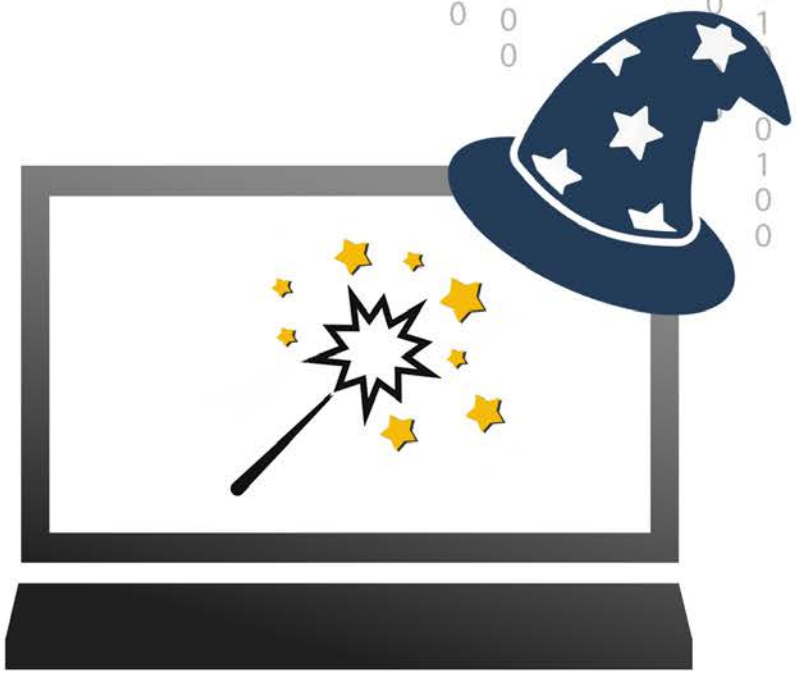
[M]AGIC TRICKS FOR DATA WIZARDS



TIPS & TRICKS FOR POWER QUERY IN EXCEL & POWER BI

Ken Puls and Miguel Escobar

Magic Tricks for Data Wizards

Tips & Tricks for Power Query in Excel & Power BI

by

Ken Puls &

Miguel Escobar

www.powerquery.training

Magic Tricks for Data Wizards

Authors: Ken Puls and Miguel Escobar

Layout: Power Query Training

Copyediting: Power Query Training

Technical Editor: Power Query Training

Design: Power Query Training

Illustrations: László Vanger & Power Query Training

Cover Illustration: László Vanger

Published online by Power Query Training on January 9, 2017

Table of Contents

Our Goal with this eBook

In the online Power Query workshops that we deliver, we commonly talk about how data professionals can often perform **"Magic"** with their data in terms of transforming, cleaning or reshaping a dataset for better consumption and analysis. We often call these data professionals "Magicians" or "Data Wizards".

Why Data Wizards? The term wizard often describes "a person who is very skilled in a particular field or activity,"' and in computing the term wizard means "a help feature of a software package that automates complex tasks by asking the user a series of easy-to-answer questions." The first definition is for you, the Excel user, and the second definition is for Power Query – together you make magic.

This eBook is written from a practical point of view with easy-to-follow examples. We've compiled the most impactful cases that we've encountered over the years to show you how to work more effectively with Power Query based on the scenario that you have.

The goal of this eBook is to help you, as a Data Wizard, learn some new tricks and tips when working with this amazing tool called Power Query.

- **Each chapter of this eBook is independent from one another**, so you can jump to a chapter of your choice and get the information that you are looking for without the need to see the rest of the eBook. However, we do encourage you to check the full contents of this eBook.
- **We've made sure that this is a short and concise eBook** that you can be able to read in just one sitting while still learning everything that you need to know to work more effectively with Power Query.
- **Use your preferred tool! –** you can follow along our examples with either Power Query for Excel or in the Power BI Desktop

Some of the content that you'll see in this eBook is based on the patterns that we've published online. However, after years of putting them to the test we've created optimized versions that you'll be the first to see in this eBook.

Start here if you are new to Power Query

Power Query is an amazing free new add-in for Microsoft Excel and is also part of the Power BI Desktop as the main component of its "Get Data" experience.

If we could put Power Query in simple terms, we would say that it's the best data transformation / manipulation / consolidation tool that Microsoft has created for end-users **ever!** We truly mean that.

There is no prior knowledge required for you to follow along or understand the concepts written in this eBook, but be sure to click the button below to visit our resource page so you can find out what Power Query is and how to get it before you continue reading.

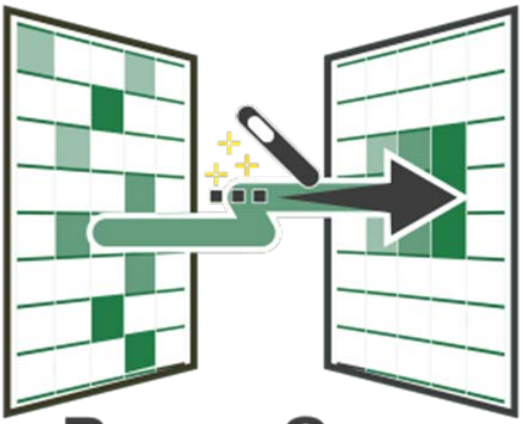
What is Power Query?

Special Message from the Authors

We'd like to take this opportunity to thank you for subscribing to our newsletter and let you know that we have big plans for 2017. We'll be keeping you up-to-date with all the exciting news and important information on new Power Query features and updates. This eBook is only the beginning.

This is also a great opportunity to thank the hundreds of "Data Wizards" that have participated in our live online workshops and the thousands that have bought our other book, *M is for Data Monkey*. The feedback that we've received from all of you has been superb and we always strive to deliver the highest quality of content that we possibly can.

If you'd like to **kickstart your Power Query journey**, or simply want to take your current knowledge to a whole new level, then we highly recommend taking our online live workshop or purchasing *M is for Data Monkey*. We update the workshop content for each session to keep it fresh and current, so you can rest assured that you'll get **the most engaging and up-to-date workshop on the market**.



Power Query Workshop

Check the course Calendar

Chapter 0: The Structure of the eBook

This eBook has a straightforward structure of six completely independent chapters, which means that you can jump straight into the one that drives your attention the most.

You'll notice that on the top part of the first page of each chapter we've added a 'Difficulty' category as follows:

- **Basic –** simple scenarios that can be solved by easily using the UI buttons
- **Intermediate –** scenarios that require simple Power Query formula usage
- **Advanced –** scenarios where manually written Power Query code is needed

The intention with this category system is to give you an idea about the topics and methods that we'll be covering in each chapter.

Do not get scared of reading a chapter labeled "Advanced". The actual advanced part is the technical aspects behind the Power Query engine, but the implementation and execution of each step in all of these chapters is relatively simple and straightforward.

Each chapter is broken into three basic sections:

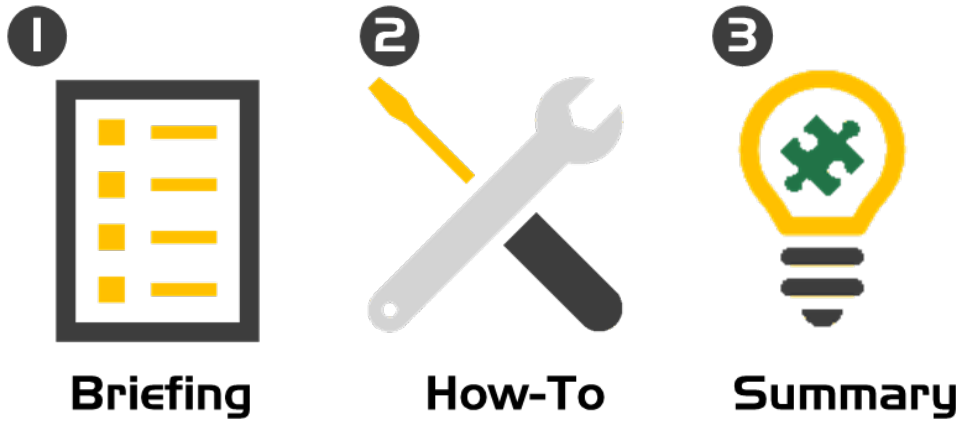


Figure 1. The structure of each chapter.

1. **A brief explanation of the scenario at hand** – which outlines the data that we have and what we would like to accomplish
2. **A guided solution to the scenario** – which delivers a simple and concise how-to guide that will help you solve the scenario using Power Query
3. **A case summary** – which provides an overview of all the topics covered in the chapter, along with our personal recommendations and thoughts for similar cases

Our goal is to give you a simple format that you can follow along across all chapters and that you can revisit at any given time.

eBook Version, Suggestion Box and Companion Files

Thanks to the power of the internet, we can make corrections and updates to this eBook when needed, and notify you of the new version because you are a subscriber.

We have created a feedback page where you can submit your questions, suggestions or simply your comments about this eBook. Simply click on the image below to tell us what you think!

When you downloaded this book, six folders for the companion files of each chapter were also included. In each folder you should find a file with a name ending with "Start' and another one ending with "Finished". Use the "Start" file to follow along with the solution outlined in chapter and use the "Finished" file to check your results at the end.

We are actively searching for new cases to write about. If you have any suggestions, comments or feedback that you'd like to give us, please visit the feedback page on our website by clicking the following image.

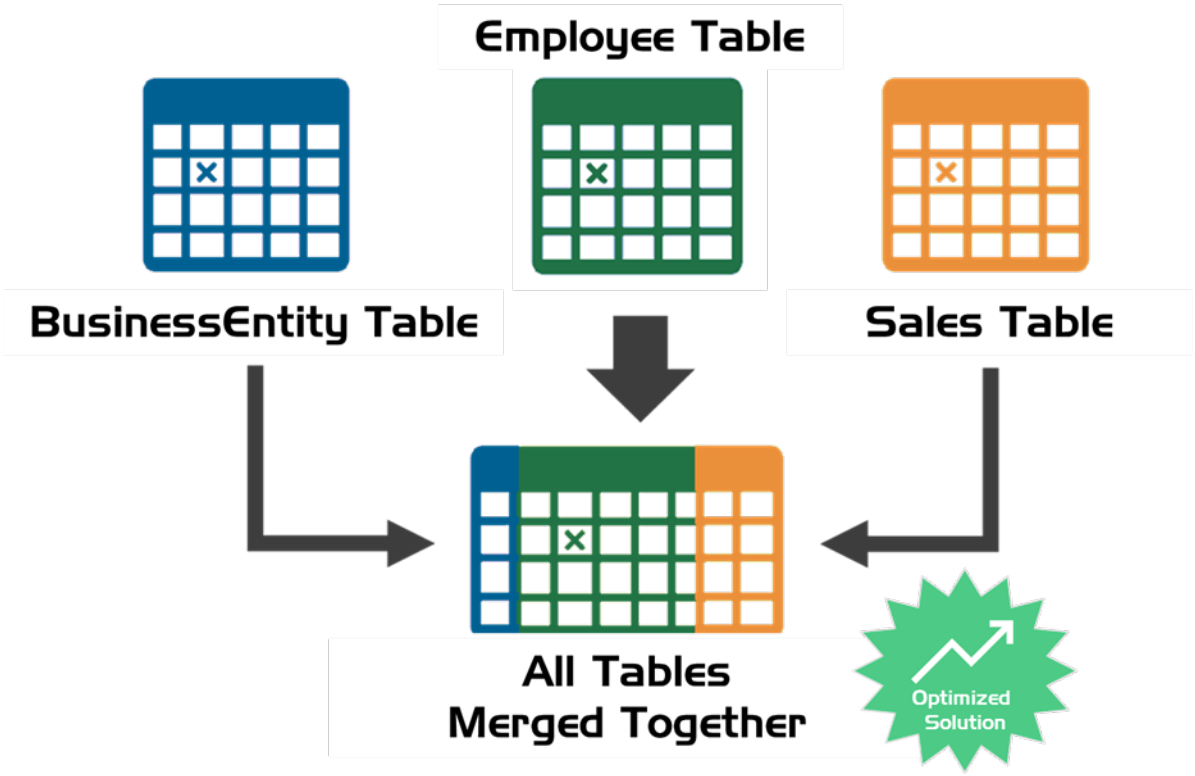
Tell us what you think!

Master Power Query and become a Data Wizard

Take our Power Query Workshop

Chapter I: Merge Tables

Difficulty: Basic



Briefing: Merging multiple tables together

We have three tables:

- **Employee –** this is a table with all the Sales employees information
- **BusinessEntity –** each employee is assigned to a specific business entity in their region. This is a reference table for all those business entities in every region.
- **Sales –** this table holds all the sales made by each of the sales employees

Our Goal

- To merge the Employee and Business entity tables so we can later group by business entity in each region
- To create a report for the overall top 10 performing employees, based on their sales numbers

Be sure to open the workbook "Merging Tables – Start.xlsx" so you can follow along.

The How-to Guide: Merging Tables

With the **"Merging Tables – Start.xlsx"** file open:

1. Go to the **Employee sheet** and click on any part of the table that you see on the screen. Then go to the Power Query ribbon and select the **"From Table" option**.

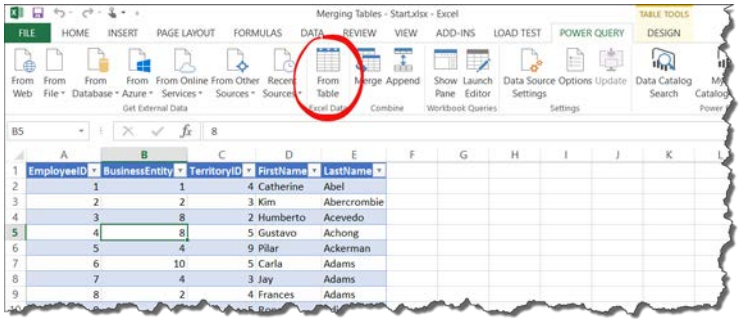


Figure 2. Be sure to select any cell within the table before clicking on the "From Table" button.

2. The previous action will pop up the Power Query window called the **"Query Editor"**. In this new window, you need to select, "Close & Load" (the first icon on your left), and from the drop-down menu select the option that says **"Close & Load To…"**

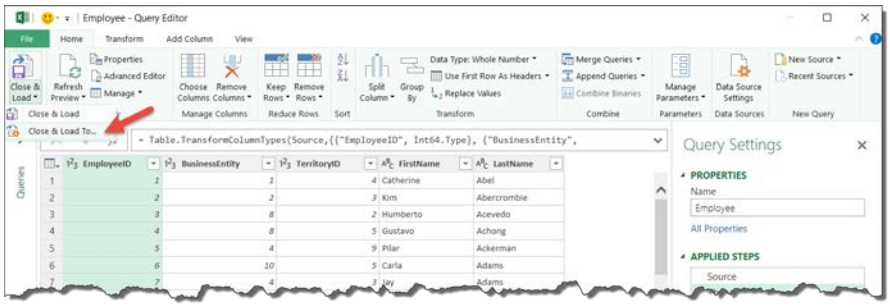


Figure 3. Be sure to select the "Close & Load To…" option from the drop-down menu.

3. This will trigger a new pop up window called "Load To" that will ask you where you'd like to load your data. Select the option that says **"Only Create Connection"** and click the "Load" button.

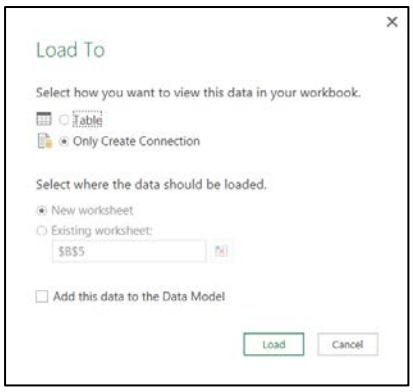


Figure 4. Select the option to "Only Create Connection" from the "Load To" window.

4. Repeat steps 1-3 for the tables inside the Sales and BusinessEntity sheets to end up with three separate queries inside Power Query. Once you've finished this process, go back to the Power Query ribbon in Excel and click the **"Launch Editor"** icon to go back to the Query Editor.

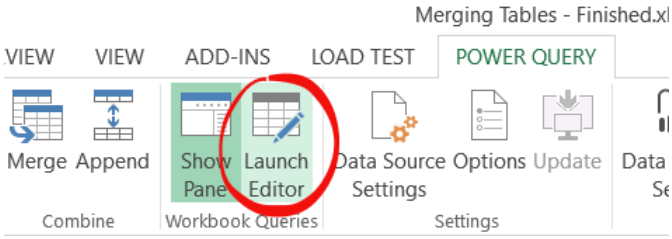


Figure 5. Click the "Launch Editor" icon to go back to the Query Editor.

5. In the Query Editor, select the Employees query from the left pane so we can start editing that specific query.

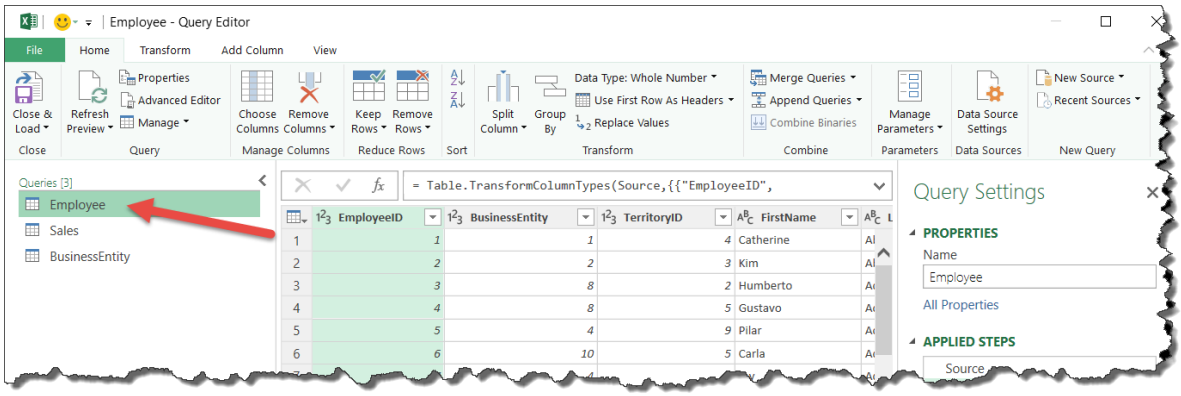


Figure 6. Click the Employee query to edit that query.

This is where the fun starts. We need to merge this Employee table with the data from the BusinessEntity table using both the BusinessEntity and the TerritoryID columns. In Power Pivot, you're only able to create relationships between two tables using just one column from each. Trying to do a VLOOKUP with two columns is simply not possible.

The usual workaround would be to concatenate both the TerritoryID and BusinessEntity columns to create a unique value that we could later use as a key to relate these two tables**. With Power Query you do not need this workaround**.

6. Go to the Home tab of the ribbon and click on the **"Merge Queries"** icon found in the "Combine" section. A new window called "Merge" will appear where you'll be able to merge your current query with other queries.

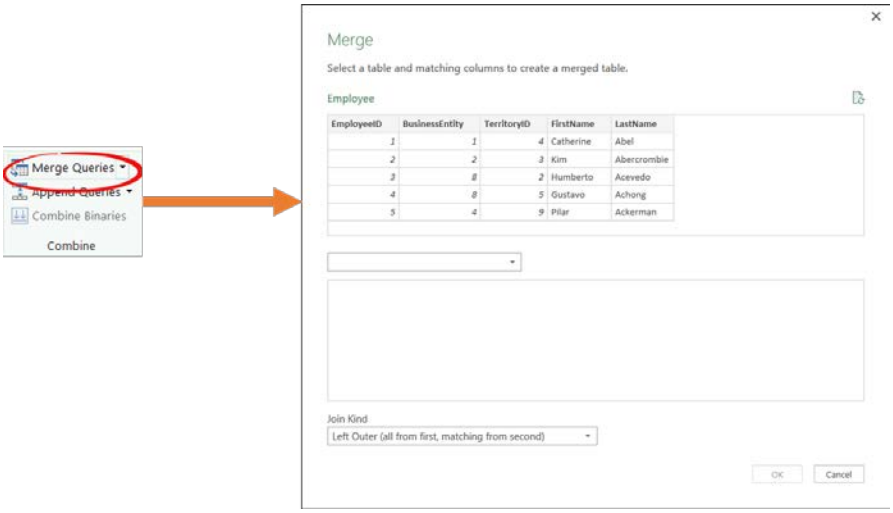


Figure 7. Click the "Merge Queries" icon and a new "Merge" window will appear.

7. Your next step will be to select the BusinessEntity query from the blank drop-down menu, and then select **TerritoryID** from both the Employee and the BusinessEntity queries

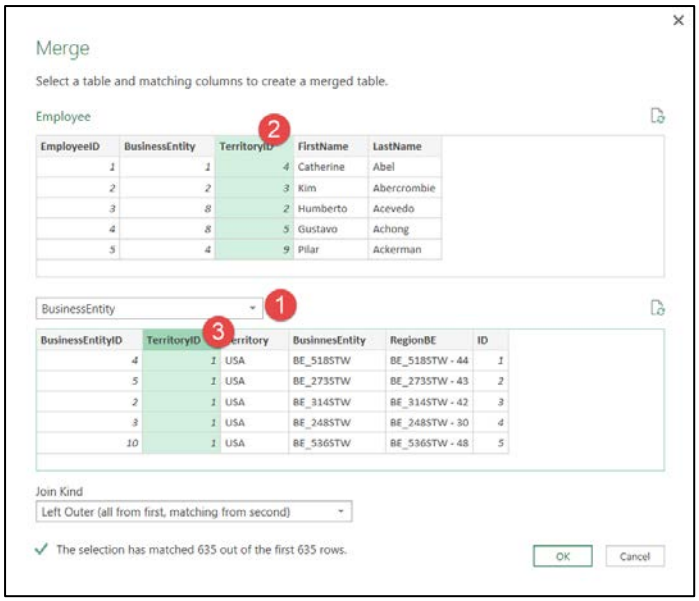


Figure 8. Select the BusinessEntity query from the dropdown (1); then select the TerritoryID column from each query (2 & 3).

8. Now, here's the cool part: if you wish to merge these tables based on multiple columns, **Territory** AND **BusinessEntity** for example, then all you need to do is press the **Ctrl key** on your keyboard and select the other columns to include. You can add as many pairs as you want, but just make sure that you select these columns in pairs, one column from the query on top and the matching column from the query on the bottom. Once you've completed the pairings, click the "OK" button.

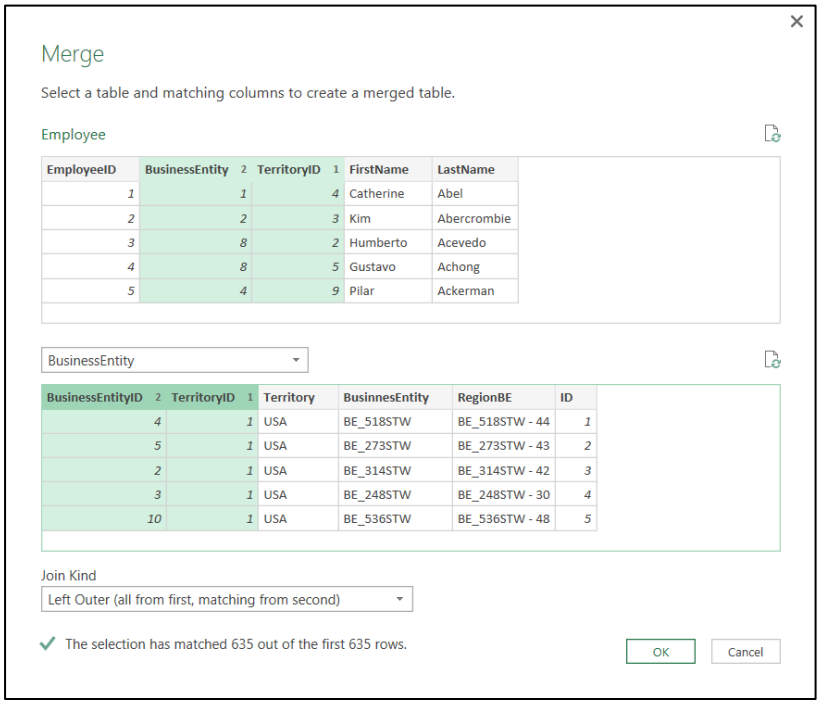


Figure 9. The numbers next to the column names represent the matching couples or pairs.

9. After clicking "OK", you'll then be redirected back to the Query Editor where you'll see that a new **step called "Merged Queries"** has been created in the "Applied Steps" section. A column called NewColumn has also been added your table, which contains table values all the way across. If you try expanding this by clicking on the double opposite arrows right next to the name of the column, you'll notice a bunch of fields that we can bring from the BusinessEntity table. Select only the RegionBE field and click the "OK" button

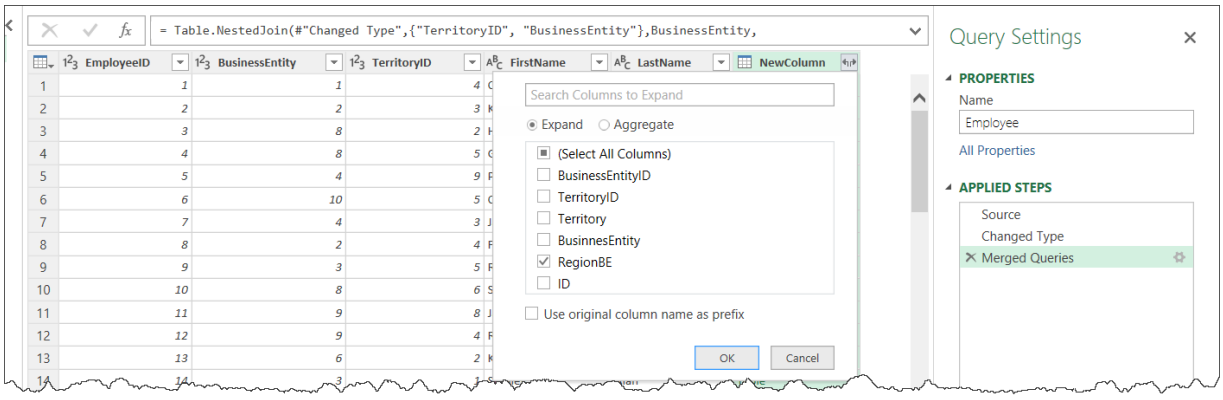


Figure 10. The newly created column can bring data from the Business Entity table.

10. Let's repeat the merging process again, but this time for the Sales table. Click on the Merge Queries button (found in the "Combine" section of the "Home" tab) and select the Sales query from the drop-down menu that appears in the "Merge" window. Next, you'll select the **EmployeeID column from both queries**. This is a simple merge, but if you needed to do a merge based on more than one column, like in the previous steps, you can absolutely do so.

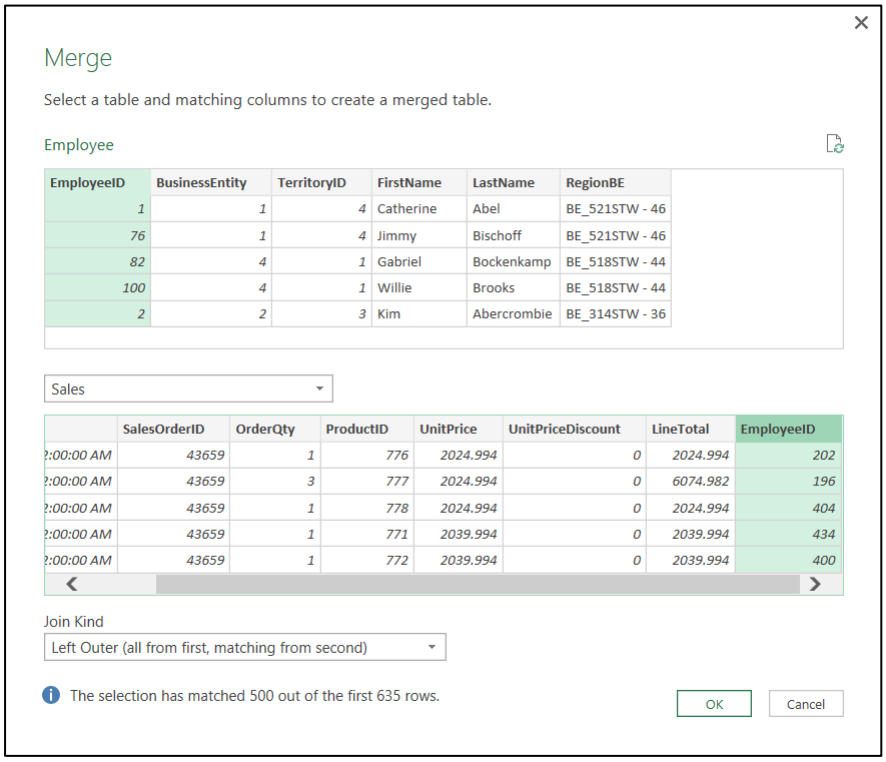


Figure 11. Merging the current query with the Sales query.

Pay close attention to the bottom section of the last figure. There's another drop-down box that has a label of "Join Kind". These are all the available options that you get:

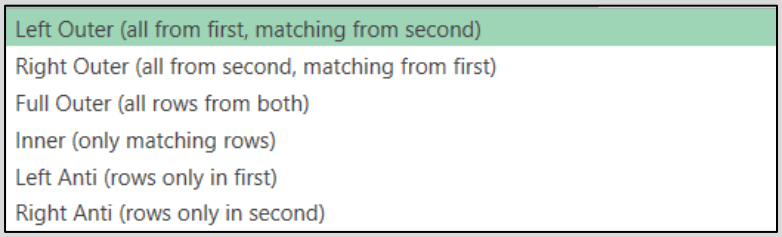


Figure 12. Whenever you see the word "first", this refers to the current query that you're in (the one that shows on top); "second" refers to the query on the bottom.

11. Based on the information shown in Figure 13 below, we should change the **"Join Kind" from "Left Outer"** (as shown above in Figure 12) **to "Right Outer"** so that we don't include the employees with no sales data. We can see that only 500 out of the 635 employees have sales data, so there's no reason for including those 135 that do not have data – we can just focus on the ones that do have sales. If you're in too deep and already clicked "OK" with "Left Outer" selected, you can go back to that step. In the "Applied Steps" section, click on the gear icon next to that step and it'll bring back that "Merge" window where you can modify its content.

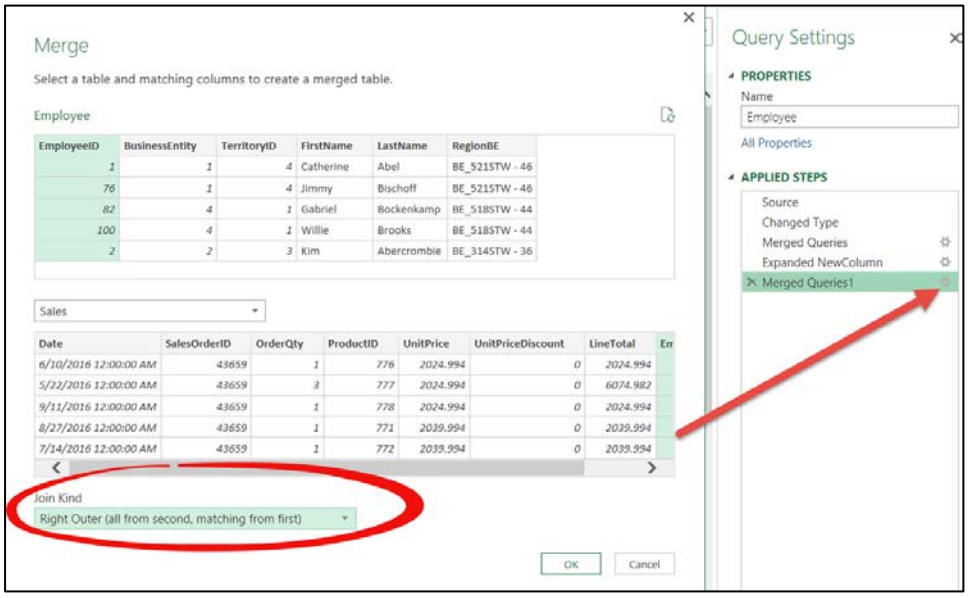


Figure 13. Click the gear icon next to a step name whenever you want to modify its behavior. Change the "Join Kind" to "Right Outer".

