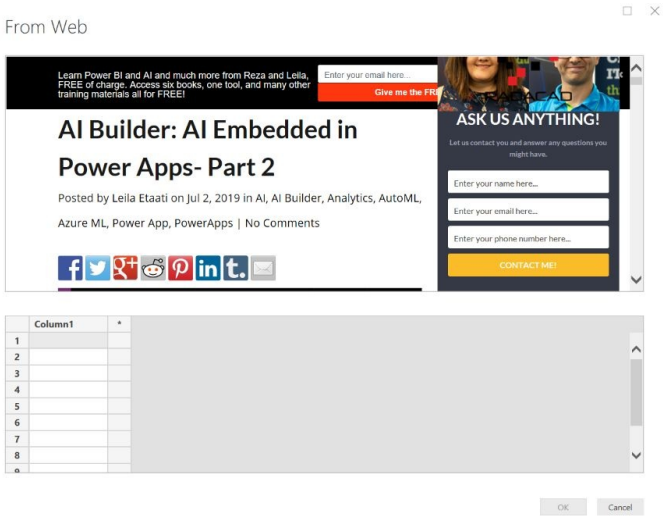


Figure 02-05: Add Column1

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

title of the blog and choose from the list displayed.

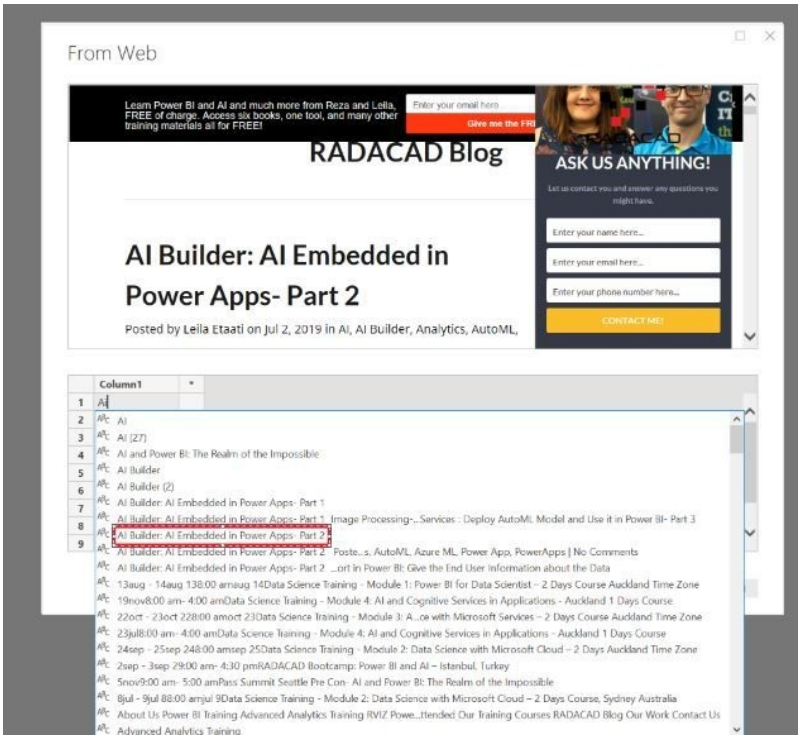


Figure 02-06: Choose Column1 from list

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

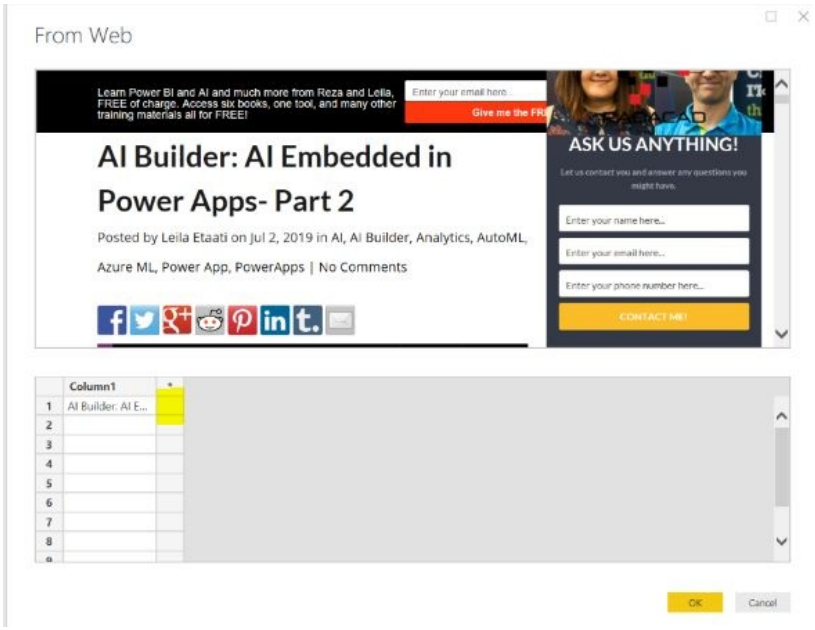


Figure 02-07: Click for Column2

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

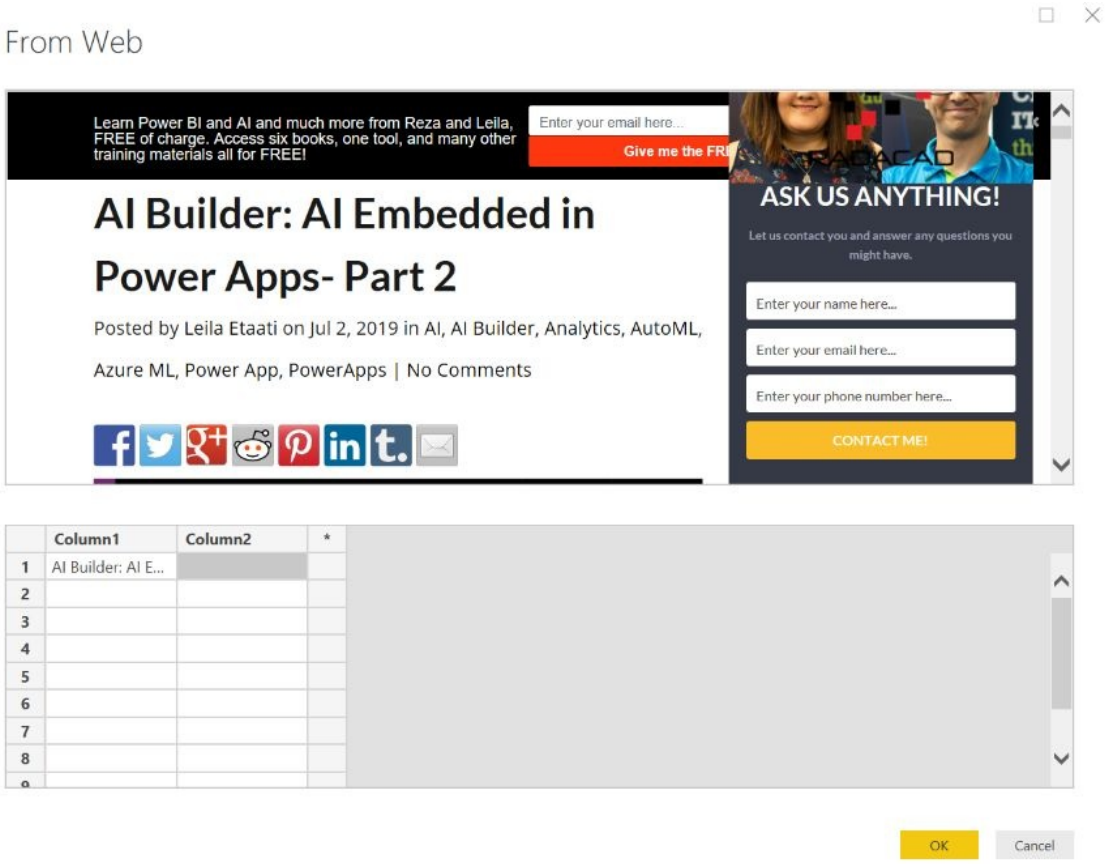


Figure 02-08: Add Column2

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

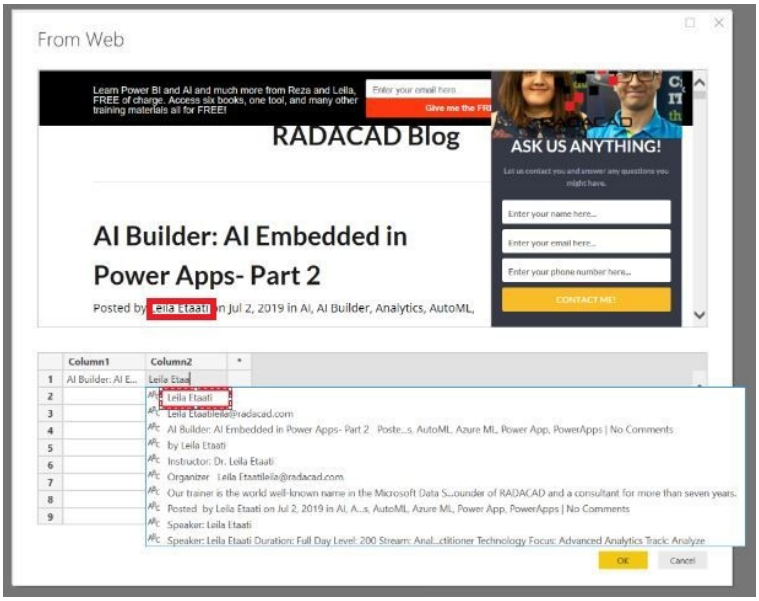


Figure 02-09: Choose Column2 from list

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.

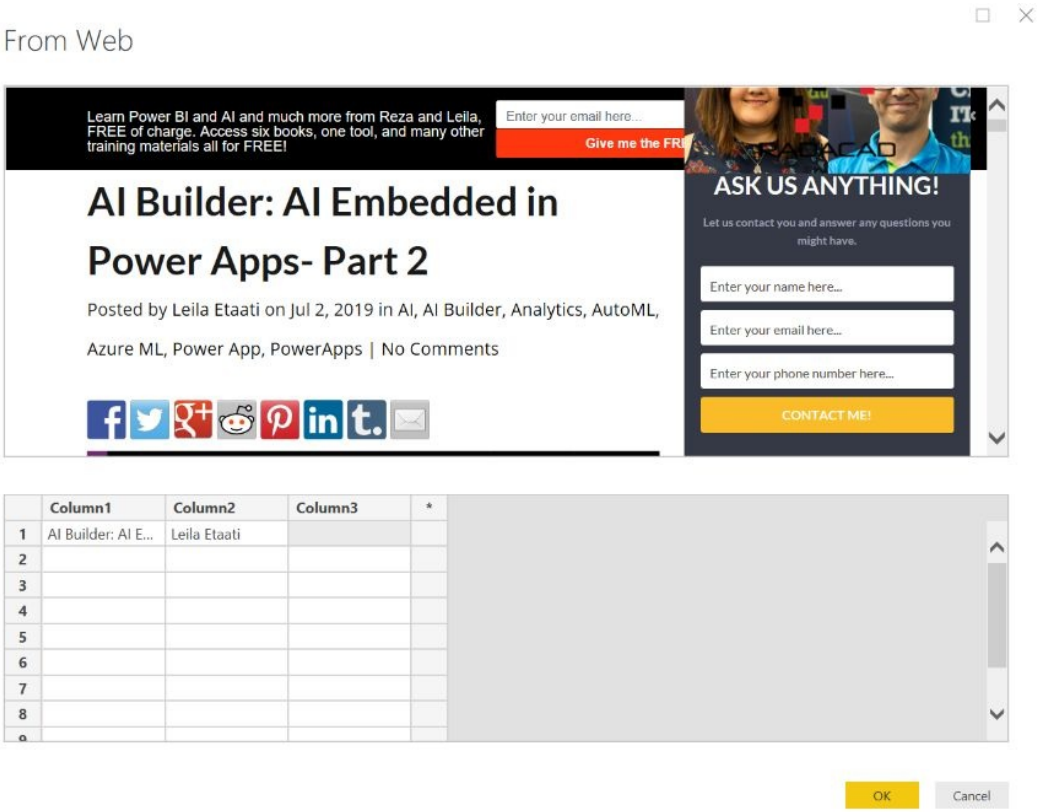


Figure 02-10: Add Column3

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

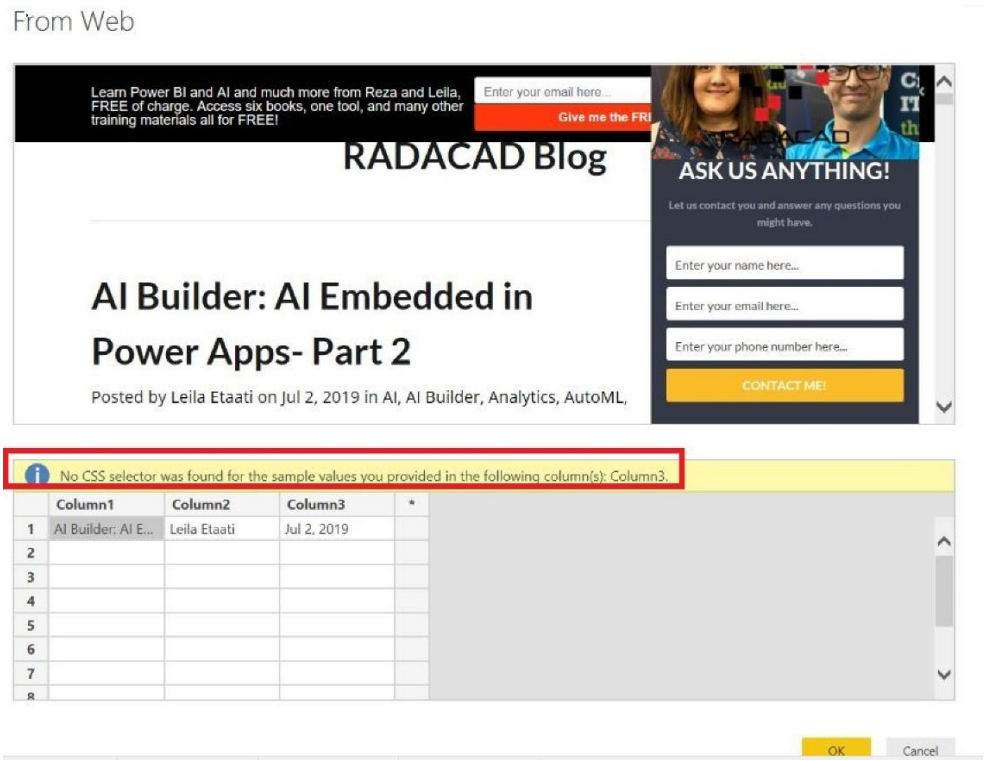


Figure 02-11: Column3 Error

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.

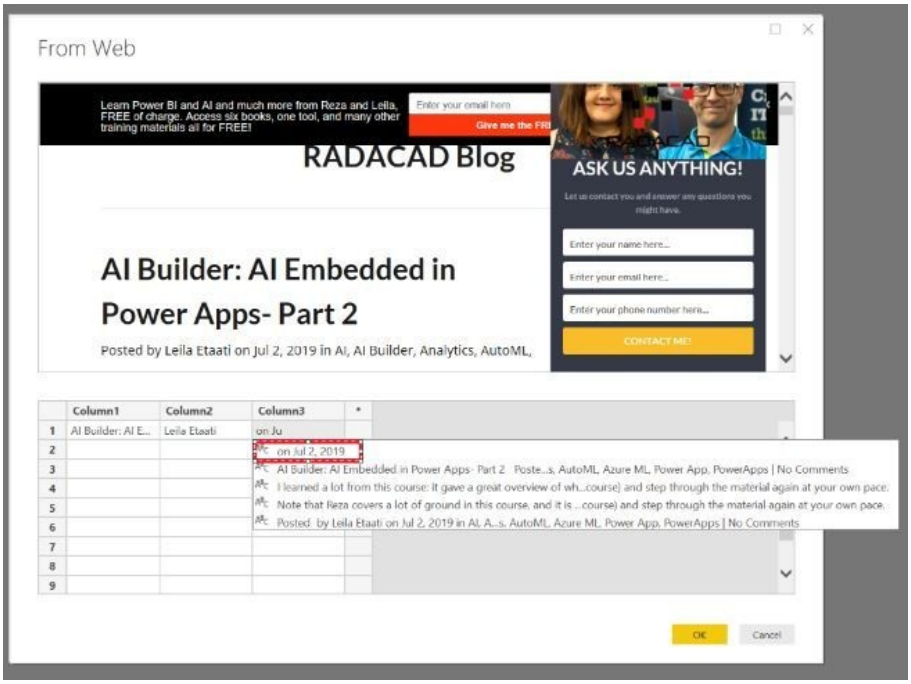


Figure 02-12: Choose Column3 from list

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

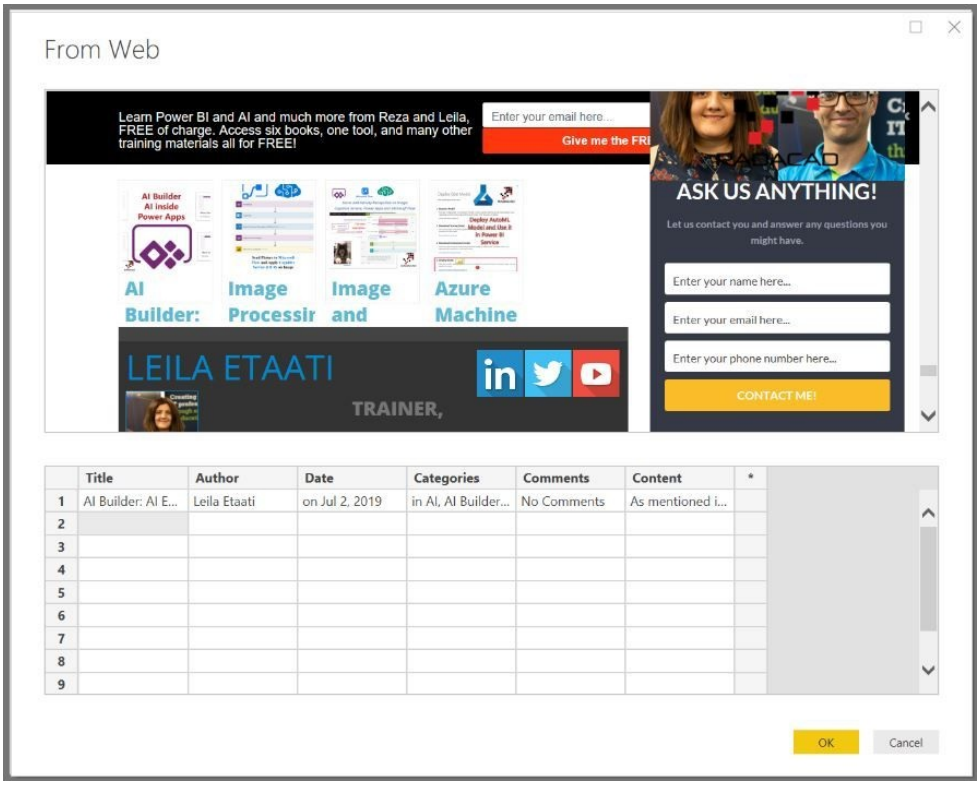


Figure 02-13: Populate remaining columns

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.

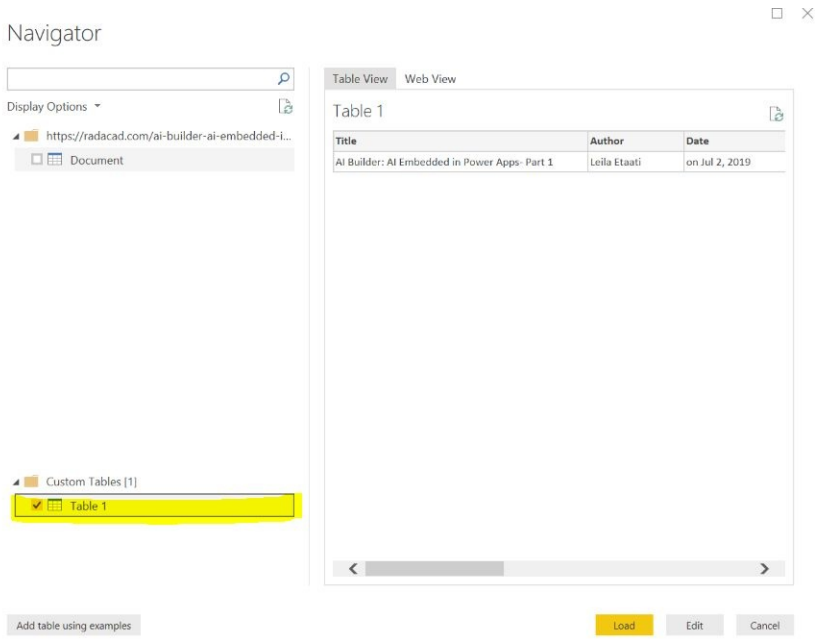


Figure 02-14: Add table from example Navigation Pane

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

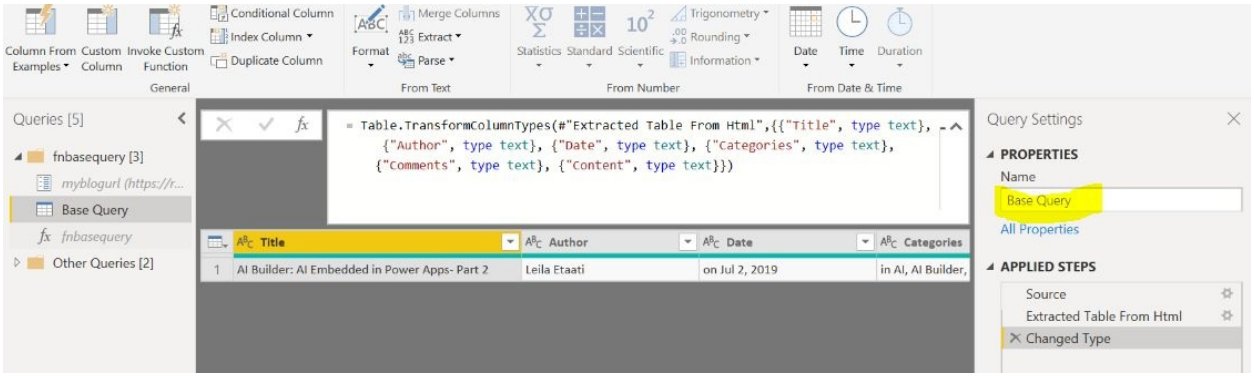


Figure 02-15: Base Query Table

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:

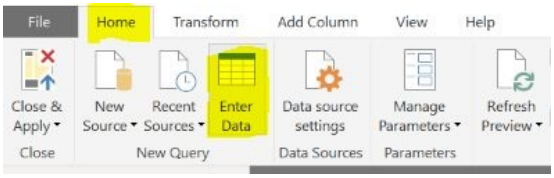


Figure 02-16: 'Enter Data' button in Power Query

The below screen appears:

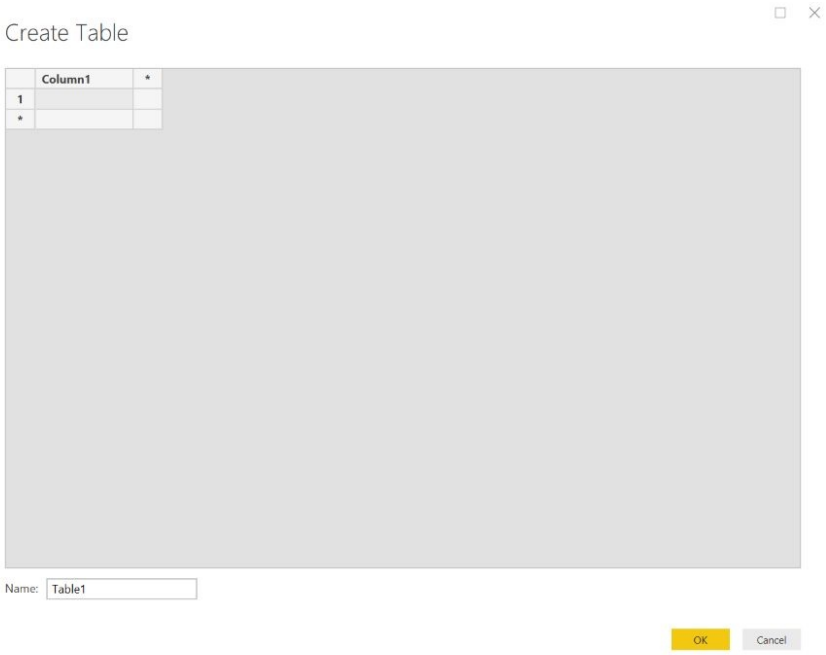


Figure 02-17: Create Table

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

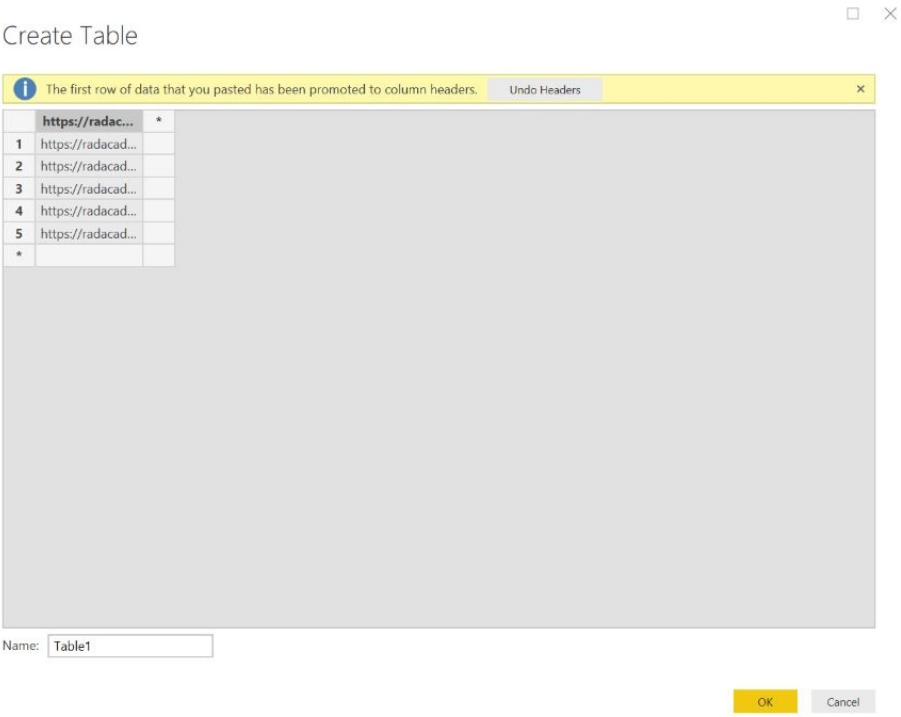


Figure 02-18: Populate Column1 with URLs

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.

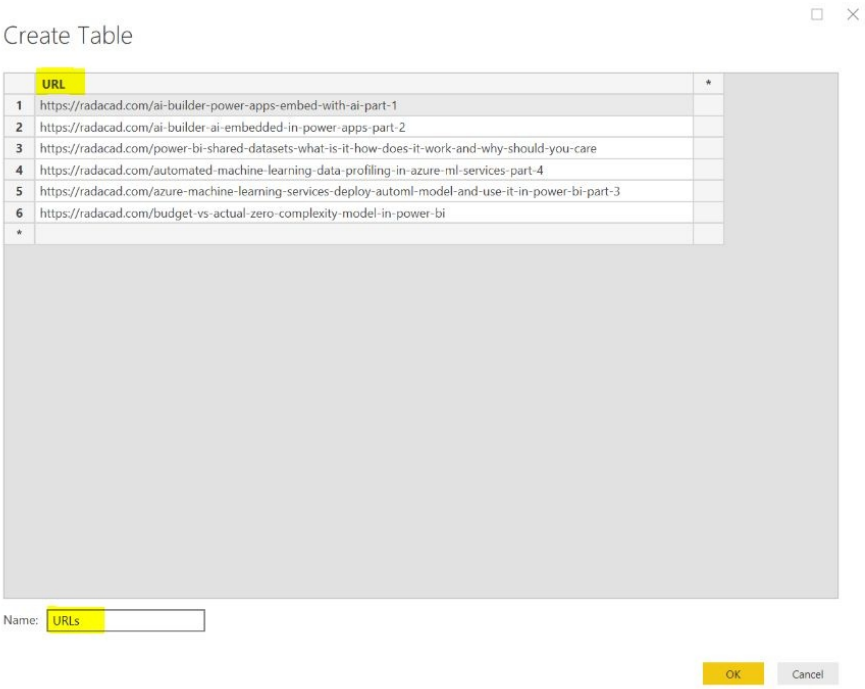


Figure 02-19: Rename Column Header

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:

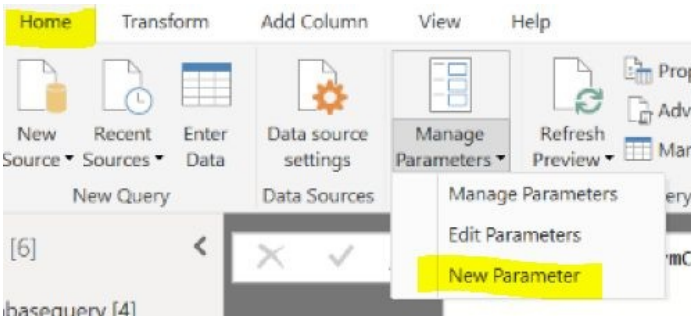


Figure 02-20: ‘Create New Parameter’ button

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.

