Get & Transform Power Query Workshop for Audit, Tax and Advisory Professionals

An Introduction

Kathryn Newitt & Liam Bastick

11/6/24

Contents

Introduction

This document supports the 'Get & Transform Power Query Workshop for Audit, Tax and Advisory Professionals'. In particular, this document focuses on the artist formerly known as Power Query, now known as Get & Transform.

This session focuses on said Power Query in its Excel form, where I look at the ease with which I can extract, clean and manipulate data from a variety of sources. It is known as an "Extract, Transform and Load" tool – or "ETL" for short.

Power Query is also available in Power BI. Changes and improvements generally appear in Power BI before Excel, and there are more data connectors available. Since the layout of Power BI is often updated, you may find that some screenshots don't match your screen exactly, but all the functionality described here will be available.

Data is one of the most valuable assets of any business, and Power Query is an invaluable tool to turn your data into information. Let's get started.

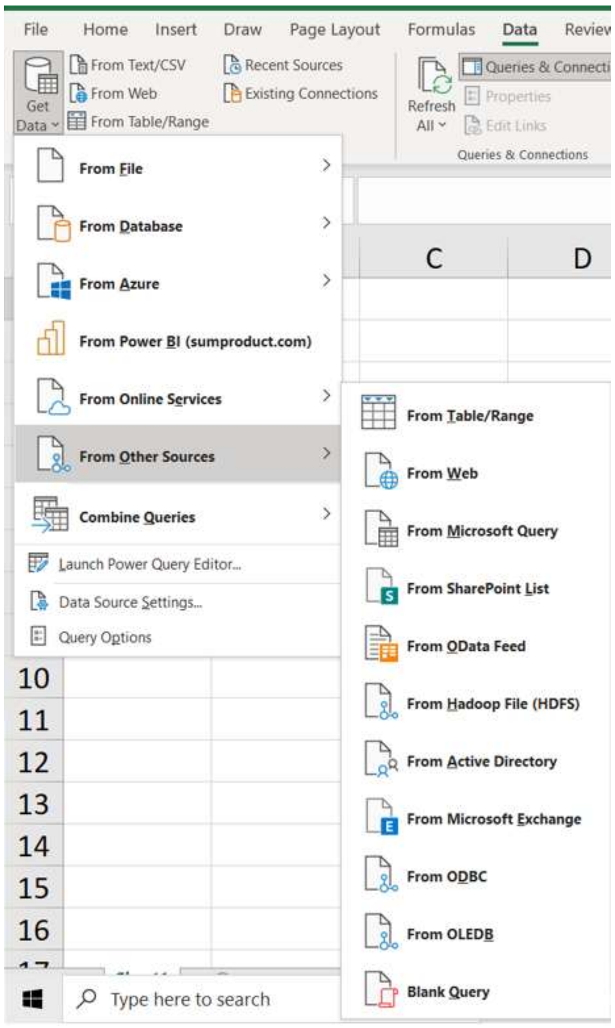
Power Query: Getting & Transforming Your Data

Regular users of Excel are well aware of the possibilities to manipulate and present data that has come from a variety of sources. Getting that data ready for each solution can be time consuming: complex solutions can require VBA or SQL expertise, and simple solutions often involve repetitive tasks before the data is ready to be uploaded to PivotTables.

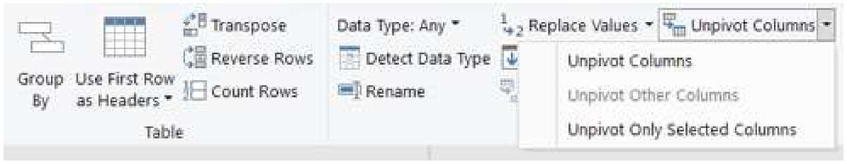
Enter Power Query (alternatively known now as "Get & Transform"), which has been designed for Excel users to enable the rapid uploading and cleaning of data without needing to turn (in)to VBA and SQL experts.

Power Query is a free add-in ETL (**E**xtract, **T**ransform and **L**oad) tool for users of Excel 2010 (Professional Plus with Software Assurance version only) and 2013, where it has its own tab on the Ribbon, and is now fully integrated into Excel 2016 onwards and Office 365, where it may be found on the data tab under 'Get & Transform'.

As shown in Excel 365, Power Query is able to upload from many sources:



Having extracted the data from the appropriate source (more on the details of this process later), the transformation can take place – cleaning away any unrequired detail, merging where appropriate (all without the erstwhile **VLOOKUP**), and enhancing by calculating and adding new columns. There is even a button to unpivot data!



Whilst the majority of transformations can be done with no formulae or coding, each step is stored in **M**, the language behind Power Query. For the more ambitious user, **M** may be used to refine and build on what Power Query creates automatically.

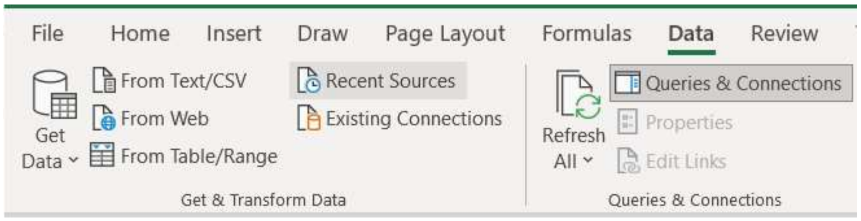
The final step for the ETL tool is to load - to Excel tables, Power Pivot, Power BI – or else just store a query for use by other queries, *i.e.* a building block that may be re-used. Since everything is recorded in **M**, you may simply reload to the same destination – simply refresh.

Please be aware that Microsoft often releases updates, usually on a monthly basis – try searching 'updates for Get & Transform in Excel' to see the latest updates.

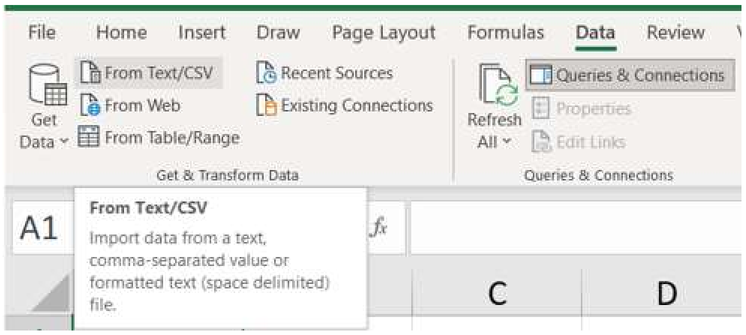
Getting Started: Importing Data

The best way to get to grips with a tool like Power Query is to start with a simple task. Excel users may often need to take data from **CSV** (comma separated values) files and transform it ready for analysis. Power Query has been designed to assist with this, so let's see how easy it can be.

Starting with a new workbook, I locate the 'Get & Transform Data' section on the Data tab:



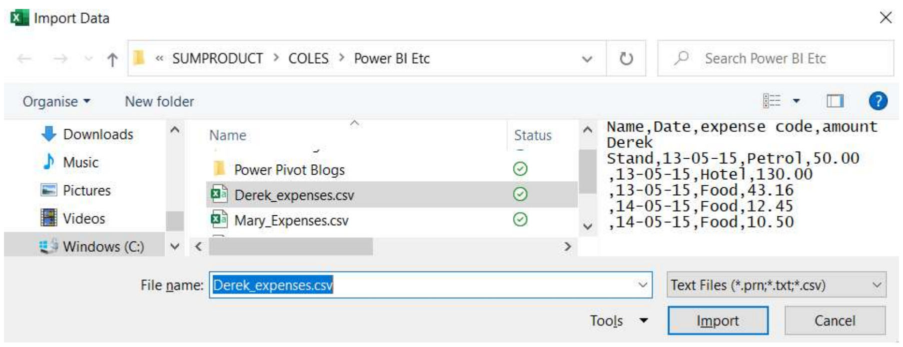
For this example, I am using the 'From File' option, and choosing 'From Text/CSV':



From Text/CSV

Import data from a text, comma-separated value or formatted text (space delimited) file.

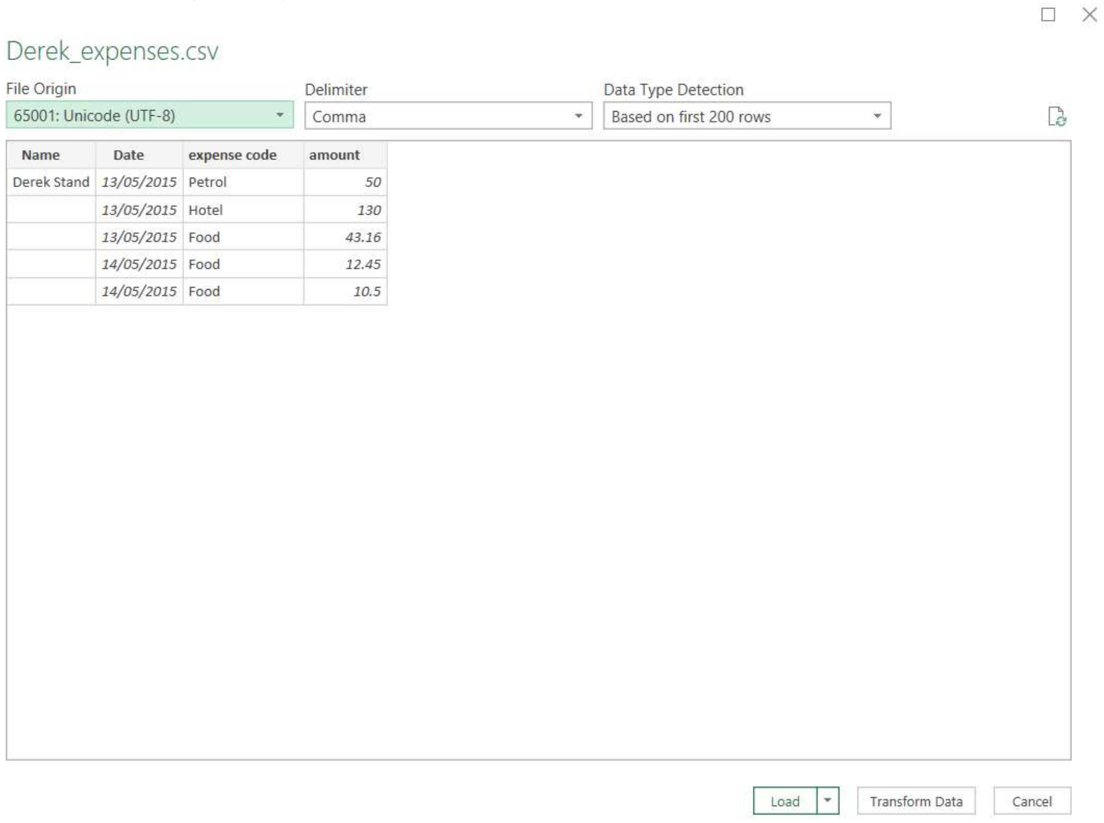
I browse to the location of a simple expense **CSV** file.



File name: Derek_expenses.csv
Text Files (*.prn;*.txt;*.csv)

```
Name,Date,expense code,amount
Derek
Stand,13-05-15,Petrol,50.00
,13-05-15,Hotel,130.00
,13-05-15,Food,43.16
,14-05-15,Food,12.45
,14-05-15,Food,10.50
```

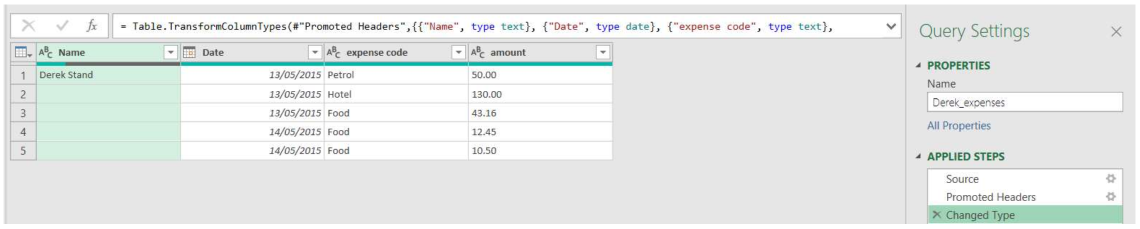
I select the file using the Import button.



File Origin: 65001: Unicode (UTF-8)
Delimiter: Comma
Data Type Detection: Based on first 200 rows

Name	Date	expense code	amount
Derek Stand	13/05/2015	Petrol	50
	13/05/2015	Hotel	130
	13/05/2015	Food	43.16
	14/05/2015	Food	12.45
	14/05/2015	Food	10.5

This screen allows me to specify properties of the **CSV** file that I am importing from. The ‘Data Type Detection’ allows me to choose whether the algorithms Power Query uses to determine the data types are used, and whether to read the whole dataset. The default is to base the types on the first 200 rows, which is enough for this short example. Whilst I have the option to load from this point, I’d always recommend the ‘Transform Data’ option to avoid any nasty surprises.



```
= Table.TransformColumnTypes(#"Promoted Headers",{{"Name", type text}, {"Date", type date}, {"expense code", type text},
```

	Name	Date	expense code	amount
1	Derek Stand	13/05/2015	Petrol	50.00
2		13/05/2015	Hotel	130.00
3		13/05/2015	Food	43.16
4		14/05/2015	Food	12.45
5		14/05/2015	Food	10.50

Getting Started: Transforming Data

The Power Query Editor screen shows the table of data as it will be uploaded. In the ‘Query Settings’, the name of the query appears under PROPERTIES and under ‘APPLIED STEPS’, where I can see the steps that Power Query has taken automatically. The source has been identified, the header column names have been assumed and the ‘Changed Type’ step sets the detected type for each column. The **M** code for ‘Changed Type’ is:

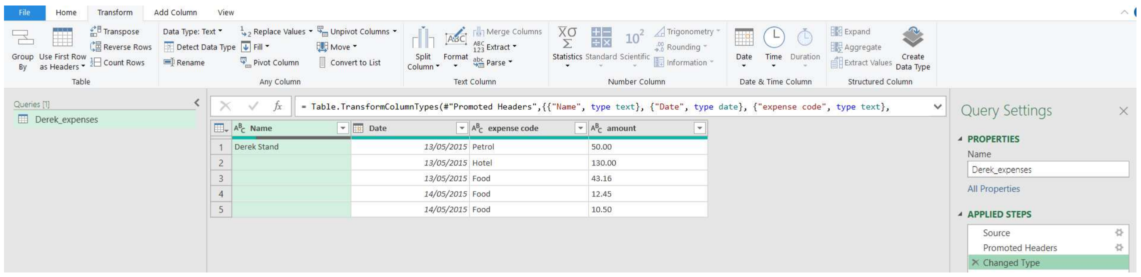
=Table.TransformColumnTypes(#"Promoted Headers",{{"Name", type text}, {"Date", type date}, {"expense code", type text}, {"amount", type number}})

It is essentially a list of column names and their assigned data types based upon the data that Power Query has analysed:

- **Name** is type text
- **Date** is type date
- **expense code** is type text
- **amount** is number.

In this case, all the assumptions look good so I may accept Power Query’s automatic assignment of data types.

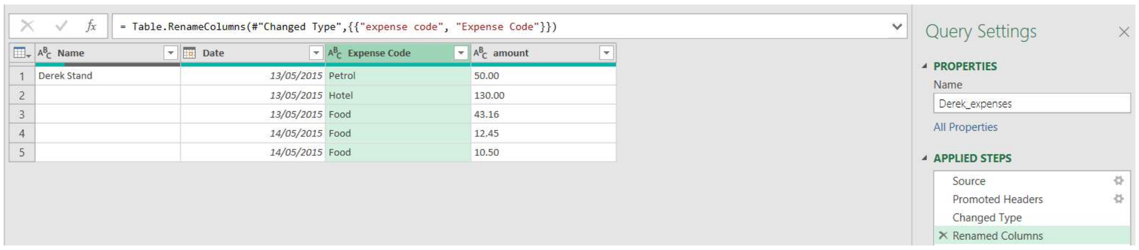
I can make any changes to the data, such as changing column names or removing columns. Selecting a column and viewing the ‘Transform’ tab reveals buttons for many of the usual transformations.



```
= Table.TransformColumnTypes(#"Promoted Headers",{{"Name", type text}, {"Date", type date}, {"expense code", type text},
```

	Name	Date	expense code	amount
1	Derek Stand	13/05/2015	Petrol	50.00
2		13/05/2015	Hotel	130.00
3		13/05/2015	Food	43.16
4		14/05/2015	Food	12.45
5		14/05/2015	Food	10.50

I can double-click on **expense code** and rename it to **Expense Code.** This creates a new step 'Renamed Columns':



```
= Table.RenameColumns(#"Changed Type",{{"expense code", "Expense Code"}})
```

	Name	Date	Expense Code	amount
1	Derek Stand	13/05/2015	Petrol	50.00
2		13/05/2015	Hotel	130.00
3		13/05/2015	Food	43.16
4		14/05/2015	Food	12.45
5		14/05/2015	Food	10.50

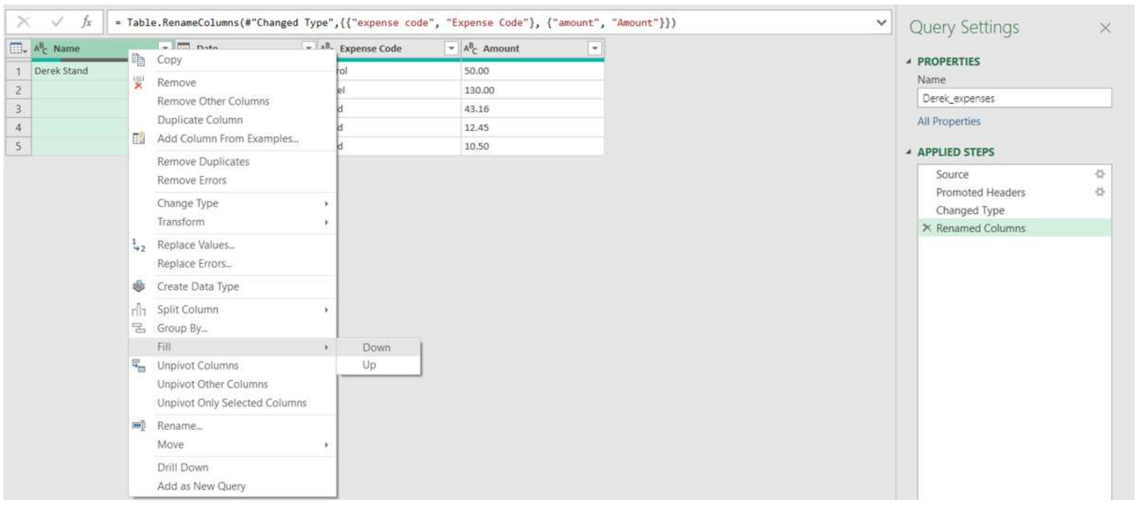
The **M** code is:

= Table.RenameColumns(#"Changed Type",{{"expense code", "Expense Code"}})

If I rename **amount** to **Amount**, Power Query recognises that I am doing and adds the change to the existing 'Changed Type' step:

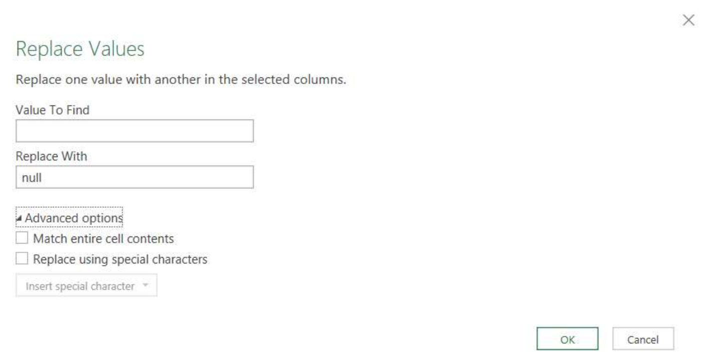
= Table.RenameColumns(#"Changed Type",{{"expense code", "Expense Code"}, {"amount", "Amount"}})

A more complex step is to make the **Name** data appear on each row. There is a 'Fill Down' option available if I right-click on the Name column, but it doesn't seem to do anything!



```
= Table.RenameColumns(#"Changed Type",{{"expense code", "Expense Code"}, {"amount", "Amount"}})
```

In order to use the 'Fill Down' functionality, the 'empty' **Name** cells that are to be populated must be set to *null*, but currently they are blank. I need to replace the blanks with *null* so that I can fill down. The option to 'Replace Values…' is also on the right-click menu. The ellipsis (…) indicates that a window will appear:


Replace Values

Replace one value with another in the selected columns.

Value To Find:

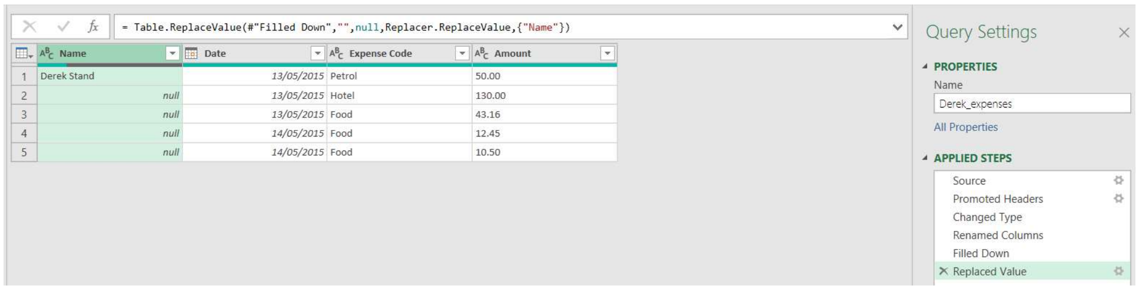
Replace With: null

Advanced options

☐ Match entire cell contents

☐ Replace using special characters

I opt to replace blanks (which I just leave empty) with *null*. I don't need the 'Advanced options' here.


```
= Table.ReplaceValue(#"Filled Down","",null,Replacer.ReplaceValue,{"Name"})
```

	Name	Date	Expense Code	Amount
1	Derek Stand	13/05/2015	Petrol	50.00
2	null	13/05/2015	Hotel	130.00
3	null	13/05/2015	Food	43.16
4	null	14/05/2015	Food	12.45
5	null	14/05/2015	Food	10.50

Query Settings

PROPERTIES

Name: Derek_expenses

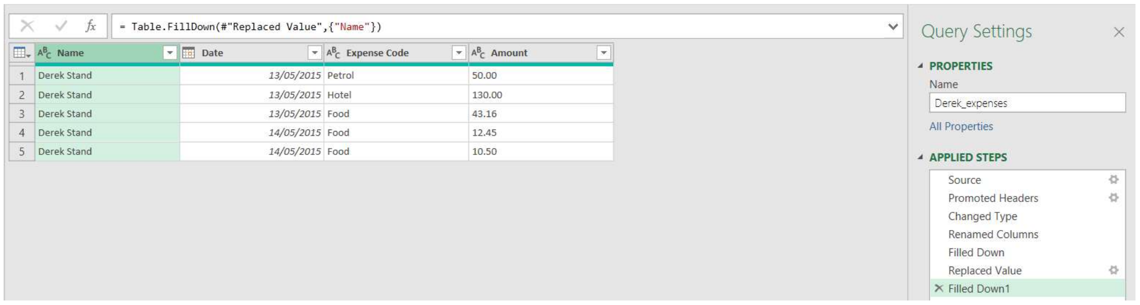
All Properties

APPLIED STEPS

- Source
- Promoted Headers
- Changed Type
- Renamed Columns
- Filled Down
- Replaced Value

The 'Filled Down' step is from my failed attempt before I set the values to *null*. I will delete this shortly, but first I 'Fill Down' again.

The steps are generated in the right-hand pane, and we are ready to load to Excel using the 'Close and Load' button on the 'File' tab. The uploaded table is shown below, and the Workbook query window displays the query that generated the data, so that it can be updated and / or refreshed as required. The 'TABLE TOOLS' tab opens automatically ready for use in the Excel workbook:


```
= Table.FillDown(#"Replaced Value",{"Name"})
```

	Name	Date	Expense Code	Amount
1	Derek Stand	13/05/2015	Petrol	50.00
2	Derek Stand	13/05/2015	Hotel	130.00
3	Derek Stand	13/05/2015	Food	43.16
4	Derek Stand	14/05/2015	Food	12.45
5	Derek Stand	14/05/2015	Food	10.50

Query Settings

PROPERTIES

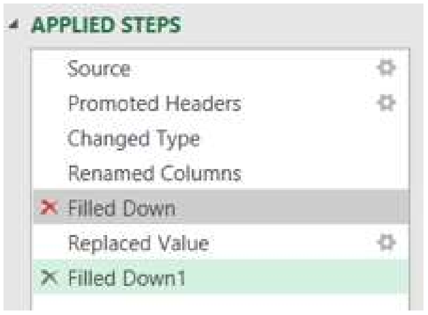
Name: Derek_expenses

All Properties

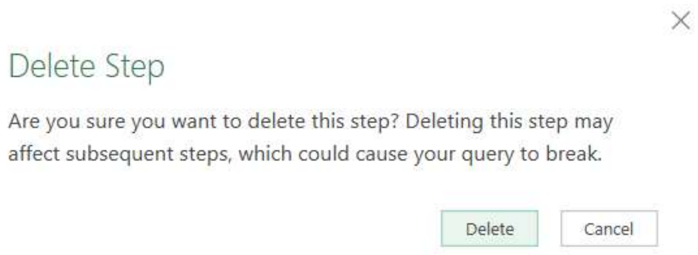
APPLIED STEPS

- Source
- Promoted Headers
- Changed Type
- Renamed Columns
- Filled Down
- Replaced Value
- Filled Down1

This time, the ‘Fill Down’ is successful. I can delete the first ‘Filled Down’ step now. I can select or hover over ‘Filled Down’, and click on the delete cross that appears next to it:



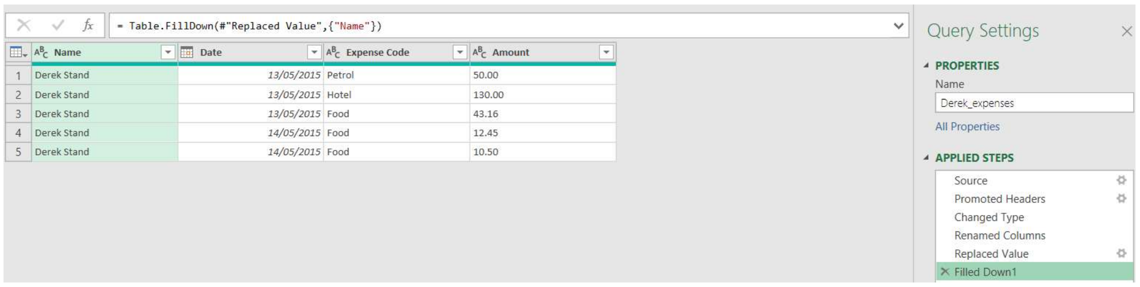
I am prompted to confirm my action:



Delete Step

Are you sure you want to delete this step? Deleting this step may affect subsequent steps, which could cause your query to break.

Power Query has been improved over time, and copes well with deleting intermediate steps now. Previous versions required me to manually change references to previous steps, but this is now automated. If I am at all concerned, I can take a copy of my query before I proceed (more on this later). In this case, I am happy to proceed:



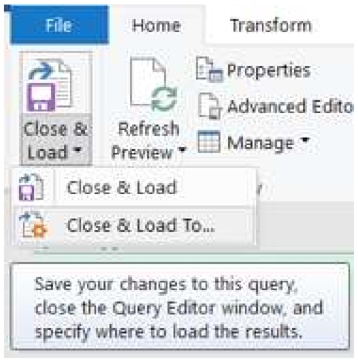
```
= Table.FillDown(#"Replaced Value",{"Name"})
```

	Name	Date	Expense Code	Amount
1	Derek Stand	13/05/2015	Petrol	50.00
2	Derek Stand	13/05/2015	Hotel	130.00
3	Derek Stand	13/05/2015	Food	43.16
4	Derek Stand	14/05/2015	Food	12.45
5	Derek Stand	14/05/2015	Food	10.50

Name: Derek_expenses

Getting Started: Loading Data

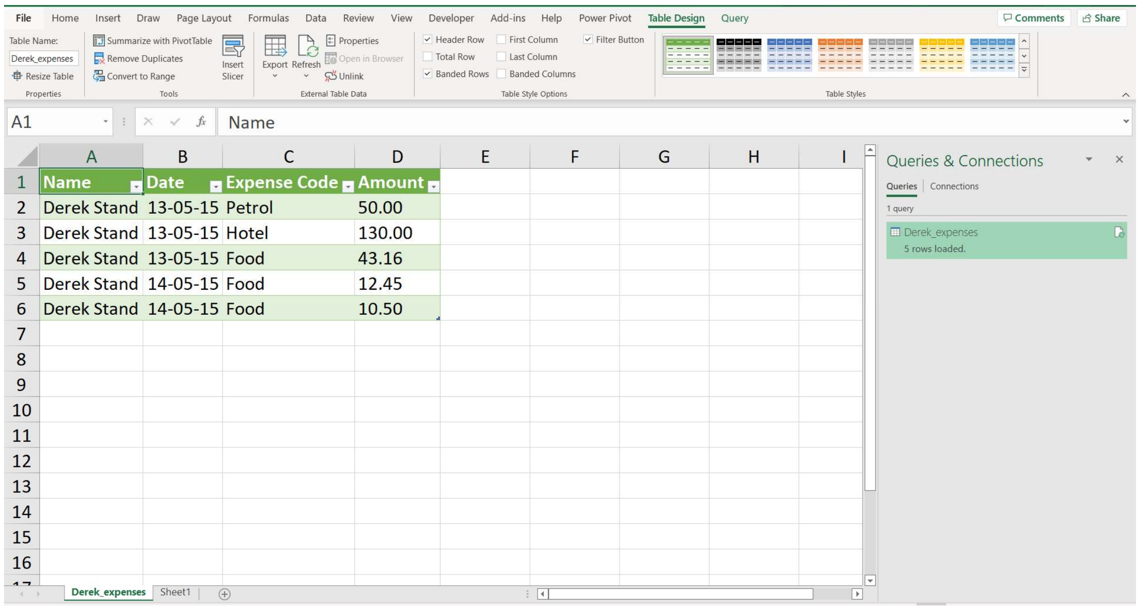
Now I am happy with the data, I can use the ‘Close & Load’ option on the Home tab to load the data to Excel:



Save your changes to this query, close the Query Editor window, and specify where to load the results.

Choosing ‘Close & Load’ will load my query into Excel as a Table in a new sheet. As the help message implies, choosing ‘Close & Load To…’ will allow me to add further specifications into a dialog. More on this later.

I am happy to load the data into a new sheet:



Name	Date	Expense Code	Amount
Derek Stand	13-05-15	Petrol	50.00
Derek Stand	13-05-15	Hotel	130.00
Derek Stand	13-05-15	Food	43.16
Derek Stand	14-05-15	Food	12.45
Derek Stand	14-05-15	Food	10.50

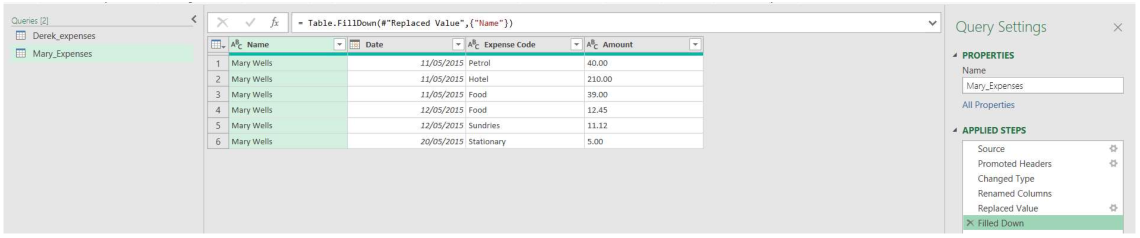
The sheet has the same name as the query, and my Table appears, and is given the same name as the query (with any spaces converted to underscores). The query that created the table appears in the ‘Queries and Connections’ pane. Refreshing this query ensures that it is extracting, transforming and loading the latest data source. The original **CSV** file is not changed by any of the steps I have created.

Let’s look at appending more data…

Appending Files

Whilst it is possible to extract multiple **CSV** (comma separated values) files at the same time (more on this later), imagine a scenario where similar files appear at intervals and need to be added to a table.

In the same workbook, I repeat the process to extract data from a **CSV** file, and use the same steps to transform the data. This time I have extracted Mary's data:



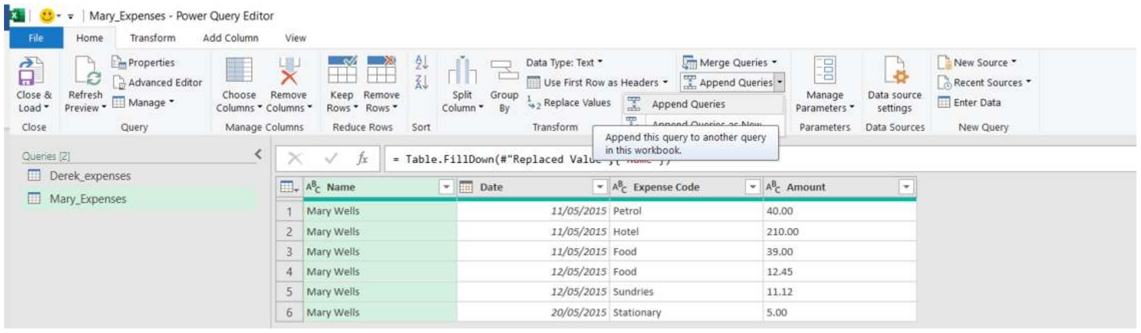
```
= Table.FillDown(#"Replaced Value",{"Name"})
```

	Name	Date	Expense Code	Amount
1	Mary Wells	11/05/2015	Petrol	40.00
2	Mary Wells	11/05/2015	Hotel	210.00
3	Mary Wells	11/05/2015	Food	39.00
4	Mary Wells	12/05/2015	Food	12.45
5	Mary Wells	12/05/2015	Sundries	11.12
6	Mary Wells	20/05/2015	Stationary	5.00

Name: Mary_Expenses

Applied steps:
- Source
- Promoted Headers
- Changed Type
- Renamed Columns
- Replaced Value
- Filled Down

I want to add this data to Derek's data. I can do this by appending this query to Derek's query. On the Home tab there is a Combine section where there is an 'Append Queries' dropdown:

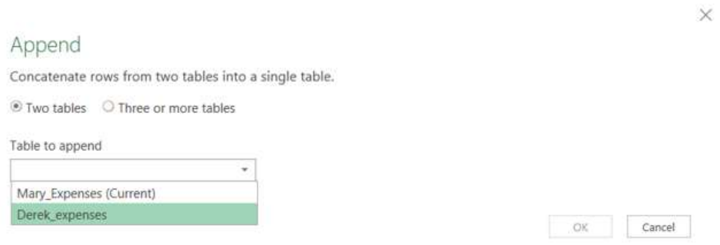


Append this query to another query in this workbook.

```
= Table.FillDown(#"Replaced Valu
```

	Name	Date	Expense Code	Amount
1	Mary Wells	11/05/2015	Petrol	40.00
2	Mary Wells	11/05/2015	Hotel	210.00
3	Mary Wells	11/05/2015	Food	39.00
4	Mary Wells	12/05/2015	Food	12.45
5	Mary Wells	12/05/2015	Sundries	11.12
6	Mary Wells	20/05/2015	Stationary	5.00

As the dropdown shows, I can either 'Append Queries' where I 'Append this query to another query in this workbook' or I can 'Append Queries as New'. Since the expense files are always required to be uploaded together, the 'Append Queries' option is fine here.



Append

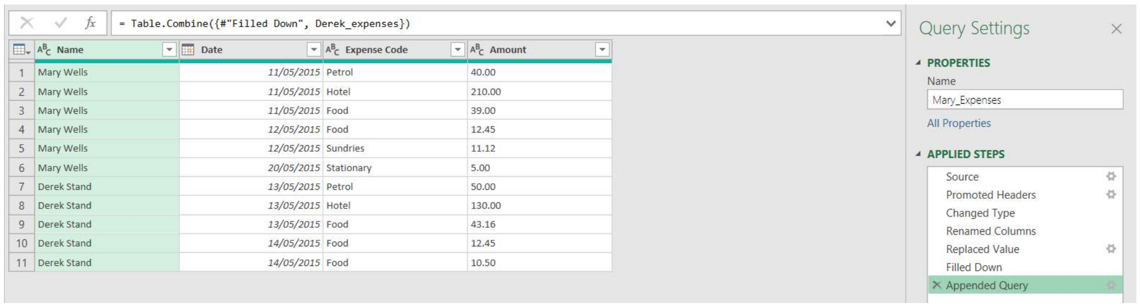
Concatenate rows from two tables into a single table.

[✓ Two tables] Three or more tables

Table to append

- Mary_Expenses (Current)
- Derek_expenses

I don't need to choose the 'Three or more tables' option; I can simply select the other query from the dropdown.

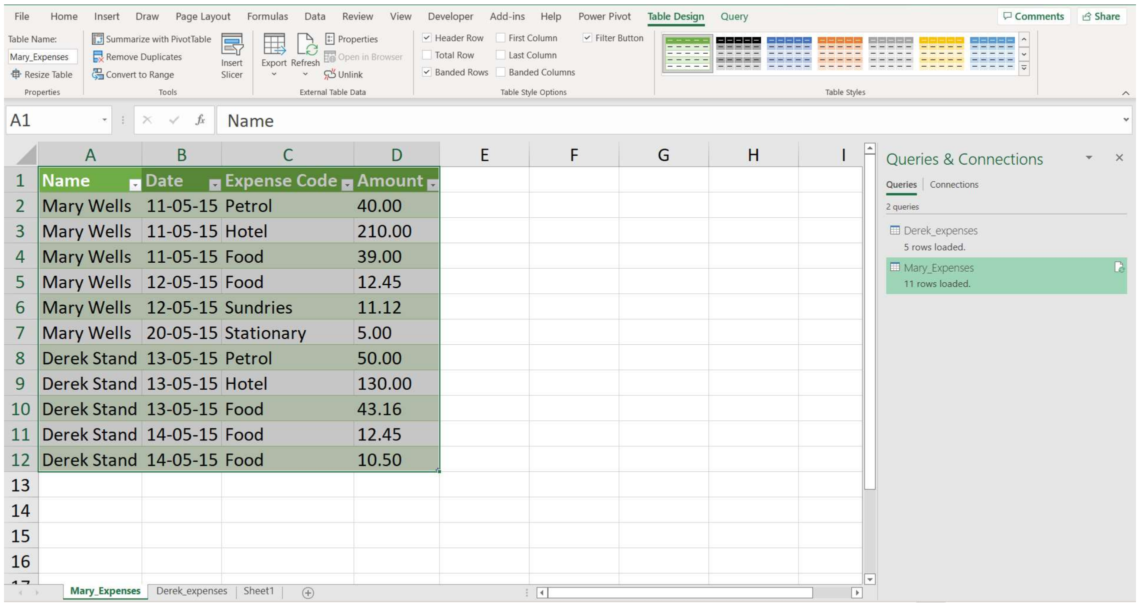


```
= Table.Combine({#"Filled Down", Derek_expenses})
```

	Name	Date	Expense Code	Amount
1	Mary Wells	11/05/2015	Petrol	40.00
2	Mary Wells	11/05/2015	Hotel	210.00
3	Mary Wells	11/05/2015	Food	39.00
4	Mary Wells	12/05/2015	Food	12.45
5	Mary Wells	12/05/2015	Sundries	11.12
6	Mary Wells	20/05/2015	Stationary	5.00
7	Derek Stand	13/05/2015	Petrol	50.00
8	Derek Stand	13/05/2015	Hotel	130.00
9	Derek Stand	13/05/2015	Food	43.16
10	Derek Stand	14/05/2015	Food	12.45
11	Derek Stand	14/05/2015	Food	10.50

The data appears in the same query, and a step is created in **M** code combining the data in the current query with the Derek's expense query that already exists. The number of columns remains the same. If any columns had been duplicated, then I should check column names are the same (including the same case and with no extra spaces) in both queries.

As before, I can 'Close & Load' to see the data in a new Excel sheet:



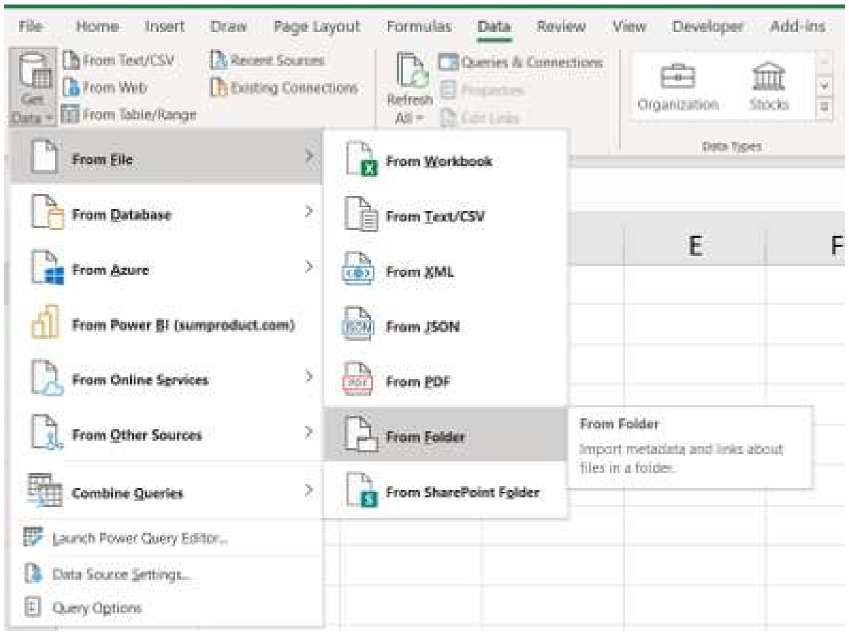
	A	B	C	D
1	Name	Date	Expense Code	Amount
2	Mary Wells	11-05-15	Petrol	40.00
3	Mary Wells	11-05-15	Hotel	210.00
4	Mary Wells	11-05-15	Food	39.00
5	Mary Wells	12-05-15	Food	12.45
6	Mary Wells	12-05-15	Sundries	11.12
7	Mary Wells	20-05-15	Stationary	5.00
8	Derek Stand	13-05-15	Petrol	50.00
9	Derek Stand	13-05-15	Hotel	130.00
10	Derek Stand	13-05-15	Food	43.16
11	Derek Stand	14-05-15	Food	12.45
12	Derek Stand	14-05-15	Food	10.50

Further files can be appended using the same method as they arrive.

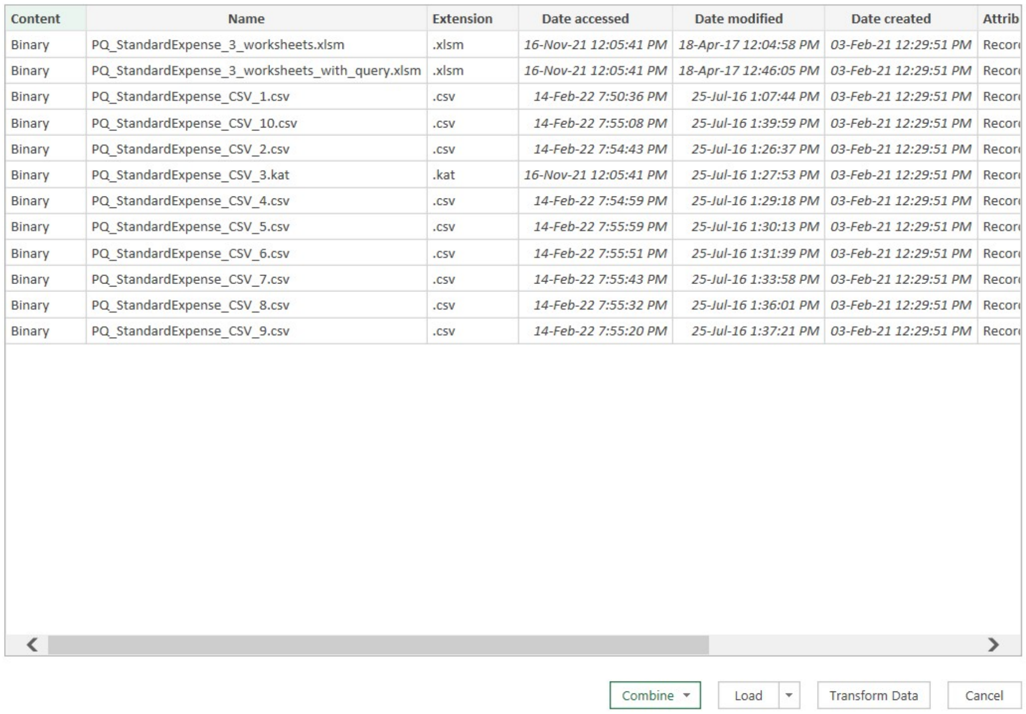
If I want to extract similar files at the same time, there is another option: extracting from a folder, and that's next…

Extracting from a Folder

In this example I have 10 expense files in a folder called **PQ_StandardExpenses**. In a blank workbook, I choose the ‘From File’ Option, and drop down to select ‘From Folder’.



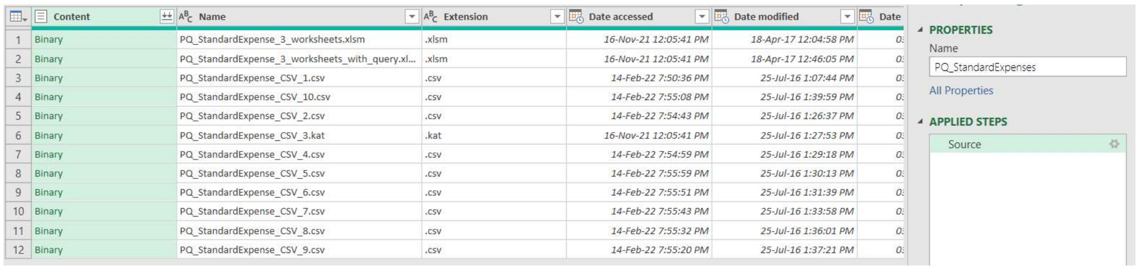
A simple browse window appears, and having chosen the correct folder, the meta data is displayed:



Content	Name	Extension	Date accessed	Date modified	Date created	Attrib
Binary	PQ_StandardExpense_3_worksheets.xlsm	.xlsm	16-Nov-21 12:05:41 PM	18-Apr-17 12:04:58 PM	03-Feb-21 12:29:51 PM	Recor
Binary	PQ_StandardExpense_3_worksheets_with_query.xlsm	.xlsm	16-Nov-21 12:05:41 PM	18-Apr-17 12:46:05 PM	03-Feb-21 12:29:51 PM	Recor
Binary	PQ_StandardExpense_CSV_1.csv	.csv	14-Feb-22 7:50:36 PM	25-Jul-16 1:07:44 PM	03-Feb-21 12:29:51 PM	Recor
Binary	PQ_StandardExpense_CSV_10.csv	.csv	14-Feb-22 7:55:08 PM	25-Jul-16 1:39:59 PM	03-Feb-21 12:29:51 PM	Recor
Binary	PQ_StandardExpense_CSV_2.csv	.csv	14-Feb-22 7:54:43 PM	25-Jul-16 1:26:37 PM	03-Feb-21 12:29:51 PM	Recor
Binary	PQ_StandardExpense_CSV_3.kat	.kat	16-Nov-21 12:05:41 PM	25-Jul-16 1:27:53 PM	03-Feb-21 12:29:51 PM	Recor
Binary	PQ_StandardExpense_CSV_4.csv	.csv	14-Feb-22 7:54:59 PM	25-Jul-16 1:29:18 PM	03-Feb-21 12:29:51 PM	Recor
Binary	PQ_StandardExpense_CSV_5.csv	.csv	14-Feb-22 7:55:59 PM	25-Jul-16 1:30:13 PM	03-Feb-21 12:29:51 PM	Recor
Binary	PQ_StandardExpense_CSV_6.csv	.csv	14-Feb-22 7:55:51 PM	25-Jul-16 1:31:39 PM	03-Feb-21 12:29:51 PM	Recor
Binary	PQ_StandardExpense_CSV_7.csv	.csv	14-Feb-22 7:55:43 PM	25-Jul-16 1:33:58 PM	03-Feb-21 12:29:51 PM	Recor
Binary	PQ_StandardExpense_CSV_8.csv	.csv	14-Feb-22 7:55:32 PM	25-Jul-16 1:36:01 PM	03-Feb-21 12:29:51 PM	Recor
Binary	PQ_StandardExpense_CSV_9.csv	.csv	14-Feb-22 7:55:20 PM	25-Jul-16 1:37:21 PM	03-Feb-21 12:29:51 PM	Recor

At this point, I could choose to combine these files and I'd be done, but editing allows the data to be transformed, and some safety features to be added.