12. After clicking the "OK" button, you'll see that now your query only has 500 rows instead of 635. You'll also notice a column called NewColumn that you can expand. Click on the double opposite arrow icon next to the column name (NewColumn) and select the radio button labeled **"Aggregate".** Instead of importing data from the Sales table, you'll be aggregating (combining) the data into a nice new table with the totals for each employee. The "LineTotal" column in the Sales table contains the sales numbers, so we can simply check that column and select what operation we'd like to perform (Sum, Average, Median, Maximum, etc.). The OrderQuantity column can also be aggregated. This convenient function gathers ALL the rows from the Sales table for your specific employee, not just one row like a traditional VLOOKUP or INDEX/MATCH.

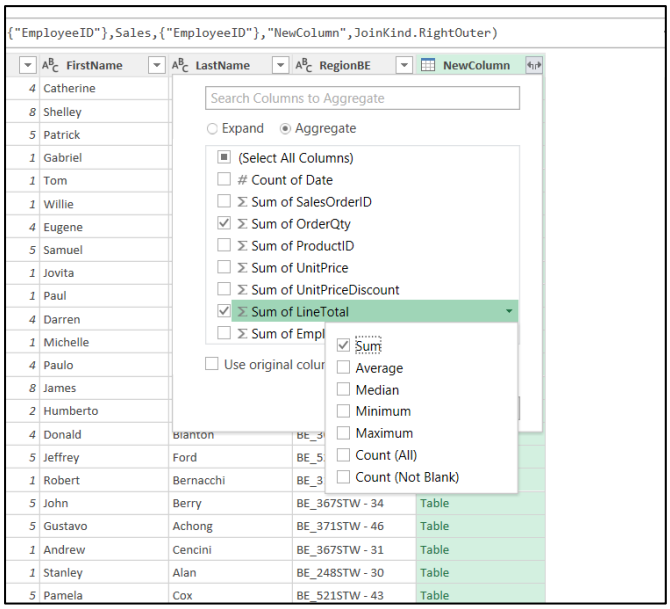


Figure 14. Click the double opposite arrow icon next to the column name. You can either expand columns or aggregate columns from other tables.

13. We need to find out the top performers by the RegionBE field. To do that, we can simply sort this table by the RegionBE field and then sort again in descending order by the new Sum of OrderQty that was created by the aggregation.

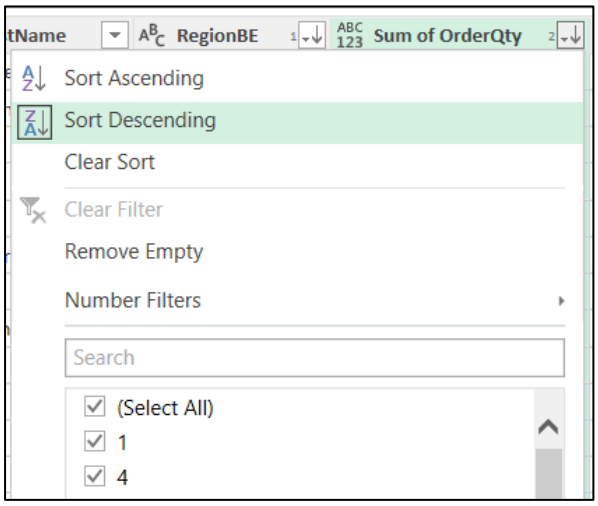


Figure 15. Just like you'd sort any Excel table, you can sort any table in Power Query. However, in Power Query the sorting order makes a difference as each sort level is preserved.

14. Once you finish the sort, you'll notice that it's not the same as the sorting in an Excel table. When sort in an Excel table, the new sorting overrides any previous sorting, but in Power Query it works in a hierarchal way where the **previous sorting is preserved.**

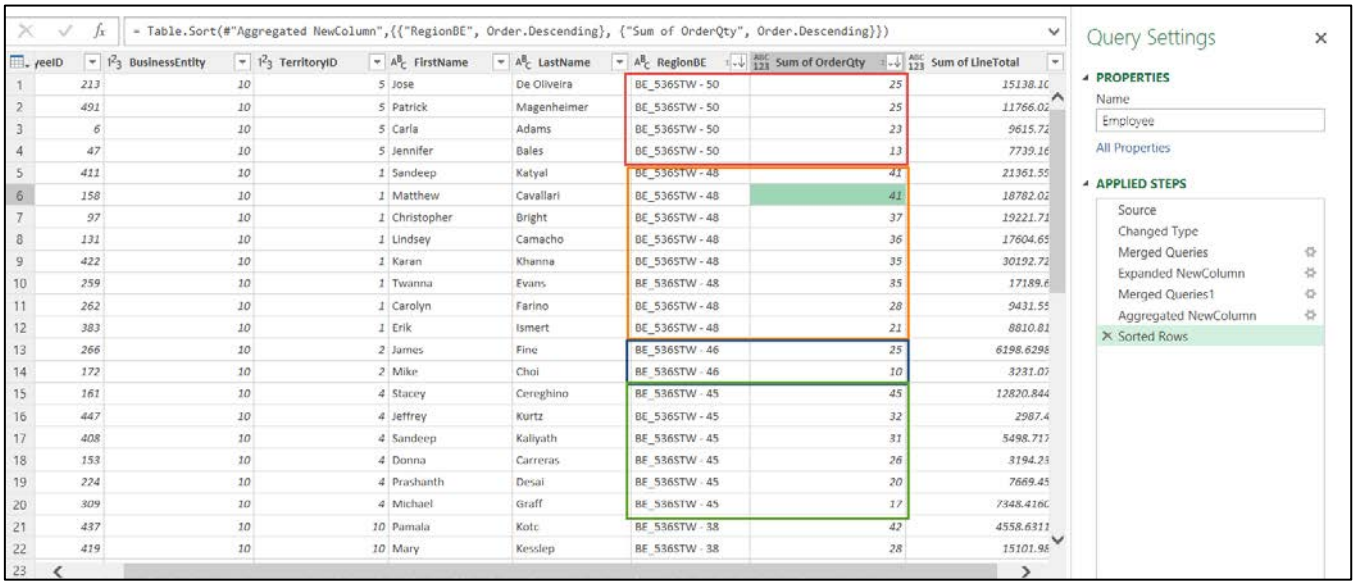


Figure 16. Sorting works in a hierarchal manner. Numbers next to the column names represent the level of the hierarchy.

15. Now that our query is ready, click the "Close & Load" icon. Then, in the Power Query ribbon, select the "Show Pane" icon which will display all our queries in the workbook. Right click the Employee query and select the "Load To" option which will bring a window

that we've seen before. In this "Load To" window, select the "Table" option and load this query to a new worksheet.

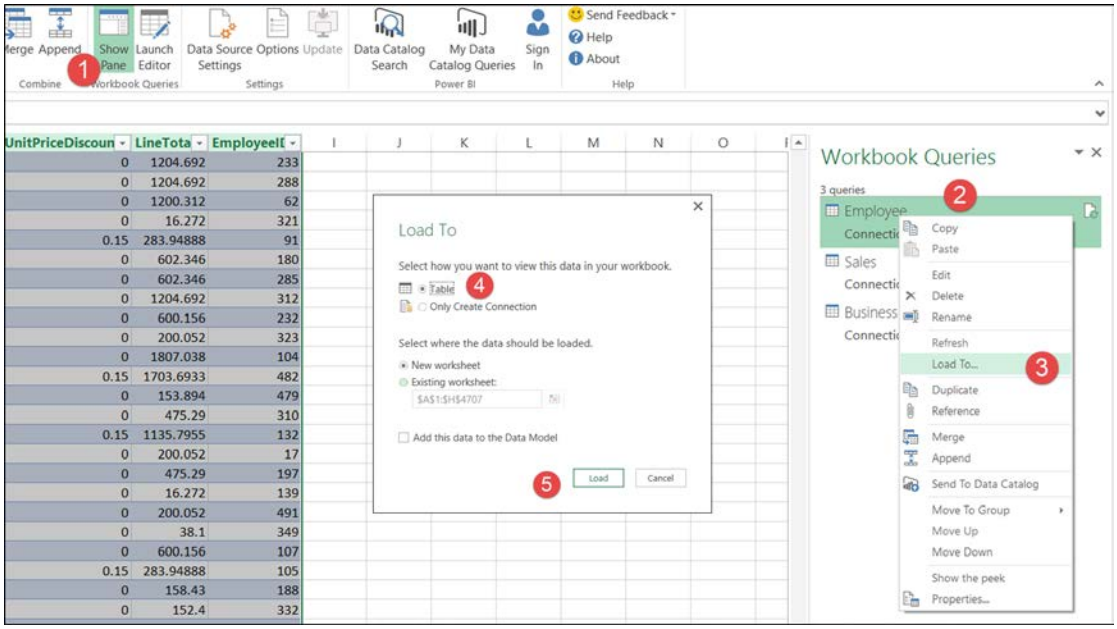
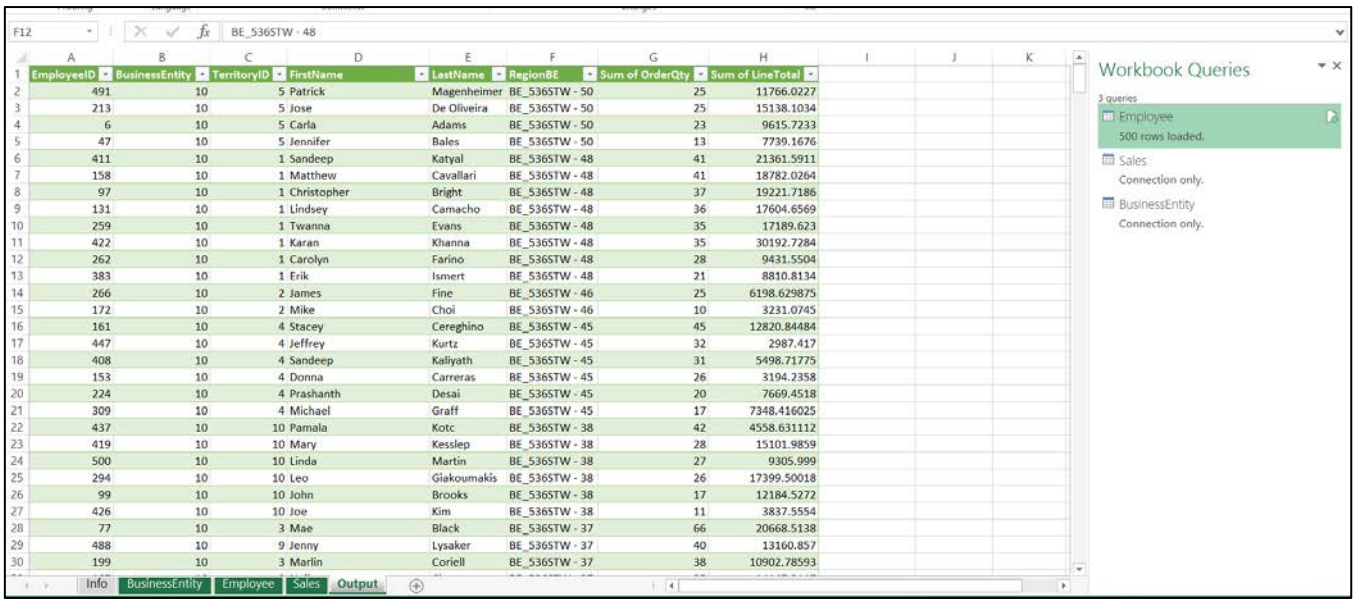


Figure 17. Outlining the steps to modify the "Load To" behavior of a query.

16. The result will be a new table inside a new sheet of the current Excel file.



EmployeeID	BusinessEntity	TerritoryID	FirstName	LastName	RegionBE	Sum of OrderQty	Sum of LineTotal
491	10	5	Patrick	Magenheimer	BE_536STW - 50	25	11766.0227
213	10	5	Jose	De Oliveira	BE_536STW - 50	25	15138.1034
6	10	5	Carla	Adams	BE_536STW - 50	23	9615.7233
47	10	5	Jennifer	Bales	BE_536STW - 50	13	7739.1676
411	10	1	Sandeep	Katyal	BE_536STW - 48	41	21361.5911
158	10	1	Matthew	Cavallari	BE_536STW - 48	41	18782.0264
97	10	1	Christopher	Bright	BE_536STW - 48	37	19221.7186
131	10	1	Lindsey	Camacho	BE_536STW - 48	36	17604.6569
259	10	1	Twanna	Evans	BE_536STW - 48	35	17189.623
422	10	1	Karan	Khanna	BE_536STW - 48	35	30192.7284
262	10	1	Carolyn	Farino	BE_536STW - 48	28	9431.5504
383	10	1	Erik	Ismert	BE_536STW - 48	21	8810.8134
266	10	2	James	Fine	BE_536STW - 46	25	6198.629875
172	10	2	Mike	Choi	BE_536STW - 46	10	3231.0745
161	10	4	Stacey	Cereghino	BE_536STW - 45	45	12820.84484
447	10	4	Jeffrey	Kurtz	BE_536STW - 45	32	2987.417
408	10	4	Sandeep	Kaliyath	BE_536STW - 45	31	5498.71775
153	10	4	Donna	Carreras	BE_536STW - 45	26	3194.2358
224	10	4	Prashanth	Desai	BE_536STW - 45	20	7669.4518
309	10	4	Michael	Graff	BE_536STW - 45	17	7348.416025
437	10	10	Pamala	Kotc	BE_536STW - 38	42	4558.631112
419	10	10	Mary	Kesslep	BE_536STW - 38	28	15101.9859
500	10	10	Linda	Martin	BE_536STW - 38	27	9305.999
294	10	10	Leo	Giakoumakis	BE_536STW - 38	26	17399.50018
99	10	10	John	Brooks	BE_536STW - 38	17	12184.5272
426	10	10	Joe	Kim	BE_536STW - 38	11	3837.5554
77	10	3	Mae	Black	BE_536STW - 37	66	20668.5138
488	10	9	Jenny	Lysaker	BE_536STW - 37	40	13160.857
199	10	3	Marlin	Coriell	BE_536STW - 37	38	10902.78593

Figure 18. The result of our hard work! You can refresh it whenever you want and it'll pick up new data from any of your tables.

Case Summary: Merge Tables

Topics covered

- Getting data from a table in current Excel workbook
- Merging tables by one or many columns
- Left and right outer joins
- Expanding and aggregating column table values
    - Both are triggered through the same double opposite arrow icon
- The gear icon on each step
    - Extremely useful feature whenever you need to modify the settings of any step that has this gear icon
- Sorting in Power Query
    - Different behavior than the sorting in regular Excel
- Basic Power Query ribbon usage
    - How to navigate to the Query Editor and the Queries Pane

Recommendations

- Reference the Employee query instead of working on top of it
    - Instead of working directly on the Employee query, we recommend referencing that query and creating a new one that would do everything that we covered on this chapter.
- Define your Data Types and do not let 'Any' data types pass by
    - It is important to know that defining Data types in your queries is crucial when loading to Power Pivot. Do not let the 'Any' data type get away. A mismatch in data types could potentially create errors in your query for certain operations.

Personal comments from the authors

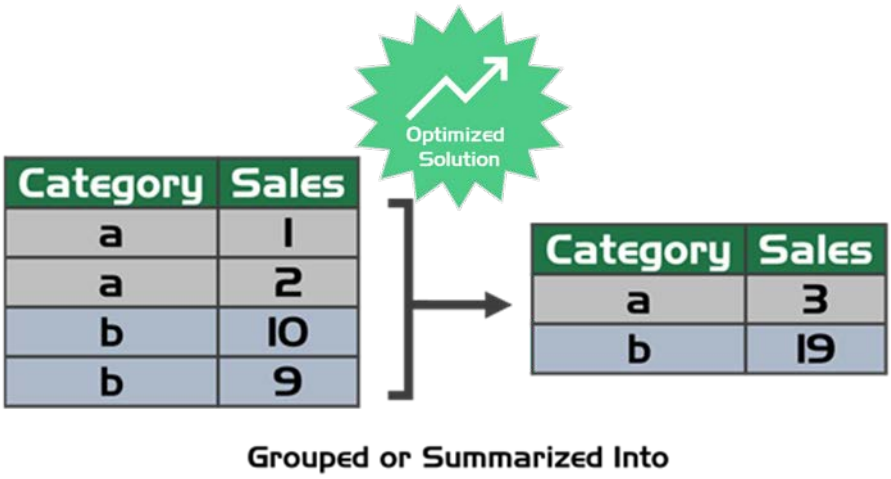
- Data Modelling factors
    - Power Query is commonly used to create better tables for your Data Model in either Power Pivot or the Power BI Desktop. Understanding the granularity of your data in each table is extremely important when trying merge or unmerge tables.
- Power Query as a tool for quick reports
    - We believe that Power Query is not just a tool to feed data to Power Pivot or the Power BI Desktop, but also a tool to create reports or even simulations that can be loaded to a new table in a workbook. Power Query is much more than what meets the eye and can be an immensely valuable tool on its own for a Data Wizard.

Master Power Query and become a Data Wizard

Take our Power Query Workshop

Chapter 2: Grouping Data

Difficulty: Intermediate



Briefing: Grouping Multiple Rows in a Table

We have one table:

- **Sales –** daily information of sales by product and ChannelName

Our Goal

- Create a Total Sales by Channel column
- Create a new column with a list of all the products sold on that day (separated by commas)
- Create another column containing the product with the highest sales
- Create a final column with the percentage share of total sales for the top selling product

Source Table

Date	ProductName	ChannelName	Amount
3/6/2015	Cap Sleeve	Reseller	164.7
3/6/2015	Long Sleeve	Online	180.43
3/10/2015	Short Sleeve	Online	229.92
3/10/2015	Sleeveless	Online	118.4

Our Desired Result

Date	ChannelName	Total Sales	Total Products	Products Sold	Top Product	Top Product Participation
3/6/2015	Reseller	164.7	1	Cap Sleeve	Cap Sleeve	100%
3/6/2015	Online	180.43	1	Long Sleeve	Long Sleeve	100%
3/10/2015	Online	348.32	2	Short Sleeve, Sleeveless	Short Sleeve	66%

Figure 19. Using Power Query, we'll reach our desired result from the information in our source table.

Be sure to open the workbook "Grouping – Start.xlsx" so you can follow along.

The How-to Guide: Grouping Data

With the **"Group by – Start.xlsx"** file open:

1. Go to the Sales worksheet, where you'll find a table. Select any cell inside that table, click the Power Query tab and choose "From Table".

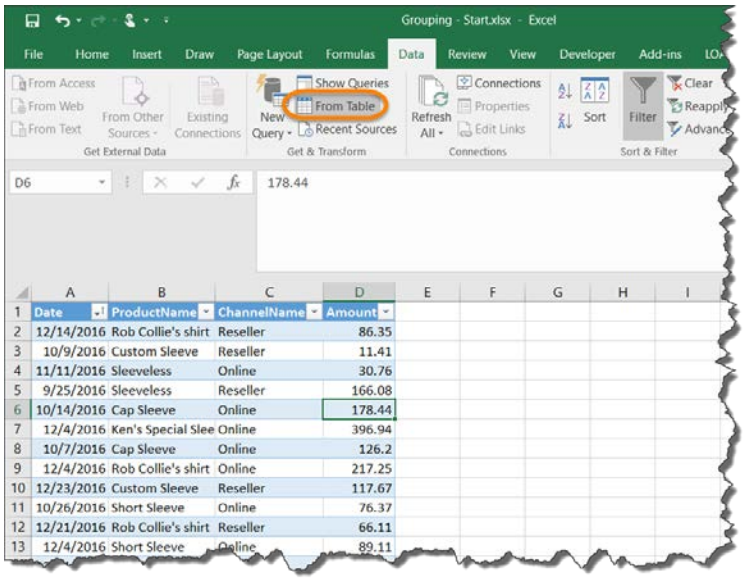


Figure 20. "From Table" functionality in Power Query using Excel 2016.

2. First, we need to group the rows in our table using some criteria. Click the "Group By" button and set it up using the criteria in Figure 21:

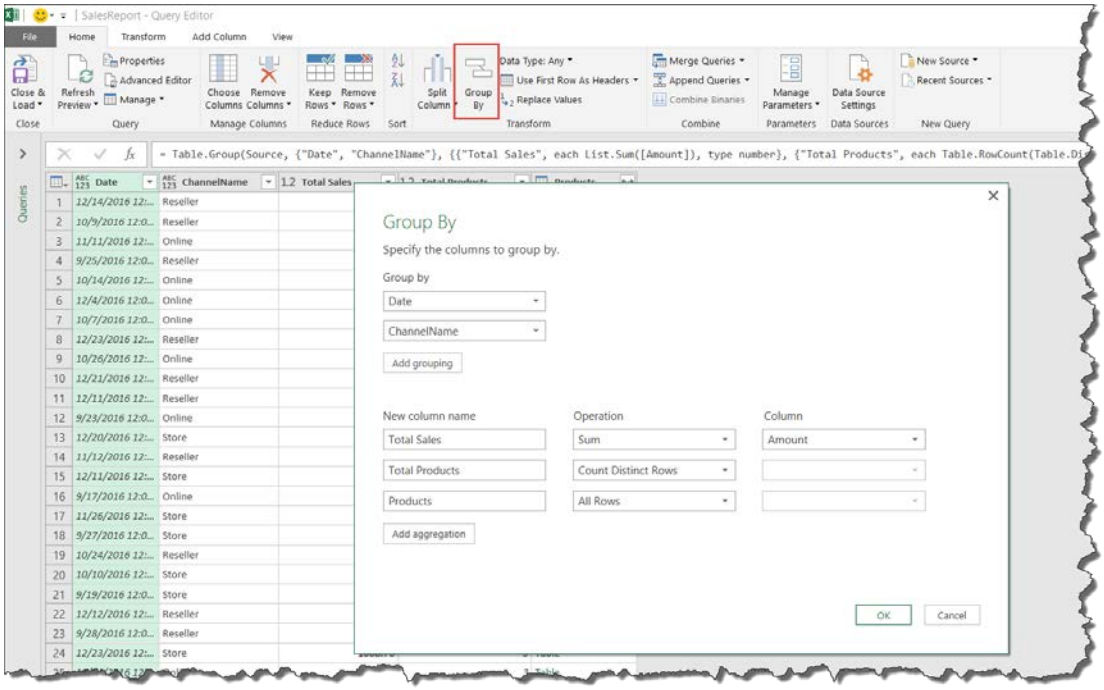


Figure 21. Using the "Group By" feature in Power Query.

3. We should now have a table with fewer rows because all the data has been grouped by Date and ChannelName. The previous step also created three new columns: a Total Sales column (containing the sum of all the sales by channel by day), a Total Products column (yielding a count of distinct products by channel by day) and a Products column. This final column contains a value of "Table" because the "All Rows" operation adds all the rows that meet the grouping criteria into a single table.

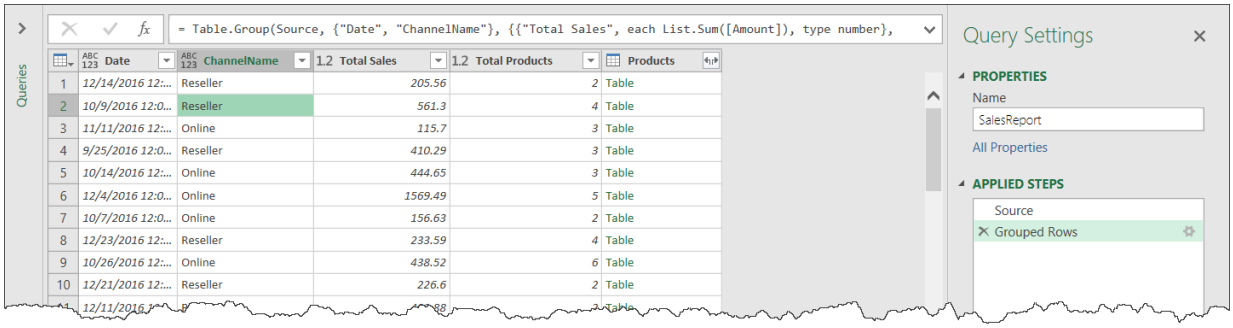


Figure 22. The result of our previous "Group By" operation.

4. Before we proceed, whenever you pull a date into Power Query, we highly recommend that you specifically define the data type for the date column. If you don't, the data could be treated as type "any", which means that it could land as either text or a value in your output instead of a date. To do this, simply **select the column** header, go to the **Transform** section of the Home tab, and change the **Data Type** to **Date**.

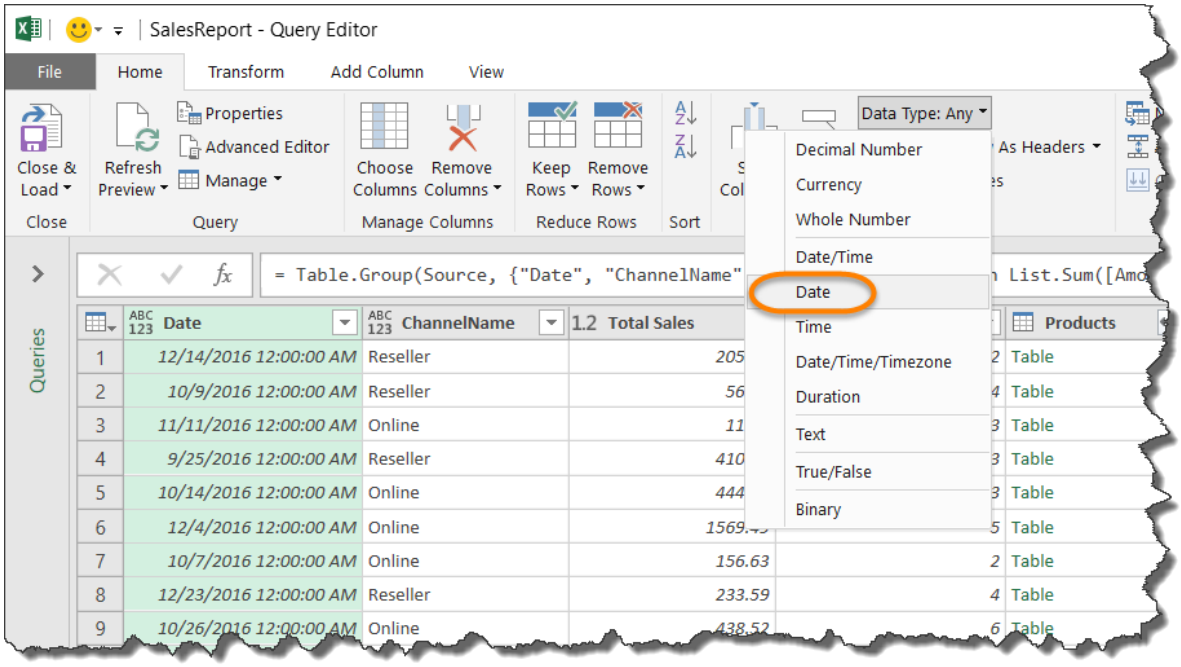


Figure 23. Defining the data type is a best practice.

5. Now we're ready to **create a new column with a list of all the products sold** on that day (separated by commas) using the "Products" column that was created with the "All Rows" operation in step 2. To achieve this:

a. From the tables in the Product column, we need to extract the column that has all the text strings that you want to combine into a single cell using the **Table.SelectColumns** function in a new **Custom Column.** We create a custom column by going to the **Add Column tab** and selecting the "Custom Column" icon. You'll need to manually enter the formula shown in Figure 24 to the "Add Custom Column" window.

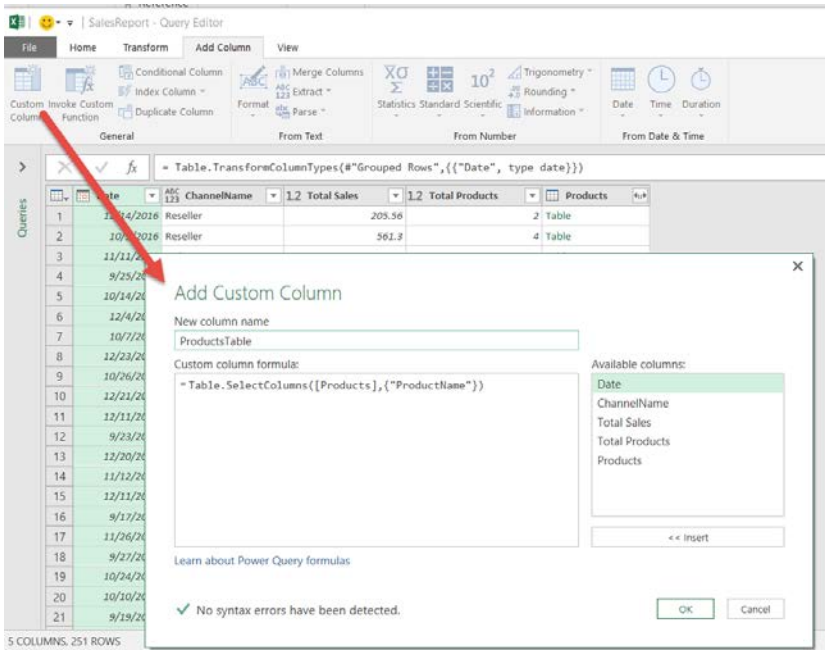


Figure 24. Creating a new custom column with a table that will only have one column, the ProductName column.

b. Now we need to create another column to transpose the ProductsTable table values so you only get one row of data and as many columns as needed using the **Table.Transpose** function. As before, create a new custom column and enter the formula shown in Figure 25.

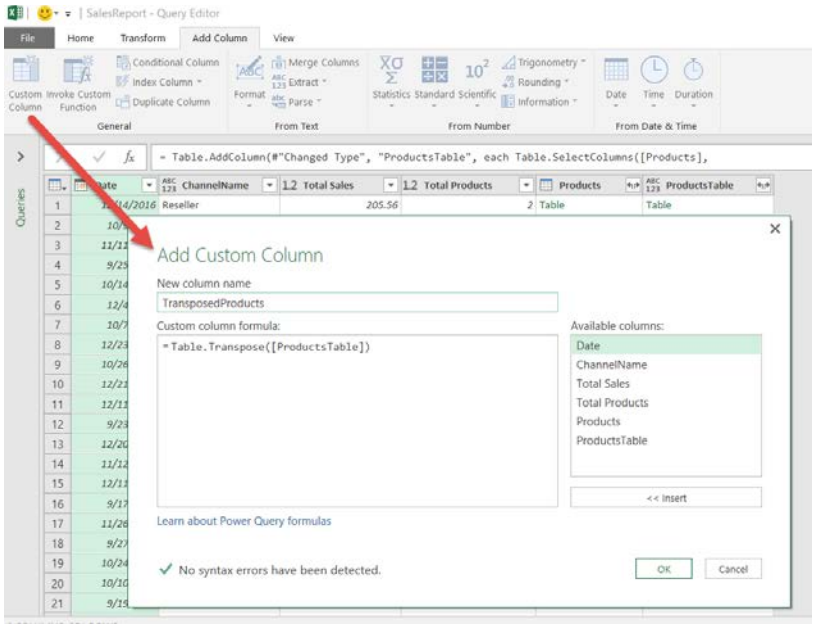


Figure 25. Creating a new custom column with a table that will only have one row and separate columns for each product.

c. Before proceeding, we will clean up our current table and delete the ProductsTable column as it is no longer needed. We can do so by simply right clicking on that column and selecting the option to **"Remove"**.