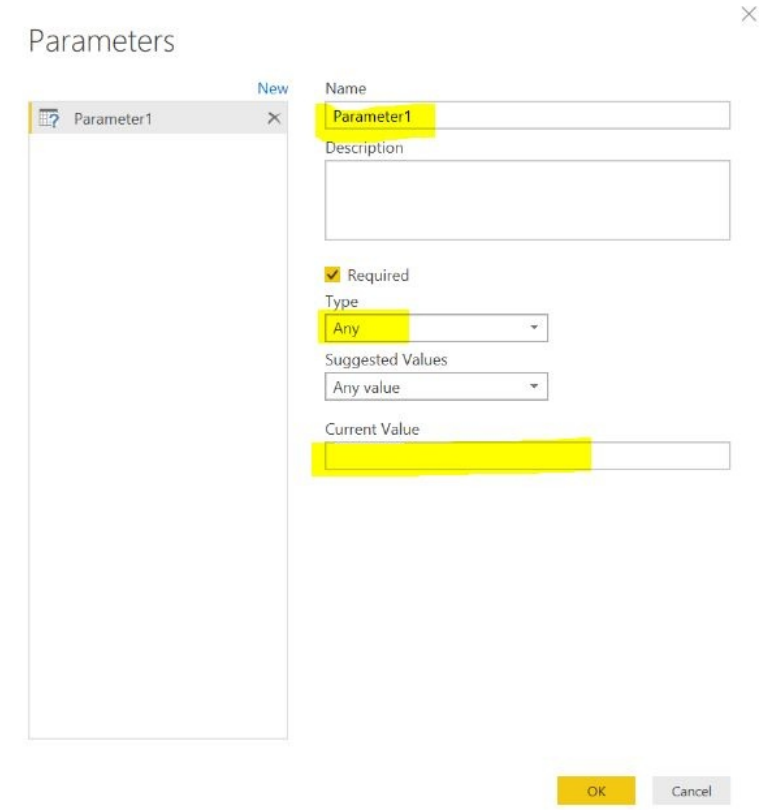


Figure 02-21: Parameters screen

Now the parameter is created. It looks as follows:

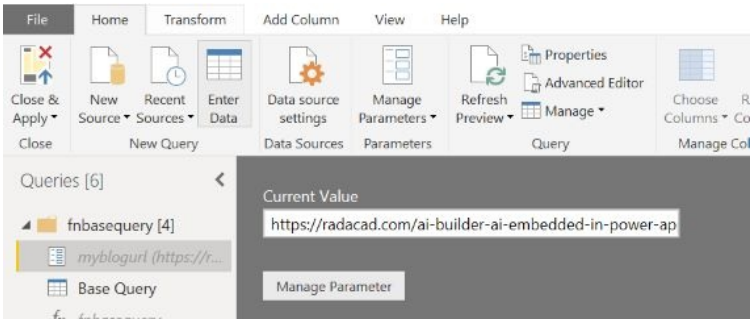


Figure 02-22: myblogurl Parameter created

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’:

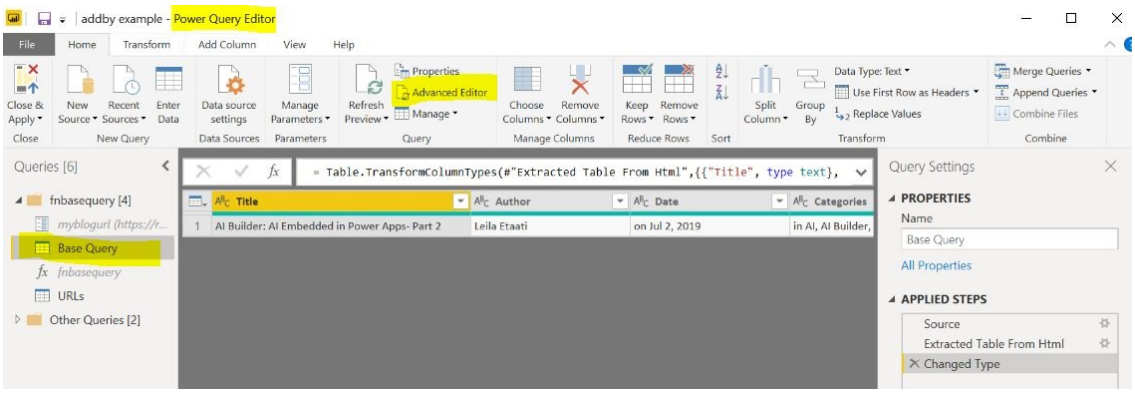


Figure 02-23: Advanced Editor button from Base Query

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'.

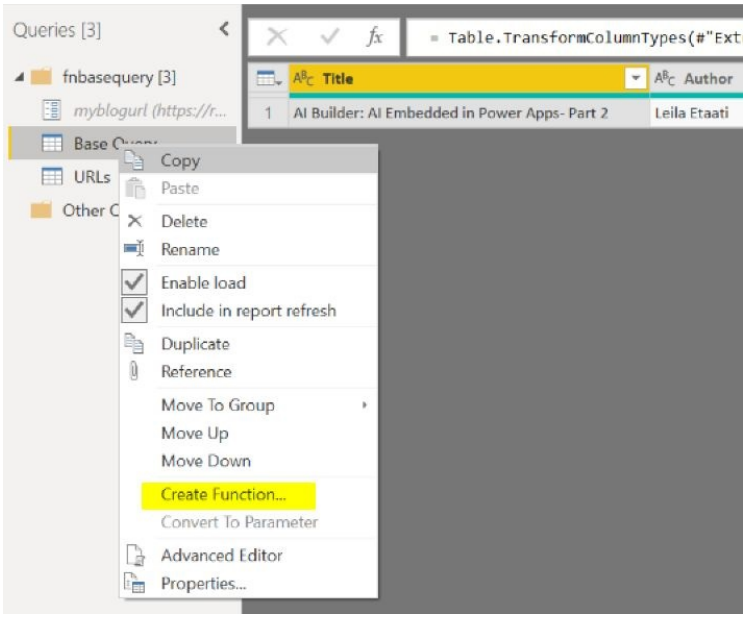


Figure 02-26: Create Function button

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:

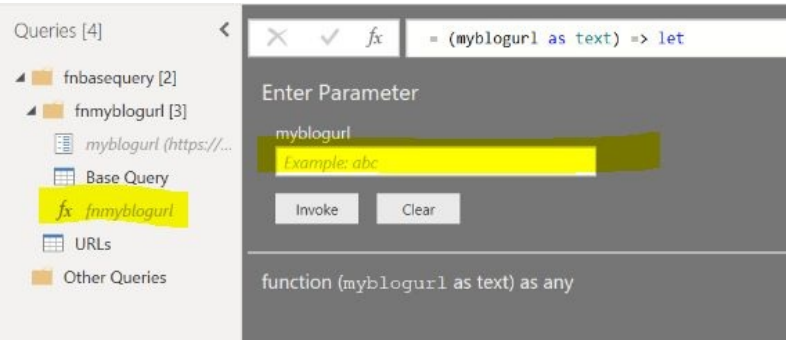


Figure 02-28: Function created

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:

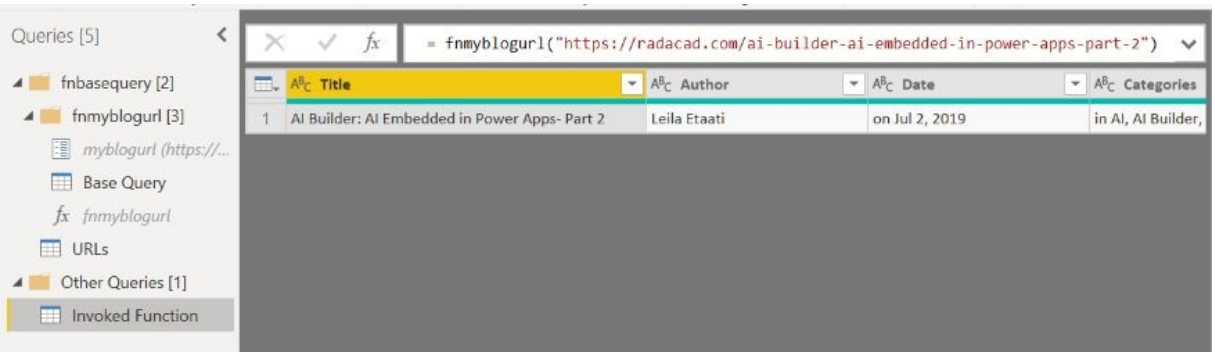


Figure 02-29: Invoked function

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’.

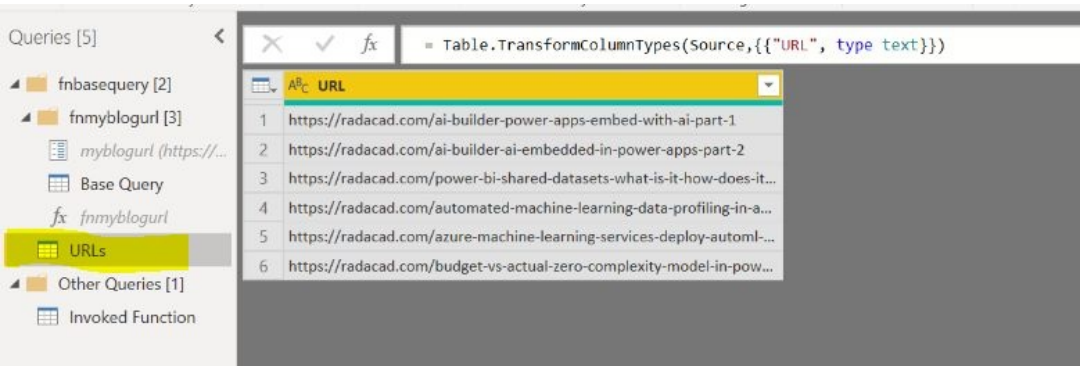


Figure 02-30: URLs table with URL column

Now we will used 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’.

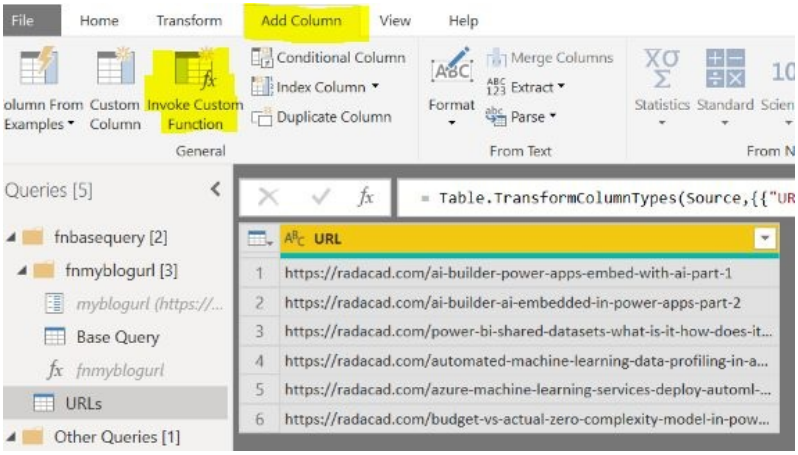


Figure 02-31: Add Column using Invoke Custom Function

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.

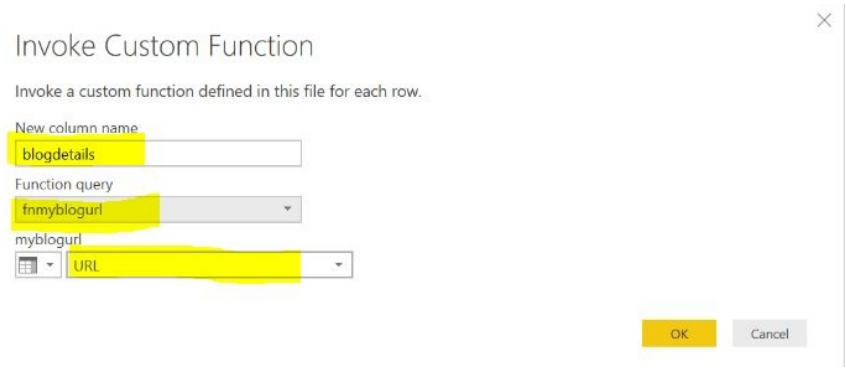


Figure 02-32: Invoke Custom Function dialog

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

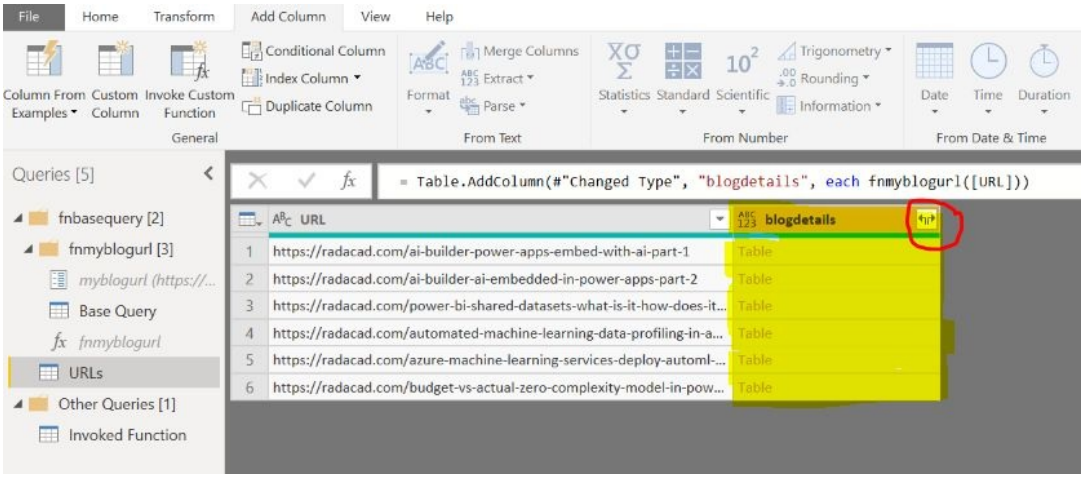


Figure 02-33: New column in URLs table

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.

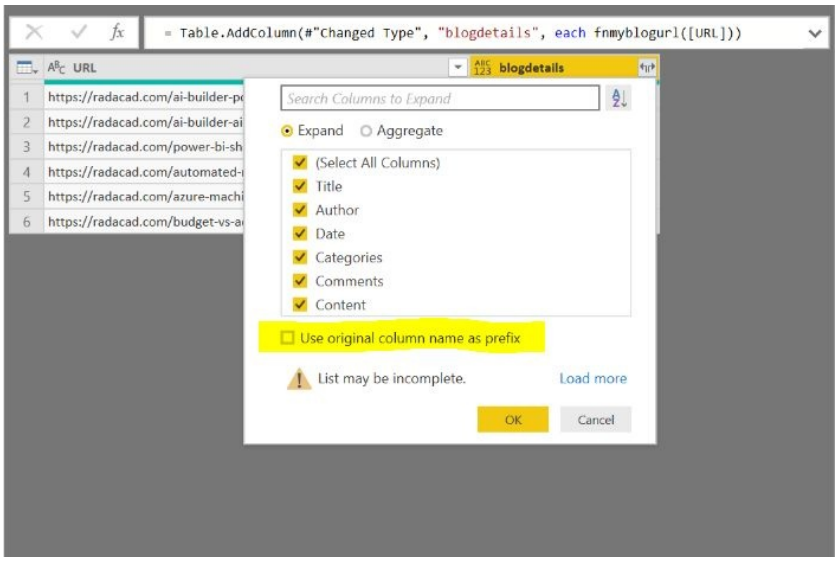


Figure 02-34: Expand button

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.

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.

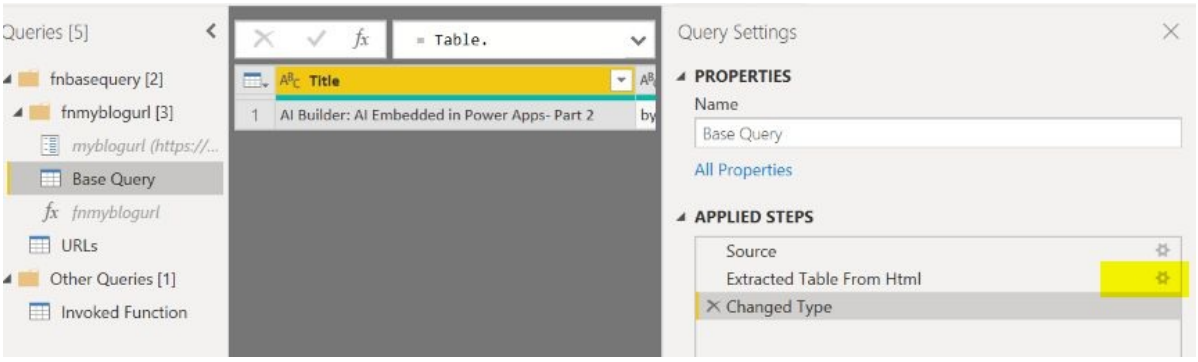


Figure 02-36: Extracted Table from HTML -- settings

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

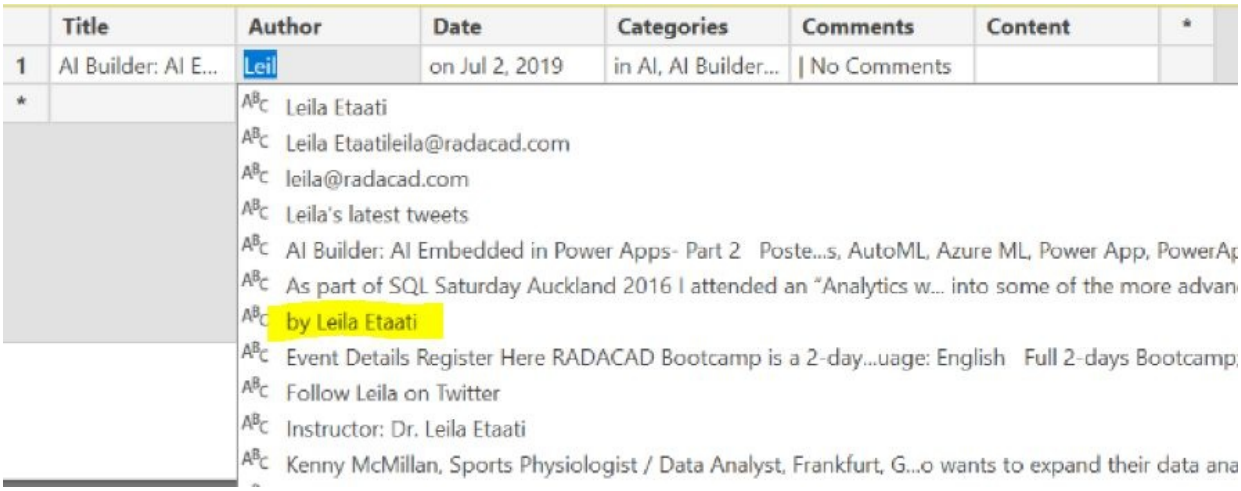


Figure 02-37: Repopulate Author column

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

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.

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:

1 2 3 ... 8 9 ... 15 16 17

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.

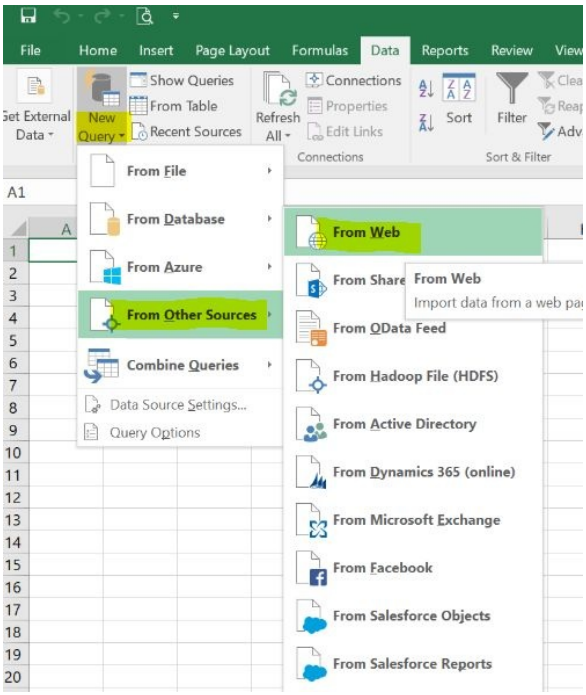
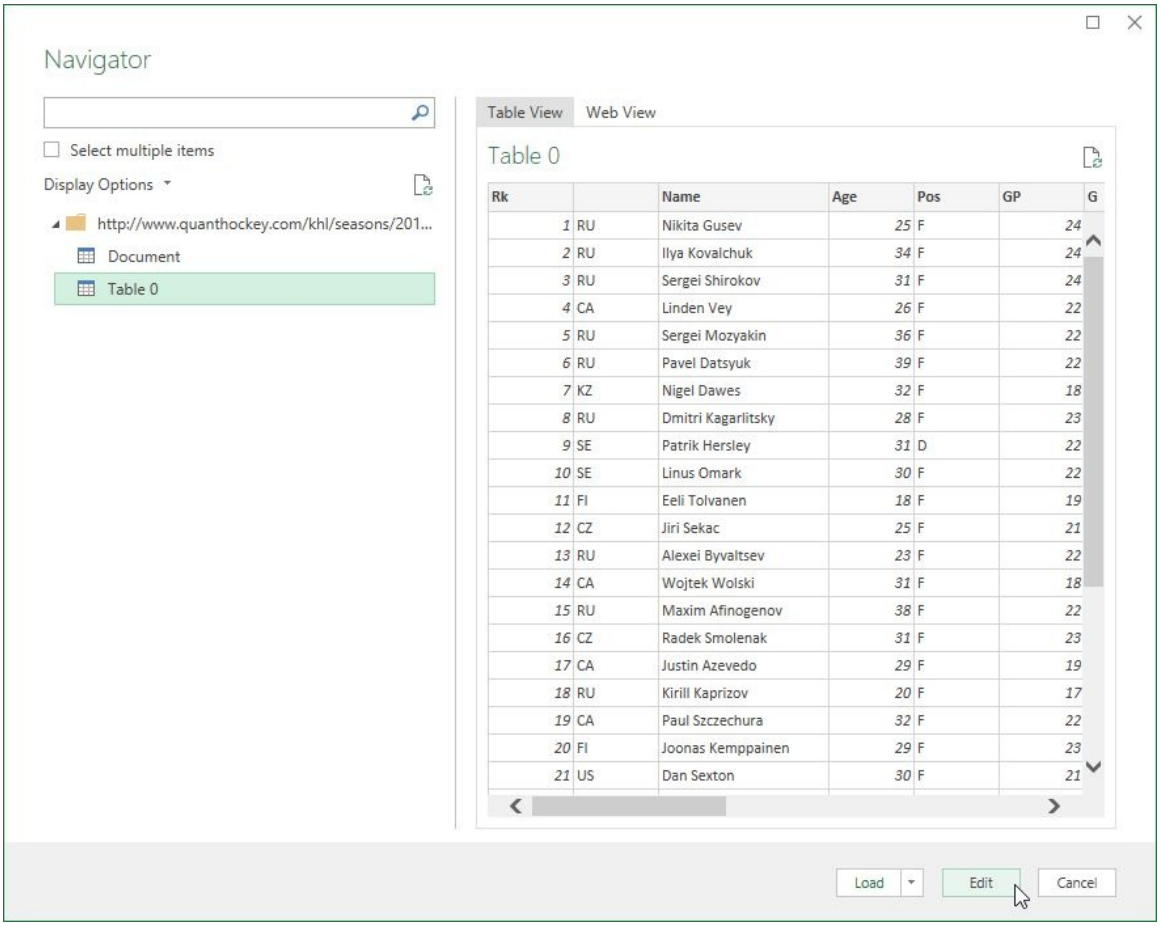


Figure 03-02: New Query

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’:



Rk		Name	Age	Pos	GP	G
1	RU	Nikita Gusev	25	F	24	
2	RU	Ilya Kovalchuk	34	F	24	
3	RU	Sergei Shirokov	31	F	24	
4	CA	Linden Vey	26	F	22	
5	RU	Sergei Mozyakin	36	F	22	
6	RU	Pavel Datsyuk	39	F	22	
7	KZ	Nigel Dawes	32	F	18	
8	RU	Dmitri Kagarlitsky	28	F	23	
9	SE	Patrik Hersley	31	D	22	
10	SE	Linus Omark	30	F	22	
11	FI	Eeli Tolvanen	18	F	19	
12	CZ	Jiri Sekac	25	F	21	
13	RU	Alexei Byvaltsev	23	F	22	
14	CA	Wojtek Wolski	31	F	18	
15	RU	Maxim Afinogenov	38	F	22	
16	CZ	Radek Smolenak	31	F	23	
17	CA	Justin Azevedo	29	F	19	
18	RU	Kirill Kaprizov	20	F	17	
19	CA	Paul Szczechura	32	F	22	
20	FI	Joonas Kemppainen	29	F	23	
21	US	Dan Sexton	30	F	21	

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’:

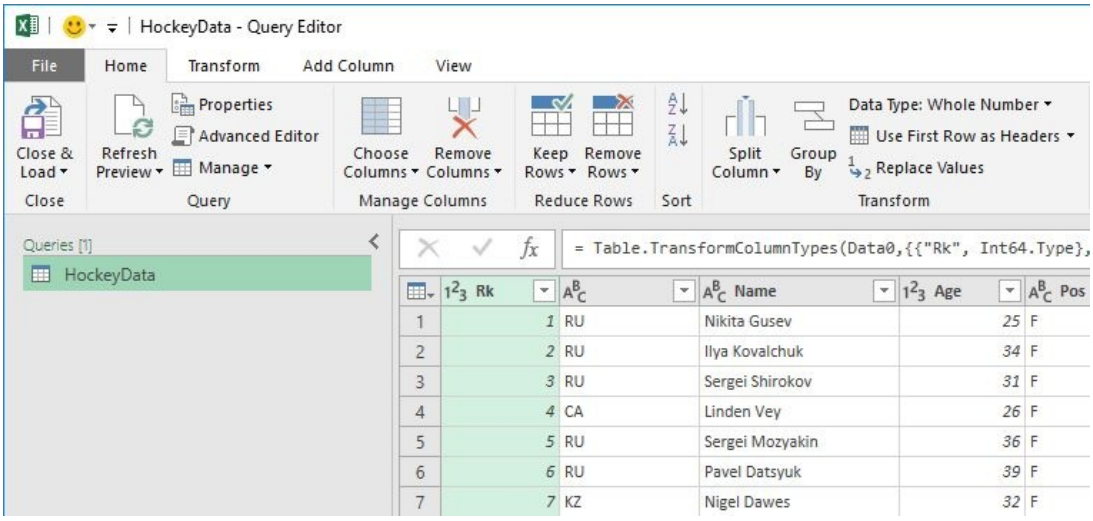


Figure 03-05: HockeyData

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

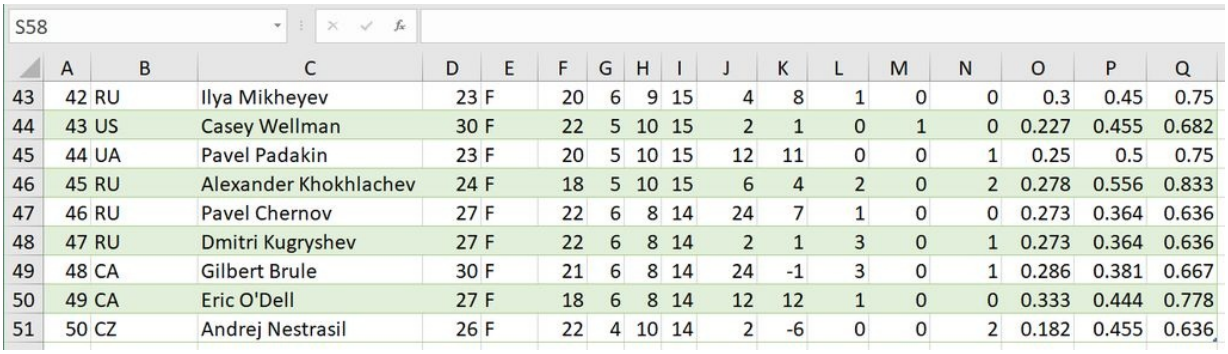


Figure 03-06: First 50 Entries Only

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':

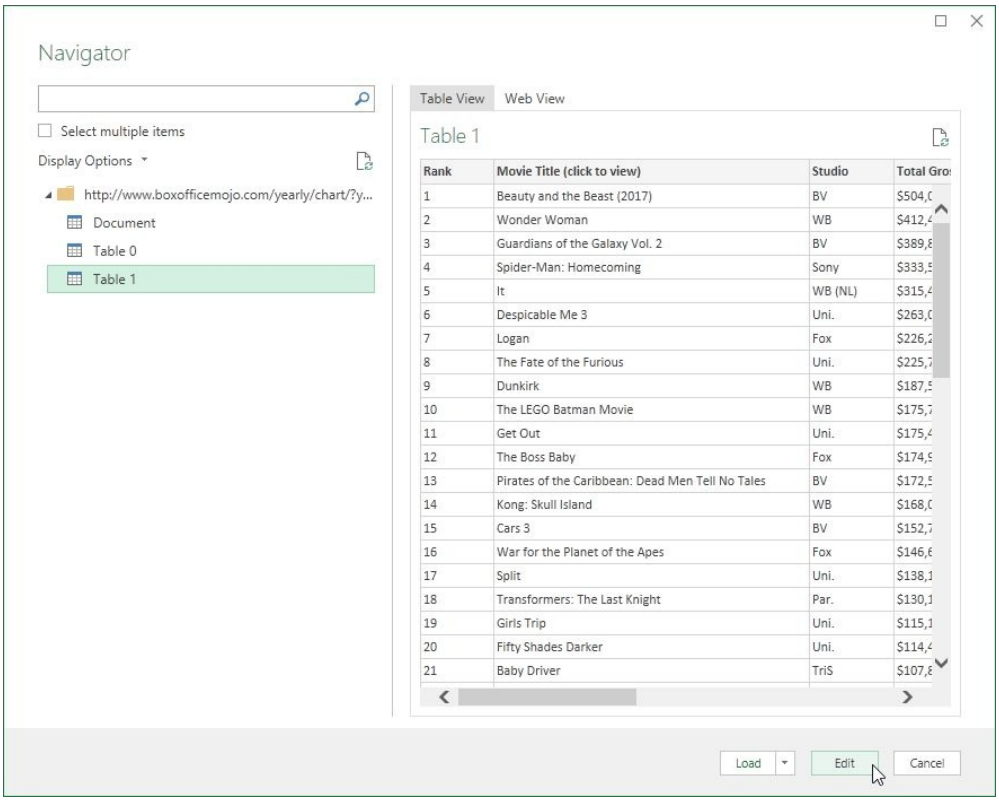


Figure 03-08: Editing Table 1

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

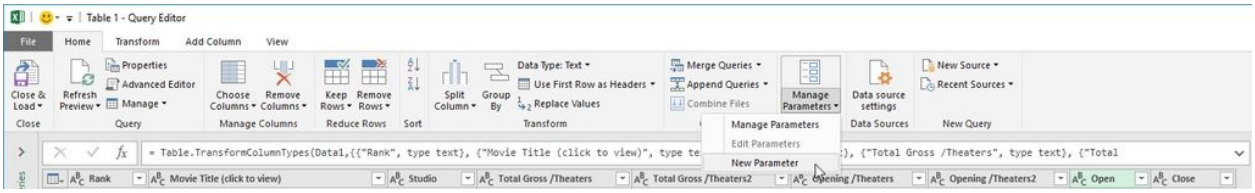


Figure 03-09: Creating a New Parameter

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

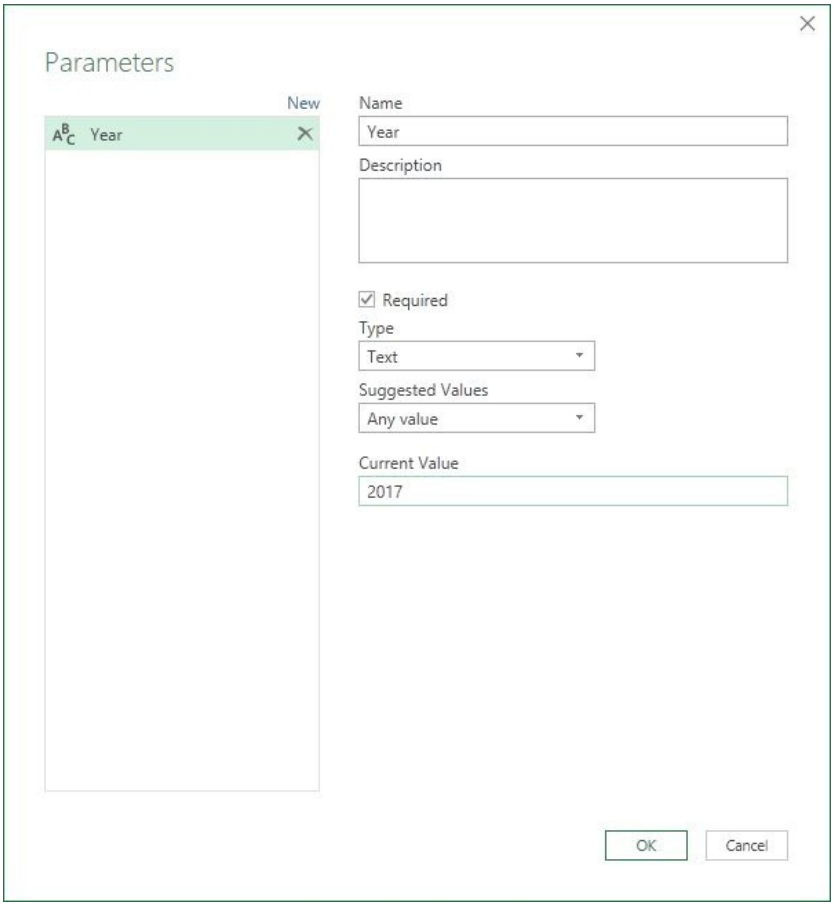


Figure 03-10: Parameters Dialog

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

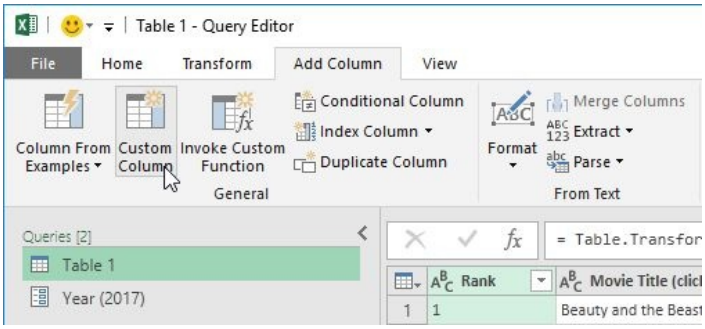


Figure 03-11: Custom Column

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

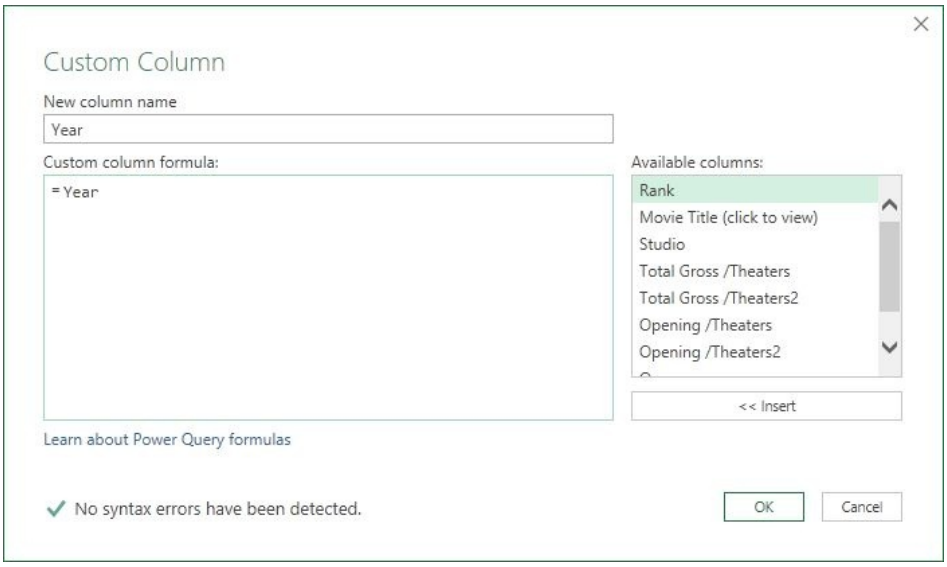


Figure 03-12: Creating the Parameter 'Year'