Some kind person has added some excel workbooks in the folder, not to mention a strange file extension 'kat'! I need to make sure nothing added to the folder in the future will mess up my query. I can also allow for users typing in **csv** or **CSV** when they create their files. I opt to 'Transform Data':



	Content	Name	Extension	Date accessed	Date modified	Date
1	Binary	PQ_StandardExpense_3_worksheets.xlsm	.xlsm	16-Nov-21 12:05:41 PM	18-Apr-17 12:04:58 PM	0:
2	Binary	PQ_StandardExpense_3_worksheets_with_query.xl...	.xlsm	16-Nov-21 12:05:41 PM	18-Apr-17 12:46:05 PM	0:
3	Binary	PQ_StandardExpense_CSV_1.csv	.csv	14-Feb-22 7:50:36 PM	25-Jul-16 1:07:44 PM	0:
4	Binary	PQ_StandardExpense_CSV_10.csv	.csv	14-Feb-22 7:55:08 PM	25-Jul-16 1:39:59 PM	0:
5	Binary	PQ_StandardExpense_CSV_2.csv	.csv	14-Feb-22 7:54:43 PM	25-Jul-16 1:26:37 PM	0:
6	Binary	PQ_StandardExpense_CSV_3.kat	.kat	16-Nov-21 12:05:41 PM	25-Jul-16 1:27:53 PM	0:
7	Binary	PQ_StandardExpense_CSV_4.csv	.csv	14-Feb-22 7:54:59 PM	25-Jul-16 1:29:18 PM	0:
8	Binary	PQ_StandardExpense_CSV_5.csv	.csv	14-Feb-22 7:55:59 PM	25-Jul-16 1:30:13 PM	0:
9	Binary	PQ_StandardExpense_CSV_6.csv	.csv	14-Feb-22 7:55:51 PM	25-Jul-16 1:31:39 PM	0:
10	Binary	PQ_StandardExpense_CSV_7.csv	.csv	14-Feb-22 7:55:43 PM	25-Jul-16 1:33:58 PM	0:
11	Binary	PQ_StandardExpense_CSV_8.csv	.csv	14-Feb-22 7:55:32 PM	25-Jul-16 1:36:01 PM	0:
12	Binary	PQ_StandardExpense_CSV_9.csv	.csv	14-Feb-22 7:55:20 PM	25-Jul-16 1:37:21 PM	0:

PROPERTIES

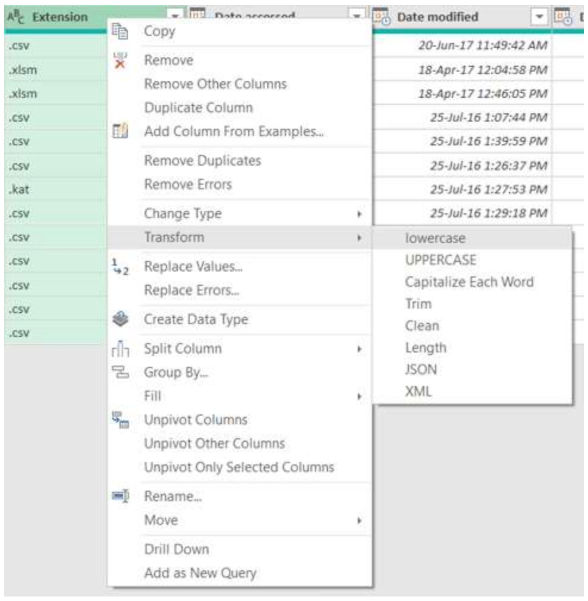
Name: PQ_StandardExpenses

All Properties

APPLIED STEPS

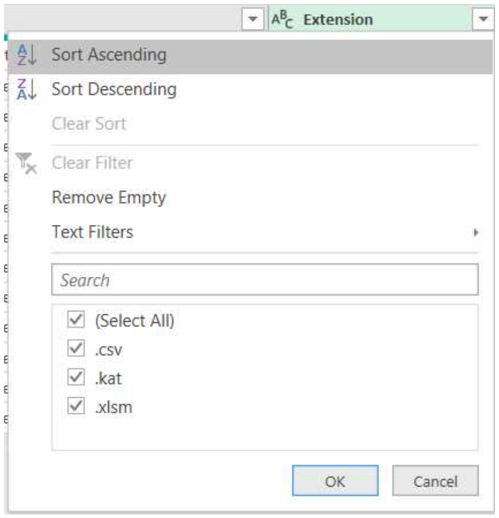
Source

I start by transforming the **Extension** data to lowercase. I select the column and right-click to find the 'Transform' to 'lowercase' option.

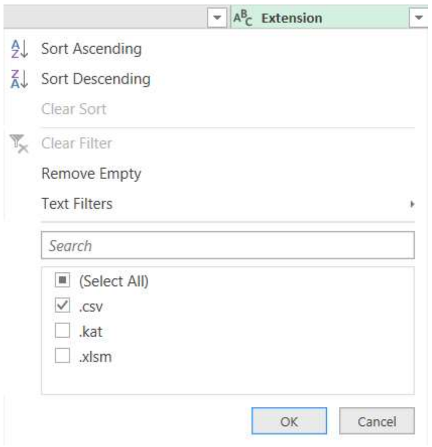


Extension	Date modified
.csv	20-Jun-17 11:49:42 AM
.xlsm	18-Apr-17 12:04:58 PM
.xlsm	18-Apr-17 12:46:05 PM
.csv	25-Jul-16 1:07:44 PM
.csv	25-Jul-16 1:39:59 PM
.csv	25-Jul-16 1:26:37 PM
.kat	25-Jul-16 1:27:53 PM
.csv	25-Jul-16 1:29:18 PM
.csv	
.csv	
.csv	
.csv	
.csv	

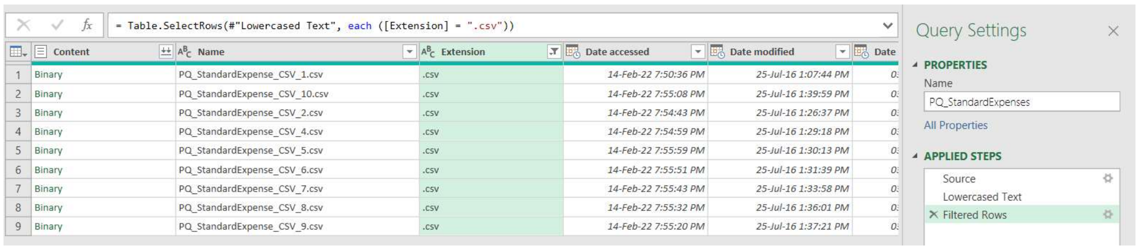
Next, I can filter to just get those files with file extension '**csv**'. At the top of the **Extension** column there is a standard filter arrow: clicking on this reveals a number of options to transform the data in the column.



I could use the ‘Text Filter’, but instead, I will choose ‘**csv**’ from the radio list:



This generates a step to filter on ‘**.csv**’:



```
= Table.SelectRows(#"Lowercased Text", each ([Extension] = ".csv"))
```

	Content	Name	Extension	Date accessed	Date modified	Date
1	Binary	PQ_StandardExpense_CSV_1.csv	.csv	14-Feb-22 7:50:36 PM	25-Jul-16 1:07:44 PM	0:
2	Binary	PQ_StandardExpense_CSV_10.csv	.csv	14-Feb-22 7:55:08 PM	25-Jul-16 1:39:59 PM	0:
3	Binary	PQ_StandardExpense_CSV_2.csv	.csv	14-Feb-22 7:54:43 PM	25-Jul-16 1:26:37 PM	0:
4	Binary	PQ_StandardExpense_CSV_4.csv	.csv	14-Feb-22 7:54:59 PM	25-Jul-16 1:29:18 PM	0:
5	Binary	PQ_StandardExpense_CSV_5.csv	.csv	14-Feb-22 7:55:59 PM	25-Jul-16 1:30:13 PM	0:
6	Binary	PQ_StandardExpense_CSV_6.csv	.csv	14-Feb-22 7:55:51 PM	25-Jul-16 1:31:39 PM	0:
7	Binary	PQ_StandardExpense_CSV_7.csv	.csv	14-Feb-22 7:55:43 PM	25-Jul-16 1:33:58 PM	0:
8	Binary	PQ_StandardExpense_CSV_8.csv	.csv	14-Feb-22 7:55:32 PM	25-Jul-16 1:36:01 PM	0:
9	Binary	PQ_StandardExpense_CSV_9.csv	.csv	14-Feb-22 7:55:20 PM	25-Jul-16 1:37:21 PM	0:

Query Settings

PROPERTIES

Name: PQ_StandardExpenses

All Properties

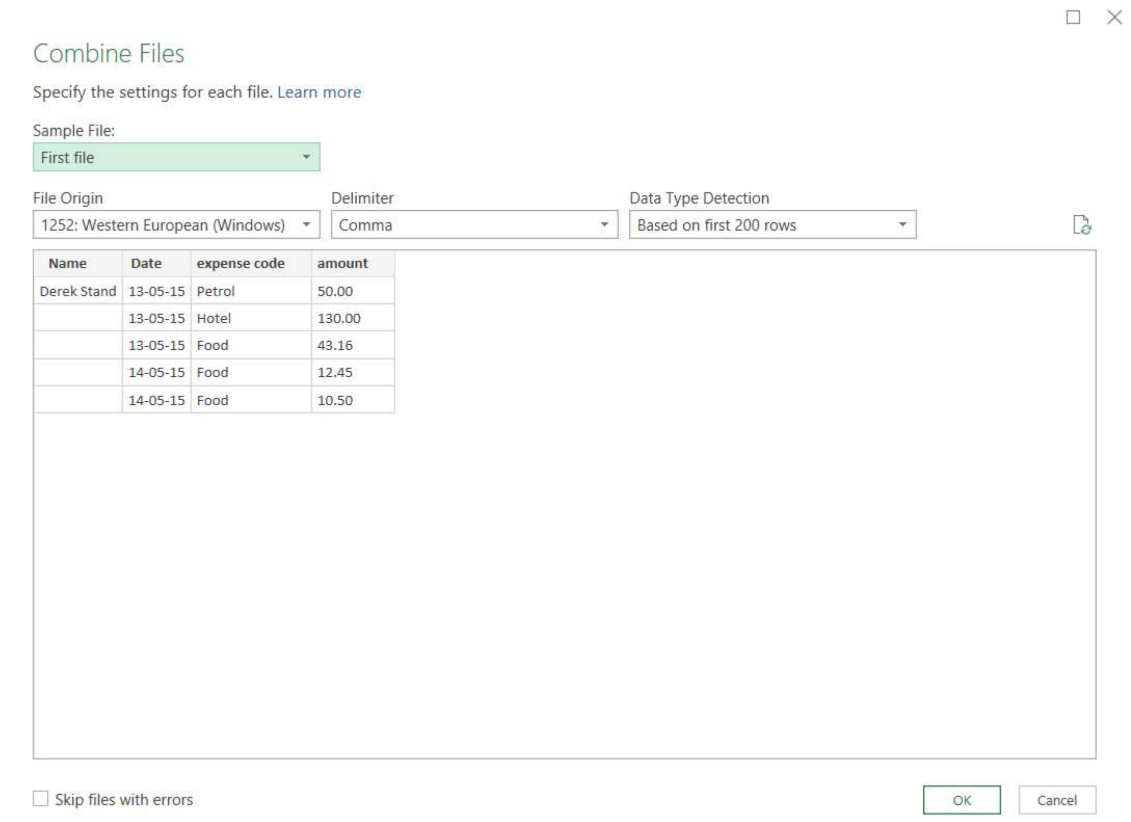
APPLIED STEPS

- Source
- Lowercased Text
- Filtered Rows

The query is now protected from stray workbooks and will include files with **.csv** and **.CSV** extensions. Simple!

In order to transform the data in the files, I need to see the table contents, and not just the metadata. Next to the **Content** column header is an icon which appears for binary files, and allows the binary to be combined – the icon.

Pressing the button with this icon makes the contents of each table appear under the table headings:



Combine Files

Specify the settings for each file. Learn more

Sample File: First file

File Origin: 1252: Western European (Windows)

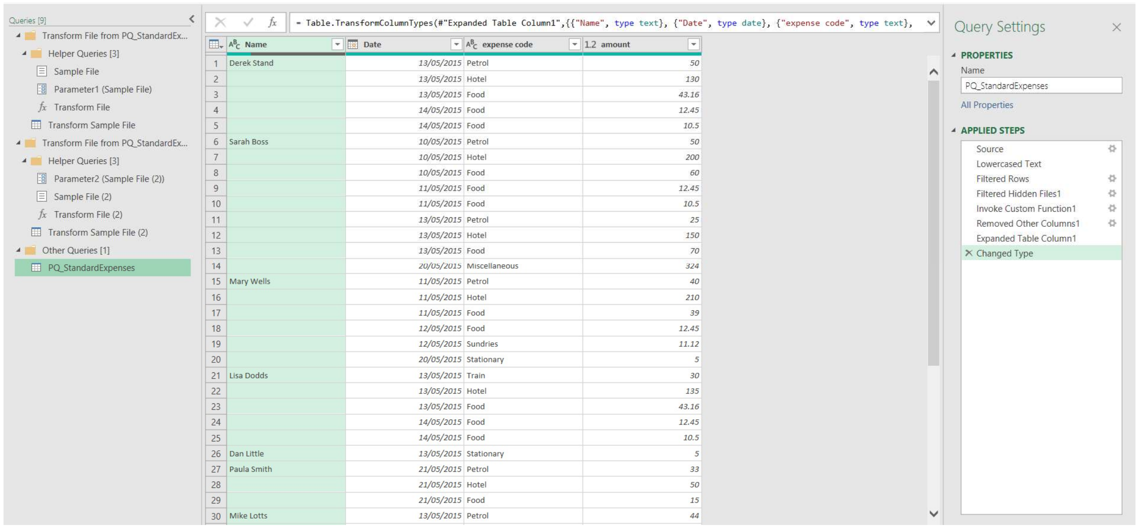
Delimiter: Comma

Data Type Detection: Based on first 200 rows

Name	Date	expense code	amount
Derek Stand	13-05-15	Petrol	50.00
	13-05-15	Hotel	130.00
	13-05-15	Food	43.16
	14-05-15	Food	12.45
	14-05-15	Food	10.50

Skip files with errors

This is just showing me the first file, and I am happy with the layout, so I click OK:



Queries [9]
- Transform File from PQ_StandardEx...
  - Helper Queries [3]
    - Sample File
    - Parameter1 (Sample File)
    - Transform File
  - Transform Sample File
- Transform File from PQ_StandardEx...
  - Helper Queries [3]
    - Parameter2 (Sample File (2))
    - Sample File (2)
    - Transform File (2)
  - Transform Sample File (2)
- Other Queries [1]
  - PQ_StandardExpenses

```
= Table.TransformColumnTypes(#"Expanded Table Column1",{{"Name", type text}, {"Date", type date}, {"expense code", type text},
```

	Name	Date	expense code	amount
1	Derek Stand	13/05/2015	Petrol	50
2		13/05/2015	Hotel	130
3		13/05/2015	Food	43.16
4		14/05/2015	Food	12.45
5		14/05/2015	Food	10.5
6	Sarah Boss	10/05/2015	Petrol	50
7		10/05/2015	Hotel	200
8		10/05/2015	Food	60
9		11/05/2015	Food	12.45
10		11/05/2015	Food	10.5
11		13/05/2015	Petrol	25
12		13/05/2015	Hotel	150
13		13/05/2015	Food	70
14		20/05/2015	Miscellaneous	324
15	Mary Wells	11/05/2015	Petrol	40
16		11/05/2015	Hotel	210
17		11/05/2015	Food	39
18		12/05/2015	Food	12.45
19		12/05/2015	Sundries	11.12
20		20/05/2015	Stationary	5
21	Lisa Dodds	13/05/2015	Train	30
22		13/05/2015	Hotel	135
23		13/05/2015	Food	43.16
24		14/05/2015	Food	12.45
25		14/05/2015	Food	10.5
26	Dan Little	13/05/2015	Stationary	5
27	Paula Smith	21/05/2015	Petrol	33
28		21/05/2015	Hotel	50
29		21/05/2015	Food	15
30	Mike Lotts	13/05/2015	Petrol	44

Query Settings

PROPERTIES

Name: PQ_StandardExpenses

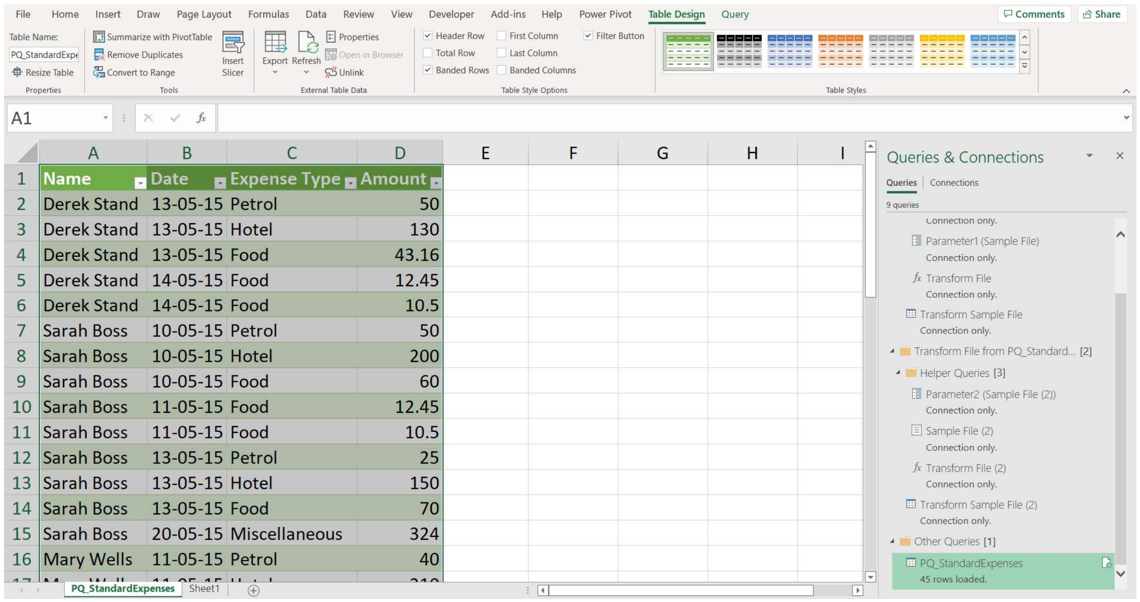
All Properties

APPLIED STEPS
- Source
- Lowercased Text
- Filtered Rows
- Filtered Hidden Files1
- Invoke Custom Function1
- Removed Other Columns1
- Expanded Table Column1
- Changed Type

This process has been improved over time, and Power Query creates a 'Helper Queries' folder in the Queries tab to help to transform my data.

In previous versions, I had to promote headings and remove heading rows from each appended file, but all of this has been done for me.   I need to do is 'Fill Down' on the **Name** column using the same method as before, by replacing blanks with nulls and right-clicking to 'Fill Down'. I also rename expense code and amount to '**Expense Type**' and '**Amount**', so that I will be able to append to other expense files with no duplication.

My data is now ready for me to 'Close & Load' to the workbook:



	A	B	C	D
1	Name	Date	Expense Type	Amount
2	Derek Stand	13-05-15	Petrol	50
3	Derek Stand	13-05-15	Hotel	130
4	Derek Stand	13-05-15	Food	43.16
5	Derek Stand	14-05-15	Food	12.45
6	Derek Stand	14-05-15	Food	10.5
7	Sarah Boss	10-05-15	Petrol	50
8	Sarah Boss	10-05-15	Hotel	200
9	Sarah Boss	10-05-15	Food	60
10	Sarah Boss	11-05-15	Food	12.45
11	Sarah Boss	11-05-15	Food	10.5
12	Sarah Boss	13-05-15	Petrol	25
13	Sarah Boss	13-05-15	Hotel	150
14	Sarah Boss	13-05-15	Food	70
15	Sarah Boss	20-05-15	Miscellaneous	324
16	Mary Wells	11-05-15	Petrol	40

Queries & Connections

Queries | Connections

9 queries

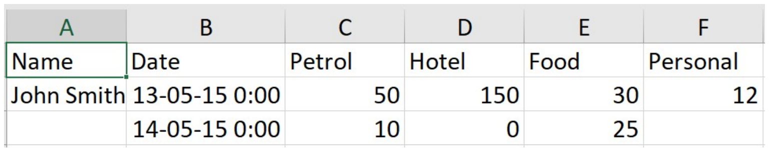
Connection only.
Parameter1 (Sample File)
Connection only.
Transform File
Connection only.
Transform Sample File
Connection only.
Transform File from PQ_Standard... [2]
Helper Queries [3]
Parameter2 (Sample File (2))
Connection only.
Sample File (2)
Connection only.
Transform File (2)
Connection only.
Transform Sample File (2)
Connection only.
Other Queries [1]
PQ_StandardExpenses
45 rows loaded.

Extracting from a folder is an efficient way to upload multiple expense files in a similar format, but what if someone sends in a **CSV** file that has been configured differently? We deal with a simple example of unpivoting data next…

Unpivoting Data

Not everyone sticks to the standard format on their expenses, as that would be far too convenient for data analysts!

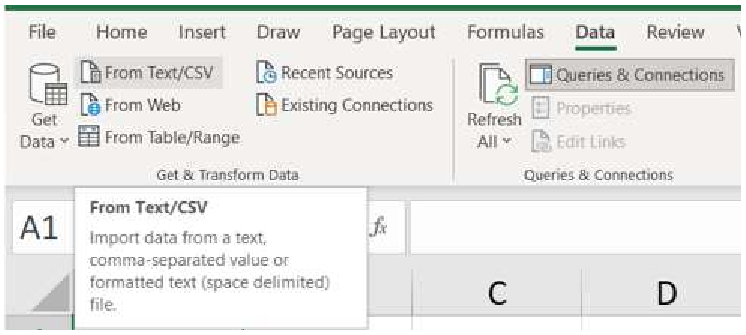
Let's imagine **PQ_NonStandardExpense_CSV** has come in, and it is an expense **csv** which is not in the usual format because John has configured his data ready to create a graph of where his expenses go. Convenient for him, but not so great for appending to the existing expense query:



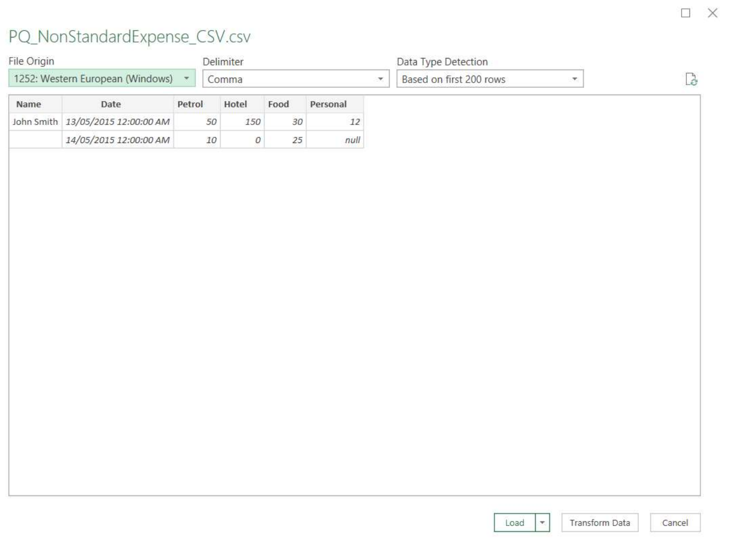
A	B	C	D	E	F
Name	Date	Petrol	Hotel	Food	Personal
John Smith	13-05-15 0:00	50	150	30	12
	14-05-15 0:00	10	0	25	

Luckily, Power Query has a button for this: in order to get this into our required column format I will need to unpivot the data.

The first step is importing the non-standard expense **CSV** as I did for the single **CSV** file earlier:



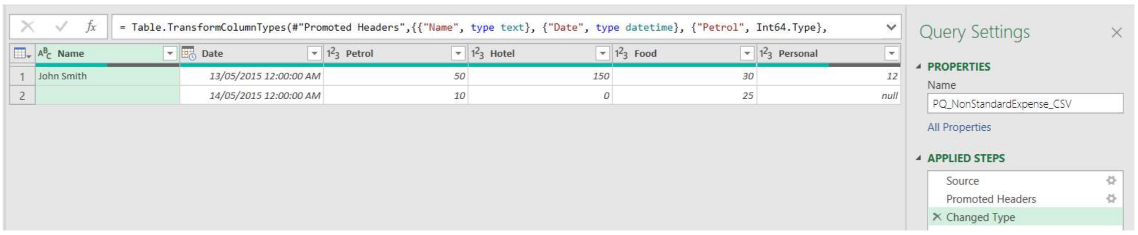
I browse to the file and select it:



File Origin: 1252: Western European (Windows)
Delimiter: Comma
Data Type Detection: Based on first 200 rows

Name	Date	Petrol	Hotel	Food	Personal
John Smith	13/05/2015 12:00:00 AM	50	150	30	12
	14/05/2015 12:00:00 AM	10	0	25	null

I will choose 'Transform Data'.



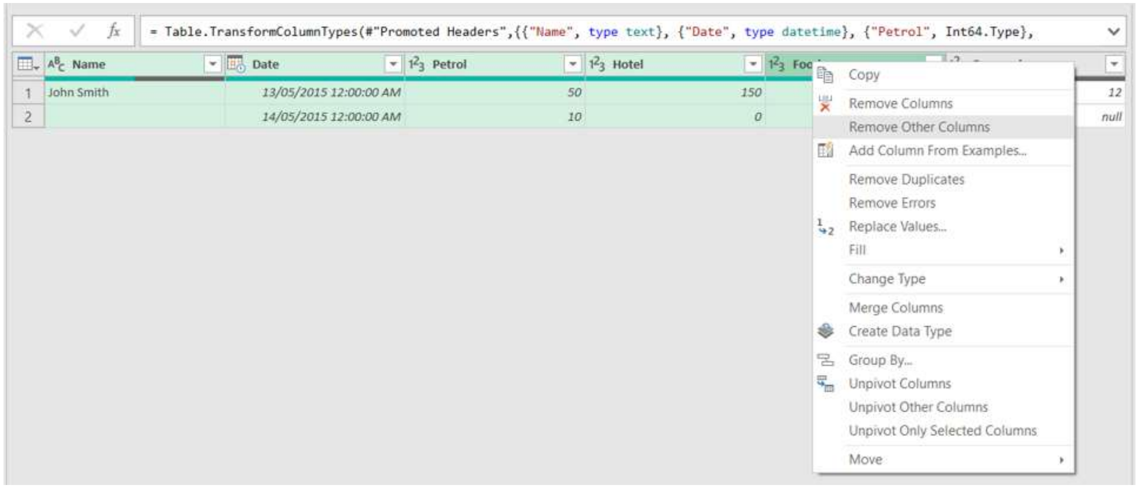
```
= Table.TransformColumnTypes(#"Promoted Headers",{{"Name", type text}, {"Date", type datetime}, {"Petrol", Int64.Type},
```

	Name	Date	Petrol	Hotel	Food	Personal
1	John Smith	13/05/2015 12:00:00 AM	50	150	30	12
2		14/05/2015 12:00:00 AM	10	0	25	null

Name: PQ_NonStandardExpense_CSV

Applied Steps:
- Source
- Promoted Headers
- Changed Type

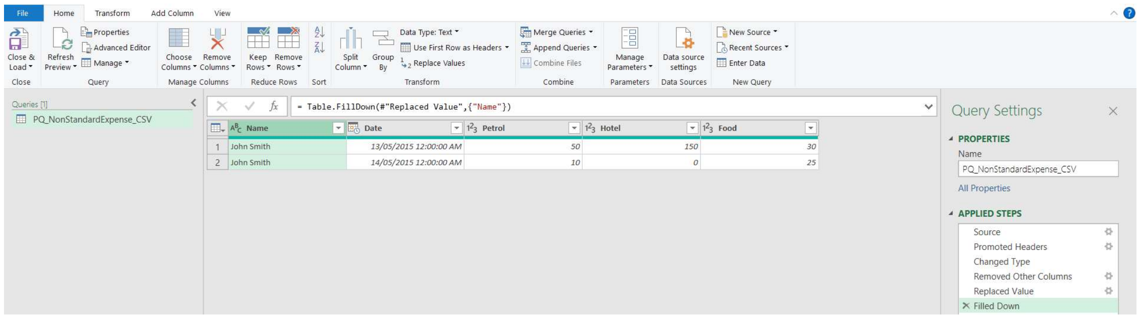
The first step for this example is to remove any unnecessary data – in this case John has added a personal expense column which is not required in the model. However, rather than remove the offending **Personal** column, it is safer to keep the columns that I know I need in order to append this query to the other expenses. I hold **CTRL** while I select the columns I want to keep. Then I can right-click and choose 'Remove other Columns'. A tip here is to select and click whilst in the column heading, otherwise Power Query can assume I am selecting a value in the column.



```
= Table.TransformColumnTypes(#"Promoted Headers",{{"Name", type text}, {"Date", type datetime}, {"Petrol", Int64.Type},
```

	Name	Date	Petrol	Hotel	Foo	
1	John Smith	13/05/2015 12:00:00 AM	50	150		12
2		14/05/2015 12:00:00 AM	10	0		null

Name can then be filled down using the same process that I used earlier: I replace the blank values with *null* and then right-click on the **Name** column and choose to 'Fill Down'.



```
= Table.FillDown(#"Replaced Value",{"Name"})
```

	Name	Date	Petrol	Hotel	Food
1	John Smith	13/05/2015 12:00:00 AM	50	150	30
2	John Smith	14/05/2015 12:00:00 AM	10	0	25

Query Settings

PROPERTIES

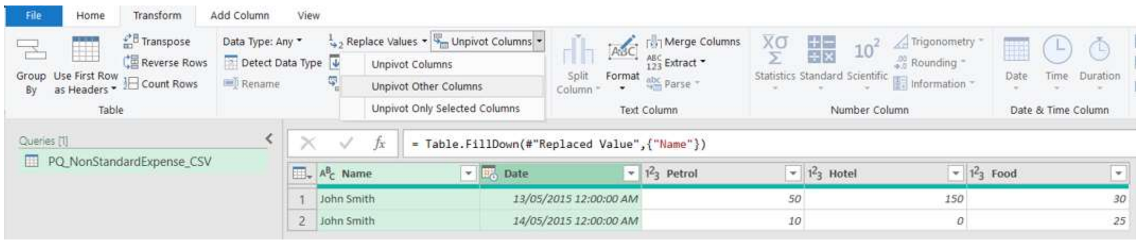
Name: PQ_NonStandardExpense_CSV

All Properties

APPLIED STEPS

- Source
- Promoted Headers
- Changed Type
- Removed Other Columns
- Replaced Value
- Filled Down

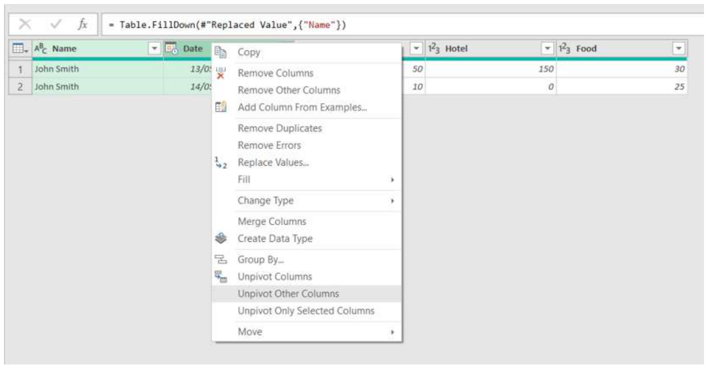
I am ready to unpivot process – I select **Name** and **Date** as I plan to unpivot the other columns. In the 'Any Columns' group on the 'Transform' tab there is an option to 'Unpivot Columns' – there are two options, either unpivot the selected columns, or everything except the selected columns, which can save time depending on the proportion of columns that need to be unpivoted.



```
= Table.FillDown(#"Replaced Value",{"Name"})
```

	Name	Date	Petrol	Hotel	Food
1	John Smith	13/05/2015 12:00:00 AM	50	150	30
2	John Smith	14/05/2015 12:00:00 AM	10	0	25

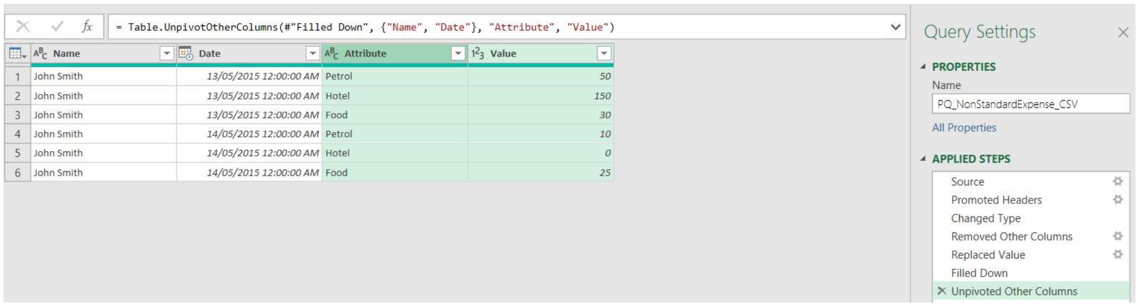
The unpivot options are also available by right-clicking any of the highlighted columns:



```
= Table.FillDown(#"Replaced Value",{"Name"})
```

	Name	Date		Hotel	Food
1	John Smith	13/0	50	150	30
2	John Smith	14/0	10	0	25

One click and everything is unpivoted as required!



```
= Table.UnpivotOtherColumns(#"Filled Down", {"Name", "Date"}, "Attribute", "Value")
```

	Name	Date	Attribute	Value
1	John Smith	13/05/2015 12:00:00 AM	Petrol	50
2	John Smith	13/05/2015 12:00:00 AM	Hotel	150
3	John Smith	13/05/2015 12:00:00 AM	Food	30
4	John Smith	14/05/2015 12:00:00 AM	Petrol	10
5	John Smith	14/05/2015 12:00:00 AM	Hotel	0
6	John Smith	14/05/2015 12:00:00 AM	Food	25

Query Settings

PROPERTIES

Name: PQ_NonStandardExpense_CSV

All Properties

APPLIED STEPS

- Source
- Promoted Headers
- Changed Type
- Removed Other Columns
- Replaced Value
- Filled Down
- Unpivoted Other Columns

The small changes left to do are to transform the unpivoted columns to match the other expenses. I rename the headings by double-clicking and editing.

The data is then ready to be uploaded or appended to an existing expenses query as covered earlier.

Relationship between Power Query and Power Pivot

This is relevant to Power BI too. So far, I have concentrated on how to use Power Query as a standalone tool. In a moment, I will look at extracting data from a table and loading to the Excel model. This will allow the data to be manipulated by another Excel Add-in: Power Pivot. Therefore, now I will take a look at the relationship between Power Query and Power Pivot.

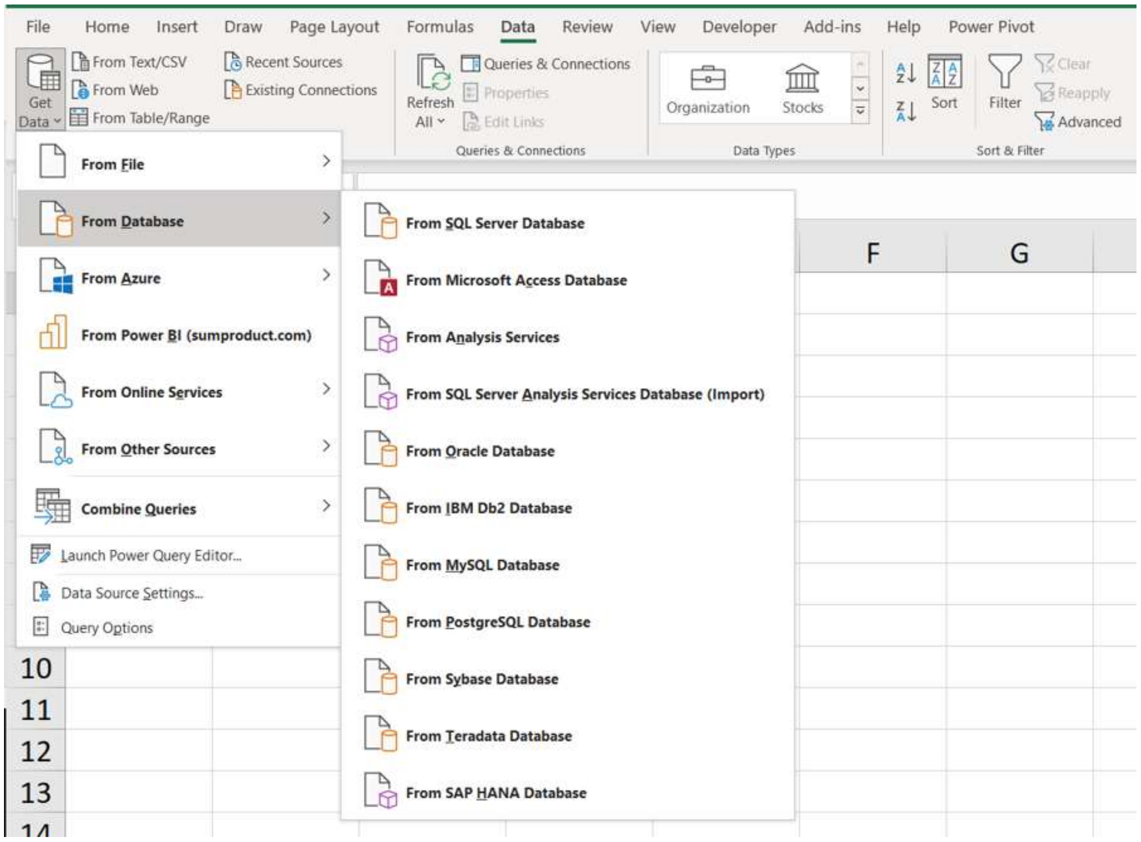
I have shown that Power Query is a useful ETL (**E**xtract, **T**ransform and **L**oad) tool. Well thought out features allow data to be filtered, manipulated and merged. In addition to the previous examples where I have loaded data into an Excel worksheet, loading to the Excel model allows further modelling and analysis by Power Pivot.

Power Pivot can fine tune the Excel Model and then perform calculations so that the data is ready to be displayed in PivotTables, charts or grids, or in a visualisation tool like Power View. Power Pivot can create relationships between tables, and create formulas and KPI's (**K**ey **P**erformance **I**ndicators). Power Pivot refines and builds on the Excel model created by Power Query. Next, I will describe how to extract a table from a database, transform the data, and load it into the Excel model ready for further analysis using Power Pivot.

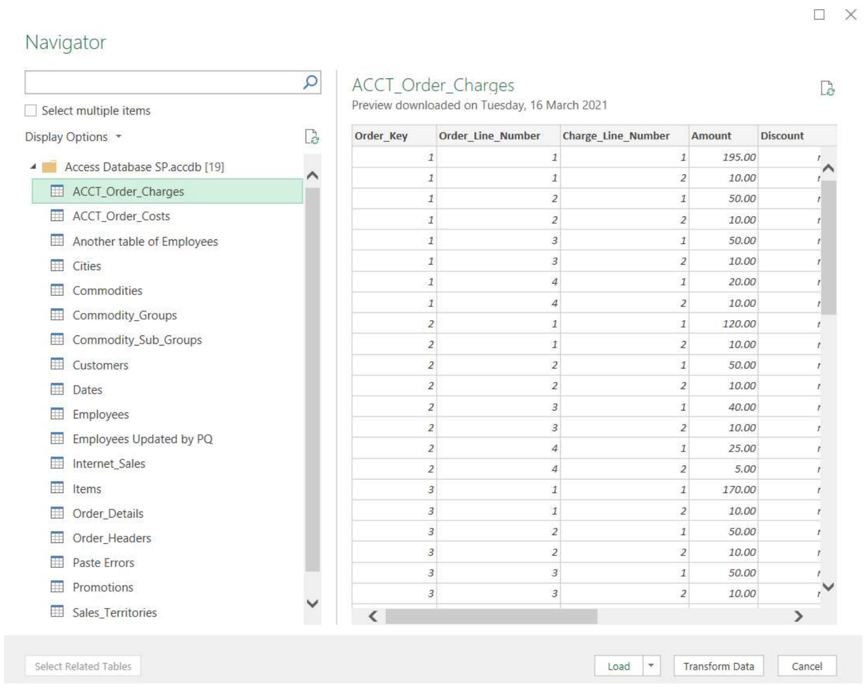
(Data) Model Building

Up until now, I have concentrated on extracting data from files and folders. For this example, I will extract data from a table in a Microsoft Access database. Although I could use this data in an Excel workbook as in the previous examples, this time I will add the table to the Excel data model, so that it can be used by Power Pivot.

A quick glance at the drop down 'From Database' under 'Get Data' on the Data tab reveals a large selection of different kinds of databases that can be accessed in order to extract data:



Choosing the 'From Microsoft Access Database' option allows me to browse for the database that I will use. Having selected my database, the navigator screen lists all the tables in that database. I choose to preview '**ACCT_Order_Charges**':



Navigator

Select multiple items

Display Options

Access Database SP.accdb [19]
- ACCT_Order_Charges
- ACCT_Order_Costs
- Another table of Employees
- Cities
- Commodities
- Commodity_Groups
- Commodity_Sub_Groups
- Customers
- Dates
- Employees
- Employees Updated by PQ
- Internet_Sales
- Items
- Order_Details
- Order_Headers
- Paste Errors
- Promotions
- Sales_Territories

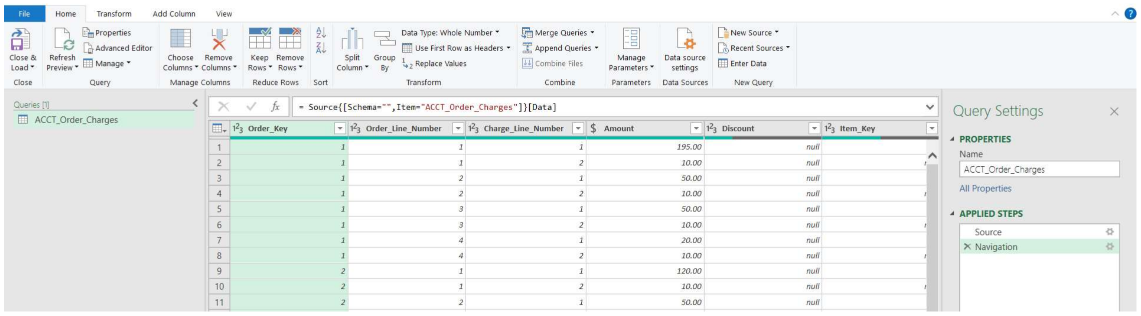
ACCT_Order_Charges

Preview downloaded on Tuesday, 16 March 2021

Order_Key	Order_Line_Number	Charge_Line_Number	Amount	Discount
1	1	1	195.00	r
1	1	2	10.00	r
1	2	1	50.00	r
1	2	2	10.00	r
1	3	1	50.00	r
1	3	2	10.00	r
1	4	1	20.00	r
1	4	2	10.00	r
2	1	1	120.00	r
2	1	2	10.00	r
2	2	1	50.00	r
2	2	2	10.00	r
2	3	1	40.00	r
2	3	2	10.00	r
2	4	1	25.00	r
2	4	2	5.00	r
3	1	1	170.00	r
3	1	2	10.00	r
3	2	1	50.00	r
3	2	2	10.00	r
3	3	1	50.00	r
3	3	2	10.00	r

Select Related Tables | Load | Transform Data | Cancel

I choose to 'Transform Data'.



Queries [1]

ACCT_Order_Charges

```
= Source{[Schema="",Item="ACCT_Order_Charges"]}[Data]
```

	Order_Key	Order_Line_Number	Charge_Line_Number	Amount	Discount	Item_Key
1	1	1	1	195.00	null	
2	1	1	2	10.00	null	
3	1	2	1	50.00	null	
4	1	2	2	10.00	null	
5	1	3	1	50.00	null	
6	1	3	2	10.00	null	
7	1	4	1	20.00	null	
8	1	4	2	10.00	null	
9	2	1	1	120.00	null	
10	2	1	2	10.00	null	
11	2	2	1	50.00	null	

Query Settings

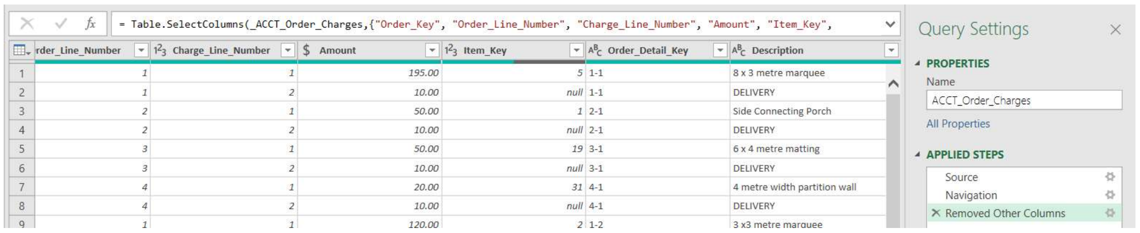
PROPERTIES

Name: ACCT_Order_Charges

All Properties

APPLIED STEPS
- Source
- Navigation

I select the columns I want to keep whilst holding **CTRL** and right-click to 'Remove other Columns' as I did for the **CSV** example.



```
= Table.SelectColumns(_ACCT_Order_Charges,{"Order_Key", "Order_Line_Number", "Charge_Line_Number", "Amount", "Item_Key",
```

	rder_Line_Number	Charge_Line_Number	Amount	Item_Key	Order_Detail_Key	Description
1	1	1	195.00	5	1-1	8 x 3 metre marquee
2	1	2	10.00	null	1-1	DELIVERY
3	2	1	50.00	1	2-1	Side Connecting Porch
4	2	2	10.00	null	2-1	DELIVERY
5	3	1	50.00	19	3-1	6 x 4 metre matting
6	3	2	10.00	null	3-1	DELIVERY
7	4	1	20.00	31	4-1	4 metre width partition wall
8	4	2	10.00	null	4-1	DELIVERY
9	1	1	120.00	2	1-2	3 x3 metre marquee

Query Settings

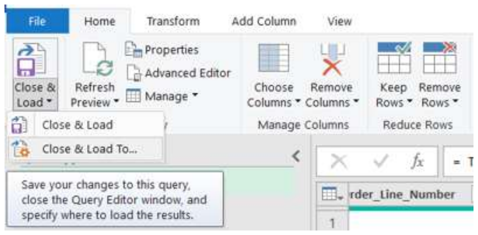
PROPERTIES

Name: ACCT_Order_Charges

All Properties

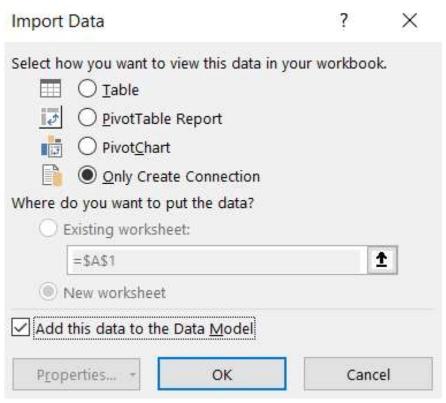
APPLIED STEPS
- Source
- Navigation
- Removed Other Columns

I can then use the 'Close & Load' option from the 'File' tab or the 'Home' tab:



Save your changes to this query, close the Query Editor window, and specify where to load the results.

I must choose 'Close & Load To...', which will allow me to specify how I want to load the data.


Import Data

Select how you want to view this data in your workbook.

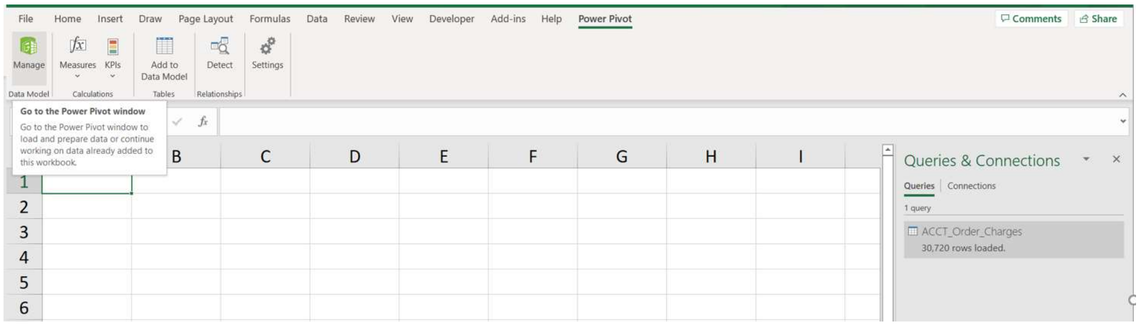
- ○ Table
- ○ PivotTable Report
- ○ PivotChart
- ● Only Create Connection

Where do you want to put the data?