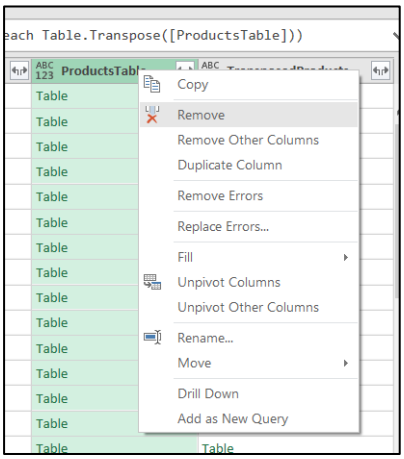


Figure 26. It is a best practice to remove any unnecessary columns or data values from columns.

d. We will now transform the TransposedProducts table, which contains just one row, into a list of just one element using the **Table.ToList** function. We also use the Combiner function so we can have more control over how each value in those columns is going to be combined or concatenated. Figure 27 below shows the formula to use:

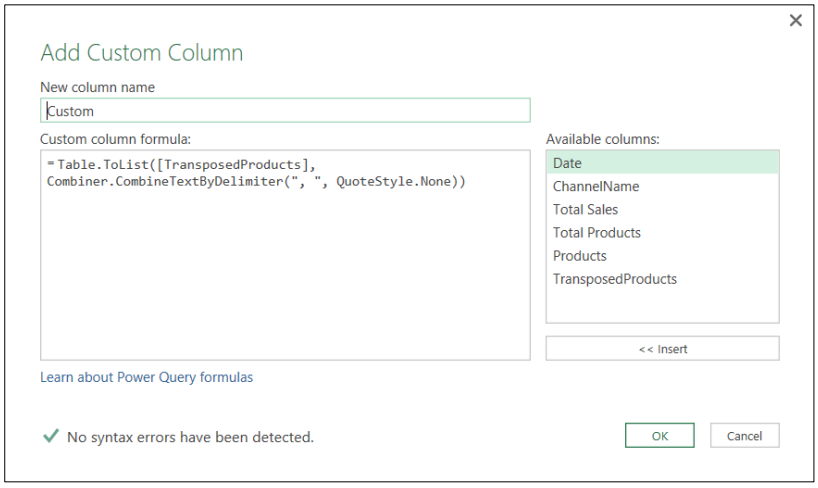


Figure 27. Using Table.ToList to concatenate multiple text strings from columns in a table.

The result of that last custom column is shown in Figure 28 next. Check the preview pane to see what the data will look like before you expand that column.

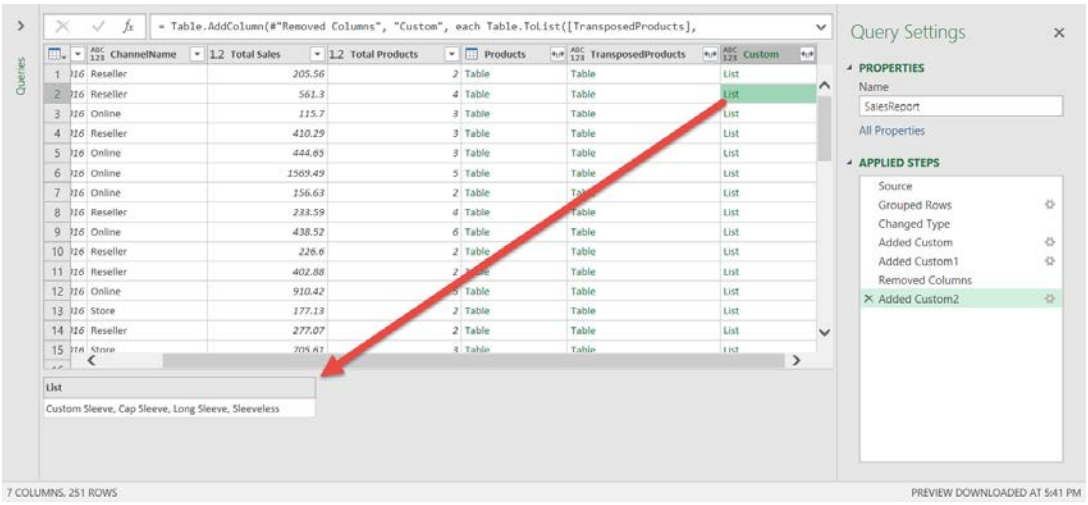


Figure 28. The Preview pane shows how Power Query concatenated multiple text strings in different columns into just one single cell.

6. Clean our current table again by removing the **TransposedProducts** column.
7. Click on the double opposite arrows next to the **Custom** column to expand it and select the values to include in our table, as shown in Figure 30 below.
8. Rename the **Custom** column as **"Products Sold"** by either double clicking on the column name or by right clicking on the column name and select the **"Rename"** option.
9. Now we want to add columns for the top selling product and amount sold of that top product. First, we need to extract the record with the highest sales from the tables found in the **Products** column. Since the values in that column are table values, we can use the **Table.Max** function. This function finds the maximum value in a column and extract the record (basically the row) where that value is located. Figure 29 below shows the formula to use:

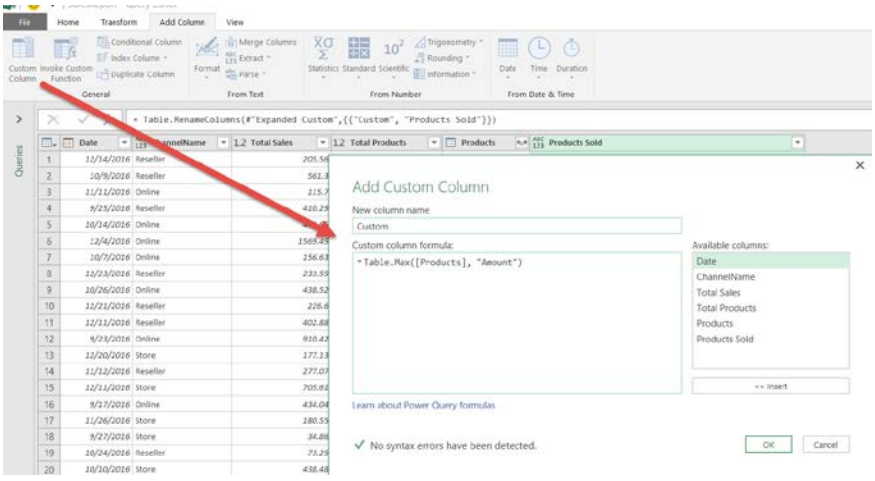


Figure 29. Creating a new custom column using a table function.

10. We can now expand the values from that newly created column by clicking the opposite double arrows icon next to the column name. Select the ProductName and Amount fields to extract the name of the top selling product and the highest sales value.

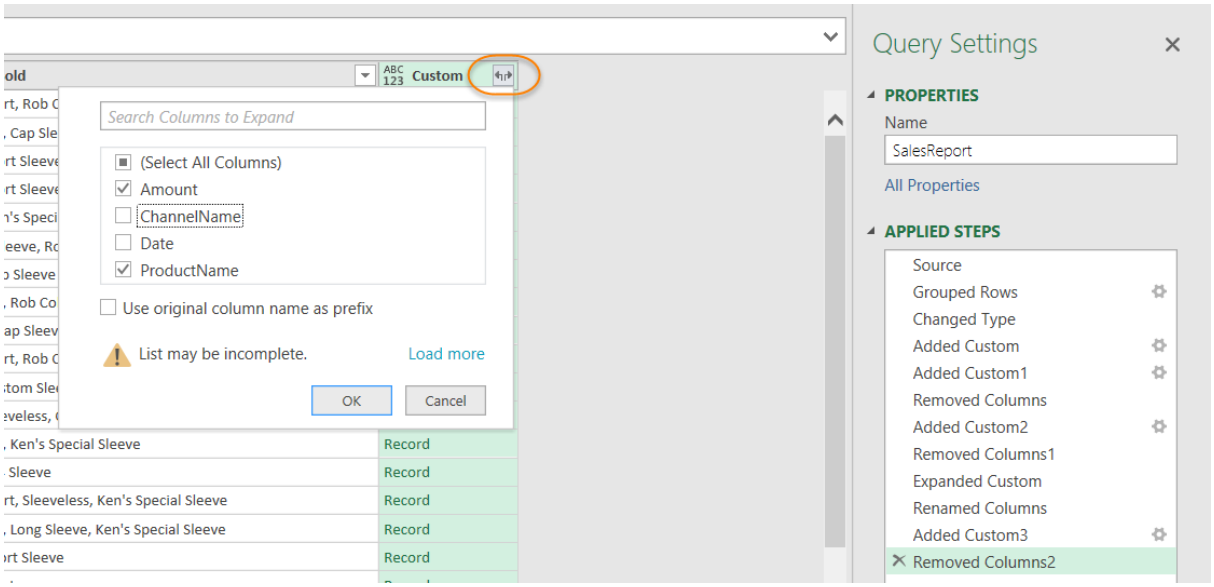


Figure 30. Expanding the custom column to get the Amount and ProductName fields for top selling product.

11. Rename the two new columns as **Top Product** and **Top Product Amount.**

12. Create a new column that will calculate the **Top Product Participation** by simply dividing the **Top Product Amount** by the **Total Sales**

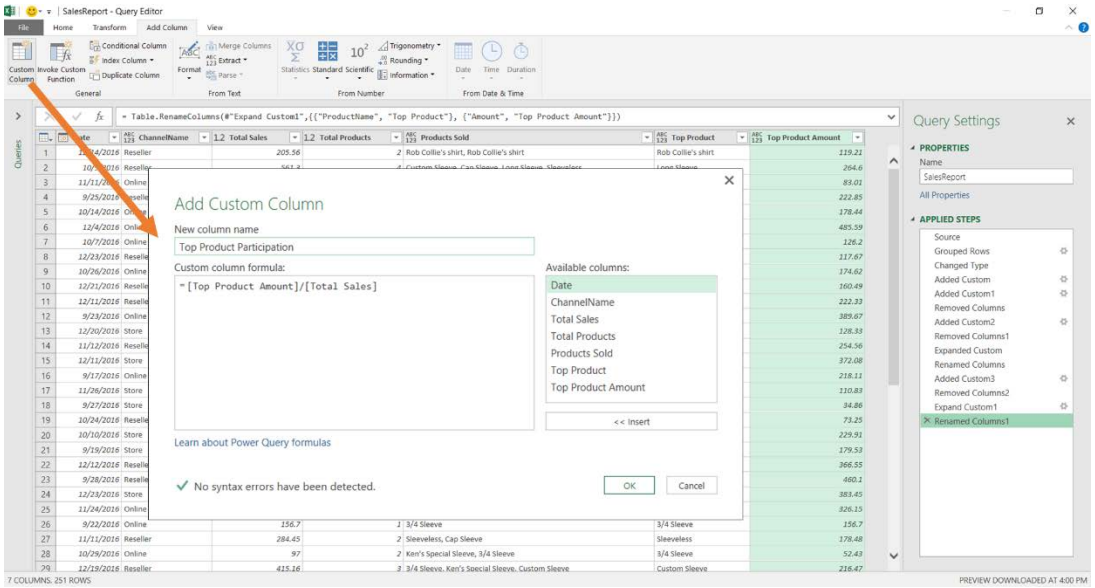


Figure 31. Creating a new Top Product Participation custom column

13. Clean up the table by deleting the **Top Product Amount** column that we no longer need.

14. Round the values in **Top Product Participation** column to two places by going to the **Transform tab** in the "Query Editor" window and select **Rounding** from the "Number Column" section. Enter 2 decimals and click OK.

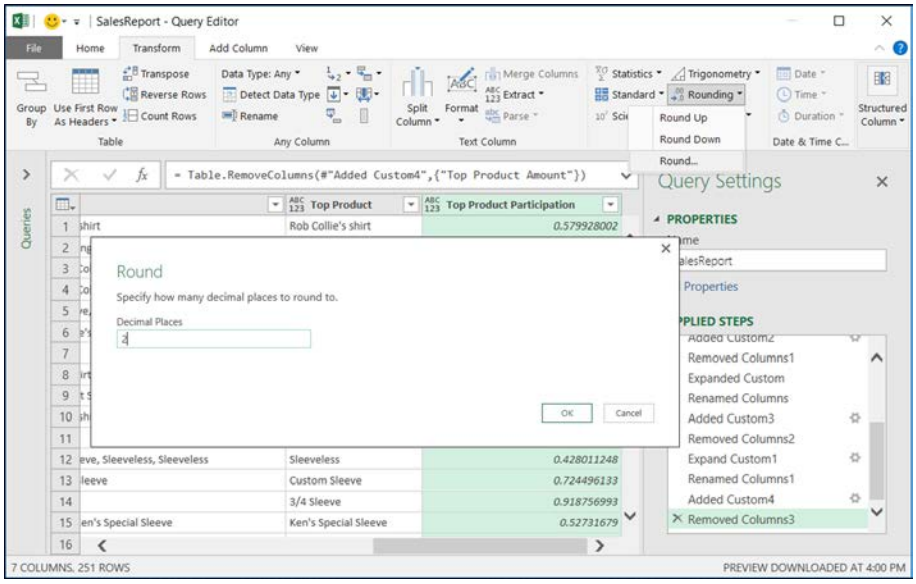


Figure 32. Rounding the Top Product Participation column to 2 decimal places.

15. Next, sort this data by Date to make it easier to read.

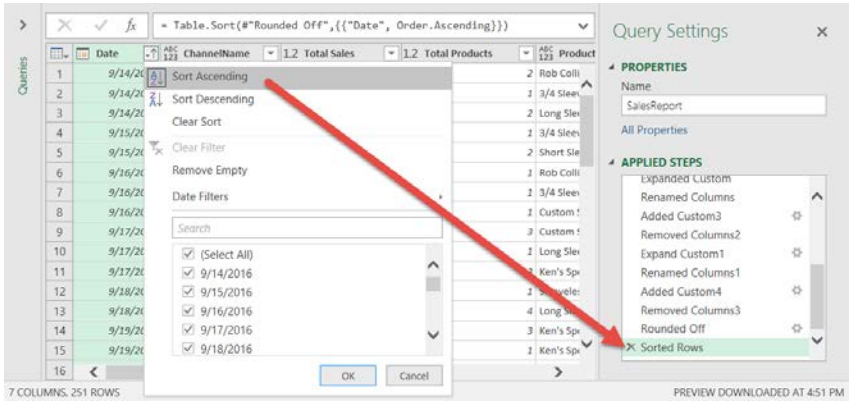


Figure 33. Sorting by the Date column.

16. Define the data types for every column to ensure they have the correct data type.

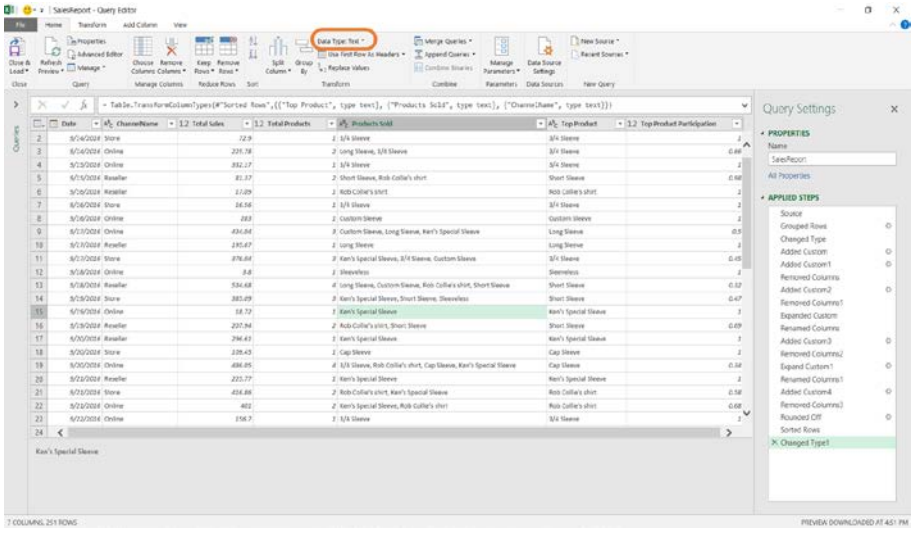


Figure 34. Defining the column data types.

17. Finally, click on “Close & Load” so we can load this data as a new table in Excel.

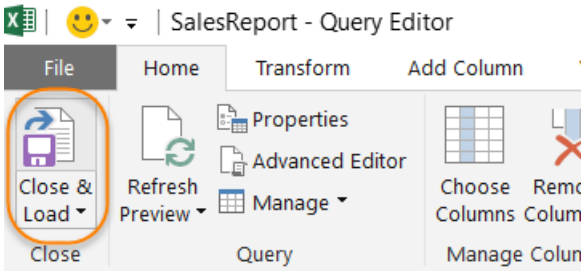


Figure 35. Selecting the “Close & Load” button.

	Date	ChannelName	Total Sales	Total Products	Products Sold	Top Product	Top Product Participation
2	9/14/2016	Store	72.9	1	3/4 Sleeve	3/4 Sleeve	1
3	9/14/2016	Reseller	161.73	2	Rob Collie's shirt, Sleeveless	Sleeveless	0.59
4	9/14/2016	Online	225.78	2	Long Sleeve, 3/4 Sleeve	3/4 Sleeve	0.66
5	9/15/2016	Online	332.17	1	3/4 Sleeve	3/4 Sleeve	1
6	9/15/2016	Reseller	81.37	2	Short Sleeve, Rob Collie's shirt	Short Sleeve	0.98
7	9/16/2016	Online	283	1	Custom Sleeve	Custom Sleeve	1
8	9/16/2016	Store	16.56	1	3/4 Sleeve	3/4 Sleeve	1
9	9/16/2016	Reseller	17.09	1	Rob Collie's shirt	Rob Collie's shirt	1
10	9/17/2016	Reseller	195.67	1	Long Sleeve	Long Sleeve	1
11	9/17/2016	Store	376.04	3	Ken's Special Sleeve, 3/4 Sleeve, Custom Sleeve	3/4 Sleeve	0.45
12	9/17/2016	Online	434.04	3	Custom Sleeve, Long Sleeve, Ken's Special Sleeve	Long Sleeve	0.5
13	9/18/2016	Reseller	534.68	4	Long Sleeve, Custom Sleeve, Rob Collie's shirt, Short Sleeve	Short Sleeve	0.32
14	9/18/2016	Online	3.8	1	Sleeveless	Sleeveless	1
15	9/19/2016	Reseller	207.94	2	Rob Collie's shirt, Short Sleeve	Short Sleeve	0.69
16	9/19/2016	Online	18.72	1	Ken's Special Sleeve	Ken's Special Sleeve	1
17	9/19/2016	Store	385.09	3	Ken's Special Sleeve, Short Sleeve, Sleeveless	Short Sleeve	0.47
18	9/20/2016	Online	486.05	4	3/4 Sleeve, Rob Collie's shirt, Cap Sleeve, Ken's Special Sleeve	Cap Sleeve	0.34
19	9/20/2016	Reseller	296.61	1	Ken's Special Sleeve	Ken's Special Sleeve	1
20	9/20/2016	Store	109.45	1	Cap Sleeve	Cap Sleeve	1
21	9/21/2016	Reseller	225.77	1	Ken's Special Sleeve	Ken's Special Sleeve	1
22	9/21/2016	Online	401	2	Ken's Special Sleeve, Rob Collie's shirt	Rob Collie's shirt	0.68
23	9/21/2016	Store	416.86	2	Rob Collie's shirt, Ken's Special Sleeve	Rob Collie's shirt	0.58
24	9/22/2016	Online	156.7	1	3/4 Sleeve	3/4 Sleeve	1
25	9/22/2016	Store	185.96	3	Rob Collie's shirt, Sleeveless, Rob Collie's shirt	Rob Collie's shirt	0.88

Figure 36. The result of our hard work! You can refresh the table whenever you want by right clicking the query or the table and selecting the Refresh option.

Case Summary: Grouping Data

Topics covered

- Extracting data from a table in current Excel workbook
- Using the Group By feature in Power Query for aggregations and other operations such as the **"All Rows"**
- Combine text strings from a column into a single cell
- Working with table values in a column using table functions
- Basic Power Query ribbon usage
    - How to navigate to the Query Editor and the Queries Pane

Recommendations

- Simplify your code
    - You could save yourself four sub-steps in the step 5 by simply applying all the code together into one single line of code like this:

```
Table.ToList(Table.Transpose(Table.SelectColumns([Products],{"ProductName"})),

Combiner.CombineTextByDelimiter(", ",QuoteStyle.None))
```

- Applying table and list functions for nested tables and lists
    - In Power Query, you might find that there are table values inside a table and perhaps inside that sub- table there is another table or even a list. Knowing your data and how to apply table or list functions against them would be extremely beneficial to you.

Personal comments from the authors

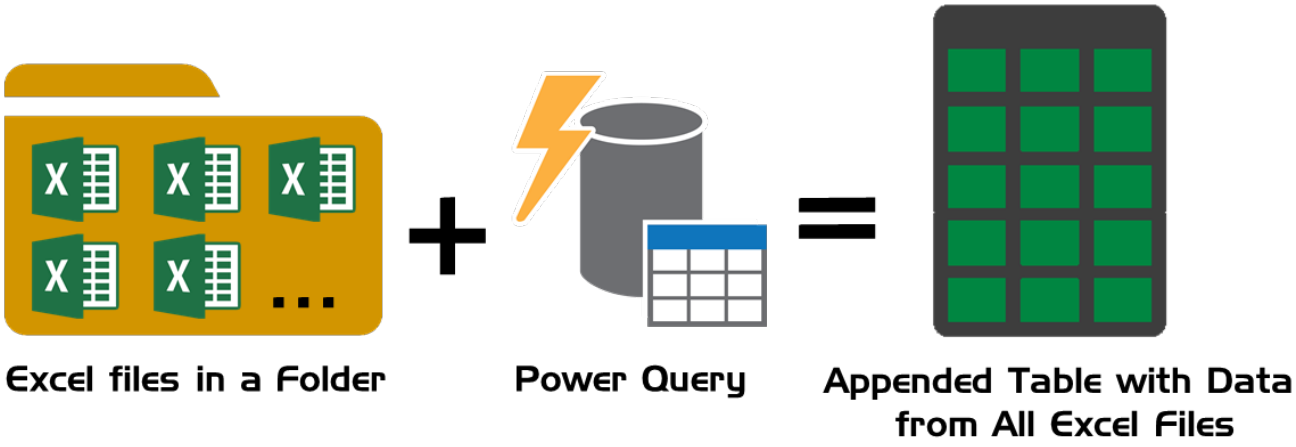
- Power Query as a tool for quick reports
    - Just like in the first scenario, we believe that Power Query is not just a tool to feed data to Power Pivot or the Power BI Desktop, but also a tool to create reports or even simulations that can be loaded to a new table in a workbook.
- Nested table and lists scenarios
    - Sometimes we use the "Group By" functionality to achieve a nested table scenario where we then apply some table or list functions against the newly grouped rows. Though more advanced, these are some of the most powerful scenarios that you can unleash with Power Query.

Master Power Query and become a Data Wizard

Take our Power Query Workshop

Chapter 3: Combining Files from a Folder

Difficulty: Intermediate



Briefing: Combining Files from a Folder

If you ever needed to append, consolidate or combine data, you might have realized that it is a tedious and not very user-friendly task.

In this chapter, we'll showcase the easiest way to connect to the files from a specific folder and combine them all together.

That's right, you can combine data from all your Excel files together. No limits on the size of the file or how many files you'd like to combine – it'll simply work!

Our Goal

We have six Excel workbooks, one file for each month running from January to June. In each file, we only have one worksheet with the appropriate data for the corresponding month. We would like to:

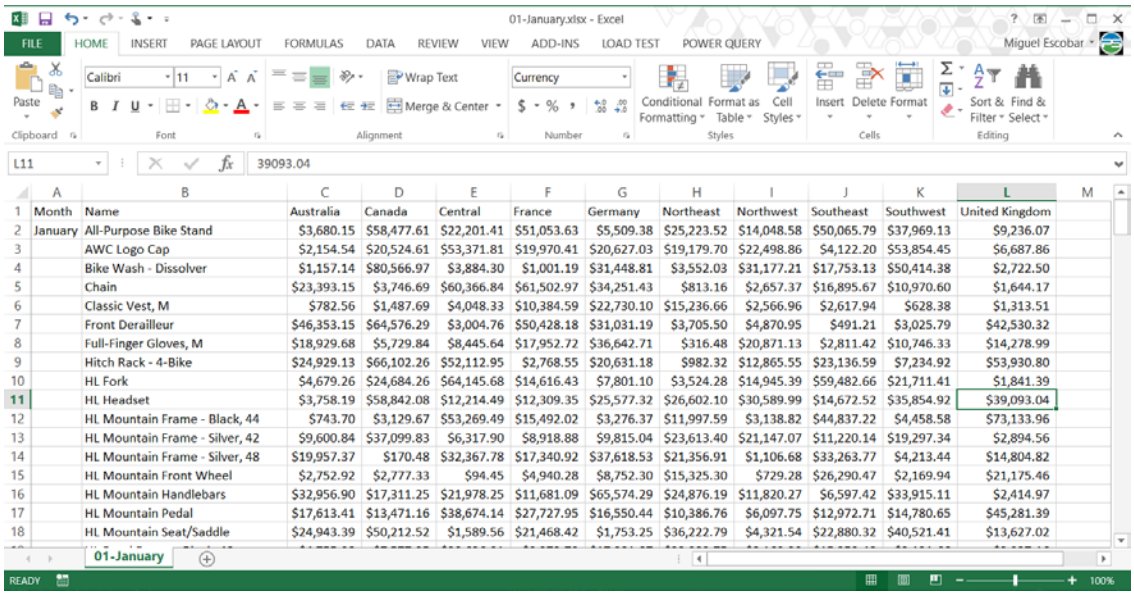
- Combine all the files into one tall table
- Do some needed transformation before consolidating the files

You can open a completely new Excel workbook or a Power BI Desktop file to follow along.

Please note that the figures for this section show a Power BI Desktop file.

The How-to Guide: Combining Files from a Folder

1. In the zip file containing all the files for this eBook, locate the folder named "Chapter 3 - Combing Files from Folder". In there is a sub-folder named **"Sales Reports"** that contains all the workbooks we would like to combine. Give each workbook a quick look so you can see what the data is like.



Month	Name	Australia	Canada	Central	France	Germany	Northeast	Northwest	Southeast	Southwest	United Kingdom
January	All-Purpose Bike Stand	$3,680.15	$58,477.61	$22,201.41	$51,053.63	$5,509.38	$25,223.52	$14,048.58	$50,065.79	$37,969.13	$9,236.07
	AWC Logo Cap	$2,154.54	$20,524.61	$53,371.81	$19,970.41	$20,627.03	$19,179.70	$22,498.86	$4,122.20	$53,854.45	$6,687.86
	Bike Wash - Dissolver	$1,157.14	$80,566.97	$3,884.30	$1,001.19	$31,448.81	$3,552.03	$31,177.21	$17,753.13	$50,414.38	$2,722.50
	Chain	$23,393.15	$3,746.69	$60,366.84	$61,502.97	$34,251.43	$813.16	$2,657.37	$16,895.67	$10,970.60	$1,644.17
	Classic Vest, M	$782.56	$1,487.69	$4,048.33	$10,384.59	$22,730.10	$15,236.66	$2,566.96	$2,617.94	$628.38	$1,313.51
	Front Derailleur	$46,353.15	$64,576.29	$3,004.76	$50,428.18	$31,031.19	$3,705.50	$4,870.95	$491.21	$3,025.79	$42,530.32
	Full-Finger Gloves, M	$18,929.68	$5,729.84	$8,445.64	$17,952.72	$36,642.71	$316.48	$20,871.13	$2,811.42	$10,746.33	$14,278.99
	Hitch Rack - 4-Bike	$24,929.13	$66,102.26	$52,112.95	$2,768.55	$20,631.18	$982.32	$12,865.55	$23,136.59	$7,234.92	$53,930.80
	HL Fork	$4,679.26	$24,684.26	$64,145.68	$14,616.43	$7,801.10	$3,524.28	$14,945.39	$59,482.66	$21,711.41	$1,841.39
	HL Headset	$3,758.19	$58,842.08	$12,214.49	$12,309.35	$25,577.32	$26,602.10	$30,589.99	$14,672.52	$35,854.92	$39,093.04
	HL Mountain Frame - Black, 44	$743.70	$3,129.67	$53,269.49	$15,492.02	$3,276.37	$11,997.59	$3,138.82	$44,837.22	$4,458.58	$73,133.96
	HL Mountain Frame - Silver, 42	$9,600.84	$37,099.83	$6,317.90	$8,918.88	$9,815.04	$23,613.40	$21,147.07	$11,220.14	$19,297.34	$2,894.56
	HL Mountain Frame - Silver, 48	$19,957.37	$170.48	$32,367.78	$17,340.92	$37,618.53	$21,356.91	$1,106.68	$33,263.77	$4,213.44	$14,804.82
	HL Mountain Front Wheel	$2,752.92	$2,777.33	$94.45	$4,940.28	$8,752.30	$15,325.30	$729.28	$26,290.47	$2,169.94	$21,175.46
	HL Mountain Handlebars	$32,956.90	$17,311.25	$21,978.25	$11,681.09	$65,574.29	$24,876.19	$11,820.27	$6,597.42	$33,915.11	$2,414.97
	HL Mountain Pedal	$17,613.41	$13,471.16	$38,674.14	$27,727.95	$16,550.44	$10,386.76	$6,097.75	$12,972.71	$14,780.65	$45,281.39
	HL Mountain Seat/Saddle	$24,943.39	$50,212.52	$1,589.56	$21,468.42	$1,753.25	$36,222.79	$4,321.54	$22,880.32	$40,521.41	$13,627.02

Figure 37. A sample of one of the files found in the 'Sales Reports' folder.

2. Open a new workbook or a new Power BI Desktop report and create a query **from a folder**.

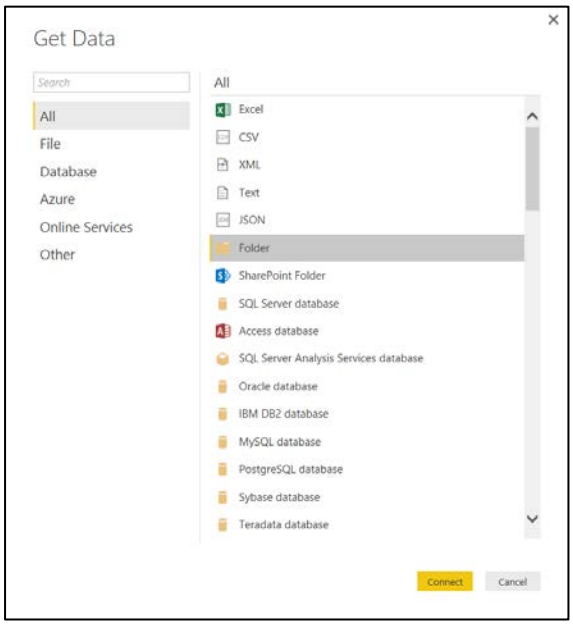


Figure 38. In Power Query for the Power BI desktop, click on the "Folder" option.

3. A new window will appear where you can **Browse** through your system and **find the folde**r where the Chapter 3 Excel workbooks are stored. Select the folder path and click "OK" (if prompted by another window after clicking "OK", please select "Edit").

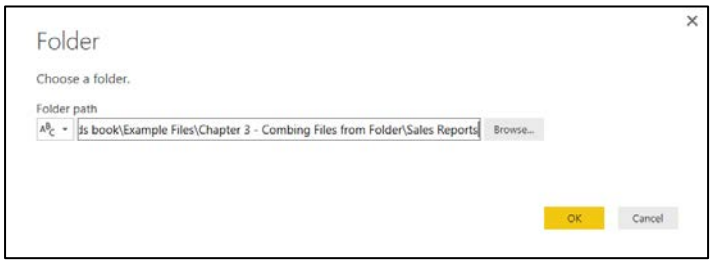


Figure 39. Find your folder path by clicking on the "Browse" button or by manually pasting your path.