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

Open	Close	Year
3/17	7/13	2017
6/2	-	2017
5/5	9/21	2017
7/7	-	2017
9/8	-	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:

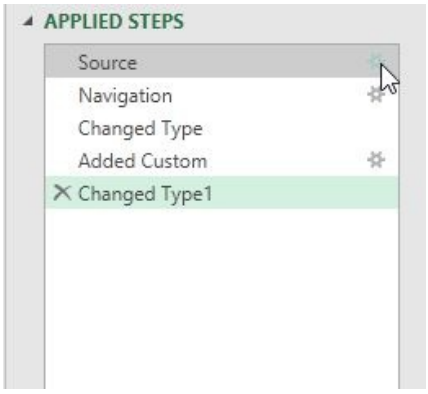


Figure 03-14: Applied Steps

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.

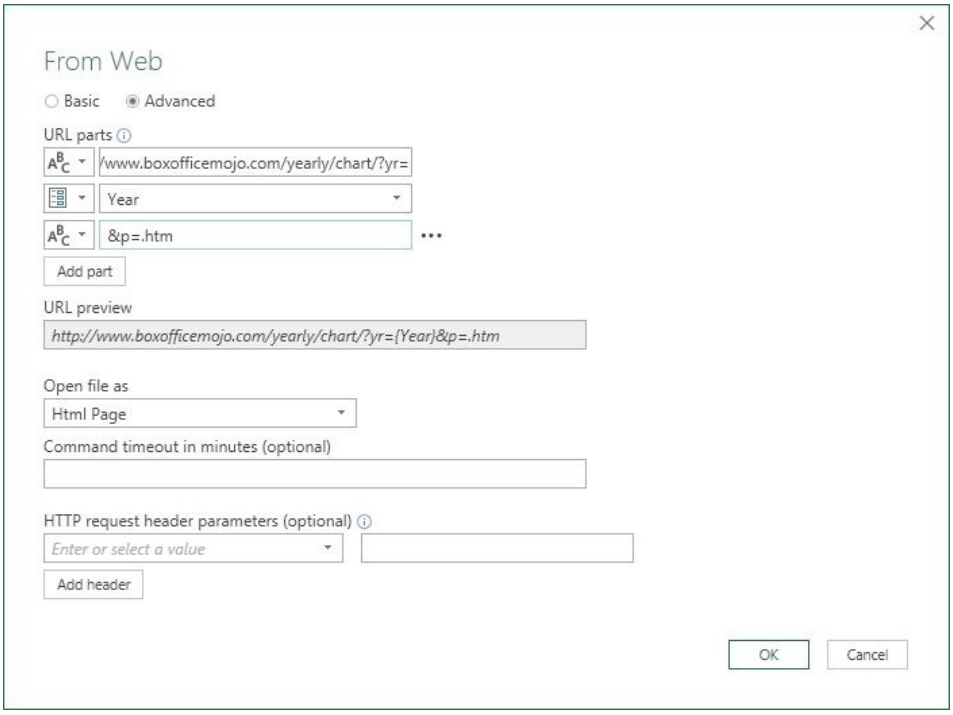


Figure 03-15: Modifying the URL

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…':

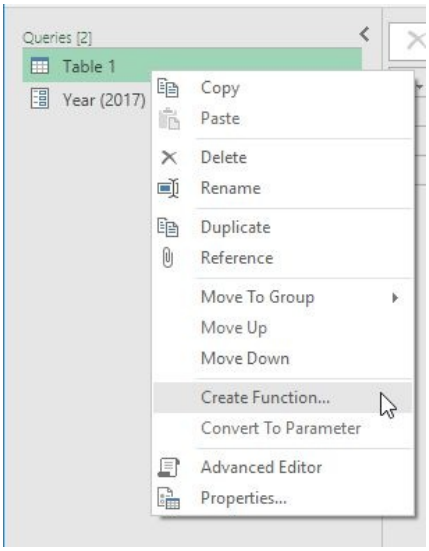


Figure 03-16: Creating a Function

Name the function 'GetMovies' then click 'OK'.

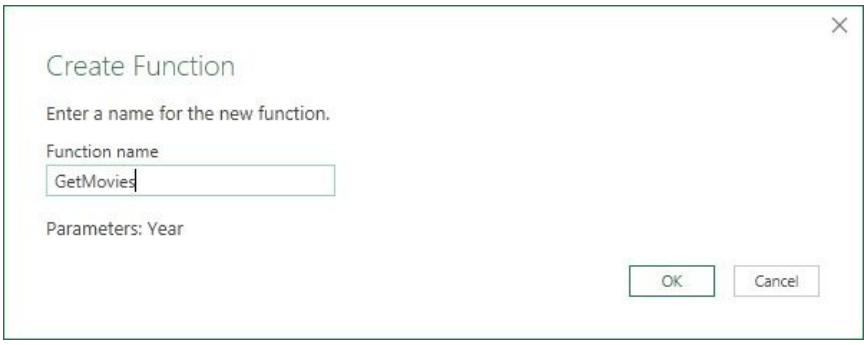


Figure 03-17: Naming the Function

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

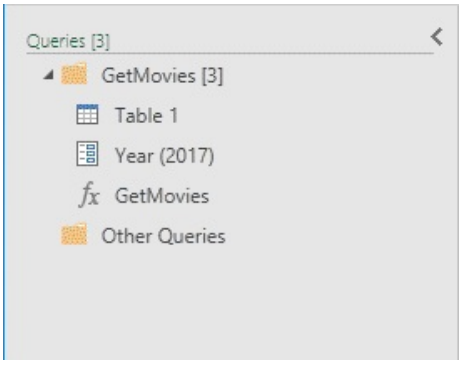


Figure 03-18: Noting the Three Queries

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**.

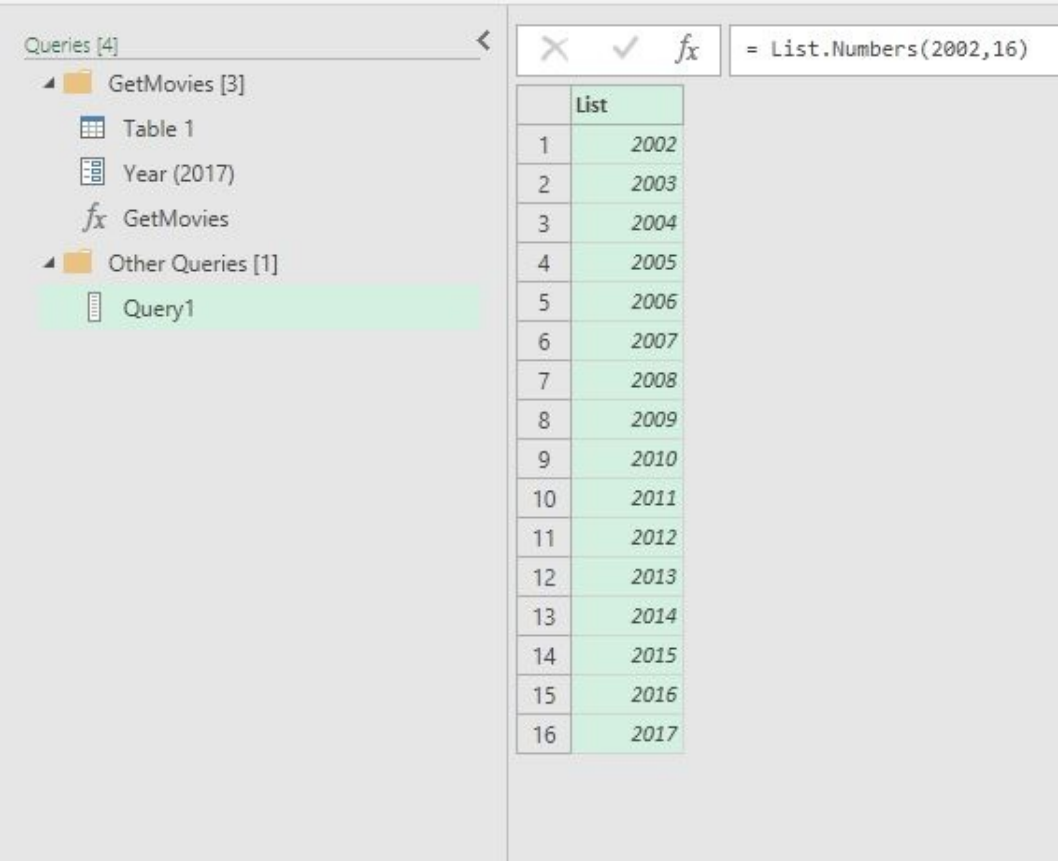


Figure 03-19: List.Numbers

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:

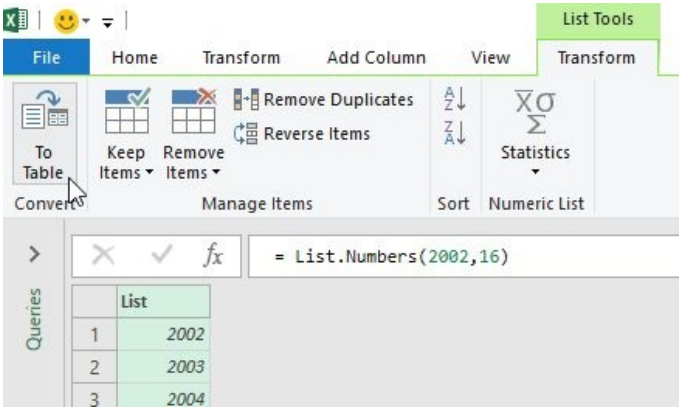


Figure 03-20: Convert to Table

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

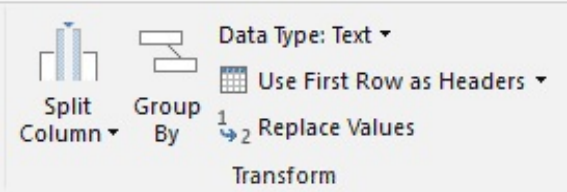


Figure 03-21: Data Type Text

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:

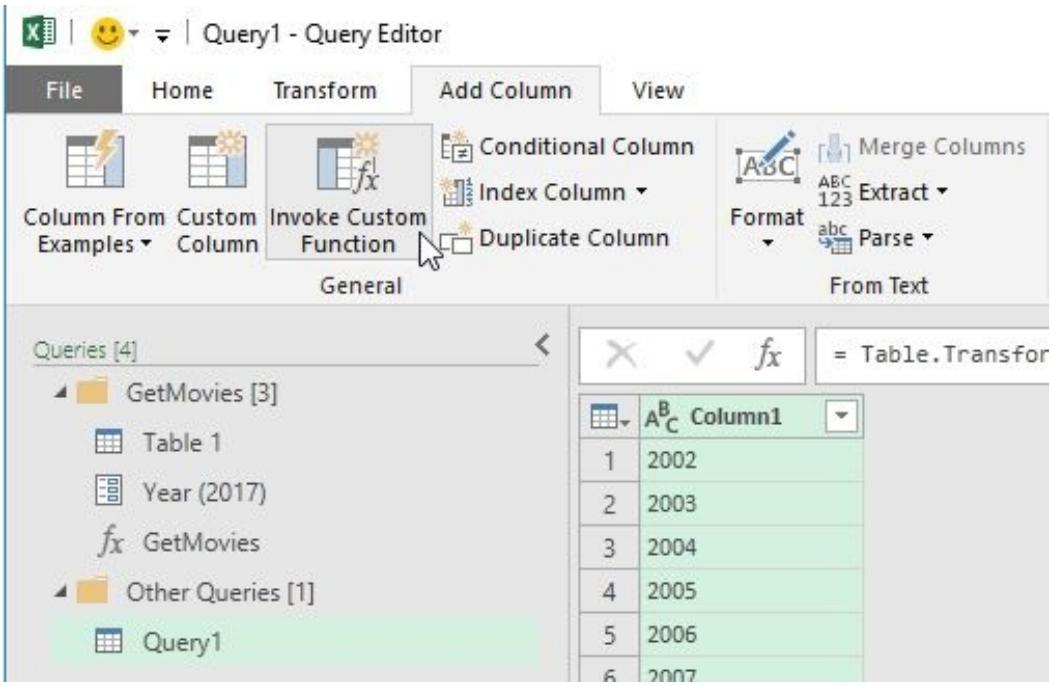


Figure 03-22: Invoke Custom Function

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

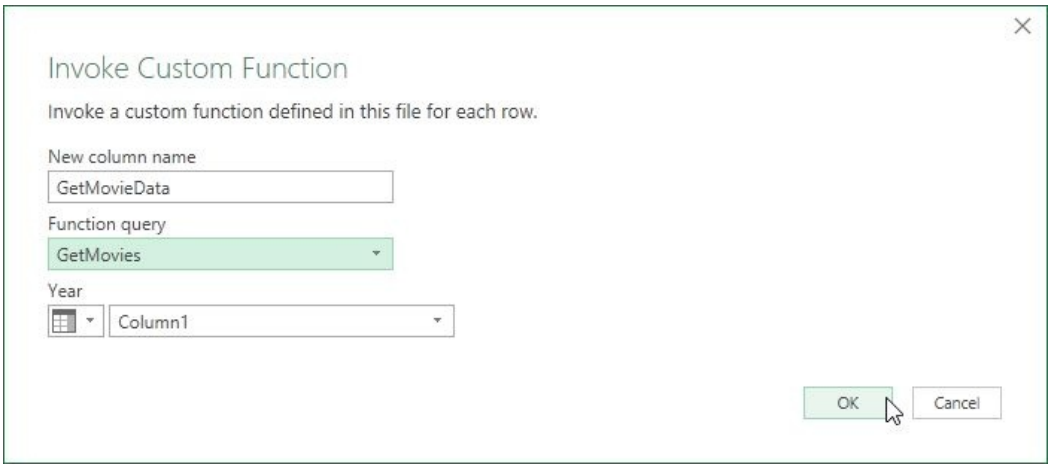


Figure 03-23: Invoke Custom Function Dialog

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.*

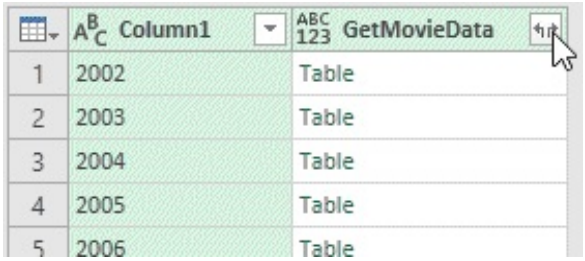
	Rank	Movie Title (click to view)	Studio	Total Gross /Theaters	Total Gross /Theaters2	Opening /Theaters	Opening /Theaters2
1	1	Pirates of the Caribbean: Dead Man's Chest	BV	$423,315,812	4,133	$135,634,554	4,133
2	2	Night at the Museum	Fox	$250,863,268	3,768	$30,433,781	3,685
3	3	Cars	BV	$244,082,982	3,988	$60,119,509	3,985
4	4	X-Men: The Last Stand	Fox	$234,362,462	3,714	$102,750,665	3,690
5	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.

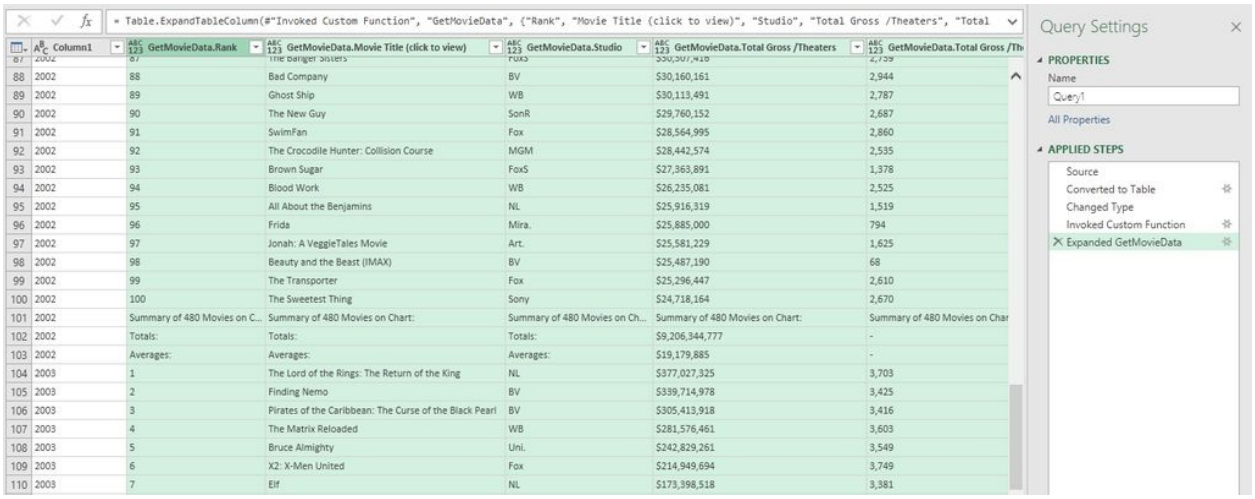
This shouldn't detract from the benefits. Moving on, click on the 'Expand' option as shown:



	Column1	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:



	Column1	GetMovieData.Rank	GetMovieData.Movie Title (click to view)	GetMovieData.Studio	GetMovieData.Total Gross /Theaters	GetMovieData.Total Gross /Th
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	$27,363,891	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 C...	Summary of 480 Movies on Chart:	Summary of 480 Movies on Ch...	Summary of 480 Movies on Chart:	Summary of 480 Movies on Char
102	2002	Totals:	Totals:	Totals:	$9,206,344,777	-
103	2002	Averages:	Averages:	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:

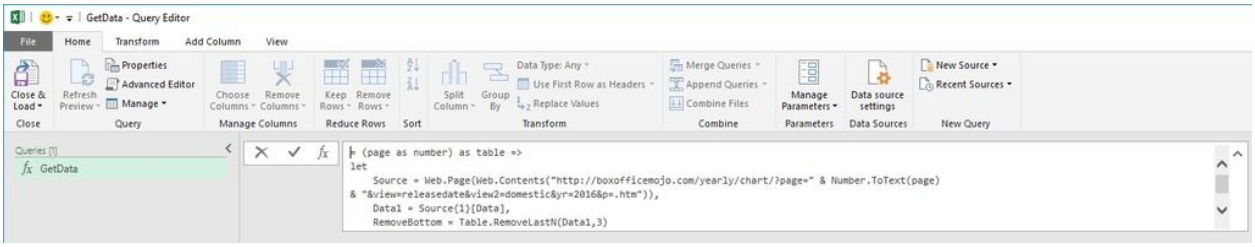


Figure 03-28: GetData Query

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.

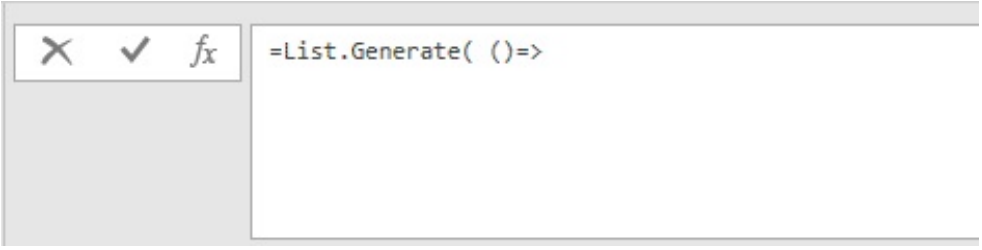


Figure 03-29: Building Up the Code #1

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:

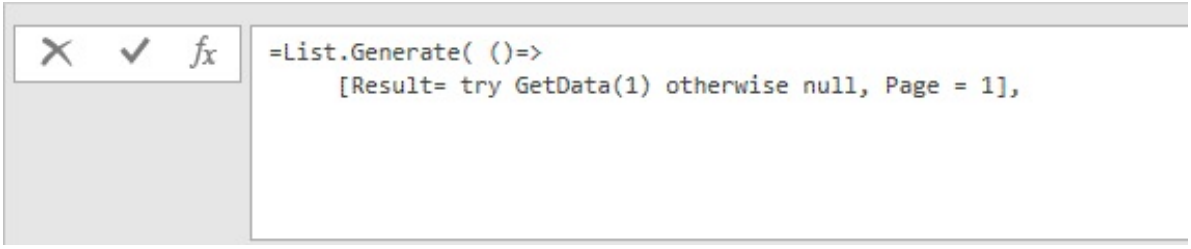


Figure 03-30: Building Up the Code #2

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.

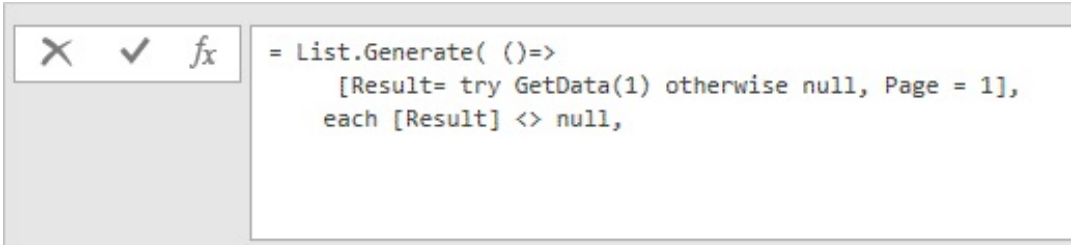


Figure 03-31: Building Up the Code #3

The next line increments the page to page 2:

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

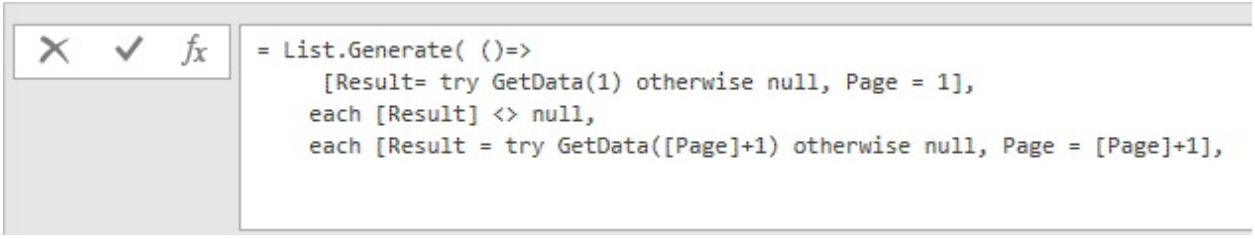


Figure 03-32: Building Up the Code #4

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:

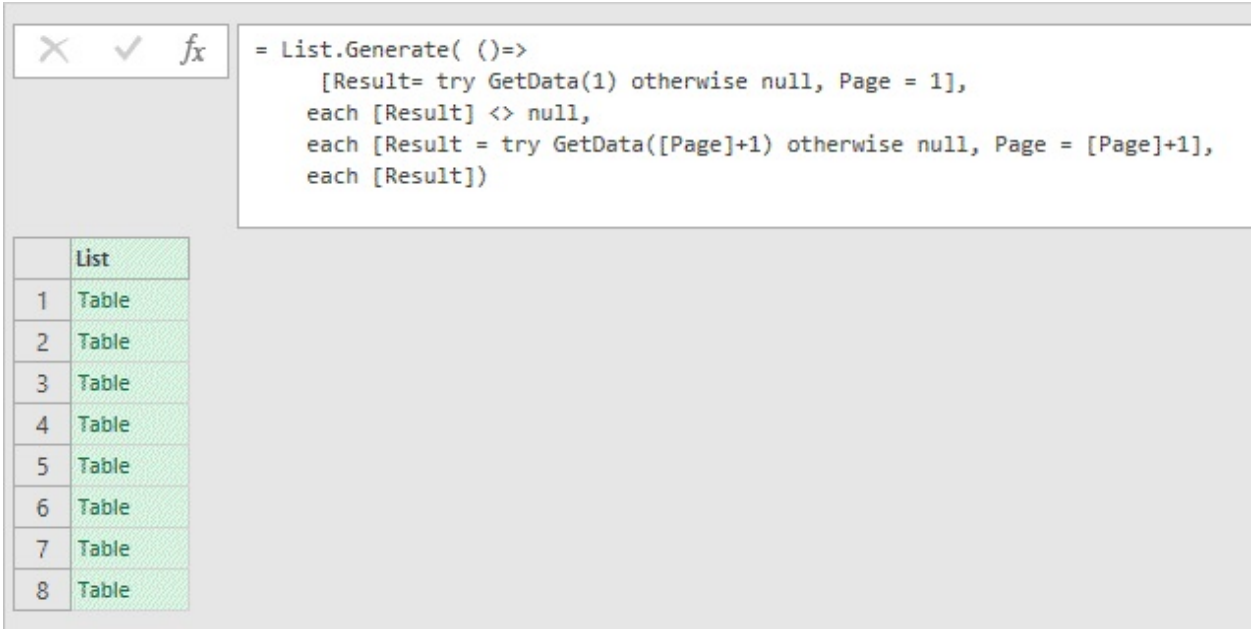


Figure 03-33: Building Up the Code #5

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:

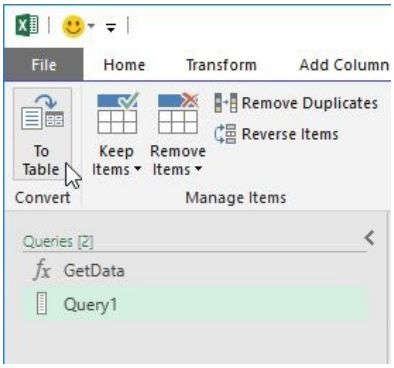


Figure 03-34: Transform to Table

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

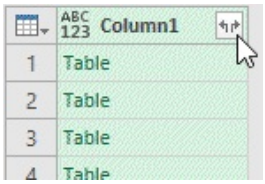


Figure 03-35: Expand the Table

The expanded table should look something like this:

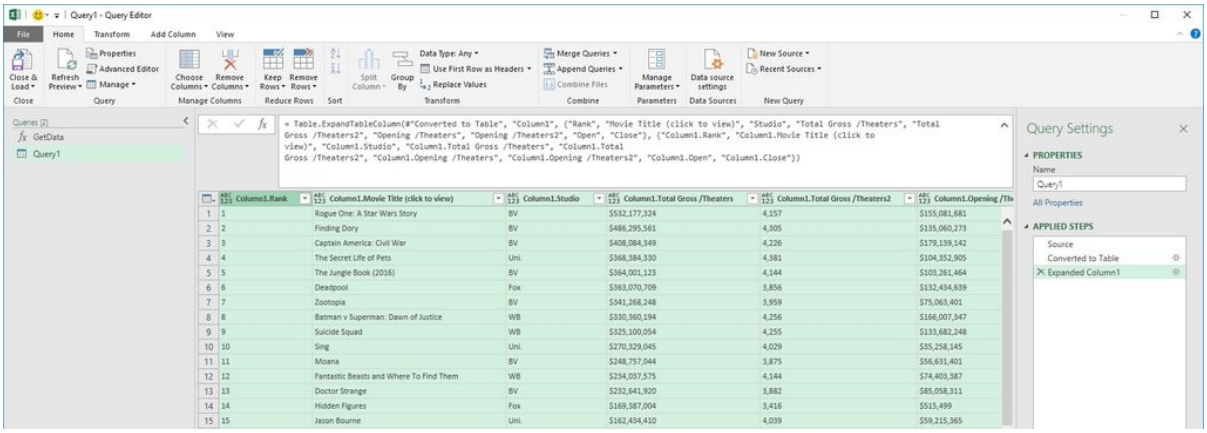


Figure 03-36: Expanded Table

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	Column
94	93	The Finest Hours	BV	$27,569,558	3,143
95	94	Florence Foster Jenkins	Par.	$27,383,770	1,528
96	95	Hell or High Water	LGF	$27,007,844	1,505
97	96	The Forest	Focus	$26,594,261	2,509
98	97	Ben-Hur (2016)	Par.	$26,410,477	3,084
99	98	The Witch	A24	$25,138,705	2,204
100	99	Bridget Jones's Baby	Uni.	$24,252,420	2,930
101	100	Kevin Hart: What Now?	Uni.	$23,591,043	2,567
102	101	Whiskey Tango Foxtrot	Par.	$23,083,334	2,413
103	102	Snowden	ORF	$21,587,519	2,443
104	103	Mechanic: Resurrection	LG/S	$21,218,403	2,258
105	104	Free State of Jones	STX	$20,810,036	2,815
106	105	Blair Witch	LGF	$20,777,061	3,121
107	106	God's Not Dead 2	PFR	$20,774,575	2,419
108	107	Keanu	WB (NL)	$20,591,853	2,681
109	108	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:

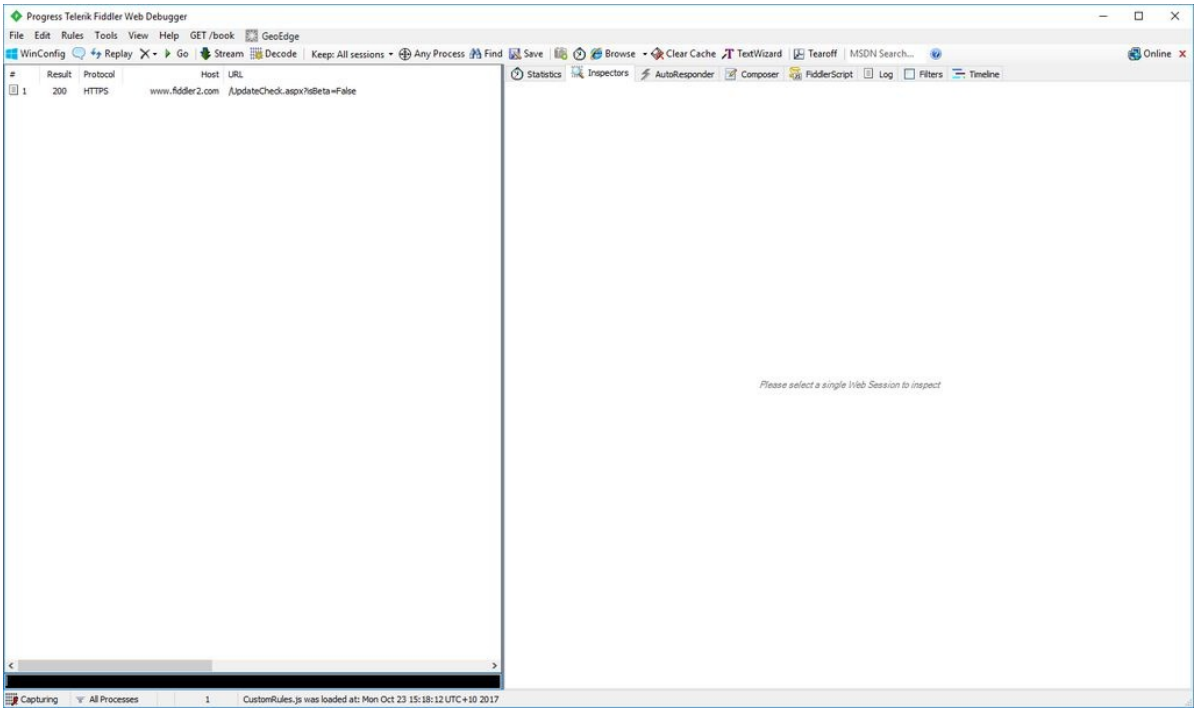


Figure 03-38: Fiddler

As the prompt 'Please select a single Web Session to impact' 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:

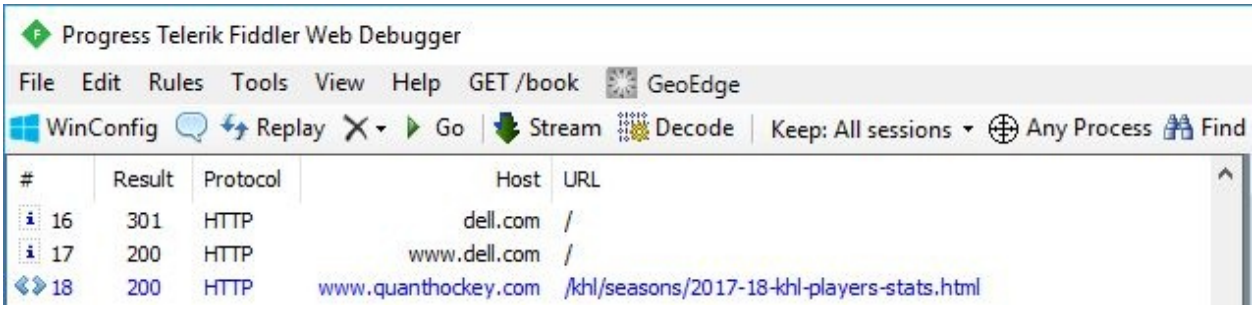


Figure 03-39: Fiddler Analysis 1

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

alternate URL:

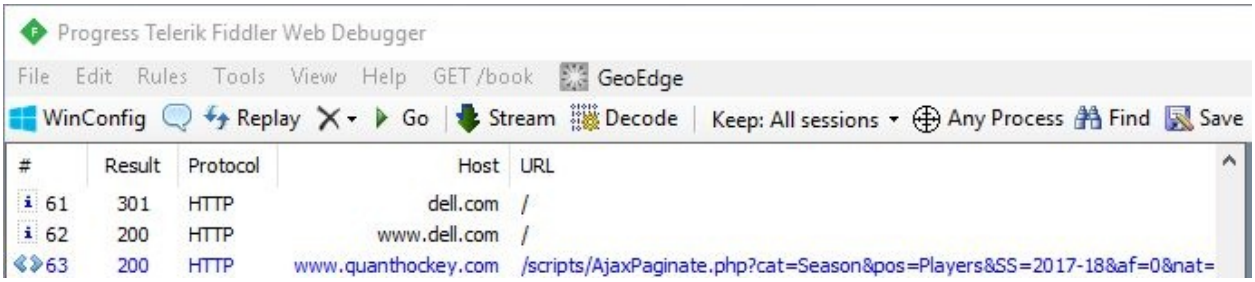


Figure 03-40: Fiddler Analysis 2

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'.

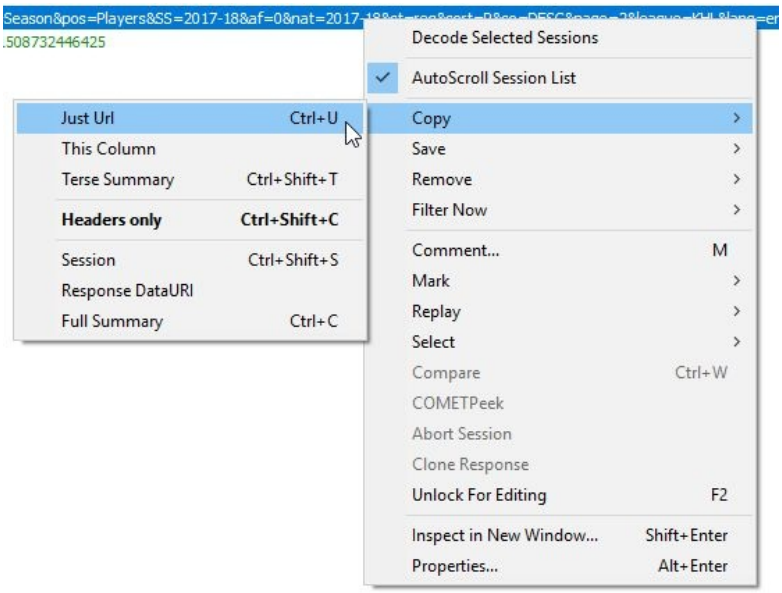


Figure 03-41: Copy Just URL

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:

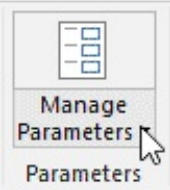


Figure 03-43: Manage Parameters

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

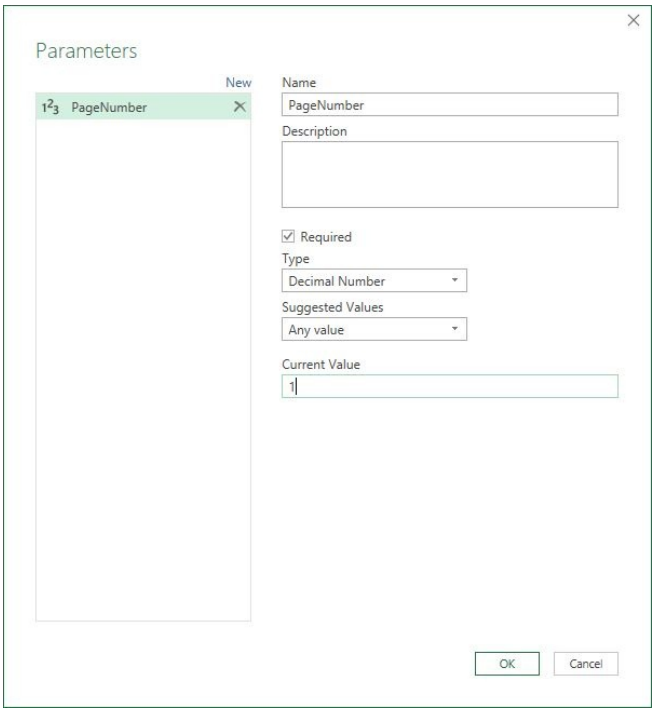


Figure 03-44: Parameters Dialog

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'):

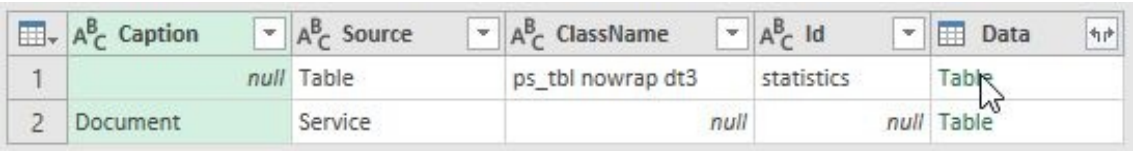
="&Text.From(PageNumber)&"

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:



	Caption	Source	ClassName	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 Nestrasil	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**:

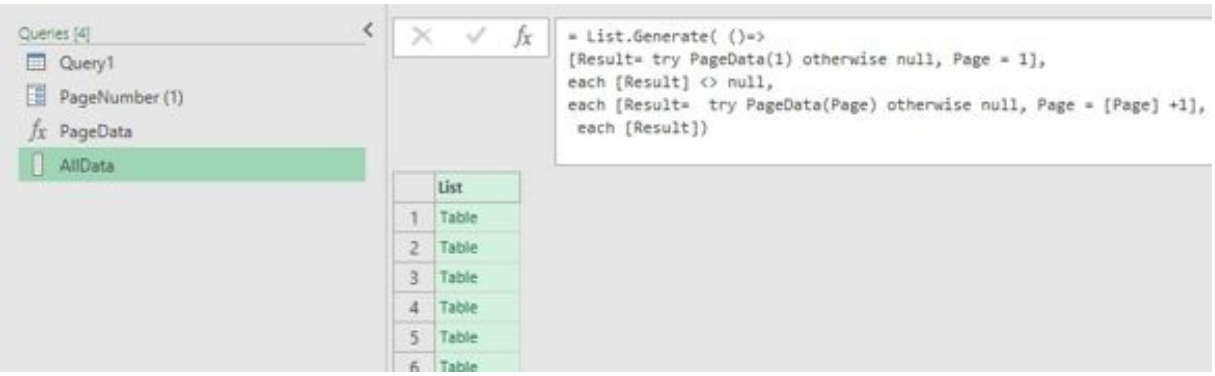


Figure 03-49: AllData

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

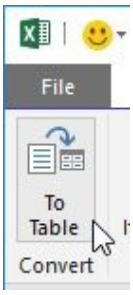


Figure 03-50: Convert to Table

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

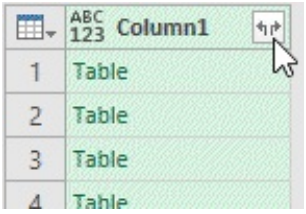


Figure 03-51: Expand to Table (Again)

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

Figure 03-52: Complete Table

We can now proceed to ‘Close & Load’.

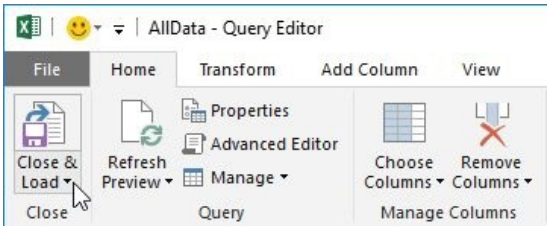
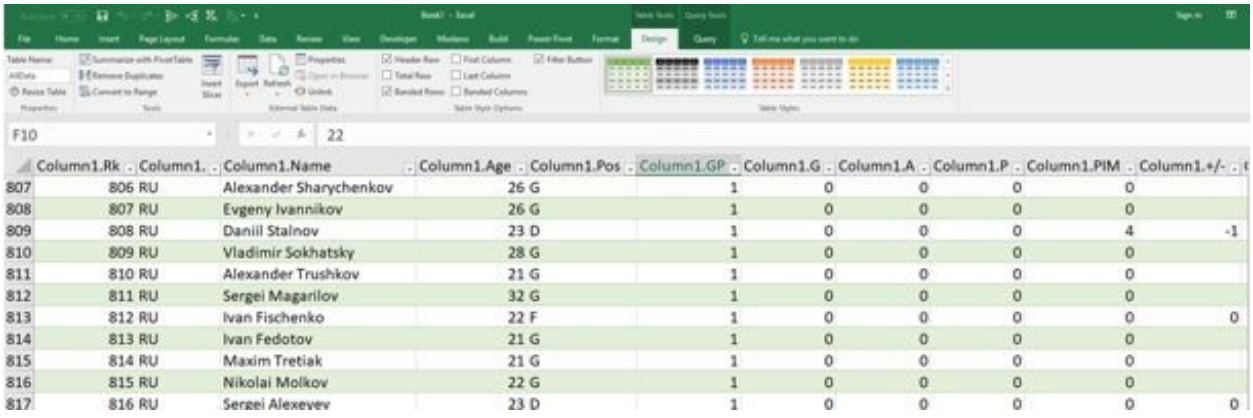


Figure 03-53: Close & Load

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



F10 22

	Column1.Rk	Column1.	Column1.Name	Column1.Age	Column1.Pos	Column1.GP	Column1.G	Column1.A	Column1.P	Column1.PIM	Column1.+/-
807	806	RU	Alexander Sharychenkov	26	G	1	0	0	0	0	
808	807	RU	Evgeny Ivannikov	26	G	1	0	0	0	0	
809	808	RU	Daniil Stalnov	23	D	1	0	0	0	4	-1
810	809	RU	Vladimir Sokhatsky	28	G	1	0	0	0	0	
811	810	RU	Alexander Trushkov	21	G	1	0	0	0	0	
812	811	RU	Sergei Magarilov	32	G	1	0	0	0	0	
813	812	RU	Ivan Fischenko	22	F	1	0	0	0	0	0
814	813	RU	Ivan Fedotov	21	G	1	0	0	0	0	
815	814	RU	Maxim Tretiak	21	G	1	0	0	0	0	
816	815	RU	Nikolai Molkov	22	G	1	0	0	0	0	
817	816	RU	Sergei Alexeyev	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:

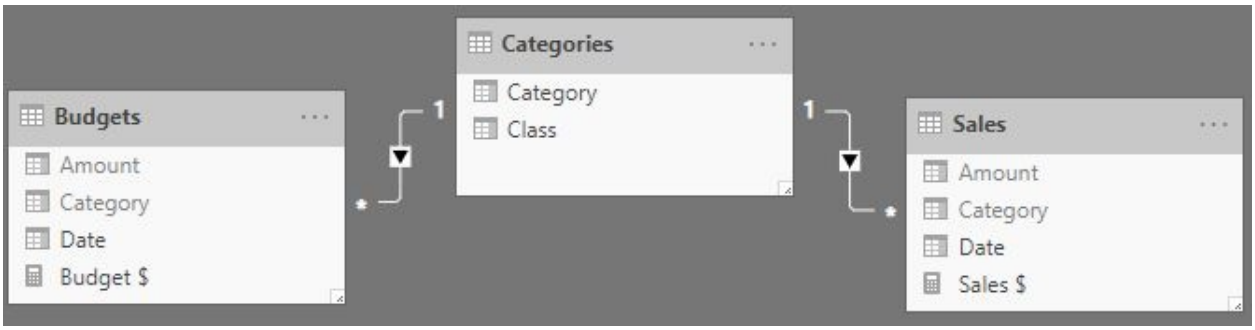


Figure 04-01: Sample data model

A sample of the data is shown below:

Categories

	Category	Class
1	Beer	Beverages
2	Wine	Beverages
3	Burgers	Food
4	Other	Other

Sales

	Date	Category	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

Budgets

	Date	Category	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.

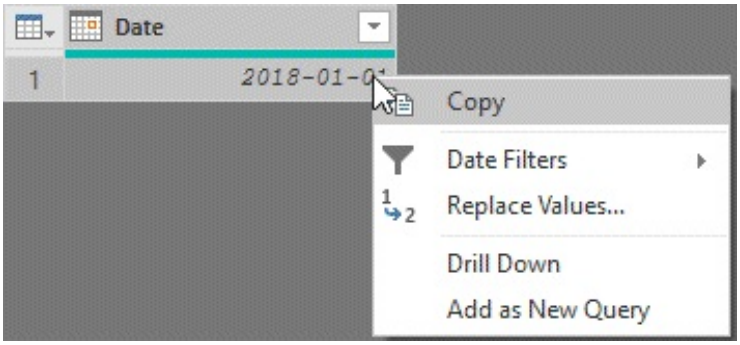


Figure 04-03: Right click the actual date, not the column header!

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:

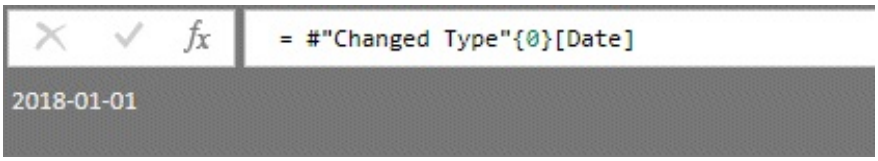


Figure 04-04: The contents of the StartDate query

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(#"Start Date")..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:

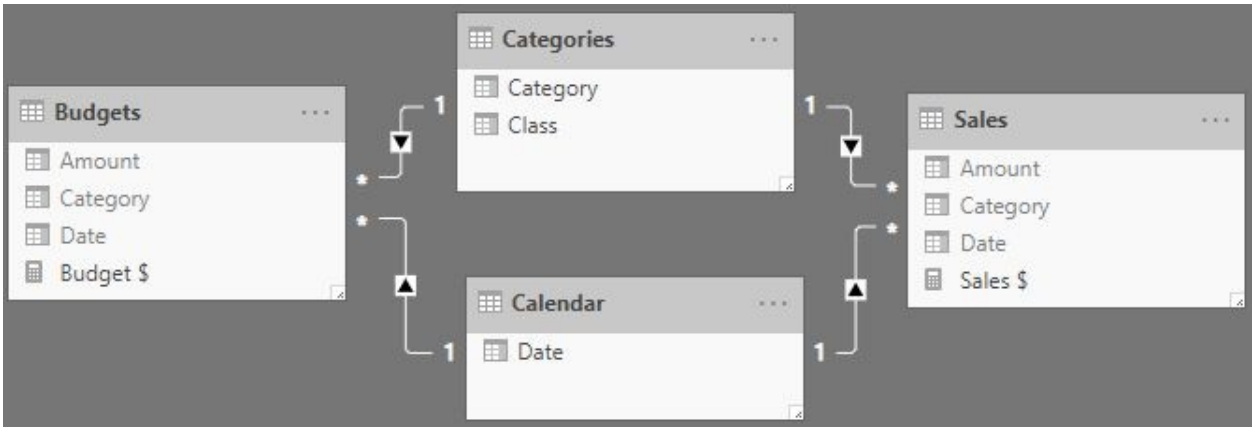


Figure 04-06: The contents of the StartDate query

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:

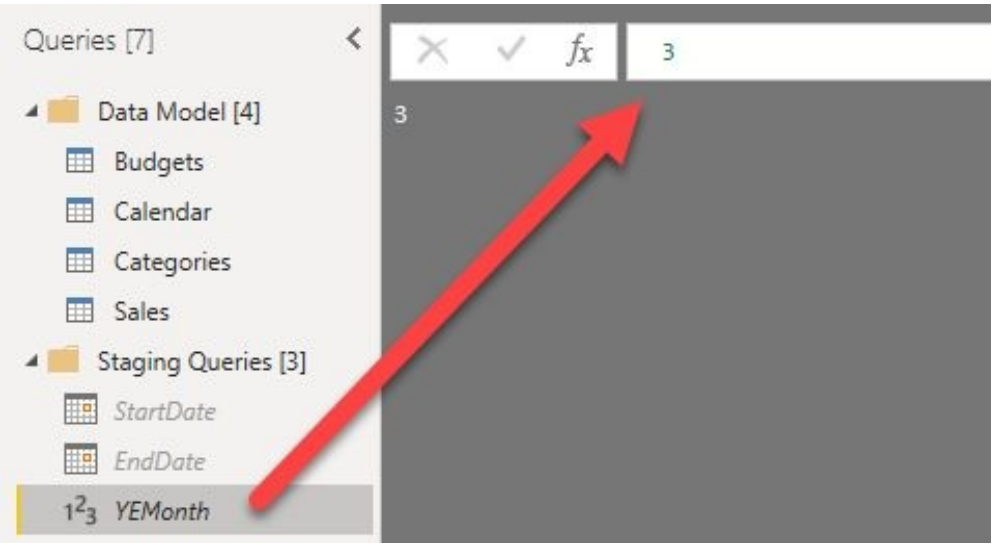


Figure 04-07: The YEMonth query set up for a March 31 year end

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: =Date.AddMonths([Date], YEMonth - 12)	• Add a custom column using this formula: =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

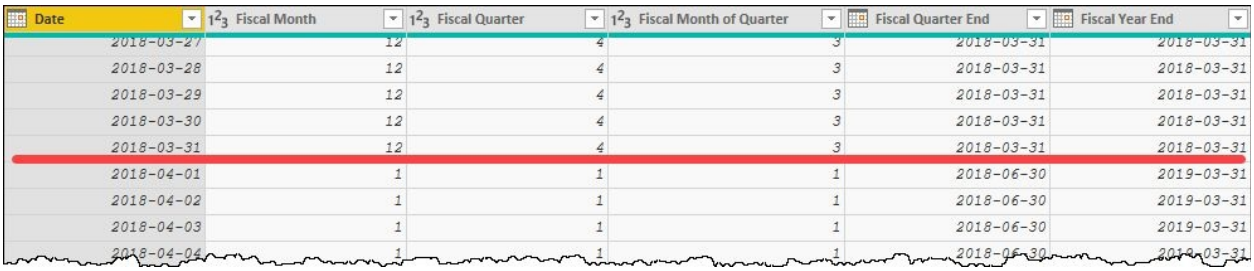
Fiscal Year End

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	Fiscal Month	Fiscal Quarter	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:

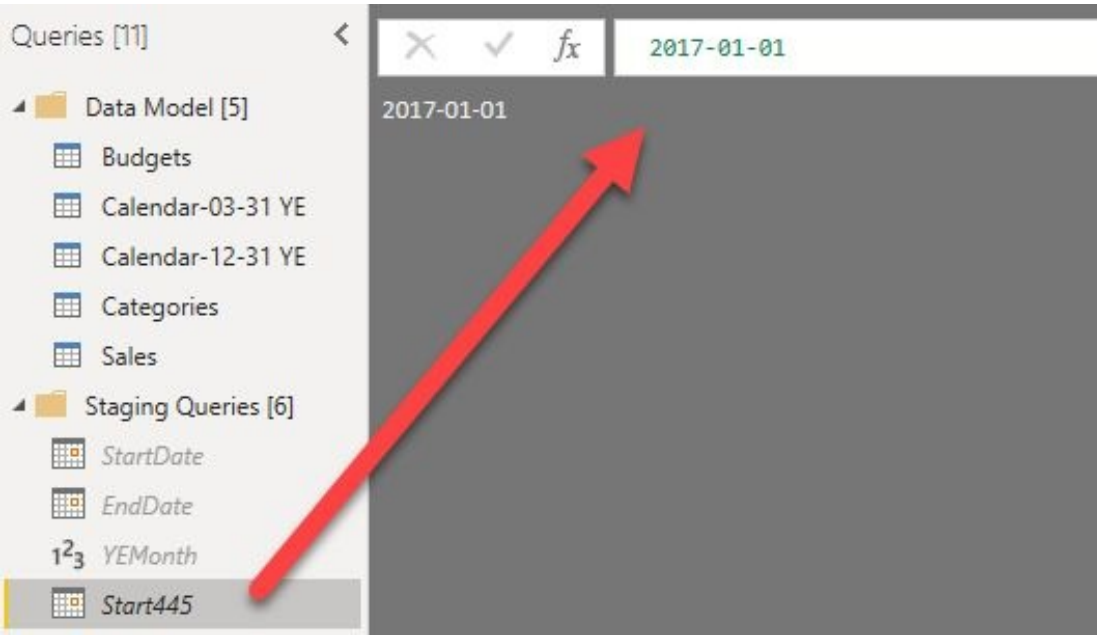


Figure 04-13: Recording the start of any fiscal year

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

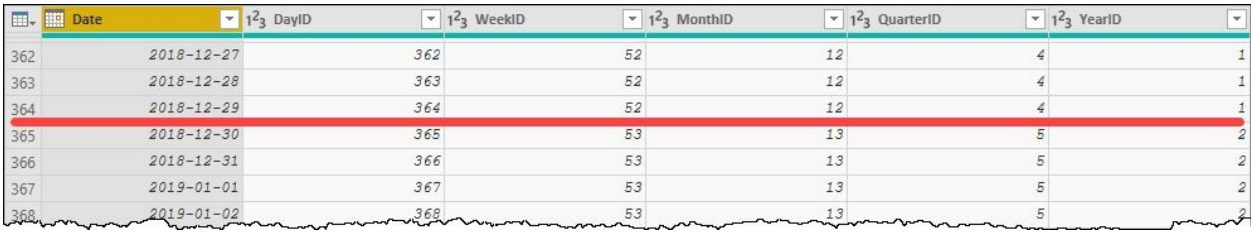
Next, you'll need to add the remaining Period ID columns in order to drive other date displays. These are all created via the Add Column -> Custom Column dialog using the following formulas:

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	DayID	WeekID	MonthID	QuarterID	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:

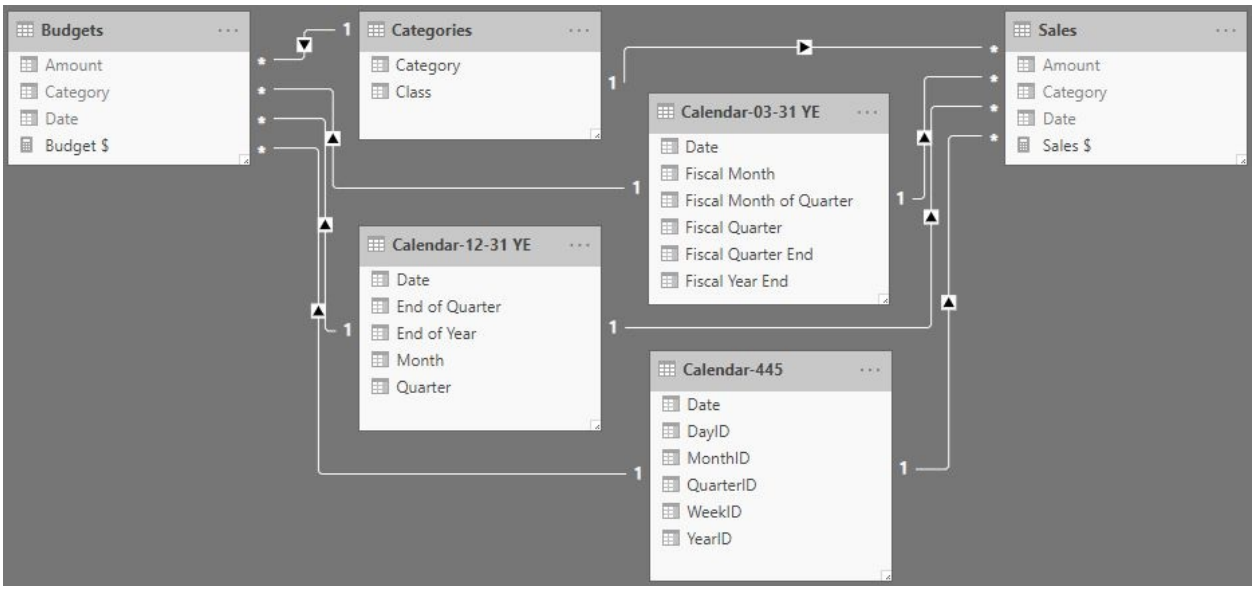


Figure 04-27: Multiple calendars linked into a single model

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

End of Year	Sales $	Budget $
December 31, 2018	**1,070,660**	**1,806,525**
January 31, 2018	128,728	158,918
February 28, 2018	143,930	136,694
March 31, 2018	182,431	84,894
April 30, 2018	131,234	147,794
May 31, 2018	143,810	204,561
June 30, 2018	181,149	72,674
July 31, 2018	151,653	224,975
August 31, 2018	7,725	95,685
September 30, 2018		245,954
October 31, 2018		96,039
November 30, 2018		207,321
December 31, 2018		131,016
Total	**1,070,660**	**1,806,525**

12-Month Mar 31 Year End

Fiscal Year End	Sales $	Budget $
March 31, 2018	**455,089**	**380,506**
January 31, 2018	128,728	158,918
February 28, 2018	143,930	136,694
March 31, 2018	182,431	84,894
March 31, 2019	**615,571**	**1,426,019**
April 30, 2018	131,234	147,794
May 31, 2018	143,810	204,561
June 30, 2018	181,149	72,674
July 31, 2018	151,653	224,975
August 31, 2018	7,725	95,685
September 30, 2018		245,954
October 31, 2018		96,039
November 30, 2018		207,321
December 31, 2018		131,016
Total	**1,070,660**	**1,806,525**

4-4-5 Calendar

EndOfYear	Sales $	Budget $
December 29, 2018	**1,070,660**	**1,675,509**
January 27, 2018	118,619	
February 24, 2018	132,276	158,918
March 10, 2018	204,194	221,588
April 28, 2018	117,020	
May 26, 2018	123,119	147,794
June 9, 2018	216,054	277,235
July 28, 2018	132,946	
August 25, 2018	26,432	224,975
September 8, 2018		95,685
October 27, 2018		245,954
November 24, 2018		96,039
December 8, 2018		207,321
December 28, 2019		**131,016**
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.cudaso.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 if 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.