- ○ Existing worksheet: =$A$1
- ○ New worksheet

☑ Add this data to the Data Model

This gives me the option to 'Add this data to the Data Model'. I also choose the 'Connection Only' option since I don't need to see the table in the workbook. Since I have Power Pivot, I can go to Power Pivot tab and select the Manage (data model) option; 'ACCT_Order_Charges' is visible in the 'Workbook Queries' pane.



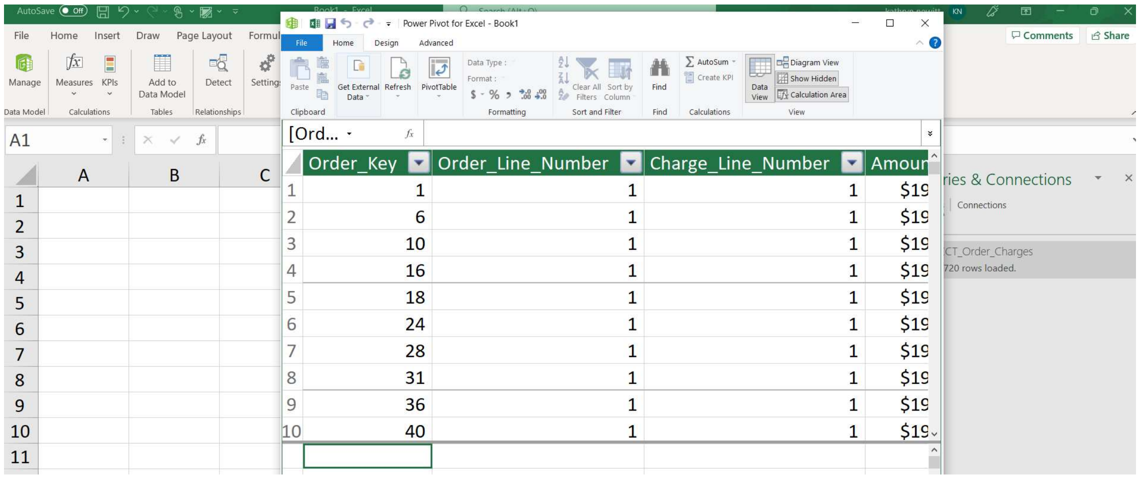
Go to the Power Pivot window

Go to the Power Pivot window to load and prepare data or continue working on data already added to this workbook.

1 query

ACCT_Order_Charges
30,720 rows loaded.

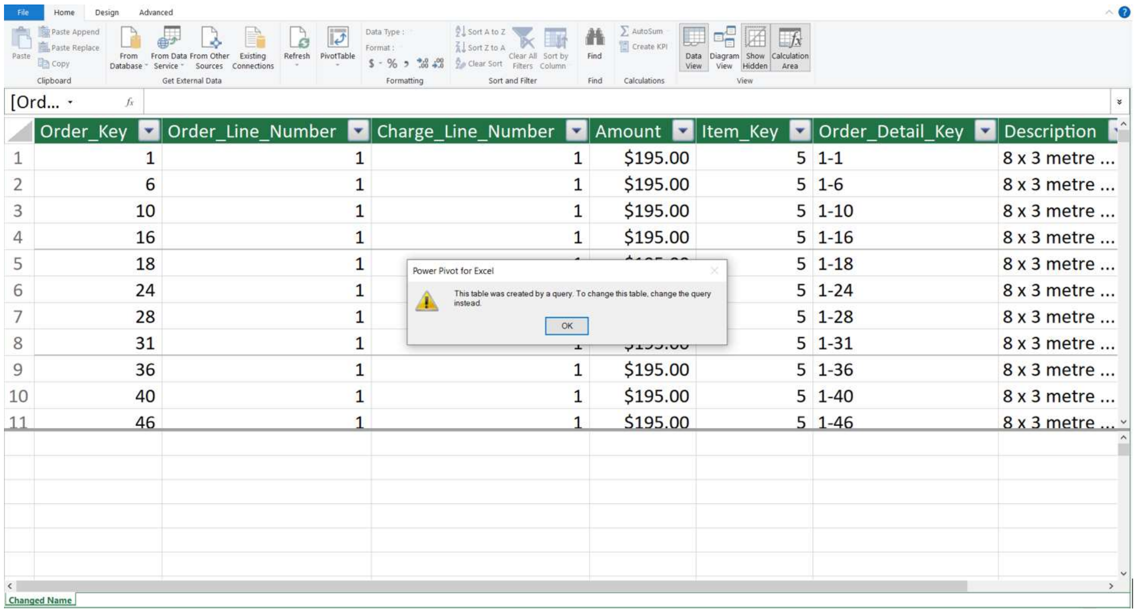
Choosing to 'Manage' allows me to view the 'ACCT_Order_Charges' table in detail, ready to refine the data further by adding calculated columns and creating and managing relationships with other tables in the model.



	Order_Key	Order_Line_Number	Charge_Line_Number	Amount
1	1	1	1	$19
2	6	1	1	$19
3	10	1	1	$19
4	16	1	1	$19
5	18	1	1	$19
6	24	1	1	$19
7	28	1	1	$19
8	31	1	1	$19
9	36	1	1	$19
10	40	1	1	$19

If I can edit the table in Power Query and Power Pivot, are there any problems to look out for? The good news for Excel 2016 onwards and Office 365 users is that since Power Query has been fully integrated into the 'Get & Transform' section, there is no choice between using Power Pivot or Power Query. In earlier versions of Excel 2013, and in Excel 2010, it was possible to corrupt the Excel model by making some types of changes in Power Pivot. The current version of Excel 2013 stops me from making these changes in Power Pivot. And that is a *good* thing.

Starting with the Excel model I created above, I've decided to change the table name to make it more user friendly:



	Order_Key	Order_Line_Number	Charge_Line_Number	Amount	Item_Key	Order_Detail_Key	Description
1	1	1	1	$195.00	5	1-1	8 x 3 metre ...
2	6	1	1	$195.00	5	1-6	8 x 3 metre ...
3	10	1	1	$195.00	5	1-10	8 x 3 metre ...
4	16	1	1	$195.00	5	1-16	8 x 3 metre ...
5	18	1	[illegible]	[illegible]	5	1-18	8 x 3 metre ...
6	24	1	[illegible]	[illegible]	5	1-24	8 x 3 metre ...
7	28	1	[illegible]	[illegible]	5	1-28	8 x 3 metre ...
8	31	1	[illegible]	[illegible]	5	1-31	8 x 3 metre ...
9	36	1	1	$195.00	5	1-36	8 x 3 metre ...
10	40	1	1	$195.00	5	1-40	8 x 3 metre ...
11	46	1	1	$195.00	5	1-46	8 x 3 metre ...

Power Pivot for Excel

This table was created by a query. To change this table, change the query instead.

OK

Changed Name

Thus, having received the message in the screen shown above, I am directed back to Power Query to rename the table. The reason that I am stopped from changing the name here is that in previous versions of Excel 2013 and 2010 users could change the name in Power Pivot which then broke the link between Power Query and Power Pivot. Therefore, there are some actions to be avoided in Power Pivot if the Excel model has been created in Power Query:

- **do not** change a table name in Power Pivot
- **do not** rename an imported column in Power Pivot
- **do not** delete an imported column in Power Pivot.

Any of these actions could result in a broken link between Power Query and Power Pivot.

It therefore makes sense to clean up data as much as possible in Power Query, renaming and deleting as required. Although merging some tables and queries in Power Query can be useful to avoid having unnecessary tables, don't try to flatten all the data into one huge table. One of the features of Power Pivot is the ability to manage relationships between tables, allowing keys to be constructed as required. This is vital in creating accurate calculations and useful new columns to aid analysis of data. As with most relationships, Power Query and Power Pivot work well together when they are allowed to do what they are best at.

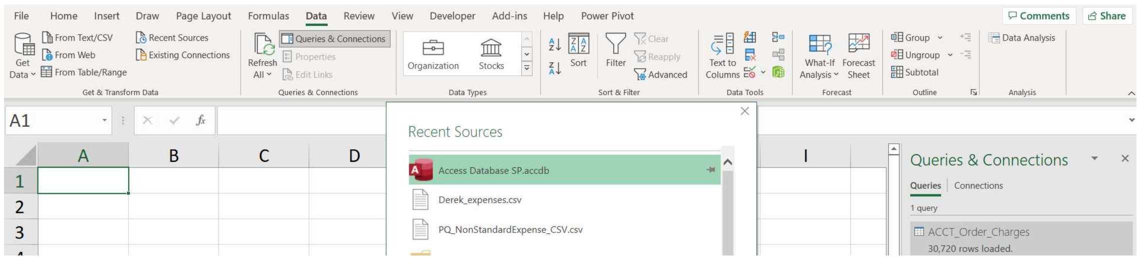
Next, I'll take a look at merging queries in Power Query…

Merging Queries

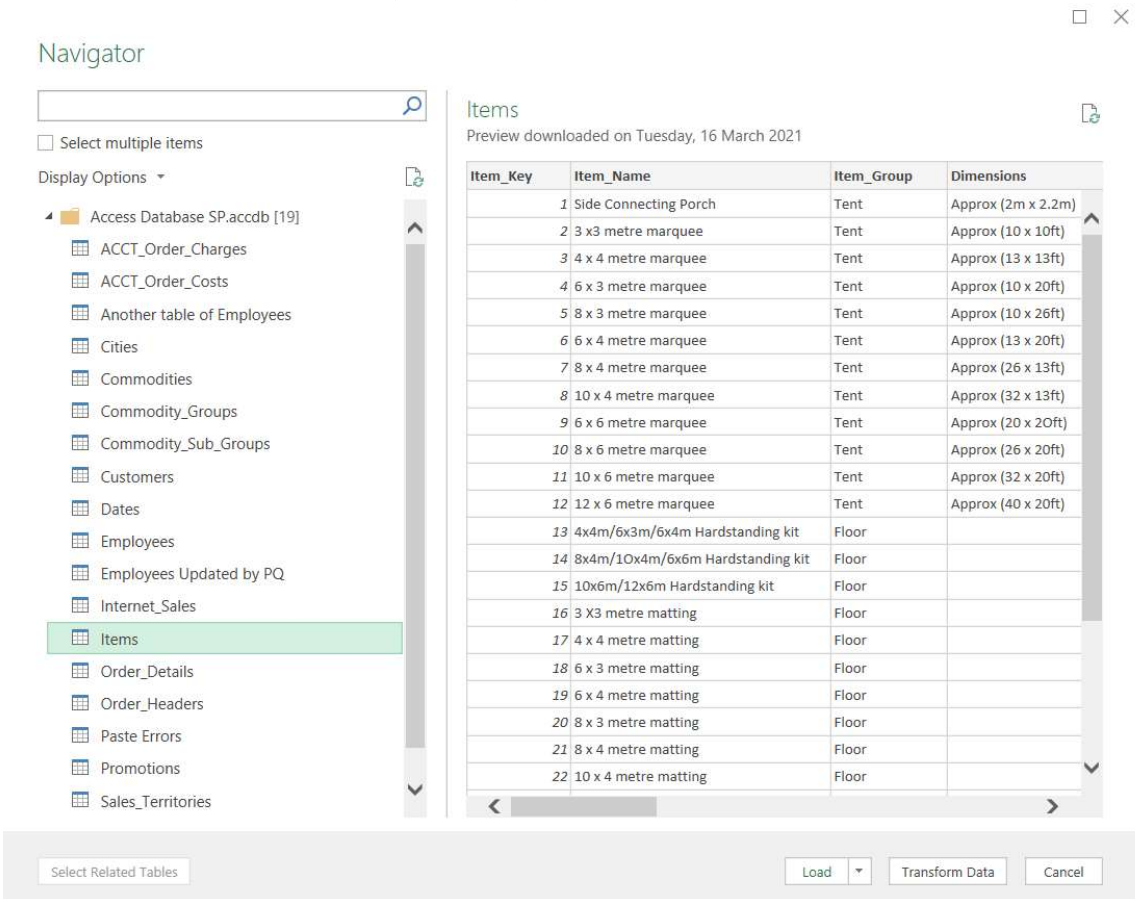
Power Query allows me to merge tables (known as queries in Power Query), without either the need to be an expert on database structure or without having to learn formulae. I only require two (or more) existing queries.

I have a query **ACCT_Order_Charges** which I have extracted from a Microsoft Access database, and I want to merge data from another table so that I can include a description of the type of items that the charges apply to.

Starting in the workbook containing the **ACCT_Order_Charges** query, I create a new query. Since I have already made the connection to the database, I can use the 'Recent Sources' button on the Data tab.



I select the database, and the Navigator dialog appears:



Navigator

Select multiple items

Display Options

Access Database SP.accdb [19]

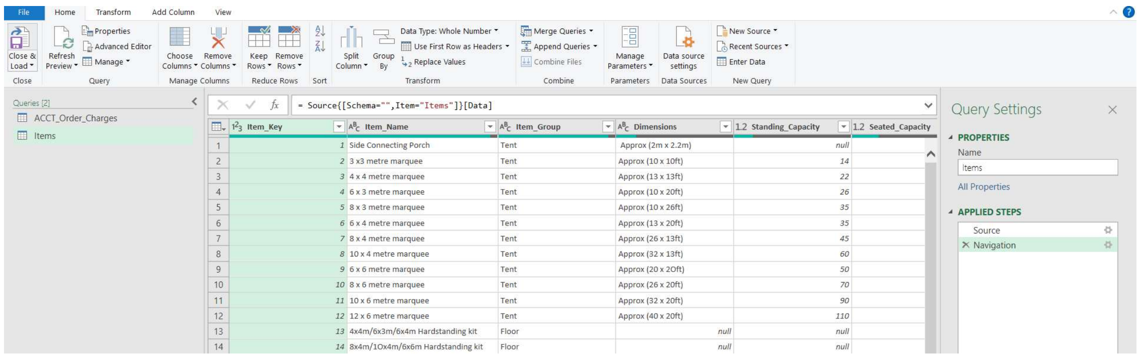
- ACCT_Order_Charges
- ACCT_Order_Costs
- Another table of Employees
- Cities
- Commodities
- Commodity_Groups
- Commodity_Sub_Groups
- Customers
- Dates
- Employees
- Employees Updated by PQ
- Internet_Sales
- Items
- Order_Details
- Order_Headers
- Paste Errors
- Promotions
- Sales_Territories

Items

Preview downloaded on Tuesday, 16 March 2021

Item_Key	Item_Name	Item_Group	Dimensions
1	Side Connecting Porch	Tent	Approx (2m x 2.2m)
2	3 x3 metre marquee	Tent	Approx (10 x 10ft)
3	4 x 4 metre marquee	Tent	Approx (13 x 13ft)
4	6 x 3 metre marquee	Tent	Approx (10 x 20ft)
5	8 x 3 metre marquee	Tent	Approx (10 x 26ft)
6	6 x 4 metre marquee	Tent	Approx (13 x 20ft)
7	8 x 4 metre marquee	Tent	Approx (26 x 13ft)
8	10 x 4 metre marquee	Tent	Approx (32 x 13ft)
9	6 x 6 metre marquee	Tent	Approx (20 x 2Oft)
10	8 x 6 metre marquee	Tent	Approx (26 x 20ft)
11	10 x 6 metre marquee	Tent	Approx (32 x 20ft)
12	12 x 6 metre marquee	Tent	Approx (40 x 20ft)
13	4x4m/6x3m/6x4m Hardstanding kit	Floor	
14	8x4m/1Ox4m/6x6m Hardstanding kit	Floor	
15	10x6m/12x6m Hardstanding kit	Floor	
16	3 X3 metre matting	Floor	
17	4 x 4 metre matting	Floor	
18	6 x 3 metre matting	Floor	
19	6 x 4 metre matting	Floor	
20	8 x 3 metre matting	Floor	
21	8 x 4 metre matting	Floor	
22	10 x 4 metre matting	Floor	

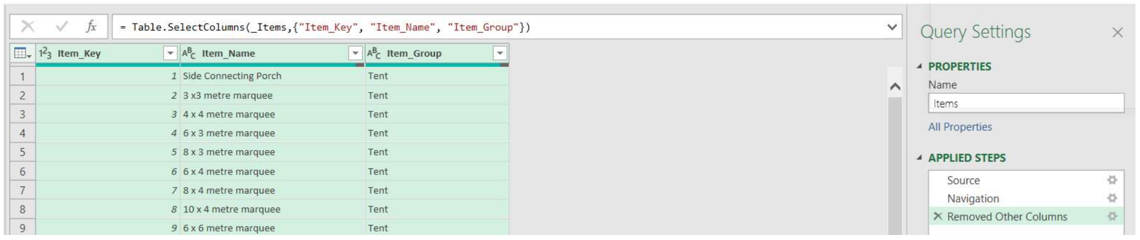
I select the 'Transform Data' option.



```
= Source{[Schema="",Item="Items"]}[Data]
```

	Item_Key	Item_Name	Item_Group	Dimensions	Standing_Capacity	Seated_Capacity
1	1	Side Connecting Porch	Tent	Approx (2m x 2.2m)	null	
2	2	3 x3 metre marquee	Tent	Approx (10 x 10ft)	14	
3	3	4 x 4 metre marquee	Tent	Approx (13 x 13ft)	22	
4	4	6 x 3 metre marquee	Tent	Approx (10 x 20ft)	26	
5	5	8 x 3 metre marquee	Tent	Approx (10 x 26ft)	35	
6	6	6 x 4 metre marquee	Tent	Approx (13 x 20ft)	35	
7	7	8 x 4 metre marquee	Tent	Approx (26 x 13ft)	45	
8	8	10 x 4 metre marquee	Tent	Approx (32 x 13ft)	60	
9	9	6 x 6 metre marquee	Tent	Approx (20 x 2Oft)	50	
10	10	8 x 6 metre marquee	Tent	Approx (26 x 20ft)	70	
11	11	10 x 6 metre marquee	Tent	Approx (32 x 20ft)	90	
12	12	12 x 6 metre marquee	Tent	Approx (40 x 20ft)	110	
13	13	4x4m/6x3m/6x4m Hardstanding kit	Floor	null	null	
14	14	8x4m/1Ox4m/6x6m Hardstanding kit	Floor	null	null	

I select **Item_Key, Item_Group** and **Item_Name** and right-click to 'Remove other Columns'. These are the fields I will need to merge the data with **ACCT_Order_Charges**.

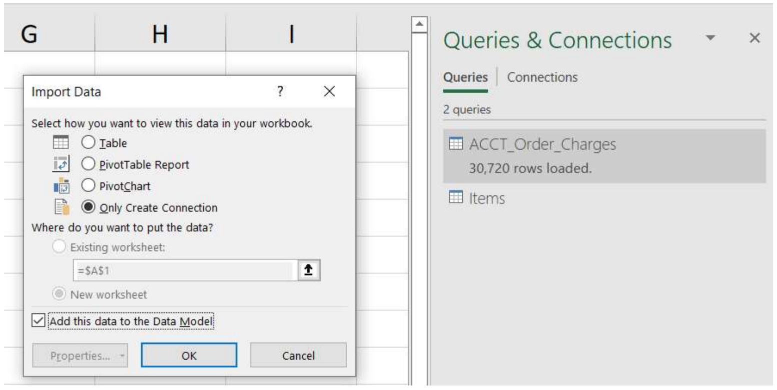


```
= Table.SelectColumns(_Items,{"Item_Key", "Item_Name", "Item_Group"})
```

	Item_Key	Item_Name	Item_Group
1	1	Side Connecting Porch	Tent
2	2	3 x3 metre marquee	Tent
3	3	4 x 4 metre marquee	Tent
4	4	6 x 3 metre marquee	Tent
5	5	8 x 3 metre marquee	Tent
6	6	6 x 4 metre marquee	Tent
7	7	8 x 4 metre marquee	Tent
8	8	10 x 4 metre marquee	Tent
9	9	6 x 6 metre marquee	Tent

Query Settings
PROPERTIES
Name: Items
All Properties
APPLIED STEPS
Source
Navigation
Removed Other Columns

I do not wish to make further changes, so from the Home tab I select ‘Close & Load to…’ from the ‘Close & Load’ button:

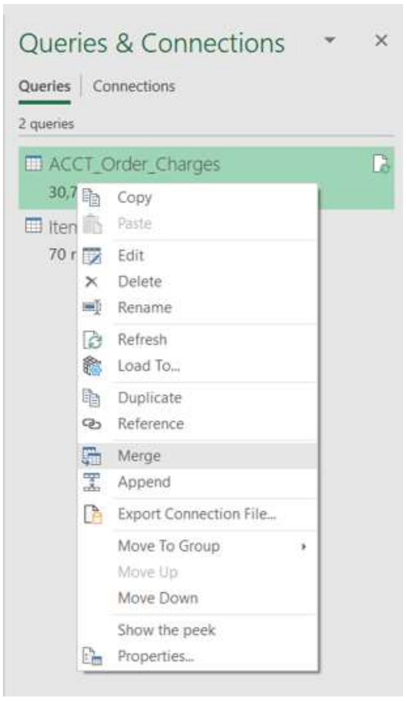


Import Data
Select how you want to view this data in your workbook.
Table
PivotTable Report
PivotChart
[✓ Only Create Connection]
Where do you want to put the data?
Existing worksheet:
=$A$1
New worksheet
[✓ Add this data to the Data Model]
Properties… | OK | Cancel

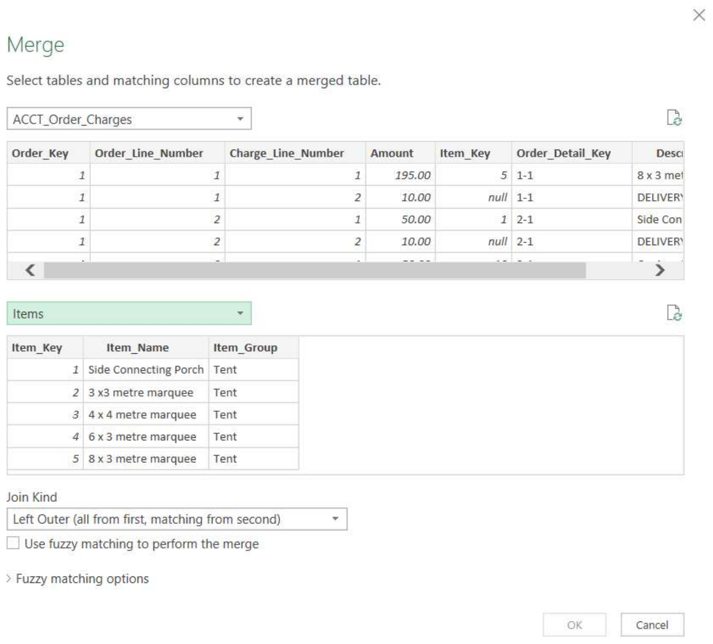
Queries & Connections
Queries | Connections
2 queries
ACCT_Order_Charges
30,720 rows loaded.
Items

Note that the Items query appears in the ‘Queries & Connections’ pane, but is not complete yet. Having made my selections as shown above, I choose to ‘Load’.

There are several ways I can access the option to merge my queries. I can double-click on either query to access the Power Query Editor, or I can right-click on one of the queries:



The Merge dialog appears where I choose my two queries:



Merge

Select tables and matching columns to create a merged table.

ACCT_Order_Charges

Order_Key	Order_Line_Number	Charge_Line_Number	Amount	Item_Key	Order_Detail_Key	Desc
1	1	1	195.00	5	1-1	8 x 3 met
1	1	2	10.00	null	1-1	DELIVERY
1	2	1	50.00	1	2-1	Side Con
1	2	2	10.00	null	2-1	DELIVERY

Items

Item_Key	Item_Name	Item_Group
1	Side Connecting Porch	Tent
2	3 x3 metre marquee	Tent
3	4 x 4 metre marquee	Tent
4	6 x 3 metre marquee	Tent
5	8 x 3 metre marquee	Tent

Join Kind

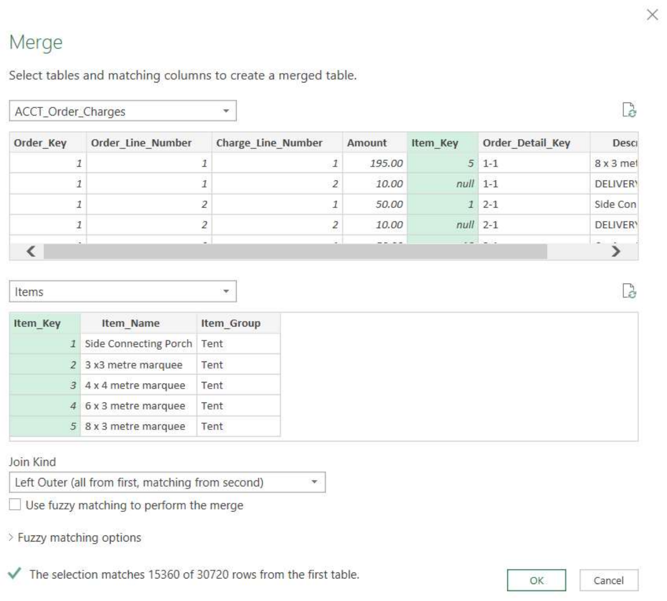
Left Outer (all from first, matching from second)

Use fuzzy matching to perform the merge

Fuzzy matching options

The dropdown at the bottom of the screen allows me to choose how to link the tables. In this case, I choose ‘Left Outer’, as I want all of the **ACCT_Order_Charges** table and matching data from Items (if I had input Items first then I would pick ‘Right Outer’). This is the point where some understanding of the data is required; picking ‘Full Outer’ may lead to duplicates.

The ‘OK’ button is not enabled yet. This is because I need to pick the columns to join. In this case, I want to join **Item_Key** in both tables, so I select this field on each table. When I select the same number of columns in each table, then ‘OK’ is enabled, and the number of connections is displayed:



Merge

Select tables and matching columns to create a merged table.

ACCT_Order_Charges

Order_Key	Order_Line_Number	Charge_Line_Number	Amount	Item_Key	Order_Detail_Key	Desc
1	1	1	195.00	5	1-1	8 x 3 met
1	1	2	10.00	null	1-1	DELIVERY
1	2	1	50.00	1	2-1	Side Con
1	2	2	10.00	null	2-1	DELIVERY

Items

Item_Key	Item_Name	Item_Group
1	Side Connecting Porch	Tent
2	3 x3 metre marquee	Tent
3	4 x 4 metre marquee	Tent
4	6 x 3 metre marquee	Tent
5	8 x 3 metre marquee	Tent

Join Kind

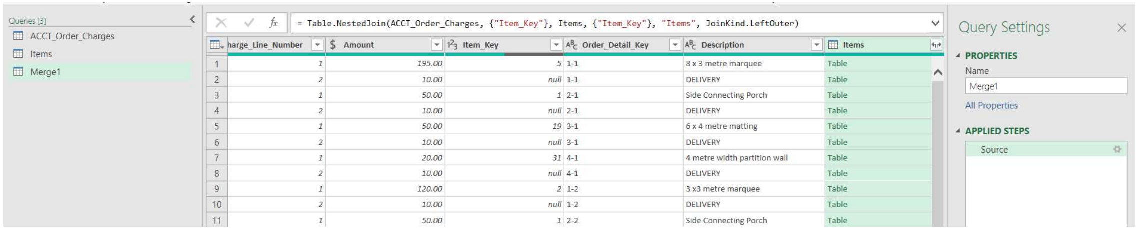
Left Outer (all from first, matching from second)

Use fuzzy matching to perform the merge

Fuzzy matching options

✓ The selection matches 15360 of 30720 rows from the first table.

I click OK, and am taken into the Power Query Editor:



Queries [3]
- ACCT_Order_Charges
- Items
- Merge1

```
= Table.NestedJoin(ACCT_Order_Charges, {"Item_Key"}, Items, {"Item_Key"}, "Items", JoinKind.LeftOuter)
```

	harge_Line_Number	Amount	Item_Key	Order_Detail_Key	Description	Items
1	1	195.00	5	1-1	8 x 3 metre marquee	Table
2	2	10.00	null	1-1	DELIVERY	Table
3	1	50.00	1	2-1	Side Connecting Porch	Table
4	2	10.00	null	2-1	DELIVERY	Table
5	1	50.00	19	3-1	6 x 4 metre matting	Table
6	2	10.00	null	3-1	DELIVERY	Table
7	1	20.00	31	4-1	4 metre width partition wall	Table
8	2	10.00	null	4-1	DELIVERY	Table
9	1	120.00	2	1-2	3 x3 metre marquee	Table
10	2	10.00	null	1-2	DELIVERY	Table
11	1	50.00	1	2-2	Side Connecting Porch	Table

Query Settings

PROPERTIES

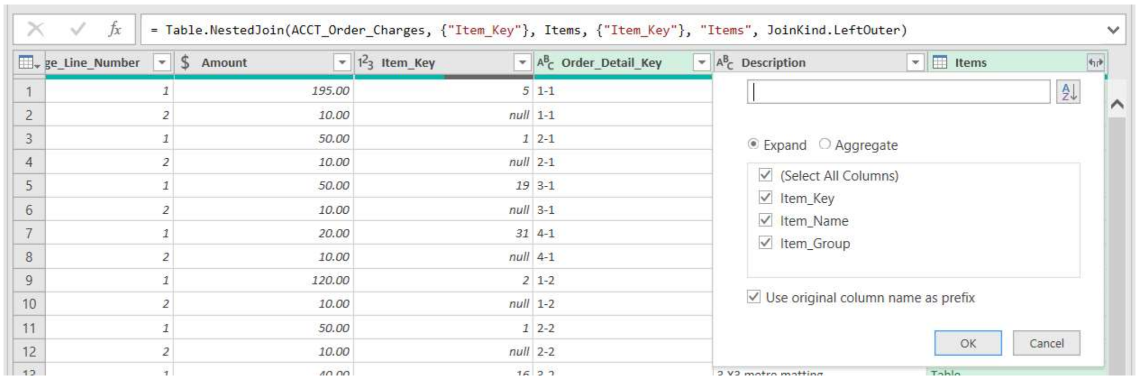
Name: Merge1

All Properties

APPLIED STEPS

- Source

Merge1 contains the columns from ***ACCT_Order_Charges***, and a new column called **Items**. Note the icon next to it with two arrows pointing away from each other: this will expand to show the new columns available:



```
= Table.NestedJoin(ACCT_Order_Charges, {"Item_Key"}, Items, {"Item_Key"}, "Items", JoinKind.LeftOuter)
```

	ge_Line_Number	Amount	Item_Key	Order_Detail_Key
1	1	195.00	5	1-1
2	2	10.00	null	1-1
3	1	50.00	1	2-1
4	2	10.00	null	2-1
5	1	50.00	19	3-1
6	2	10.00	null	3-1
7	1	20.00	31	4-1
8	2	10.00	null	4-1
9	1	120.00	2	1-2
10	2	10.00	null	1-2
11	1	50.00	1	2-2
12	2	10.00	null	2-2

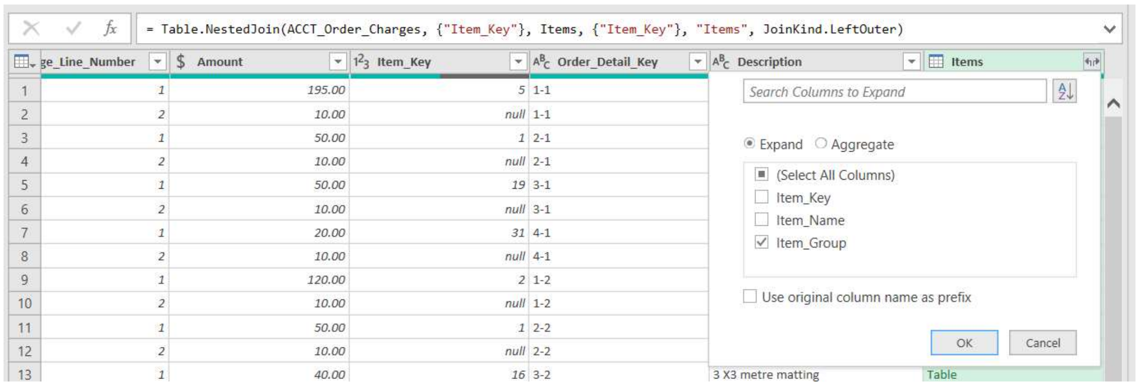
[✓ Expand] Aggregate

- [✓] (Select All Columns)
- [✓] Item_Key
- [✓] Item_Name
- [✓] Item_Group

[✓] Use original column name as prefix

OK Cancel

I can uncheck columns that are already in the first table to prevent duplication. In this case description is very similar to **Item_Name** so I uncheck **Item_Name** and **Item_Key** (since I only included the key to enable me to link the table data).



```
= Table.NestedJoin(ACCT_Order_Charges, {"Item_Key"}, Items, {"Item_Key"}, "Items", JoinKind.LeftOuter)
```

	ge_Line_Number	Amount	Item_Key	Order_Detail_Key	Description	Items
1	1	195.00	5	1-1		
2	2	10.00	null	1-1		
3	1	50.00	1	2-1		
4	2	10.00	null	2-1		
5	1	50.00	19	3-1		
6	2	10.00	null	3-1		
7	1	20.00	31	4-1		
8	2	10.00	null	4-1		
9	1	120.00	2	1-2		
10	2	10.00	null	1-2		
11	1	50.00	1	2-2		
12	2	10.00	null	2-2		
13	1	40.00	16	3-2	3 X3 metre matting	Table

Search Columns to Expand

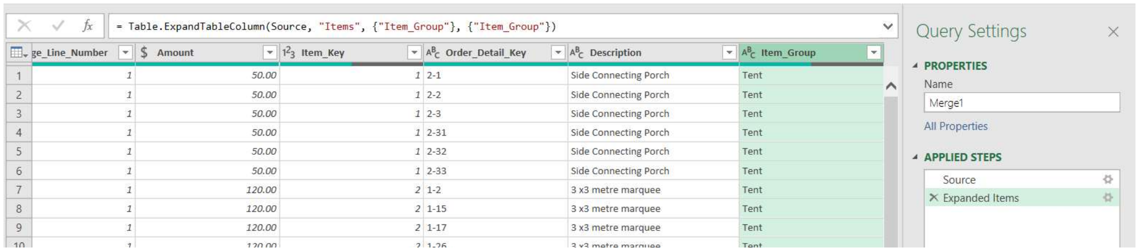
[✓ Expand] Aggregate

- [■] (Select All Columns)
- [ ] Item_Key
- [ ] Item_Name
- [✓] Item_Group

[ ] Use original column name as prefix

OK Cancel

I unselect 'Use original column name as prefix'. I would only need to choose this option if the table already contained a column with the same name.



```
= Table.ExpandTableColumn(Source, "Items", {"Item_Group"}, {"Item_Group"})
```

	ge_Line_Number	Amount	Item_Key	Order_Detail_Key	Description	Item_Group
1	1	50.00	1	2-1	Side Connecting Porch	Tent
2	1	50.00	1	2-2	Side Connecting Porch	Tent
3	1	50.00	1	2-3	Side Connecting Porch	Tent
4	1	50.00	1	2-31	Side Connecting Porch	Tent
5	1	50.00	1	2-32	Side Connecting Porch	Tent
6	1	50.00	1	2-33	Side Connecting Porch	Tent
7	1	120.00	2	1-2	3 x3 metre marquee	Tent
8	1	120.00	2	1-15	3 x3 metre marquee	Tent
9	1	120.00	2	1-17	3 x3 metre marquee	Tent
10	1	120.00	2	1-26	3 x3 metre marquee	Tent

Query Settings

PROPERTIES

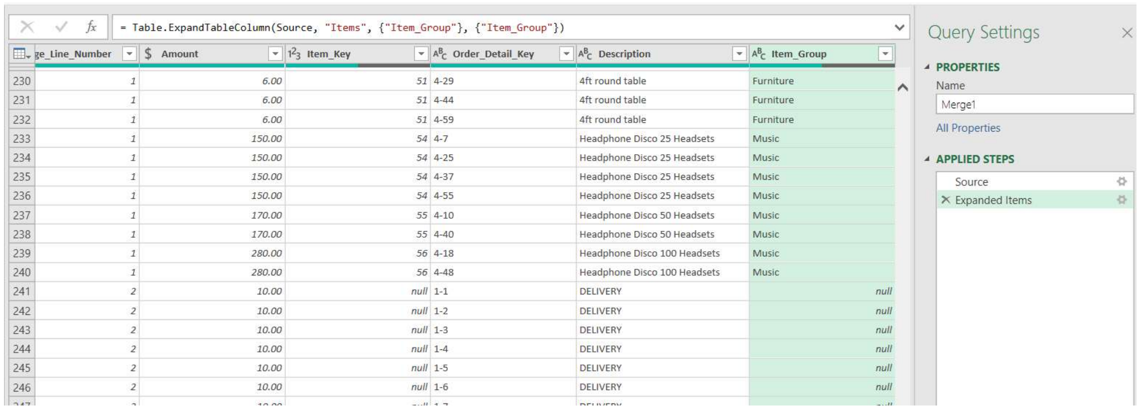
Name: Merge1

All Properties

APPLIED STEPS

- Source
- Expanded Items

Note that the order of the rows has changed because the non-linked rows which had no item key have moved to the bottom.



```
= Table.ExpandTableColumn(Source, "Items", {"Item_Group"}, {"Item_Group"})
```

	ge_Line_Number	Amount	Item_Key	Order_Detail_Key	Description	Item_Group
230	1	6.00	51	4-29	4ft round table	Furniture
231	1	6.00	51	4-44	4ft round table	Furniture
232	1	6.00	51	4-59	4ft round table	Furniture
233	1	150.00	54	4-7	Headphone Disco 25 Headsets	Music
234	1	150.00	54	4-25	Headphone Disco 25 Headsets	Music
235	1	150.00	54	4-37	Headphone Disco 25 Headsets	Music
236	1	150.00	54	4-55	Headphone Disco 25 Headsets	Music
237	1	170.00	55	4-10	Headphone Disco 50 Headsets	Music
238	1	170.00	55	4-40	Headphone Disco 50 Headsets	Music
239	1	280.00	56	4-18	Headphone Disco 100 Headsets	Music
240	1	280.00	56	4-48	Headphone Disco 100 Headsets	Music
241	2	10.00	null	1-1	DELIVERY	null
242	2	10.00	null	1-2	DELIVERY	null
243	2	10.00	null	1-3	DELIVERY	null
244	2	10.00	null	1-4	DELIVERY	null
245	2	10.00	null	1-5	DELIVERY	null
246	2	10.00	null	1-6	DELIVERY	null

Query Settings

PROPERTIES

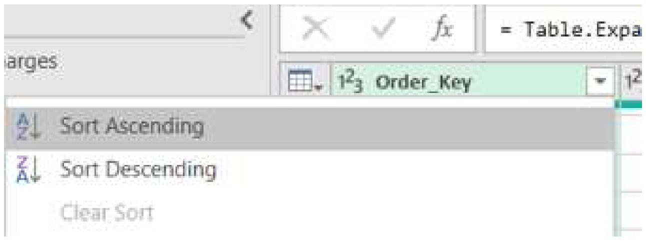
Name: Merge1

All Properties

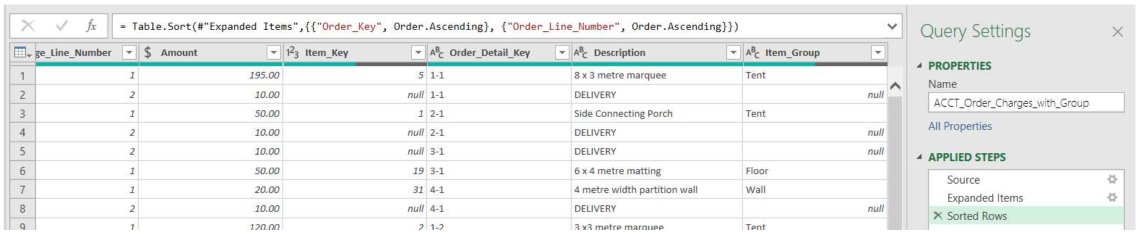
APPLIED STEPS

- Source
- Expanded Items

I can fix this by sorting on **Order_Key** and **Order_Line_Number**. There is a downward arrow icon next to each column heading which I can click and then choose to sort in ascending order.



Having done this for both columns, Power Query combines these sorts into one step. I also choose to rename the table to something more meaningful than **Merge1**.



```
= Table.Sort(#"Expanded Items",{{"Order_Key", Order.Ascending}, {"Order_Line_Number", Order.Ascending}})
```

	ge_Line_Number	Amount	Item_Key	Order_Detail_Key	Description	Item_Group
1	1	195.00	5	1-1	8 x 3 metre marquee	Tent
2	2	10.00	null	1-1	DELIVERY	null
3	1	50.00	1	2-1	Side Connecting Porch	Tent
4	2	10.00	null	2-1	DELIVERY	null
5	2	10.00	null	3-1	DELIVERY	null
6	1	50.00	19	3-1	6 x 4 metre matting	Floor
7	1	20.00	31	4-1	4 metre width partition wall	Wall
8	2	10.00	null	4-1	DELIVERY	null
9	1	120.00	2	1-2	3 x3 metre marquee	Tent

Query Settings

PROPERTIES

Name: ACCT_Order_Charges_with_Group

All Properties

APPLIED STEPS

- Source
- Expanded Items
- Sorted Rows

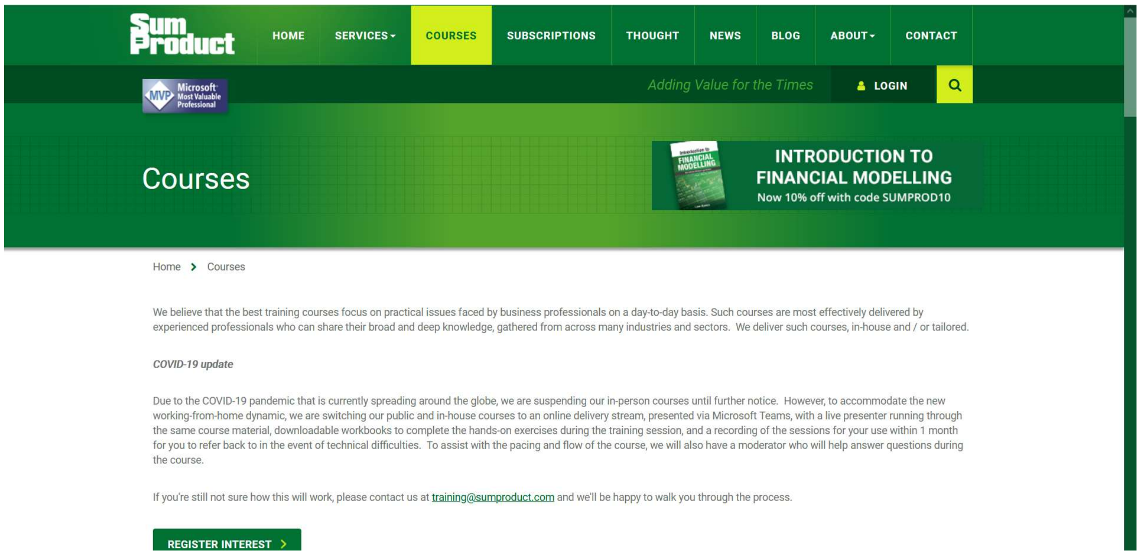
I can now 'Close & Load' to create a new merged table, which can be used in Excel workbooks and the Excel data model. When merging queries do keep in mind the warnings above; if the queries are for use by Power Pivot only, then consider whether it would be more useful to load one merged table or separate tables that may be managed in Power Pivot. However, for creating Excel workbooks, merging is a useful way of pulling in data from many tables in order to view the data in a single table. Data may of course come from other external sources.

Next, I will look at extracting data from a website…

Extracting from a Website

Whilst Power Query will allow data to be extracted from the web, for many webpages, some knowledge of HTML is needed, and even then, a great deal of transformation is often required to get the data into a tabular form. However, if the page uses tables, then the data can be much easier to extract. Today, I show how to extract data from a webpage that holds information in a table.

For my example, I will use an excellent webpage which provides a list of training events by Excel specialists: perhaps it looks familiar?



Courses

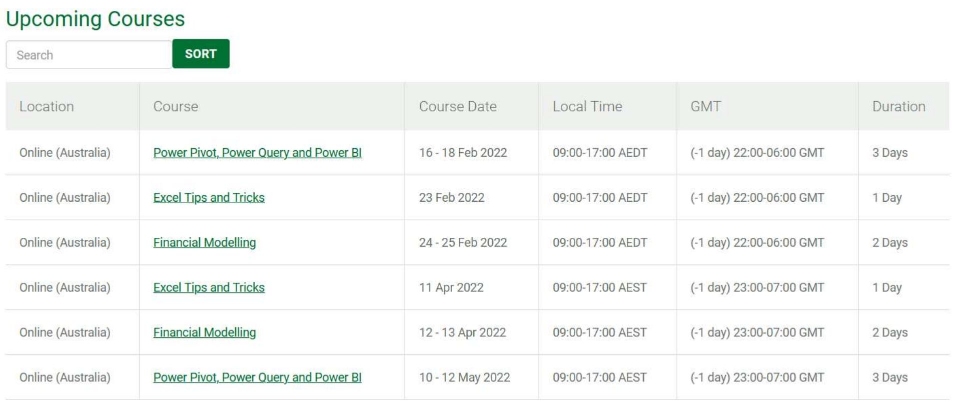
We believe that the best training courses focus on practical issues faced by business professionals on a day-to-day basis. Such courses are most effectively delivered by experienced professionals who can share their broad and deep knowledge, gathered from across many industries and sectors. We deliver such courses, in-house and / or tailored.

COVID-19 update

Due to the COVID-19 pandemic that is currently spreading around the globe, we are suspending our in-person courses until further notice. However, to accommodate the new working-from-home dynamic, we are switching our public and in-house courses to an online delivery stream, presented via Microsoft Teams, with a live presenter running through the same course material, downloadable workbooks to complete the hands-on exercises during the training session, and a recording of the sessions for your use within 1 month for you to refer back to in the event of technical difficulties. To assist with the pacing and flow of the course, we will also have a moderator who will help answer questions during the course.

If you're still not sure how this will work, please contact us at training@sumproduct.com and we'll be happy to walk you through the process.

On this page, there is a table of upcoming courses:

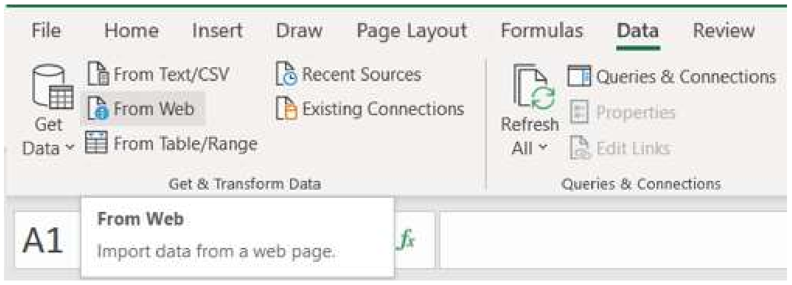


Upcoming Courses

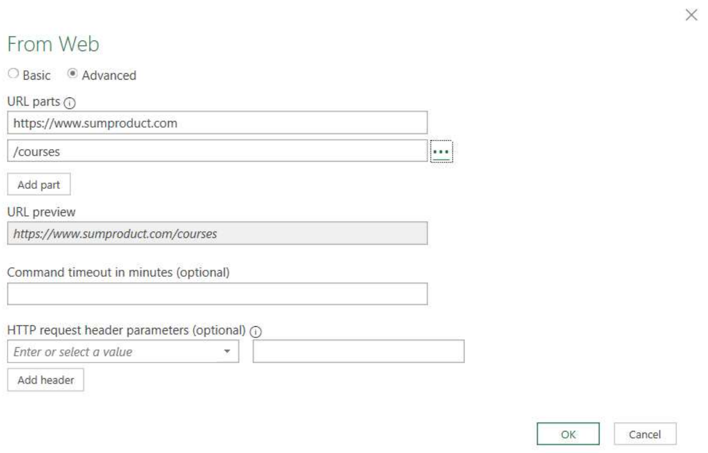
Search SORT

Location	Course	Course Date	Local Time	GMT	Duration
Online (Australia)	Power Pivot, Power Query and Power BI	16 - 18 Feb 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	3 Days
Online (Australia)	Excel Tips and Tricks	23 Feb 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	1 Day
Online (Australia)	Financial Modelling	24 - 25 Feb 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	2 Days
Online (Australia)	Excel Tips and Tricks	11 Apr 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	1 Day
Online (Australia)	Financial Modelling	12 - 13 Apr 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	2 Days
Online (Australia)	Power Pivot, Power Query and Power BI	10 - 12 May 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	3 Days

I start in the 'Get & Transform Data' section on the Data tab, by choosing to get my data 'From Web'.