4. Pay close attention to what happens once Power Query finishes loading your files from Folder. You'll notice that a list of all the files inside that folder will appear, so you should have six rows in that table – one for each file. But what happens if you see a seventh row like in Figure 40?

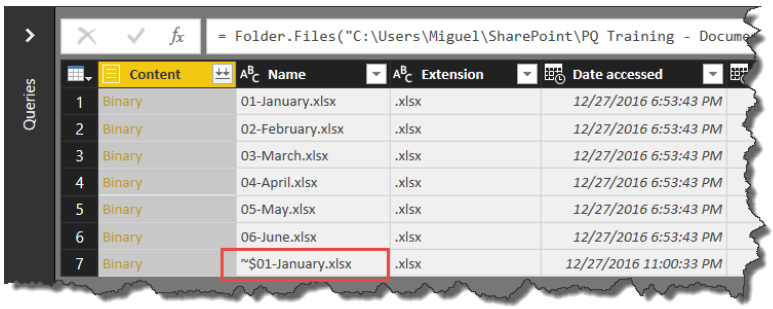


Figure 40. Files currently in use will appear with a prefix of "~$" added to the file name. In this image, we see that the "01-January.xlsx" file is in use so the "~$01-January.xlsx" file is also included in our table.

5. As a best practice, we recommend that you filter this table as a fail-safe so you only include the files that you actually want to combine. These filters include:

   a. **By the Extension column** – filter this column to only include files with the .xlsx or .xls format structure.

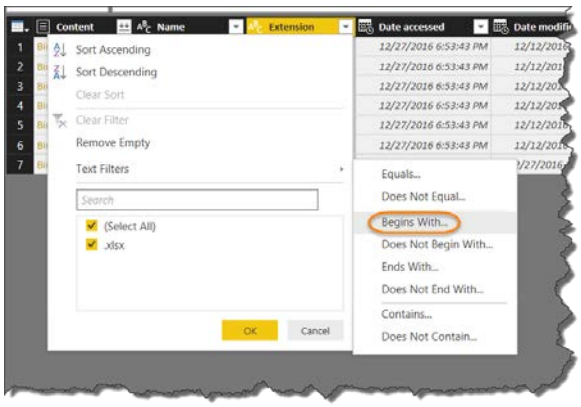


Figure 41. Filtering the Extension column by using the "Begins With" option.

b. **By the Name column** – filter this column to get rid of any files that start with the prefix "~$".

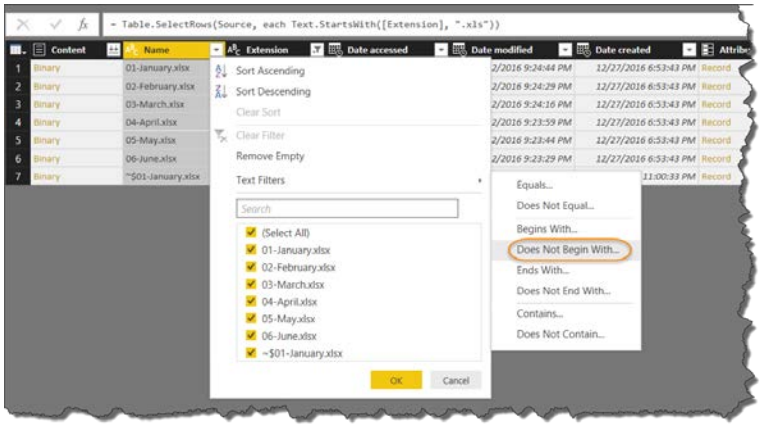


Figure 42. Filtering the Name column by using the "Does Not Begin With" option.

6. You can add more filters depending the scenario. Once you're done setting up all the filters for your table, click on the **"Combine Binaries" button** to combine all the files.

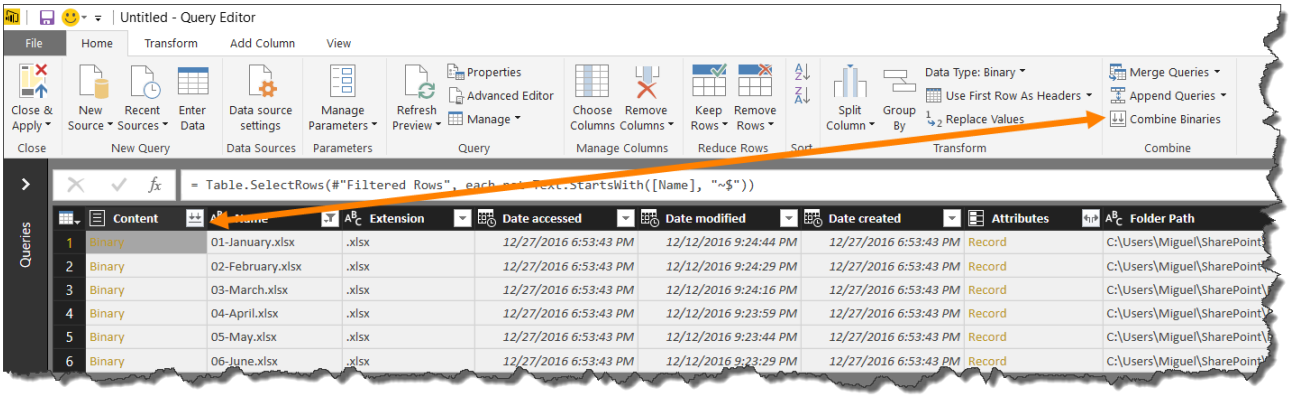


Figure 43. The "Combine Binaries" button can be found on the Content column and on the Home tab.

Note: The "Combine Binaries" function used to only work for flat files such as .txt and .csv files, but in November of 2016 the Power Query team released a new functionality that allows users to combine other types of files as well. At the time of publishing, only the Excel 2016 Insider Preview and the Power BI Desktop have this functionality, but you will be able to download an updated version of the Power Query add-in for Excel 2010 and Excel 2013 that will let you use this functionality later in 2017.

7. Once you click on the "Combine Binaries" button, a new window will pop up that allows you to specify how each of the files from your folder should be accessed.

   In plain English, Power Query is asking: **What objects do you want me to extract from each of the files?** Power Query even gives you a sample of the data that could be extracted and what objects you could have access to. It is important to note that this suggestion is based on a sample from one of the files from your folder and not all of them (based on the filters that you defined).

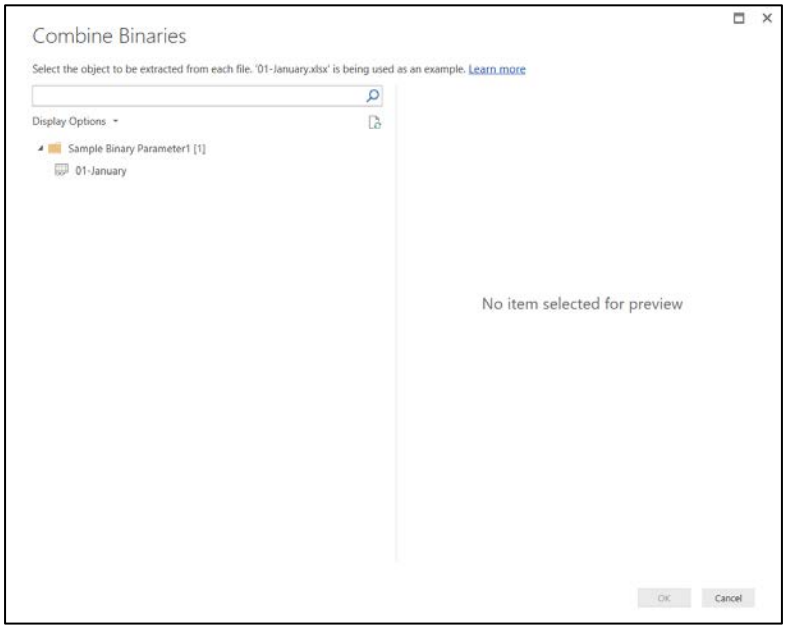


Figure 44. The new "Combine Binaries" window displays a sample based on the first record of the table, in this case, the data from the "01-January.xlsx" file.

Note: From this point forward, we have two options or methods:

a) **We can tell Power Query to give us access to all objects from all the files** or in plain English, give us all the data from each file, or

b) **We can tell Power Query to access only one specific object from each of the files** or in plain English, access the object (a table, range or sheet) with the name 'XYZ' from each of the files.

Each one of these methods has its pros and cons, and we'll look at each of them next.

Method I: Accessing All Objects from Files

The following set of steps will help you access all objects from each of your files and then consolidate them with a few clicks.

8. Select "Sample Binary Parameter1" as shown in Figure 44 above, and click "OK". This will create a few new steps in your query, as well as some new queries under the "Transform Binary from Sales Reports" group. These new queries act together and Power Query reads each of them to understand how to read each file from your folder.

Figure 45. The new "Combine Binaries" window displays a sample based on the first record of the first table, which happens to be "01-January.xlsx".

9. We now have two options:

- **a.** Continue working on the Sales Reports query, or
- **b.** Work in a completely new way by selecting "Transform Sample Binary" from the Sales Reports query.

In this case, we're going with option 'b' because it will work more easily for the task at hand. However, you can still use option 'a' to expand the tables and apply all the transformations within the Sales Reports query.

10. On the query pane to your left, select the "Transform Sample Binary" from the Sales Reports query and click on the space next to the "table" text highlighted in yellow. This action will allow you to **preview the data** from that table in the preview pane at the bottom of your screen. You will notice that the headers are not defined, and the best way to promote them would be to updated the formula in the formula bar by changing "null" value to be "true".

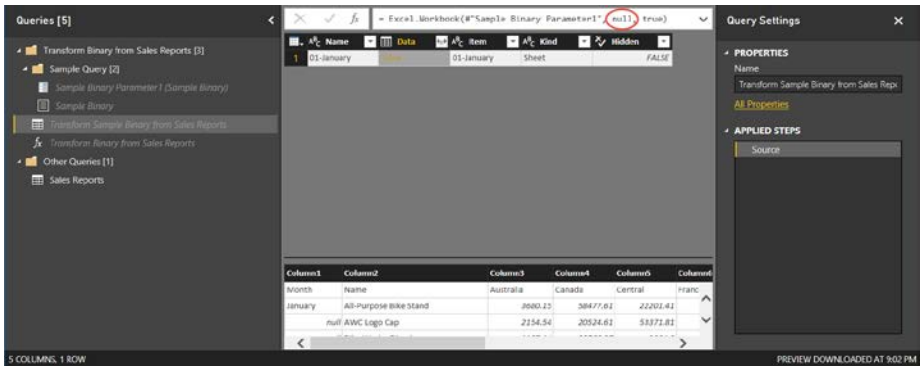


Figure 45. Our current query without any modifications and the "null" value still in the formula.

This behavior only happens for objects from the workbook like defined names and worksheets, but not for tables. The easiest and quickest fix is to simply **change the "null" in the formula bar to be "true".**

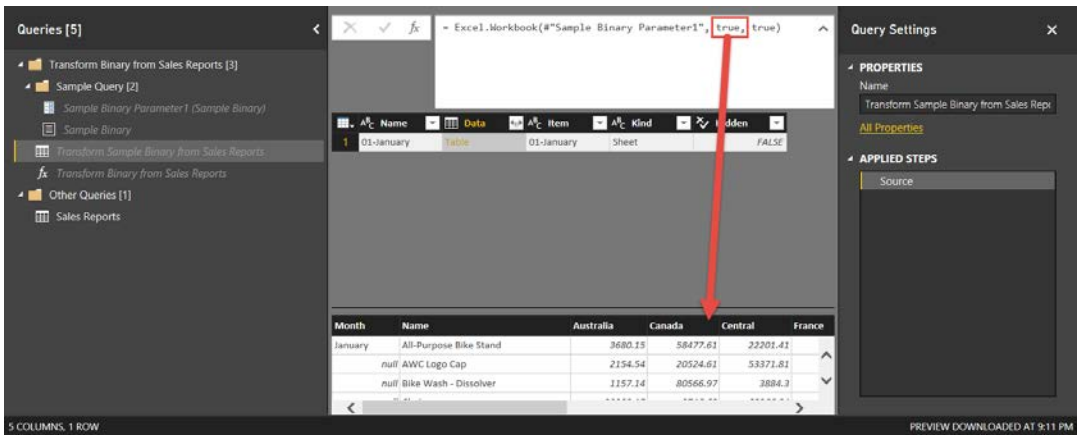


Figure 46. Our current query with the modifications to the formula bar to promote worksheet headers and defined names.

11. The next step is to set up fail-safes just like we did in the original Sales Reports query. We can run any type of filters, including:

a. Filter by the Kind column –extract data from only sheets to avoid data duplication.

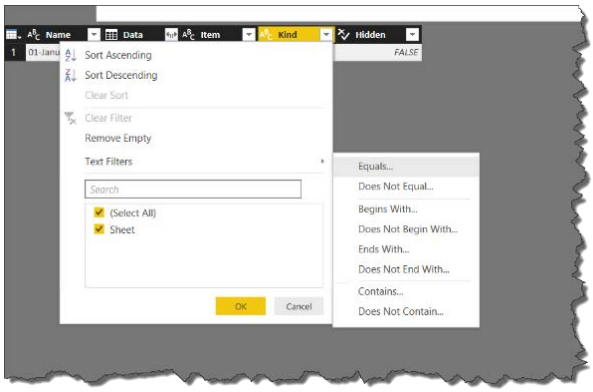


Figure 47. Filtering the Kind column to only get data from Sheets.

b. Filter by the Hidden column – do not include data from sheets that are hidden in the workbook.

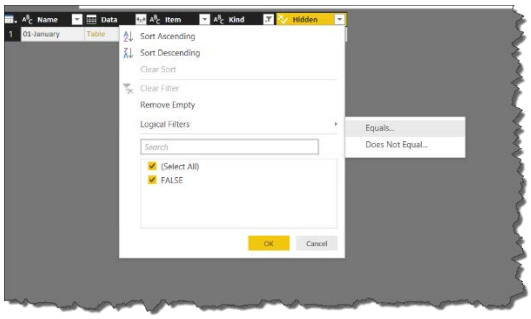


Figure 48. Filtering by the Hidden column to only include visible sheets.

12. Double click on the "Name" column and rename it as "FileName". You could also right click on that column to select the **"Rename"** option.

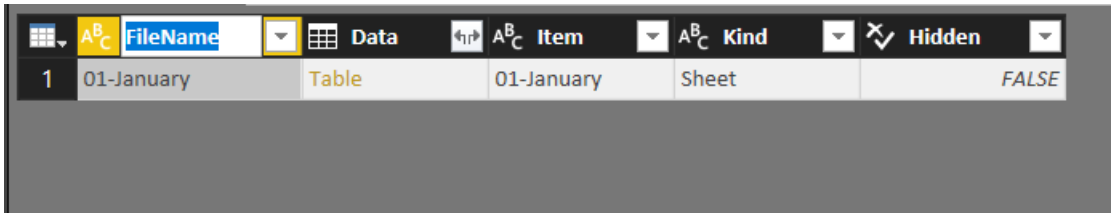


Figure 49. Renaming a column.

13. Select both the **"FileName"** and **"Data"** columns (select one and then hold the Ctrl key to select the other as well) and right click on either one. On the contextual menu, select **"Remove Other Columns"** as we no longer need them.

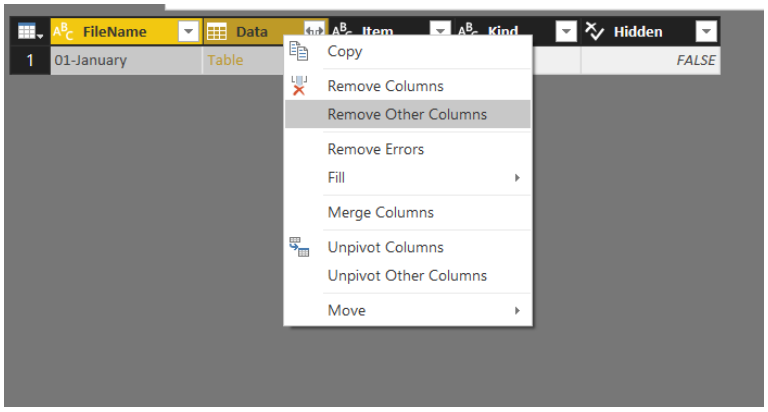


Figure 50. Removing unnecessary columns.

14. You should now have two columns. **Click on the double opposite arrows** icon that you see beside the "Data" column to expand the columns from that table. Be sure to select all the columns that you see listed.

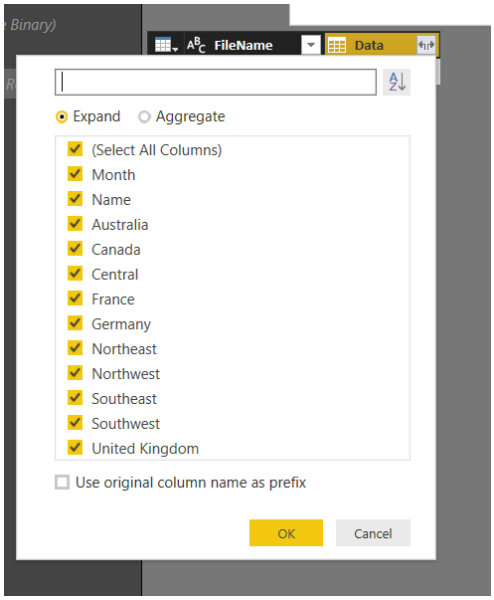


Figure 51. Expanding the columns from the "Data" column.

15. Clean up the query by removing all the columns that we do not need. For example, Month appears as both the filename and the "Month" column from the table, so we can **delete the "Month" column and simply keep the filename.**

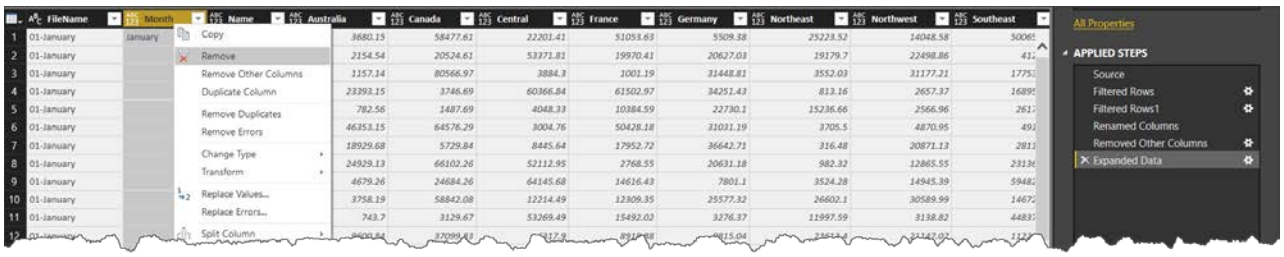


Figure 52. Removing the "Month" column.

16. Select both the **"Filename"** and **"Name"** columns and right click either one. From the contextual menu, **select "Unpivot Other Columns"**. This will unpivot all the columns except the ones that we've selected.

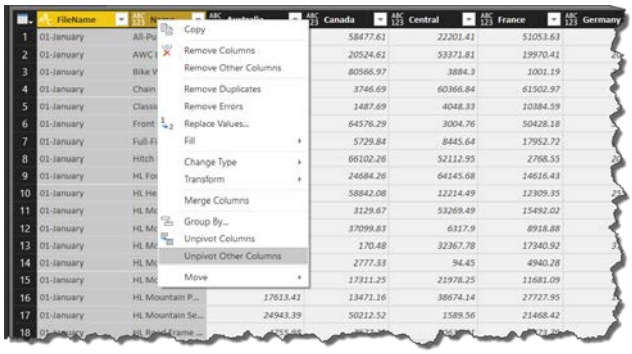


Figure 53. Unpivoting the columns of a table.

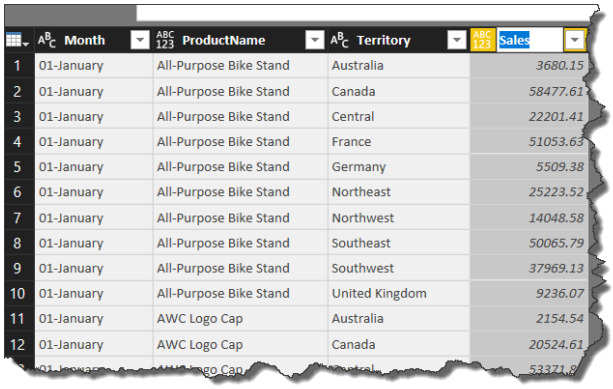
17. Rename these columns as follows:

a. **FileName → Month**

b. **Name → ProductName**

c. **Attribute → Territory**

d. **Value → Sales**



	Month	ProductName	Territory	Sales
1	01-January	All-Purpose Bike Stand	Australia	3680.15
2	01-January	All-Purpose Bike Stand	Canada	58477.61
3	01-January	All-Purpose Bike Stand	Central	22201.41
4	01-January	All-Purpose Bike Stand	France	51053.63
5	01-January	All-Purpose Bike Stand	Germany	5509.38
6	01-January	All-Purpose Bike Stand	Northeast	25223.52
7	01-January	All-Purpose Bike Stand	Northwest	14048.58
8	01-January	All-Purpose Bike Stand	Southeast	50065.79
9	01-January	All-Purpose Bike Stand	Southwest	37969.13
10	01-January	All-Purpose Bike Stand	United Kingdom	9236.07
11	01-January	AWC Logo Cap	Australia	2154.54
12	01-January	AWC Logo Cap	Canada	20524.61

Figure 54. Renaming the columns of the table.

18. Define the data types for each column of our table by simply right clicking them and select the "Change Type" option. Alternatively, you can click on the icon to the left of the column name and define the data type from there. In this case, you should only need to define two of these data types as follows (the other ones should be already defined):

    a. **ProductName** → Text data type

    b. **Sales** → Fixed Decimal Number data type

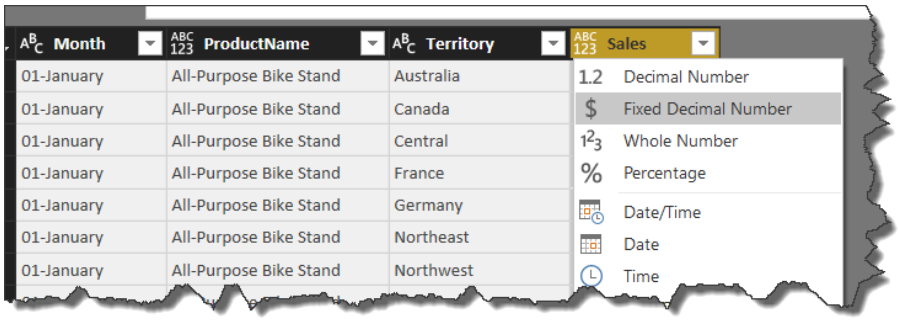


Figure 55. Defining data types in Power Query.

19. Return to the Sales Reports query and note that the last step is returning a different output than the one that you saw the first time. That's because we told Power Query how each file should be accessed and transformed in the **Transform Sample Binary from Sales Reports query**. You can check that the same steps we applied to that query have been applied to all the files in the Sales Reports query. All the data is ready for consumption either in Power Pivot or a new table in Excel, except for one thing. Look at how the icon next to most of the columns names is giving you an **indicator that no data type has been defined**, which we should do as a best practice.

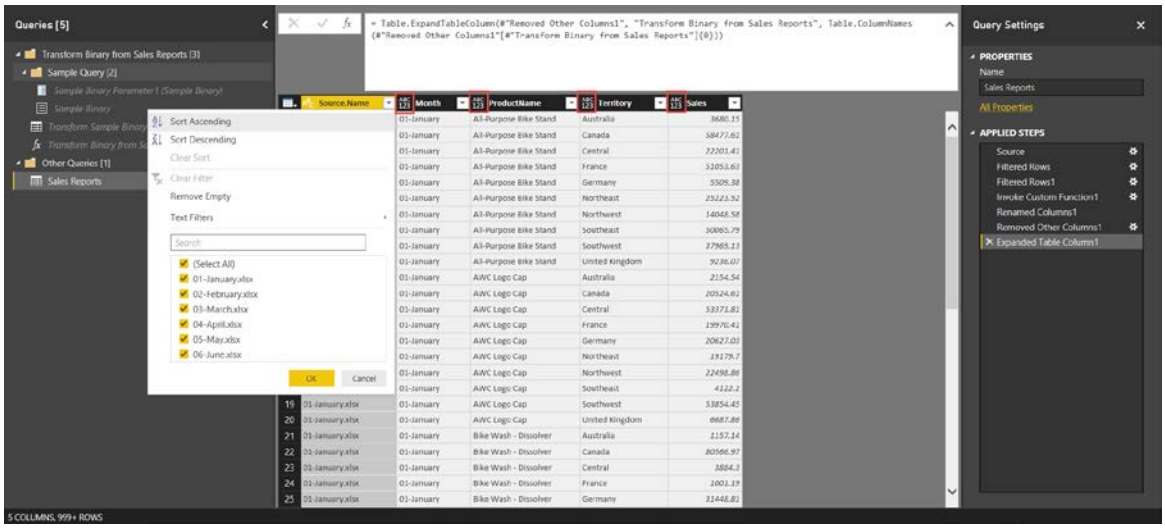


Figure 56. Return to the Sales Reports query and define the data types.

20. Once all the data types for each column has been defined, select "Close & Apply" and load the table to your Power BI data model or Excel.*Example Files\Chapter 3 - Combing Files from Folder\Chapter 3 - Combining Files from Folder [Method1].pbix*

Method 2: Read the Excel File and access a specific object-name combination

The following set of steps will show how to access a specific object-name combination in each of your files and append them.

21. Continuing from step 7 above, instead of selecting ”Sample Binary Parameter [1]”, select the”01-January” option from the list. When you click on it, you’ll notice a preview of the data to your right.

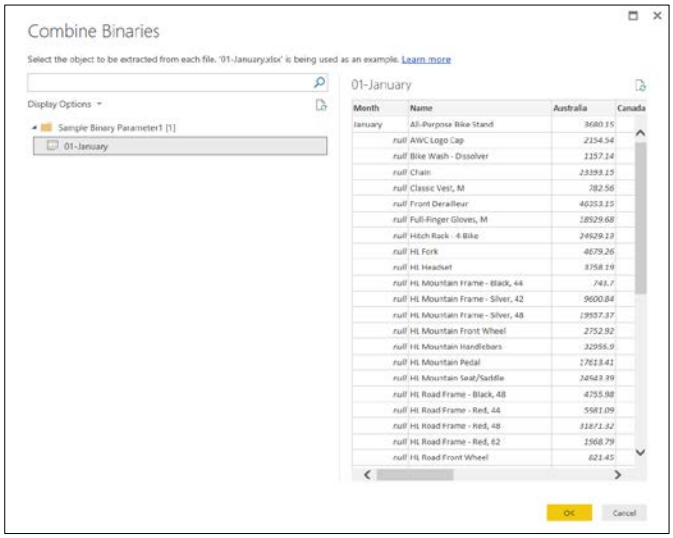


Figure 57. Select the ”01-January” option from the list. You’ll notice that this name is the name of the sheet from your “01-January.xlsx” file.

22. After clicking “OK”, the Sales Reports query now looks like it has combined the data. However, when you try to sort or filter the table, only data for the “01-January” worksheet from the ”01-January.xlsx” workbook is available.

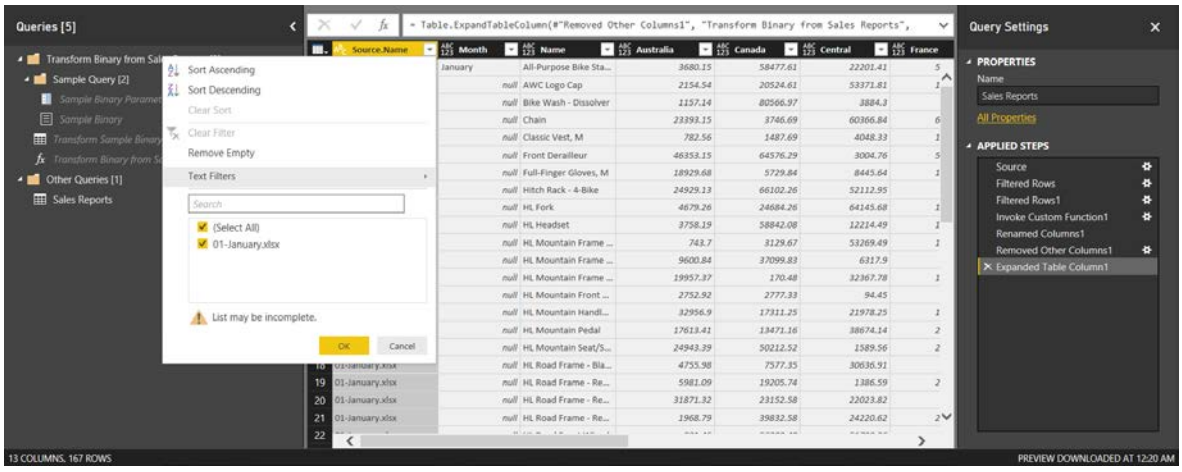


Figure 58. We can only see data for January, but not for the other files. Why is this?

The reason this is happening is because we selected the ”01-January” object in step #21. Essentially, we told Power Query to go through all the files from the Sales Reports query and only extract the data from worksheets named ”01-January”. It just so happens that only the January file has a worksheet with the name “01-January” .

This would’ve worked perfectly if all the worksheets from each of the files had the same name, even something simple like “Sheet 1”. Unfortunately, this is not the case and so it’s not the best route for us.

You might be wondering, “When or why should I go this route instead of the previous one?” It really is a matter of preference and how streamlined your process is. If you just want to access one object from your workbook, and all your workbooks look the same in terms of format and name of worksheets, then this could be the easiest route for you.

Final Note: Be sure to check out our Online Power Query Workshop where we provide additional optimized techniques for combining tables from any source (SQL, non-relational, .csv or any of the previously mentioned file types). All our subscribers get special discounts on the registration price,so be on the lookout for an email from us with your coupon code for a future online workshop.

Case Summary: Combining Files from a Folder

Topics covered

- Connecting to files in a folder
- Filtering columns
- Using the Power Query formula bar
- Types of objects inside an Excel workbook
- Combining binaries
- Renaming columns
- Unpivot other columns

Recommendations

- Power Query will also display subfolders
    - If you connect to a folder that has subfolders, you might want to filter out the subfolders. Alternatively, you can connect to a top-level folder and thereby also connect to any of the subfolders inside of it to combine any of those files as well.
- Combining binaries only works with same type of binaries
    - You can combine csv files, text files, json documents, html documents or any type of binary file from a folder, as long as you don't mix them together and only combine "oranges with oranges" and "lemons with lemons". If you wish to combine Excel, csv and text files together then you'll need to find another route.

Personal comments from the authors

- Other ways to combine multiple tables
    - This is a topic that we cover in-depth in our Power Query workshop. If your goal is to combine tables from Excel, csv, text or even from a database then your best bet is to use the Table.Combine function in Power Query.f you need to combine many tables from different sources, then you could create either a query for each of them and use the manual Append operation or you could combine the Table.Combine function with List.Combine(). Alternatively, if you have other scenarios where you do not know the name of the columns from each of the files that you're trying to combine and you need to preserve some data from the folder, such as the file name, then we recommend creating a dynamic list of headers.

For more information on how to solve other scenarios where you need to combine, consolidate or append data, then we recommend using the following pattern on our site:

Append csv, txt and Excel files from Folder

Master Power Query and become a Data Wizard

Take our Power Query Workshop

Chapter 4: Dynamic Calendar Table

Difficulty: Intermediate

Briefing: Creating a Dynamic Calendar Table

When you are facing a scenario where you need to create a dynamic calendar table (or date dimension table), we can create such table in Excel and use it later in our time intelligence scenarios with Power Query or Power Pivot.

However, this date dimension table is often too long and contains dates that are not relevant to the current report. (For example, including dates up to the year 2020 when we only have data until the year 2017 translates into 5 years of emptiness or waste in our report.)