ɔte: 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.

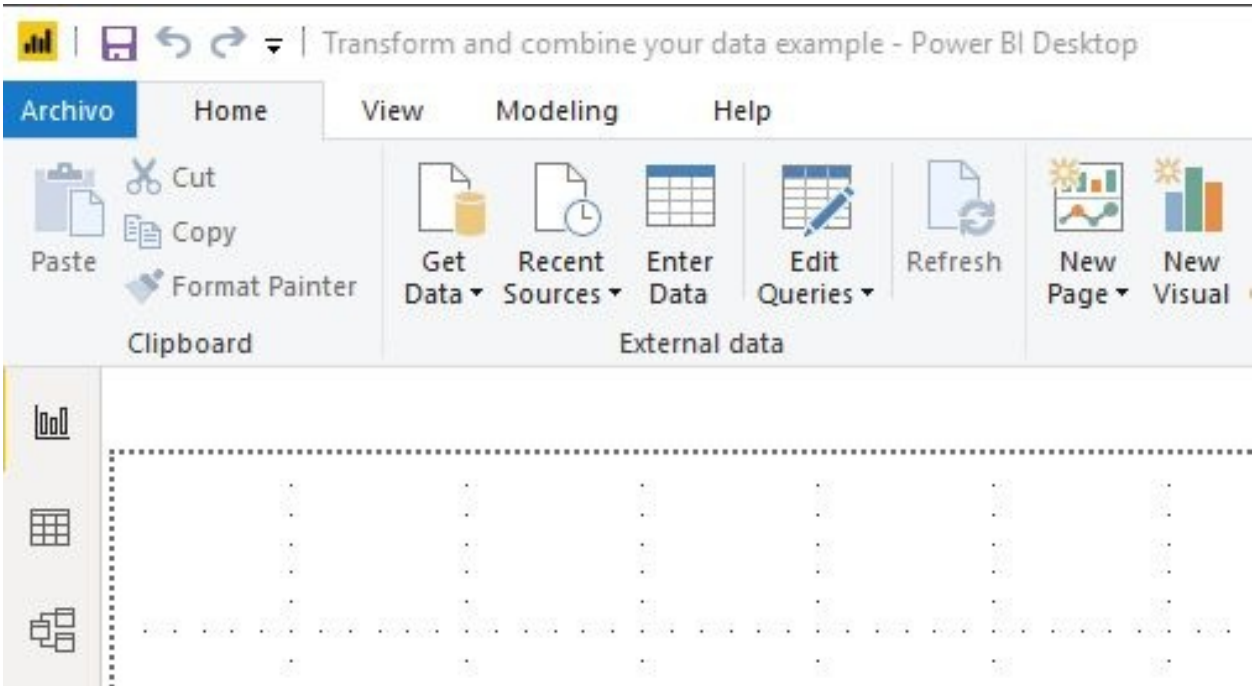


Figure 05-01: Launching the Power Query editor

Why do we get taken to a blank pane?

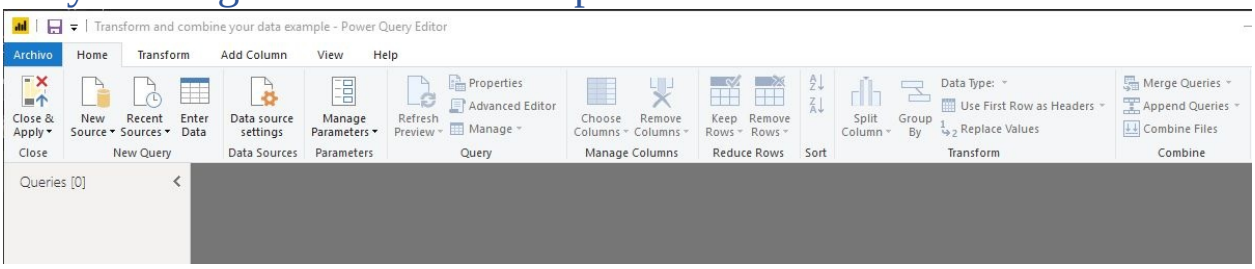


Figure 05-02: Black pane in Power Query editor

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.

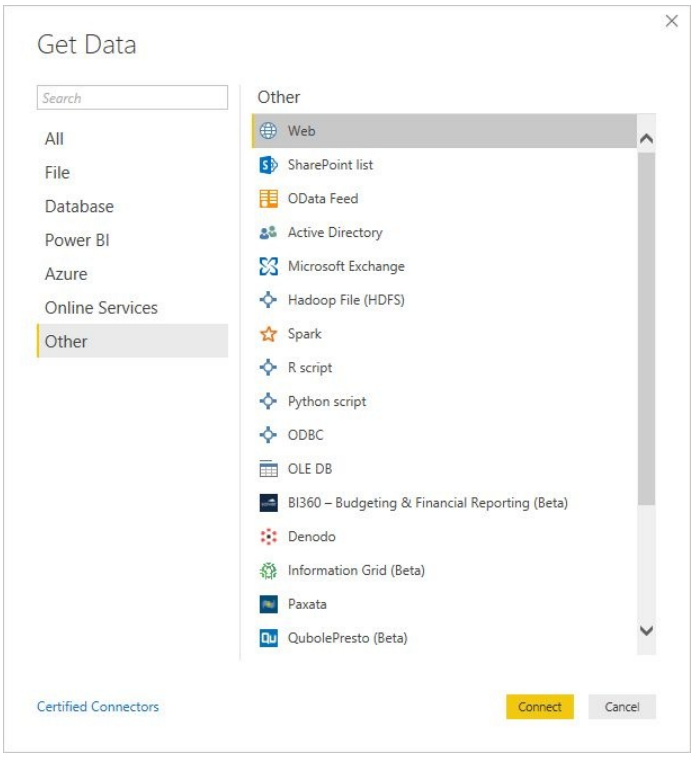


Figure 05-03: Get Data window, to select the data source

Type the URL source and click the OK button.

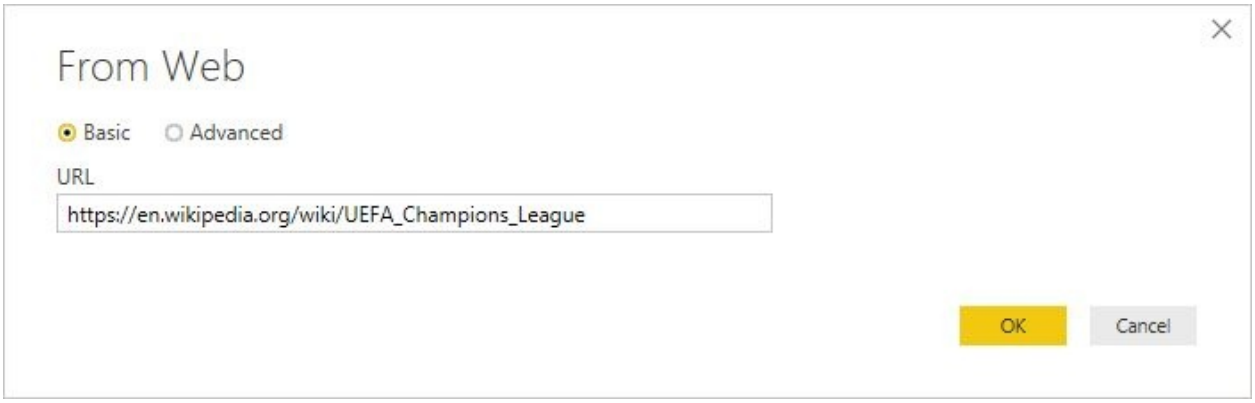


Figure 05-04: Typing the URL source

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.

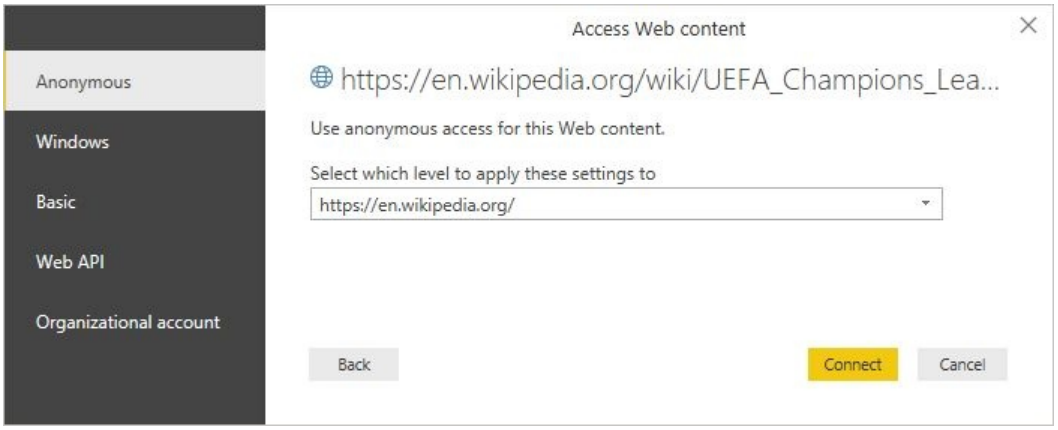


Figure 05-05: Asking us the type access

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.

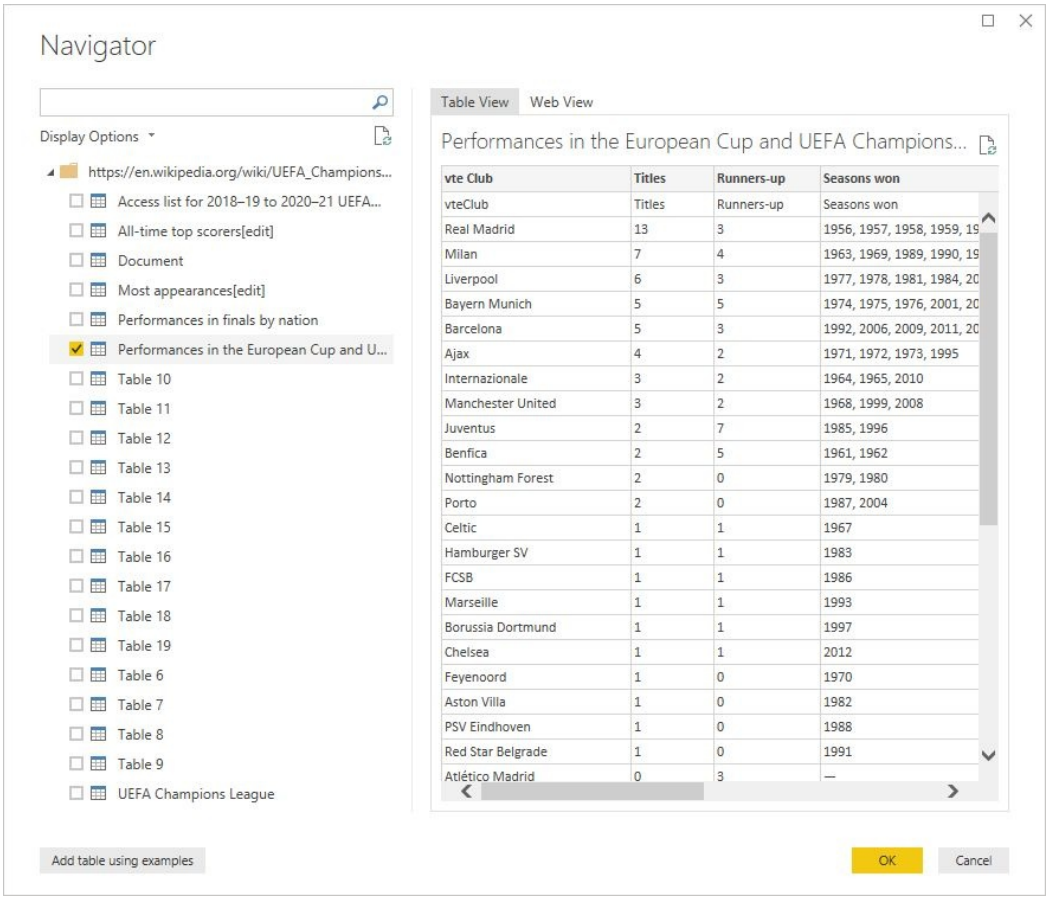


Figure 05-07: Navigator window allow the first look to the data uploaded

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

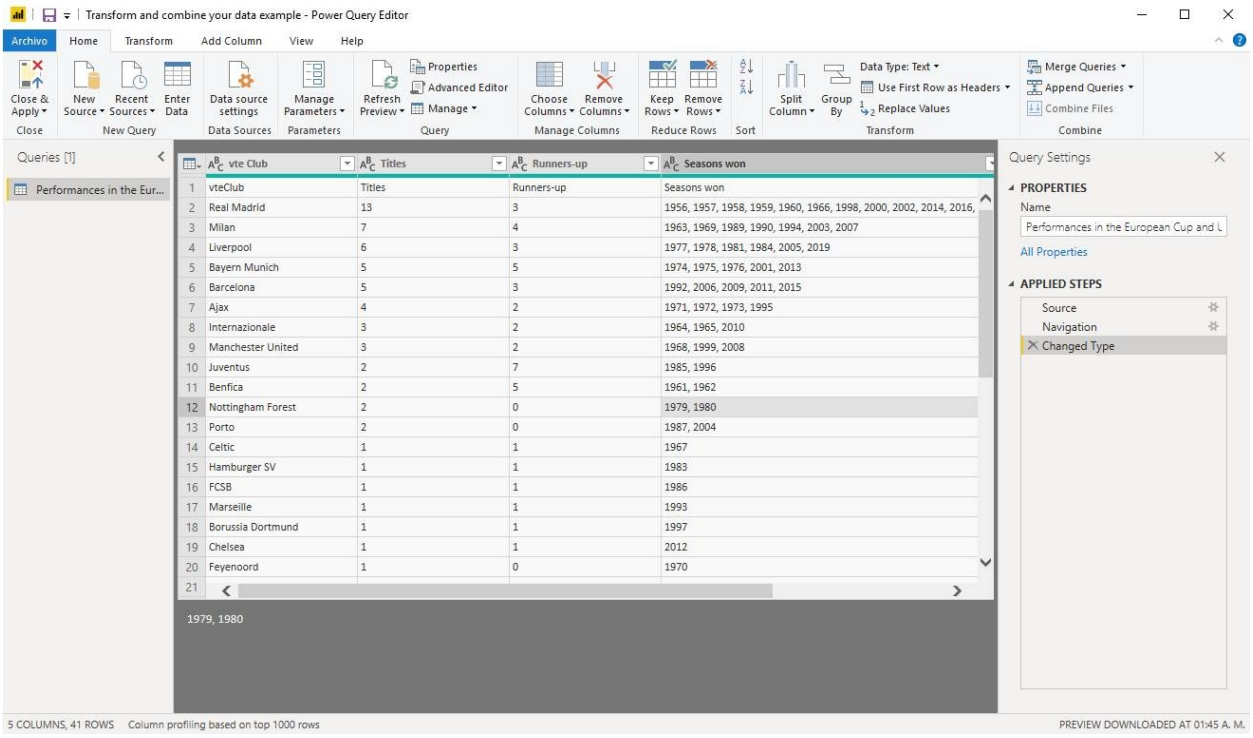


Figure 05-08: Power Query Editor window

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.

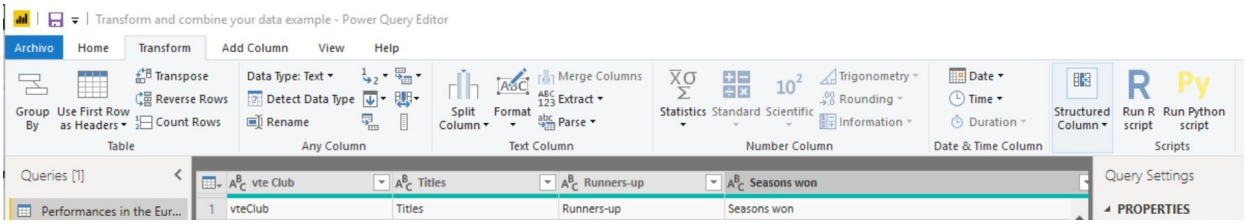


Figure 05-09: Transform ribbon tab, general look

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.

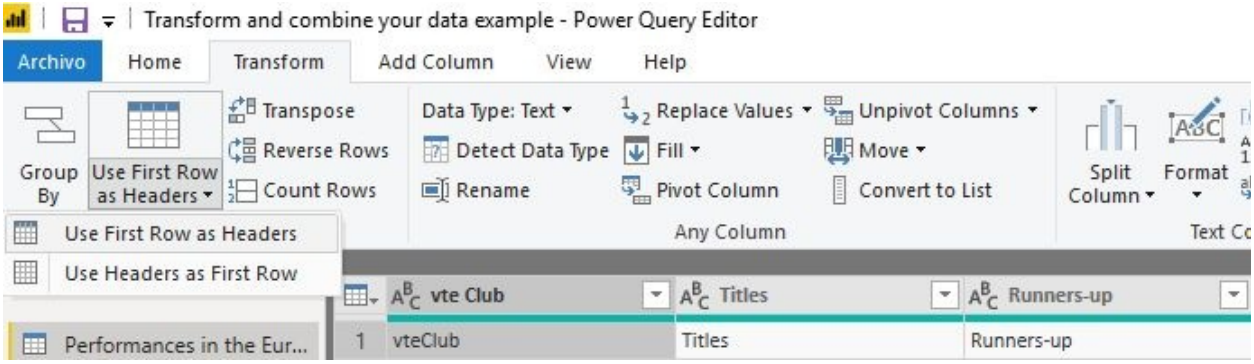


Figure 05-10: Launching the task: Use First Row as Headers

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

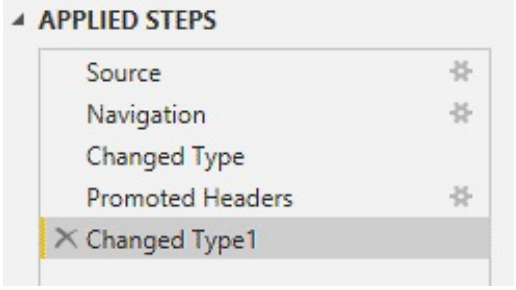


Figure 05-11: In the Query Settings pane, the APPLIED STEPS list the

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.

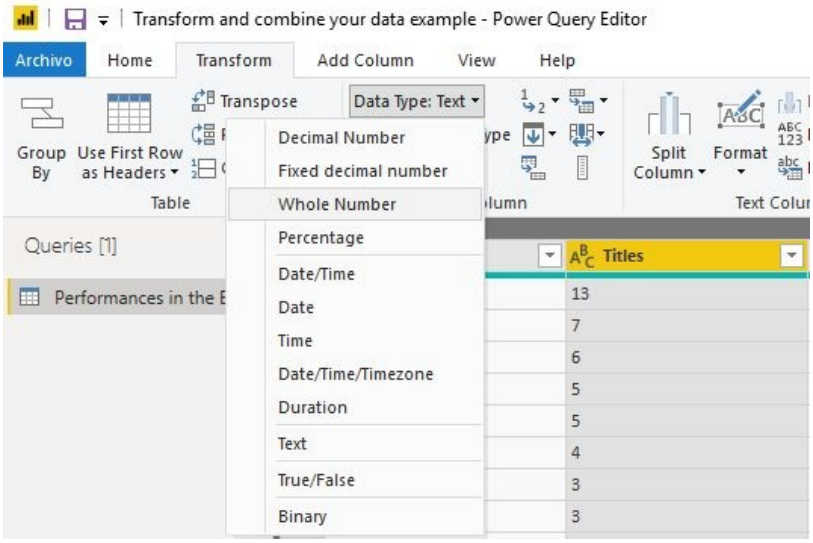


Figure 05-12: Changing data type from char to whole number

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.

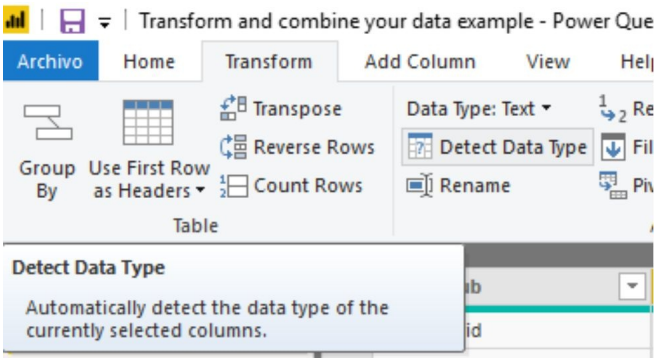


Figure 05-13: Using automatic detection button

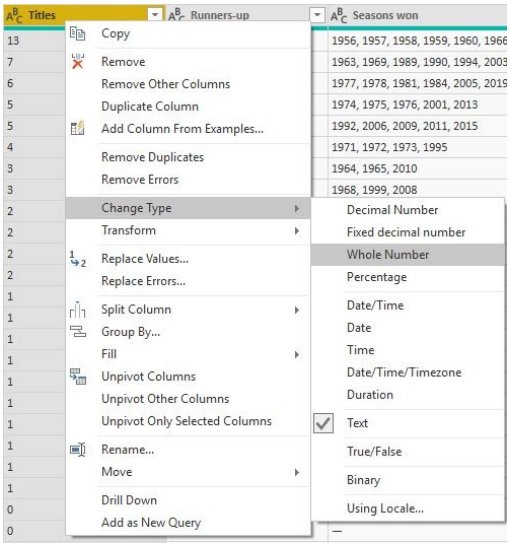


Figure 05-14: Using Change Type, on popup menu, from a column

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

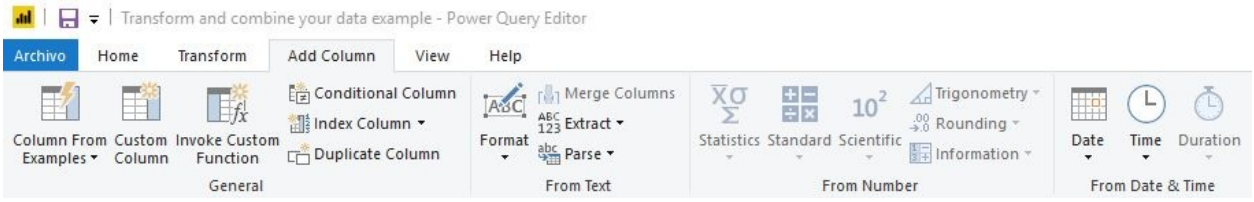


Figure 05-15: Transform Add Column tab, general look

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"]

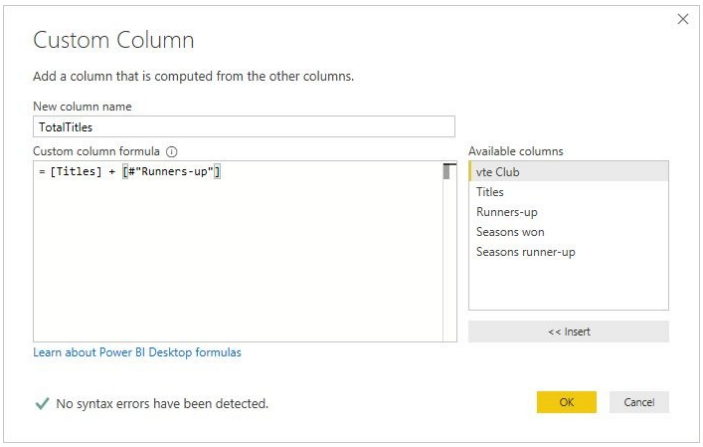


Figure 05-16: Building a Custom column with Power Query M language

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

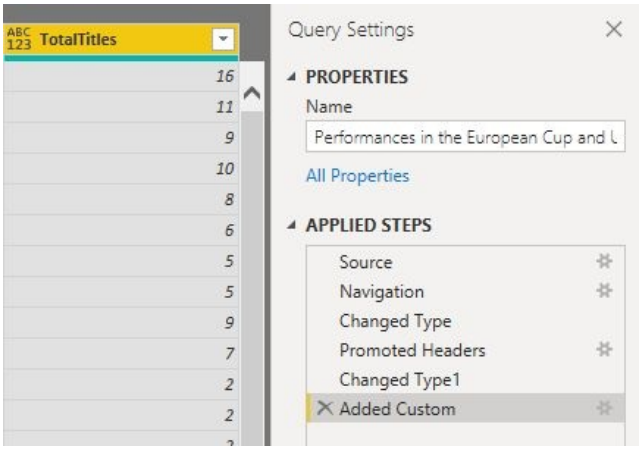


Figure 05-17: Added Custom appears as new query step at the APPLIED STEPS list

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.

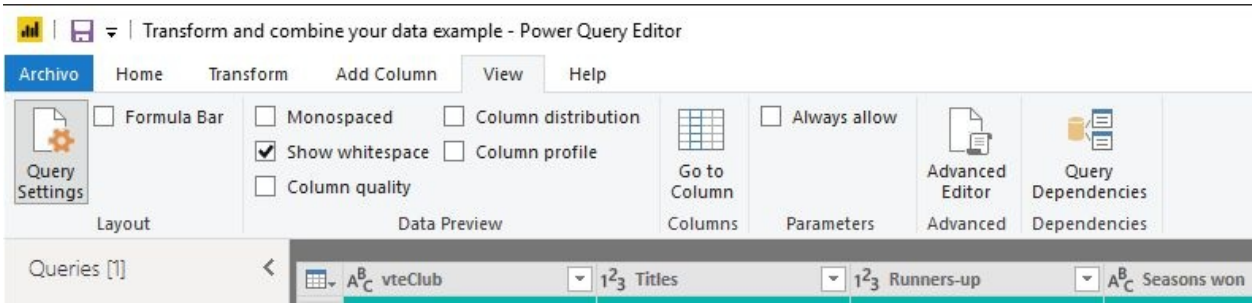


Figure 05-18: View tab, general look

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:

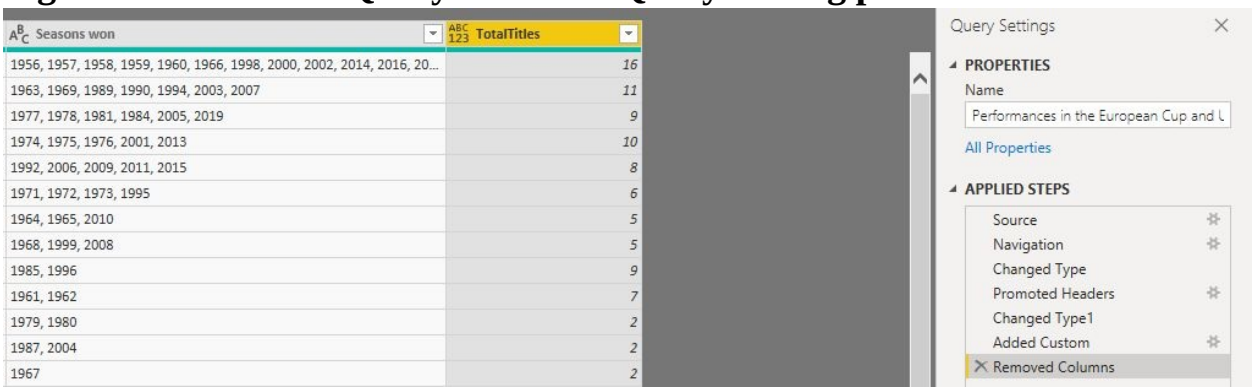
Figure 05-19: Advanced Editor with M language

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



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 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.

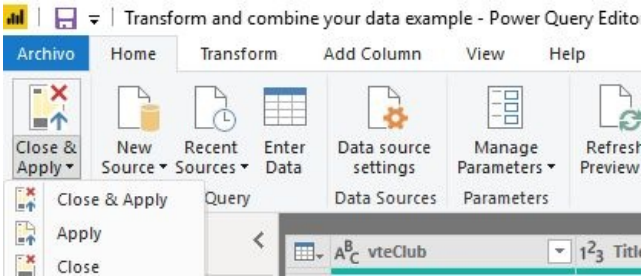


Figure 05-21: Close & Apply button

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

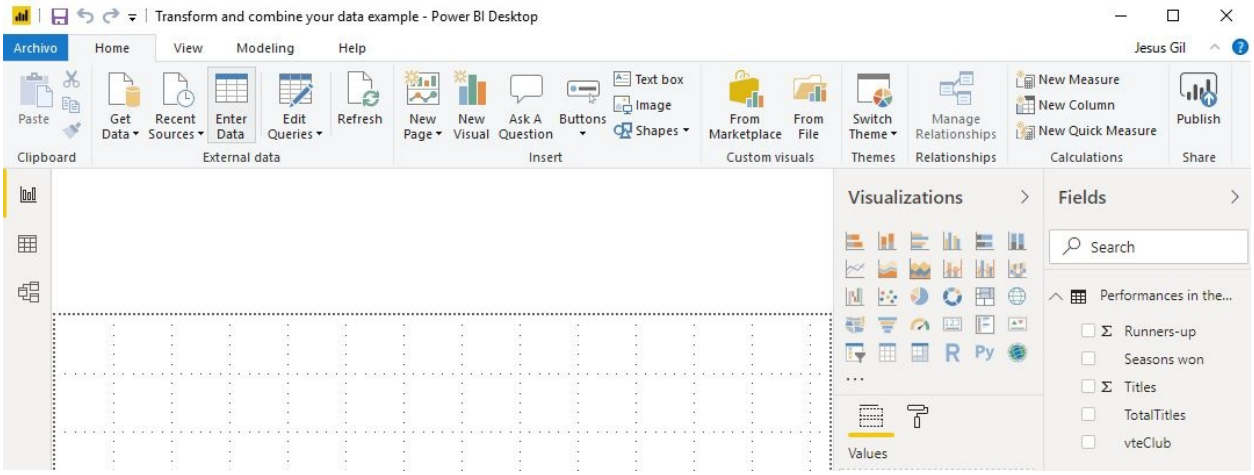


Figure 05-22: Dataset ready to use in the Power BI Desktop

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	Warehoue Y	12
3/02/2019	PN 13	Warehouse Z	43
4/02/2019	PN 50	Warehouse X	1
5/02/2019	PN 34	Warehoue Y	5
6/02/2019	PN 13	Warehouse Z	48
7/02/2019	PN 50	Warehouse X	65
8/02/2019	PN 34	Warehoue 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	Product	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.