The window that pops up gives me a choice of 'Basic' or 'Advanced' options. I could have used the basic option since my full web page is fairly short, but I have chosen the advanced option to show how webpage addresses may be built.



From Web

Basic [✓ Advanced]

URL parts
https://www.sumproduct.com
/courses

Add part

URL preview
https://www.sumproduct.com/courses

Command timeout in minutes (optional)

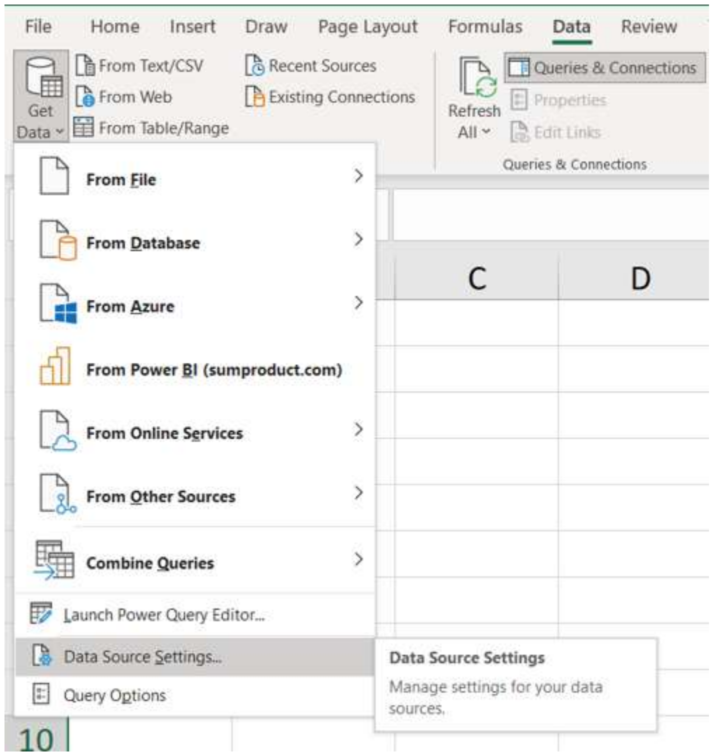
HTTP request header parameters (optional)
Enter or select a value

Add header

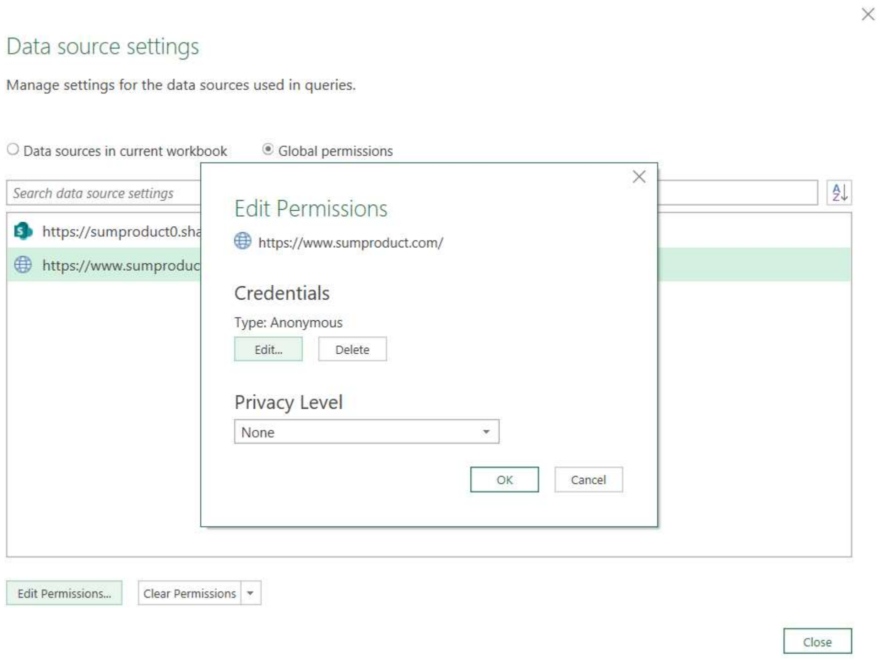
OK Cancel

I have chosen to access https://www.sumproduct.com/courses; I can opt to set timeouts if I find I am having problems with a particular website, but I will choose not to set any in this example (since I know how great this website is!).

If I were accessing www.sumproduct.com for the first time, then I would be prompted to confirm the authentication options, which default to 'Anonymous'. It is possible to see the same authentication options from 'Data Source Settings' at the bottom of the 'Get Data' dropdown:



I select the settings for SumProduct:


Data source settings

Manage settings for the data sources used in queries.

○ Data sources in current workbook [✓ Global permissions]

Edit Permissions

https://www.sumproduct.com/

Credentials

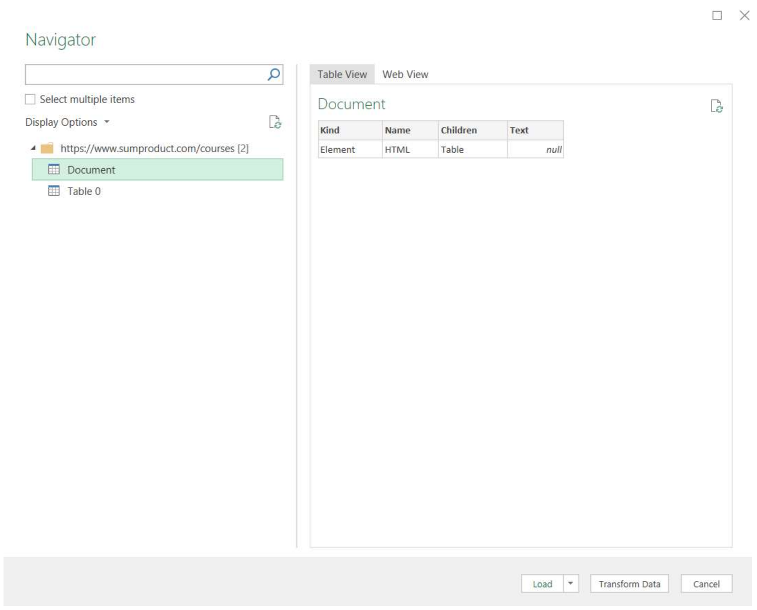
Type: Anonymous

Privacy Level

None

I have chosen to use the anonymous setting, since the webpage I am viewing does not require any password or other security information. Future queries accessing the same website will use the existing authentication settings.

Once the authentication method has been defined, Power Query links to the webpage and returns the recognised content.



Document

Kind	Name	Children	Text
Element	HTML	Table	*null*

The navigator pane shows that Power Query has identified a document and a table. The document highlighted clearly doesn't show any recognisable data about courses. More work would be required in order to obtain useful information from this. The table, however, proves more fruitful, as shown below:

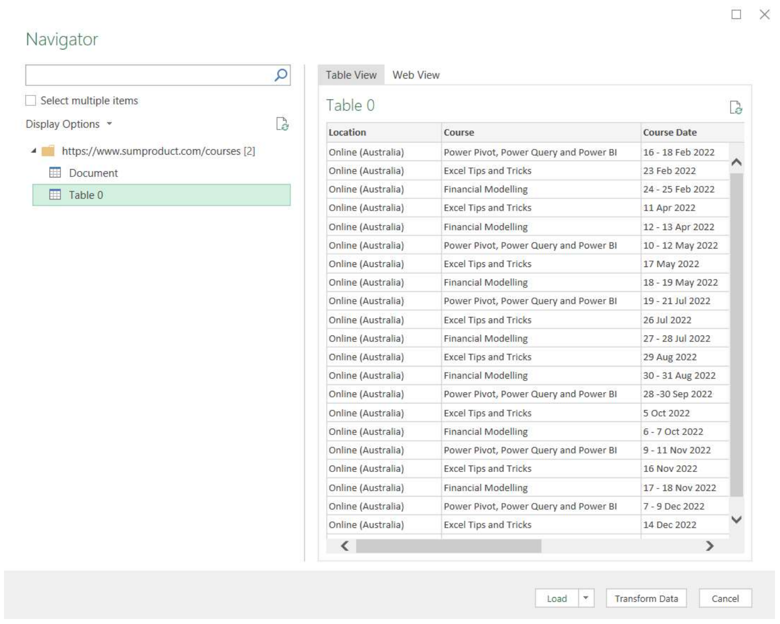
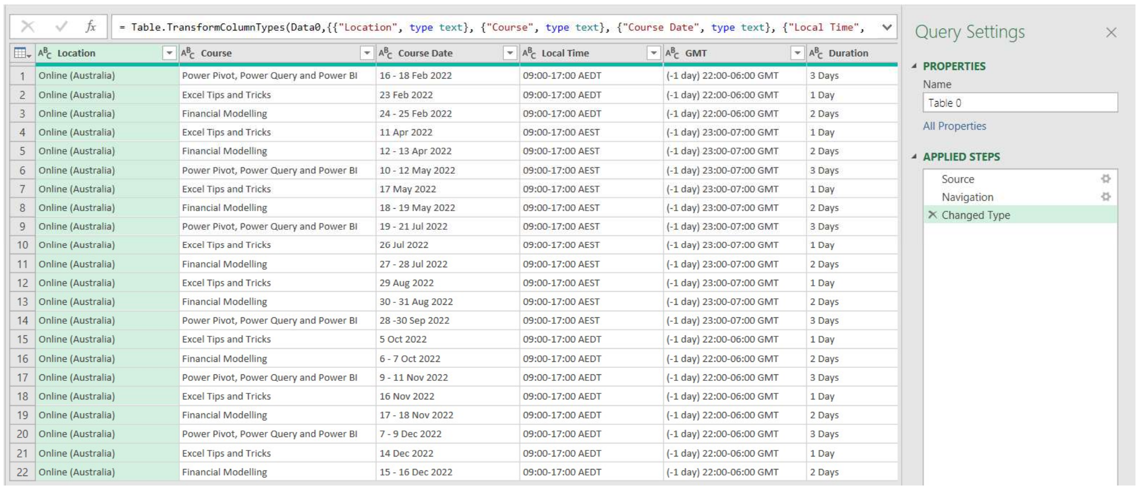


Table 0

Location	Course	Course Date
Online (Australia)	Power Pivot, Power Query and Power BI	16 - 18 Feb 2022
Online (Australia)	Excel Tips and Tricks	23 Feb 2022
Online (Australia)	Financial Modelling	24 - 25 Feb 2022
Online (Australia)	Excel Tips and Tricks	11 Apr 2022
Online (Australia)	Financial Modelling	12 - 13 Apr 2022
Online (Australia)	Power Pivot, Power Query and Power BI	10 - 12 May 2022
Online (Australia)	Excel Tips and Tricks	17 May 2022
Online (Australia)	Financial Modelling	18 - 19 May 2022
Online (Australia)	Power Pivot, Power Query and Power BI	19 - 21 Jul 2022
Online (Australia)	Excel Tips and Tricks	26 Jul 2022
Online (Australia)	Financial Modelling	27 - 28 Jul 2022
Online (Australia)	Excel Tips and Tricks	29 Aug 2022
Online (Australia)	Financial Modelling	30 - 31 Aug 2022
Online (Australia)	Power Pivot, Power Query and Power BI	28 -30 Sep 2022
Online (Australia)	Excel Tips and Tricks	5 Oct 2022
Online (Australia)	Financial Modelling	6 - 7 Oct 2022
Online (Australia)	Power Pivot, Power Query and Power BI	9 - 11 Nov 2022
Online (Australia)	Excel Tips and Tricks	16 Nov 2022
Online (Australia)	Financial Modelling	17 - 18 Nov 2022
Online (Australia)	Power Pivot, Power Query and Power BI	7 - 9 Dec 2022
Online (Australia)	Excel Tips and Tricks	14 Dec 2022

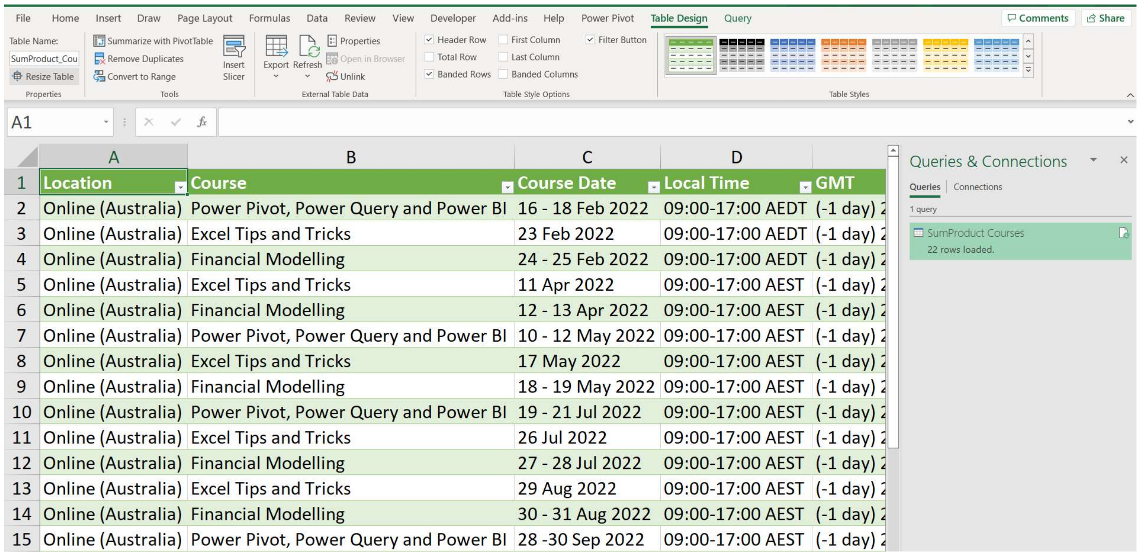
I have a list of courses, and I choose to 'Transform Data' ready to load to an Excel workbook.



```
= Table.TransformColumnTypes(Data0,{{"Location", type text}, {"Course", type text}, {"Course Date", type text}, {"Local Time",
```

	Location	Course	Course Date	Local Time	GMT	Duration
1	Online (Australia)	Power Pivot, Power Query and Power BI	16 - 18 Feb 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	3 Days
2	Online (Australia)	Excel Tips and Tricks	23 Feb 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	1 Day
3	Online (Australia)	Financial Modelling	24 - 25 Feb 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	2 Days
4	Online (Australia)	Excel Tips and Tricks	11 Apr 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	1 Day
5	Online (Australia)	Financial Modelling	12 - 13 Apr 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	2 Days
6	Online (Australia)	Power Pivot, Power Query and Power BI	10 - 12 May 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	3 Days
7	Online (Australia)	Excel Tips and Tricks	17 May 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	1 Day
8	Online (Australia)	Financial Modelling	18 - 19 May 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	2 Days
9	Online (Australia)	Power Pivot, Power Query and Power BI	19 - 21 Jul 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	3 Days
10	Online (Australia)	Excel Tips and Tricks	26 Jul 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	1 Day
11	Online (Australia)	Financial Modelling	27 - 28 Jul 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	2 Days
12	Online (Australia)	Excel Tips and Tricks	29 Aug 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	1 Day
13	Online (Australia)	Financial Modelling	30 - 31 Aug 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	2 Days
14	Online (Australia)	Power Pivot, Power Query and Power BI	28 -30 Sep 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	3 Days
15	Online (Australia)	Excel Tips and Tricks	5 Oct 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	1 Day
16	Online (Australia)	Financial Modelling	6 - 7 Oct 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	2 Days
17	Online (Australia)	Power Pivot, Power Query and Power BI	9 - 11 Nov 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	3 Days
18	Online (Australia)	Excel Tips and Tricks	16 Nov 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	1 Day
19	Online (Australia)	Financial Modelling	17 - 18 Nov 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	2 Days
20	Online (Australia)	Power Pivot, Power Query and Power BI	7 - 9 Dec 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	3 Days
21	Online (Australia)	Excel Tips and Tricks	14 Dec 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	1 Day
22	Online (Australia)	Financial Modelling	15 - 16 Dec 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	2 Days

It all looks good, so I rename the query SumProduct Courses and 'Close & Load'.



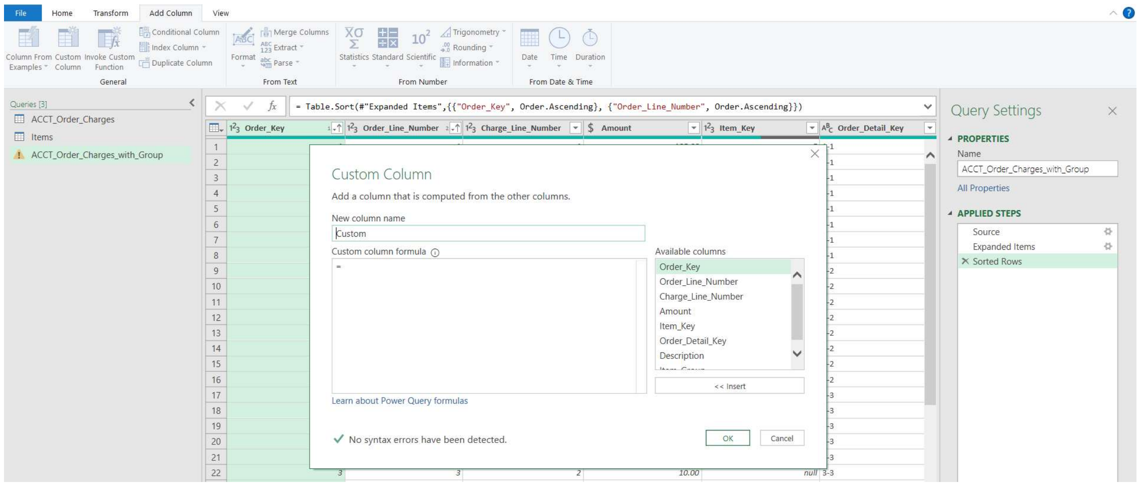
	A	B	C	D	
1	Location	Course	Course Date	Local Time	GMT
2	Online (Australia)	Power Pivot, Power Query and Power BI	16 - 18 Feb 2022	09:00-17:00 AEDT	(-1 day)
3	Online (Australia)	Excel Tips and Tricks	23 Feb 2022	09:00-17:00 AEDT	(-1 day)
4	Online (Australia)	Financial Modelling	24 - 25 Feb 2022	09:00-17:00 AEDT	(-1 day)
5	Online (Australia)	Excel Tips and Tricks	11 Apr 2022	09:00-17:00 AEST	(-1 day)
6	Online (Australia)	Financial Modelling	12 - 13 Apr 2022	09:00-17:00 AEST	(-1 day)
7	Online (Australia)	Power Pivot, Power Query and Power BI	10 - 12 May 2022	09:00-17:00 AEST	(-1 day)
8	Online (Australia)	Excel Tips and Tricks	17 May 2022	09:00-17:00 AEST	(-1 day)
9	Online (Australia)	Financial Modelling	18 - 19 May 2022	09:00-17:00 AEST	(-1 day)
10	Online (Australia)	Power Pivot, Power Query and Power BI	19 - 21 Jul 2022	09:00-17:00 AEST	(-1 day)
11	Online (Australia)	Excel Tips and Tricks	26 Jul 2022	09:00-17:00 AEST	(-1 day)
12	Online (Australia)	Financial Modelling	27 - 28 Jul 2022	09:00-17:00 AEST	(-1 day)
13	Online (Australia)	Excel Tips and Tricks	29 Aug 2022	09:00-17:00 AEST	(-1 day)
14	Online (Australia)	Financial Modelling	30 - 31 Aug 2022	09:00-17:00 AEST	(-1 day)
15	Online (Australia)	Power Pivot, Power Query and Power BI	28 -30 Sep 2022	09:00-17:00 AEST	(-1 day)

The link to the webpage may be refreshed in the same way as other data sources. A word of caution, however: in this case, I have used a reliable source, which I know will be well maintained, and which will not be subject to abrupt format changes without warning. Using content which I have no control over is risky and can lead to bad decisions being taken as a result of using out of date or erroneous data.

Introduction to M

For users already familiar with Power Pivot and Data Analysis eXpressions (DAX, the associated programming language), it might seem logical that Power Query would use a similar language and perhaps even the same formulae. Not so. Power Query has its own language, **M**, and its own formula syntax. So, having abandoned any expectation of familiarity, I need a good place to start looking at **M** language. I will create a custom column and look at a formula that can be associated with that new column.

I start out in the worksheet for the merged query I created earlier and open **ACCT_Order_Charges_with_Group** to access the query editor. On the ‘Add Column’ tab, I choose to ‘Custom Column’:



```
= Table.Sort(#"Expanded Items",{{"Order_Key", Order.Ascending}, {"Order_Line_Number", Order.Ascending}})
```

Custom Column

Add a column that is computed from the other columns.

New column name: Custom

Custom column formula: =

Available columns: Order_Key, Order_Line_Number, Charge_Line_Number, Amount, Item_Key, Order_Detail_Key, Description

Learn about Power Query formulas

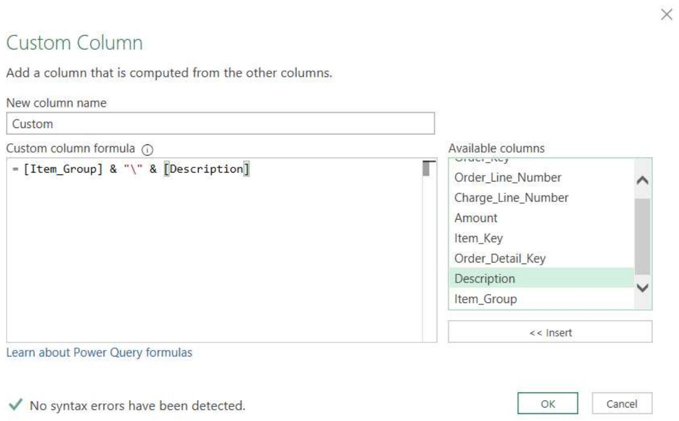
No syntax errors have been detected.

The dialog box has a large section for the formula beneath the name I choose for my new custom column. Available columns in the query are shown and double clicking them adds them to the formula (or I can select and then choose to ‘Insert’ them).

Notice the option to ‘Learn about Power Query formulas’ at the bottom of the dialog box. This is the best place to find out what formulae are available in Power Query. Clicking here will take me to the Microsoft help page, which has a links to explain the Power Query **M** language and the functions available.

Having said that the formulae do not tend to match those for Power Pivot, there are some functions that are reassuringly familiar from Excel, as I will show now by concatenating two existing columns.

I decide to create a column that combines **Item_Group** and **Description** by double clicking each column (or using ‘Insert’). I type in an ‘&’ between the columns, which is the same as I would do in an Excel formula, and include a ‘/’ to separate the data in the column to make it easier to read:



Custom Column

Add a column that is computed from the other columns.

New column name: Custom

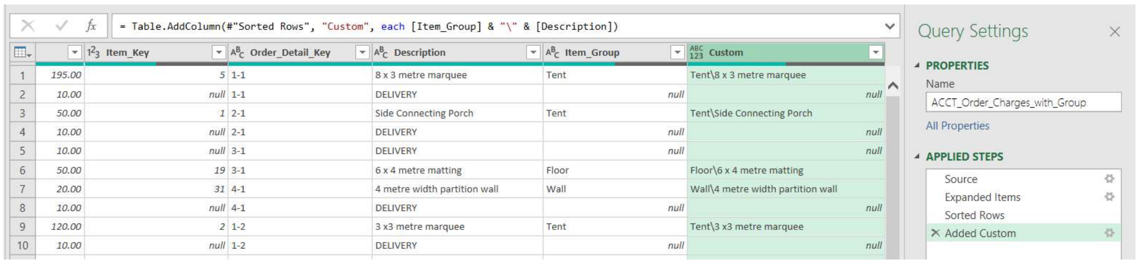
Custom column formula:

```
= [Item_Group] & "\" & [Description]
```

Available columns: Order_Line_Number, Charge_Line_Number, Amount, Item_Key, Order_Detail_Key, Description, Item_Group

No syntax errors have been detected.

I click 'OK' and my custom column is generated, ready to be loaded to my worksheet.



```
= Table.AddColumn(#"Sorted Rows", "Custom", each [Item_Group] & "\" & [Description])
```

		Item_Key	Order_Detail_Key	Description	Item_Group	Custom
1	195.00	5	1-1	8 x 3 metre marquee	Tent	Tent\8 x 3 metre marquee
2	10.00	null	1-1	DELIVERY	null	null
3	50.00	1	2-1	Side Connecting Porch	Tent	Tent\Side Connecting Porch
4	10.00	null	2-1	DELIVERY	null	null
5	10.00	null	3-1	DELIVERY	null	null
6	50.00	19	3-1	6 x 4 metre matting	Floor	Floor\6 x 4 metre matting
7	20.00	31	4-1	4 metre width partition wall	Wall	Wall\4 metre width partition wall
8	10.00	null	4-1	DELIVERY	null	null
9	120.00	2	1-2	3 x3 metre marquee	Tent	Tent\3 x3 metre marquee
10	10.00	null	1-2	DELIVERY	null	null

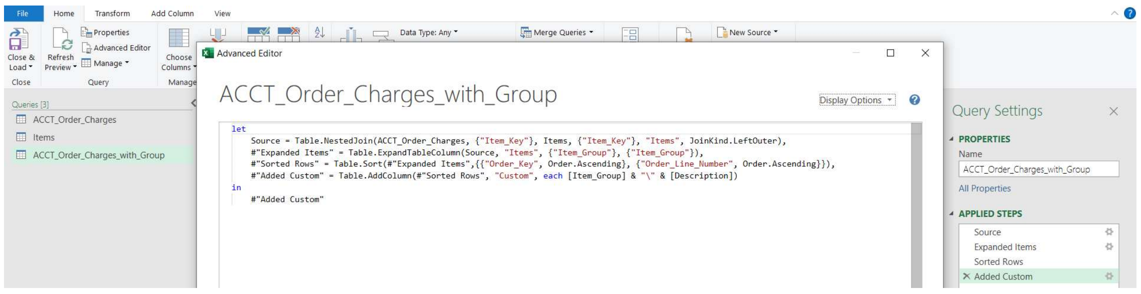
Query Settings — PROPERTIES — Name: ACCT_Order_Charges_with_Group — APPLIED STEPS: Source, Expanded Items, Sorted Rows, Added Custom

There are other similarities with Excel: the symbols '+', '-', '*' and '/' are used for add, subtract, multiply and divide respectively too.

There are however some points to bear in mind when comparing Power Query formulae with Excel:

- Excel formulae are not case sensitive, but Power Query formulae are (this is a classic *gotcha*)
- Excel counts using a base of one [1] (*e.g.* the first letter in a string is at position 1), but Power Query uses a base of zero [0] (so the same letter would be at position 0)
- Excel will automatically convert data (*e.g.* concatenating a text column to a numerical column will work as Excel converts them to text automatically); Power Query will not (*e.g.* trying to concatenate text to a value will give errors in the new column – the value must be converted to text first). This is why I picked two text columns for my example above.

If I go to the 'Advanced Editor' on the Home tab, I can see all the code for the current query.



Advanced Editor

ACCT_Order_Charges_with_Group

```
let
    Source = Table.NestedJoin(ACCT_Order_Charges, {"Item_Key"}, Items, {"Item_Key"}, "Items", JoinKind.LeftOuter),
    #"Expanded Items" = Table.ExpandTableColumn(Source, "Items", {"Item_Group"}, {"Item_Group"}),
    #"Sorted Rows" = Table.Sort(#"Expanded Items",{{"Order_Key", Order.Ascending}, {"Order_Line_Number", Order.Ascending}}),
    #"Added Custom" = Table.AddColumn(#"Sorted Rows", "Custom", each [Item_Group] & "\" & [Description])
in
    #"Added Custom"
```

Queries [3]: ACCT_Order_Charges, Items, ACCT_Order_Charges_with_Group

This allows me to view a short section of **M** language, which demonstrates some of the **M** language syntax rules:

The first line of every query must begin with '**let**':

```
let
   Source = Table.NestedJoin(ACCT_Order_Charges, {"Item_Key"}, Items, {"Item_Key"}, "Items",
JoinKind.LeftOuter),
   #"Expanded Items" = Table.ExpandTableColumn(Source, "Items", {"Item_Group"},
{"Item_Group"}),
   #"Sorted Rows" = Table.Sort(#"Expanded Items",{{"Order_Key", Order.Ascending},
{"Order_Line_Number", Order.Ascending}}),
   #"Added Custom" = Table.AddColumn(#"Sorted Rows", "Custom", each [Item_Group] & "\" &
[Description])
in
   #"Added Custom"
```

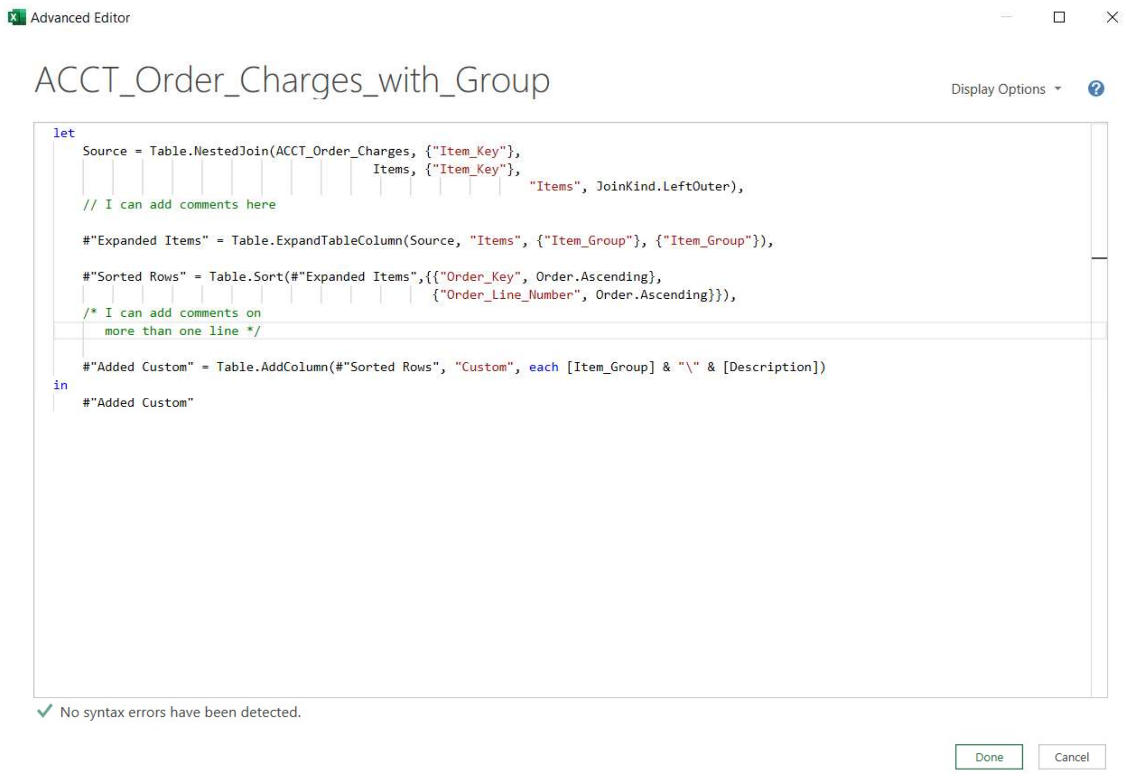
In this case, I have five lines which refer to steps defining the source of the query (which I can see in the 'APPLIED STEPS' section in the Power Query Editor) and data manipulation. Some of the steps in the Advanced Editor are preceded by a '**#**', but more on this in a moment. The lines are separated by a comma ('**,**'). The end of the query is indicated by '**in**' and a further step:

```
        in
   #"Added Custom"
```

This tells Power Query which step to return to once the query has been executed, in this case 'Added Custom'. The '**#**' at the beginning of some of the lines is a message to Power Query to ignore the spaces in the names (*e.g.* in 'Sorted Rows'). This is useful to know when creating steps manually.

The steps are sequential, and need to be linked to the previous step. Hence, the 'Sorted Rows' step refers to the 'Renamed Columns' step and the 'Renamed Columns' step refers to the 'Expanded NewColumn', *etc*. When inserting lines, I must adjust the surrounding lines to make sure the sequence is intact, otherwise I will get an error. Power Query has moved on since it was first created, and now if I am working with the Power Query editor, I can use the GUI interface to insert steps, and Power Query will automatically adjust the step names on the next step as required.

The lines are quite long and can get quite convoluted, so I can edit the format to make it easier to read:

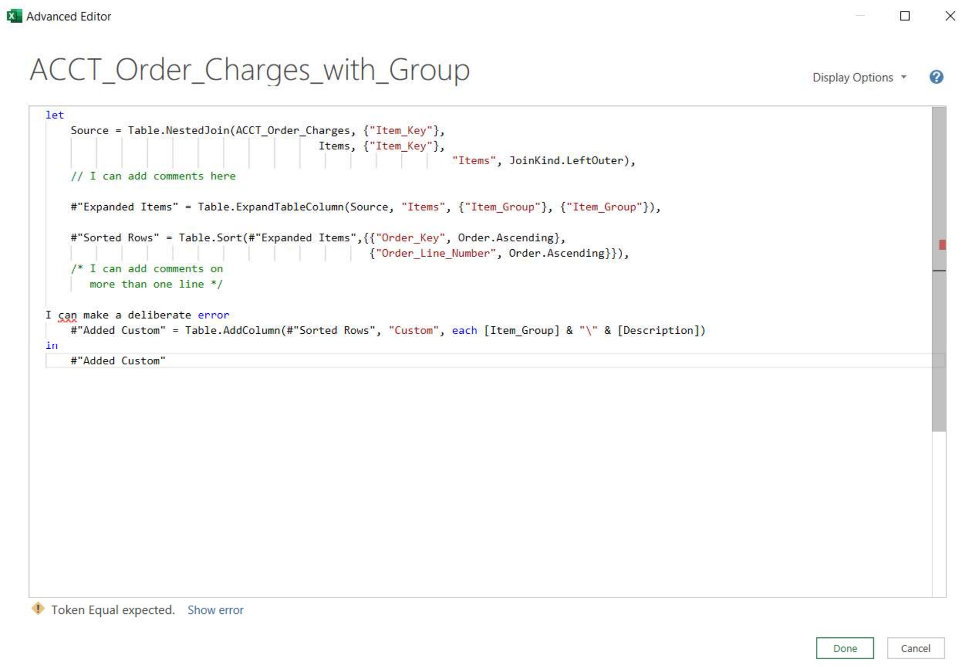


```
let
    Source = Table.NestedJoin(ACCT_Order_Charges, {"Item_Key"},
                                            Items, {"Item_Key"},
                                                         "Items", JoinKind.LeftOuter),
    // I can add comments here

    #"Expanded Items" = Table.ExpandTableColumn(Source, "Items", {"Item_Group"}, {"Item_Group"}),

    #"Sorted Rows" = Table.Sort(#"Expanded Items",{{"Order_Key", Order.Ascending},
                                                    {"Order_Line_Number", Order.Ascending}}),
    /* I can add comments on
       more than one line */

    #"Added Custom" = Table.AddColumn(#"Sorted Rows", "Custom", each [Item_Group] & "\" & [Description])
in
    #"Added Custom"
```

✓ No syntax errors have been detected.

I have split the lines up to make them less of a long list of code. This may be useful when trying to keep track of whether all brackets have been closed (and closed in the right place). I have also inserted a single comment by preceding the line with '**//**', and a section of comments by preceding with '**/***' and ending with '***/**'. Users of earlier versions of Power Query may notice that the 'token comma' is now not needed after a comment section: do be guided by the syntax checker.

The checker is at the bottom of the screen which will warn me if the syntax is wrong and allow me to jump to the error.

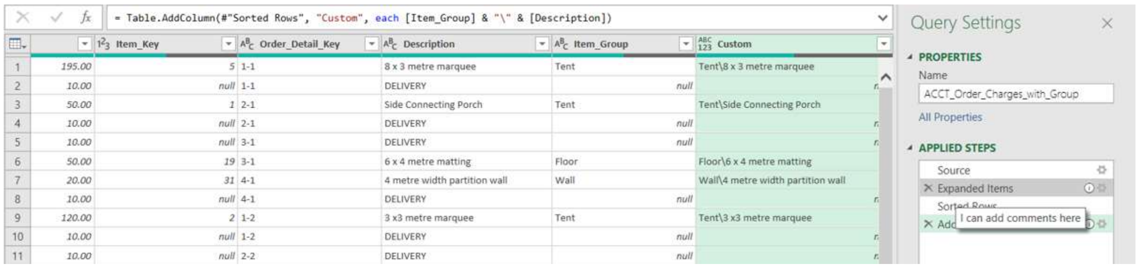

Advanced Editor

ACCT_Order_Charges_with_Group

```
let
    Source = Table.NestedJoin(ACCT_Order_Charges, {"Item_Key"},
                              Items, {"Item_Key"},
                                        "Items", JoinKind.LeftOuter),
    // I can add comments here

    #"Expanded Items" = Table.ExpandTableColumn(Source, "Items", {"Item_Group"}, {"Item_Group"}),

    #"Sorted Rows" = Table.Sort(#"Expanded Items",{{"Order_Key", Order.Ascending},
                                {"Order_Line_Number", Order.Ascending}}),
    /* I can add comments on
       more than one line */

I can make a deliberate error
    #"Added Custom" = Table.AddColumn(#"Sorted Rows", "Custom", each [Item_Group] & "\" & [Description])
in
    #"Added Custom"
```

Token Equal expected. Show error

A more recent development in Power Query is to make the comments visible from 'APPLIED STEPS':


```
= Table.AddColumn(#"Sorted Rows", "Custom", each [Item_Group] & "\" & [Description])
```

		Item_Key	Order_Detail_Key	Description	Item_Group	Custom
1	195.00	5	1-1	8 x 3 metre marquee	Tent	Tent\8 x 3 metre marquee
2	10.00	null	1-1	DELIVERY	null	
3	50.00	1	2-1	Side Connecting Porch	Tent	Tent\Side Connecting Porch
4	10.00	null	2-1	DELIVERY	null	
5	10.00	null	3-1	DELIVERY	null	
6	50.00	19	3-1	6 x 4 metre matting	Floor	Floor\6 x 4 metre matting
7	20.00	31	4-1	4 metre width partition wall	Wall	Wall\4 metre width partition wall
8	10.00	null	4-1	DELIVERY	null	
9	120.00	2	1-2	3 x3 metre marquee	Tent	Tent\3 x3 metre marquee
10	10.00	null	1-2	DELIVERY	null	
11	10.00	null	2-2	DELIVERY	null	

Query Settings

PROPERTIES

Name: ACCT_Order_Charges_with_Group

All Properties

APPLIED STEPS

Source
Expanded Items
Sorted Rows
Add

I can add comments here

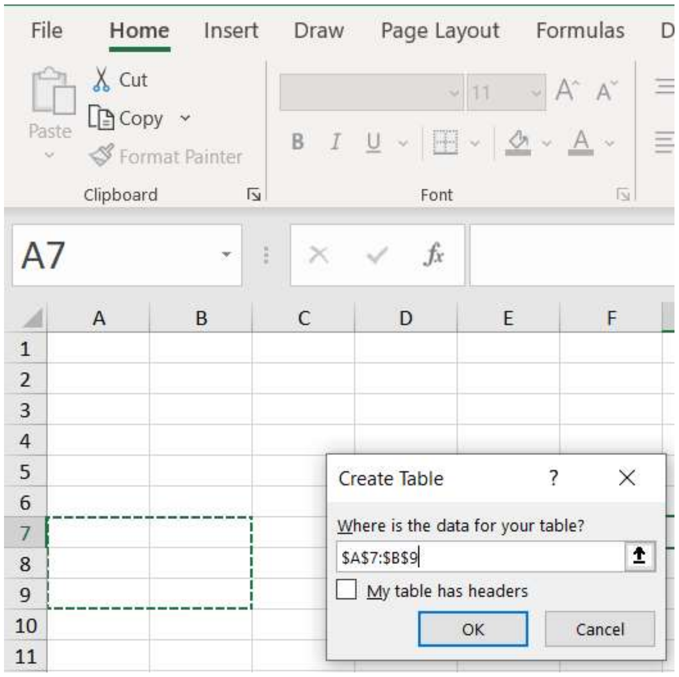
The next sections are all about building a Calendar query. The four steps are:

- **Step 1:** Create a table called **Parameters** in an Excel Worksheet to hold the calendar boundaries
- **Step 2:** Create a function **fnGetParameter** which uses the calendar boundaries as its parameters
- **Step 3:** Build the basic dynamic calendar framework
- **Step 4:** Add any required calendar columns.

Creating a Calendar – Step 1: Creating Parameters

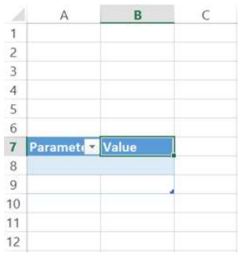
In a new blank Excel Worksheet, I create a new table with some specific properties. My table is called **Parameters** and has two columns, **Parameter** and **Value**.

To begin, I will open a new blank Excel worksheet – in the 'Insert' Section, there is an option to 'Insert Table' which brings up a 'Create Table' dialogue box (**CTRL + T** is the shortcut). I choose an area that covers two columns and three rows (I chose **A7:B9** in the example shown below). I check the box 'My Table has Headers' and click OK.



Just as an aside, normally here at SumProduct, we will advocate always starting Tables in cell **A1** and calling the worksheet the same name as the Table. I am not going to do that here to show it is not necessary, but there is merit in doing this if data is exported to other programs too. It's best to give data exporting a "helping hand" on occasion.

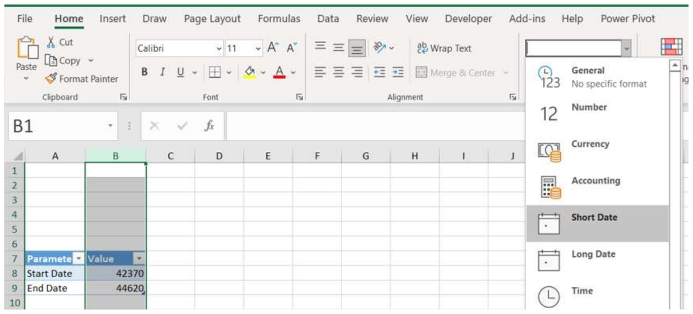
Returning to my example, the 'Table Name' can be changed in the top left of the screen and must be set to **Parameters**. Clicking on the column names allows me to set them to **Parameter** and **Value**:



The parameters the function will be using are ***Start Date*** and ***End Date***, and these are the entries in my first column. In the second column I enter the date I wish my calendar to begin at, and a formula that will show the last day of the current month.

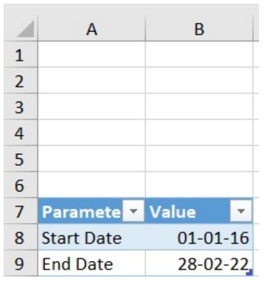
Parameter	**Value**
Start Date	**1/1/2016**
End Date	**=EOMONTH(TODAY(),0)**

This may look like a long number to begin with: the data format on the column needs to be set to **Short Date** as shown below:



	A	B
7	Paramete	Value
8	Start Date	42370
9	End Date	44620

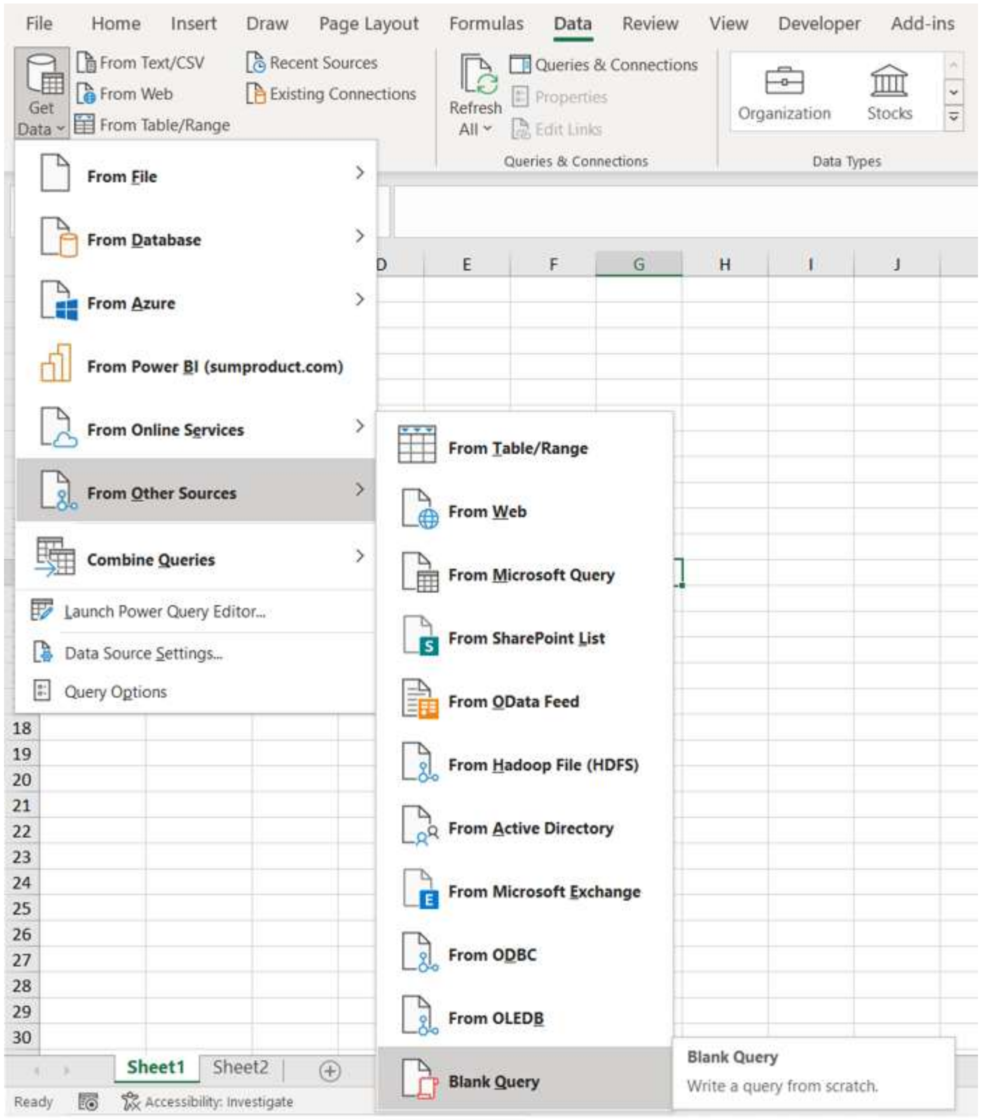
My **Parameter** table is now ready for the **fnGetParameter** function:



	A	B
7	Paramete	Value
8	Start Date	01-01-16
9	End Date	28-02-22

Creating a Calendar – Step 2: Create the fnGetParameter Function

I choose to create a new blank query by going to the ‘From Other Sources’ section of the ‘Get Data’ dropdown and choosing ‘Blank Query’ from the dropdown:



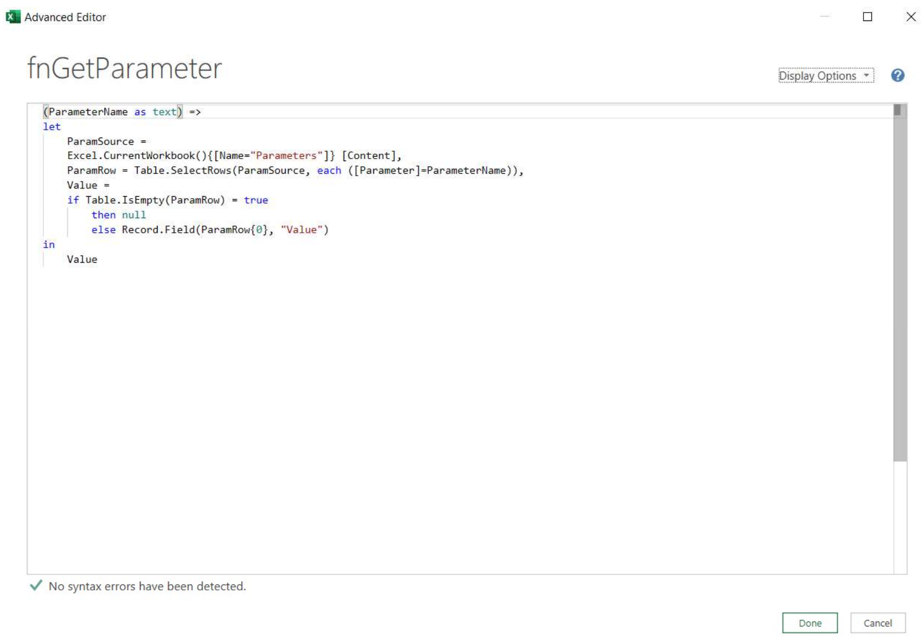
In the ‘Query Editor’ screen, I choose the ‘Advanced Editor’ from the ‘Home’ tab.