There are other scenarios where you want a calendar table with culture-specific text labels or other formats (Spanish, Portuguese, etc.).

This chapter will show you how to create a calendar table using nothing but Power Query.

Our goal

- Create a dynamic calendar table that will range from a fixed start date to a dynamic end date based on the current date.

You can open a completely new blank Excel workbook or a Power BI Desktop file to follow along

The How-to Guide: Creating a Dynamic Calendar Table

For this example, we'll be using the Power BI Desktop instead of Excel, but the steps are the same in Excel once you get inside the Query Editor window.

1. From the "Get Data" experience, select "Blank Query" and click the "Connect" button.

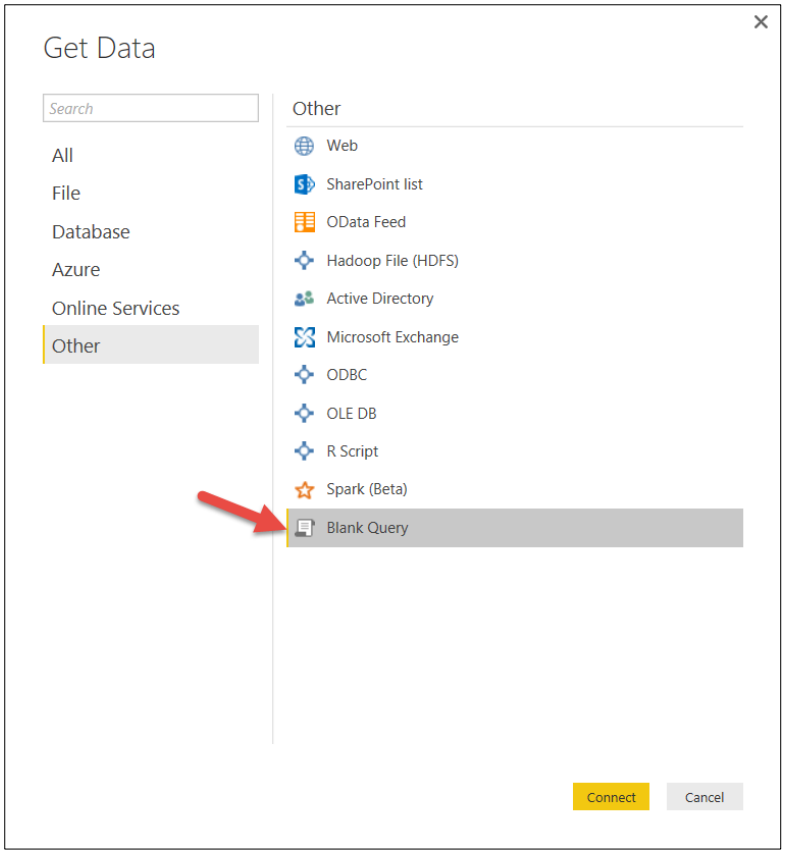


Figure 59. Select "Blank Query" from the "Get Data" window.

2. In the "Query Editor" interface, go to the "View" tab in the ribbon and check the "**Formula Bar" radio box** to enable the formula bar.

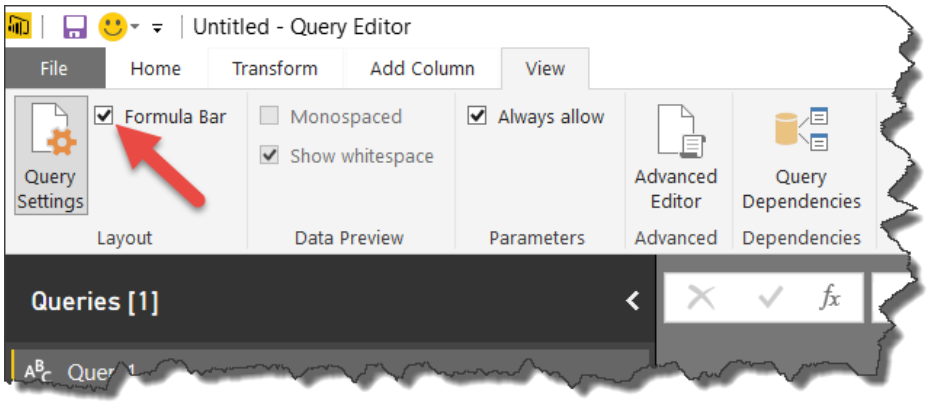


Figure 60. Enabling the formula bar within the "Query Editor".

3. With the formula bar now visible, type the start date of our calendar table by using the literal #date(Year,Month,Day) as shown in Figure 61 below, and press "Enter".

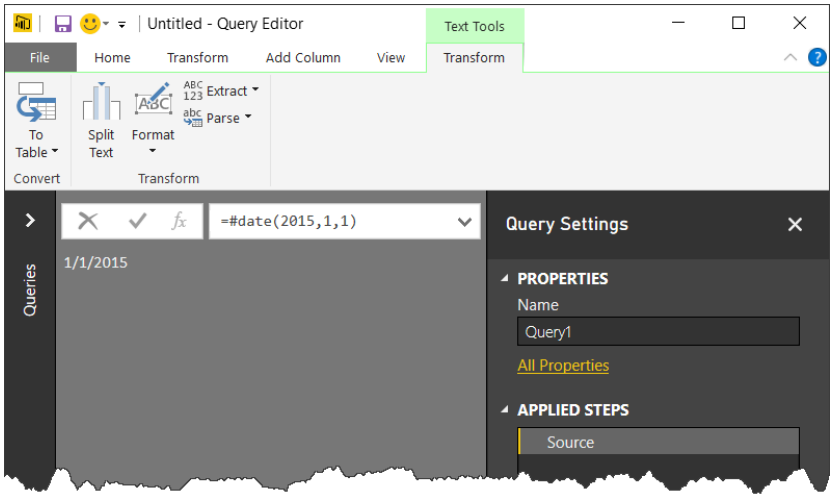


Figure 61. Adding the first date of our calendar table in the formula bar.

4. Next, click the "**fx" icon** next to the formula bar. A new step in the "Applied Steps" field has been created and the formula bar now reads "Source" instead of the numeric date that we entered.

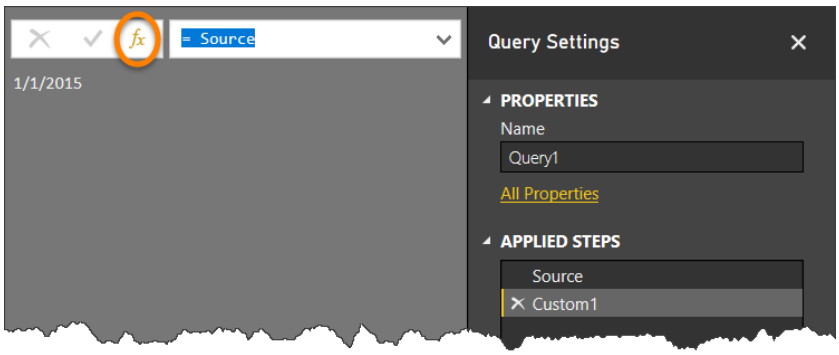


Figure 62. Clicking the "fx" icon creates a new custom step that references the previous step. In our case this previous step is called "Source".

5. Replace the text in the formula bar (**"= Source")** to the following formula:

```
=     List.Dates(Source,     Number.From(DateTime.LocalNow())-     Number.From(Source)
,#duration(1,0,0,0))
```

Let's take a closer look each element of this formula:

a. **List.Dates** – this function helps us a create a list of dates. The inputs are:
   i. A start date,
   ii. The number of values to return, and
   iii. The increment to add.

b. **Source** – as you already know, this is the value of our start date, so it goes inside the first parameter.

c. **Number.From** – we use this function multiple times to transform any type of value into a number to be used in math operations.

d. **DateTime.LocalNow** – this function is the equivalent of "NOW()" for Power Query in Excel. It returns the current date and time to ensure we get a dynamic range of dates until the present day.

e. **#duration(1,0,0,0)** – another literal that adds 1 day (so that our increment is daily).

This formula returns a list of dates as illustrated in Figure 63 below.

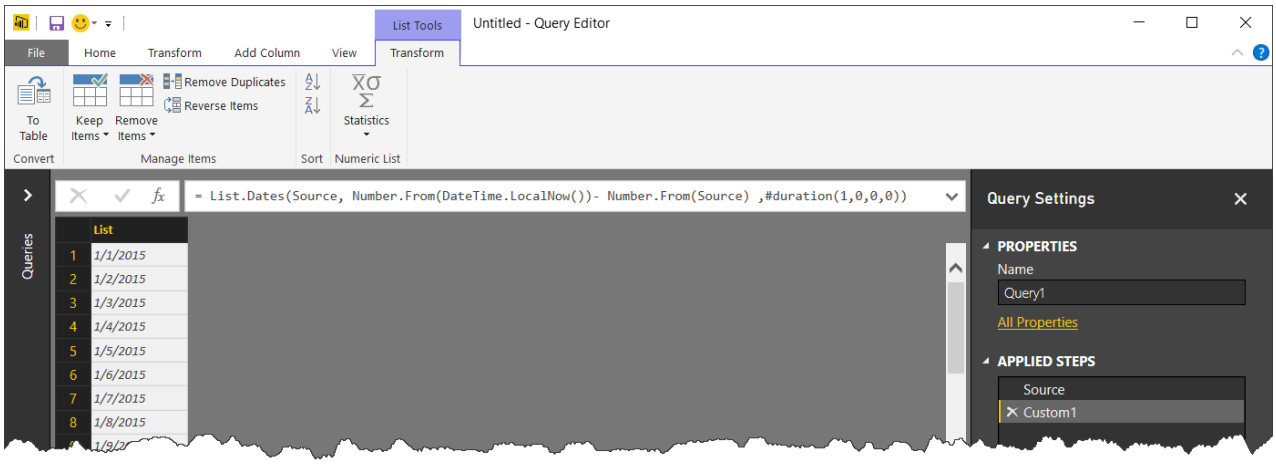


Figure 63. The result of the formula is a list of dates from the fixed start date that we defined in the Source step, until the very last date of the current year.

6. Lists are awesome, but we would like to work with a table instead of a list so we need to click on the **"To Table" icon** on the top left of our screen. Once in the "To Table" window, click "OK" (the defaults here have the correct values).

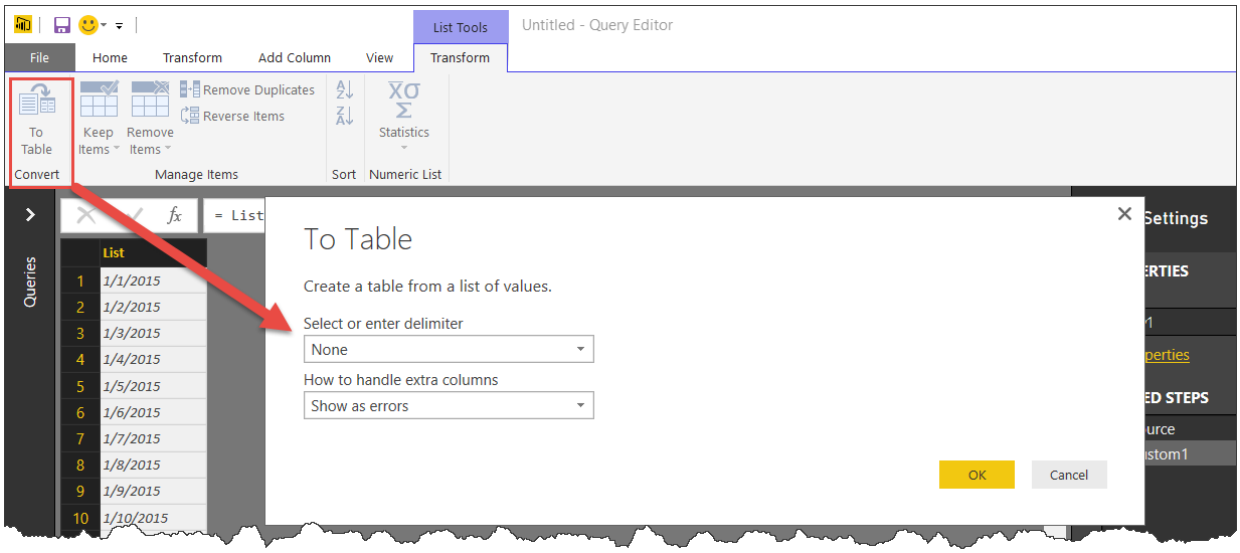


Figure 64. Transforming the list of dates into a table of dates.

7. Double click on the column and rename it as "Date". We recommend you also define the data type of this column as "Date".

8. Now that the data is a table with a column containing dates, we can take advantage of the Power Query user interface by adding new number-based columns using the options from the **Add Column tab** in the ribbon**.** Select the date column, go to the Add Column tab in the ribbon and click the "Date" icon in the "From Date & Time" section. This will display a list of options for new columns that can be created. For this scenario, select "Year".

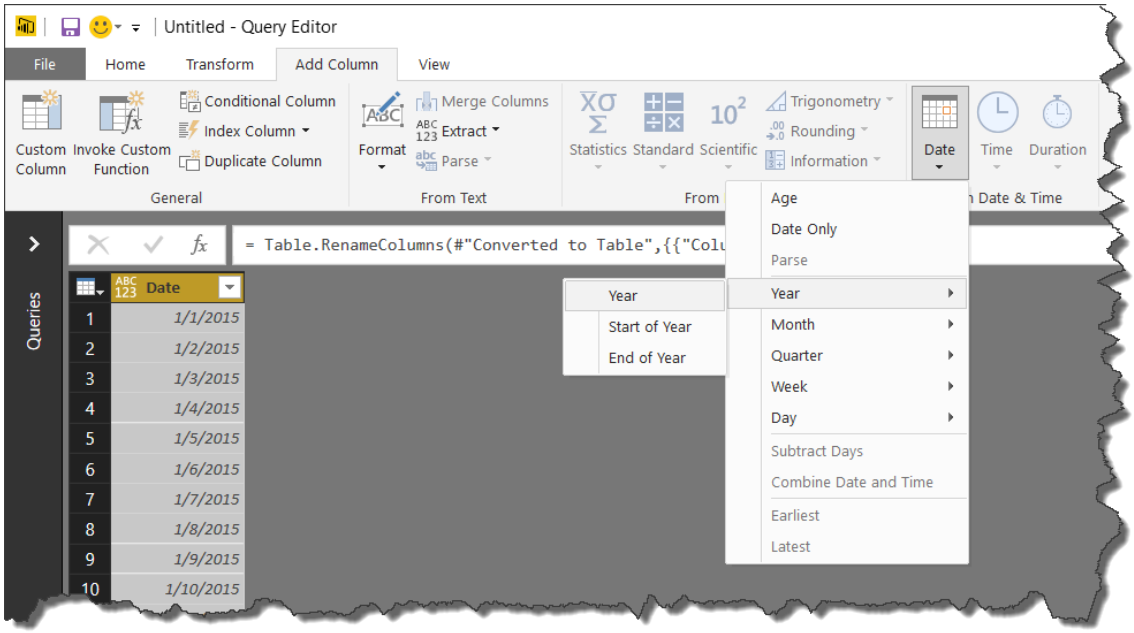


Figure 65. Selecting a column of dates will enable the "Date" selection on the Add Column tab.

9. Repeat step 8 two more times, - first selecting "Month" and then selecting "Quarter of Year". The result will look like Figure 66 below.

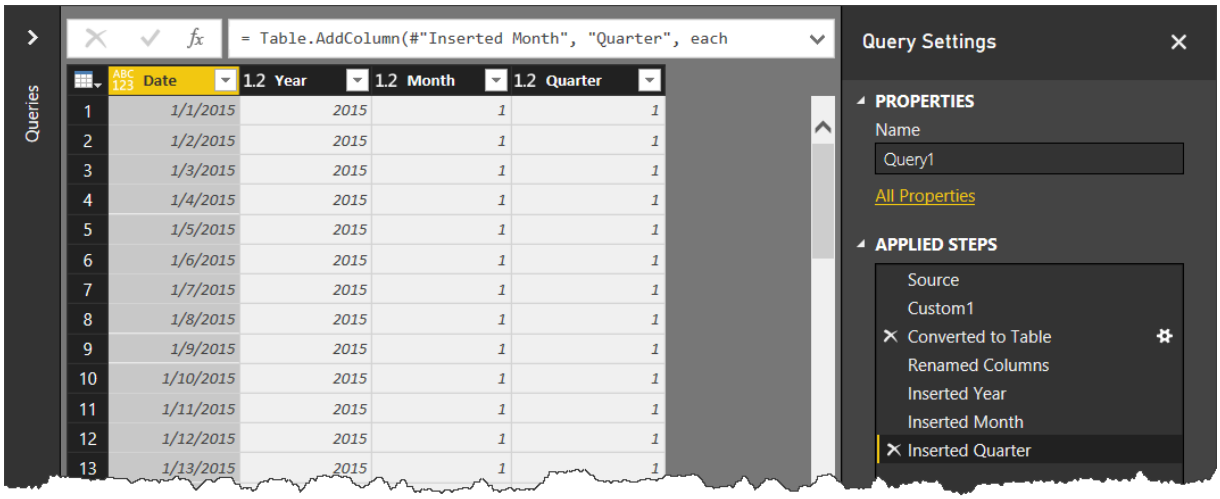


Figure 66. The Year, Month and Quarter columns were created by simply using the UI.

10. To add a column for the name of the month, we repeat the same operation by going to the "Month" section and selecting "Name of Month". One thing to mention is that the result will vary depending on the regional settings of your OS. If for some reason you get

the result in another language, go to the Applied Steps list, click the gear icon and change the name of month to the language that you want. In Figure67, the name is changed from English to Spanish (Panama) by simply clicking on the gear and icon and setting the locale.

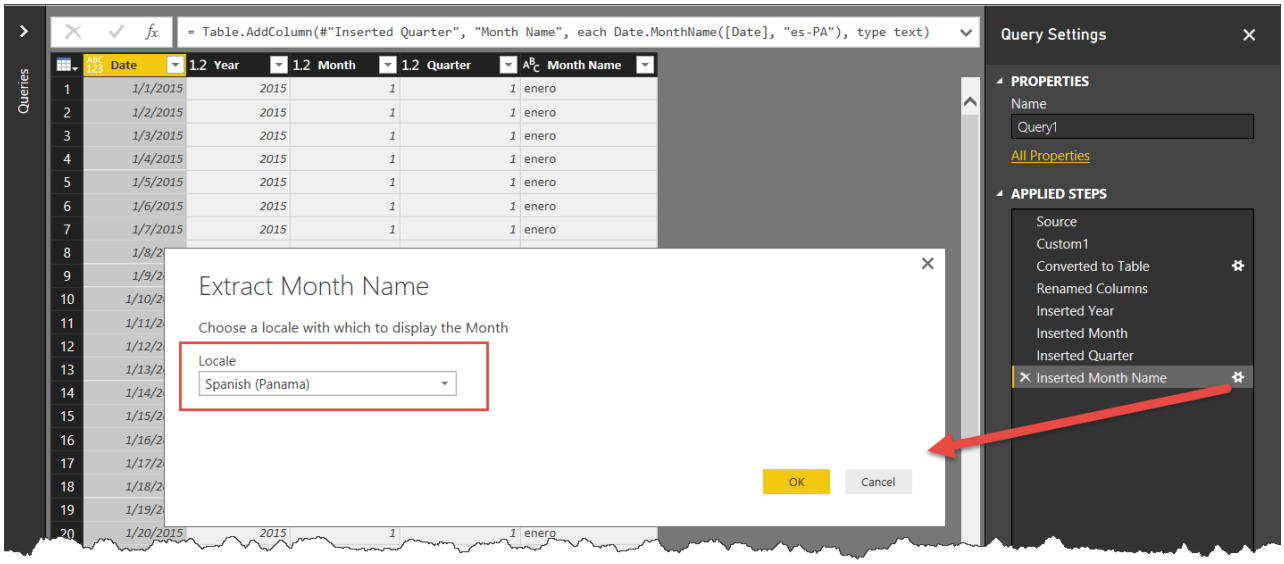


Figure 67. Defining the locale for our newly created text-based column.

11. Keep adding new columns as desired and once you're happy with your result, click on "Close & Load" (or "Close & Apply" for the Power BI Desktop). The resulting table will look like Figure 68 below.

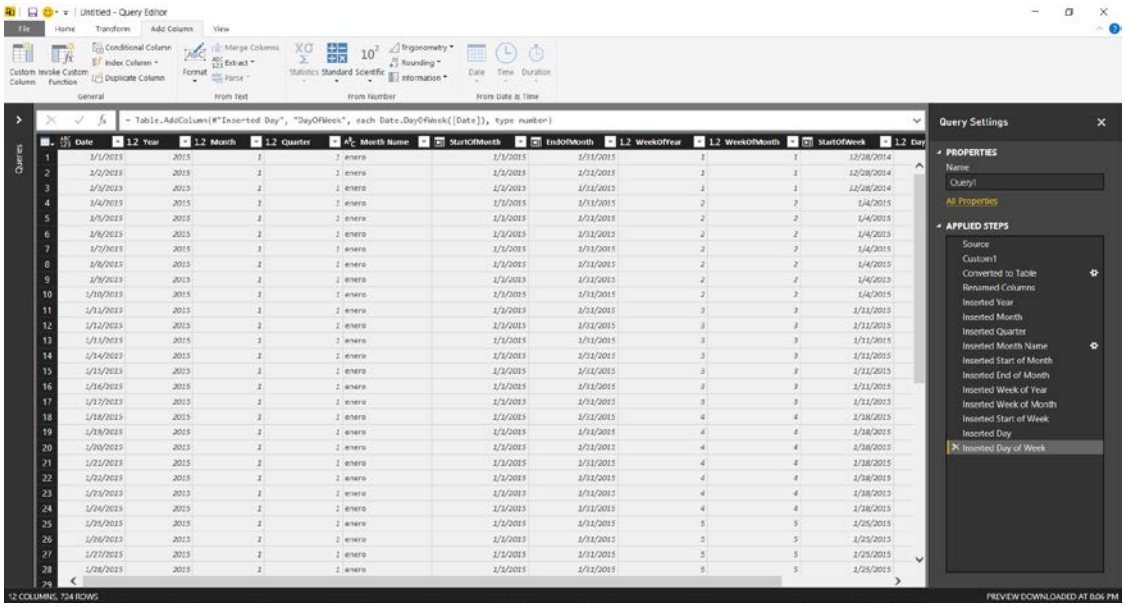


Figure 68. A dynamic calendar table created in Power Query.

Case Summary: Creating a Dynamic Calendar Table

Topics covered

- Using the Power Query formula bar
- Literals in Power Query (#date, #duration)
- Using the Date.List function to create a list of dates
- DateTime.LocalNow is the equivalent of the Excel NOW() function in Power Query
- Cultural references or locale for Power Query functions
- Basic Power Query ribbon usage
    - How to create new columns based on current data

Recommendations

- Take the server settings into consideration.
    - Whenever you publish your workbook or pbix file to Power BI, the refresh operation will occur at the server level. The time zone of the server might be different to yours, so the `DateTime.LocalNow()` function might provide different results. To find out your server's time zone, use the `DateTimeZone.FixedLocalNow()` function instead and adjust your solution accordingly to have the right context and calculations across your model.
- If your objective is to use this table for Power Pivot time intelligence functions:
    - Change the formula in step 5 to:

      ```
      List.Dates(Source,Number.From(Date.EndOfYear(DateTime.LocalNow
      ()))-Number.From(Source), #duration(1,0,0,0))
      ```

      DAX requires the end date of your calendar be the very last date of the year, otherwise some calculations might not yield correct results.

Personal comments from the authors

- Other type of calendar tables:
    - You might be curious to know how to achieve a 4-4-5 table or other types of calendar tables. With Power Query, you can create any type of grouping using the "Group By" functions combined with other techniques such as indexing.
- Adding new columns based on existing data with the Power Query UI:
    - Power Query's User Interface is probably one of the best UI's that we've seen for Data Analysts. It excels in its capabilities and is incredibly easy to use. We encourage you to explore the "Add Column" tab in the ribbon to see how intuitive it is. For example, depending on the data type of your column, some icons will be greyed out. This provides an easy visual cue to quickly determine if an option is available or not for that particular data type.

Master Power Query and become a Data Wizard

Take our Power Query Workshop

Chapter 5: Time Intelligence

Difficulty: Advanced

Briefing: Creating a Dynamic Calendar Table

In DAX (Power Pivot's Language), we have functions like DATEADD that can bring us the dates of a previous or upcoming year, month or day. We also have other functions such as SAMEPERIODLASTYEAR that brings us the same period of dates but for last year. We also have functions such as TOTALMTD, TOTALQTD and TOTALYTD that can give us a cumulative value based on the year-to-date, quarter-to-date or even month-to-date.

Over the past few years, we've been pushing Power Query to its limits to truly understand its capabilities and weaknesses. We've provided hundreds of solutions to customers worldwide and have conclude that performing Time Intelligence with Power Query is in fact possible. However, there are some considerations to bear in mind when doing these types of operations.

Our goals

We have a Sales table with a "Date" column and a "Sales" column. We need to perform the following calculations:

- The previous year's values for the current date
- The month-to-date total of sales
  - The previous year's month-to-date value for that same day/month combination

Be sure to open the workbook 'Time Intelligence – Start.xlsx' so you can follow along.

The How-to Guide: Time Intelligence

Section I

The following set of steps will help you calculate the previous year's sales by date.

With the **"Time Intelligence – Start.xlsx"** file open:

1. In the Sales worksheet, select any cell inside the table, click the Power Query tab and choose **"From Table".**

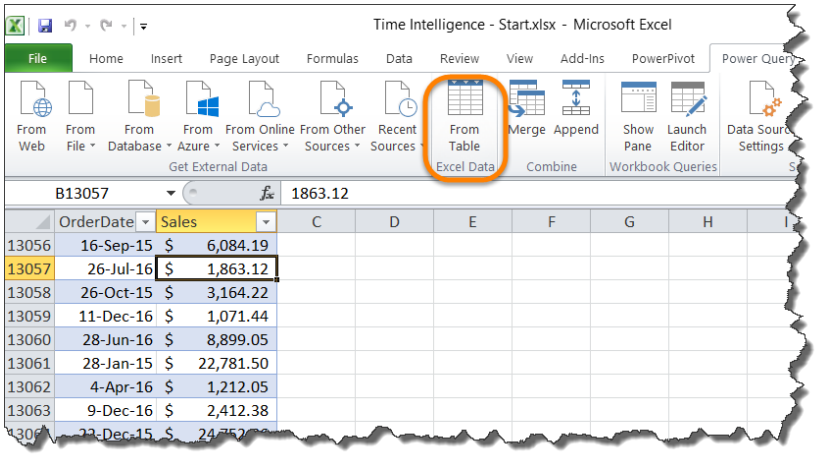


Figure 69. The "From Table" functionality in Power Query using Excel 2010.

2. Once in the Query Editor, the "Order Date" column has a data type of Date/Time. Change that to only be Date by right clicking on the "Order Date" column, choosing the **"Change Type" option** and selecting "Date".

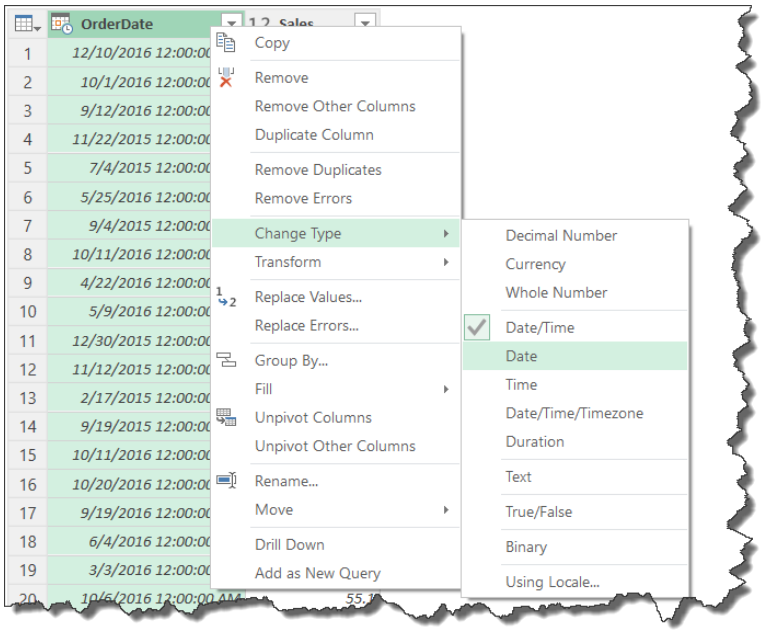


Figure 70. Using the Columns contextual menu to change its data type.

3. Our current table has duplicates in the ”OrderDate” Column, and we would like to summarize by “Order Date” so we can calculate the total sales by date. To do this, we use Power Query’s **”Group By” feature** and summarize our data by ”OrderDate”.

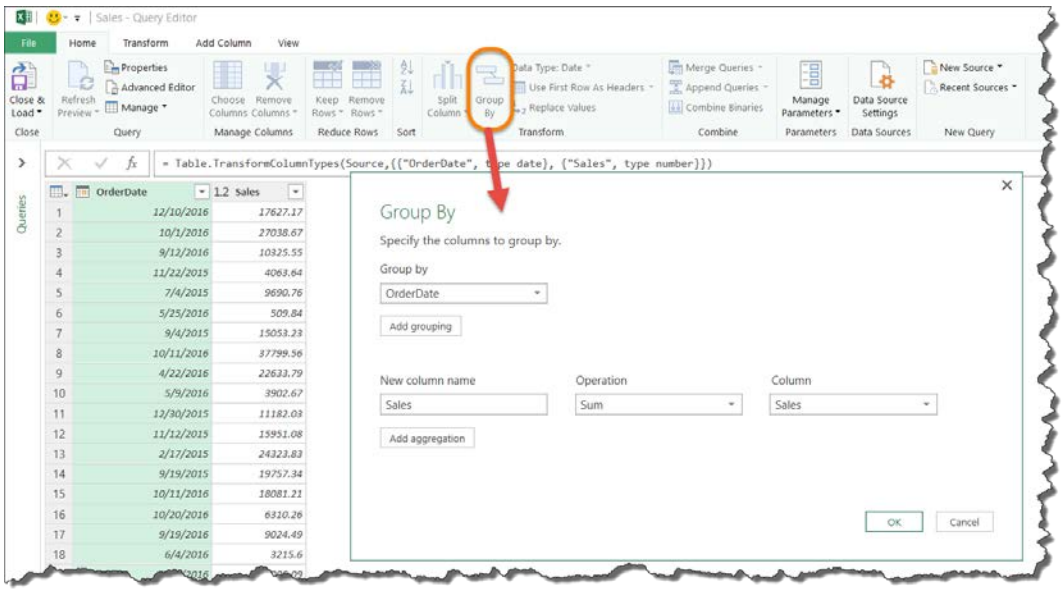


Figure 71. Using the ”Group By” feature to summarize total sales by Order Date.

4. **Sort** the ”OrderDate” column by ascending order so the earliest dates are at the top.

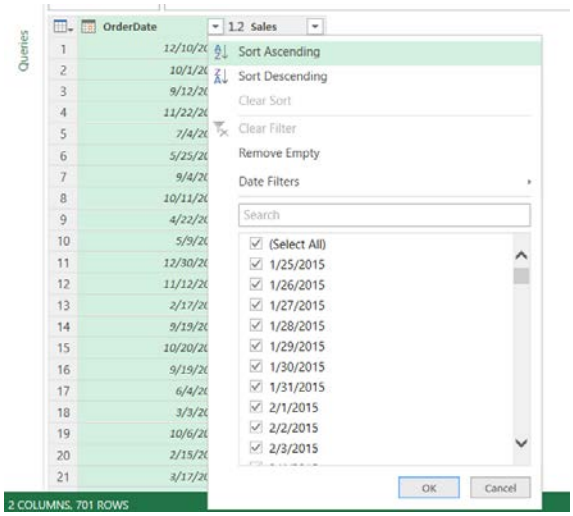


Figure 72. Perform an ascending sort on the “OrderDate” column.