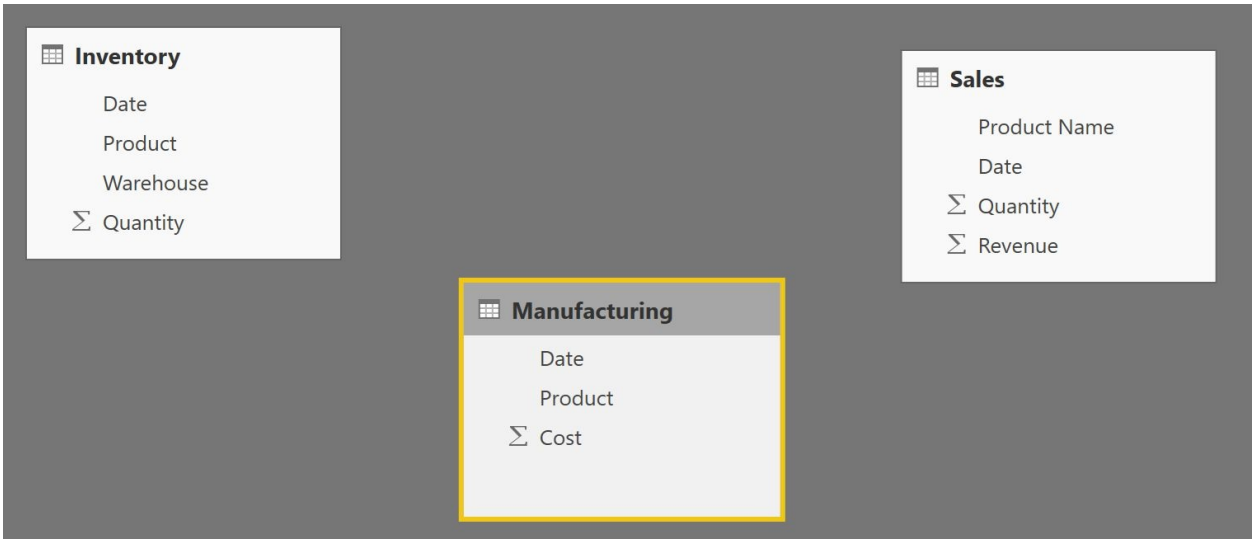


Figure 06-04: Design Challenge of the three tables

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!

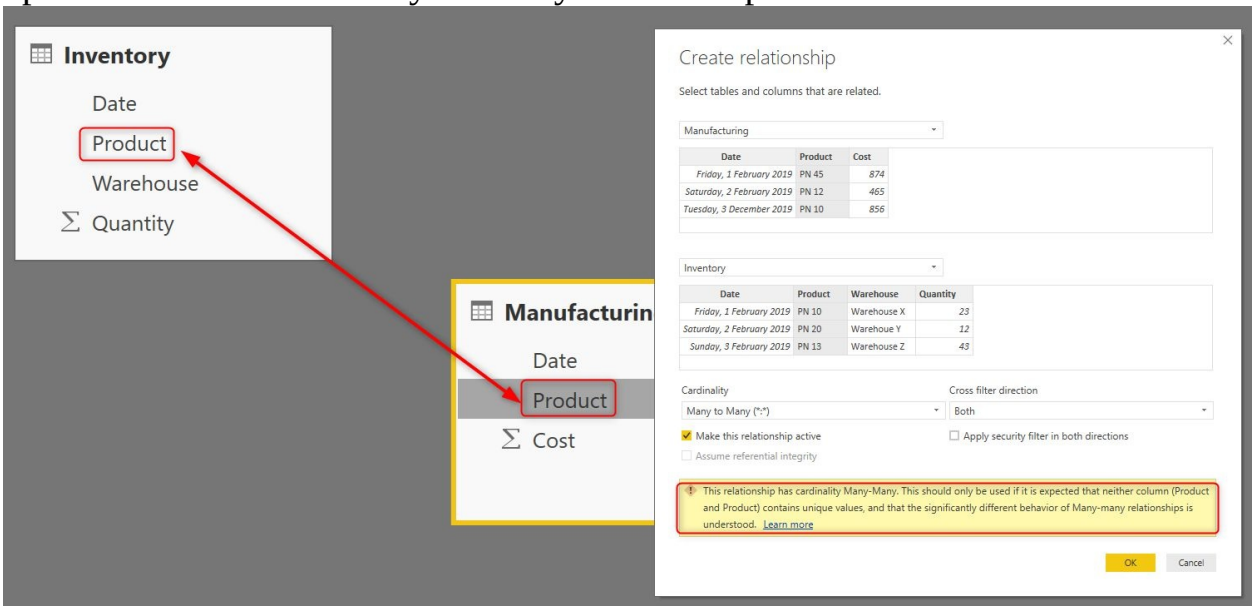


Figure 06-05: Many-to-many relationship Issue

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 that 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.

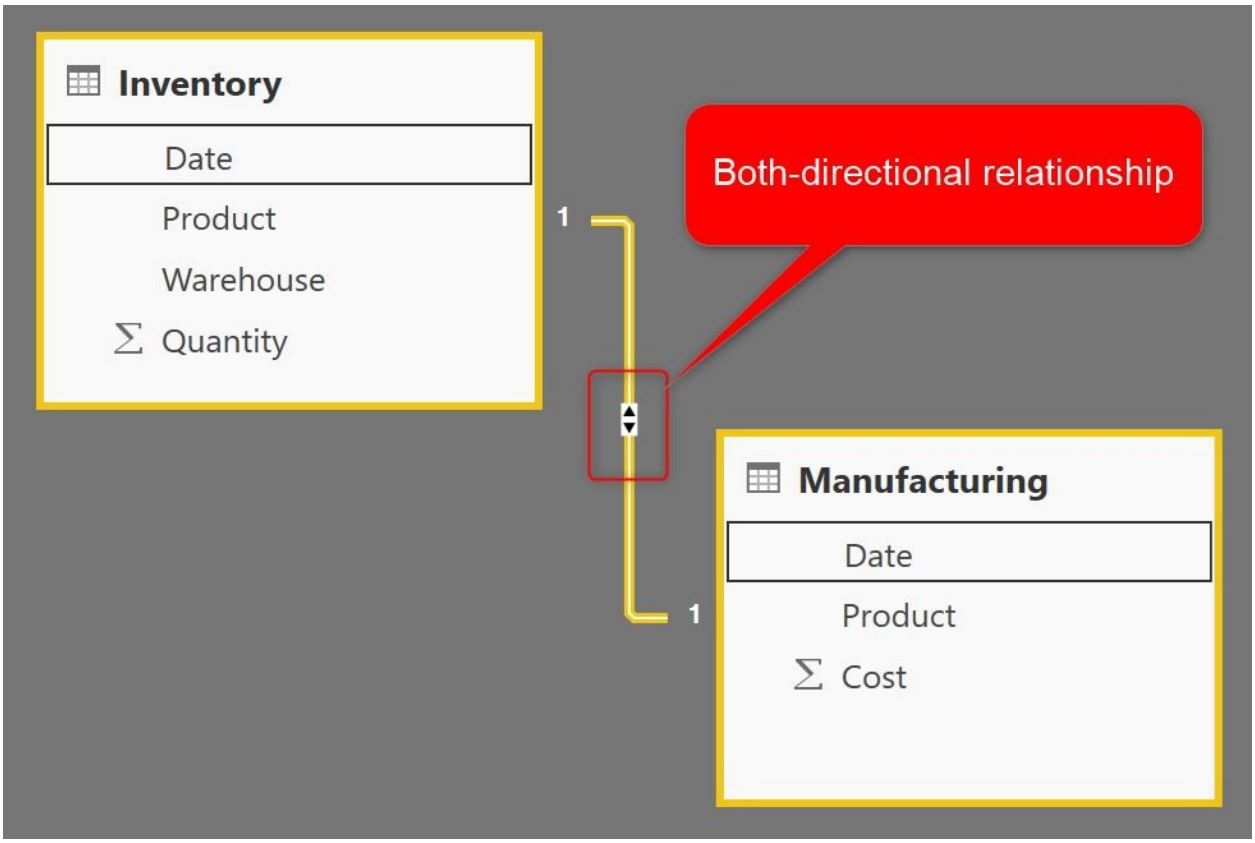


Figure 06-06: Both-directional Relationship Issue

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.

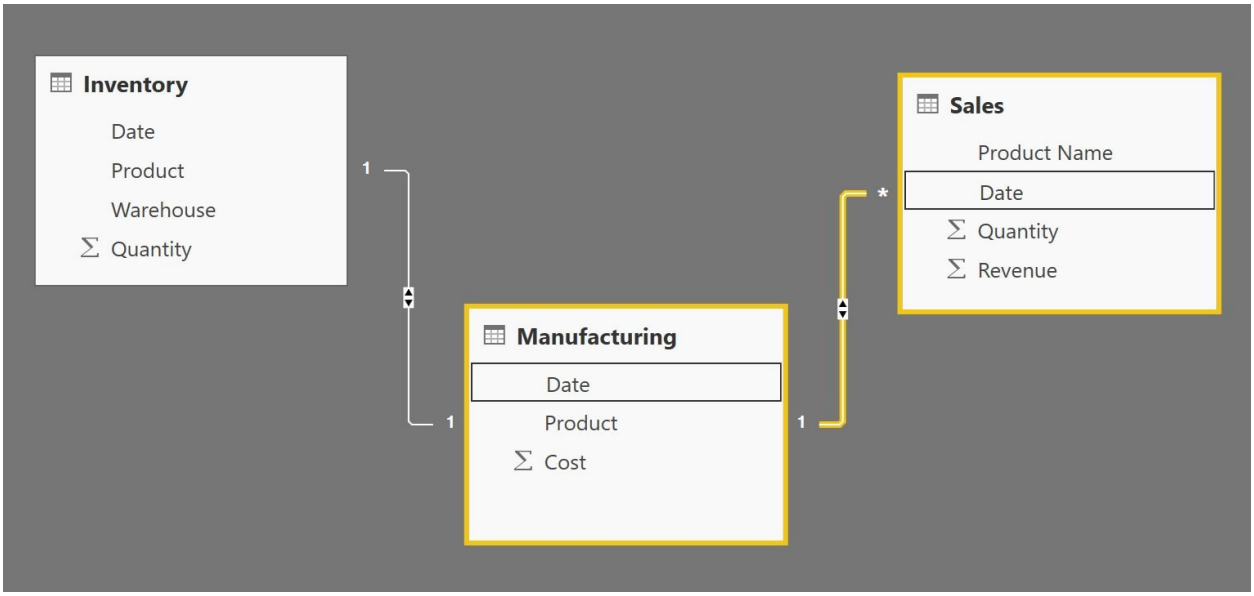
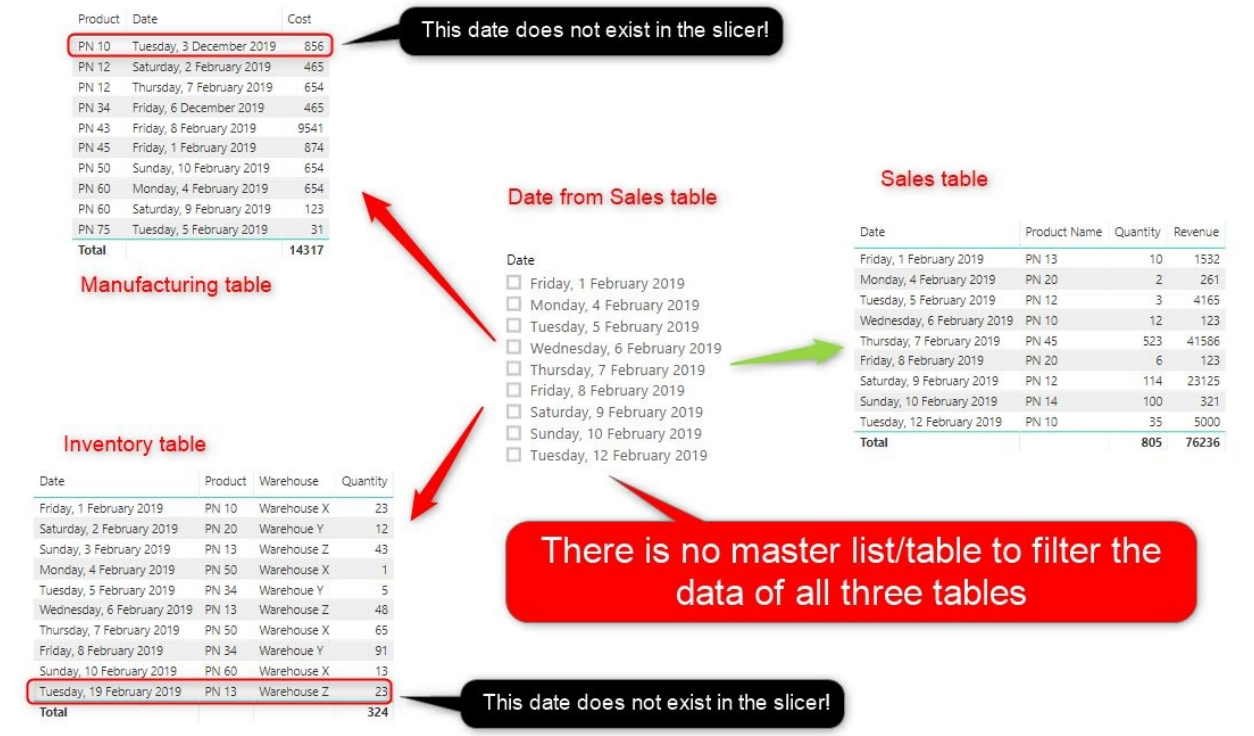


Figure 06-07: All both-directional relationship Issue

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;



Product	Date	Cost
PN 10	Tuesday, 3 December 2019	856
PN 12	Saturday, 2 February 2019	465
PN 12	Thursday, 7 February 2019	654
PN 34	Friday, 6 December 2019	465
PN 43	Friday, 8 February 2019	9541
PN 45	Friday, 1 February 2019	874
PN 50	Sunday, 10 February 2019	654
PN 60	Monday, 4 February 2019	654
PN 60	Saturday, 9 February 2019	123
PN 75	Tuesday, 5 February 2019	31
Total		14317

Date

- Friday, 1 February 2019
- Monday, 4 February 2019
- Tuesday, 5 February 2019
- Wednesday, 6 February 2019
- Thursday, 7 February 2019
- Friday, 8 February 2019
- Saturday, 9 February 2019
- Sunday, 10 February 2019
- Tuesday, 12 February 2019

Date	Product Name	Quantity	Revenue
Friday, 1 February 2019	PN 13	10	1532
Monday, 4 February 2019	PN 20	2	261
Tuesday, 5 February 2019	PN 12	3	4165
Wednesday, 6 February 2019	PN 10	12	123
Thursday, 7 February 2019	PN 45	523	41586
Friday, 8 February 2019	PN 20	6	123
Saturday, 9 February 2019	PN 12	114	23125
Sunday, 10 February 2019	PN 14	100	321
Tuesday, 12 February 2019	PN 10	35	5000
Total		805	76236

Date	Product	Warehouse	Quantity
Friday, 1 February 2019	PN 10	Warehouse X	23
Saturday, 2 February 2019	PN 20	Warehoue Y	12
Sunday, 3 February 2019	PN 13	Warehouse Z	43
Monday, 4 February 2019	PN 50	Warehouse X	1
Tuesday, 5 February 2019	PN 34	Warehoue Y	5
Wednesday, 6 February 2019	PN 13	Warehouse Z	48
Thursday, 7 February 2019	PN 50	Warehouse X	65
Friday, 8 February 2019	PN 34	Warehoue Y	91
Sunday, 10 February 2019	PN 60	Warehouse X	13
Tuesday, 19 February 2019	PN 13	Warehouse Z	23
Total			324

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:

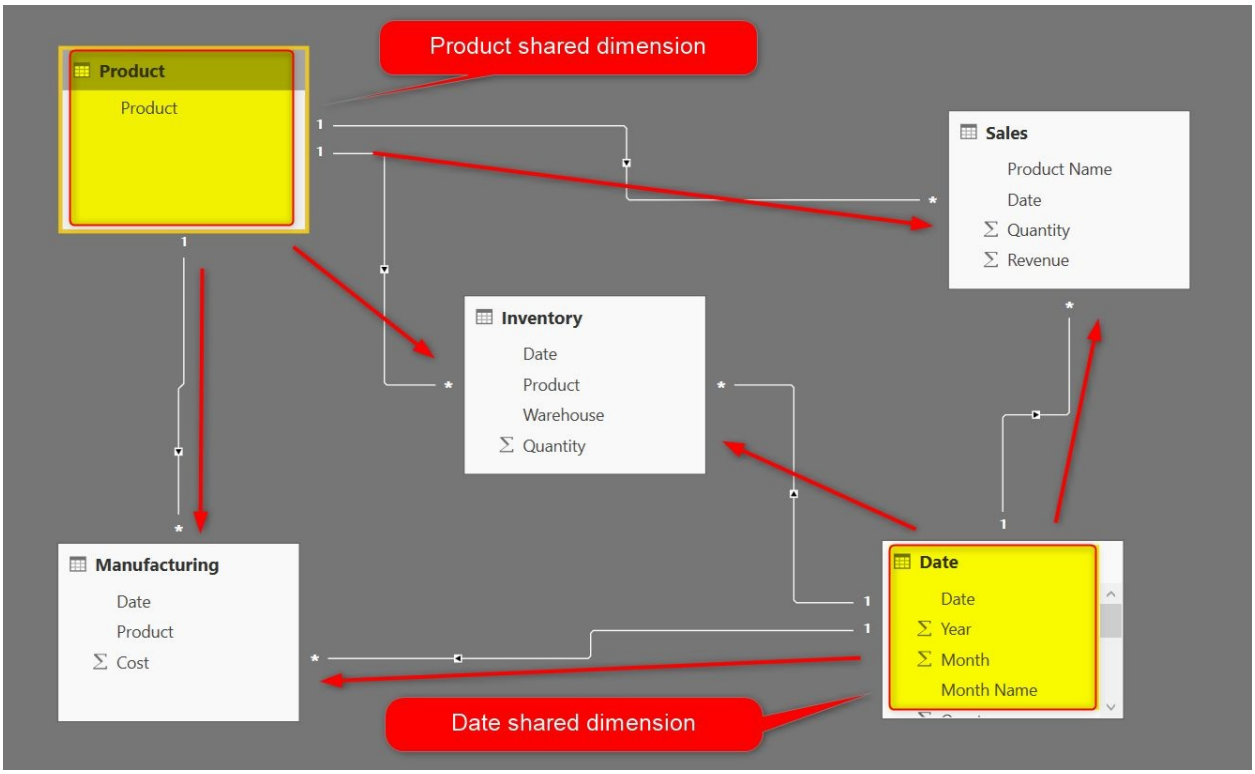


Figure 06-09: Solution Relationship Diagram

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;

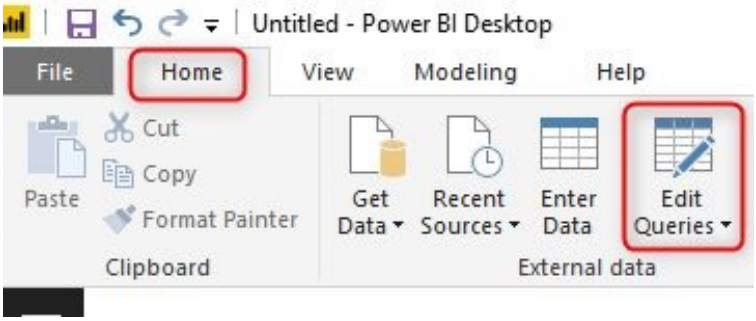


Figure 06-10: Edit Queries

Prepare sub-tables

In the Power Query Editor window, right click on Inventory table, and create a reference from the query:

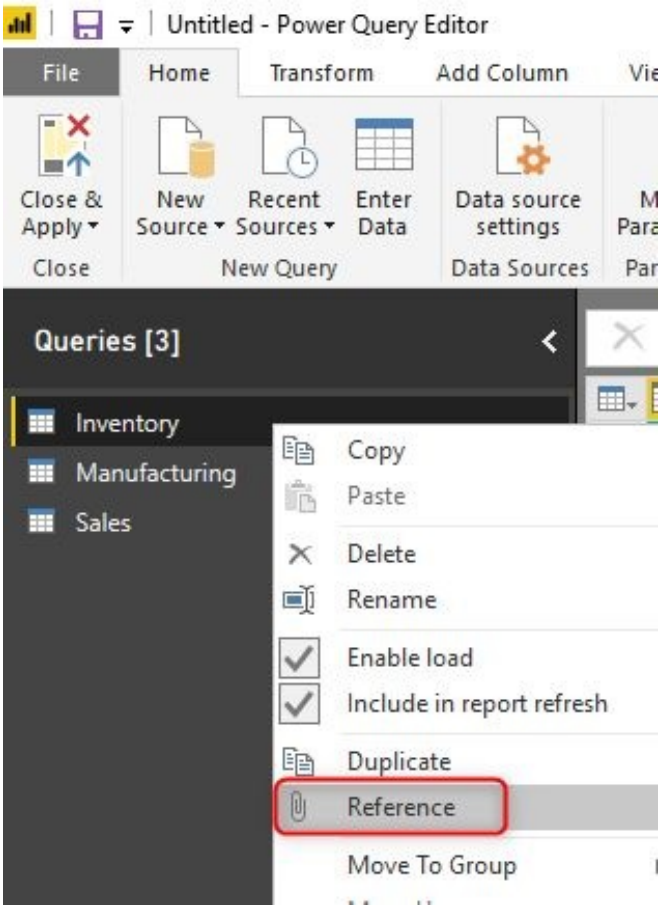


Figure 06-11: Reference from the existing query

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.

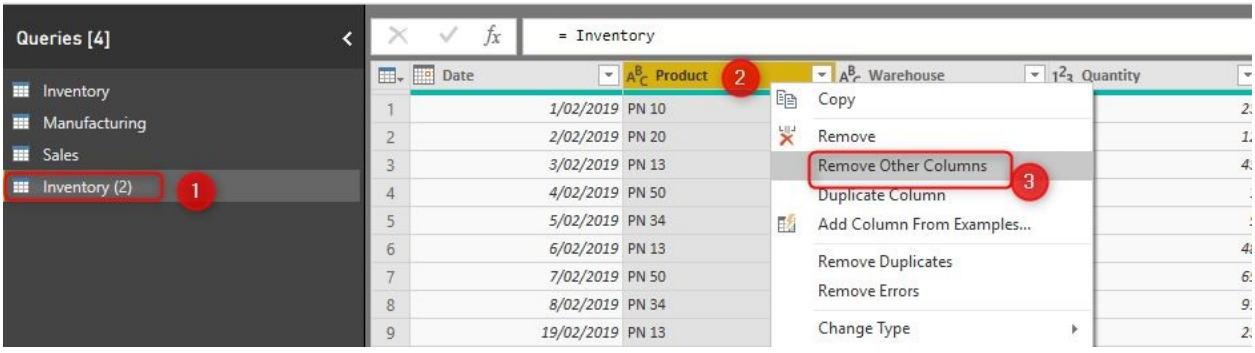


Figure 06-12: Removing Other Columns

The Inventory table should now look like this:

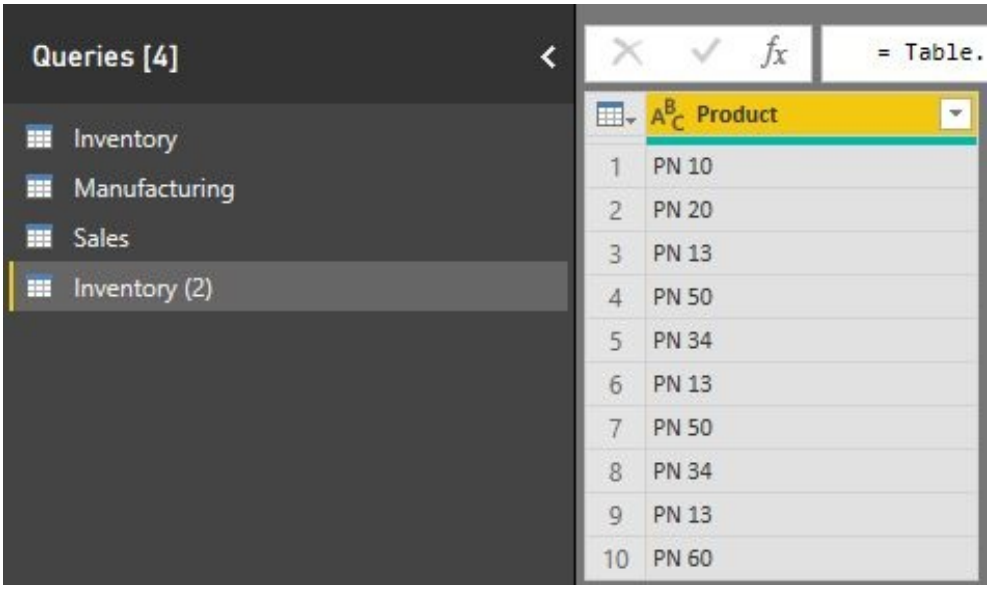


Figure 06-13: Inventory(2) Table with only Product Column

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.

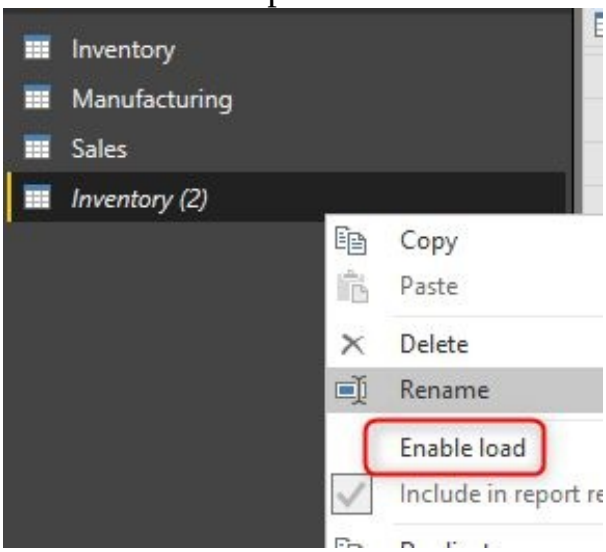


Figure 06-14: Disable Load for the Inventory(2) table

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:

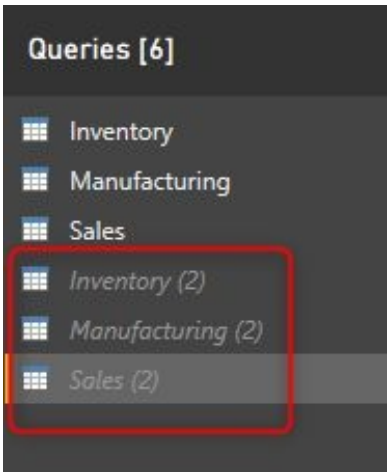


Figure 06-15: Three new tables all disabled load

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.

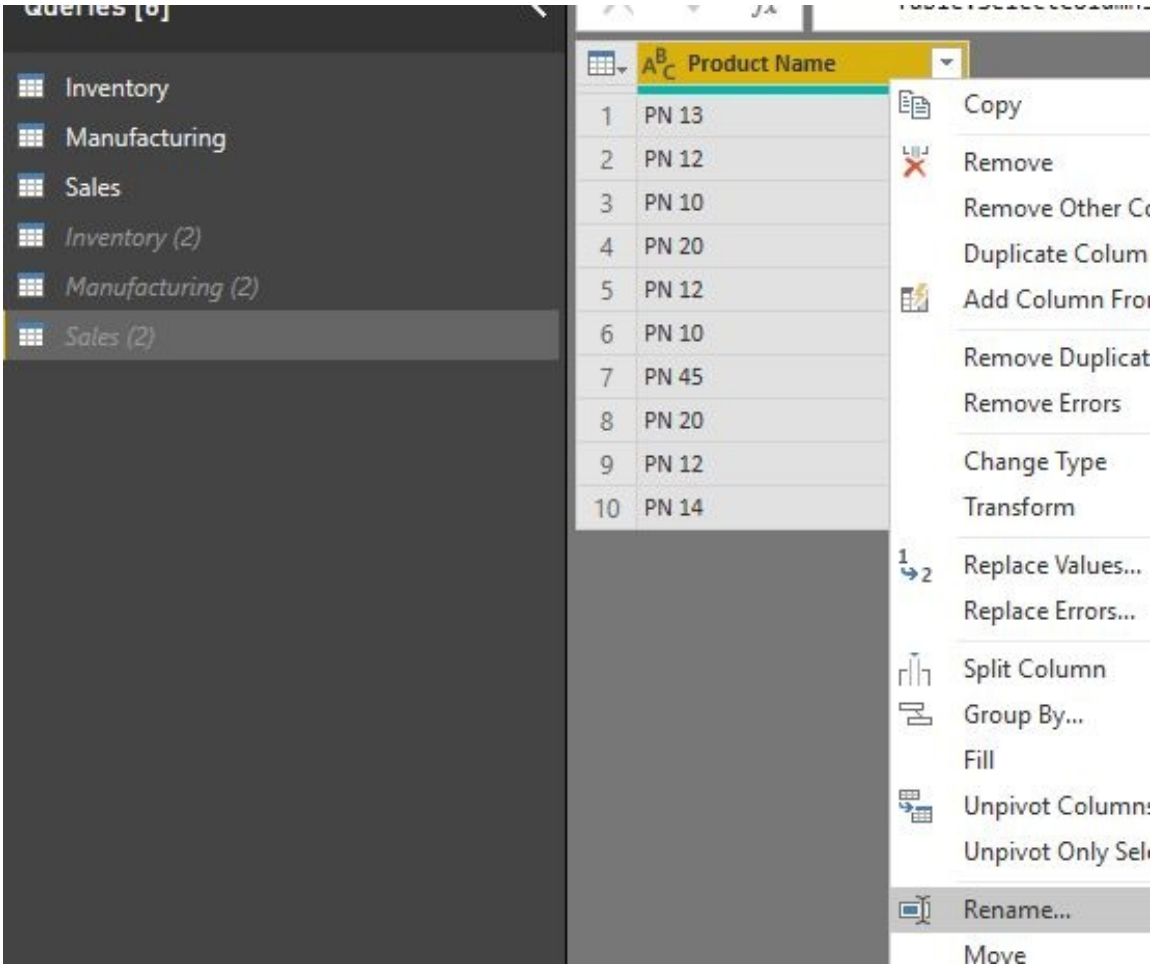


Figure 06-16: Renaming Product Name column in Sales (2) to be Product. All columns should be the same name.

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.

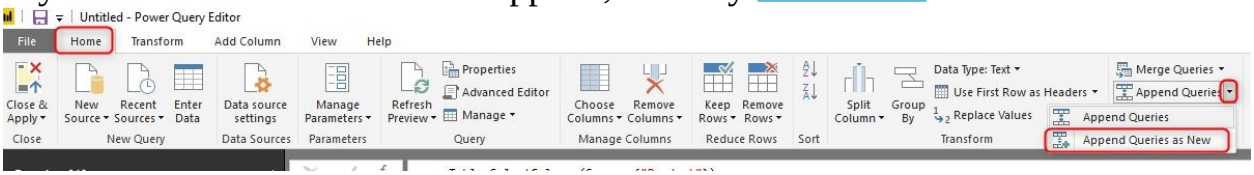


Figure 06-17: Append Queries as New

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

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.