I enter the following **M** language:

```
(ParameterName as text) =>
 let
ParamSource = Excel.CurrentWorkbook(){[Name="Parameters"]}[Content],
ParamRow = Table.SelectRows(ParamSource, each ([Parameter]=ParameterName)),
        Value=
        if Table.IsEmpty(ParamRow)=true
                then null
                else Record.Field(ParamRow{0},"Value")
in
```

```
    Value
```

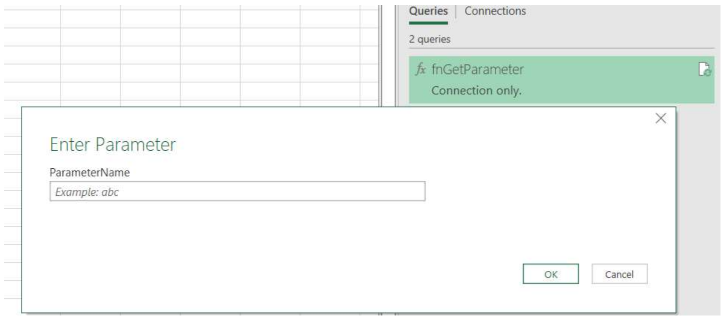
I then rename the query **fnGetParameter.**



```
(ParameterName as text) =>
let
    ParamSource =
    Excel.CurrentWorkbook(){[Name="Parameters"]} [Content],
    ParamRow = Table.SelectRows(ParamSource, each ([Parameter]=ParameterName)),
    Value =
    if Table.IsEmpty(ParamRow) = true
        then null
        else Record.Field(ParamRow{0}, "Value")
in
    Value
```

No syntax errors have been detected.

When I choose to 'Close and Load' from the 'Home' tab the new query automatically saves as connection only.   Double-clicking on it invokes the function:



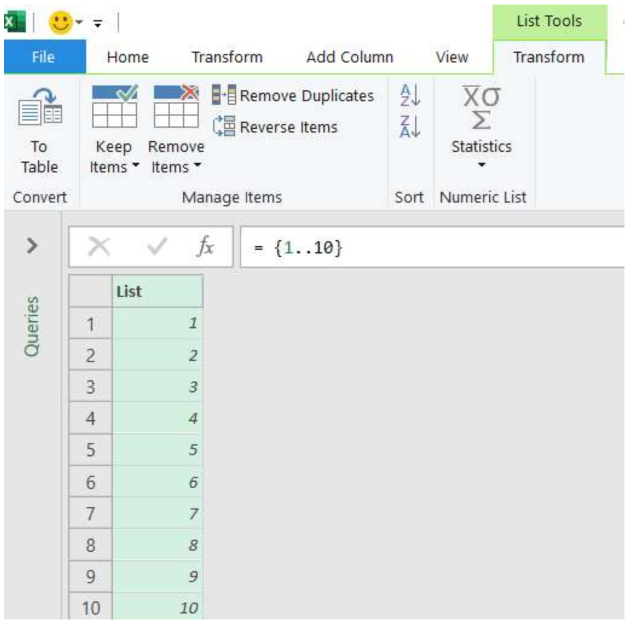
Now I have my parameters and a function which will read them, so that I may create my calendar framework and add columns to create a useful calendar.

Creating a Calendar – Step 3: Build Framework

Now I have set up my parameters and created a function to extract the values, I can begin to build my calendar. In the worksheet I created last time, I will start by creating another blank query, following the same procedure as before, and this time I will call it **Calendar**. The process I follow to create my calendar framework is to build a list of dates between the two parameters that I have specified.

I begin by entering a simple list in the formula bar:

```
= {1..10}
```

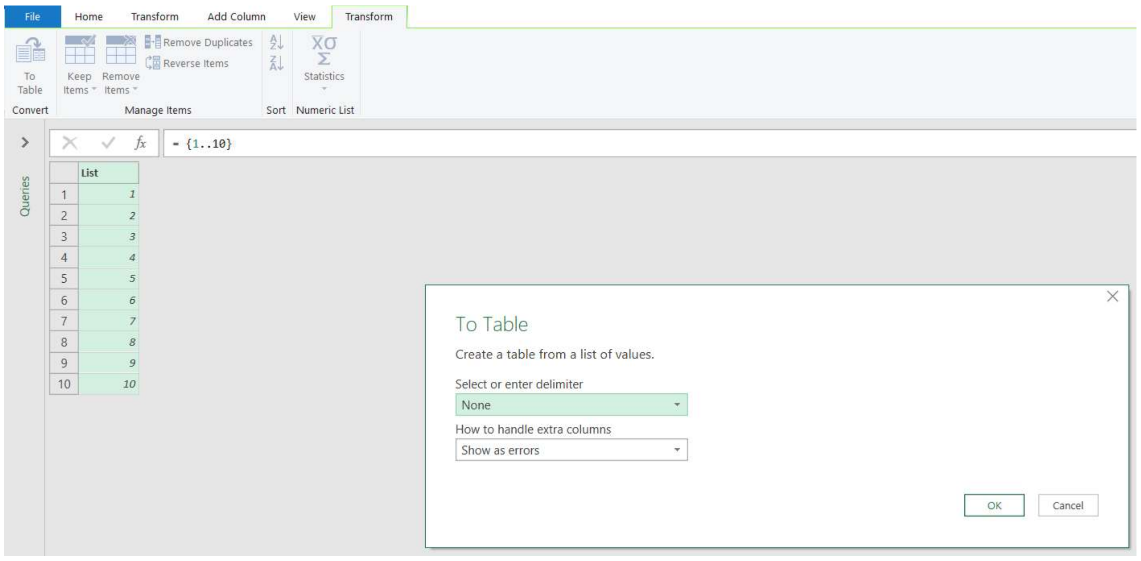


```
= {1..10}
```

	List
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

This gives a column of 10 numbers, which I can prepare for calendar format by transforming. Notice that Power Query automatically presents me with options for transforming a list.

In the 'Convert' section I choose to convert my list 'to Table', and take the default options:



```
= {1..10}
```

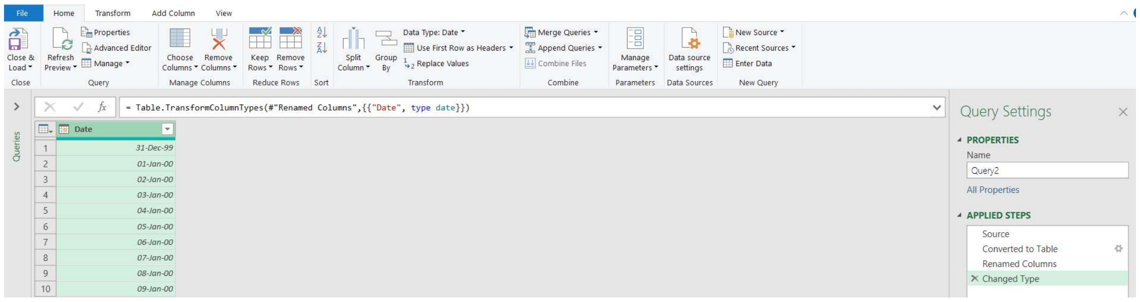
To Table

Create a table from a list of values.

Select or enter delimiter: None

How to handle extra columns: Show as errors

In the resulting table I right click my column and ‘Change Type’ to ‘Date’, and then rename my column **Date.**



```
= Table.TransformColumnTypes(#"Renamed Columns",{{"Date", type date}})
```

	Date
1	31-Dec-99
2	01-Jan-00
3	02-Jan-00
4	03-Jan-00
5	04-Jan-00
6	05-Jan-00
7	06-Jan-00
8	07-Jan-00
9	08-Jan-00
10	09-Jan-00

Query Settings

PROPERTIES

Name: Query2

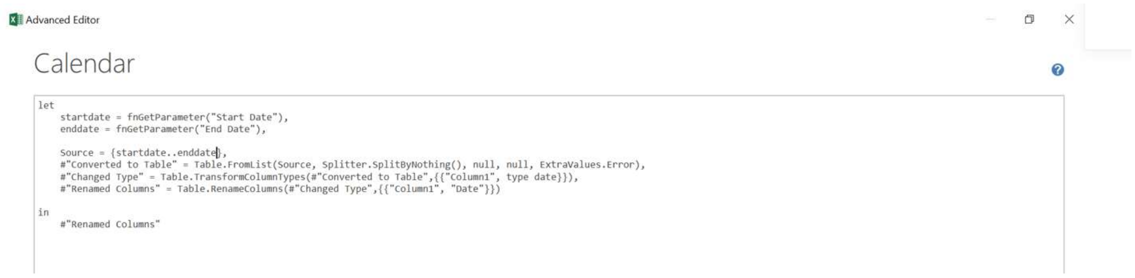
APPLIED STEPS

- Source
- Converted to Table
- Renamed Columns
- Changed Type

They may not be current, but they are dates! Now I need my parameters.

I choose the ‘Advanced Editor’ from the ‘Home’ section. The editor shows the lines already created as a result of the transforming I have done. The ‘Source’ step currently shows my original list of 10 numbers: I need to change the source to look at my parameters instead of 1 and 10. Therefore, after ‘let’ and before the ‘Source’ line, I need to add my parameters, which I will call **startdate** and **enddate** thus:

```
startdate = fnGetParameter(“Start Date”),
enddate = fnGetParameter(“End Date”),
```

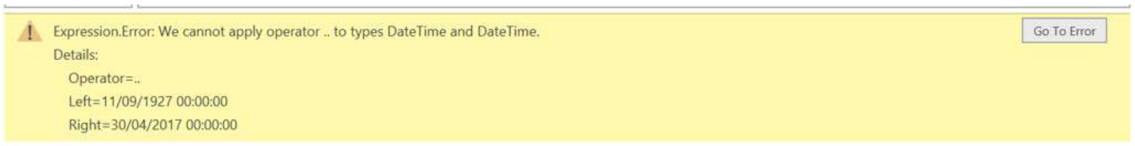


Advanced Editor

Calendar

```
let
    startdate = fnGetParameter("Start Date"),
    enddate = fnGetParameter("End Date"),

    Source = {startdate..enddate},
    #"Converted to Table" = Table.FromList(Source, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    #"Changed Type" = Table.TransformColumnTypes(#"Converted to Table",{{"Column1", type date}}),
    #"Renamed Columns" = Table.RenameColumns(#"Changed Type",{{"Column1", "Date"}})

in
    #"Renamed Columns"
```

However, if I try this, I get the following error:



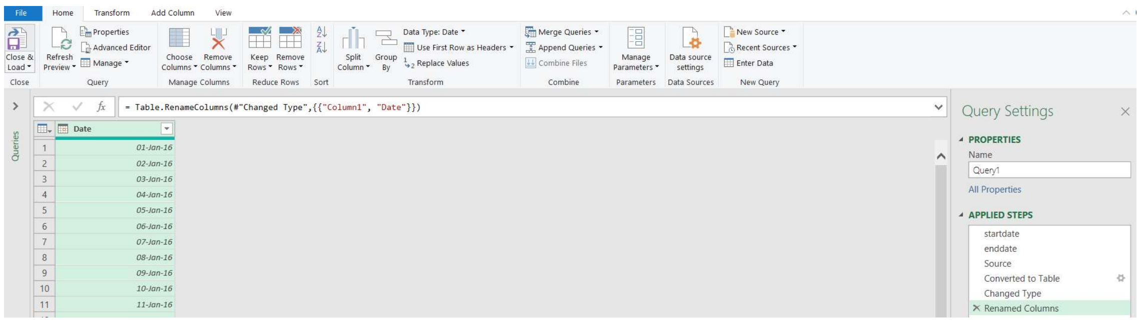
```
Expression.Error: We cannot apply operator .. to types DateTime and DateTime.
Details:
    Operator=..
    Left=11/09/1927 00:00:00
    Right=30/04/2017 00:00:00
```

Go To Error

This is because I started with a list of numbers and *then* I tried to use dates instead. I need to express the dates as numbers, so I need to use the **Number.From** function:

startdate = Number.From(fnGetParameter(“Start Date”)),
enddate = Number.From(fnGetParameter(“End Date”)),

When I use these definitions instead, my query returns a column of dates:



```
= Table.RenameColumns(#"Changed Type",{{"Column1", "Date"}})
```

	Date
1	01-Jan-16
2	02-Jan-16
3	03-Jan-16
4	04-Jan-16
5	05-Jan-16
6	06-Jan-16
7	07-Jan-16
8	08-Jan-16
9	09-Jan-16
10	10-Jan-16
11	11-Jan-16

Query Settings

PROPERTIES

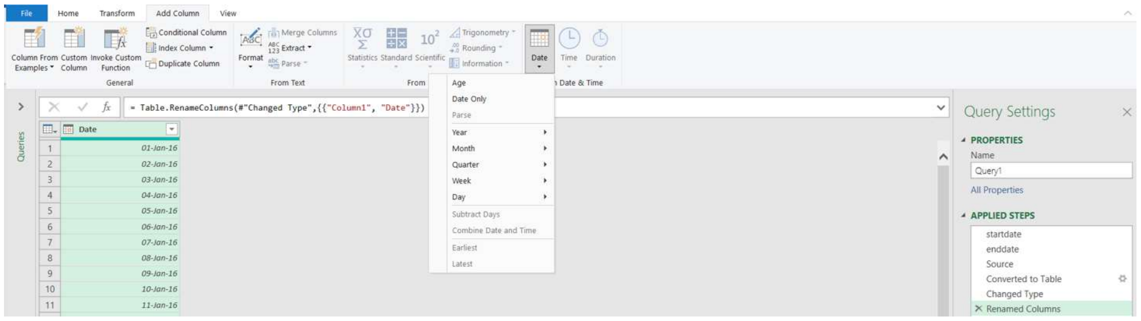
Name: Query1

APPLIED STEPS

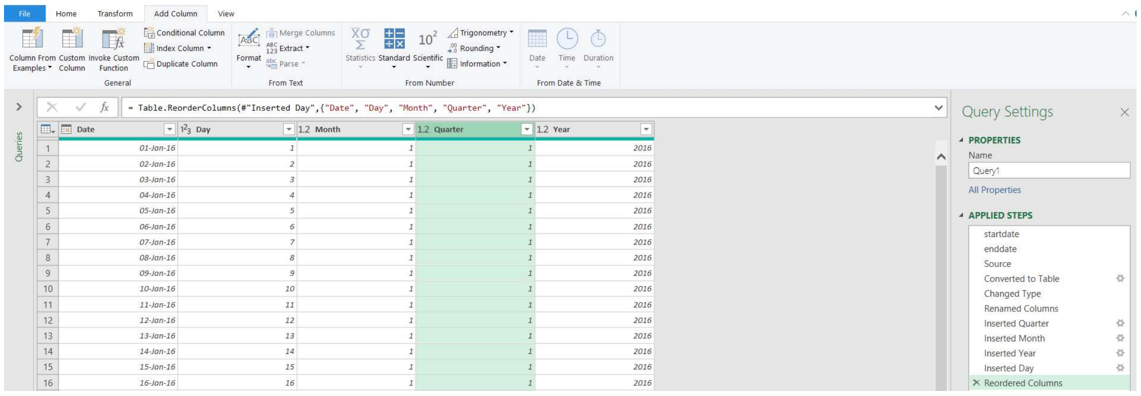
- startdate
- enddate
- Source
- Converted to Table
- Changed Type
- Renamed Columns

Creating a Calendar – Step 4: Adding More Date Columns

Although I have a calendar, it is a very basic calendar. A few more columns would be useful. For this, I am going to repeatedly use the 'Date' section of the 'Add Column' tab and use the options that are shown on the dropdown below:



I choose to add the 'Year', 'Month', 'Day' and 'Quarter (of Year)' options. It doesn't matter what order I create them in as I can drag the columns to change the order:



```
= Table.ReorderColumns(#"Inserted Day",{"Date", "Day", "Month", "Quarter", "Year"})
```

	Date	Day	Month	Quarter	Year
1	01-Jan-16	1	1	1	2016
2	02-Jan-16	2	1	1	2016
3	03-Jan-16	3	1	1	2016
4	04-Jan-16	4	1	1	2016
5	05-Jan-16	5	1	1	2016
6	06-Jan-16	6	1	1	2016
7	07-Jan-16	7	1	1	2016
8	08-Jan-16	8	1	1	2016
9	09-Jan-16	9	1	1	2016
10	10-Jan-16	10	1	1	2016
11	11-Jan-16	11	1	1	2016
12	12-Jan-16	12	1	1	2016
13	13-Jan-16	13	1	1	2016
14	14-Jan-16	14	1	1	2016
15	15-Jan-16	15	1	1	2016
16	16-Jan-16	16	1	1	2016

Query Settings

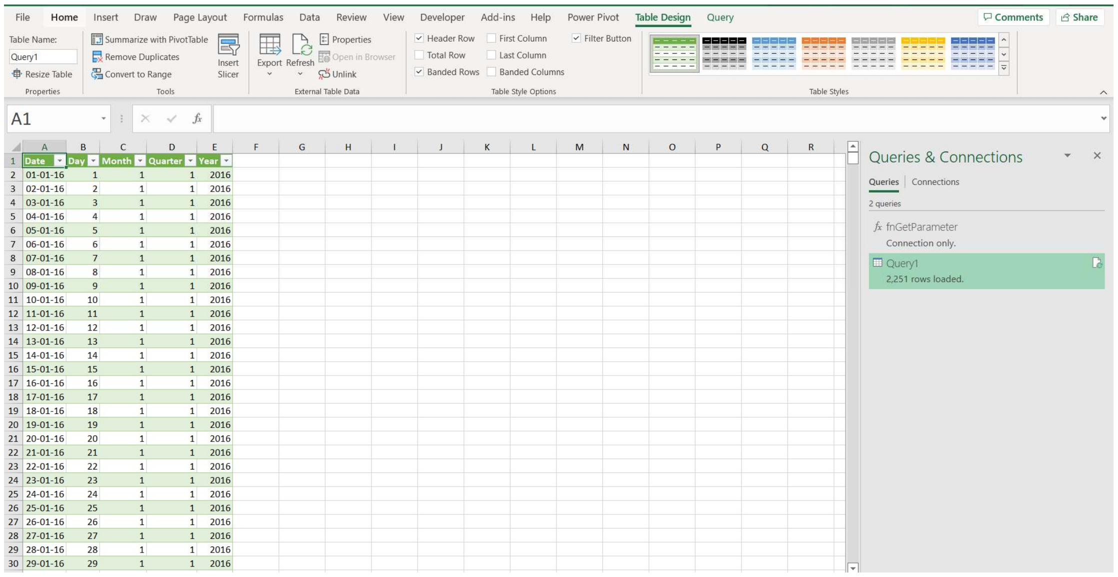
PROPERTIES

Name: Query1

APPLIED STEPS

- startdate
- enddate
- Source
- Converted to Table
- Changed Type
- Renamed Columns
- Inserted Quarter
- Inserted Month
- Inserted Year
- Inserted Day
- Reordered Columns

On the ‘Home’ tab, I then choose to ‘Close and Load’ to see that all the Calendar entries have been created as I expected. The entries automatically appear in a separate sheet to my **Parameters** table:



Date	Day	Month	Quarter	Year
01-01-16	1	1	1	2016
02-01-16	2	1	1	2016
03-01-16	3	1	1	2016
04-01-16	4	1	1	2016
05-01-16	5	1	1	2016
06-01-16	6	1	1	2016
07-01-16	7	1	1	2016
08-01-16	8	1	1	2016
09-01-16	9	1	1	2016
10-01-16	10	1	1	2016
11-01-16	11	1	1	2016
12-01-16	12	1	1	2016
13-01-16	13	1	1	2016
14-01-16	14	1	1	2016
15-01-16	15	1	1	2016
16-01-16	16	1	1	2016
17-01-16	17	1	1	2016
18-01-16	18	1	1	2016
19-01-16	19	1	1	2016
20-01-16	20	1	1	2016
21-01-16	21	1	1	2016
22-01-16	22	1	1	2016
23-01-16	23	1	1	2016
24-01-16	24	1	1	2016
25-01-16	25	1	1	2016
26-01-16	26	1	1	2016
27-01-16	27	1	1	2016
28-01-16	28	1	1	2016
29-01-16	29	1	1	2016

Queries & Connections

Queries | Connections

2 queries

fnGetParameter
Connection only.

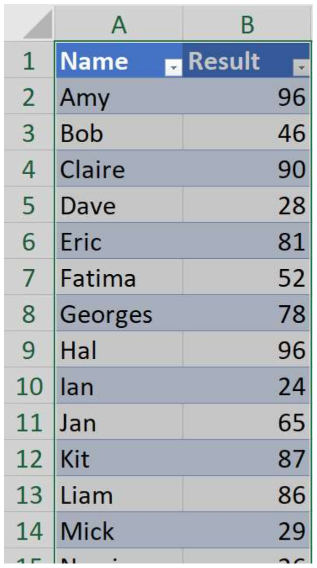
Query1
2,251 rows loaded.

My calendar is ready for use.

More on Parameters

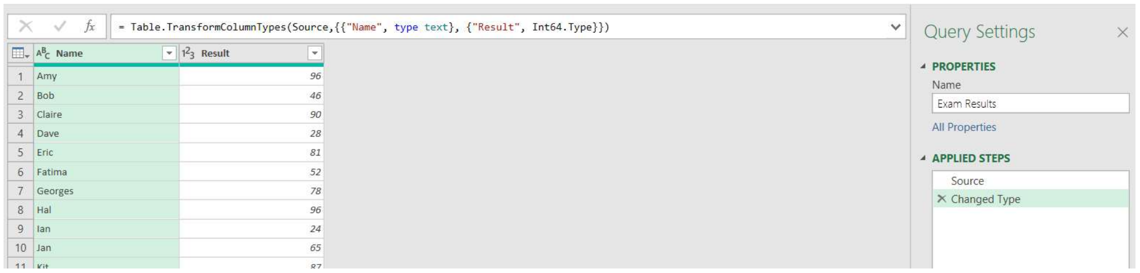
Parameters are not just useful for creating Calendars. In this extended section I look at some other uses:

I am looking at some exam results:



Name	Result
Amy	96
Bob	46
Claire	90
Dave	28
Eric	81
Fatima	52
Georges	78
Hal	96
Ian	24
Jan	65
Kit	87
Liam	86
Mick	29

I will be grading the results and I will be using this example to explore parameters. I'll start by extracting my data into Power Query, where I will create the grade column. To extract my data, I choose 'From Table/Range' from the 'Get & Transform' section of the Data tab.



```
= Table.TransformColumnTypes(Source,{{"Name", type text}, {"Result", Int64.Type}})
```

	Name	Result
1	Amy	96
2	Bob	46
3	Claire	90
4	Dave	28
5	Eric	81
6	Fatima	52
7	Georges	78
8	Hal	96
9	Ian	24
10	Jan	65
11	Kit	87

Query Settings

PROPERTIES

Name: Exam Results

All Properties

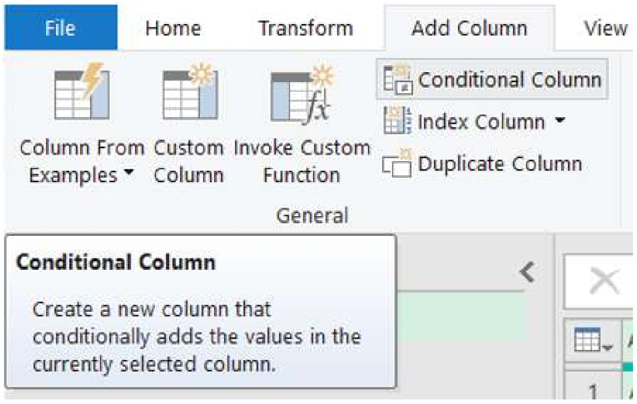
APPLIED STEPS

Source

Changed Type

Conditional Columns

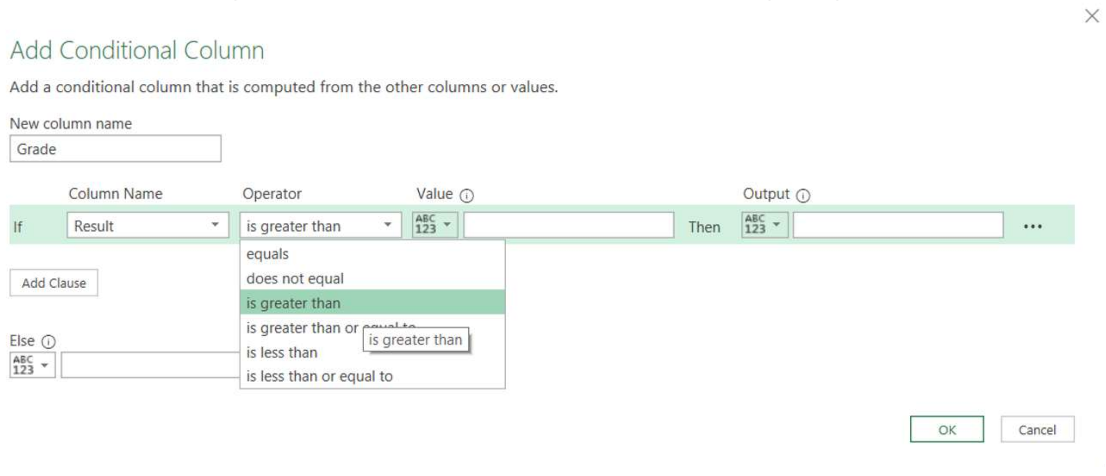
I have called my query **Exam Results**. I will begin by creating a Conditional Column from the 'Add Column' tab:



Conditional Column

Create a new column that conditionally adds the values in the currently selected column.

I call the new column **Grade**, and create the grade bands for the results. Each Clause will look at whether the Result 'is greater than' a Value, and I will start with the highest grade.



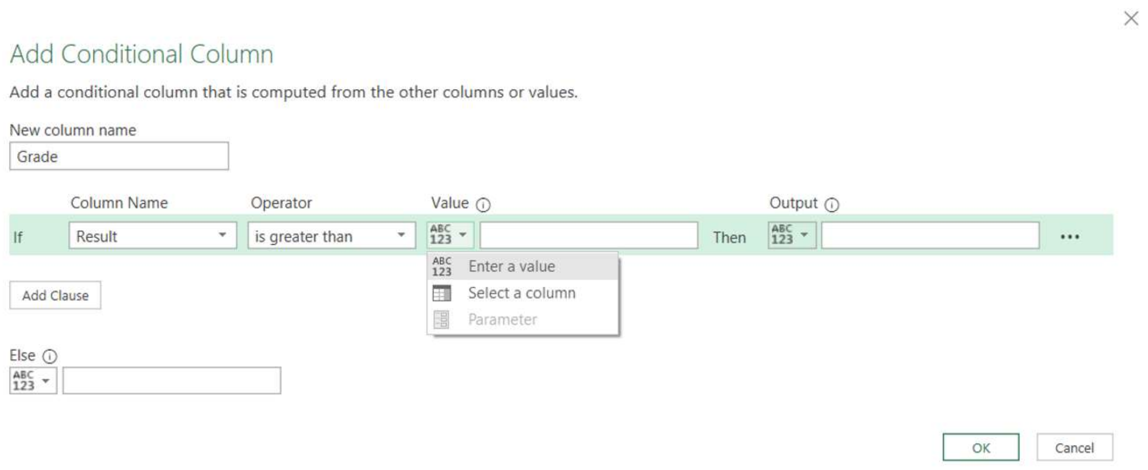
Add Conditional Column

Add a conditional column that is computed from the other columns or values.

New column name: Grade

	Column Name	Operator	Value		Output
If	Result	is greater than		Then	

For now, I will be entering values, but I plan to replace this with parameters later.



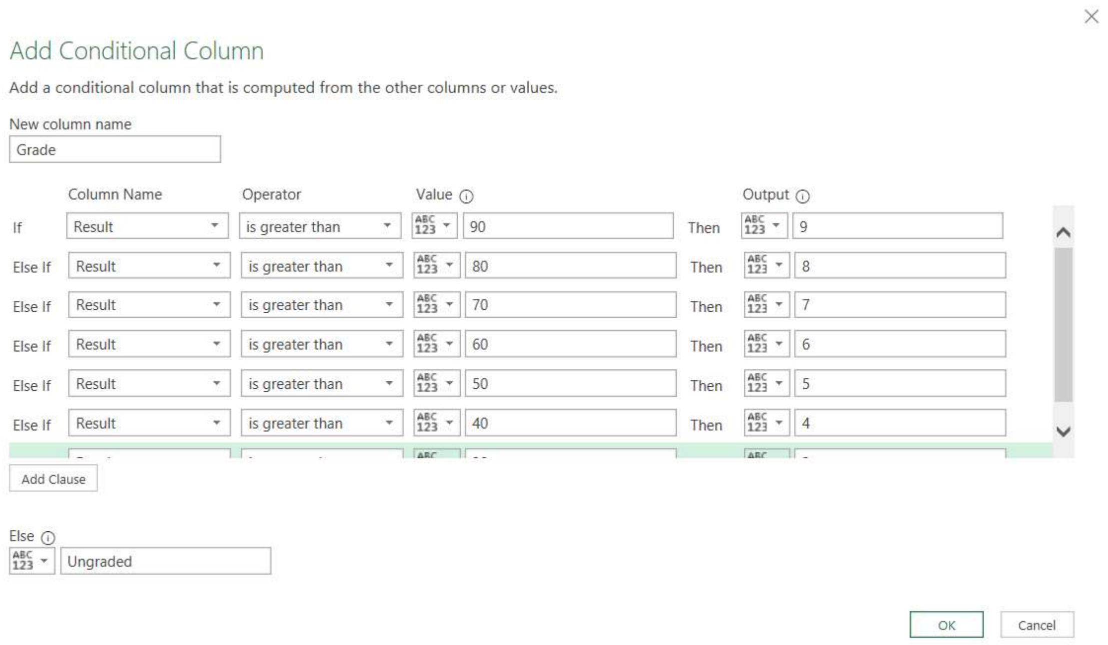
Add Conditional Column

Add a conditional column that is computed from the other columns or values.

New column name: Grade

	Column Name	Operator	Value		Output
If	Result	is greater than		Then	

I create a Clause for each band.



Add Conditional Column

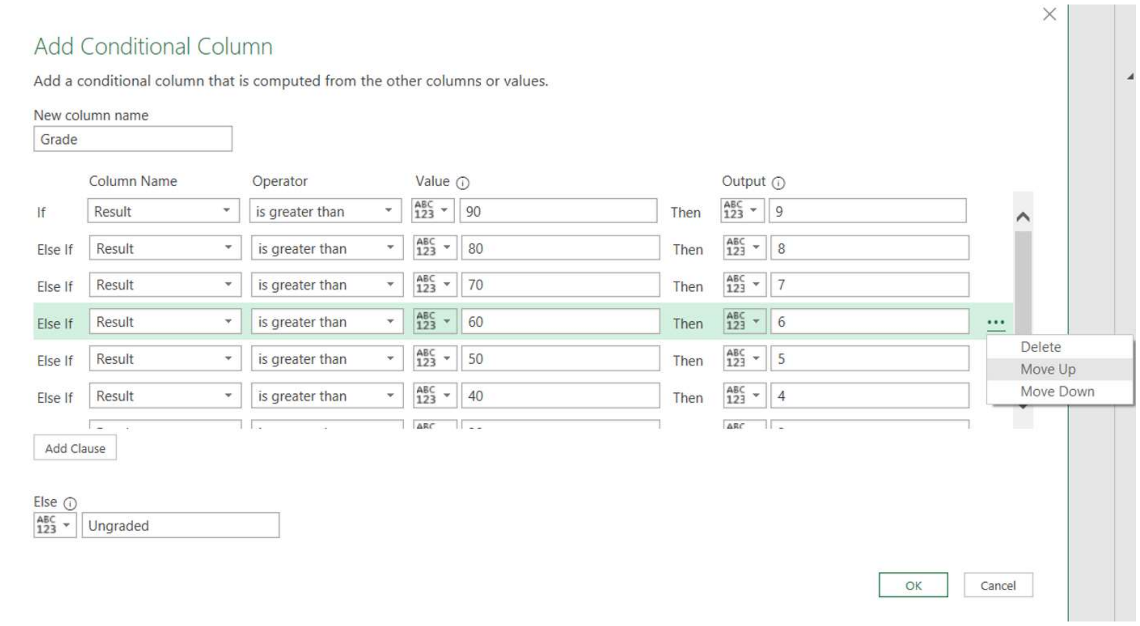
Add a conditional column that is computed from the other columns or values.

New column name: Grade

	Column Name	Operator	Value		Output
If	Result	is greater than	90	Then	9
Else If	Result	is greater than	80	Then	8
Else If	Result	is greater than	70	Then	7
Else If	Result	is greater than	60	Then	6
Else If	Result	is greater than	50	Then	5
Else If	Result	is greater than	40	Then	4

Else: Ungraded

Note that, if I miss one, I can add it later and then change the order using the menu next to each Clause.



Add Conditional Column

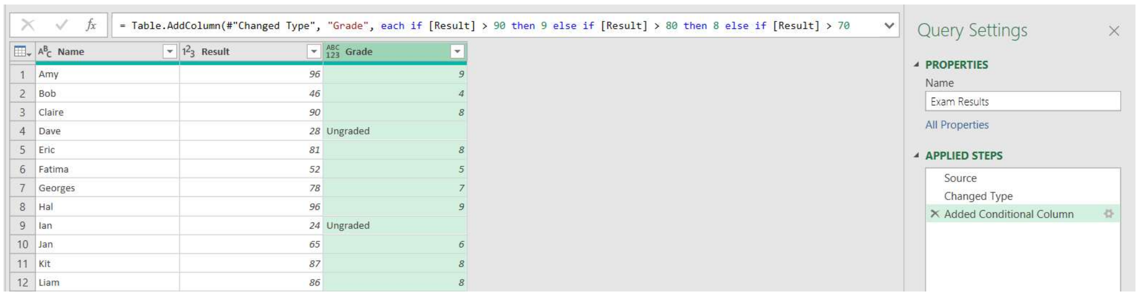
Add a conditional column that is computed from the other columns or values.

New column name: Grade

	Column Name	Operator	Value		Output
If	Result	is greater than	90	Then	9
Else If	Result	is greater than	80	Then	8
Else If	Result	is greater than	70	Then	7
Else If	Result	is greater than	60	Then	6
Else If	Result	is greater than	50	Then	5
Else If	Result	is greater than	40	Then	4

Else: Ungraded

I click OK to see the new column:



```
= Table.AddColumn(#"Changed Type", "Grade", each if [Result] > 90 then 9 else if [Result] > 80 then 8 else if [Result] > 70
```

	Name	Result	Grade
1	Amy	96	9
2	Bob	46	4
3	Claire	90	8
4	Dave	28	Ungraded
5	Eric	81	8
6	Fatima	52	5
7	Georges	78	7
8	Hal	96	9
9	Ian	24	Ungraded
10	Jan	65	6
11	Kit	87	8
12	Liam	86	8

Query Settings

PROPERTIES

Name: Exam Results

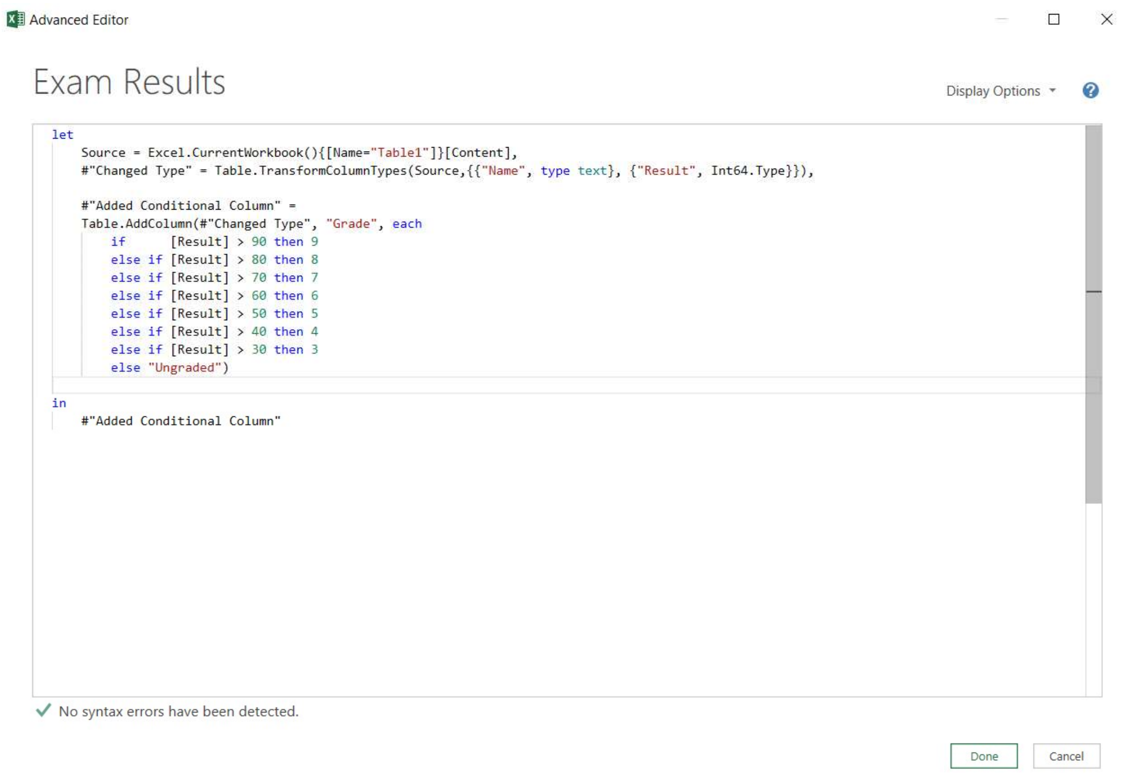
APPLIED STEPS

- Source
- Changed Type
- Added Conditional Column

The **M** code generated for this step is as follows:

```
Table.AddColumn(#"Changed Type", "Grade", each if [Result] > 90 then 9 else if [Result] >
80 then 8 else if [Result] > 70 then 7 else if [Result] > 60 then 6 else if [Result] > 50
then 5 else if [Result] > 40 then 4 else if [Result] > 30 then 3 else "Ungraded")
```

I can change this in the Advanced Editor, which I access from the Home tab. I want to format it so that the boundaries are easier to see. I can split each step over any number of lines, a comma (,) or (as in this case) the '**in**' statement indicates when the step is complete.



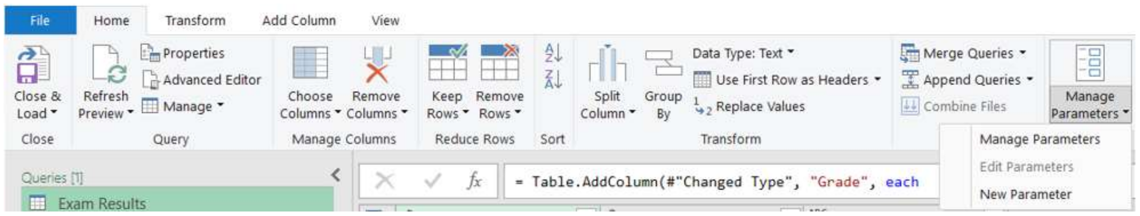
```
let
    Source = Excel.CurrentWorkbook(){[Name="Table1"]}[Content],
    #"Changed Type" = Table.TransformColumnTypes(Source,{{"Name", type text}, {"Result", Int64.Type}}),

    #"Added Conditional Column" =
    Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > 90 then 9
        else if [Result] > 80 then 8
        else if [Result] > 70 then 7
        else if [Result] > 60 then 6
        else if [Result] > 50 then 5
        else if [Result] > 40 then 4
        else if [Result] > 30 then 3
        else "Ungraded")

in
    #"Added Conditional Column"
```

No syntax errors have been detected.

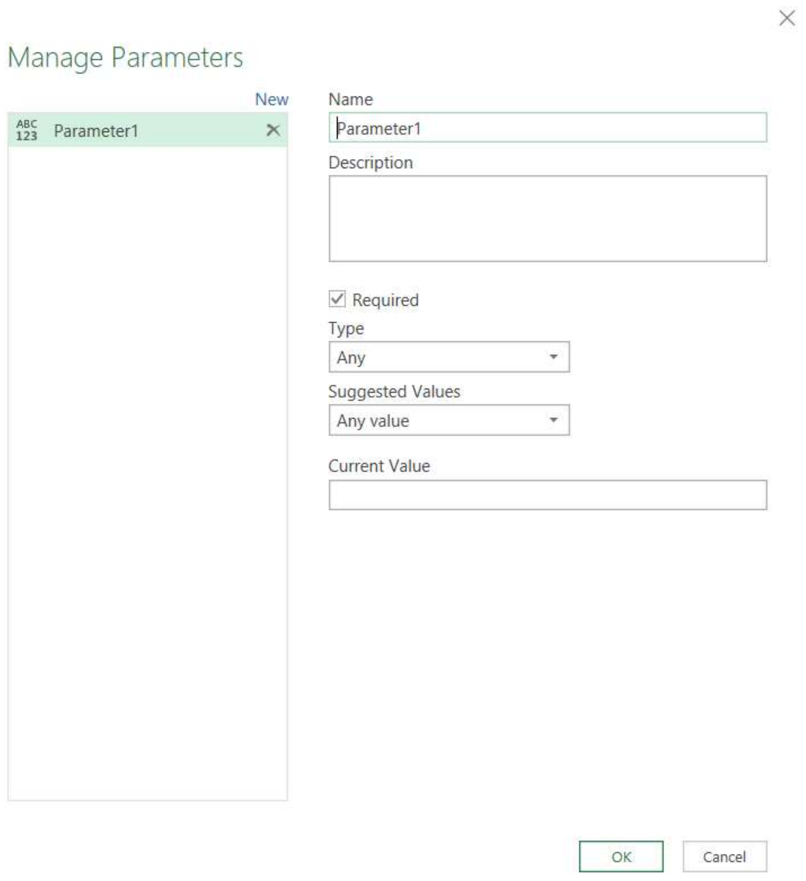
Power Query Parameters

I am now ready to create some parameters to decide the grades.

To begin with, I will enter these parameters from the ‘Manage Parameter’ option on the Home tab:



I opt to create a ‘New Parameter’. This brings up a dialog:



Name: Parameter1

Description:

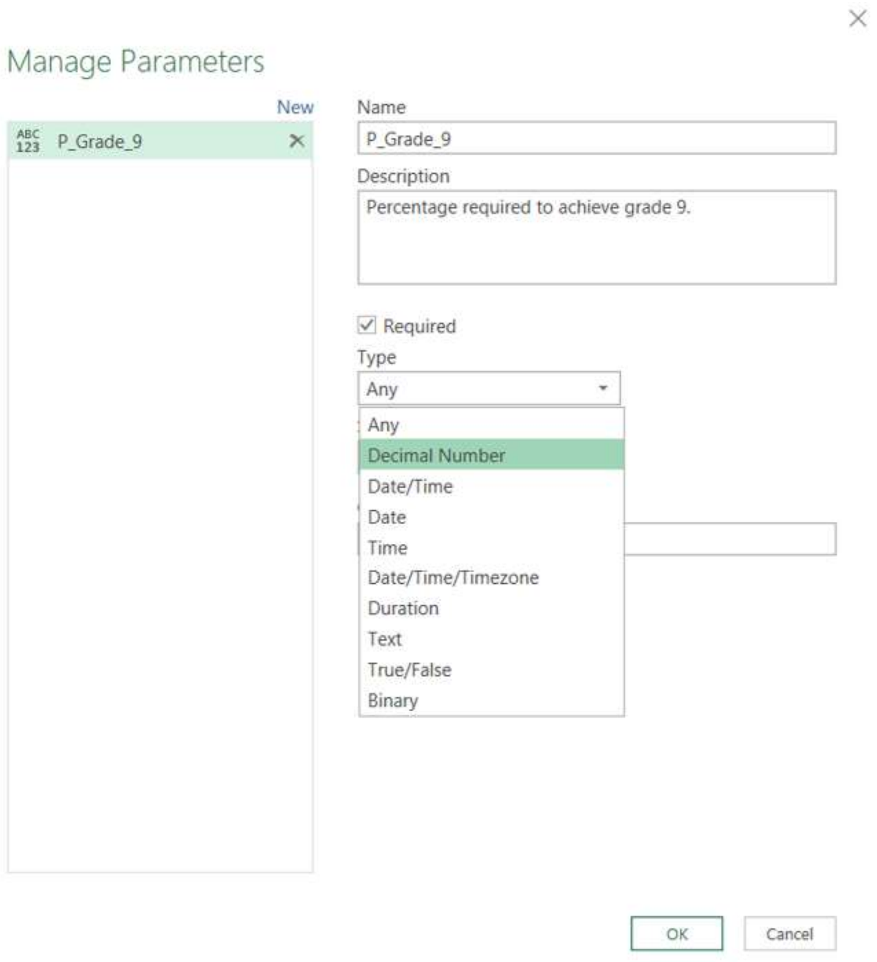
☑ Required

Type: Any

Suggested Values: Any value

Current Value:

I am going to enter a parameter for each grade. I call my first parameter **P_Grade_9**:



Manage Parameters

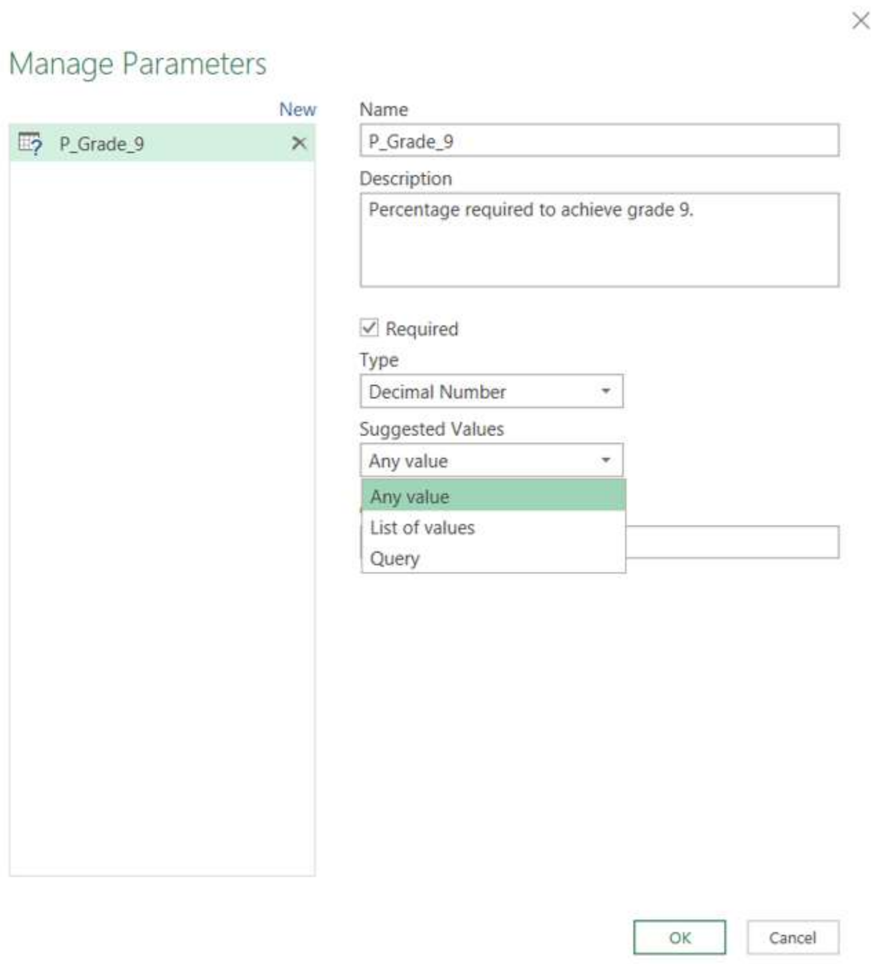
Name: P_Grade_9

Description: Percentage required to achieve grade 9.