5. Using Power Query’s powerful UI, create two new columns based on the the data in ”OrderDate” column. To do this, select any value in the ”OrderDate” column and go to the Add Column tab on the ribbon. Click on the ”Date” icon located on the right (it should not be greyed out), select the “Year” option and then click on “Year”. Repeat this process, but instead of choosing ”Year”, select ”Month” and then ”Month” again.

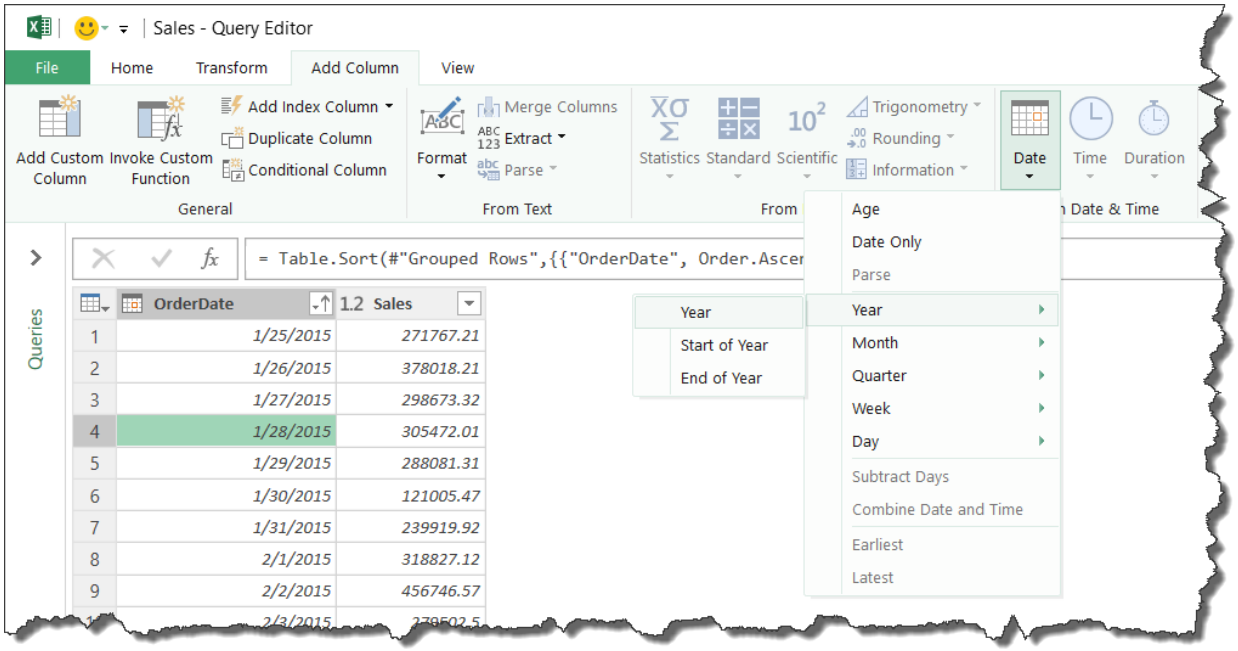


Figure 73. Create new columns based on existing data in your table.

6. Return to the Home tab in the ribbon. Click on the arrow in the corner of the "Close & Load" icon, and select "Close & Load To..." from the dropdown menu. In the "Load To" window, select "Only Create Connection" and click the "Load" button.

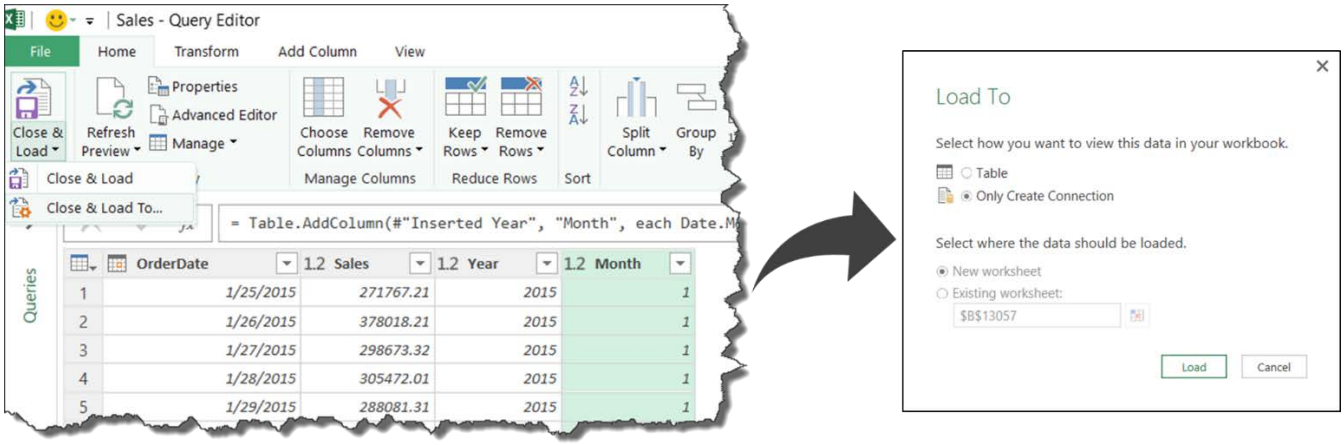


Figure 74. Set the "Sales" query as a "Connection Only" query.

7. By default, once you click on "Load" you'll activate the query pane on the right side of your Excel window. If you do not see it, enable it by clicking the "Show Pane" icon in the Power Query ribbon. Once enabled, you will see two queries:

   a. **FilterContext** - this was created by the authors to use later.

   b. **Sales** – this is the query that we just created.

   In the query pane, right click the Sales query and click "Reference" to create a new query that will reference the Sales query.

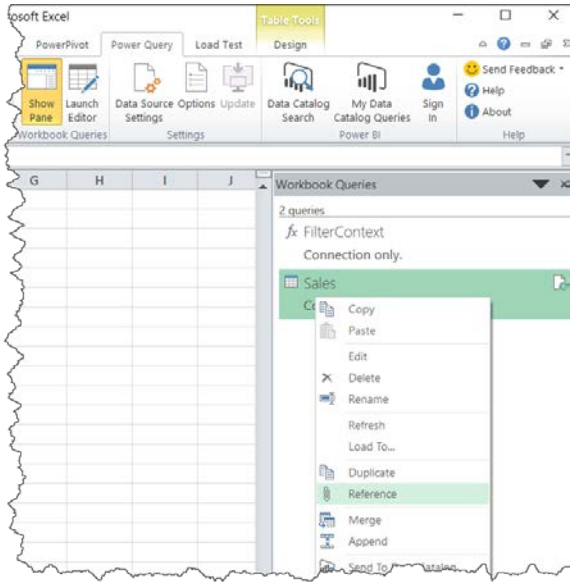


Figure 75. Create a new query by referencing the Sales query.

8. Once you're in the query editor, change the name of this query from "Sales (2)" to "**LastYearSales-ByDate".**

9. Add a custom column by using the **Date.AddYears** function that will subtract one year from the date that in the "OrderDate" column. For example, if the value is 1-March-2016, then the formula will return 1-March-2015. Rename this new column as **'LastYearDate'** and set the data type to "Date".

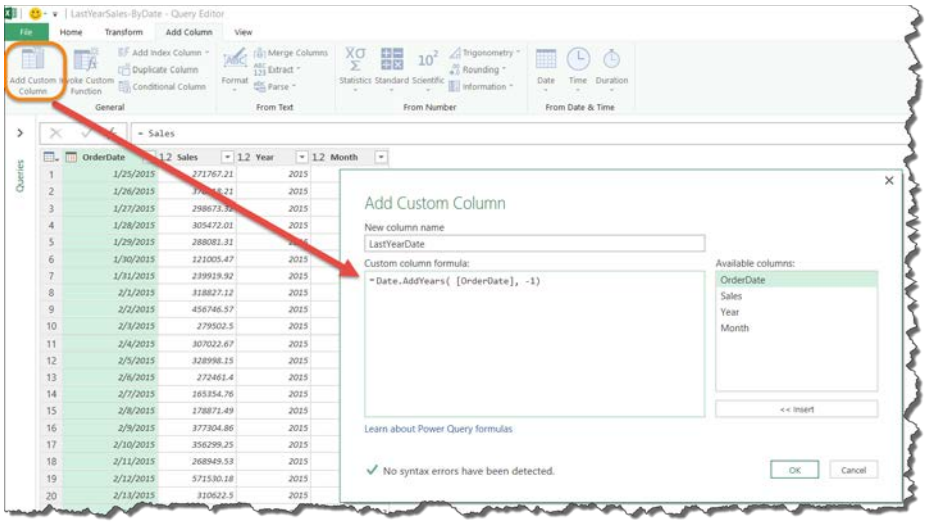


Figure 76. Add a custom column that will return the same date but for the previous year.

10. Merge the Sales query with this new query to create a new column for **last year's sales** by clicking on the "Merge Queries" icon on the Home tab. Select the "LastYearDate" column from the current query (on the top) and the "OrderDate" column from the Sales query (on the bottom), and click "OK".

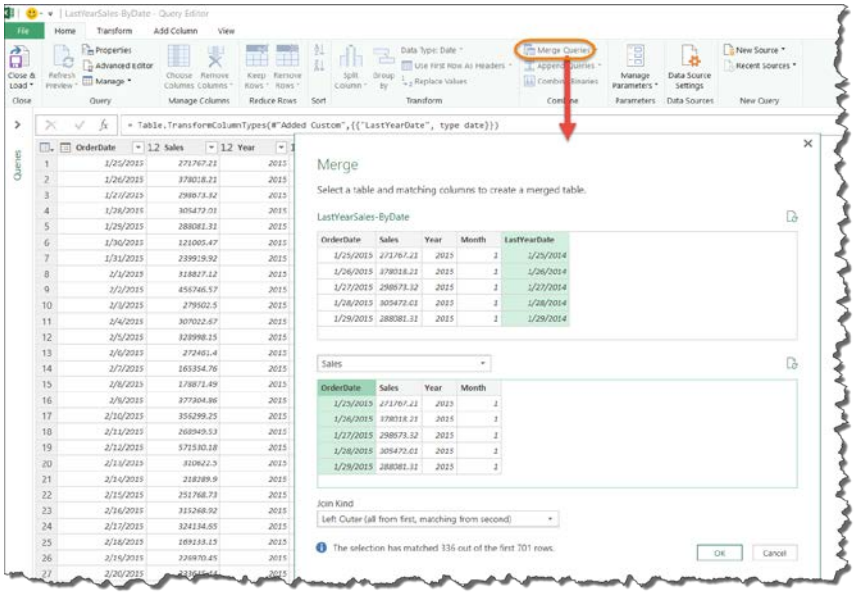


Figure 77. Merging the current query with the Sales query using the new "LastYearDate" column.

11. The previous operation creates a new column called 'NewColumn'. Click on the double opposite arrows to expand the values, selecting only the "Sales" field.

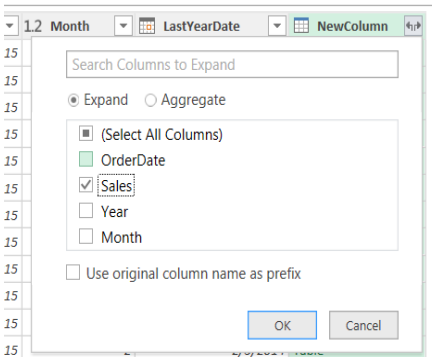


Figure 78. Expanding the fields in the "NewColumn" column.

12. A new column called "Sales.1" was created. Rename it as **"LastYearSales"** and delete all the columns except "OrderDate", "Sales" and "LastYearSales".

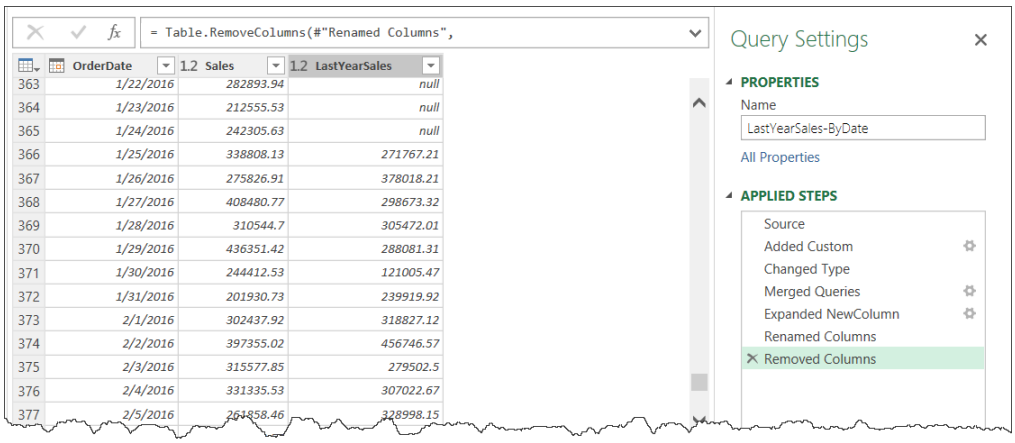


Figure 79. The output of the LastYearSales-ByDate query.

13. Click on the "Close & Load" icon and load the output into a new worksheet instead of creating a connection-only query as we did in step 6.

Section 2

The following set of steps will help you calculate last year's sales by month.

14. Reference the Sales Query again, changing the name of this new query from **"Sales (2)"** to **"LastYearSales-ByMonth".**

15. Group this query by the "Year" and "Month" fields and aggregate the "Sales" column.

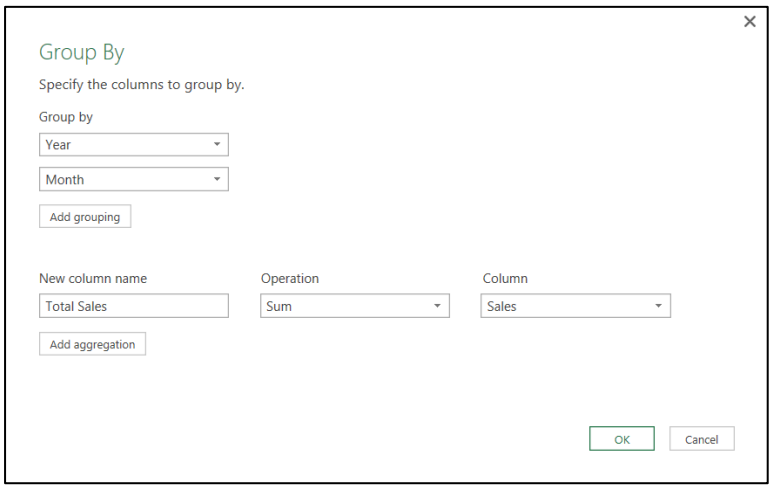


Figure 80. Group by the "Year" and "Month" fields and aggregating by the "Sales" field.

16. Add a custom column that will subtract 1 from the value in the "Year" column to return the previous year. Don't forget to define the data type as "Numeric – Whole Number".

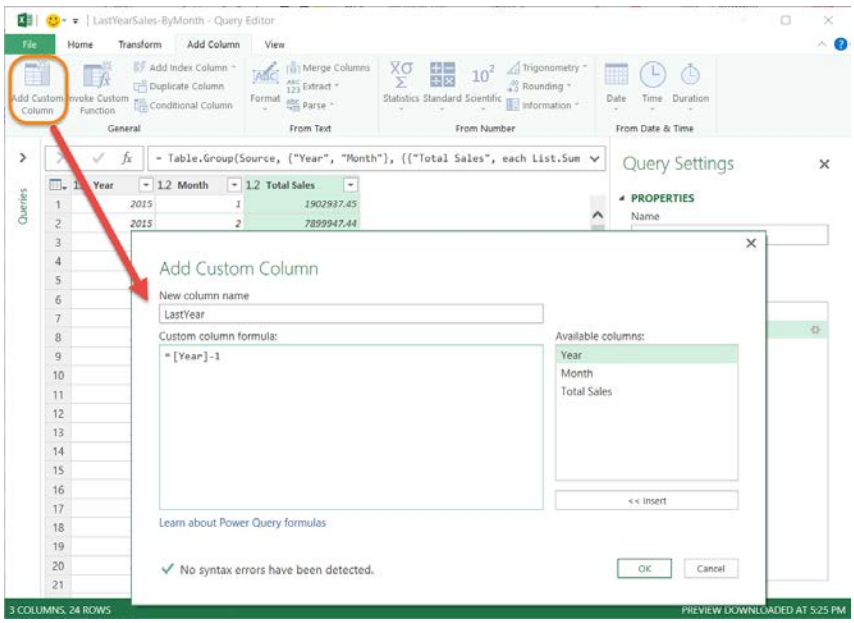


Figure 81. Subtracting 1 from the "Year" value will give us the value for last year.

17. Merge the Sales query with this new query to create a new column for **last year's sales** by clicking on the "Merge Queries" icon. Remember that you need to define the columns in pairs. First, click on the "LastYear" column of the current query, then the "Year" column from the Sales query. Hold down the Ctrl key on your keyboard and select the "Month" columns from both queries. Finally, click the "OK" button.

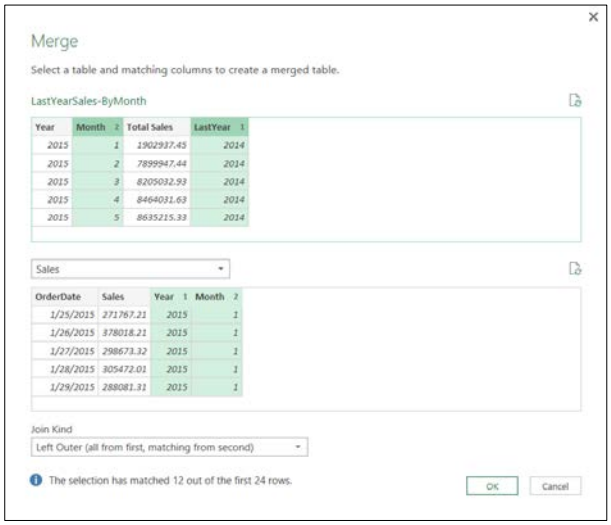


Figure 82. Merge the current query with the Sales query using the "Year" and "Month" columns.

18. The previous operation creates a column named **"NewColumn"**. Instead of expanding the values from this column, click on the "Aggregate" radio box and aggregate these values for the "Sales" field using the same operation.

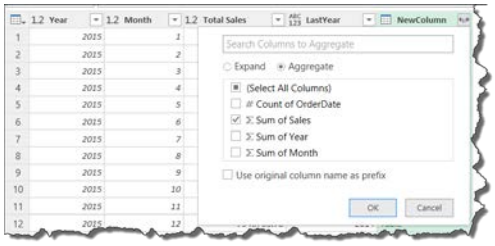


Figure 83. Aggregating the values from a table column named "NewColumn".

19. Sort this table by the "Year" column first (ascending) and then the "Month" column (ascending).

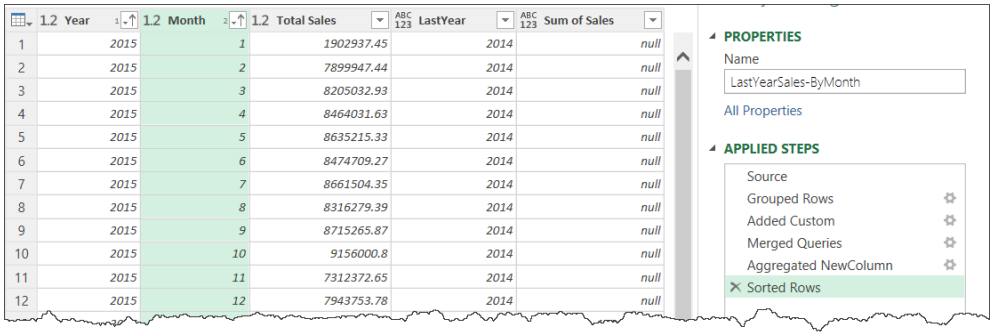
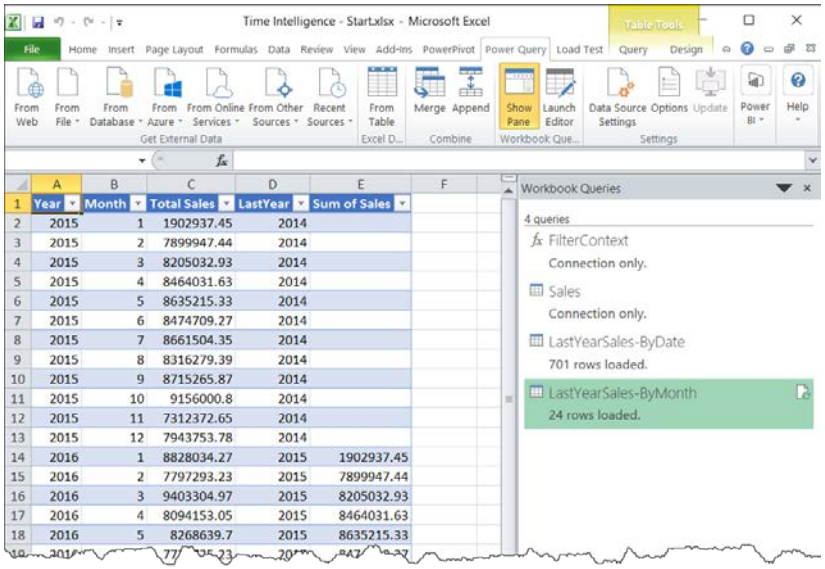


Figure 84. Sorting by the "Year" column and then the "Month" column.

20. Finally, define the data types for the **"LastYear"** and "**Sum of Sales"** columns as "Numeric" and then click on the "Close & Load" icon to load this query to a new worksheet.



Year	Month	Total Sales	LastYear	Sum of Sales
2015	1	1902937.45	2014	
2015	2	7899947.44	2014	
2015	3	8205032.93	2014	
2015	4	8464031.63	2014	
2015	5	8635215.33	2014	
2015	6	8474709.27	2014	
2015	7	8661504.35	2014	
2015	8	8316279.39	2014	
2015	9	8715265.87	2014	
2015	10	9156000.8	2014	
2015	11	7312372.65	2014	
2015	12	7943753.78	2014	
2016	1	8828034.27	2015	1902937.45
2016	2	7797293.23	2015	7899947.44
2016	3	9403304.97	2015	8205032.93
2016	4	8094153.05	2015	8464031.63
2016	5	8268639.7	2015	8635215.33

Figure 85. So far, we've created three queries and loaded two of them as new tables in our workbook.

Section 3

The following steps will help you calculate the total sales accumulated by the date in the current month.

21. Reference the Sales query one more time and call this new query "TotalMTD".
22. Use the "Group By" feature to group by "Year" and "Month", and use the "All Rows" feature as shown in Figure 86.

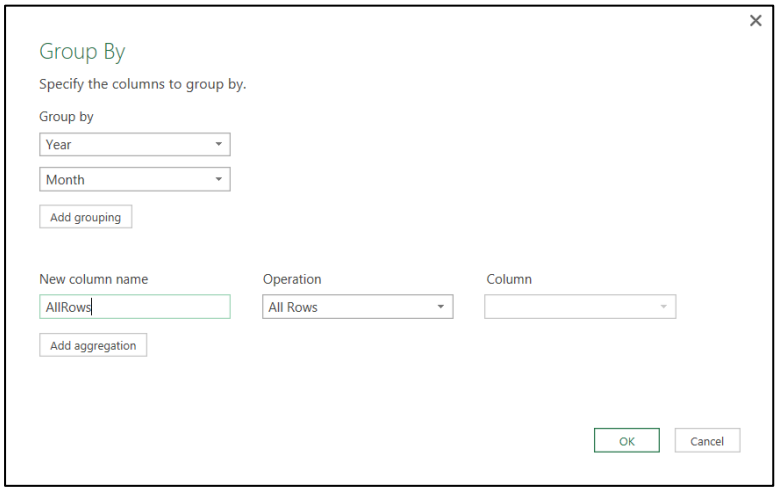


Figure 86. Grouping by the columns year and month, whilst creating a new column using the "All Rows" operation.

23. Right click on the new "AllRows" column and duplicate that column.

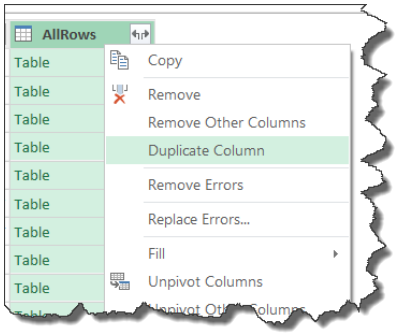


Figure 87. Duplicating the "AllRows" column.

24. Expand the "OrderDate" field from the original "AllRows" column.

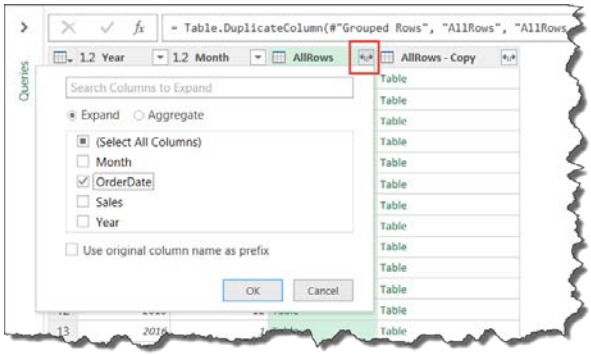


Figure 88. Expanding the "OrderDate" column from the "AllRows" table column.

25. Add a custom column using the custom FilterContest function shown in Figure 98 to create the "Total Month-to-Date" column for each record of the current table.

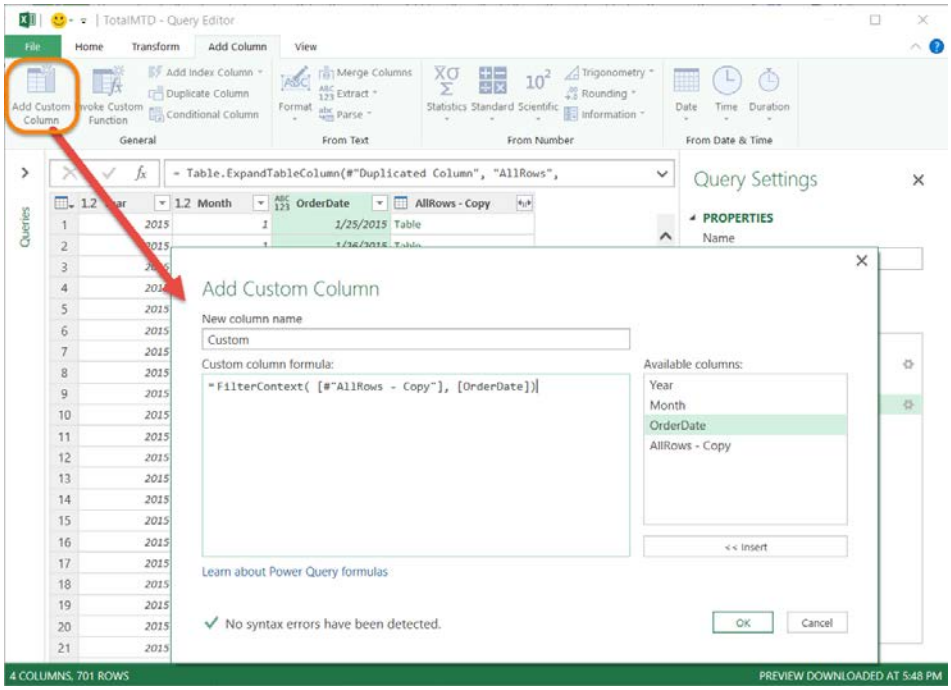


Figure 89. Using a custom function to create a month-to-date calculation.

26. Delete the copy of the ”All Rows” column.

27. Aggregate the values from the “Custom” column by the ”Sales” field using a sum operation.

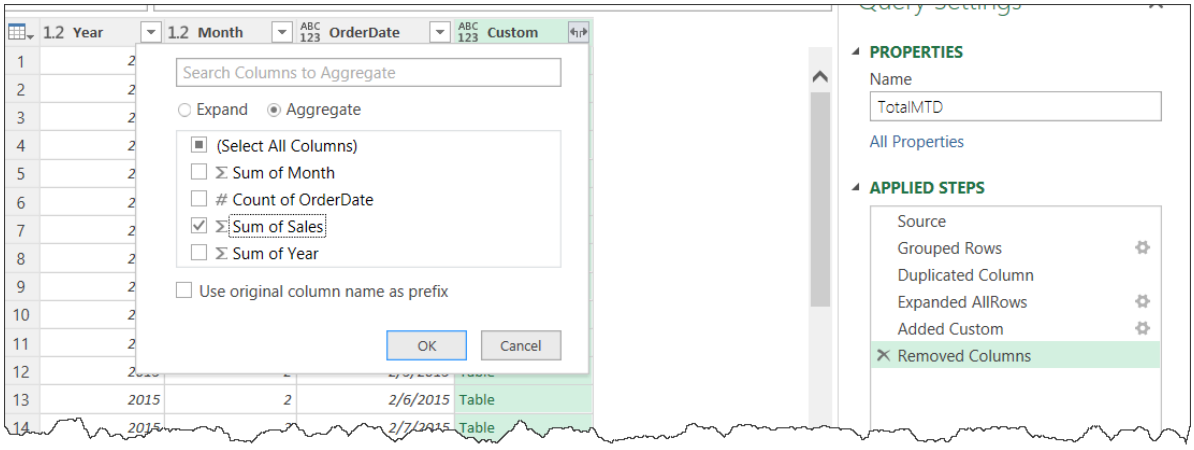


Figure 90. Aggregating the “Sales” field from the “Custom” column so we can have the total month-to-date at the day level.

28. Don’t forget to define the data type of each column.

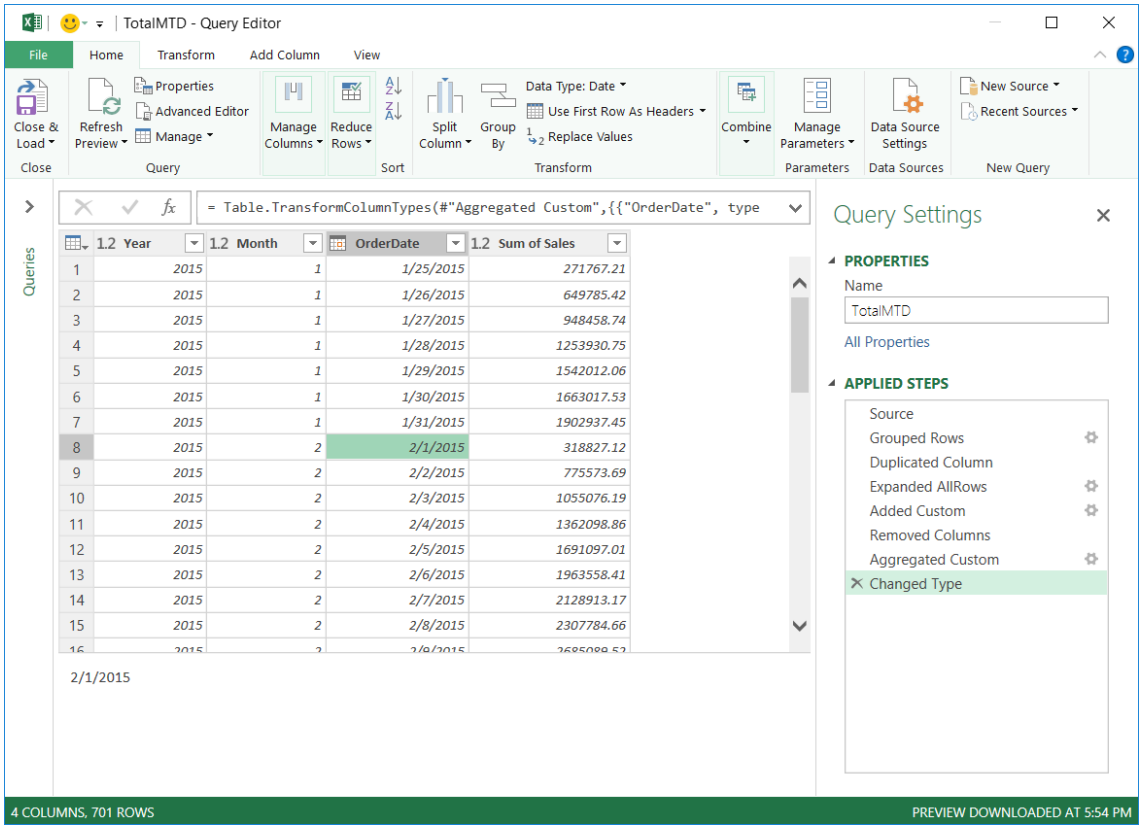


Figure 91. The end-result of our total month-to-date query.