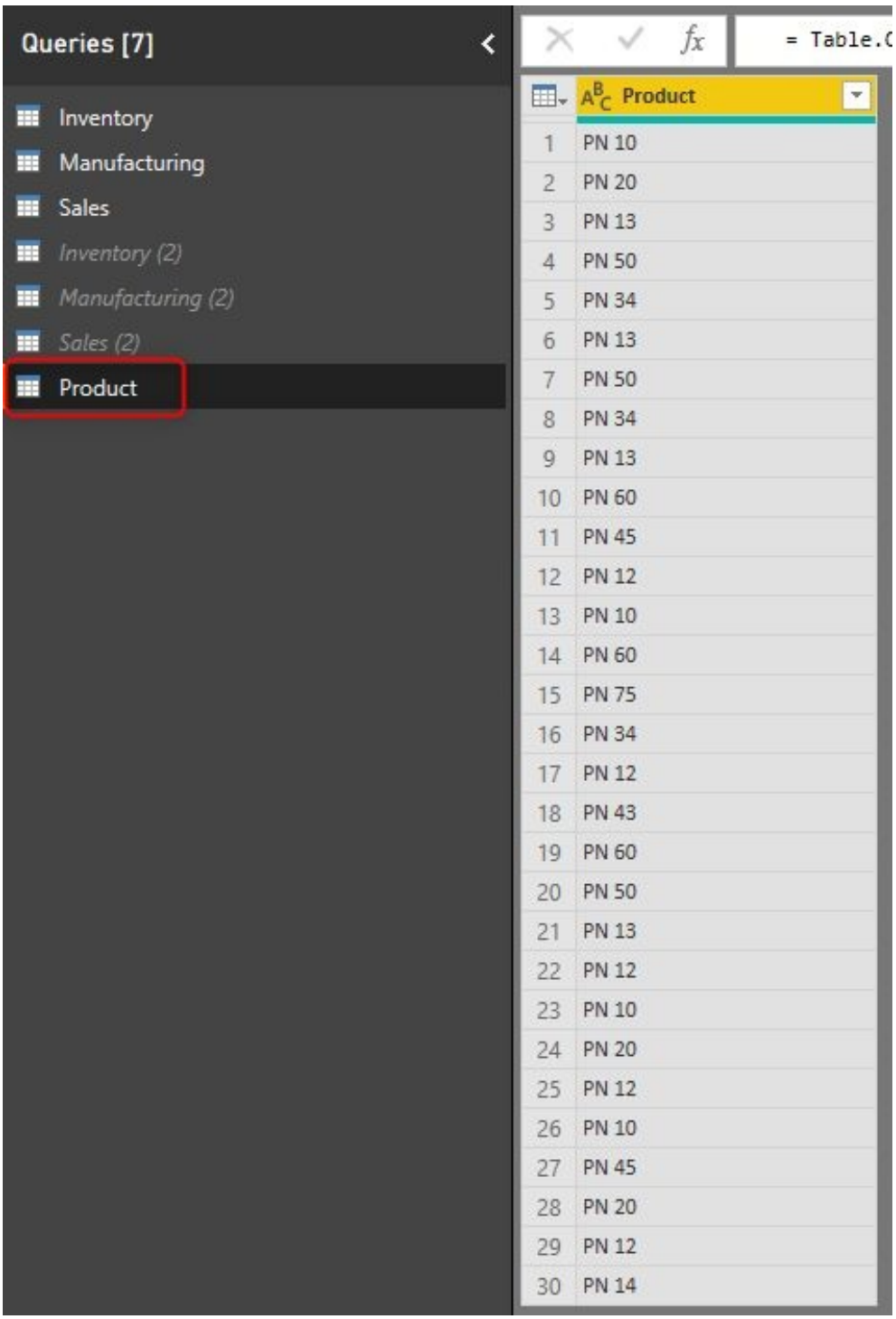


Figure 06-19: Product table appended the result.

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:

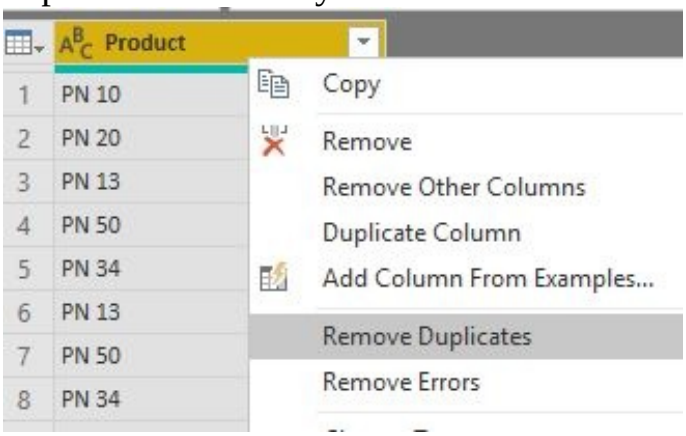


Figure 06-20: Remove Duplicates from the Product table

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.

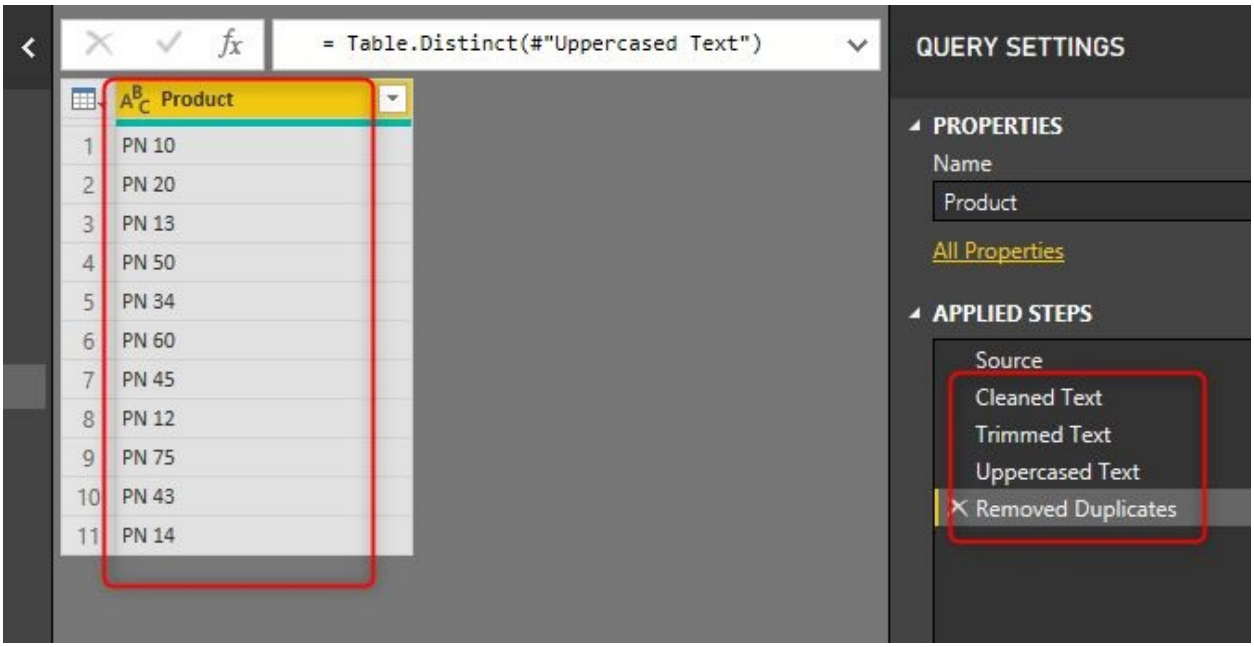


Figure 06-21: Product table with all the steps. Master List created.

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;

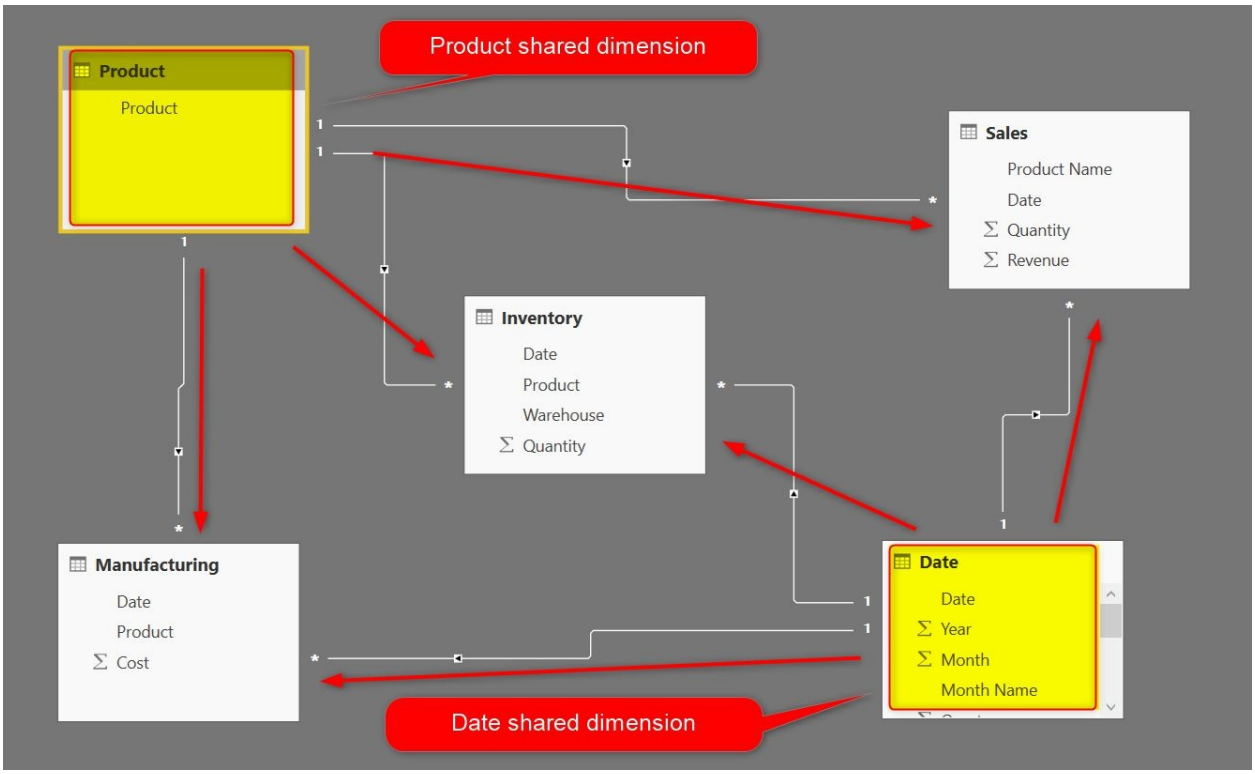


Figure 06-22: Solution Diagram with Date and Product Shared Dimensions

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:

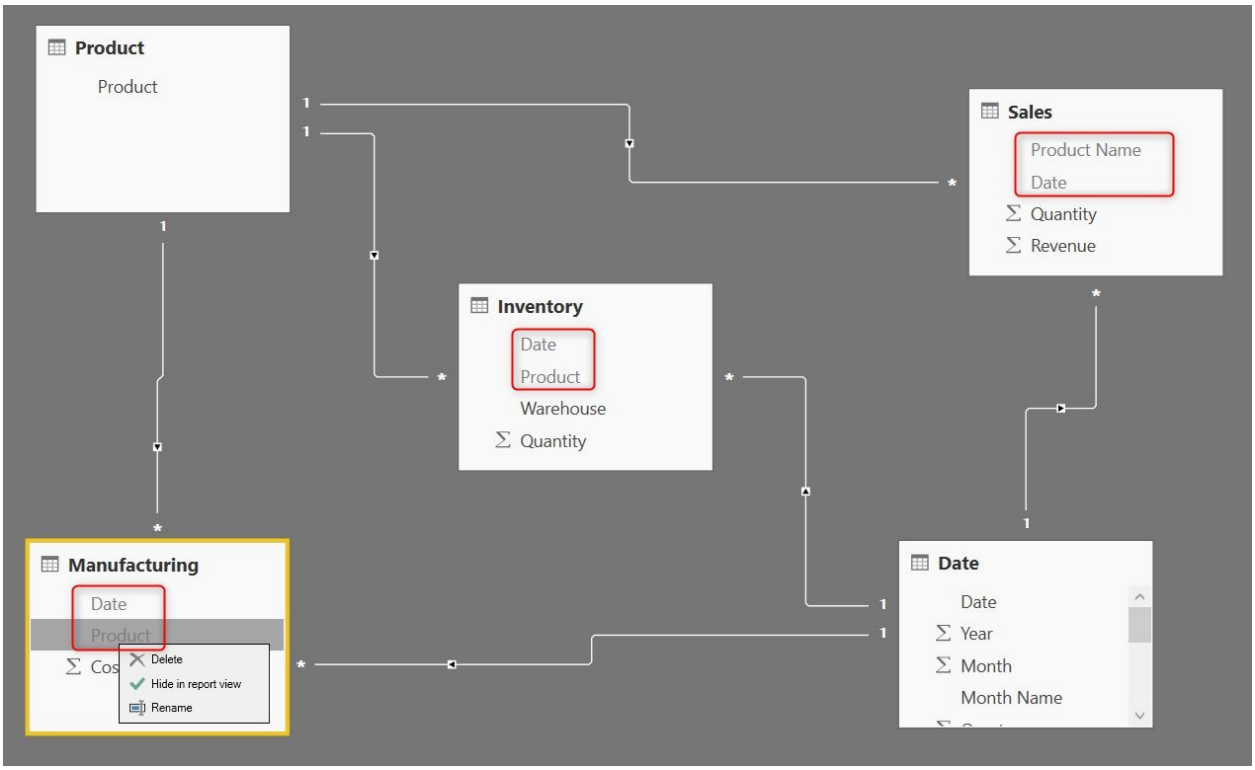


Figure 06-23: Hide Date and Product Column in Non-Dimension tables.

This solution can now provide correct reporting capabilities as below;

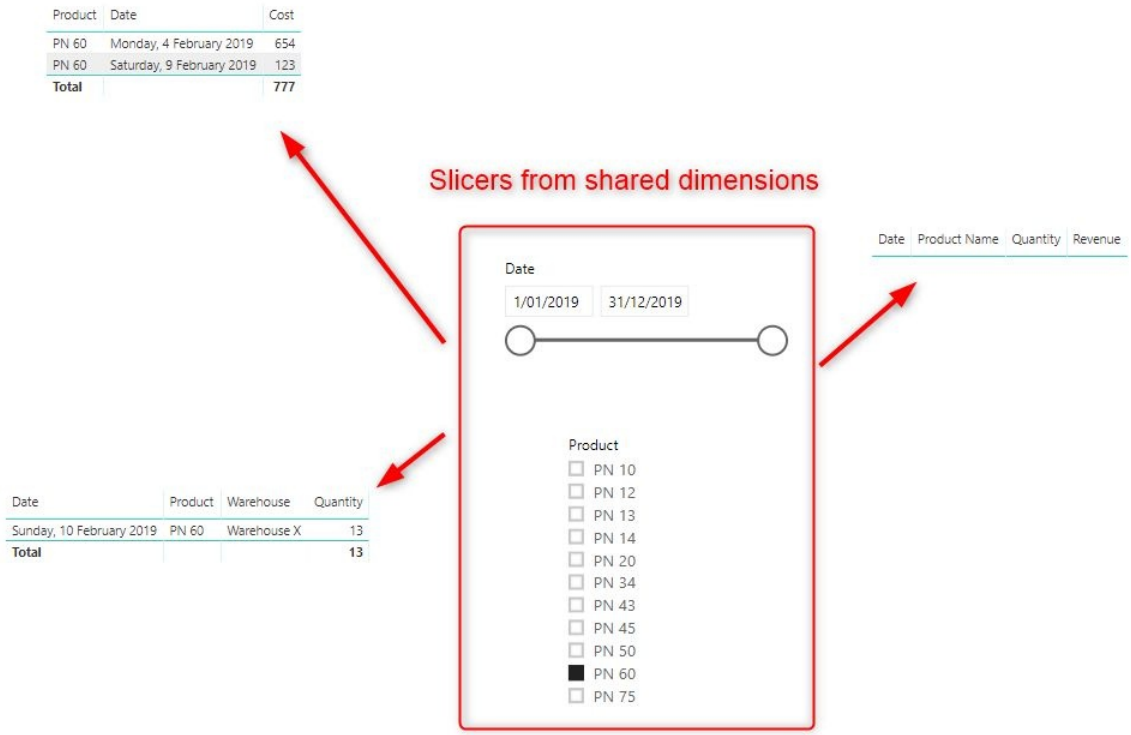


Figure 06-24: Testing the result

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://radadcad.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.

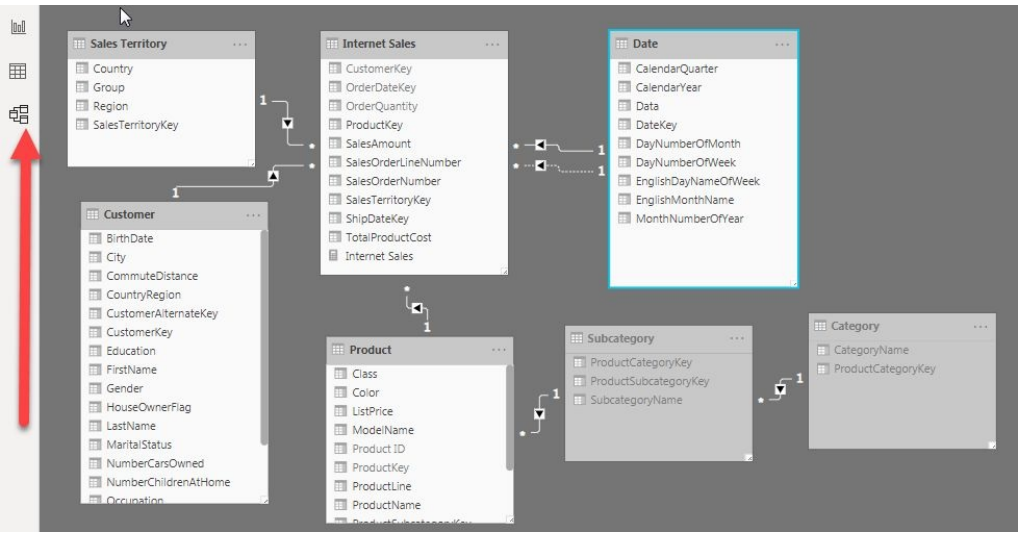


Figure 07-01: Internet Sales Data Mart Relationships

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 repopulate 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.

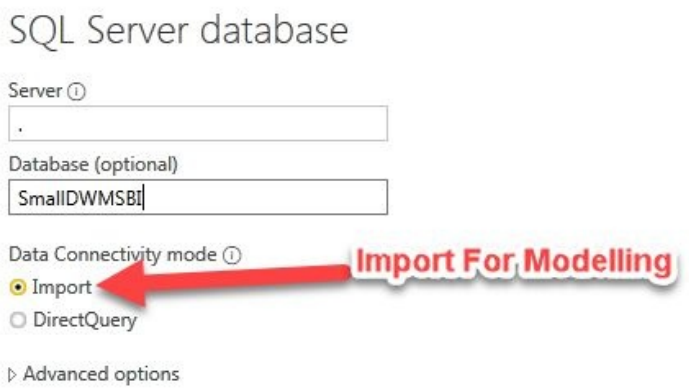


Figure 07-02: Import for Data Modeling

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.

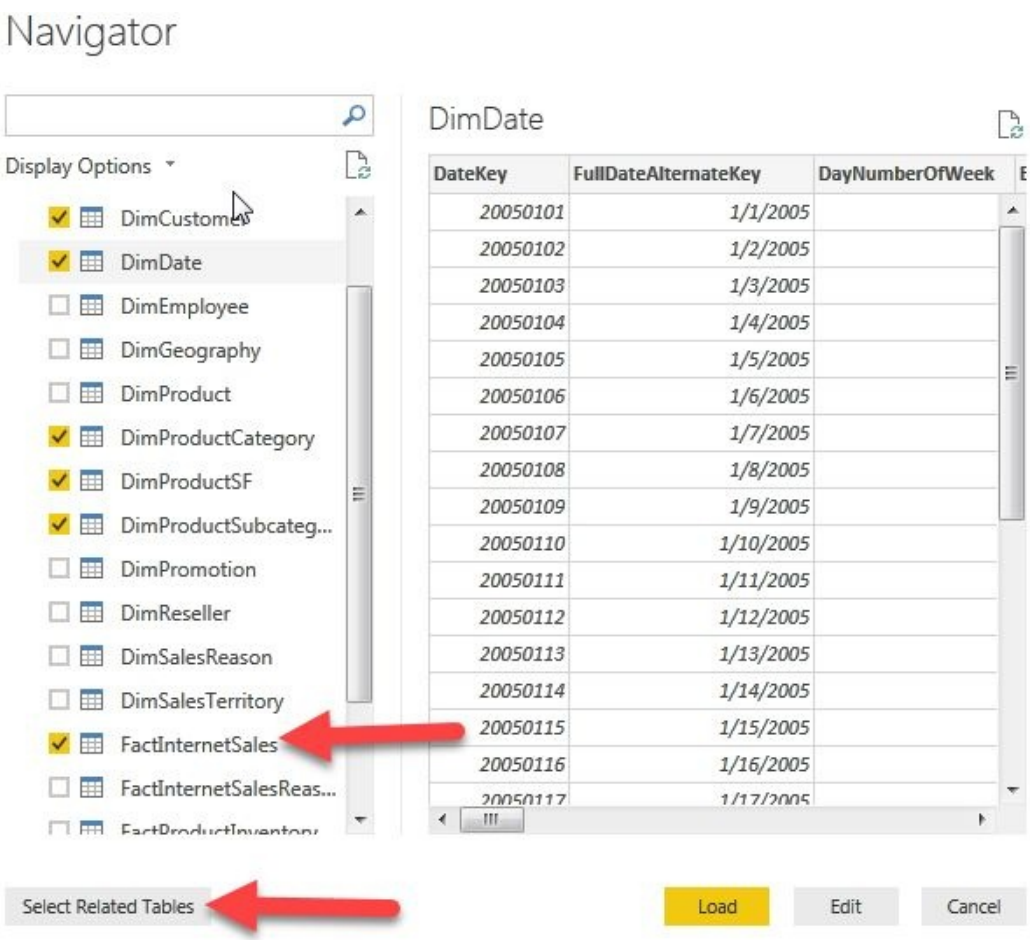


Figure 07-03: Importing Tables

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.

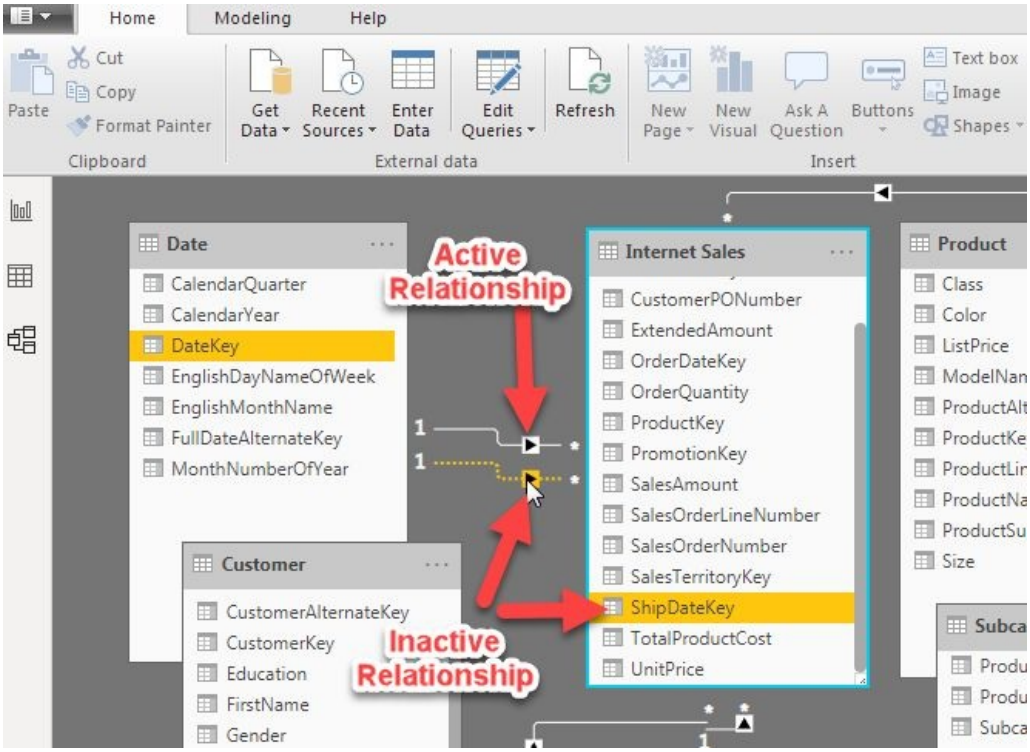


Figure 07-04: Multiple Relationships between Fact and Dimension

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.

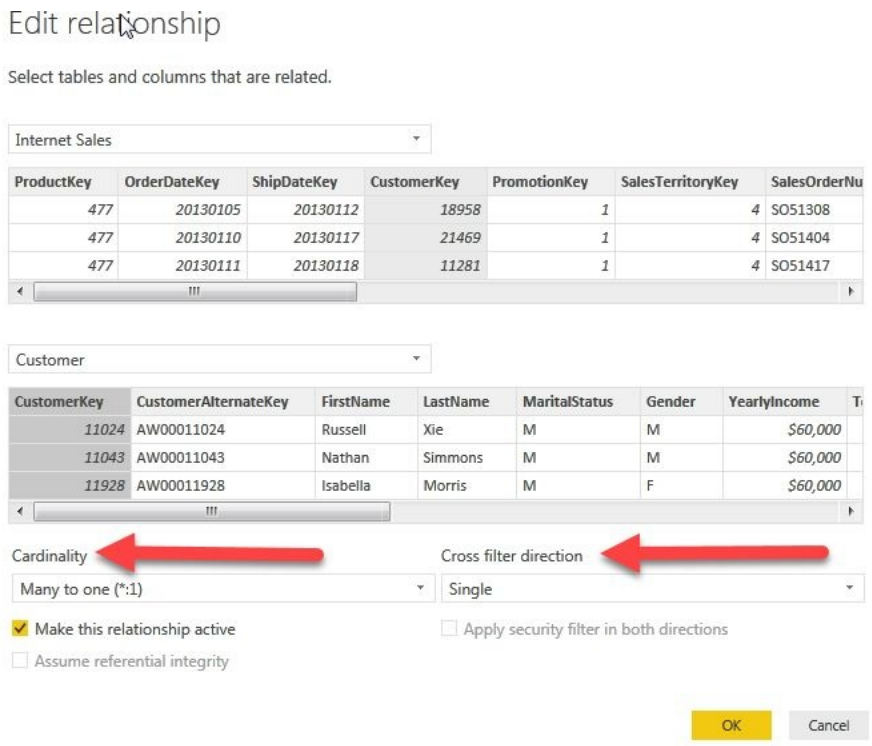


Figure 07-05: Edit Relationship

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.

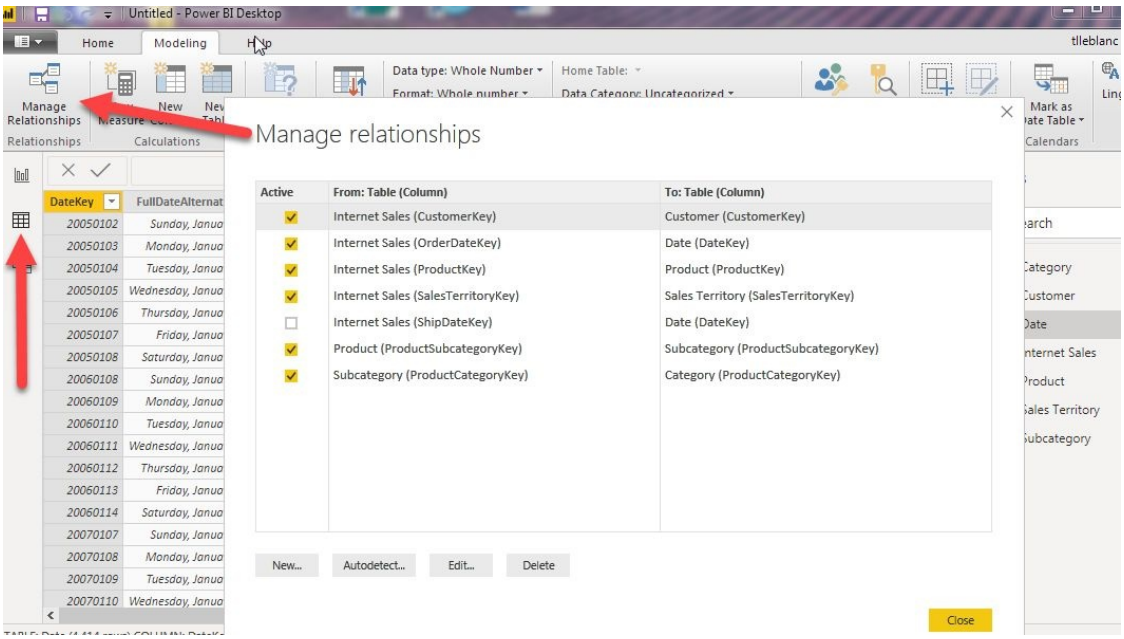


Figure 07-06: Manage relationships

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).

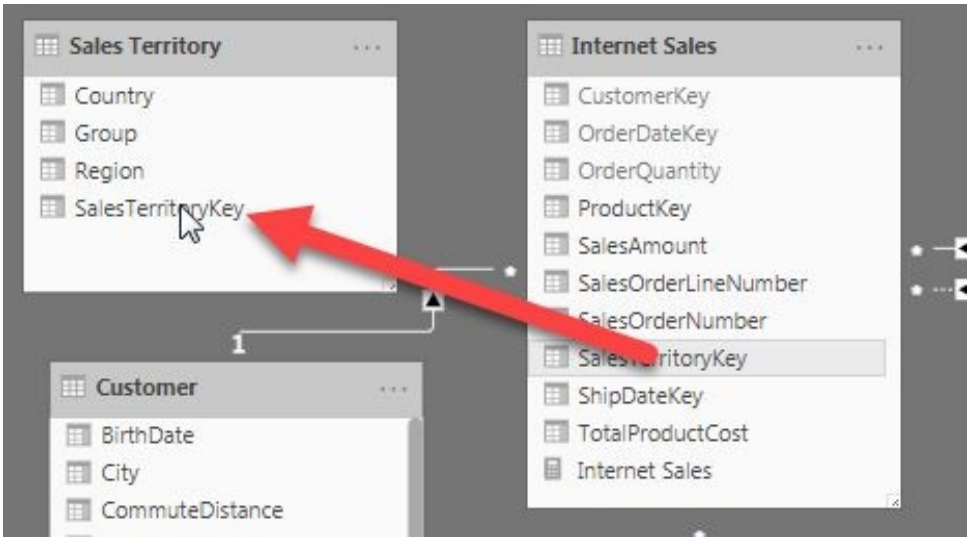


Figure 07-07: Create Sales Territory Relationship

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

sliced by Martial 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.