☑ Required

Type: Any

The '**P_**' is to indicate it is a **p**arameter. This will make it easy to spot in the list of queries. I also enter a description. When I choose the Type, there is no option for percentage, so I make it a 'Decimal Number' instead.



Name: P_Grade_9

Description: Percentage required to achieve grade 9.

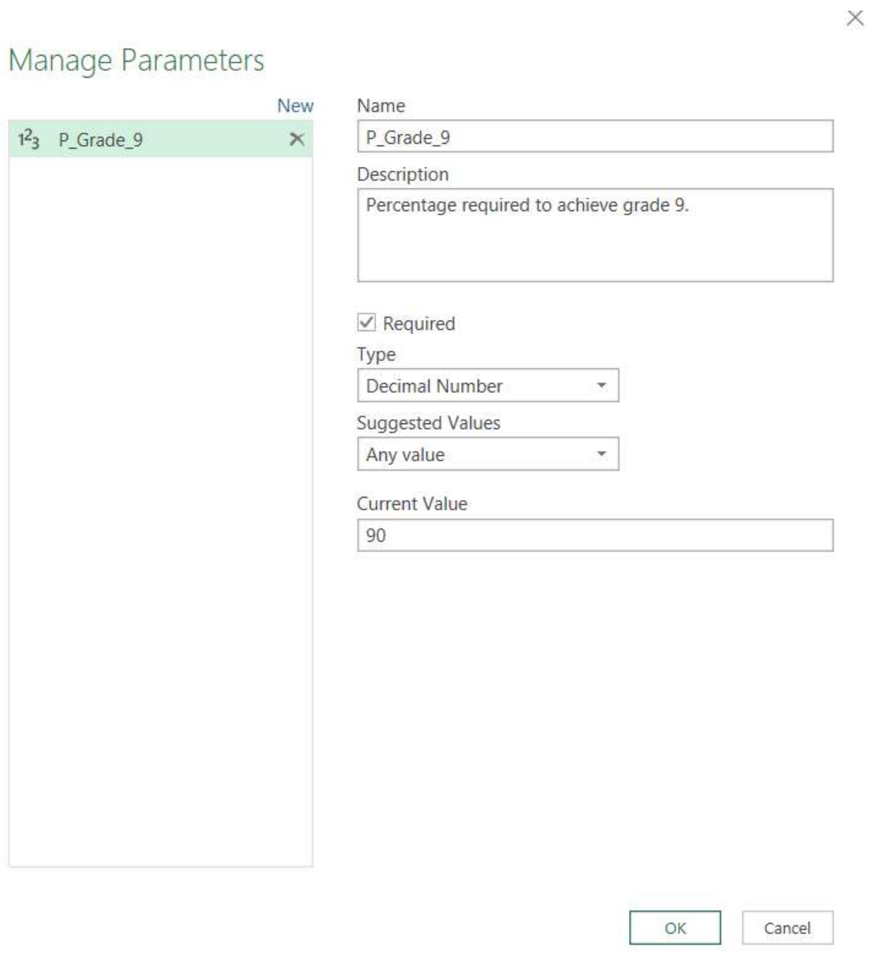
☑ Required

Type: Decimal Number

Suggested Values: Any value

Any value
List of values
Query

For now, I will allow 'Any value'. I will be revisiting the 'Suggested Values' dropdown for this example later.


Manage Parameters

P_Grade_9

Name: P_Grade_9

Description: Percentage required to achieve grade 9.

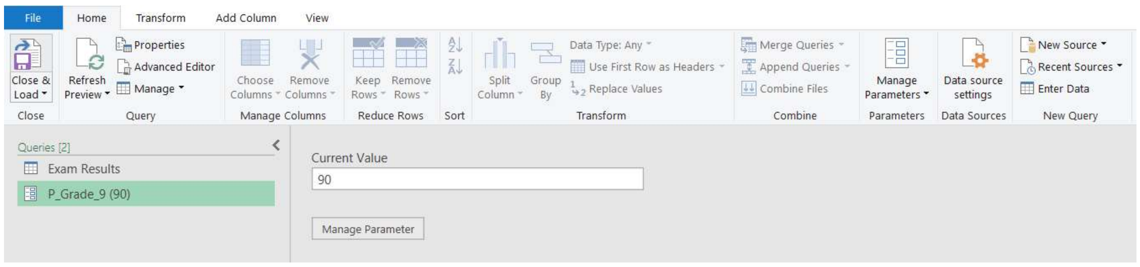
Required: ✓

Type: Decimal Number

Suggested Values: Any value

Current Value: 90

To complete this parameter, I enter a 'Current Value' of 90. If I were to click 'OK' at this point, the parameter would be created, and I would automatically exit the dialog:


Queries [2]

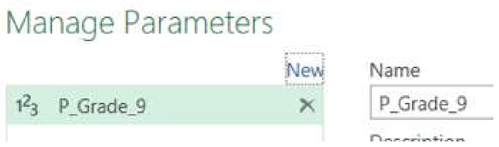
Exam Results

P_Grade_9 (90)

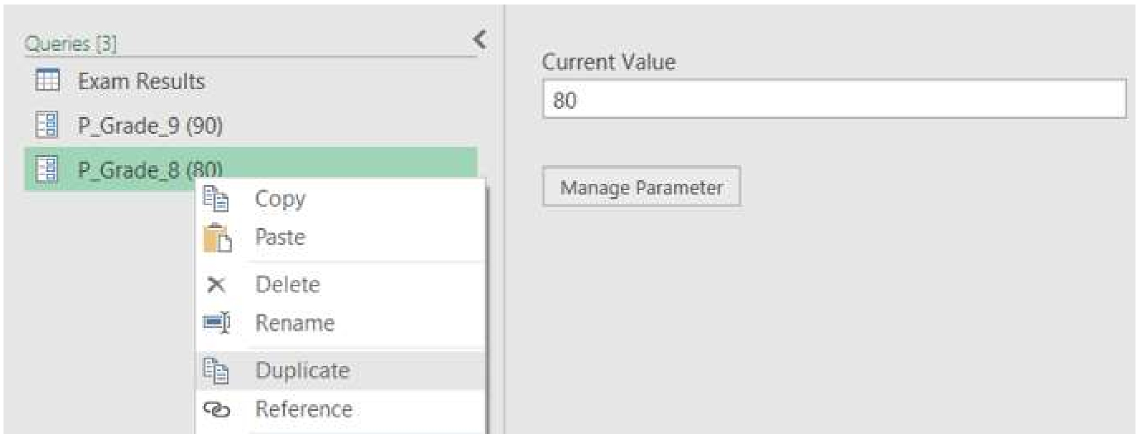
Current Value: 90

I can see the parameter in the Queries panel. If the parameter is selected, then I can edit the 'Current Value' if I wish. The 'Current Value' also appears next to the parameter in brackets '(90)' so that I can always see what it is set to in the Queries panel.

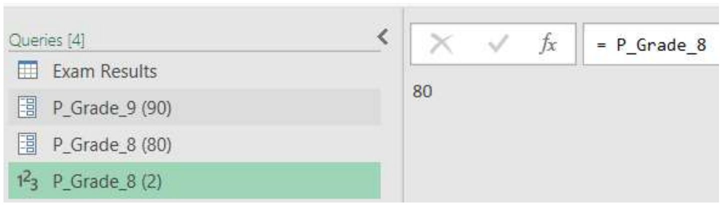
If I want to create multiple parameters, then I can stay in the ‘Manage Parameter’ dialog by clicking ‘New’ instead of ‘OK’ when I have finished each parameter:



Note that since my parameters are similar, I can also copy and paste or create Duplicates:

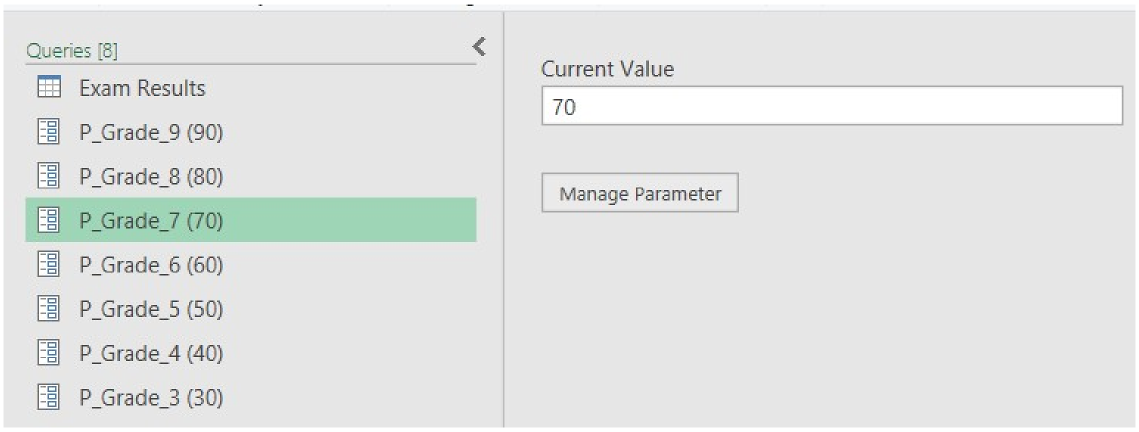


This is fine, as long as I remember to change the name, description and ‘Current Value’. Do not create a ‘Reference’. This would return the ‘Current Value’ of the parameter:



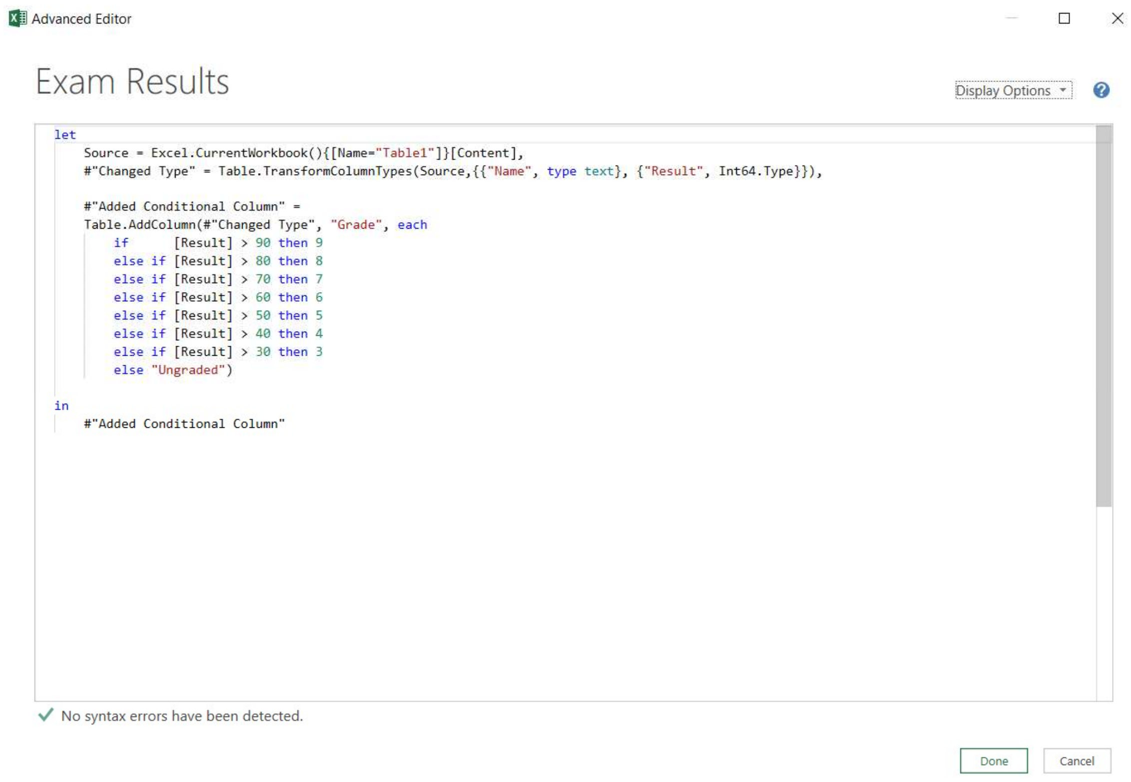
The icon next to **P_Grade_8 (2)** indicates a number. Note also that the ‘**(2)**’ in this case is created because there is already a **P_Grade_8** and is not the value!

I now have all my parameters ready for the next step:



Current Value: 70

I can now edit the original query using the Advanced Editor, which I access from the Home tab:



```
let
    Source = Excel.CurrentWorkbook(){[Name="Table1"]}[Content],
    #"Changed Type" = Table.TransformColumnTypes(Source,{{"Name", type text}, {"Result", Int64.Type}}),

    #"Added Conditional Column" =
    Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > 90 then 9
        else if [Result] > 80 then 8
        else if [Result] > 70 then 7
        else if [Result] > 60 then 6
        else if [Result] > 50 then 5
        else if [Result] > 40 then 4
        else if [Result] > 30 then 3
        else "Ungraded")

in
    #"Added Conditional Column"
```

No syntax errors have been detected.

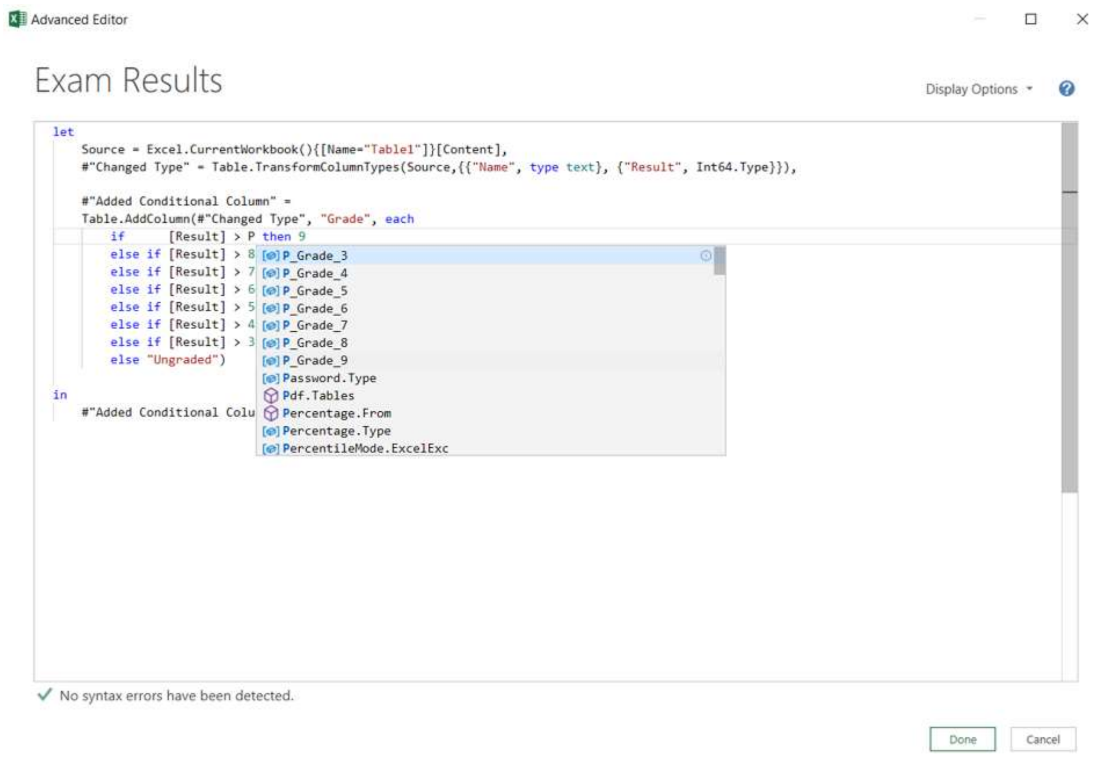
I change the **M** code in the ‘Added Conditional Column’ step from:

```
Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > 90 then 9
        else if [Result] > 80 then 8
        else if [Result] > 70 then 7
        else if [Result] > 60 then 6
        else if [Result] > 50 then 5
        else if [Result] > 40 then 4
        else if [Result] > 30 then 3
        else "Ungraded")
```

to

```
Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > P_Grade_9 then 9
        else if [Result] > P_Grade_8 then 8
        else if [Result] > P_Grade_7 then 7
        else if [Result] > P_Grade_6 then 6
        else if [Result] > P_Grade_5 then 5
        else if [Result] > P_Grade_4 then 4
        else if [Result] > P_Grade_3 then 3
        else "Ungraded")
```

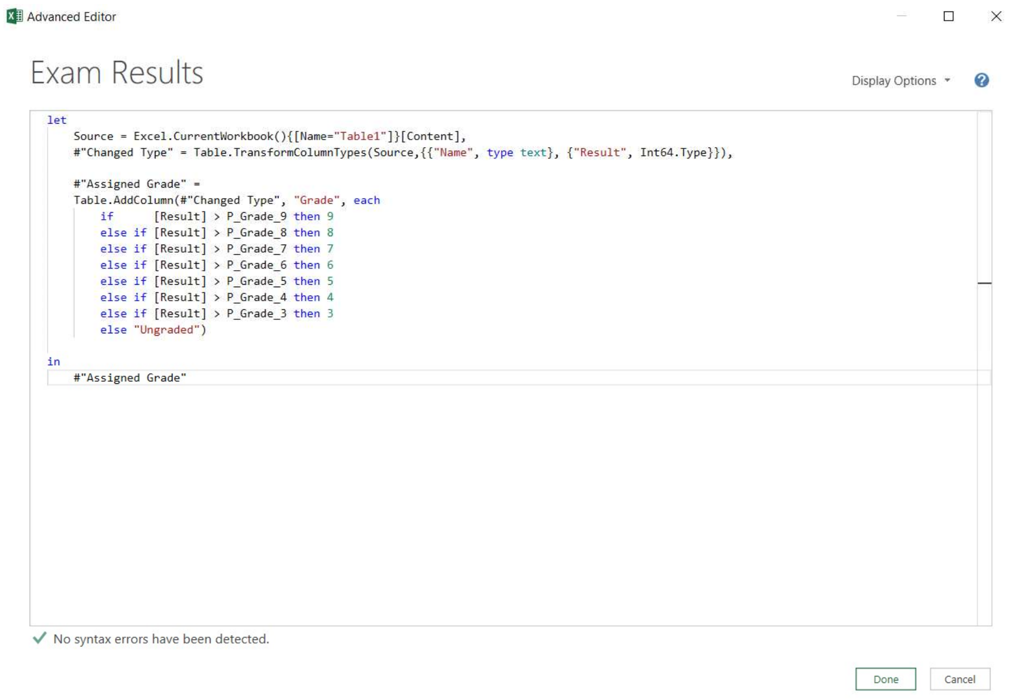
I can use the Intellisense to make sure I enter the correct name for each parameter:



```
let
    Source = Excel.CurrentWorkbook(){[Name="Table1"]}[Content],
    #"Changed Type" = Table.TransformColumnTypes(Source,{{"Name", type text}, {"Result", Int64.Type}}),

    #"Added Conditional Column" =
    Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > P then 9
        else if [Result] > 8
        else if [Result] > 7
        else if [Result] > 6
        else if [Result] > 5
        else if [Result] > 4
        else if [Result] > 3
        else "Ungraded")

in
    #"Added Conditional Colu
```

```
P_Grade_3
P_Grade_4
P_Grade_5
P_Grade_6
P_Grade_7
P_Grade_8
P_Grade_9
Password.Type
Pdf.Tables
Percentage.From
Percentage.Type
PercentileMode.ExcelExc
```

I also rename the step to ‘Assigned Grade’:



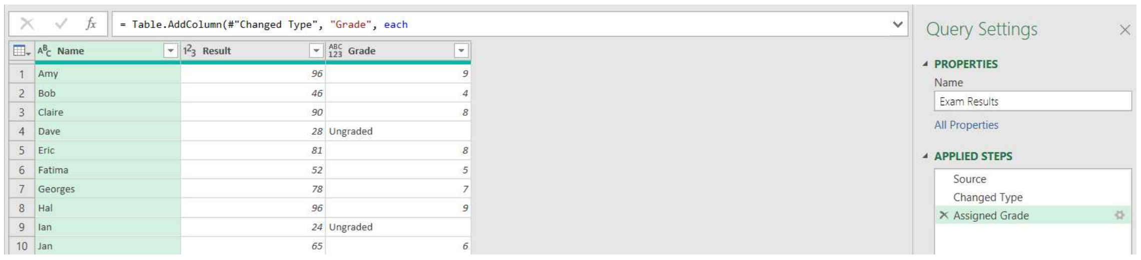
Advanced Editor

Exam Results

```
let
    Source = Excel.CurrentWorkbook(){[Name="Table1"]}[Content],
    #"Changed Type" = Table.TransformColumnTypes(Source,{{"Name", type text}, {"Result", Int64.Type}}),

    #"Assigned Grade" =
    Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > P_Grade_9 then 9
        else if [Result] > P_Grade_8 then 8
        else if [Result] > P_Grade_7 then 7
        else if [Result] > P_Grade_6 then 6
        else if [Result] > P_Grade_5 then 5
        else if [Result] > P_Grade_4 then 4
        else if [Result] > P_Grade_3 then 3
        else "Ungraded")

in
    #"Assigned Grade"
```

No syntax errors have been detected.

I click ‘Done’ to make sure that the query still works as I expect:



```
= Table.AddColumn(#"Changed Type", "Grade", each
```

	Name	Result	Grade
1	Amy	96	9
2	Bob	46	4
3	Claire	90	8
4	Dave	28	Ungraded
5	Eric	81	8
6	Fatima	52	5
7	Georges	78	7
8	Hal	96	9
9	Ian	24	Ungraded
10	Jan	65	6

Query Settings

PROPERTIES

Name: Exam Results

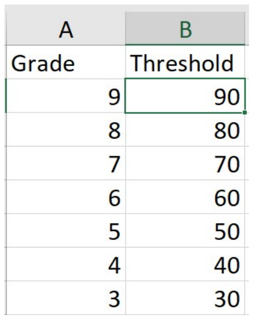
All Properties

APPLIED STEPS

- Source
- Changed Type
- Assigned Grade

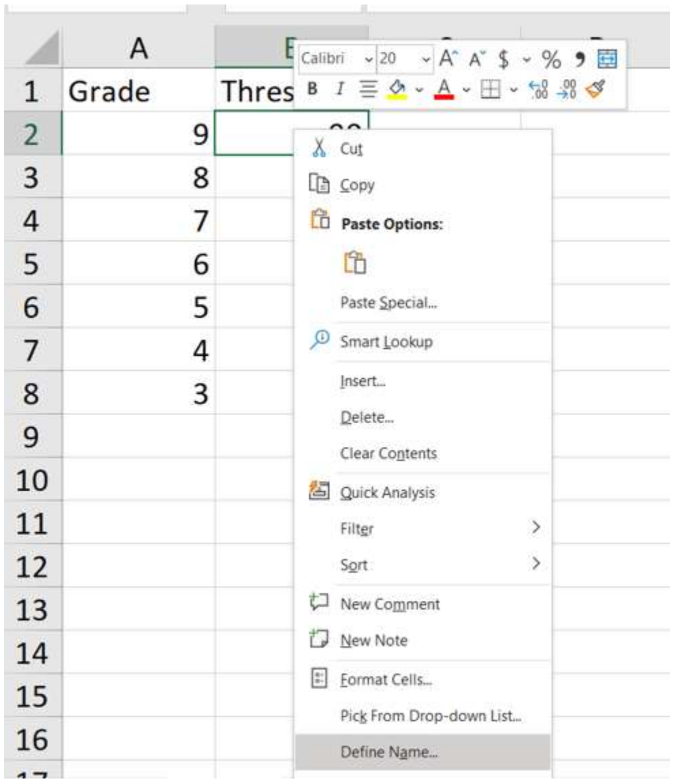
Using Defined Names as Parameters

Whilst I can change these parameters in Power Query, I'd now like to have parameters that I can change from Excel. On a new Excel Sheet, I have some data for the thresholds:

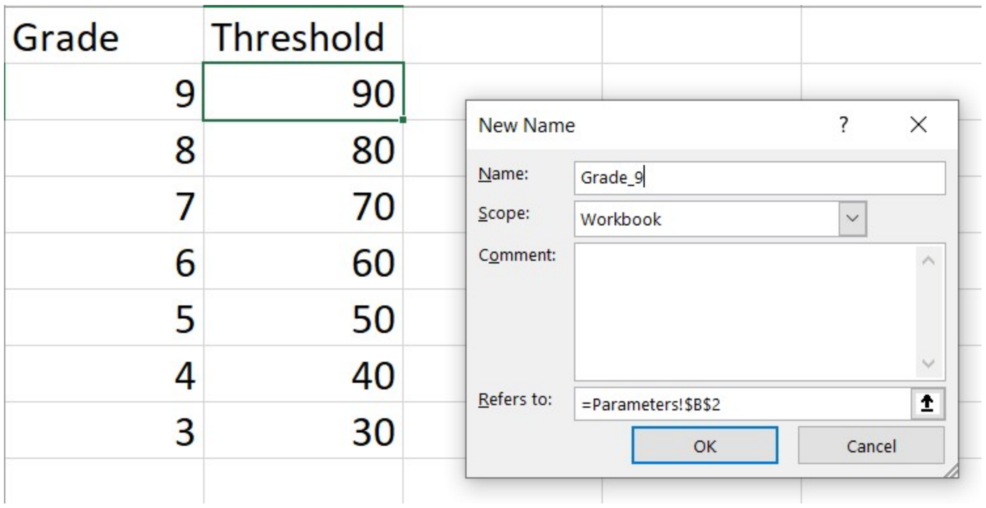


Grade	Threshold
9	90
8	80
7	70
6	60
5	50
4	40
3	30

I start by defining a Name for the first threshold. I can do this by selecting the cell and right-clicking:



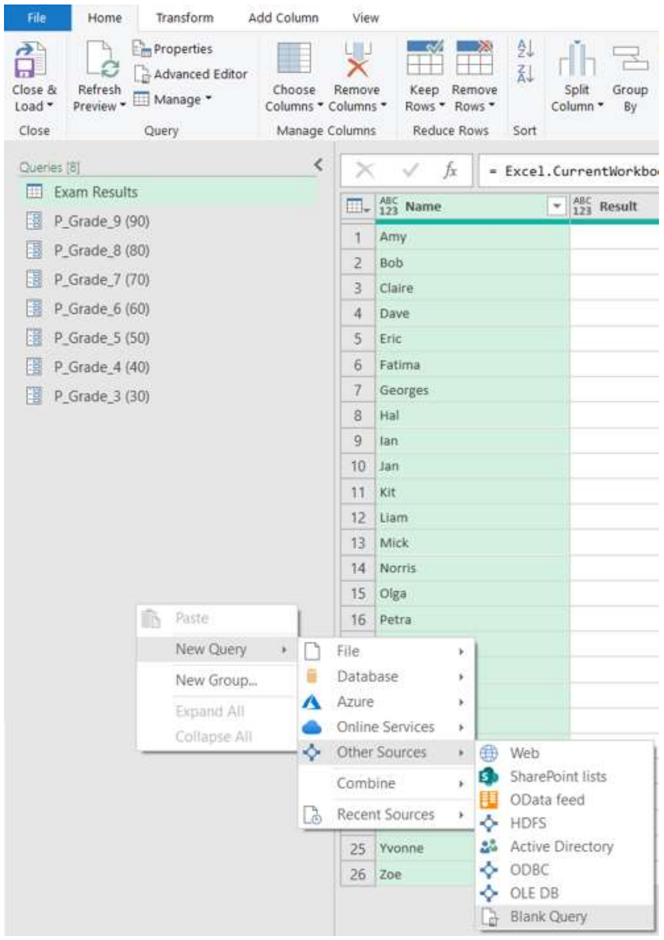
I define the Name to be 'Grade_9':



Grade	Threshold
9	90
8	80
7	70
6	60
5	50
4	40
3	30

Name:	Grade_9
Scope:	Workbook
Comment:	
Refers to:	=Parameters!$B$2

I can now see this in Power Query. In the Power Query Editor, I create a new Blank Query. I can do this by right-clicking in the Queries pane (this is one of several methods to create a Blank Query):



Queries [8]:
- Exam Results
- P_Grade_9 (90)
- P_Grade_8 (80)
- P_Grade_7 (70)
- P_Grade_6 (60)
- P_Grade_5 (50)
- P_Grade_4 (40)
- P_Grade_3 (30)

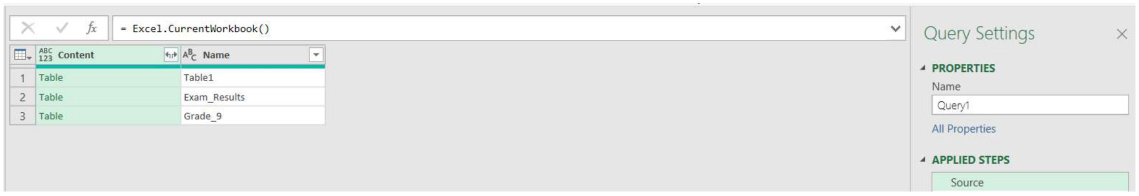
= Excel.CurrentWorkbo

	Name	Result
1	Amy	
2	Bob	
3	Claire	
4	Dave	
5	Eric	
6	Fatima	
7	Georges	
8	Hal	
9	Ian	
10	Jan	
11	Kit	
12	Liam	
13	Mick	
14	Norris	
15	Olga	
16	Petra	
25	Yvonne	
26	Zoe	

In my Blank Query, I enter the following **M** code:

= Excel.CurrentWorkbook()

This will show me what is in the current Excel Workbook:

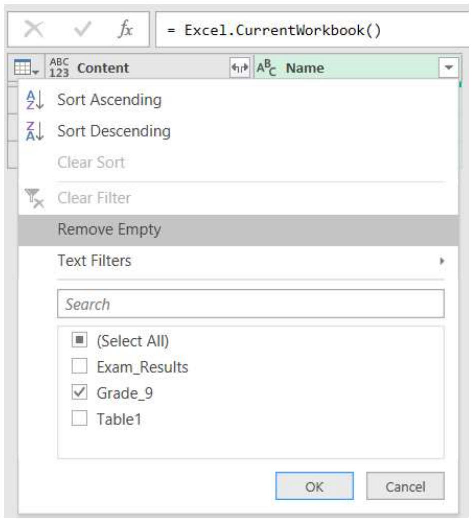


```
= Excel.CurrentWorkbook()
```

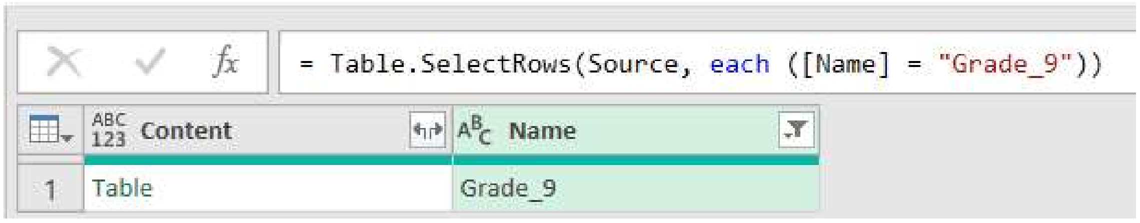
	Content	Name
1	Table	Table1
2	Table	Exam_Results
3	Table	Grade_9

There is the **Grade_9** I created. The value will be in the 'Table' next to it.

I filter **Name** to get just the 'Grade_9' row:



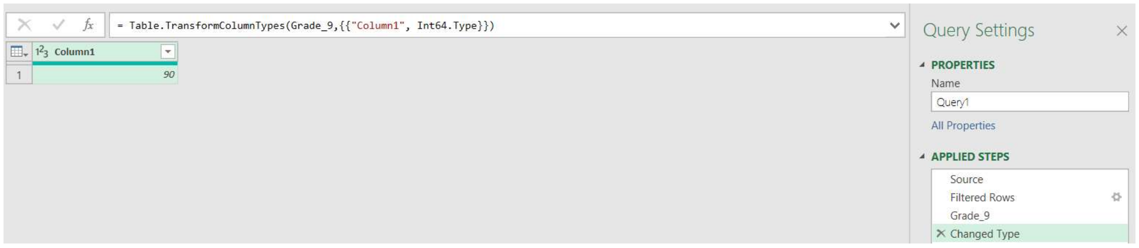
This gives me just one table:



```
= Table.SelectRows(Source, each ([Name] = "Grade_9"))
```

	Content	Name
1	Table	Grade_9

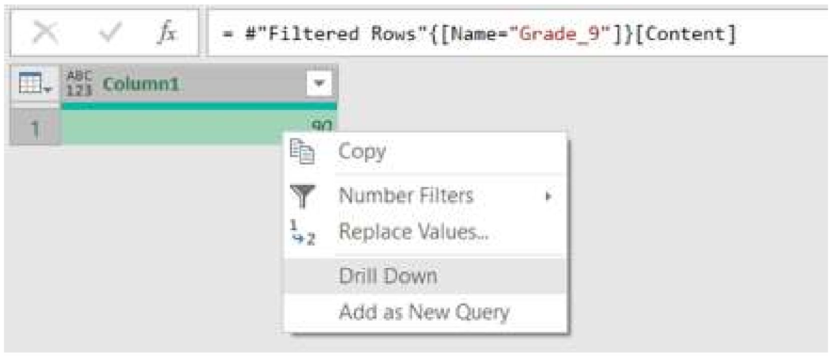
I click on the green 'Table'.



```
= Table.TransformColumnTypes(Grade_9,{{"Column1", Int64.Type}})
```

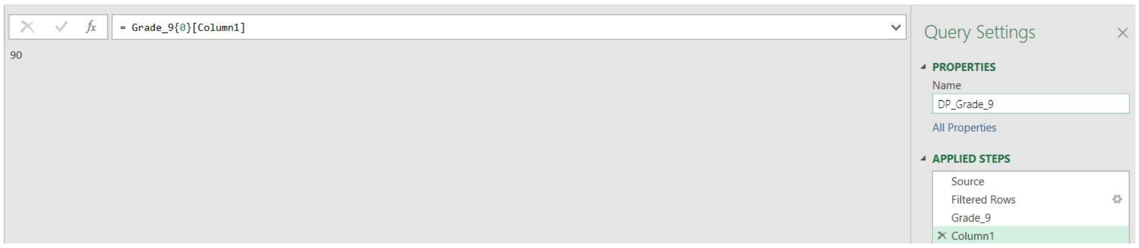
	Column1
1	90

I now have the value of **Grade_9** in the column. I can remove the 'Changed Type' step and right-click and drill down on the value.



```
= #"Filtered Rows"{[Name="Grade_9"]}[Content]
```

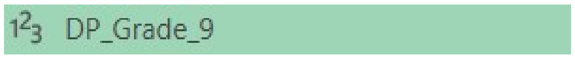
This gives me the value, and I now rename my query **DP_Grade_9**:



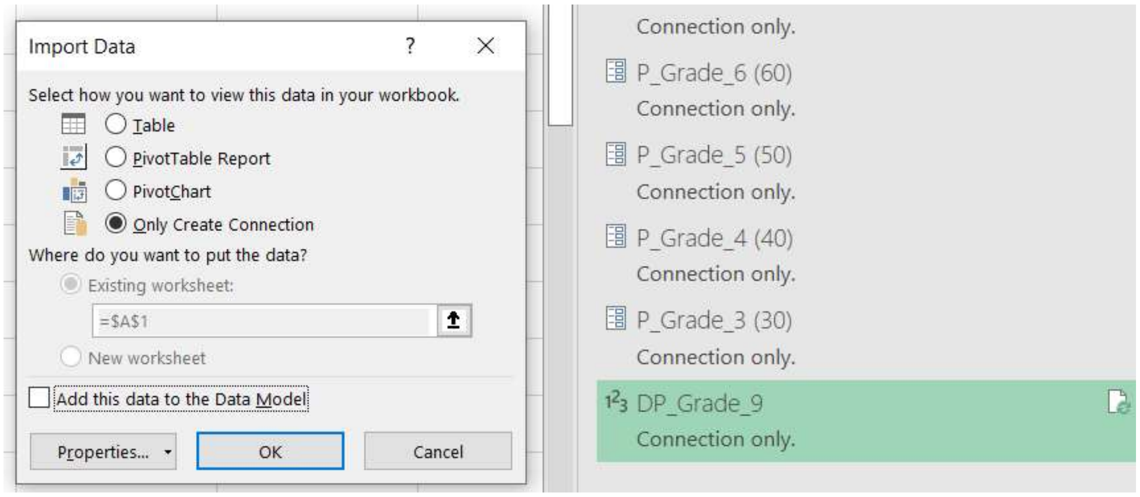
```
= Grade_9{0}[Column1]
```

90

Since this query returns a value, the icon next to it indicates a whole number:



Note that when I 'Close & Load' my queries to Excel, I should make sure that **DP_Grade_9** is set to 'Connection Only':



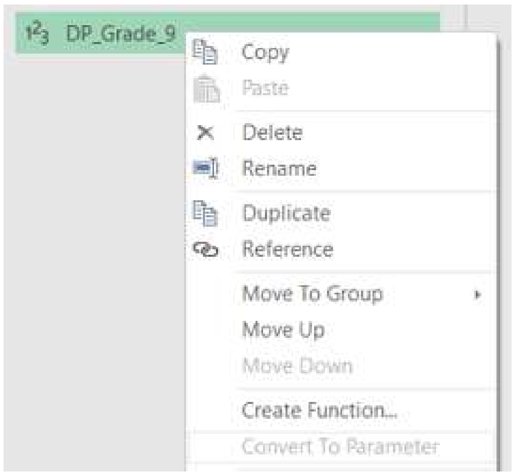
Import Data

Select how you want to view this data in your workbook.
- Table
- PivotTable Report
- PivotChart
- [✓ Only Create Connection]

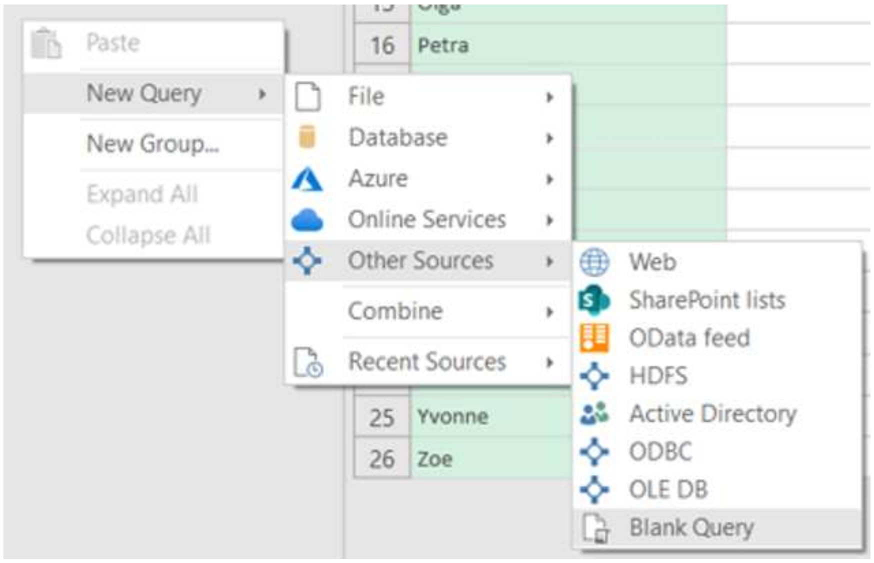
Where do you want to put the data?
- Existing worksheet: =$A$1
- New worksheet

Add this data to the Data Model

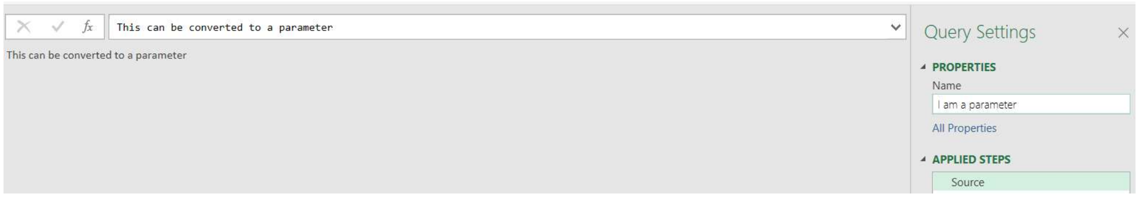
This is a default for queries created as Parameters in Power Query, but not for **DP_Grade_9**, as it has been created from a query. This brings me to another point. I right-click on **DP_Grade_9** in the Queries pane. There is an option to 'Convert to Parameter' but it is greyed out.



Whilst the final result of my query is a single value, I am not allowed to convert it. This option is only available if I create a very simple query which equals a value. I can create a new Blank Query to demonstrate this by right-clicking in the Queries pane:



I create a query which is set to a single text value:



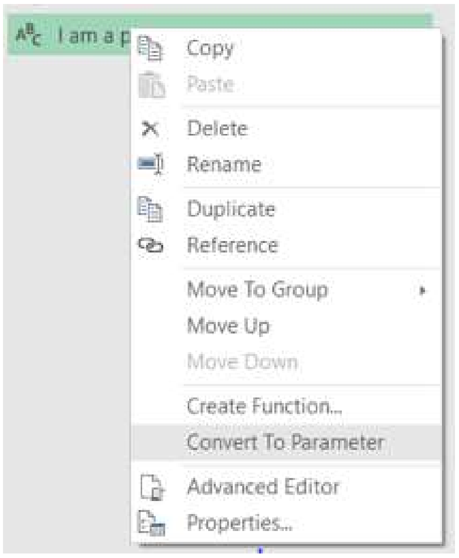
```
This can be converted to a parameter
```

This can be converted to a parameter

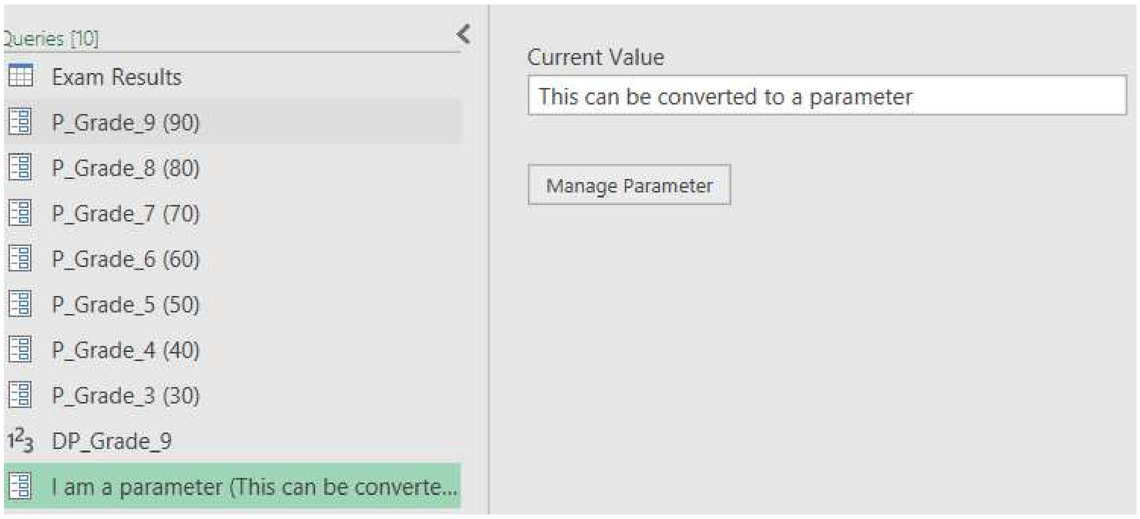
The **M** code I used to create this is simply:

```
= “This can be converted to a parameter”
```

When I right click on this query in the Queries pane,



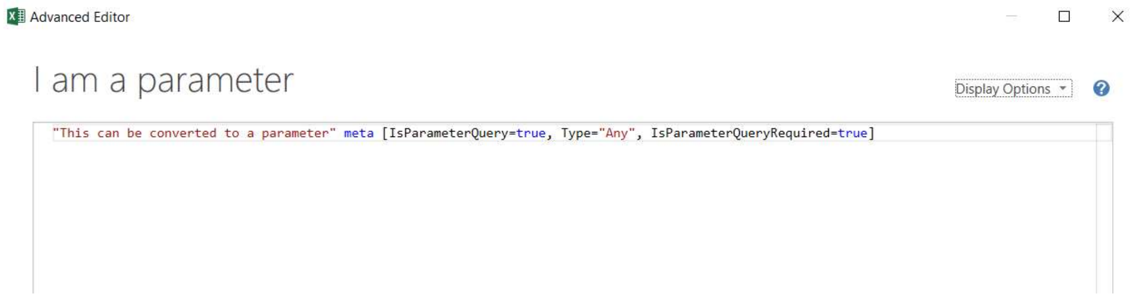
I can ‘Convert to Parameter’ and it looks just like the other ‘**P_Grade**…’ parameters that I created:


Queries [10]

- Exam Results
- P_Grade_9 (90)
- P_Grade_8 (80)
- P_Grade_7 (70)
- P_Grade_6 (60)
- P_Grade_5 (50)
- P_Grade_4 (40)
- P_Grade_3 (30)
- DP_Grade_9
- I am a parameter (This can be converte...

Current Value: This can be converted to a parameter

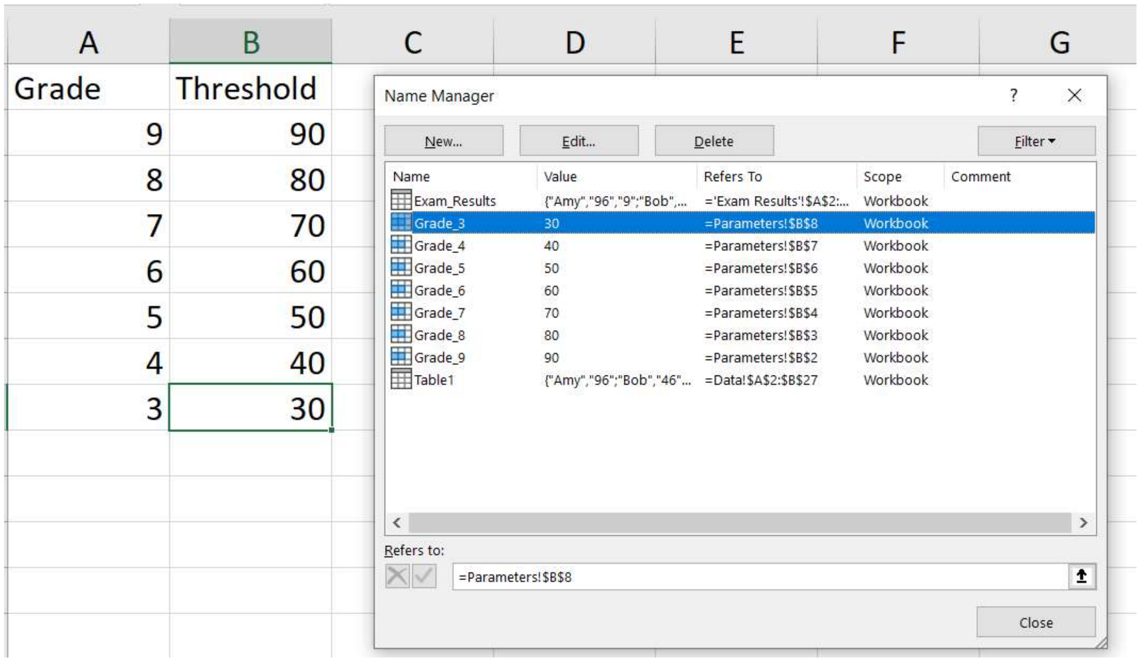
I would like this to be available for queries like **DP_Grade_9** too, so that I could have the current value in brackets and the ability to select it as a parameter from other functions. However, this is not an option. I suspect this is because the query is converted to Metadata, as indicated by the Advanced Editor view of **I am a parameter**:



```
"This can be converted to a parameter" meta [IsParameterQuery=true, Type="Any", IsParameterQueryRequired=true]
```

The previous source step is no longer available. This would imply that I can't keep the previous steps of **DP_Grade_9** and convert it to a parameter. I've seen this question on forums, and this is my conclusion!