29. Finally, click **“Close & Load”** and load this query to a new worksheet in your current workbook.

Case Summary: Time Intelligence

Topics covered

- Using the "Group By" feature
- Creating a new custom column
    - Using date functions to move through interval of dates
- Merging tables by dates columns
- Working with nested tables
- Using custom functions
- Basic Power Query ribbon usage
    - How to create new columns based on current data

Recommendations

- Table-Record-List navigation can also be used to achieve the Time Intelligence solution that we're looking for. That method is described here, but we highly recommend the steps shown in this chapter as they perform much better. After years of applying both ways to real-life scenarios, we concluded that the Table-Record-List navigation is useful, but it requires a lot of computing power in order to calculate and perform all the necessary scans before delivering the expected result.
- If you'd like to make a more advanced Time Intelligence scenario, like grouping by fiscal year instead of calendar year, then we encourage you to merge the results of your Sales table with your very own Calendar table and then do all the calculations based on how your Calendar table is designed. Check out Chapter 4 for more information on how to create your very own Calendar table.

Personal comments from the authors

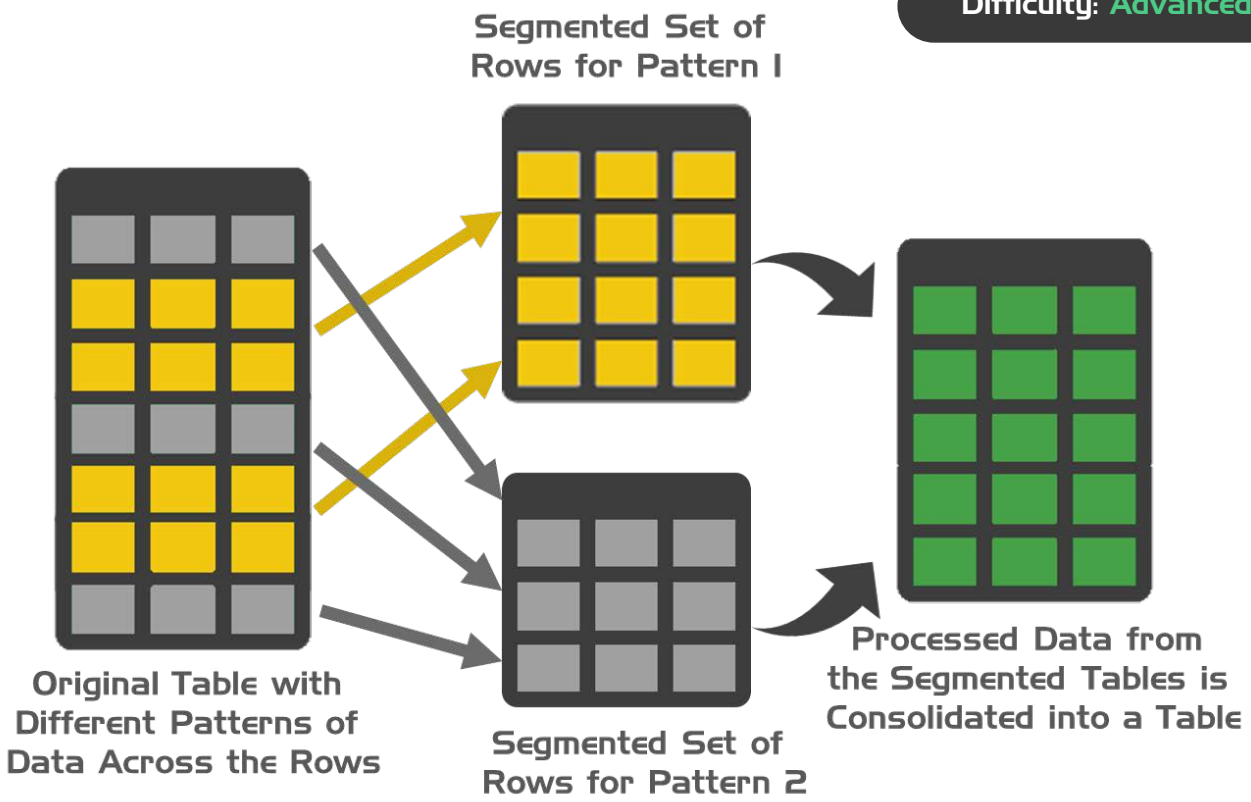
- Other Time Intelligence calculations
    - You can use the concepts described in this chapter as base for any type of business intelligence solution you are building. The crucial part is the grouping of your dates, as this will define everything.
- **Why Time Intelligence in Power Query and Power Pivot?**
    - Power Pivot's engine was specifically designed for calculations just like Time Intelligence and it even has some time intelligence-ready functions, unlike Power Query. The real reason why you'd want to create this calculation in Power Query and not Power Pivot is because you need to use the output value in Power Query for other queries. We mainly see these types of scenarios is simulations. For example, we calculate the data from the last seven days and would like to use that data to forecast the rest of the month, or perhaps do it by month against another variable, and merge this table against another one where we'd need to do other transformations.

Master Power Query and become a Data Wizard

Take our Power Query Workshop

Chapter 6: Handling Multiple Data Patterns in a Table

Difficulty: Advanced



Briefing: Handling Multiple Data Patterns in a Table

This scenario is one we often receive in the follow-up Q&A session of our live online workshops. We deliver solutions to any questions that we receive and we hope to see you in one of our upcoming sessions soon.

When dealing with any type of transformation process in Power Query (or any other tool), we try to find patterns in our data. Once we establish the pattern, we try to work with it to get the result in the format we need. In certain cases, you might find that your data has more than one pattern across a table and we usually try to create a single process that can tackle both patterns at the same time. However, **what happens if you have two or more completely different data patterns in a table that require their own their set of transformation steps?**

For these type of scenarios, we recommend that you create segments from your table. Each one of those segments will then be a new query, and within those queries we will apply the needed transformation steps that will give us the result we need. After we have the two queries from each segment created and with the correct output, we then create another query that will append/combine both queries.

Open the '**Creating Segments – Start.xlsx'** workbook to view the data in its current state.

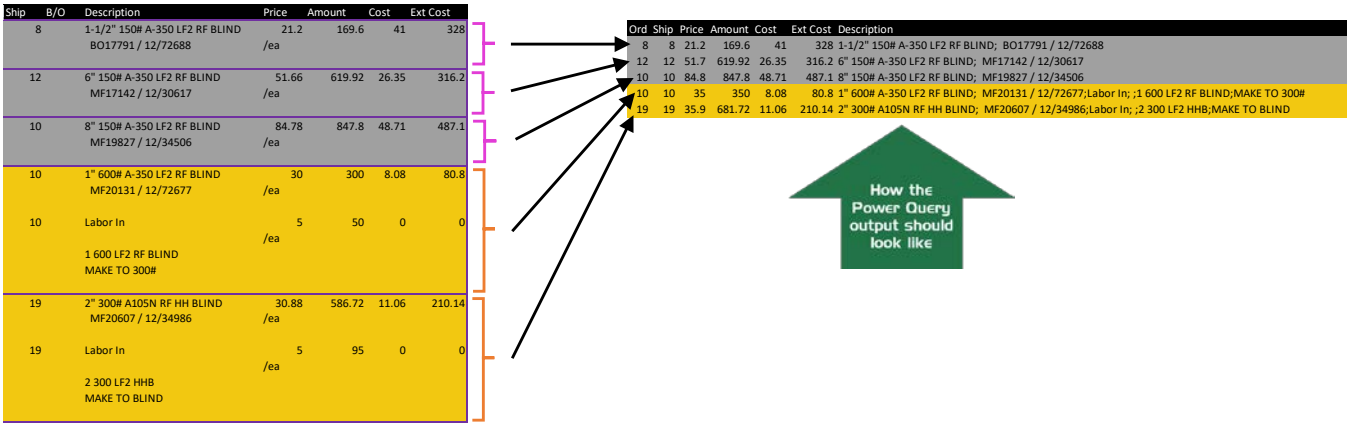


Figure 92. Our source table with color coded rows for easier comprehension of the data patterns.

We have one table with two different patterns across rows but, **how do we differentiate both patterns?** We have taken the time to color code the rows based on the pattern. We have the gray pattern that has a set of three rows for each record and the yellow pattern that has a total of eight rows. Still, how did we find that pattern in the first place or what's the logic behind those patterns?

For the gray pattern, we simply combine the text strings from the Description column, but with the yellow pattern we need to find the group of eight rows, then combine the text strings from the Description column and aggregate the values from the Price, Amount, Cost and Ext Cost columns.

The other main difference between the gray and yellow rows is that the all the yellow ones have a row with the value **"Labor In"** in the Description column.

The business logic between these two patterns is that gray represents a standard product, and yellow represents products that were customized or some labor was added to it before being sold. Therefore, we have the three first rows for the "Original Product" and after the "Labor In" row we have the customization that was added to it.

Here's an example so you can understand this table better:

Imagine a Gibson electric guitar. There are hundreds of these made on a production line, but some had added labor and they are branded as the Gibson Custom Electric guitar series. So, the gray rows are for the regular Gibson guitars and the yellow rows are for the Custom Gibson guitars.

Goals with this Table

- Create two segments from the original table based on the patterns
- Process and transform both segments based on their unique needs
- Combine both segments into one Tall Table

Open the "Creating Segments – Start.xlsx" workbook to follow along.

The How-to Guide: Handling Multiple Data Patterns in a Table

1. Click anywhere on the left-hand table, and select the "From Table" option on the Power Query ribbon.

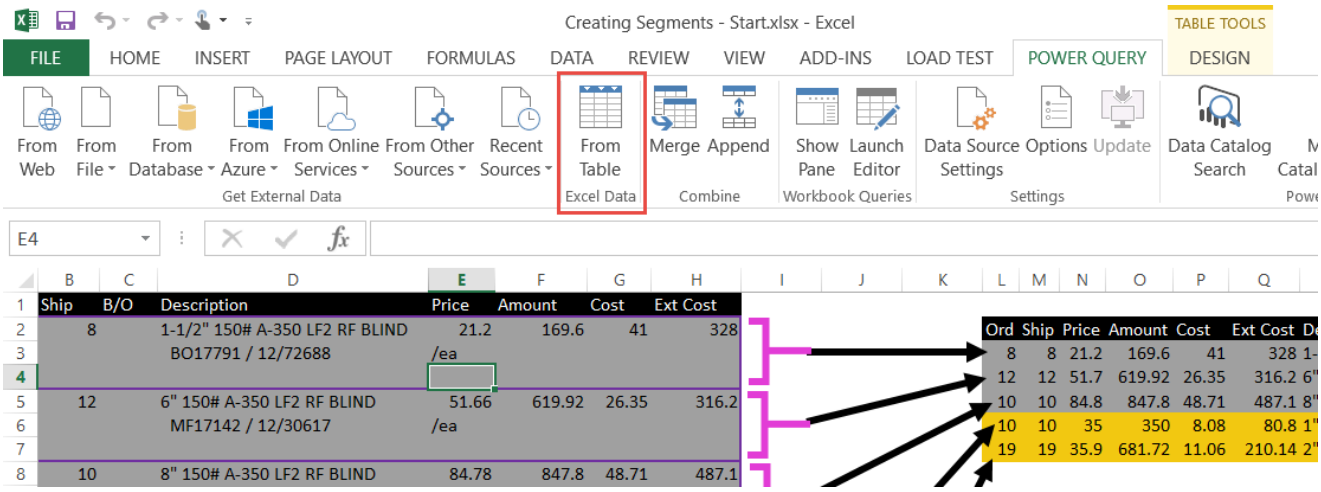


Figure 93. Getting a table into Power Query

2. The "Query Editor" window is where you can perform all the needed transformations. The first one a simple filter on the "Description" column to get rid of the null values by deselecting the "(null)" value and clicking "OK".

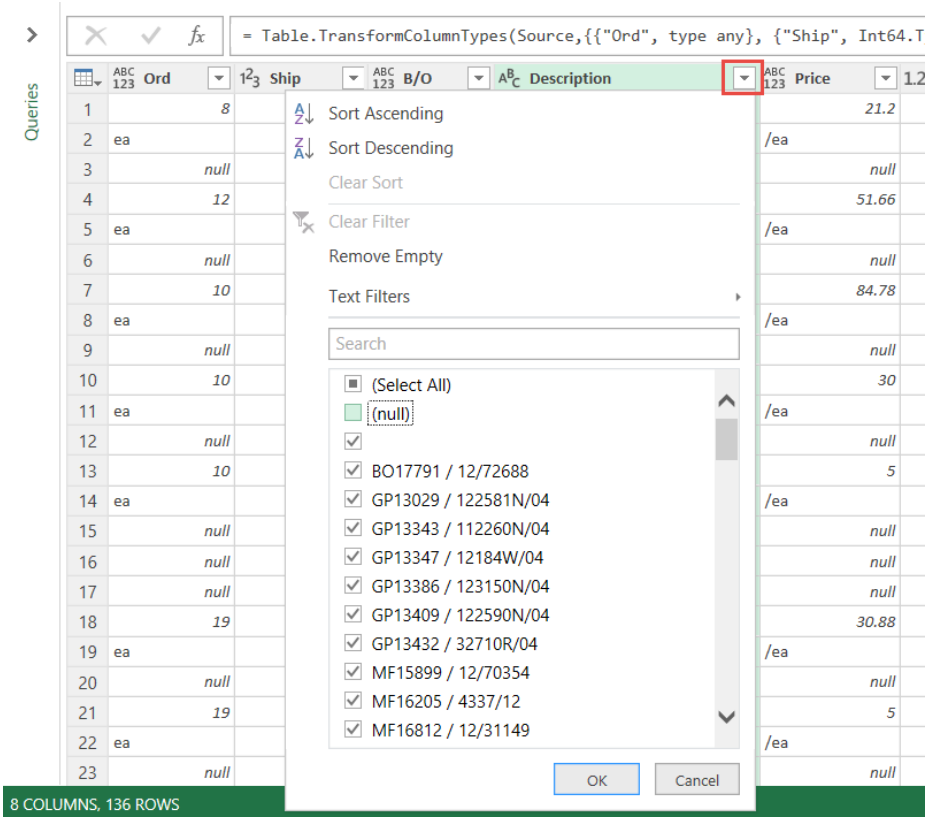


Figure 94. Filter the "Description" column to get rid of any null values.

3. Change the data type of the "Ord" and "Price" columns to decimal by selecting both (select one and then hold the Ctrl while selecting the other one) and then defining the data type in the Home tab.

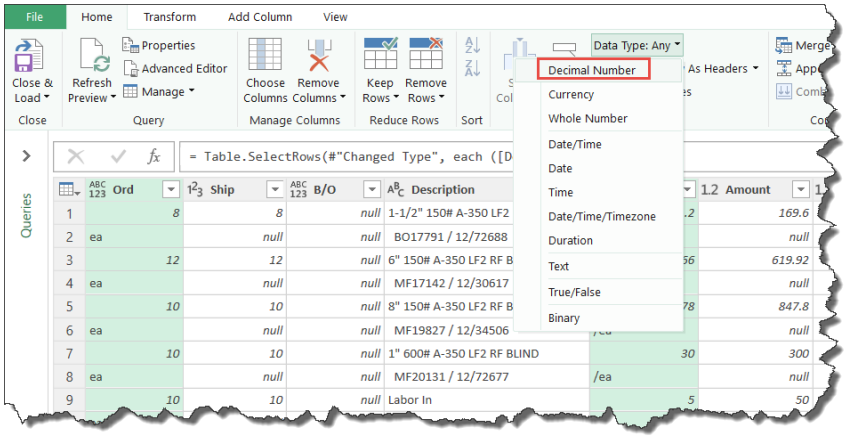


Figure 95. Changing the data type of two columns at the same time.

4. You'll notice that there are errors in some rows of those columns as there was a text string in those rows that couldn't be converted to a numeric value. We want to replace those errors with null values. To do that, select just one column, right click on it and select **"Replace Errors…"** and enter "null" as the value.

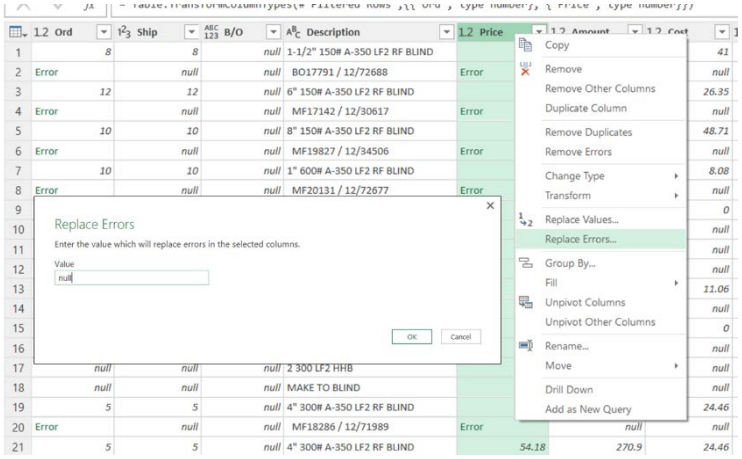


Figure 96. Replacing errors with null values.

5. Repeat the same operation for the "Ord¨ column.

6. Next, create an index column that will help us later with some operations. Click on the Add Columns tab on the ribbon, select "Add Index Column" and choose "From 1".

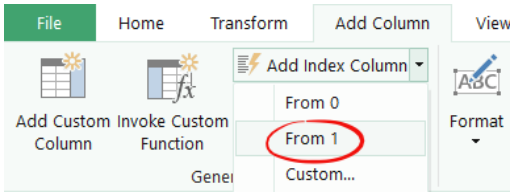


Figure 97. Adding an index column that starts at 1.

7. Based on the yellow Pattern, we can find the middle point of every set of rows for that pattern by locating the ”Llabor In” text string in the “Description” column. Create a new custom column that searches for the text string and give us TRUE/FALSE value to identify which rows have the text string. To do this, go to the Add Column tab and select ”Add Custom Column”. When Custom Column window opens, enter the formula in Figure 98.

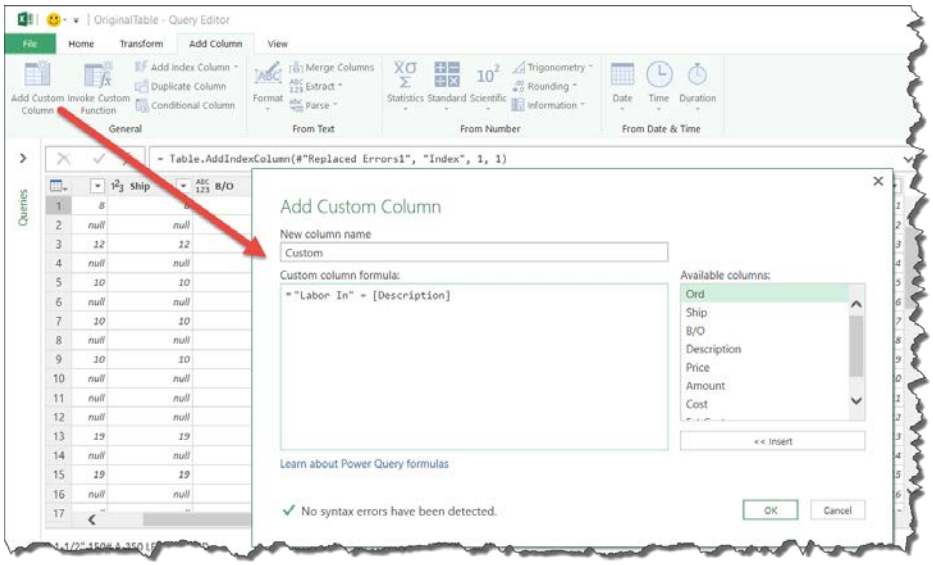


Figure 98. Creating a custom column that finds the rows where the “Labor In” text string appears.

Processing the Yellow Coded Rows

The following set of steps will show you how extract and transform the yellow coded rows.

8. Go to the left side of your screen and expand the Queries pane. From it, right click on our query and select the “Reference” option to create a new query that will reference our original query. Rename this new query as ”YellowPattern”.

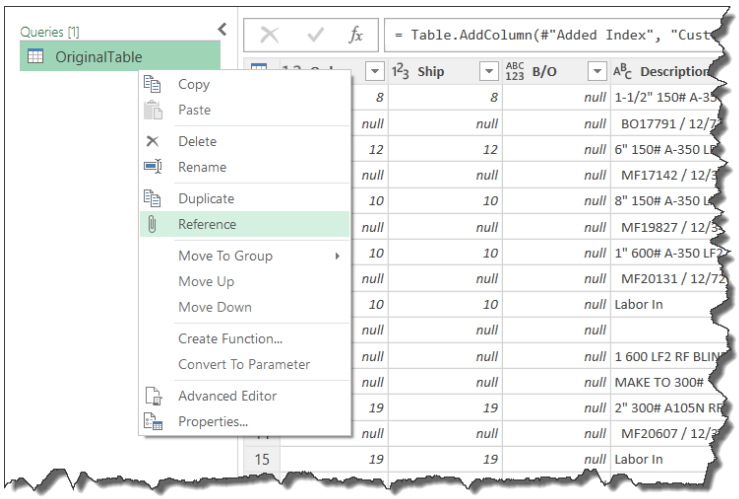


Figure 99. Creating a new query by referencing an existing one.

9. Next, filter the current table for the ”YellowPattern” query by the “Custom” column. Select ”TRUE” from the filter options and click “OK”.

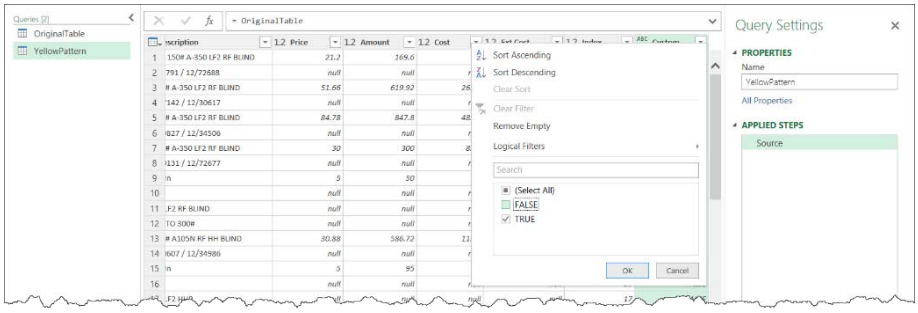


Figure 100. Filtering our current table by the custom column to display only the rows where the ”Labor In” text string appears.

10. Remove all columns from this table except the ”Index” column by right clicking on the “Index” column and selecting ”Remove Other Columns”.

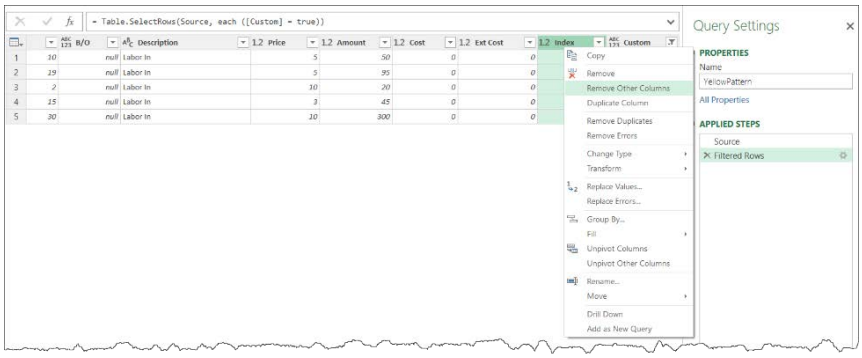


Figure 101. Removing columns so we see only the Index column, which gives us the specific position of the row on the original table.

11. Thanks to the previous operation, we now know the exact position of the ”Labor In” text string, but we still don’t know where the set of rows start or end. However, we have a fixed position for the ”Labor In” text. It appears on the fourth position of the set of rows, and the total number of rows for each set is six (initially it was eight, but we removed the null values so we ended up with six). Following our log, we can find the row just before our set of rows starts by simply subtracting 3 from the “Index” column. Create a new custom column and enter the formula as shown in Figure 102.

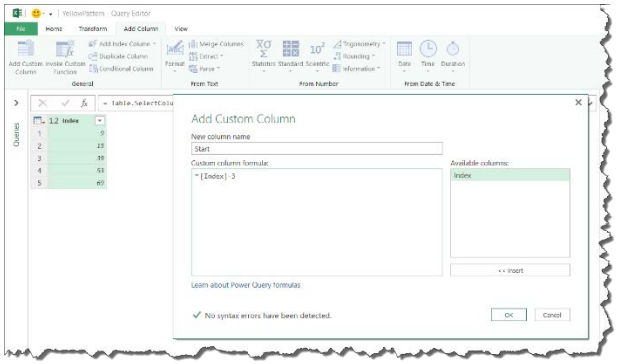


Figure 102. Creating the range for the set of rows of the YellowPattern query.

12. We now know where the set of rows start and that each set is six rows. With this information, we can use the **Table.Range** function to pull the yellow pattern data from our original query. Create a new custom column and enter the formula as shown in Figure 103.

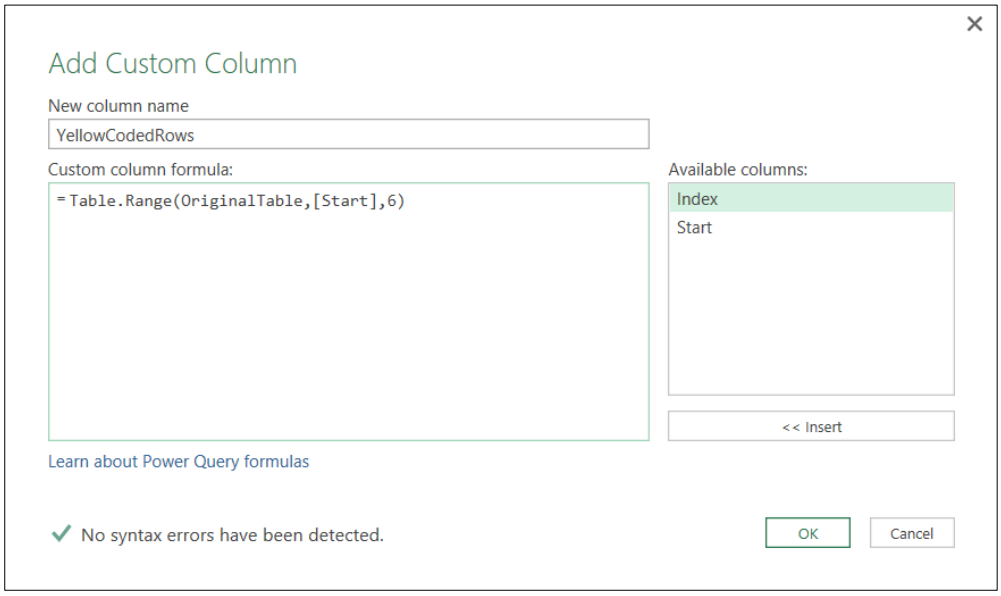


Figure 103. Extracting a specific range of rows from the OriginalTable query.

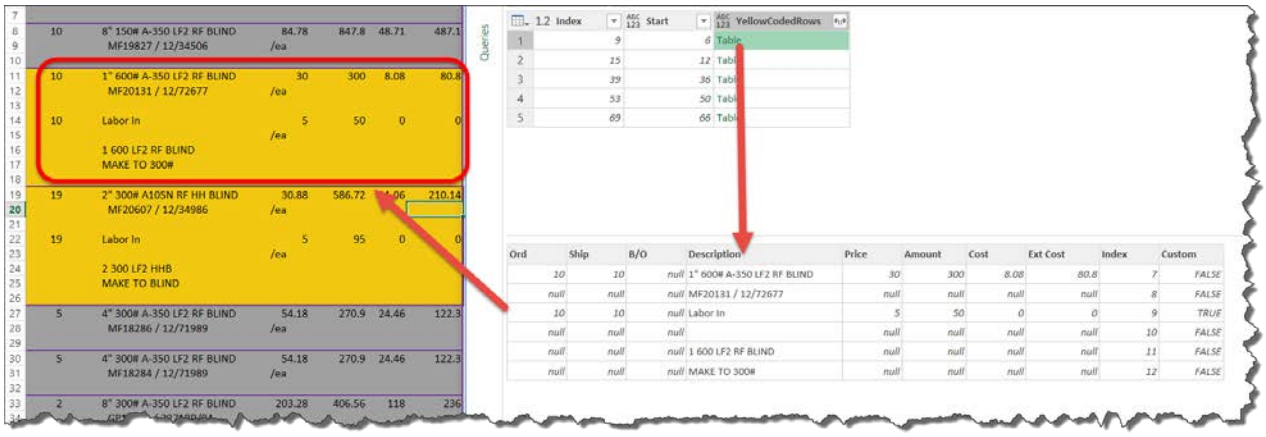


Figure 104. You can preview the data and compare it to the original table in Excel.

13. We now have a column with table values and each of those tables is basically the set of rows for each instance of the yellow pattern. Next, we combine all the text strings from the "Description" column by creating a new custom column and using the following formula:

```
Table.ToList( Table.Transpose(
     Table.SelectColumns([YellowCodedRows],{"Description"})
     ),
 Combiner.CombineTextByDelimiter(";",QuoteStyle.None) )
```

14. Clean up our query by removing the "Start" and "Index" columns.

15. Click on the double opposite arrow icon next to the new column to expand it. Expanding the list will give you a text string of the combination of all the text strings that Power Query found on the ”Description” column.

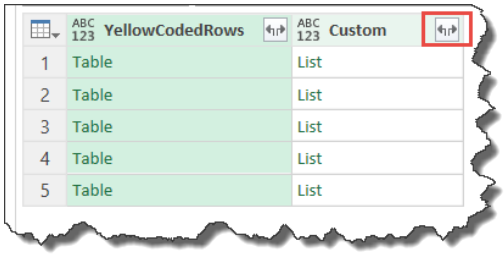


Figure 105. Click on the arrow icon to expand the values for this column.

16. Rename the column as ”Description” and change the data type to text.

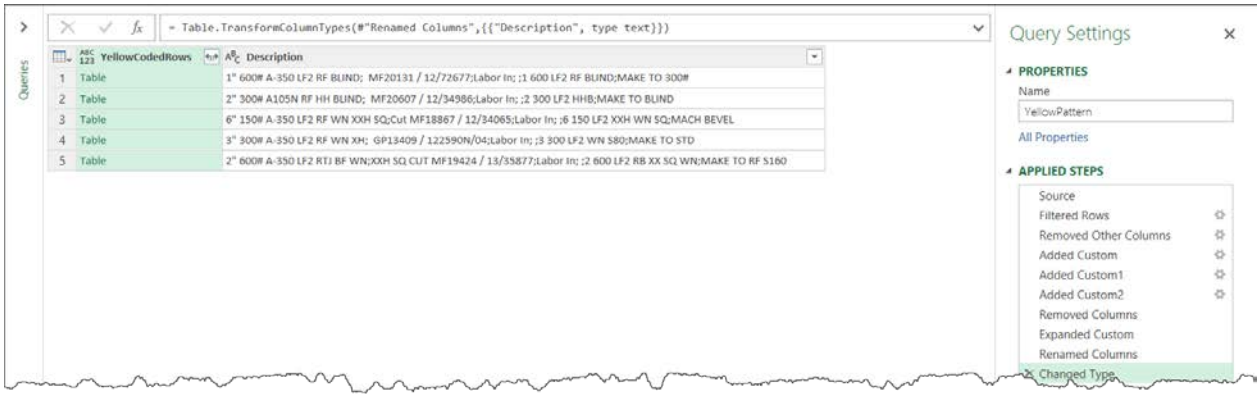


Figure 106. This is how our query looks so far.