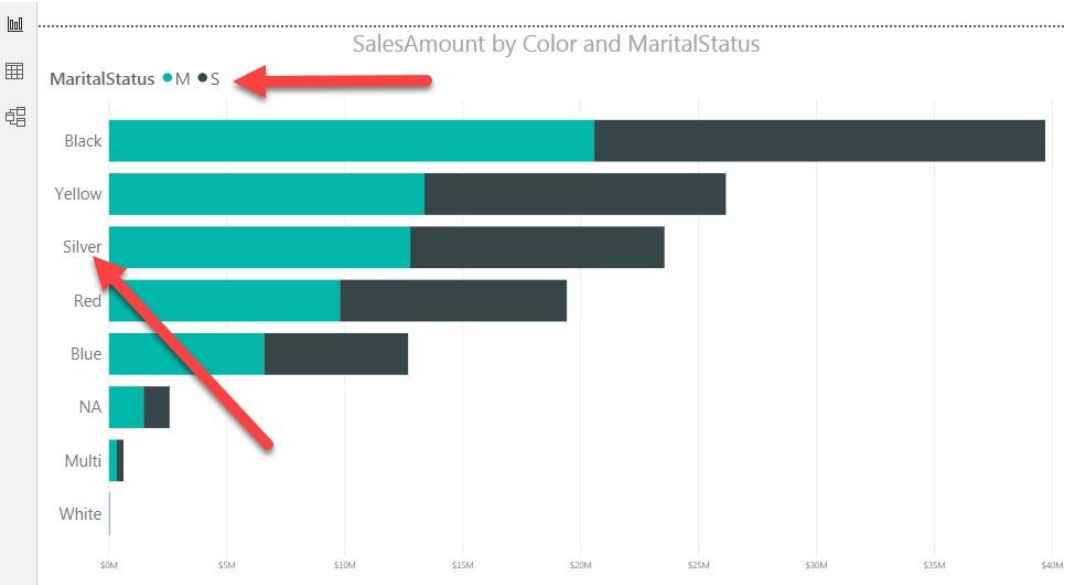


Figure 07-08: Slicing and Dicing Internet Sales Amount

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.

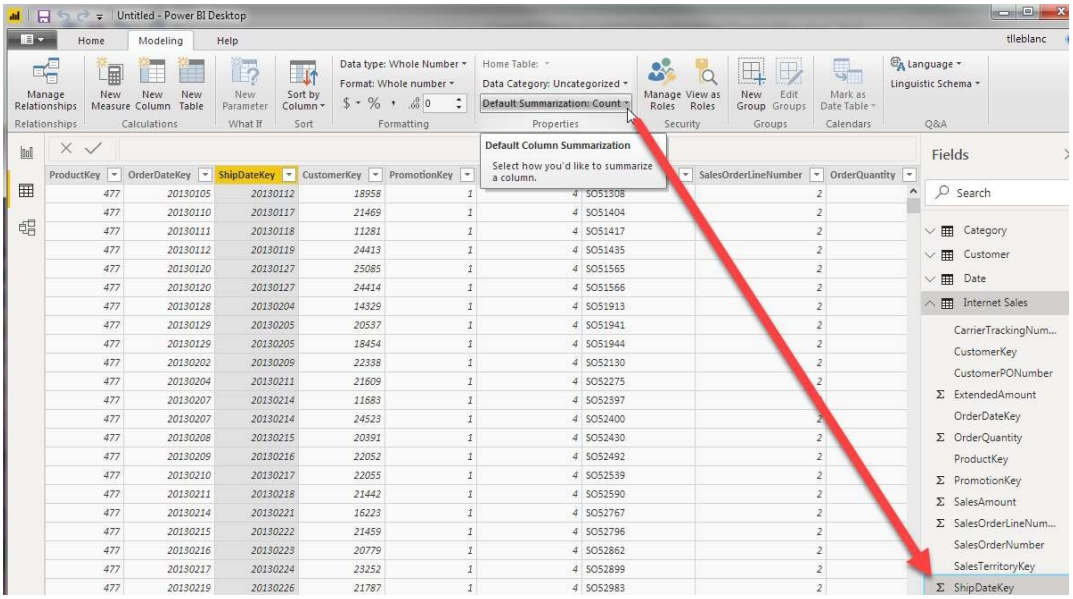


Figure 07-09: Count Default Summarization for Key Columns

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.

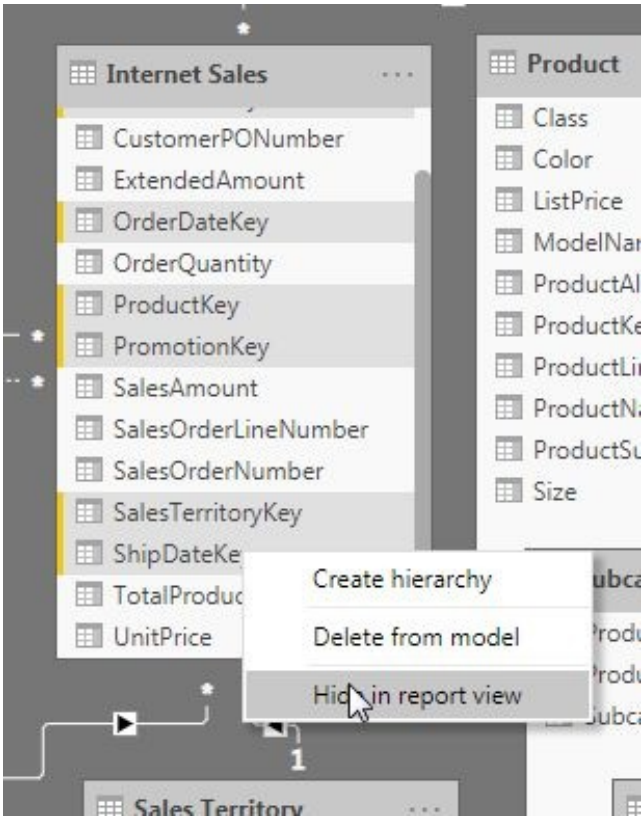


Figure 07-10: Hide Key Columns

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.

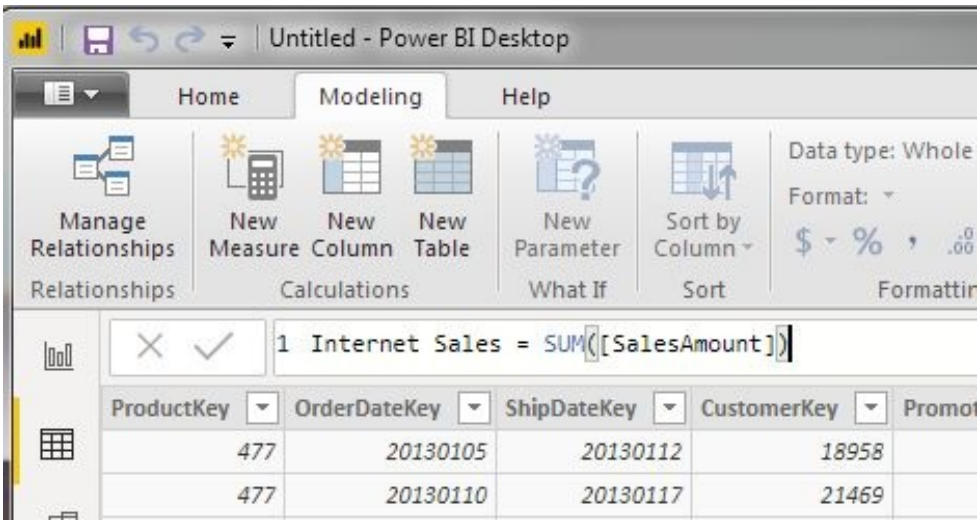


Figure 07-11: Internet Sales Measure

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.

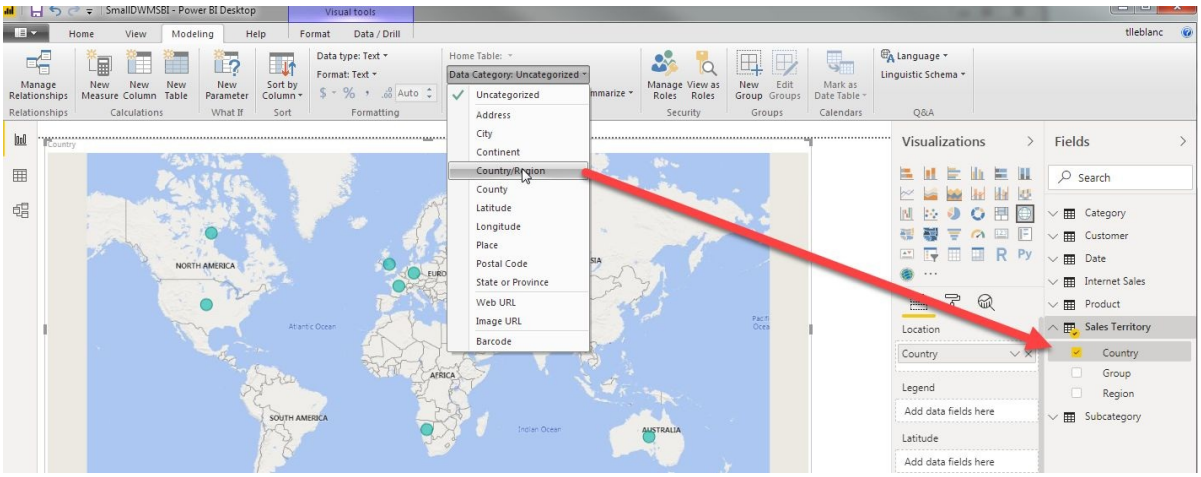


Figure 07-12: Country Category for Map Visual

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

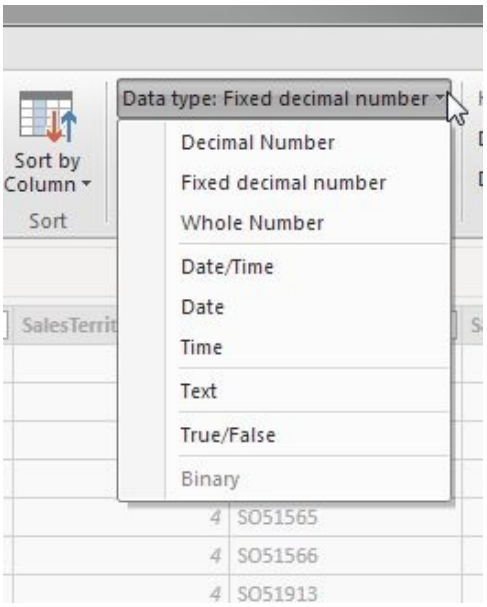


Figure 07-13: Data Types

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

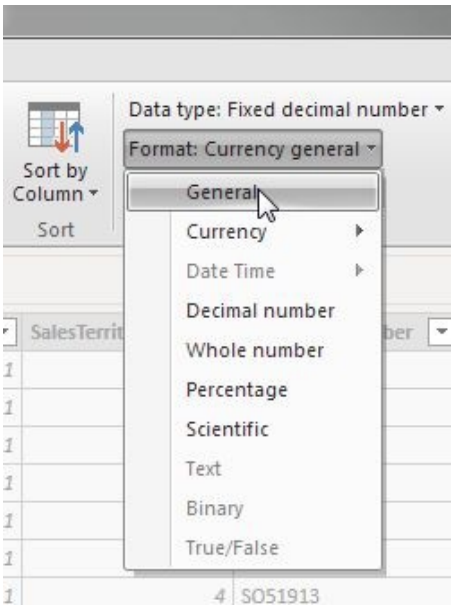


Figure 07-14: Column/Measure Formatting

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.

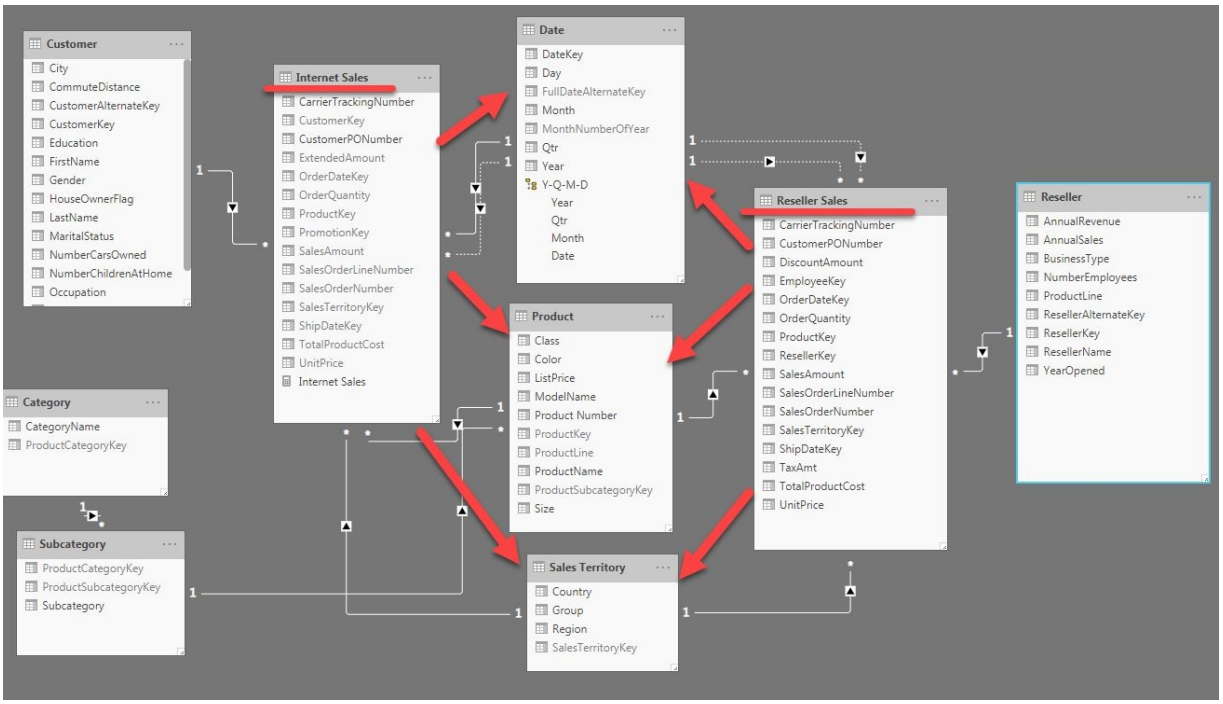


Figure 07-15: Multiple Fact Tables

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.

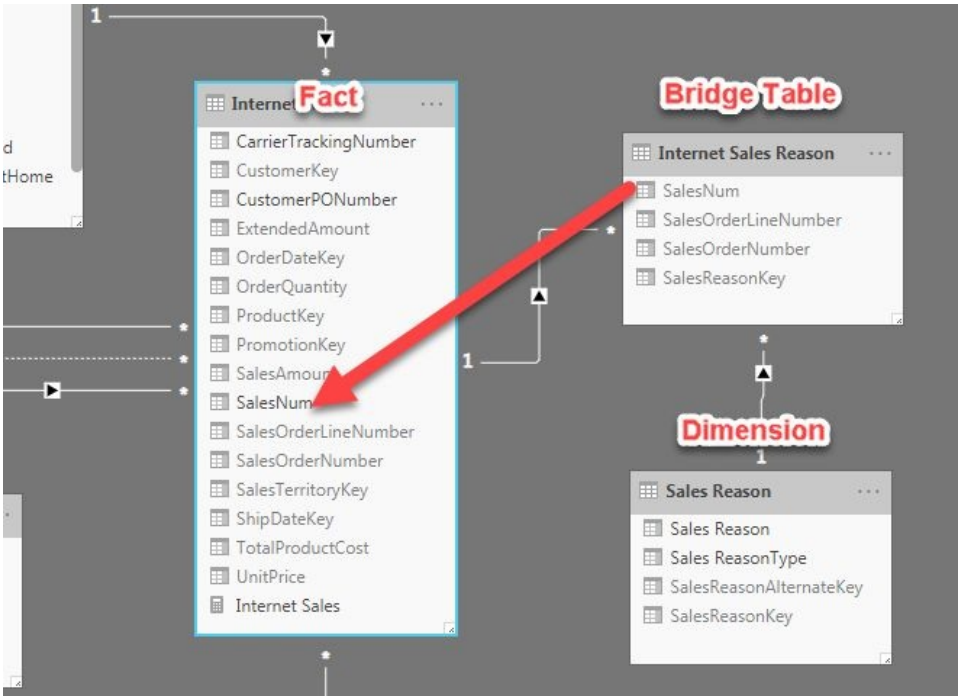


Figure 07-17: Many-To-Many Relationship

The last step would be to change the relationship to Both in the relationship property Directional cross filter. The 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.

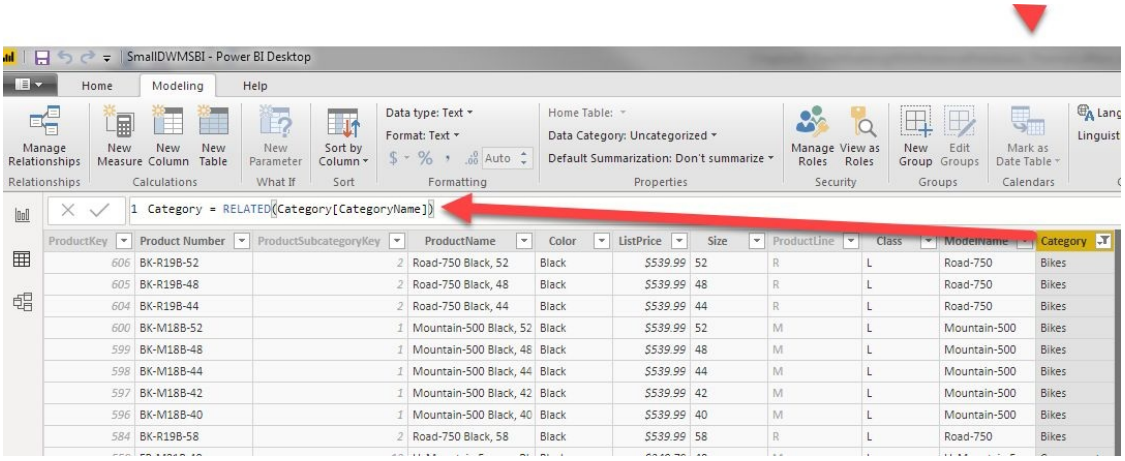


Figure 07-18: RELATED() DAX Function

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).

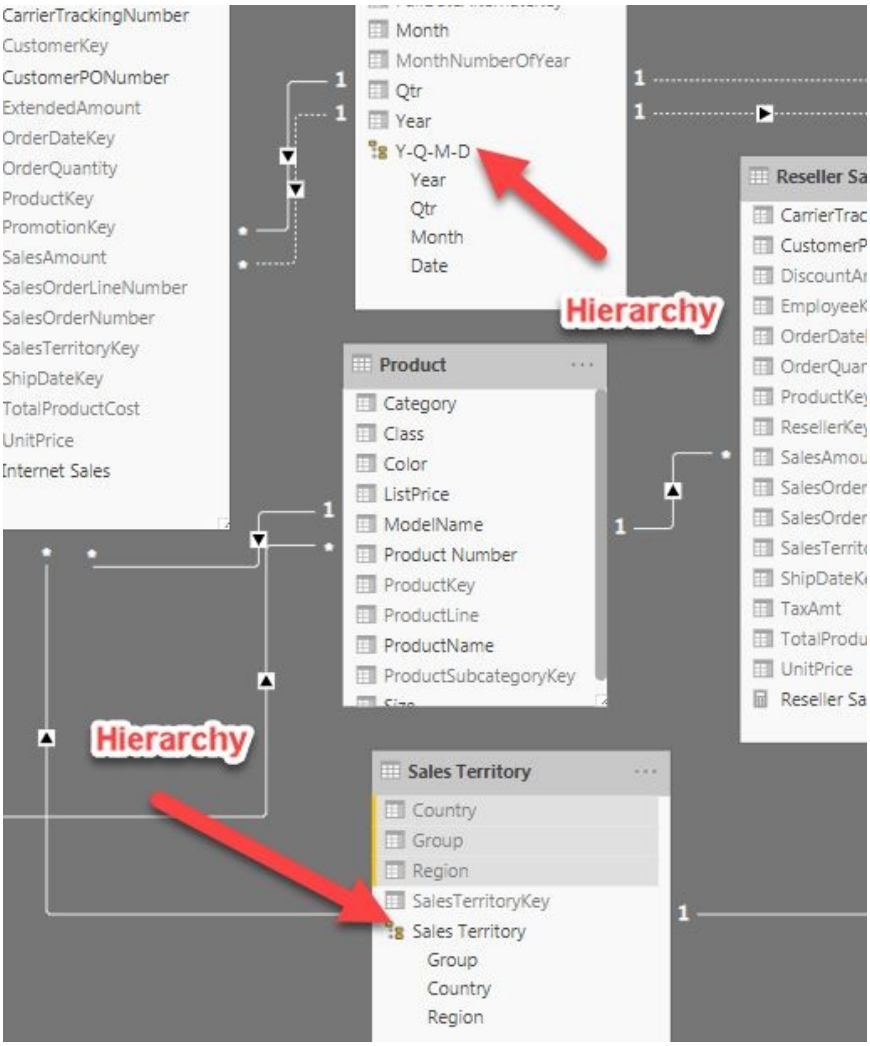


Figure 07-19: Y-Q-M-D and Sales Territory Hierarchy

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.

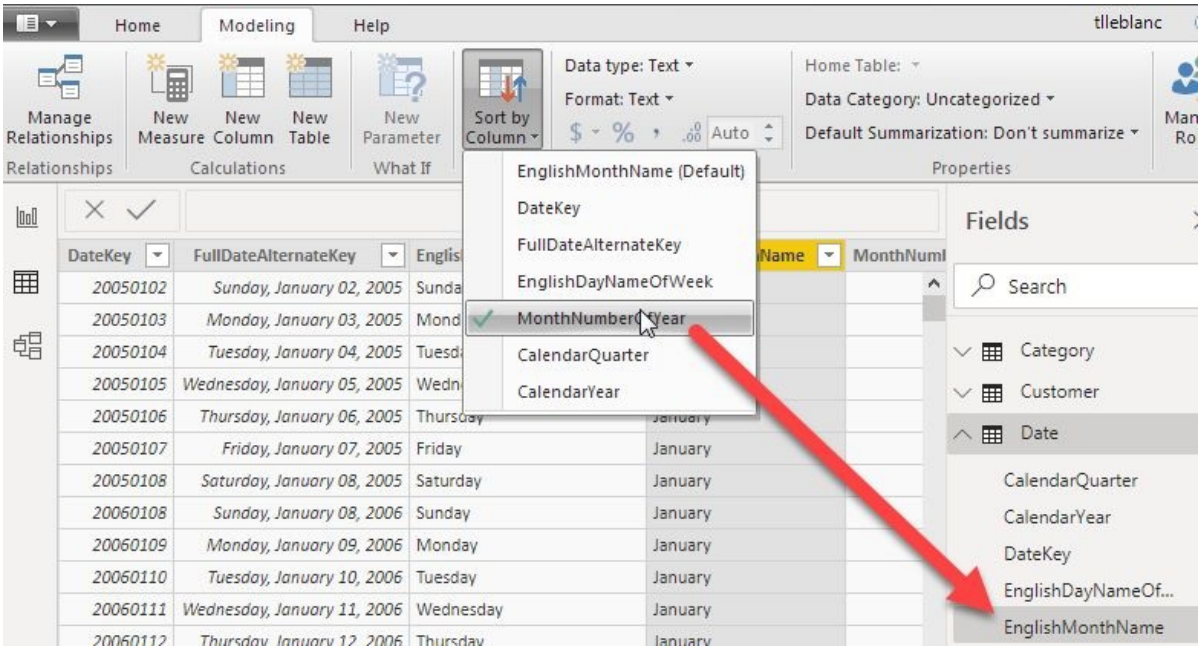


Figure 07-20: Sort By Column

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 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

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."

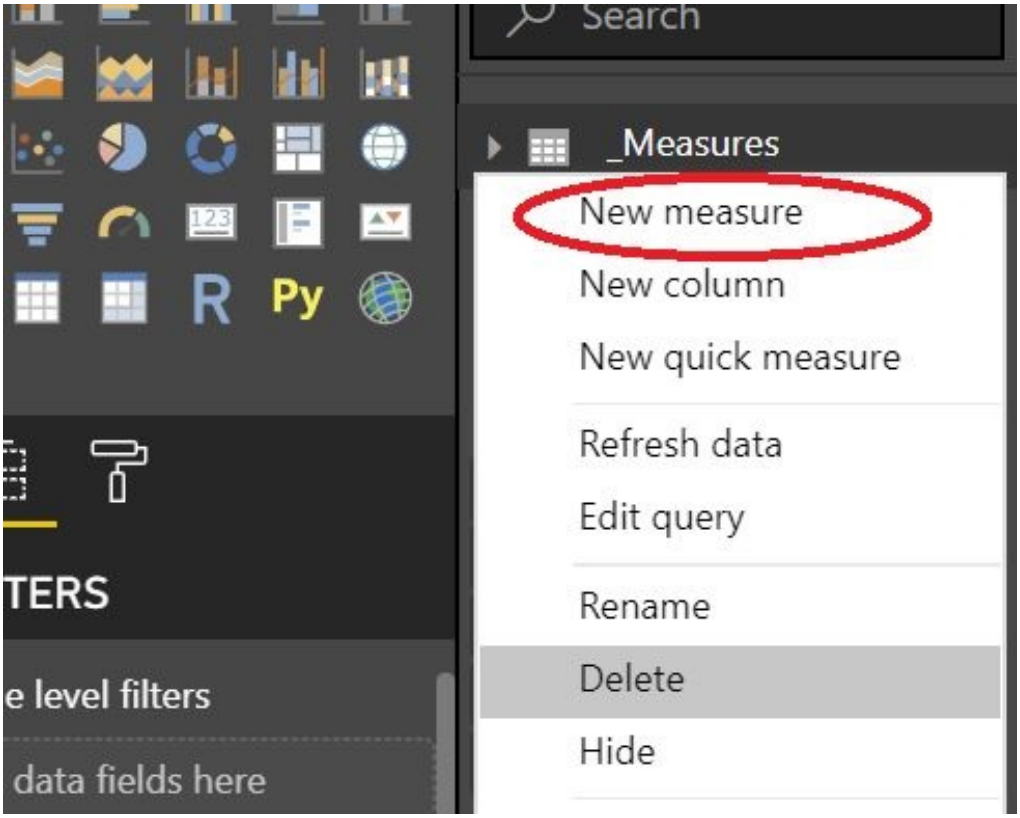


Figure 08-05: The New Measure Button

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:

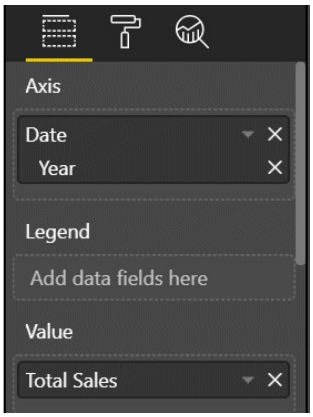


Figure 08-06: The New Fields Pane

6. Your stacked column chart should look like this:

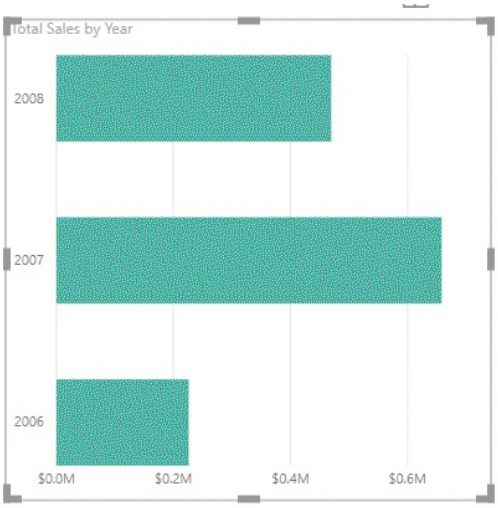


Figure 08-07: The Stacked Column Chart with Total Sales

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 the break it down.

You did it! You just created your first measure.

DAX Expressions always breakdown like the following diagram:

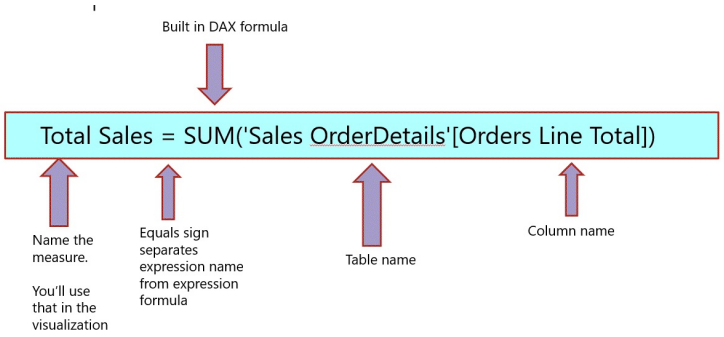


Figure 08-08: The DAX Expression Breakdown

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.

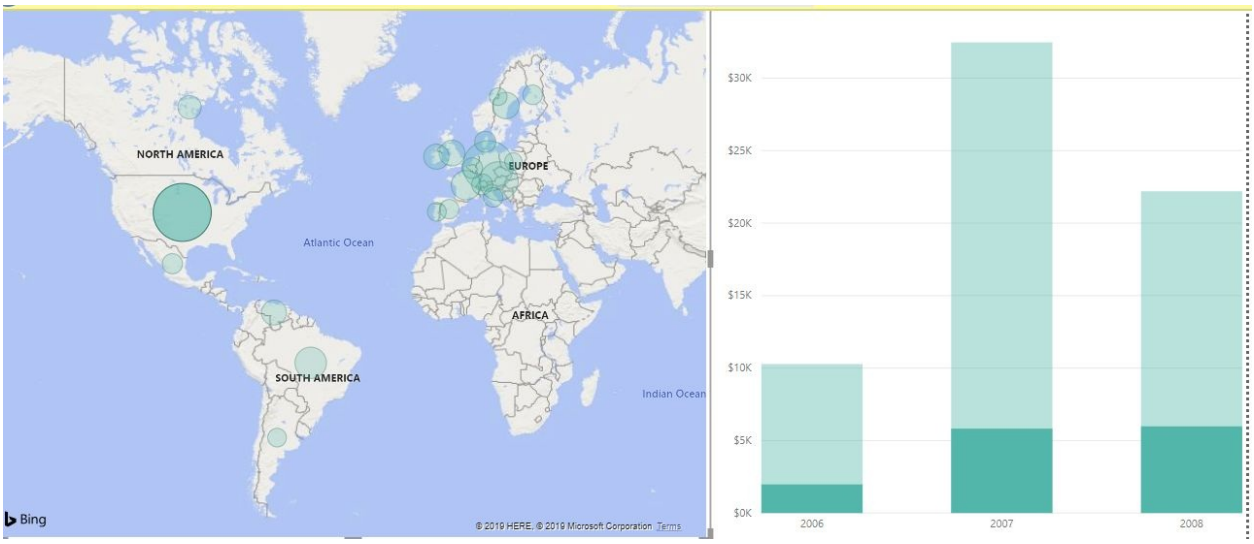


Figure 08-09: The Column Chart affecting the Map visual

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:

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
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:

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.