I now define names for all the cells that I want to use as parameters:

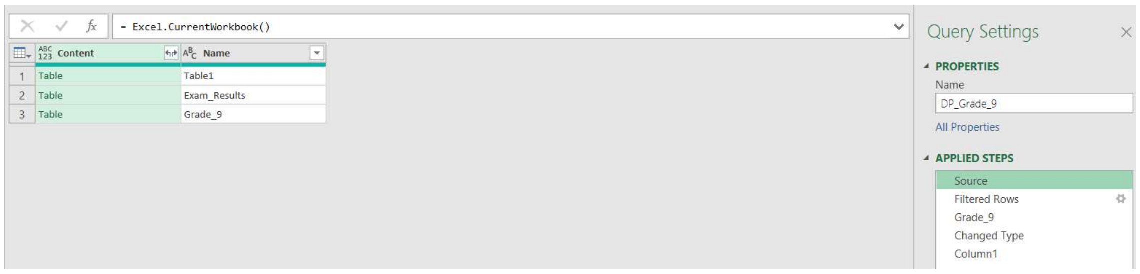


Grade	Threshold
9	90
8	80
7	70
6	60
5	50
4	40
3	30

Name	Value	Refers To	Scope	Comment
Exam_Results	{"Amy","96","9";"Bob",...	='Exam Results'!$A$2:...	Workbook	
Grade_3	30	=Parameters!$B$8	Workbook	
Grade_4	40	=Parameters!$B$7	Workbook	
Grade_5	50	=Parameters!$B$6	Workbook	
Grade_6	60	=Parameters!$B$5	Workbook	
Grade_7	70	=Parameters!$B$4	Workbook	
Grade_8	80	=Parameters!$B$3	Workbook	
Grade_9	90	=Parameters!$B$2	Workbook	
Table1	{"Amy","96";"Bob","46"...	=Data!$A$2:$B$27	Workbook	

Refers to: =Parameters!$B$8

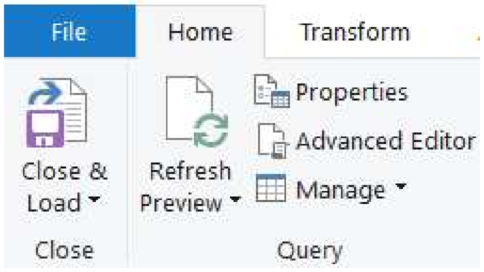
If I go to the **DP_Grade_9** query in the Power Query Editor, I can view the Source step:



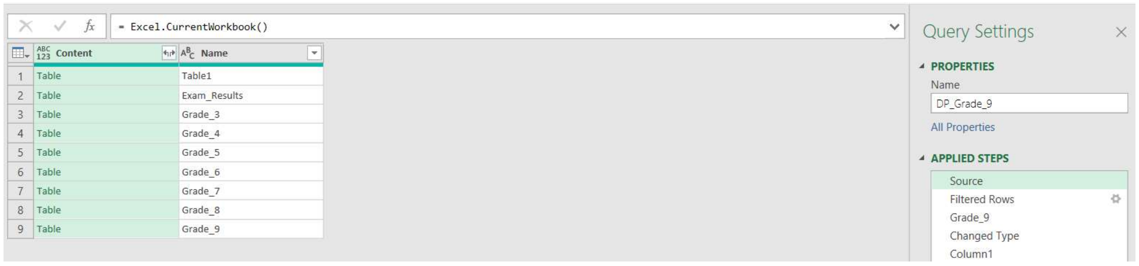
```
= Excel.CurrentWorkbook()
```

	Content	Name
1	Table	Table1
2	Table	Exam_Results
3	Table	Grade_9

I 'Refresh Preview', using the option on the Home tab:



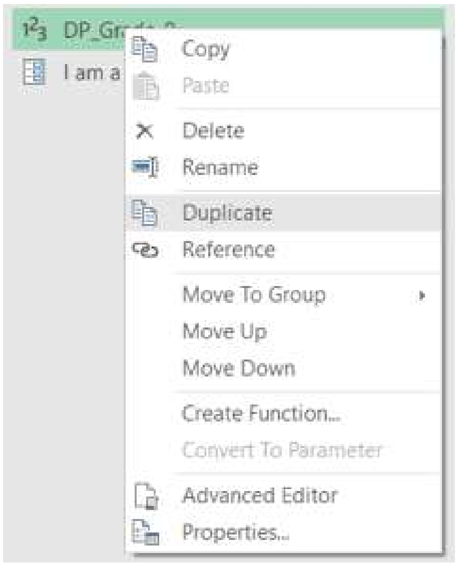
I notice that the other named cells now appear:



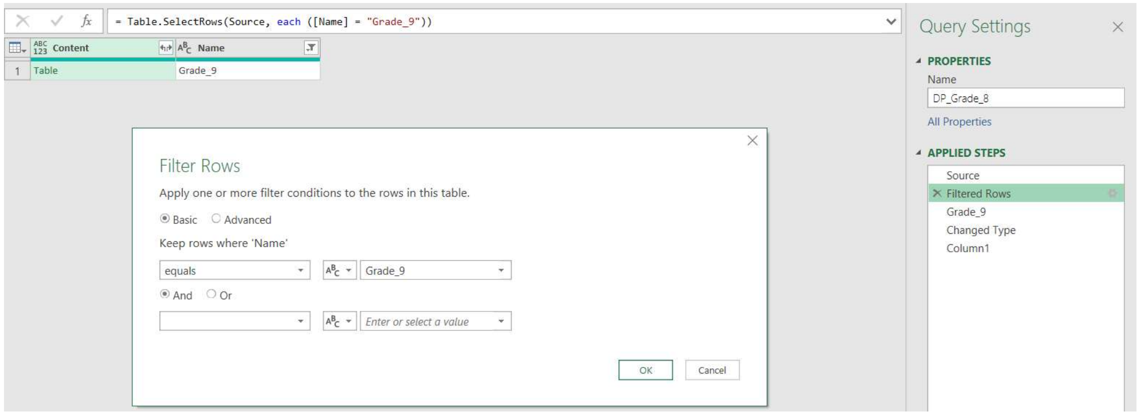
```
= Excel.CurrentWorkbook()
```

	Content	Name
1	Table	Table1
2	Table	Exam_Results
3	Table	Grade_3
4	Table	Grade_4
5	Table	Grade_5
6	Table	Grade_6
7	Table	Grade_7
8	Table	Grade_8
9	Table	Grade_9

I can create a duplicate of **DP_Grade_9**:



I can use this as a template to create the other queries:



```
= Table.SelectRows(Source, each ([Name] = "Grade_9"))
```

	Content	Name
1	Table	Grade_9

Filter Rows

Apply one or more filter conditions to the rows in this table.

Basic [✓] Advanced

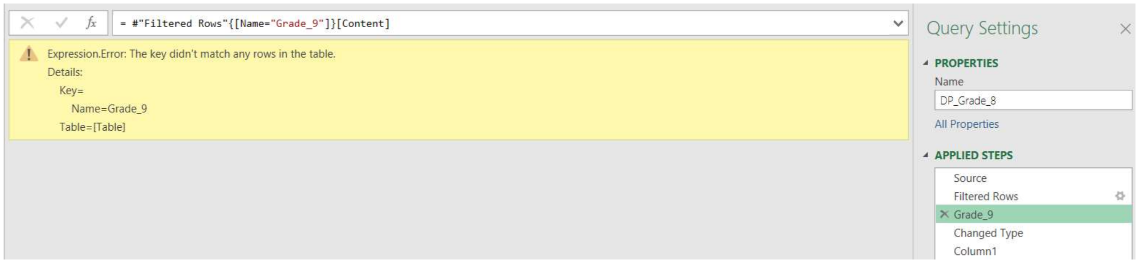
Keep rows where 'Name'

equals | Grade_9

And [✓] Or

Enter or select a value

I have renamed the duplicate query, and I click on the cog next to 'Filtered Rows' to amend the value of **Name** selected to 'Grade_8'. Note that the step 'Grade_9' will also need to be edited, as I will see as soon as I click OK and move to that step.



```
= #"Filtered Rows"{[Name="Grade_9"]}[Content]
```

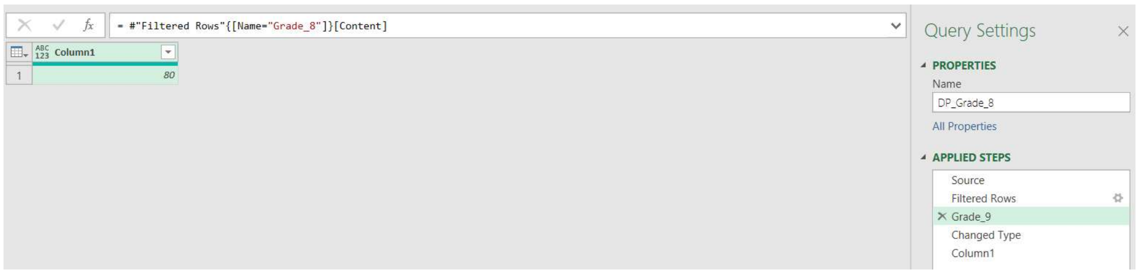
```
Expression.Error: The key didn't match any rows in the table.
Details:
    Key=
        Name=Grade_9
    Table=[Table]
```

This step is trying to expand the **Content** column which corresponds to the **Name** 'Grade_9'. The **M** code is:

```
= #"Filtered Rows"{[Name="Grade_9"]}[Content]
```

I can change this to:

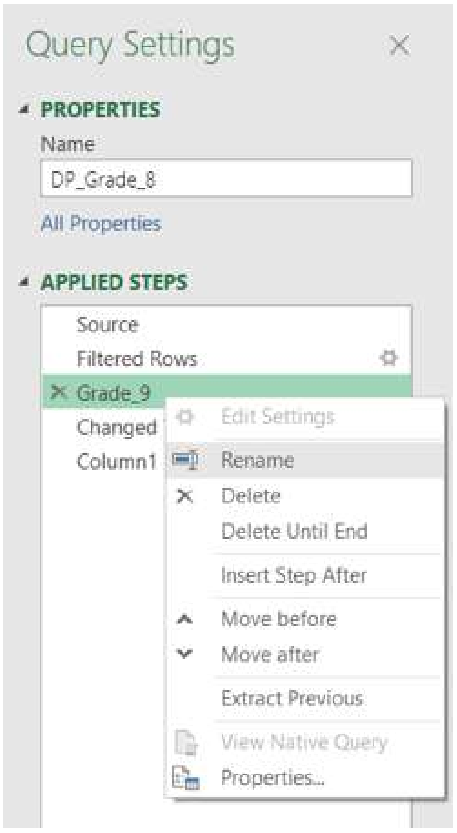
```
= #"Filtered Rows"{[Name="Grade_8"]}[Content]
```



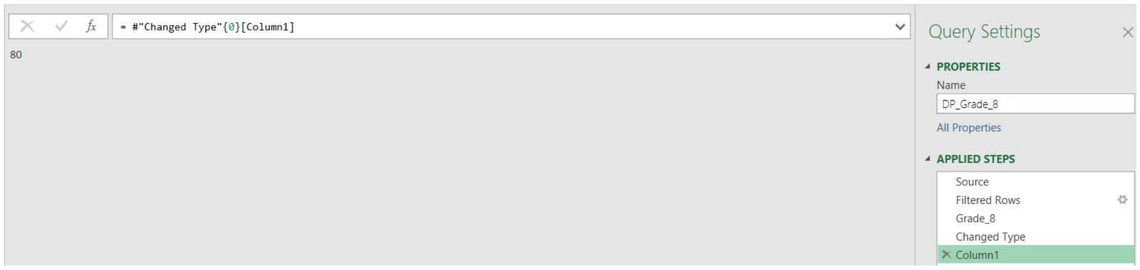
```
= #"Filtered Rows"{[Name="Grade_8"]}[Content]
```

	Column1
1	80

I have the correct result, but I should also right-click on the step and change the name to avoid confusion:



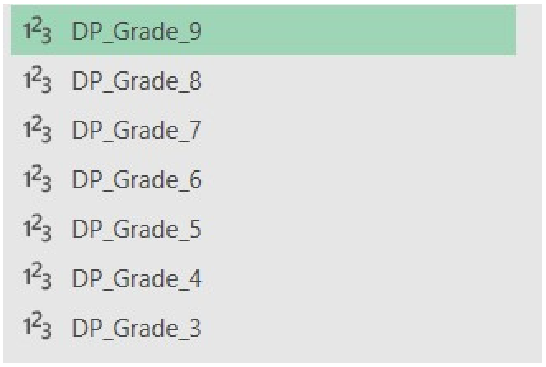
I have completed this query:



```
= #"Changed Type"{0}[Column1]
```

80

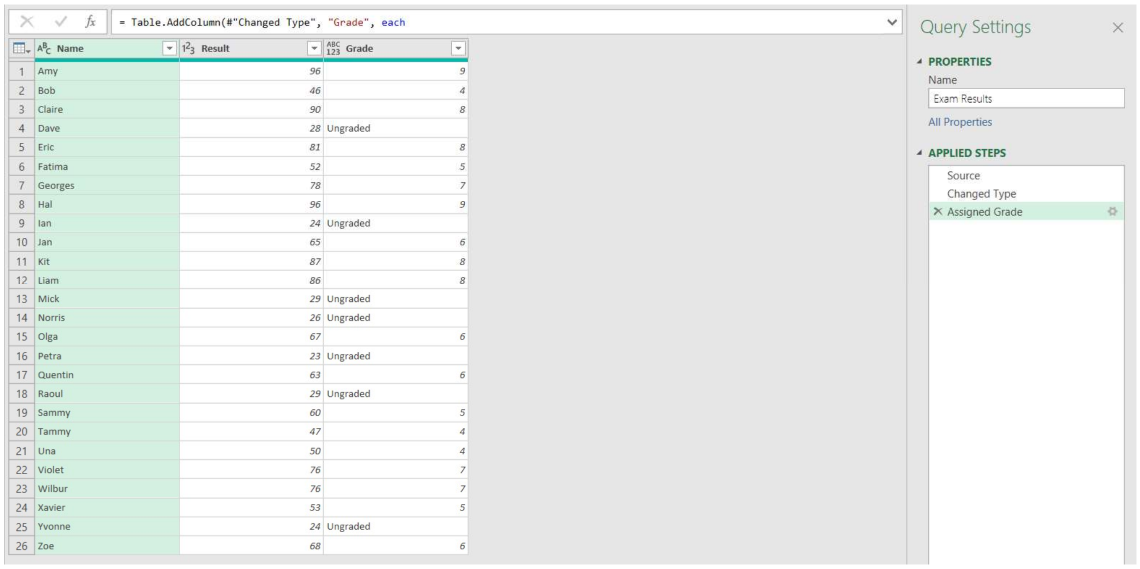
I repeat this process for the other named cells:



DP_Grade_9
DP_Grade_8
DP_Grade_7
DP_Grade_6
DP_Grade_5
DP_Grade_4
DP_Grade_3

I will now apply these parameters to the **Exam Results** query and check that any changes to the Excel cells affect the outcome of the query.

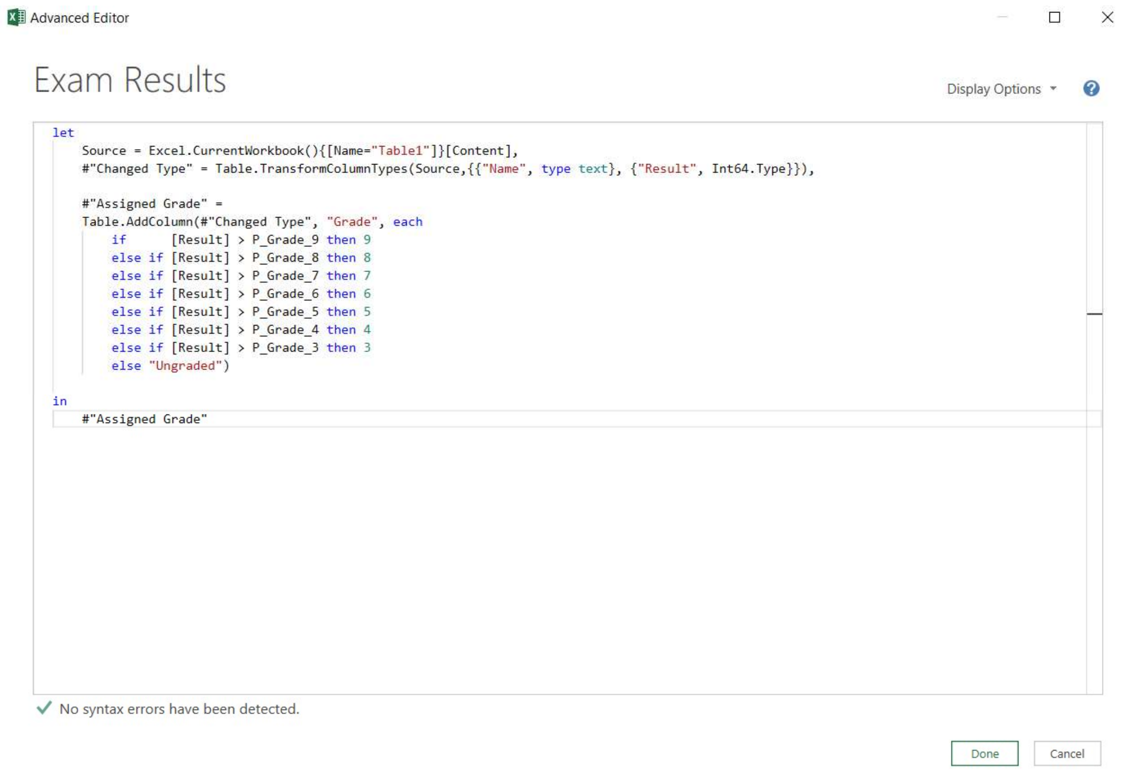
I now return to the **Exam Results** query.



```
= Table.AddColumn(#"Changed Type", "Grade", each
```

	Name	Result	Grade
1	Amy	96	9
2	Bob	46	4
3	Claire	90	8
4	Dave	28	Ungraded
5	Eric	81	8
6	Fatima	52	5
7	Georges	78	7
8	Hal	96	9
9	Ian	24	Ungraded
10	Jan	65	6
11	Kit	87	8
12	Liam	86	8
13	Mick	29	Ungraded
14	Norris	26	Ungraded
15	Olga	67	6
16	Petra	23	Ungraded
17	Quentin	63	6
18	Raoul	29	Ungraded
19	Sammy	60	5
20	Tammy	47	4
21	Una	50	4
22	Violet	76	7
23	Wilbur	76	7
24	Xavier	53	5
25	Yvonne	24	Ungraded
26	Zoe	68	6

The easiest way to change the query to use the Excel controlled parameters is by editing in the Advanced Editor, which I can access from the Home tab:



Exam Results

```
let
    Source = Excel.CurrentWorkbook(){[Name="Table1"]}[Content],
    #"Changed Type" = Table.TransformColumnTypes(Source,{{"Name", type text}, {"Result", Int64.Type}}),

    #"Assigned Grade" =
    Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > P_Grade_9 then 9
        else if [Result] > P_Grade_8 then 8
        else if [Result] > P_Grade_7 then 7
        else if [Result] > P_Grade_6 then 6
        else if [Result] > P_Grade_5 then 5
        else if [Result] > P_Grade_4 then 4
        else if [Result] > P_Grade_3 then 3
        else "Ungraded")

in
    #"Assigned Grade"
```

✓ No syntax errors have been detected.

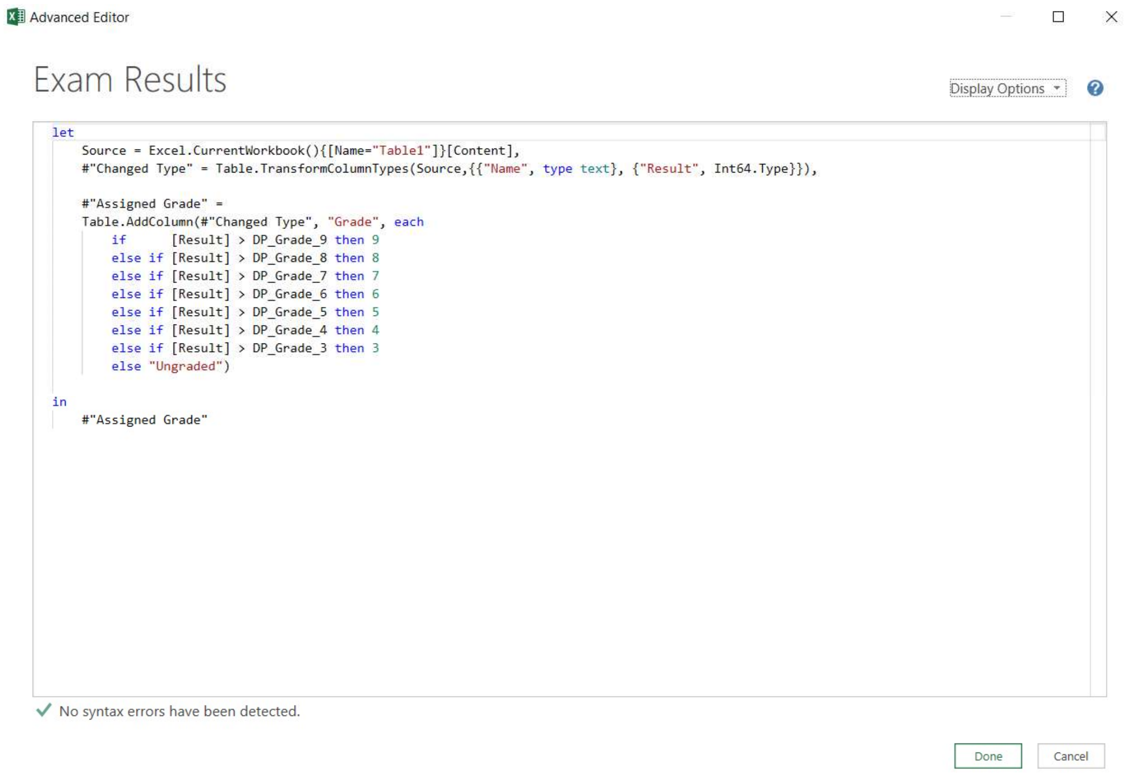
The ‘Assigned Grade’ step is changed from:

```
#"Assigned Grade" =
    Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > P_Grade_9 then 9
        else if [Result] > P_Grade_8 then 8
        else if [Result] > P_Grade_7 then 7
        else if [Result] > P_Grade_6 then 6
        else if [Result] > P_Grade_5 then 5
        else if [Result] > P_Grade_4 then 4
        else if [Result] > P_Grade_3 then 3
        else "Ungraded")
```

to

```
#"Assigned Grade" =
    Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > DP_Grade_9 then 9
        else if [Result] > DP_Grade_8 then 8
        else if [Result] > DP_Grade_7 then 7
        else if [Result] > DP_Grade_6 then 6
        else if [Result] > DP_Grade_5 then 5
        else if [Result] > DP_Grade_4 then 4
        else if [Result] > DP_Grade_3 then 3
        else "Ungraded")
```

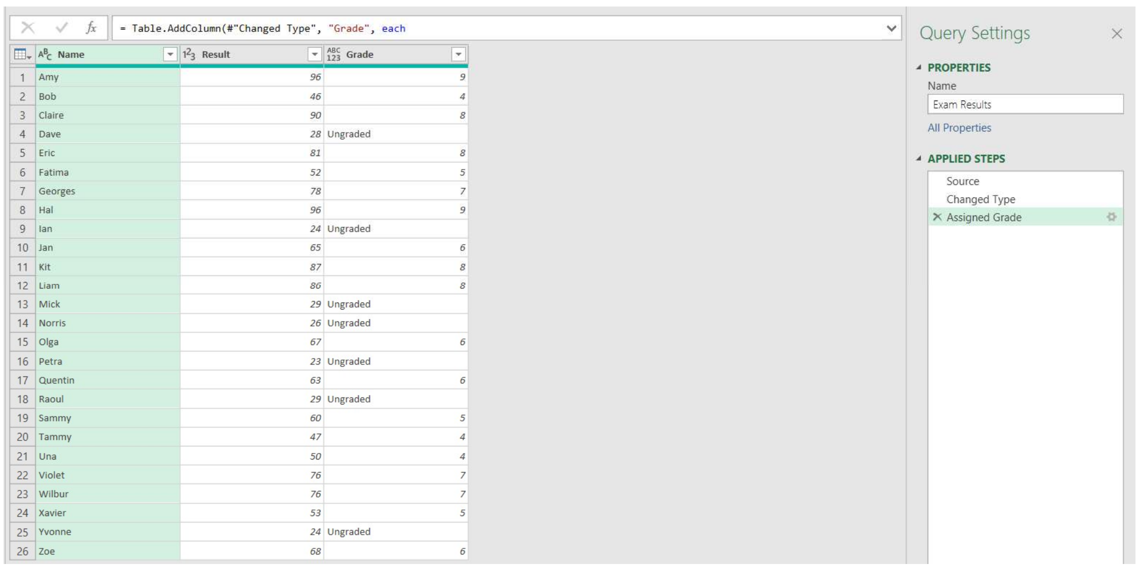
This works because each '**DP_**' query represents one value:



```
let
    Source = Excel.CurrentWorkbook(){[Name="Table1"]}[Content],
    #"Changed Type" = Table.TransformColumnTypes(Source,{{"Name", type text}, {"Result", Int64.Type}}),

    #"Assigned Grade" =
    Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > DP_Grade_9 then 9
        else if [Result] > DP_Grade_8 then 8
        else if [Result] > DP_Grade_7 then 7
        else if [Result] > DP_Grade_6 then 6
        else if [Result] > DP_Grade_5 then 5
        else if [Result] > DP_Grade_4 then 4
        else if [Result] > DP_Grade_3 then 3
        else "Ungraded")

in
    #"Assigned Grade"
```

No syntax errors have been detected.

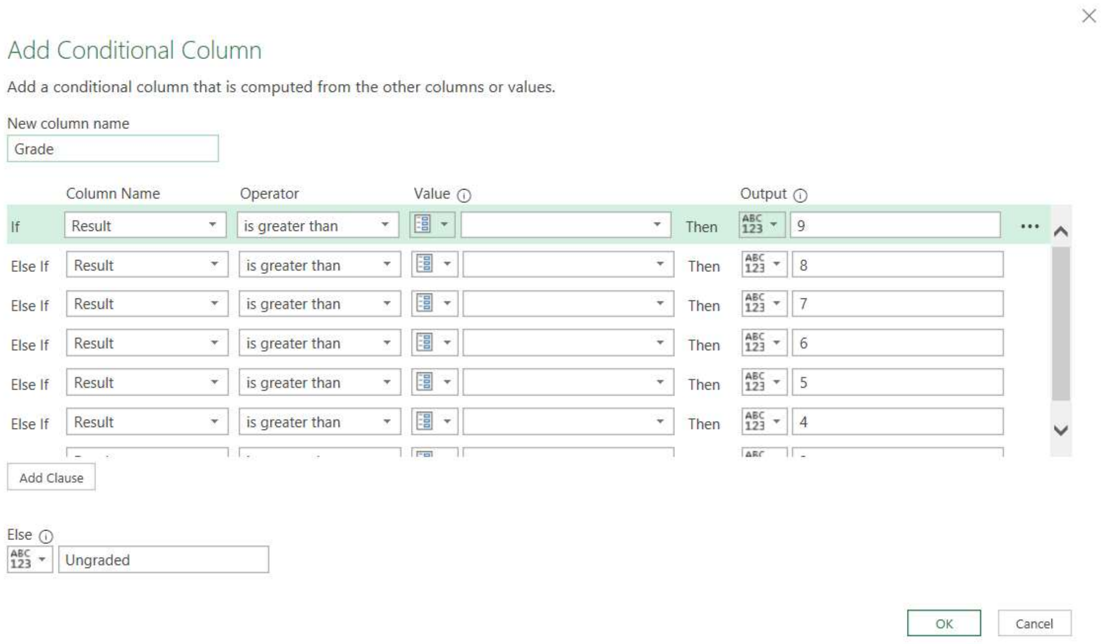
This currently has no effect on the results of the query, since the values are the same:



```
= Table.AddColumn(#"Changed Type", "Grade", each
```

	Name	Result	Grade
1	Amy	96	9
2	Bob	46	4
3	Claire	90	8
4	Dave	28	Ungraded
5	Eric	81	8
6	Fatima	52	5
7	Georges	78	7
8	Hal	96	9
9	Ian	24	Ungraded
10	Jan	65	6
11	Kit	87	8
12	Liam	86	8
13	Mick	29	Ungraded
14	Norris	26	Ungraded
15	Olga	67	6
16	Petra	23	Ungraded
17	Quentin	63	6
18	Raoul	29	Ungraded
19	Sammy	60	5
20	Tammy	47	4
21	Una	50	4
22	Violet	76	7
23	Wilbur	76	7
24	Xavier	53	5
25	Yvonne	24	Ungraded
26	Zoe	68	6

However, it does have an effect on the screen accessed by clicking on the cog (gear icon) next to the 'Assigned Grade' step:



Add Conditional Column

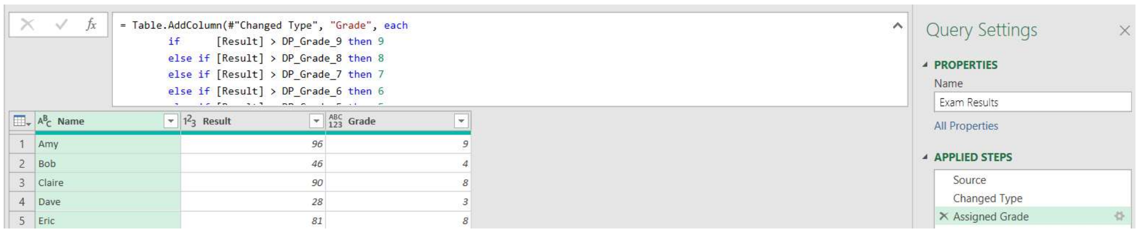
Add a conditional column that is computed from the other columns or values.

New column name: Grade

	Column Name	Operator	Value		Output
If	Result	is greater than		Then	9
Else If	Result	is greater than		Then	8
Else If	Result	is greater than		Then	7
Else If	Result	is greater than		Then	6
Else If	Result	is greater than		Then	5
Else If	Result	is greater than		Then	4

Else: Ungraded

Note that I cannot view the **Value** column now. Any changes must be made directly to the **M** code, either from the Advanced Editor or the Formula Bar:



```
= Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > DP_Grade_9 then 9
        else if [Result] > DP_Grade_8 then 8
        else if [Result] > DP_Grade_7 then 7
        else if [Result] > DP_Grade_6 then 6
```

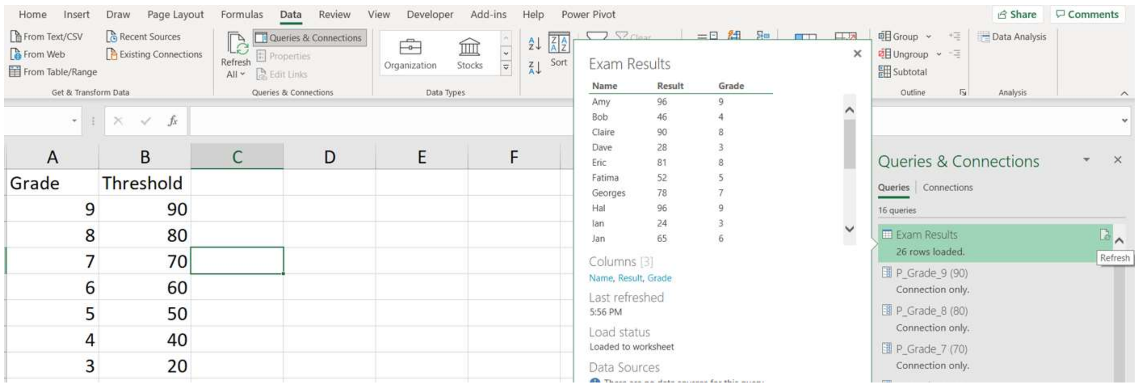
	Name	Result	Grade
1	Amy	96	9
2	Bob	46	4
3	Claire	90	8
4	Dave	28	3
5	Eric	81	8

Query Settings

PROPERTIES
Name: Exam Results
All Properties

APPLIED STEPS
Source
Changed Type
Assigned Grade

Back in Excel, if I change the Named cell **Grade_3** from 30 to 20 percent, the outcome will change when I refresh the **Exam Results** query:



	A	B
	Grade	Threshold
	9	90
	8	80
	7	70
	6	60
	5	50
	4	40
	3	20

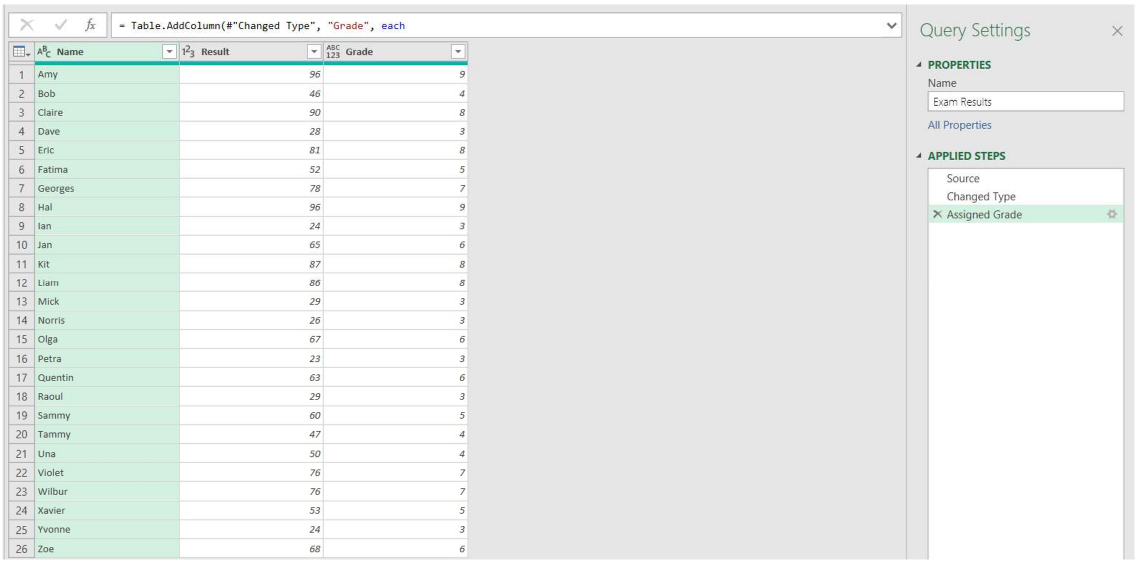
Exam Results

Name	Result	Grade
Amy	96	9
Bob	46	4
Claire	90	8
Dave	28	3
Eric	81	8
Fatima	52	5
Georges	78	7
Hal	96	9
Ian	24	3
Jan	65	6

Columns [3]
Name, Result, Grade
Last refreshed
5:56 PM
Load status
Loaded to worksheet
Data Sources

Queries & Connections
Queries | Connections
16 queries
Exam Results – 26 rows loaded.
P_Grade_9 (90) – Connection only.
P_Grade_8 (80) – Connection only.
P_Grade_7 (70) – Connection only.

If I go back to **Exam Results** and view all the data, I can see that everyone has passed now!



```
= Table.AddColumn(#"Changed Type", "Grade", each
```

	Name	Result	Grade
1	Amy	96	9
2	Bob	46	4
3	Claire	90	8
4	Dave	28	3
5	Eric	81	8
6	Fatima	52	5
7	Georges	78	7
8	Hal	96	9
9	Ian	24	3
10	Jan	65	6
11	Kit	87	8
12	Liam	86	8
13	Mick	29	3
14	Norris	26	3
15	Olga	67	6
16	Petra	23	3
17	Quentin	63	6
18	Raoul	29	3
19	Sammy	60	5
20	Tammy	47	4
21	Una	50	4
22	Violet	76	7
23	Wilbur	76	7
24	Xavier	53	5
25	Yvonne	24	3
26	Zoe	68	6

Query Settings

PROPERTIES

Name: Exam Results

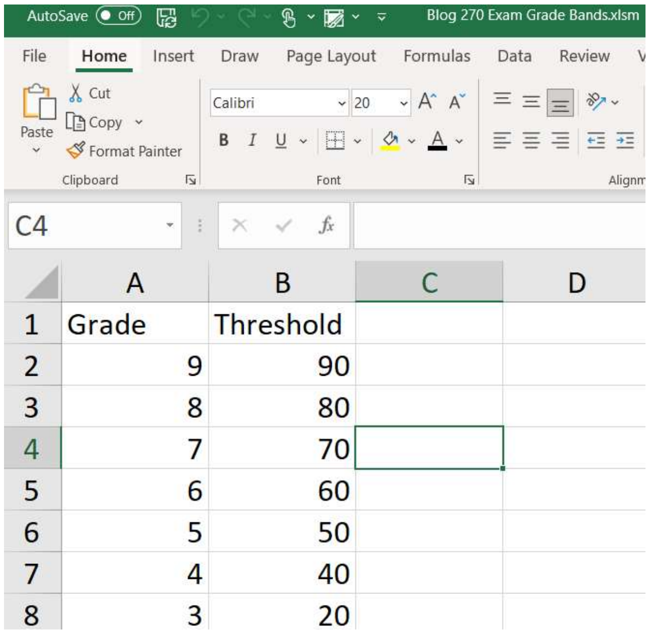
All Properties

APPLIED STEPS

- Source
- Changed Type
- Assigned Grade

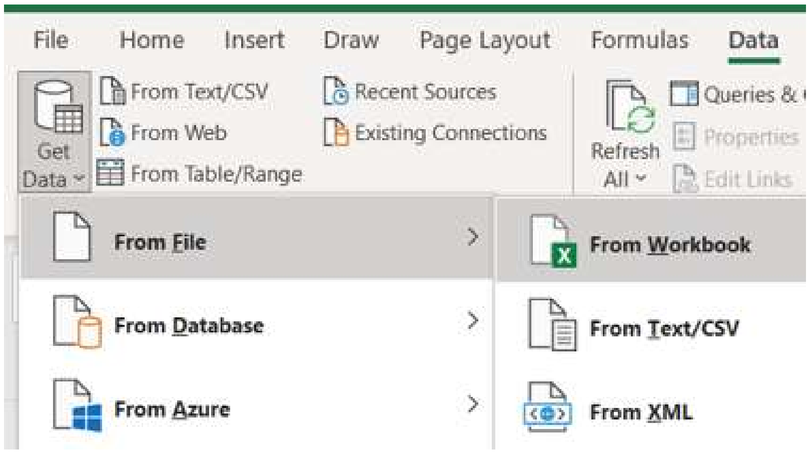
Using Parameters as Locations

However, I now want to store the grading bands in a separate workbook away from the exam results.

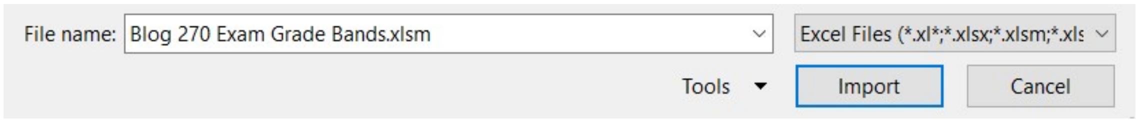


	A	B	C	D
1	Grade	Threshold		
2	9	90		
3	8	80		
4	7	70		
5	6	60		
6	5	50		
7	4	40		
8	3	20		

Back in my original workbook, I need to extract the data. I can do this by creating a new query from the 'Get Data' dropdown in the 'Get & Transform' section of the Data tab. Note that to extract data 'From Workbook', the Excel file I am extracting from must **not** be open.

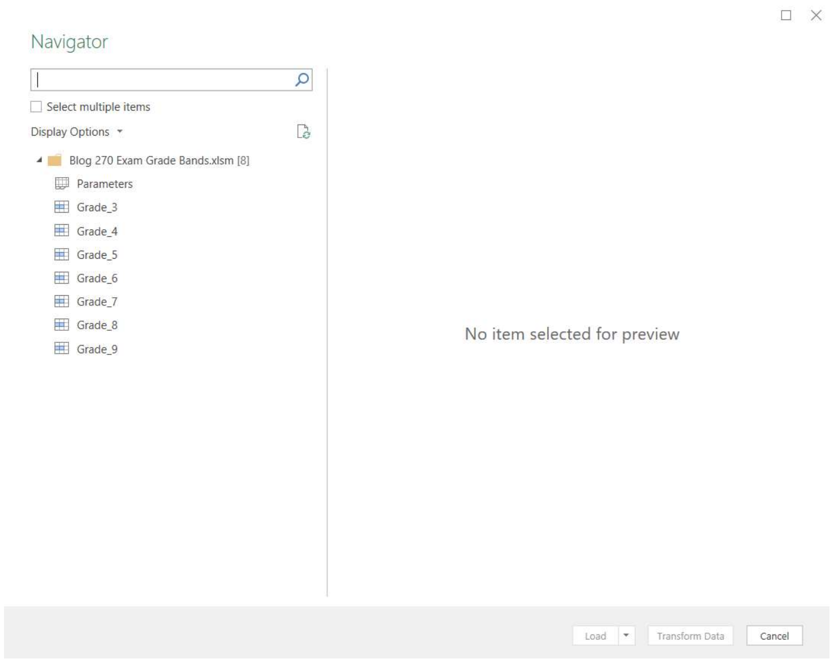


In the dialog, I find the workbook I wish to use:

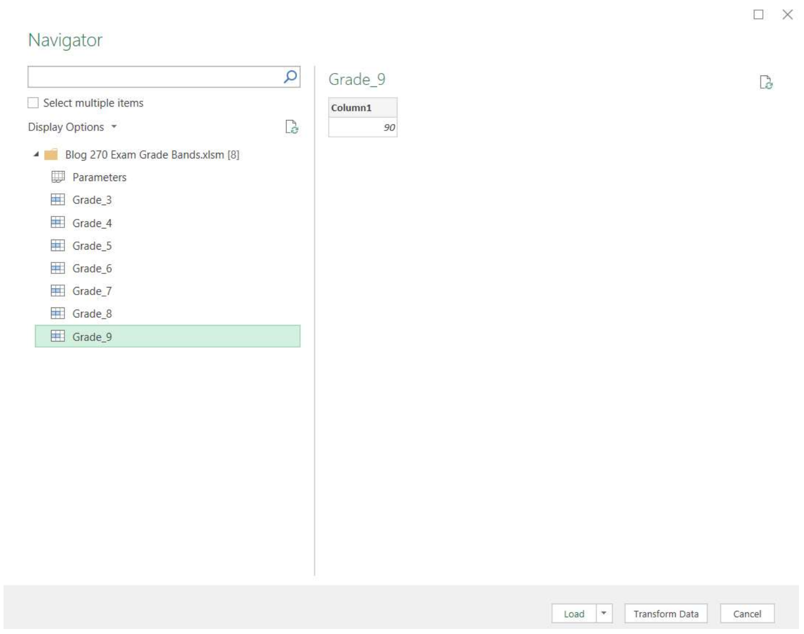

File name: Blog 270 Exam Grade Bands.xlsm

Excel Files (*.xl*;*.xlsx;*.xlsm;*.xls

I can then choose to 'Import'.



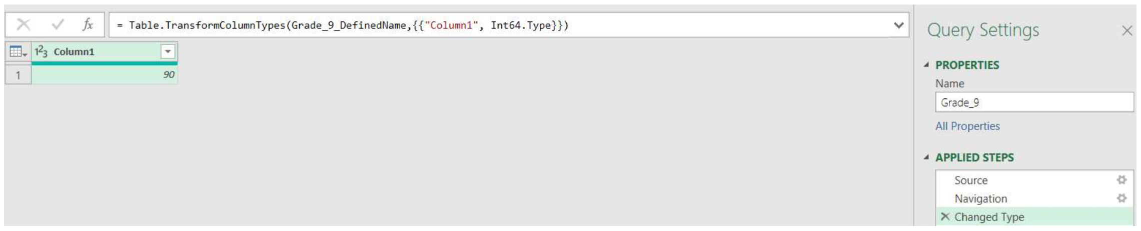
In the Navigator dialog, I can see the Named Cells and the 'Parameters' Sheet. I select **Grade_9**:



Grade_9

Column1
90

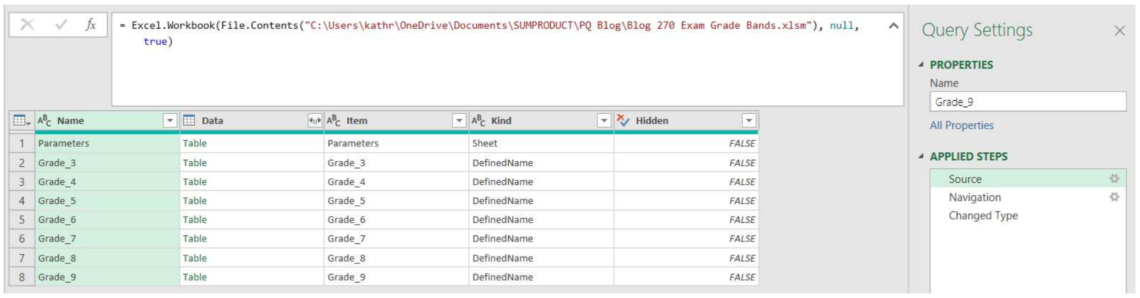
I plan use to this as my base query. I opt to 'Transform Data':



```
= Table.TransformColumnTypes(Grade_9_DefinedName,{{"Column1", Int64.Type}})
```

	Column1
1	90

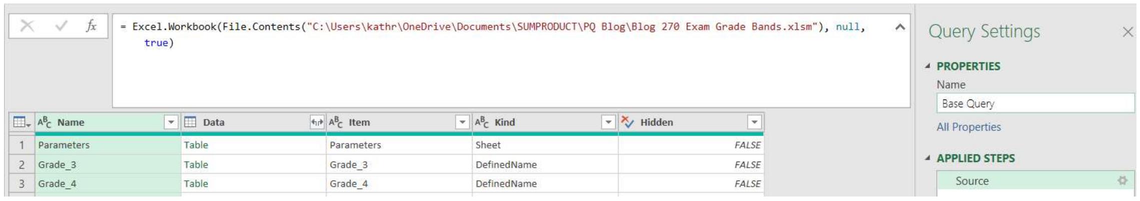
I am interested in the Source step:



```
= Excel.Workbook(File.Contents("C:\Users\kathr\OneDrive\Documents\SUMPRODUCT\PQ Blog\Blog 270 Exam Grade Bands.xlsm"), null,
    true)
```

	Name	Data	Item	Kind	Hidden
1	Parameters	Table	Parameters	Sheet	FALSE
2	Grade_3	Table	Grade_3	DefinedName	FALSE
3	Grade_4	Table	Grade_4	DefinedName	FALSE
4	Grade_5	Table	Grade_5	DefinedName	FALSE
5	Grade_6	Table	Grade_6	DefinedName	FALSE
6	Grade_7	Table	Grade_7	DefinedName	FALSE
7	Grade_8	Table	Grade_8	DefinedName	FALSE
8	Grade_9	Table	Grade_9	DefinedName	FALSE

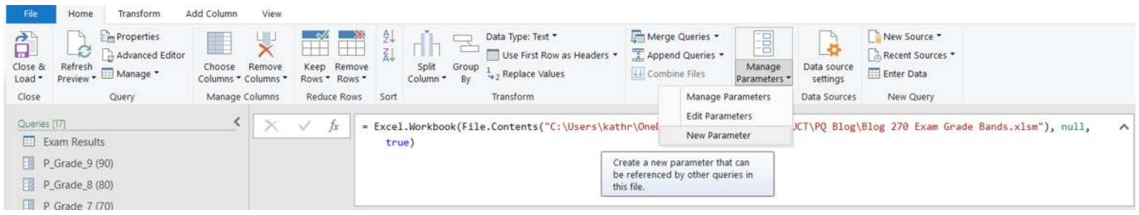
The Source step points at the Excel file I have created. To reduce future maintenance, and to add flexibility I am going to create a base query which only includes the Source step, and then I am going to use a parameter to point to the file location. If I have a base query, then any changes I make to the Source step only need to be made once. I delete the other steps and call the query '**Base Query**':



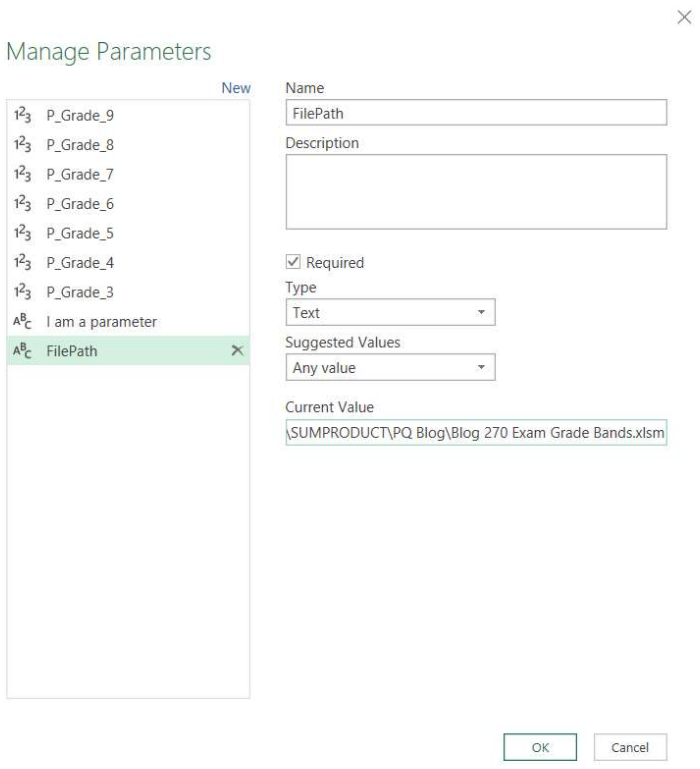
```
= Excel.Workbook(File.Contents("C:\Users\kathr\OneDrive\Documents\SUMPRODUCT\PQ Blog\Blog 270 Exam Grade Bands.xlsm"), null,
    true)
```

	Name	Data	Item	Kind	Hidden
1	Parameters	Table	Parameters	Sheet	FALSE
2	Grade_3	Table	Grade_3	DefinedName	FALSE
3	Grade_4	Table	Grade_4	DefinedName	FALSE

I create a new Parameter from the ‘Manage Parameters’ option on the Home tab:


Manage Parameters
Edit Parameters
New Parameter
Create a new parameter that can be referenced by other queries in this file.