17. Click on the double opposite arrow icon next to the ”YellowCodedRows” column header and select ”Aggregate” instead of ”Expand” option. Select the options as shown in Figure 107.

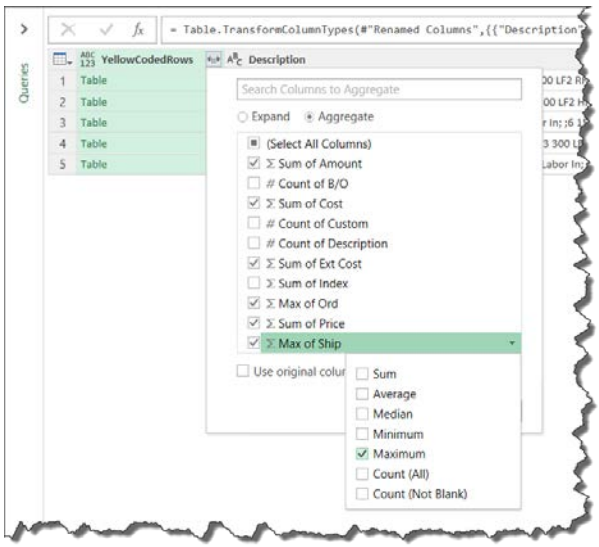
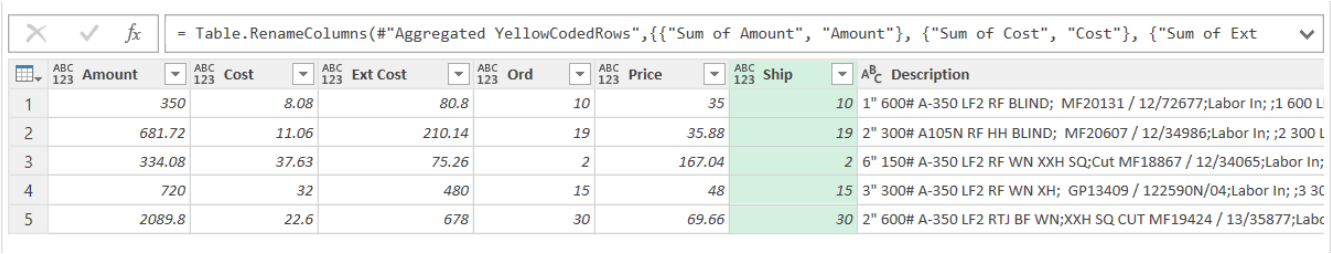


Figure 107. Aggregating data from a table can be extremely useful.

18. Several new columns have been created, so rename them as follow:
    a. Sum of Amount → Amount
    b. Sum of Cost → Cost
    c. Sum of Ext Cost → Ext Cost
    d. Max of Ord → Ord
    e. Sum of Price → Price
    f. Max of Ship → Ship



	Amount	Cost	Ext Cost	Ord	Price	Ship	Description
1	350	8.08	80.8	10	35	10	1" 600# A-350 LF2 RF BLIND; MF20131 / 12/72677;Labor In; ;1 600 L
2	681.72	11.06	210.14	19	35.88	19	2" 300# A105N RF HH BLIND; MF20607 / 12/34986;Labor In; ;2 300 L
3	334.08	37.63	75.26	2	167.04	2	6" 150# A-350 LF2 RF WN XXH SQ;Cut MF18867 / 12/34065;Labor In;
4	720	32	480	15	48	15	3" 300# A-350 LF2 RF WN XH; GP13409 / 122590N/04;Labor In; ;3 30
5	2089.8	22.6	678	30	69.66	30	2" 600# A-350 LF2 RTJ BF WN;XXH SQ CUT MF19424 / 13/35877;Labc

Figure 108. Our current table after renaming the columns.

19. Change the data type of all columns to numeric except the "Description" column which should be a decimal number.

Processing the Gray Coded Rows

The following set of steps will show you how extract and transform the gray coded rows.

20. Reference the original query again, but this time name this new query "GrayPattern".
21. Filter the custom column to only have "TRUE" values.
22. Remove all the columns except the "Index" column.

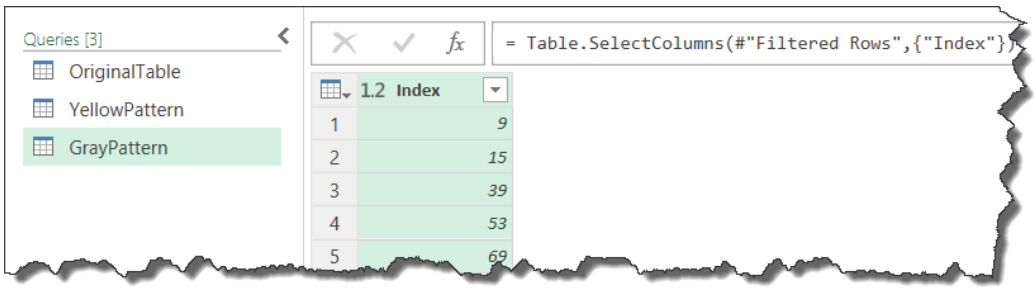


Figure 109. How our GrayPattern query looks at this point.

You'll notice that at this point we've been doing pretty much the same that we did for the yellow coded rows. What's next is something specific for the gray coded rows.

23. Add a custom column using the column name and formula shown in Figure 110.

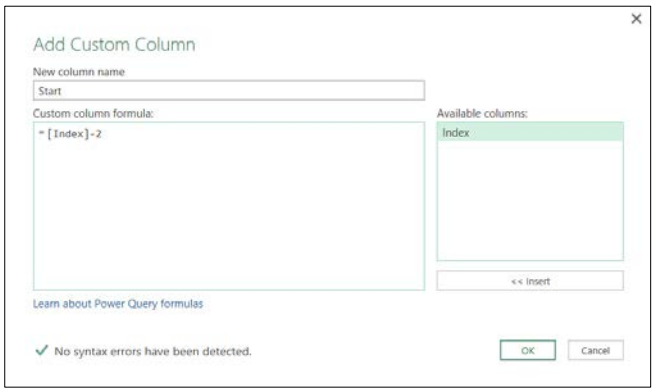


Figure 110. Creating a new column for the start of a new range of rows.

In the formula, we subtract 2 from the "Index" column to create this new column in order to exact row number where the first element of the "YellowPattern" occurs.

24. Add another custom column using the column name and formula shown in Figure 111.

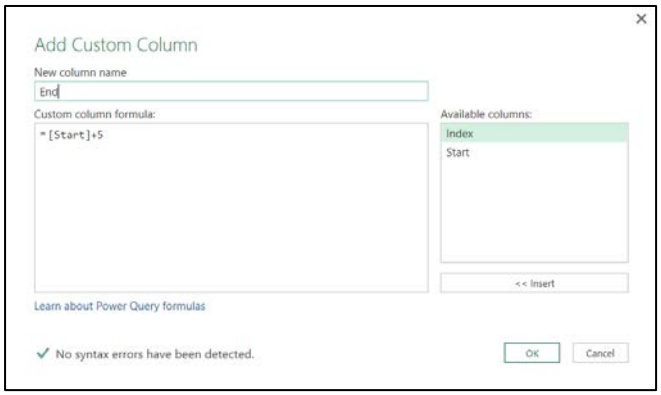


Figure 111. Creating a new column for the end of a new range of rows

In the formula, we add 5 to the "Start" column to create this new column in order to exact row number where the last element of the "YellowPattern" occurs.

25. Remove the "Index" column.

26. Create a new custom column that will output a sequential list of numbers from the values of the "Start" and "End" columns. Name this new custom column "IndexedRows" and use the following formula:

= {[Start]..[End]}

27. Remove the "Start" and "End" columns.

28. Expand the "IndexedRows" column.

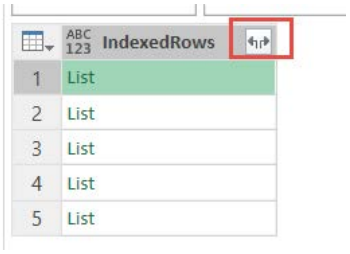


Figure 112. Click on the double opposite arrows icon to expand the data from that column.

29. After expanding the “IndexedRows” column, you’ll have a column with a bunch of numbers. These are, in a sense, the row numbers for all the elements of the “YellowPattern”. We want to merge this query with the ”OriginalTable” query using the ”IndexedRows” and the ”Index” column. However, you might be asking yourself, “Wouldn’t that give us the same rows?” It all depends on the type of join that we define. In this case, we want all the rows from the ”OriginalTable” that are not present in our current query, so we use the “Right” Anti join. Basically, we’re telling Power Query to give us all the rows from the “OriginalTable” query *except* the ones where the “Index” value is equal to those in our “IndexedRows” column.

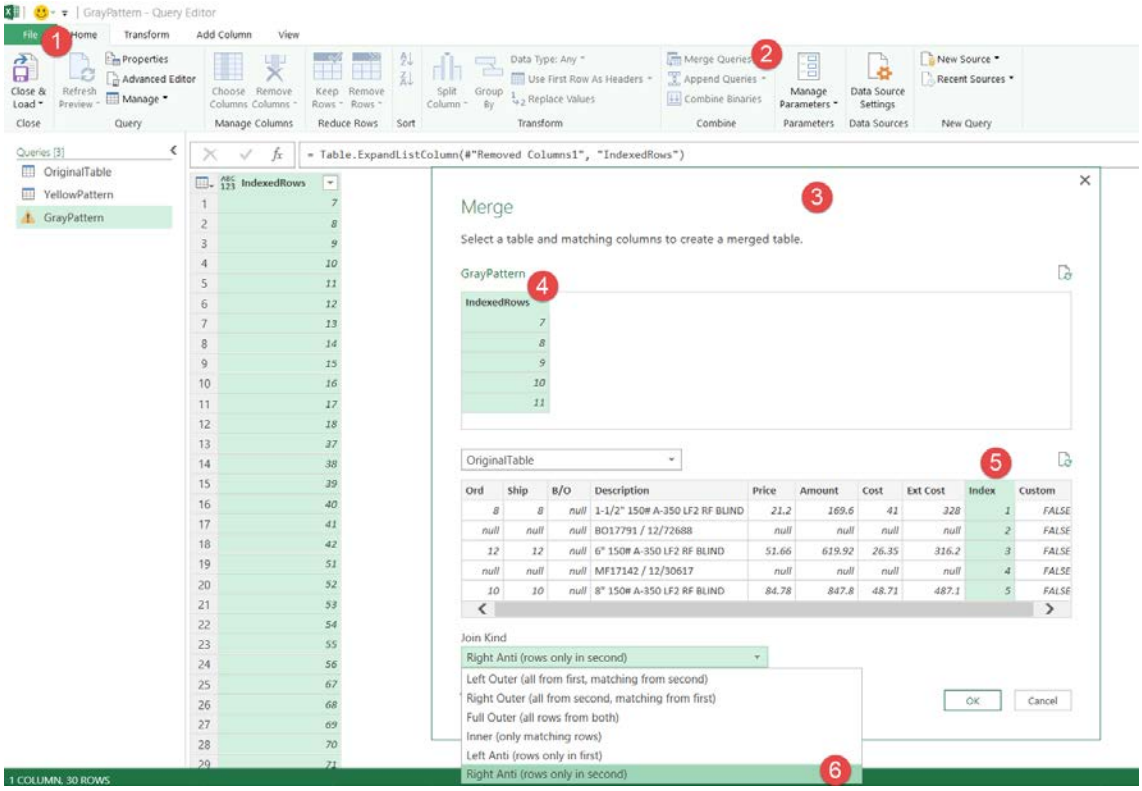


Figure 113. The steps of the merge operation.

Follow these simple steps to create the merge operation:

1. Go to the Home tab.
2. Find the ”Merge Queries” icon.
3. A new ”Merge” window will appear.
4. Select the “IndexedRows” column with a left click.
5. Select the “OriginalTable” query from the dropdown menu and select the ”Index” column.
6. Finally, from the “Join Kind” dropdown menu, select the “Right Anti” join and click the “OK” button.

30. Remove the “IndexedRows” column.

31. Expand the “NewColumn” as shown in Figure 114.

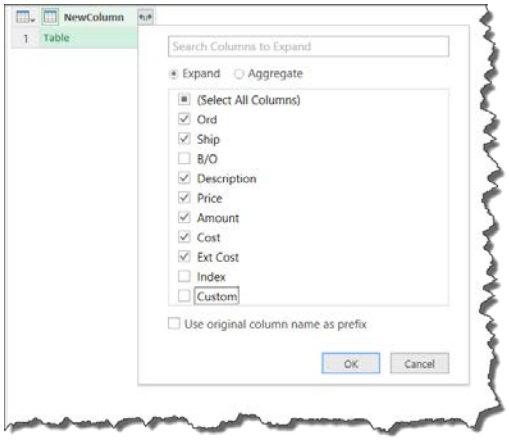


Figure 114. Expanding the 'NewColumn' created in the merge operation.

32. Add a new index column that starts "From 0".

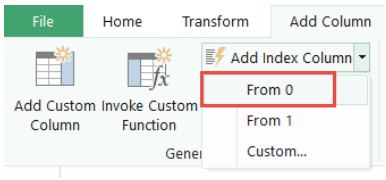


Figure 115. Adding a new index column that starts from the number 0.

33. Next, insert a new modulo column by selecting the "Index" column, going to the Add Column tab, selecting "Standard" in the "From Number" group and finally selecting the "Modulo" option. Once you get the Modulo window, input the value as 2 and click "OK".

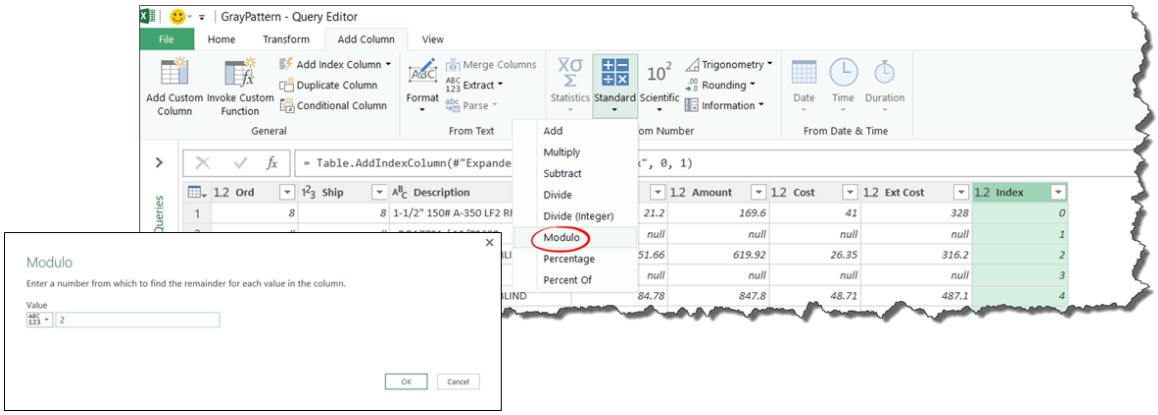


Figure 116. Adding a new modulo column.

We use the number 2 because our data comes in sequence of two. Every two rows you get a new record for the "GrayPattern". If the pattern was for every four rows, entered 4 in the Modulo window.

34. We usually unpivot with Power Query, but this time we want to pivot a column. First, select the "Inserted Modulo" column and then select the "Pivot Column" option from the Transform tab. Inside the Pivot Column window, select the "Description" column for the "Values Column" field. In the advanced options section, choose "Don't Aggregate" and click "OK".

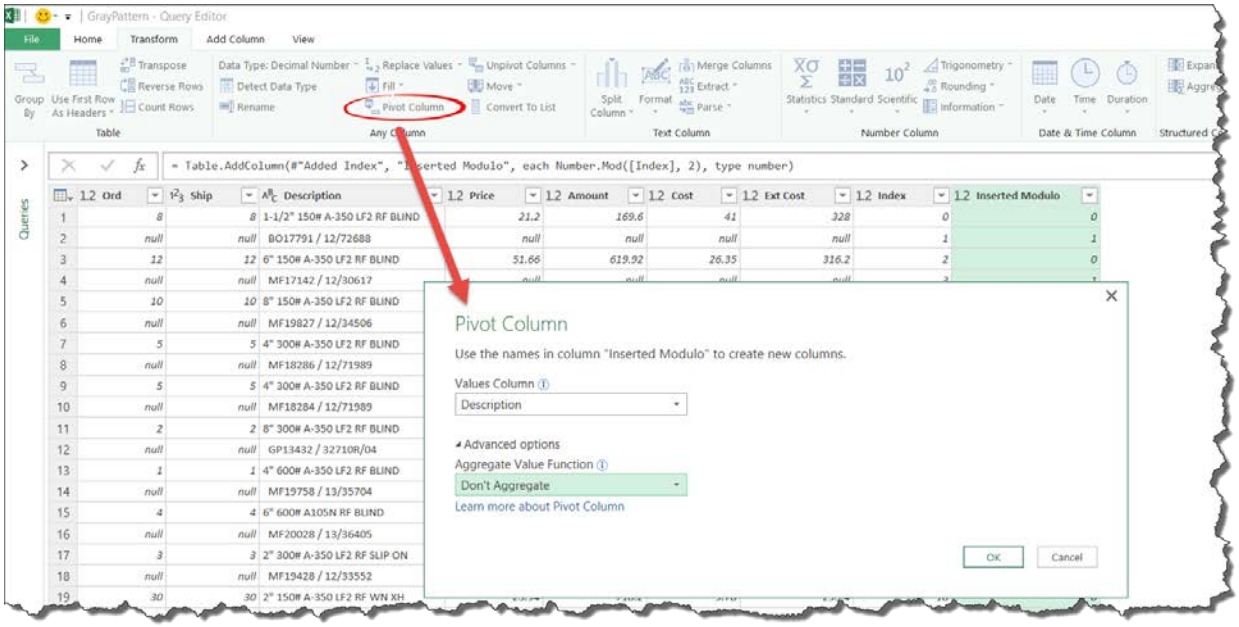


Figure 117. Pivoting a column in Power Query without any aggregations.

35. Your data may look weird and all scrambled at first, but don't panic. Sort the rows in ascending order using the "Index" Column.

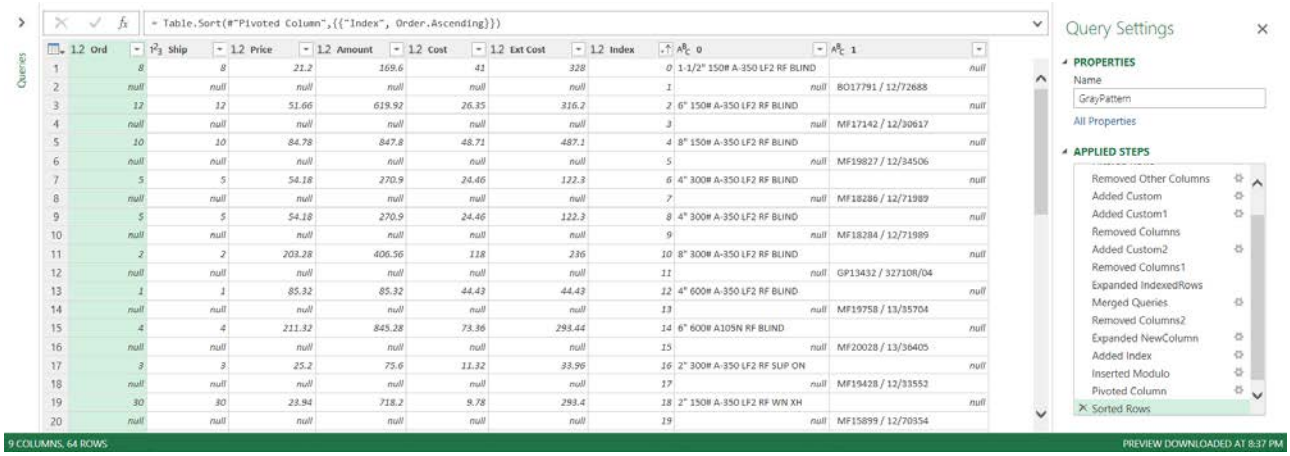


Figure 118. Our table after sorting.

36. Select the column named "1" and go to the Transform tab. Inside this tab you'll find an option called "Fill" and select "Fill Up" from the dropdown menu.

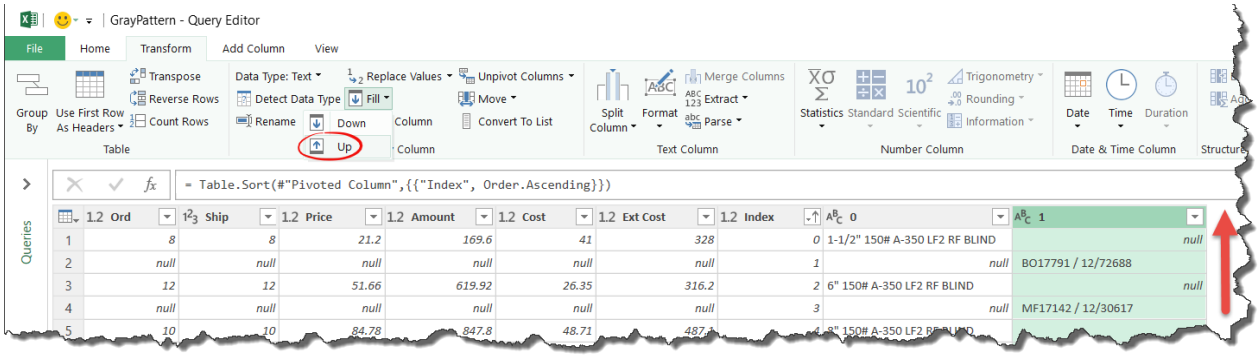


Figure 119. The "fill up" operation will replace any null values with the next value that it finds on the column.

37. Filter the "Ord" column to get rid of the null values.

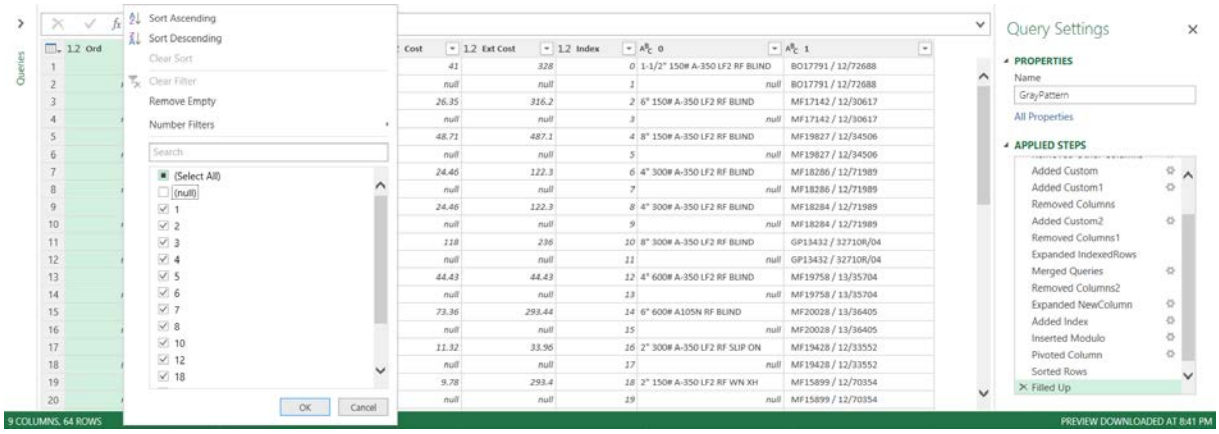


Figure 120. Filtering to get rid of the null values.

38. Remove the "Index" column as we no longer need it.

39. Select the columns both "0" and "1". Right click on either one and select "Merge Columns". In the Merge Columns window, use the semicolon separator and input the name of the column ("Description").

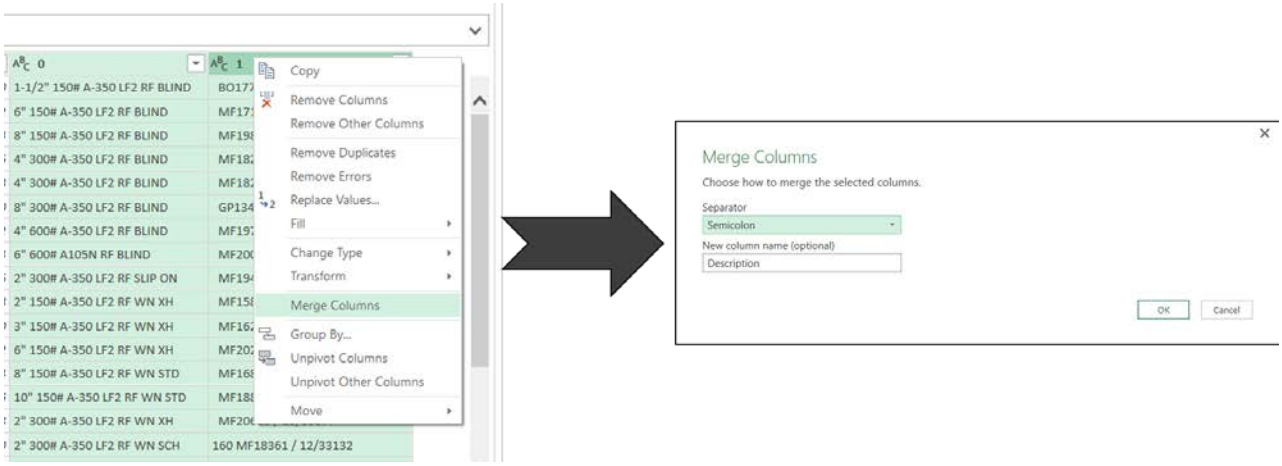


Figure 121. Merging two text columns.

Appending the Queries

The following set of steps will show you how to combine the YellowPattern and the GrayPattern queries into one query.

40. Return to the Home tab and click the "Append Queries" button. From the dropdown menu, select "Append Queries as New".

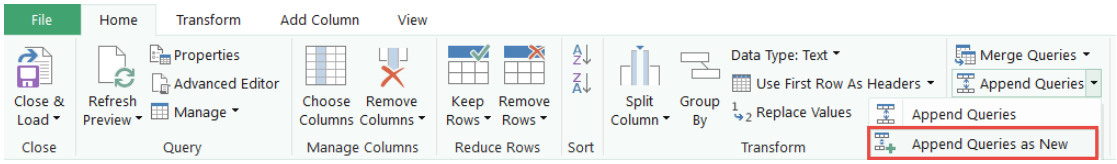


Figure 122. Selecting "Append Queries as New".

41. In the Append window, select the queries that you'd like to combine.

Figure 123. Appending the YellowPattern query to the GrayPattern query.

42. After clicking "OK", a new query will be created, andwe suggest that you rename this query as "Output". This query has 37 rows exactly in the format that we needed.

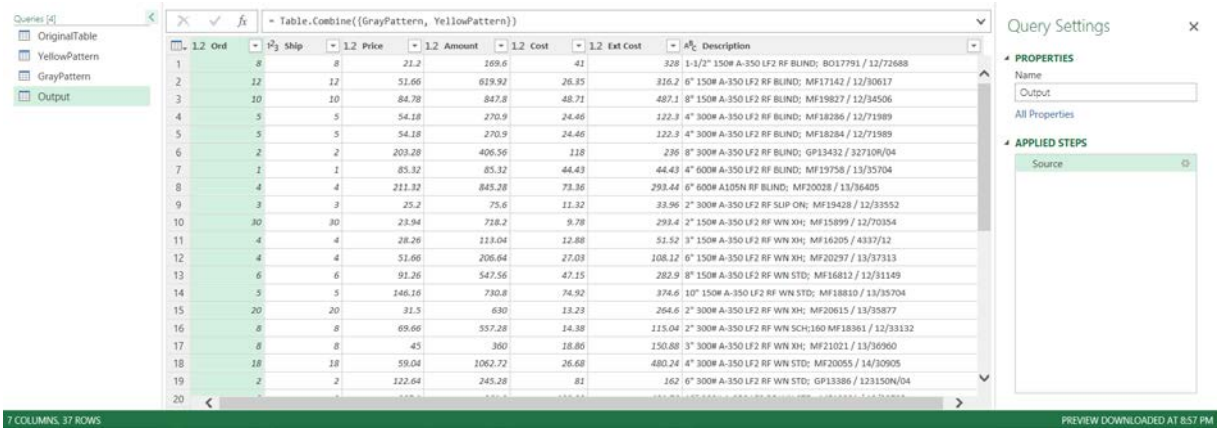


Figure 124. Our final query.

43. Finally, you can click on "Close & Load" and load this new Output query to a new table or your data model.

Case Summary: Handling Multiple Data Patterns in a Table

Topics covered

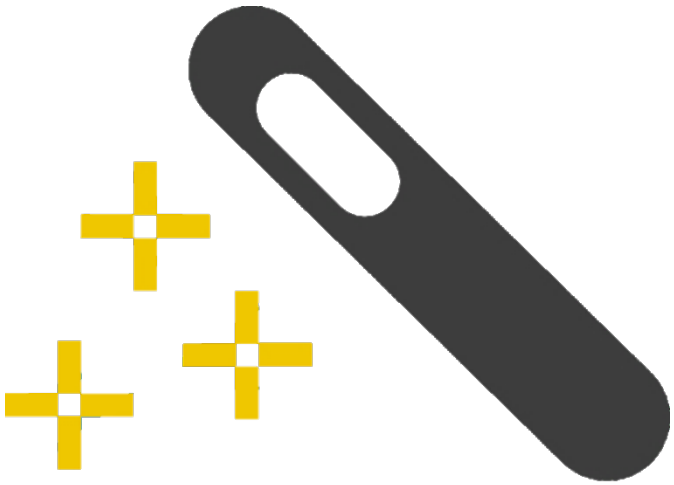
- Filtering columns
- Replacing errors
- Adding an index column
- Using the Table.Range function
- Combining multiple text strings from a column
- Aggregating data from a table
- Creating a list of sequential numbers
- Merging using the Right.Anti join
- Using a modulo column
- Pivoting a column
- Appending queries

Recommendations

- **What if I have more than two patterns in my data?**
    - The key is creating the helper columns needed to differentiate the rows for each segment. If you have the logic behind each of those patterns, you only need to translate that logic into the Power Query language. Then you can find either the first, middle or the last row of the set rows for each pattern, create the sequence of numbers,use the Table.Range function, or the Right.Anti join, and then combine the queries.
- **My data is all scrambled up and my patterns are not sequential (it jumps between rows). How can I tackle that type of scenario?**
    - Our best suggestion would be to **sort and arrange** the data as needed. There are other ways to create segments as shown in Chapter 2, using the "Group By" feature and the "All Rows" operation. This automatically creates segments of data based on the shared values between rows of a table. There are other, more advanced ways to create segments, but it is usually a case-by-case scenario. If you have any specific scenario that you'd like to share with us, visit this book's feedback webpage here and leave a comment.

Personal comments from the authors

We tackle even more advanced scenarios in the follow-up Q&A session of our live, online workshops. It is fascinating to see how Power Query can solve even the most complex scenarios in a matter of minutes and with an amazing performance. If you wish to learn more, be sure to join a future online Power Query workshop. As part of our newsletter list, you will receive an exclusive coupon code to join us in the next session of our workshop.



Become a Data Wizard