I call the new Parameter **FilePath**:



Name	FilePath
Description	
Required	✓
Type	Text
Suggested Values	Any value
Current Value	\SUMPRODUCT\PQ Blog\Blog 270 Exam Grade Bands.xlsm

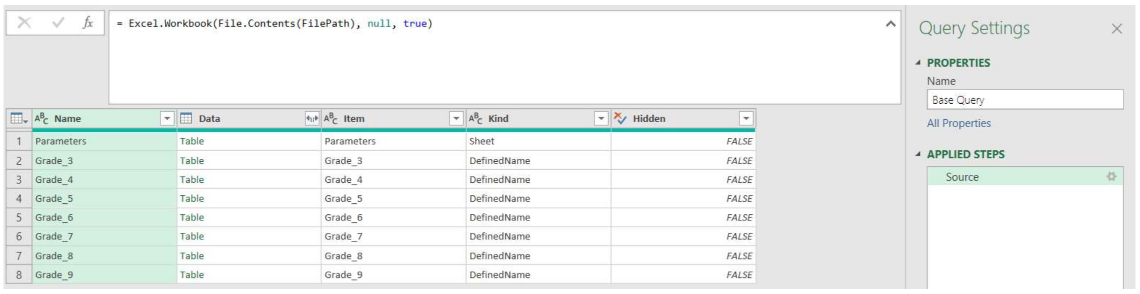
I define the type as ‘Text’ and enter the ‘Current Value’ as the location of the external workbook that contains the grading bands. Having created the **FilePath** parameter, I return to **Base Query**. For me, the **M** code for the Source step is:

```
= Excel.Workbook(File.Contents("C:\Users\kathr\OneDrive\Documents\SUMPRODUCT\PQ Blog\Blog
270 Exam Grade Bands.xlsm"), null, true)
```

I am going to replace the path with the **FilePath** parameter:

`= Excel.Workbook(File.Contents(FilePath), null, true)`

This is easier to read, and now I can change the path by changing the **FilePath** parameter.

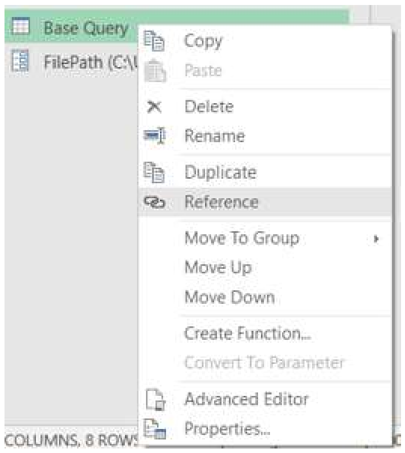


```
= Excel.Workbook(File.Contents(FilePath), null, true)
```

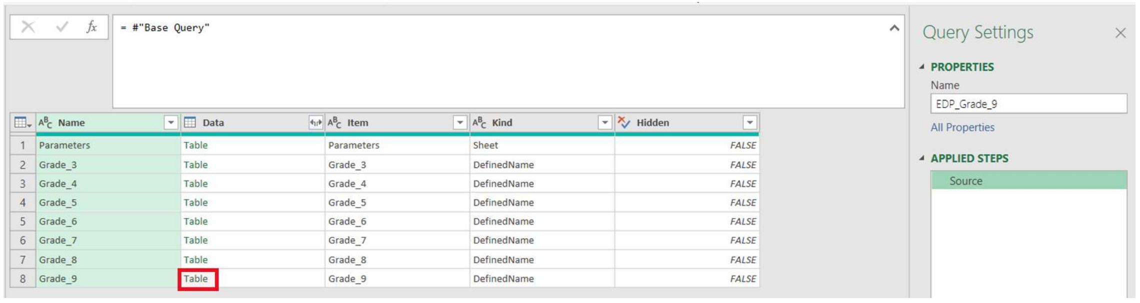
	Name	Data	Item	Kind	Hidden
1	Parameters	Table	Parameters	Sheet	FALSE
2	Grade_3	Table	Grade_3	DefinedName	FALSE
3	Grade_4	Table	Grade_4	DefinedName	FALSE
4	Grade_5	Table	Grade_5	DefinedName	FALSE
5	Grade_6	Table	Grade_6	DefinedName	FALSE
6	Grade_7	Table	Grade_7	DefinedName	FALSE
7	Grade_8	Table	Grade_8	DefinedName	FALSE
8	Grade_9	Table	Grade_9	DefinedName	FALSE

Query Settings
PROPERTIES
Name: Base Query
All Properties
APPLIED STEPS
Source

I can now select **Base Query** in the Queries panel and right-click to create reference queries which will become the new grade band **Parameters**:



I rename the first new query **EDP_Grade_9**. The Source step points to **Base Query**. I can click on the 'Table' text next to **Grade_9** to expand the data for that row:

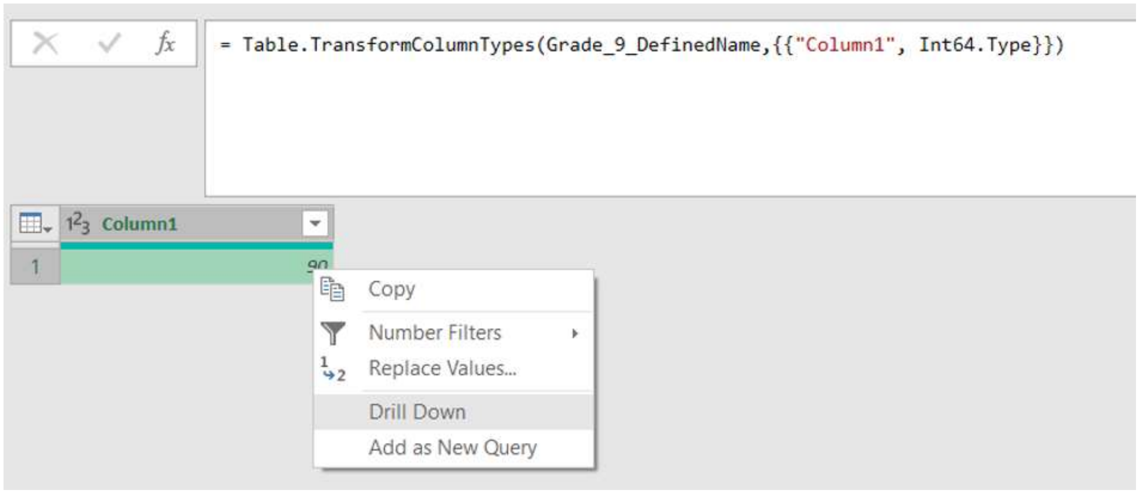


```
= #"Base Query"
```

	Name	Data	Item	Kind	Hidden
1	Parameters	Table	Parameters	Sheet	FALSE
2	Grade_3	Table	Grade_3	DefinedName	FALSE
3	Grade_4	Table	Grade_4	DefinedName	FALSE
4	Grade_5	Table	Grade_5	DefinedName	FALSE
5	Grade_6	Table	Grade_6	DefinedName	FALSE
6	Grade_7	Table	Grade_7	DefinedName	FALSE
7	Grade_8	Table	Grade_8	DefinedName	FALSE
8	Grade_9	Table	Grade_9	DefinedName	FALSE

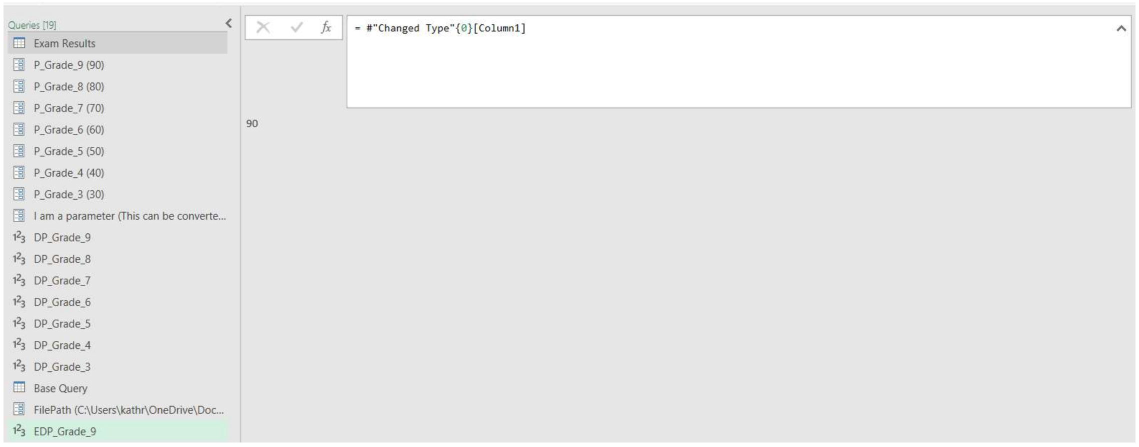
Query Settings
PROPERTIES
Name: EDP_Grade_9
All Properties
APPLIED STEPS
Source

This gives me the data for **Grade_9** from the workbook, and I can right-click and drill down to the value:



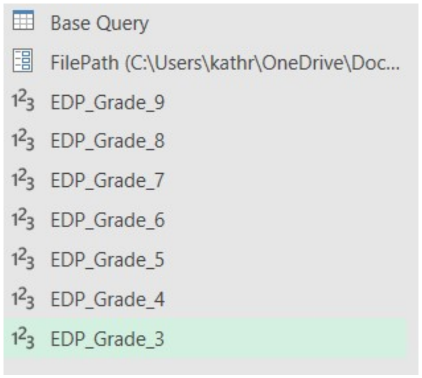
```
= Table.TransformColumnTypes(Grade_9_DefinedName,{{"Column1", Int64.Type}})
```

This gives me the first parameter:

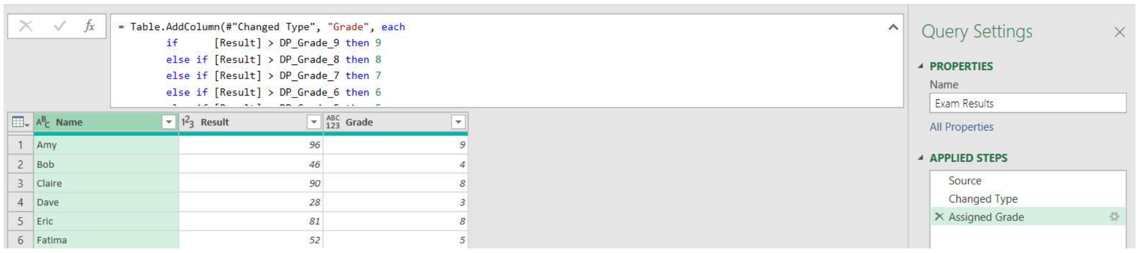


```
= #"Changed Type"{0}[Column1]
```

I can then make more references of **Base Query** and repeat this process to get the other '**EDP_Grade_**' parameters.



I can go back to the **Exam Results** query and use these parameters.



```
= Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > DP_Grade_9 then 9
        else if [Result] > DP_Grade_8 then 8
        else if [Result] > DP_Grade_7 then 7
        else if [Result] > DP_Grade_6 then 6
```

	Name	Result	Grade
1	Amy	96	9
2	Bob	46	4
3	Claire	90	8
4	Dave	28	3
5	Eric	81	8
6	Fatima	52	5

Query Settings

PROPERTIES

Name: Exam Results

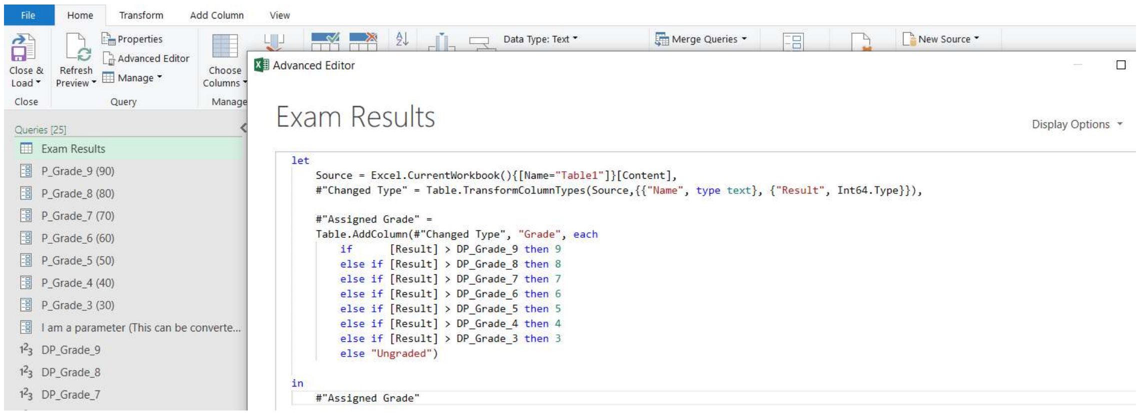
All Properties

APPLIED STEPS

- Source
- Changed Type
- Assigned Grade

I showed earlier that it is not possible to edit using the cog next to the 'Assigned Grade' step to change the parameters, as they are not shown in the dialog unless they are true Power Query parameters. I also looked at the difference between Power Query parameters and other queries that can be used as parameters.

I use the Advanced Editor, available on the Home tab, to change the **M** code:



Queries [25]

- Exam Results
- P_Grade_9 (90)
- P_Grade_8 (80)
- P_Grade_7 (70)
- P_Grade_6 (60)
- P_Grade_5 (50)
- P_Grade_4 (40)
- P_Grade_3 (30)
- I am a parameter (This can be converte...
- DP_Grade_9
- DP_Grade_8
- DP_Grade_7

Advanced Editor

Exam Results

```
let
    Source = Excel.CurrentWorkbook(){[Name="Table1"]}[Content],
    #"Changed Type" = Table.TransformColumnTypes(Source,{{"Name", type text}, {"Result", Int64.Type}}),

    #"Assigned Grade" =
    Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > DP_Grade_9 then 9
        else if [Result] > DP_Grade_8 then 8
        else if [Result] > DP_Grade_7 then 7
        else if [Result] > DP_Grade_6 then 6
        else if [Result] > DP_Grade_5 then 5
        else if [Result] > DP_Grade_4 then 4
        else if [Result] > DP_Grade_3 then 3
        else "Ungraded")

in
    #"Assigned Grade"
```

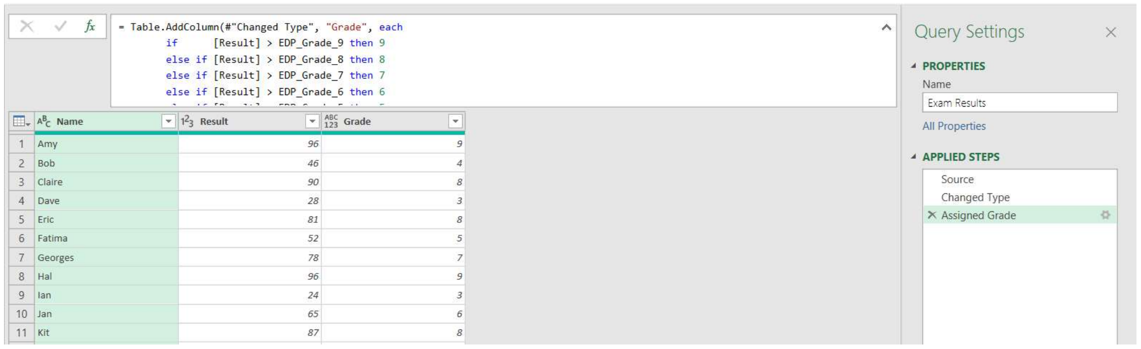
I change the 'Assigned Grade' step from this:

```
#"Assigned Grade" =
    Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > DP_Grade_9 then 9
        else if [Result] > DP_Grade_8 then 8
        else if [Result] > DP_Grade_7 then 7
        else if [Result] > DP_Grade_6 then 6
        else if [Result] > DP_Grade_5 then 5
        else if [Result] > DP_Grade_4 then 4
        else if [Result] > DP_Grade_3 then 3
        else "Ungraded")
```

to this:

```
#"Assigned Grade" =
    Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > EDP_Grade_9 then 9
        else if [Result] > EDP_Grade_8 then 8
        else if [Result] > EDP_Grade_7 then 7
        else if [Result] > EDP_Grade_6 then 6
        else if [Result] > EDP_Grade_5 then 5
        else if [Result] > EDP_Grade_4 then 4
        else if [Result] > EDP_Grade_3 then 3
        else "Ungraded")
```

This has no immediate effect on the results of the query:



```
= Table.AddColumn(#"Changed Type", "Grade", each
        if      [Result] > EDP_Grade_9 then 9
        else if [Result] > EDP_Grade_8 then 8
        else if [Result] > EDP_Grade_7 then 7
        else if [Result] > EDP_Grade_6 then 6
```

	Name	Result	Grade
1	Amy	96	9
2	Bob	46	4
3	Claire	90	8
4	Dave	28	3
5	Eric	81	8
6	Fatima	52	5
7	Georges	78	7
8	Hal	96	9
9	Ian	24	3
10	Jan	65	6
11	Kit	87	8

Query Settings

PROPERTIES

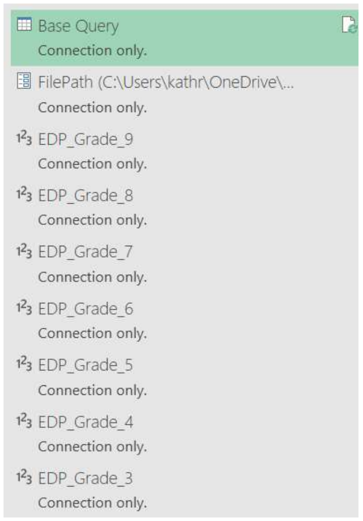
Name: Exam Results

All Properties

APPLIED STEPS

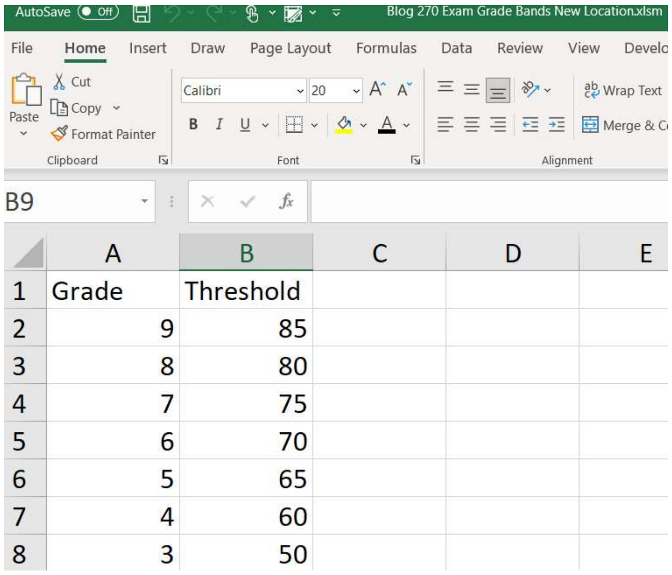
- Source
- Changed Type
- Assigned Grade

To show how **FilePath** allows me to point to another workbook, I have another workbook where I have the same Named Cells for grading bands set up. You should note that I must *close* the Power Query Editor in the workbook containing **Exam Results** before I can edit another workbook, so I 'Close & Load To' and choose 'Connection Only' for the new '**EDP_Grade_'** queries and **Base Query**:



Base Query
Connection only.
FilePath (C:\Users\kathr\OneDrive\...
Connection only.
EDP_Grade_9
Connection only.
EDP_Grade_8
Connection only.
EDP_Grade_7
Connection only.
EDP_Grade_6
Connection only.
EDP_Grade_5
Connection only.
EDP_Grade_4
Connection only.
EDP_Grade_3
Connection only.

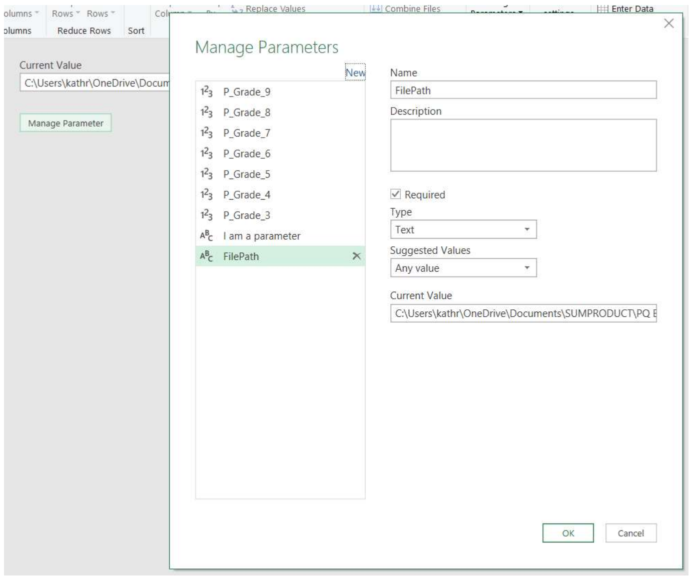
The new workbook I have created has different values for the grading bands:



	A	B	C	D	E
1	Grade	Threshold			
2	9	85			
3	8	80			
4	7	75			
5	6	70			
6	5	65			
7	4	60			
8	3	50			

The workbook also has a different name and is located in a different folder.

I close the new workbook, and go back to the workbook containing the **Exam Results** query, where I select the **FilePath** parameter in the Queries pane, and use the 'Manage Parameter' button to access the dialog:

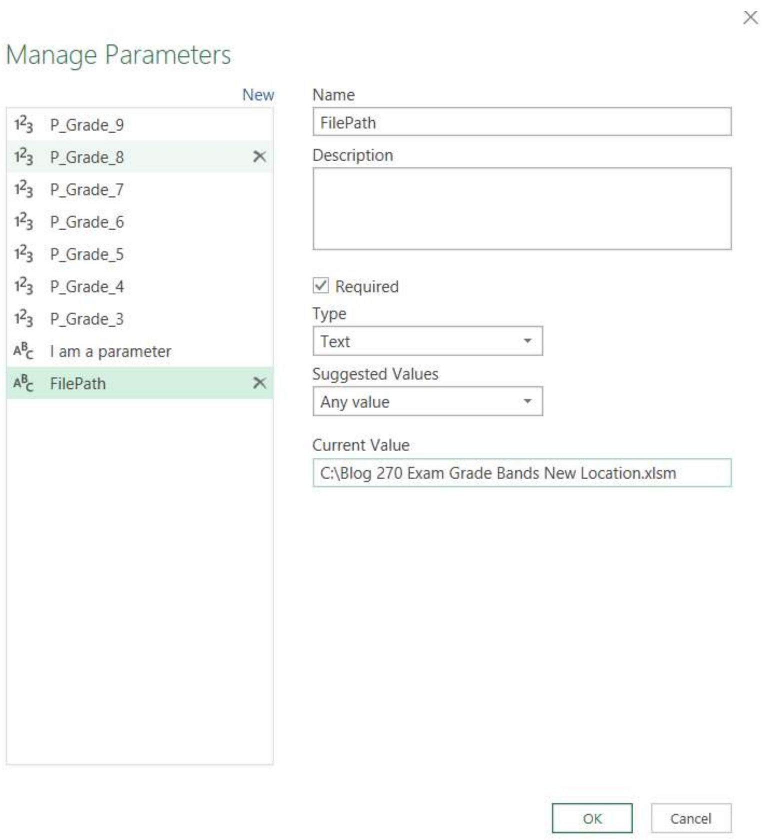


Current Value
C:\Users\kathr\OneDrive\Docum

P_Grade_9
P_Grade_8
P_Grade_7
P_Grade_6
P_Grade_5
P_Grade_4
P_Grade_3
I am a parameter
FilePath

Name: FilePath
Description:
Required ✓
Type: Text
Suggested Values: Any value
Current Value: C:\Users\kathr\OneDrive\Documents\SUMPRODUCT\PQ E

I change the ‘Current Value’ to the new workbook name and location:



Manage Parameters

New

- P_Grade_9
- P_Grade_8
- P_Grade_7
- P_Grade_6
- P_Grade_5
- P_Grade_4
- P_Grade_3
- I am a parameter
- FilePath

Name: FilePath

Description:

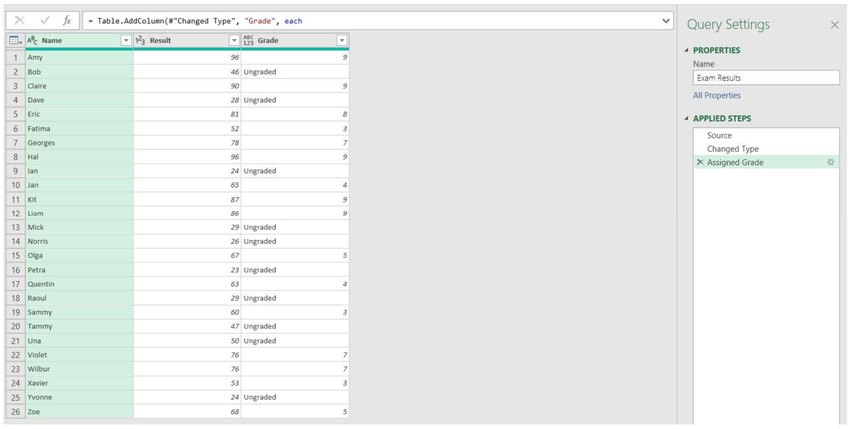
☑ Required

Type: Text

Suggested Values: Any value

Current Value: C:\Blog 270 Exam Grade Bands New Location.xlsm

When I go back to the **Exam Results** query, the results have changed:



```
= Table.AddColumn(#"Changed Type", "Grade", each
```

	Name	Result	Grade
1	Amy	96	9
2	Bob	46	Ungraded
3	Claire	90	9
4	Dave	28	Ungraded
5	Eric	81	8
6	Fatima	52	3
7	Georges	78	7
8	Hal	96	9
9	Ian	24	Ungraded
10	Jan	65	4
11	Kit	87	9
12	Liam	86	9
13	Mick	29	Ungraded
14	Norris	26	Ungraded
15	Olga	67	5
16	Petra	23	Ungraded
17	Quentin	63	4
18	Raoul	29	Ungraded
19	Sammy	60	3
20	Tammy	47	Ungraded
21	Una	50	Ungraded
22	Violet	76	7
23	Wilbur	76	7
24	Xavier	53	3
25	Yvonne	24	Ungraded
26	Zoe	68	5

Query Settings

PROPERTIES

Name: Exam Results

All Properties

APPLIED STEPS

- Source
- Changed Type
- Assigned Grade

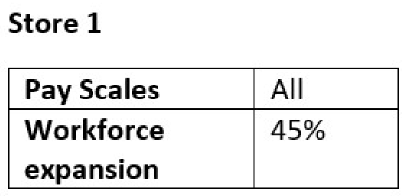
The results have changed, and it’s not looking good for the class!

Note that if users of the workbook containing **Exam Results** needed to maintain the **FilePath** parameter without editing in Power Query, I could link **FilePath** to a Named Excel Cell in the workbook as I have done for the ‘**DP_Grade_**’ and ‘**EDP_Grade_**’ parameters.

Importing from PDF Files

This example will allow me to look at both importing PDF files and splitting columns.

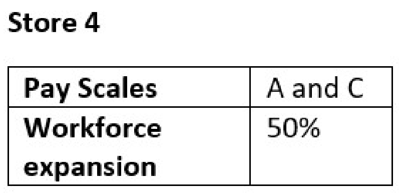
Business is doing well, and the UK division of my company has plans to expand the workforce. I have a PDF file, with data for 10 stores. The information could change for any of those stores, and the way that the information is given is similar to the following:



Store 1

Pay Scales	All
Workforce expansion	45%

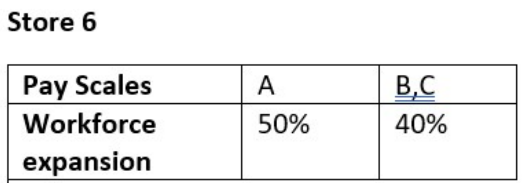
or maybe like this:



Store 4

Pay Scales	A and C
Workforce expansion	50%

or even like this:



Store 6

Pay Scales	A	B,C
Workforce expansion	50%	40%

My goal is to get all this information into one table.

Somewhere in the middle of my data, I have some text, which although very useful to the company, is no help to me.

'There may be plans to build a further store which would require a new workforce, but planning permission has not been granted. Also, we would like to see the Projected Pay increases on the report.'

This could be added to, reduced or removed.

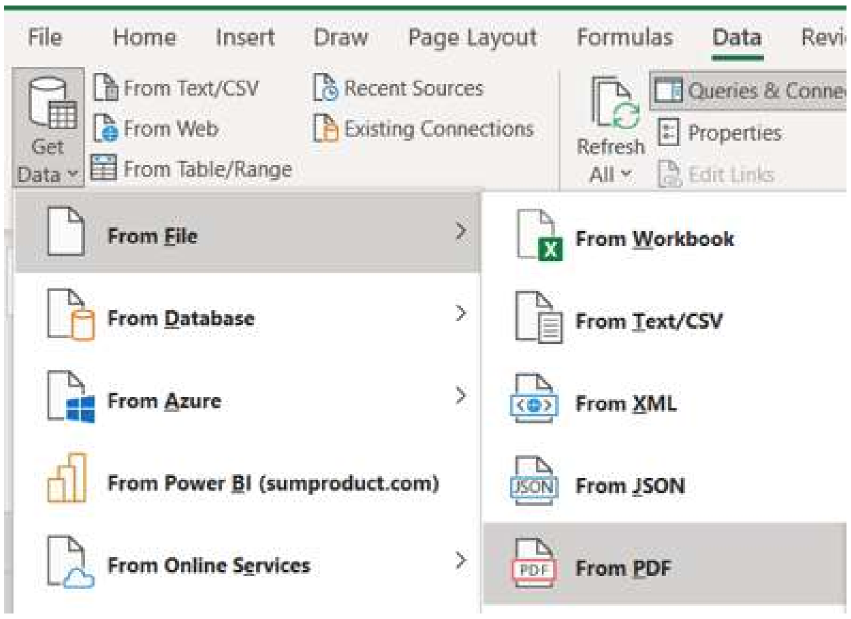
However, there is also another table, which I do wish to extract, although not in its current form (that would be too easy!).

Pay Increases (Proposed)

Proportion of Standard Increase (20%)		
Pay Scale A	£0 – £15,000	1.5
Pay Scale B	£15,0001 - £20,000	1.25
Pay Scale C	£20,0001 and above	1

This means I don't know how many pages are in the PDF. I would normally expect to see data for 10 stores, and they are all split into three [3] pay scales.

I start in an empty Excel Workbook, in the 'Get and Transform' section of the Data tab, where I will use 'Get Data' and then 'From File', where I may choose 'From PDF'.



This allows me to browse to choose the PDF I want to extract. I am then presented with the following dialog:

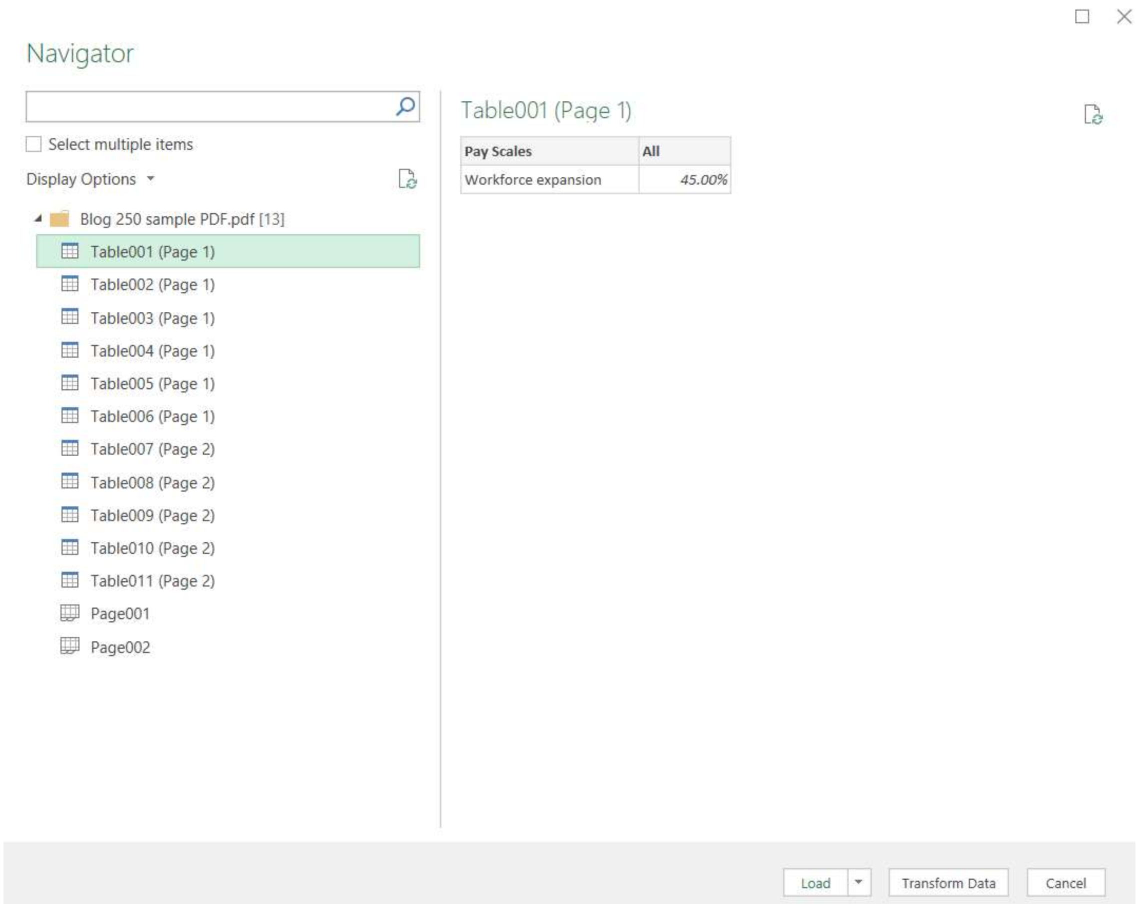


Table001 (Page 1)

Pay Scales	All
Workforce expansion	45.00%

This is my first decision: I need to know whether to pick tables or pages. The best way to decide this is to look at what is in them by selecting one of them for preview.

I skip to the end of the content list, and look at one of the tables:

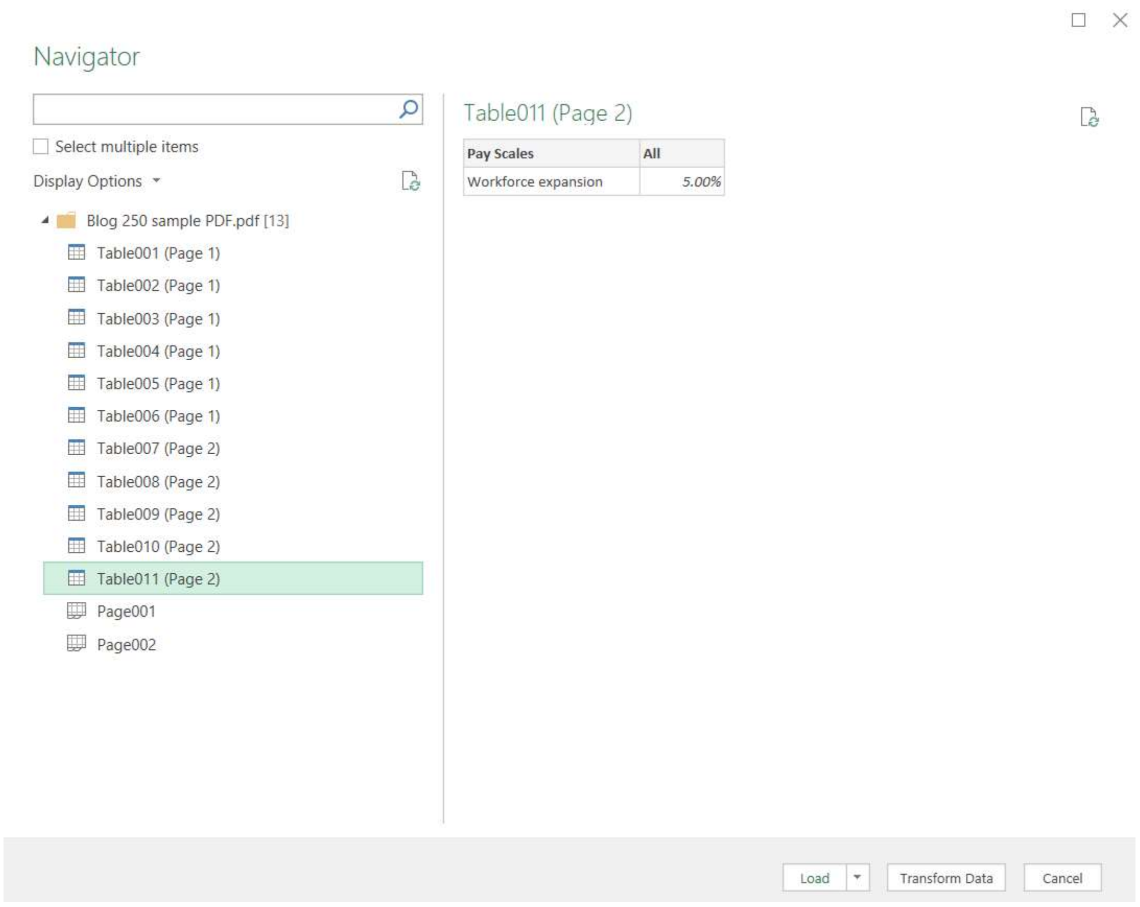
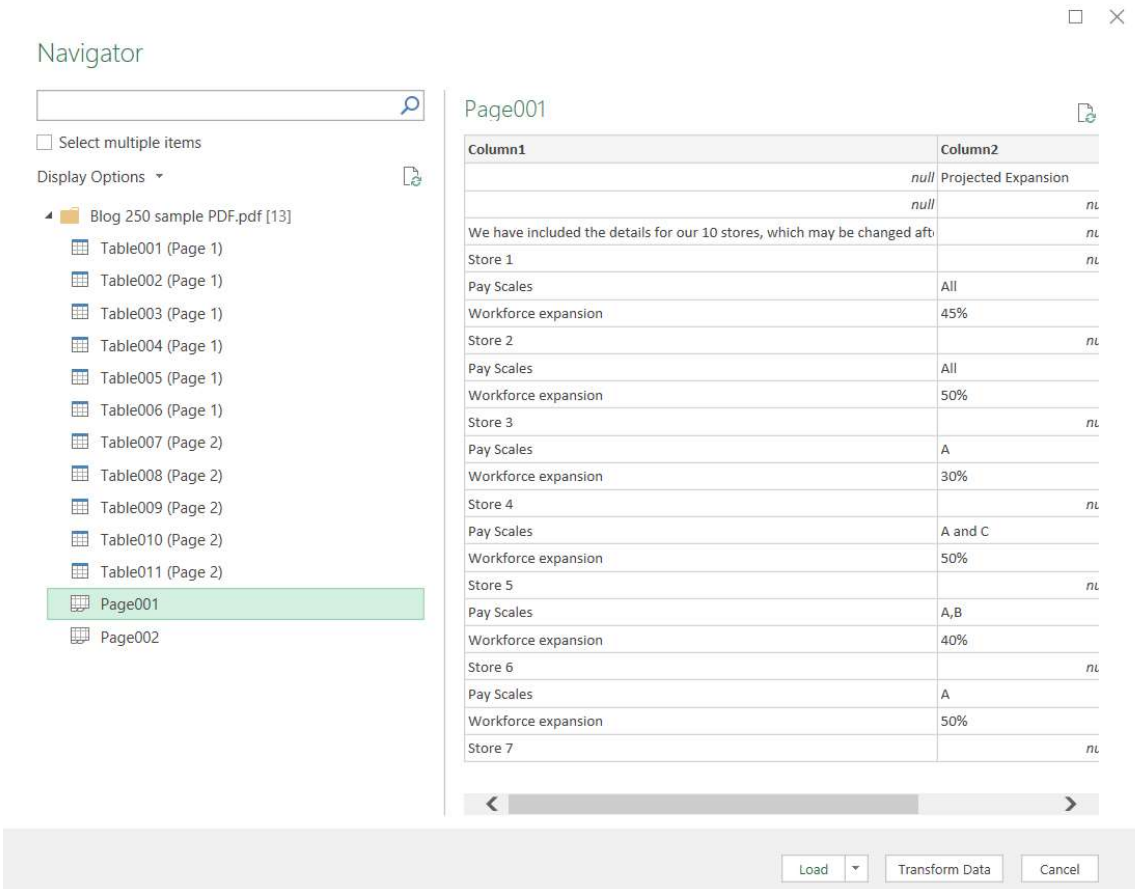


Table011 (Page 2)

Pay Scales	All
Workforce expansion	5.00%

Not much in there, do I really need to get every table? There's not even a way to link this to a store.

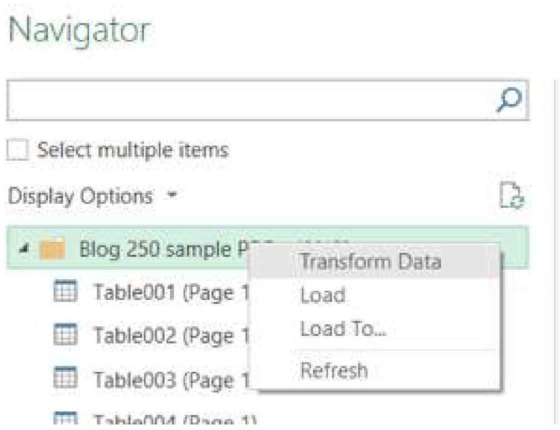
I look at what is in the pages:



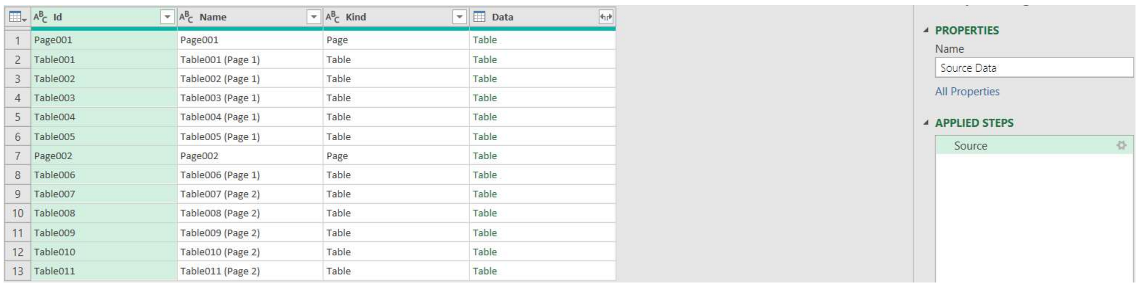
Column1	Column2
null	Projected Expansion
null	nu
We have included the details for our 10 stores, which may be changed aft	nu
Store 1	nu
Pay Scales	All
Workforce expansion	45%
Store 2	nu
Pay Scales	All
Workforce expansion	50%
Store 3	nu
Pay Scales	A
Workforce expansion	30%
Store 4	nu
Pay Scales	A and C
Workforce expansion	50%
Store 5	nu
Pay Scales	A,B
Workforce expansion	40%
Store 6	nu
Pay Scales	A
Workforce expansion	50%
Store 7	nu

Definitely more data, and stores are also included. I can select multiple items, but do I select all the pages? What if there are more next time?

There is an easier way. If I select the folder icon, I have other options…



I can 'Transform Data'. This means all the data in the folder will be loaded into Power Query. This is exactly what I want.



	Id	Name	Kind	Data
1	Page001	Page001	Page	Table
2	Table001	Table001 (Page 1)	Table	Table
3	Table002	Table002 (Page 1)	Table	Table
4	Table003	Table003 (Page 1)	Table	Table
5	Table004	Table004 (Page 1)	Table	Table
6	Table005	Table005 (Page 1)	Table	Table
7	Page002	Page002	Page	Table
8	Table006	Table006 (Page 1)	Table	Table
9	Table007	Table007 (Page 2)	Table	Table
10	Table008	Table008 (Page 2)	Table	Table
11	Table009	Table009 (Page 2)	Table	Table
12	Table010	Table010 (Page 2)	Table	Table
13	Table011	Table011 (Page 2)	Table	Table

PROPERTIES
Name
Source Data
All Properties

APPLIED STEPS
Source

Here it is. All the tables, and all the pages. The **M** code for this is:

= Pdf.Tables(File.Contents("FileLocation\Blog 250 Sample PDF.pdf"), [Implementation="1.3"])

FileLocation is where I have stored the file on my PC, and 'Blog 250 Sample PDF' is the name of the file.

This is using the **M** function **Pdf.Tables():**

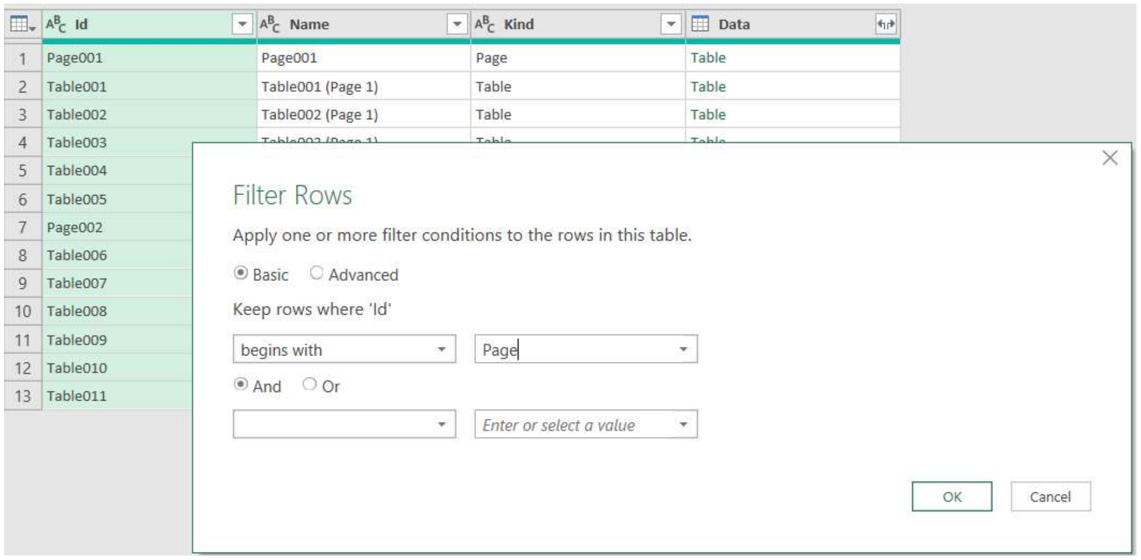
Pdf.Tables(pdf as binary, optional **options** as nullable record**)** as table

This returns any tables found in **pdf**. An optional record parameter, **options**, may be provided to specify additional properties. The record can contain the following fields:

- **Implementation:** the version of the algorithm to use when identifying tables. Old versions are available only for backwards compatibility, to prevent old queries from being broken by algorithm updates. The newest version should always give the best results. Valid values are "1.3", "1.2", "1.1", or *null*
- **StartPage:** specifies the first page in the range of pages to examine; the default value is one [1]
- **EndPage:** specifies the last page in the range of pages to examine. The default value here is the last page of the document
- **MultiPageTables:** controls whether similar tables on consecutive pages will be automatically combined into a single table. Here, the default value is true
- **EnforceBorderLines:** controls whether border lines are always enforced as cell boundaries (when true), or simply used as one hint among many for determining cell boundaries (when false).   The default value here is false.

This explains where [Implementation="1.3"] comes from. It is the algorithm version, and appears to be the latest version, which is reassuring.

So back to my extracted data; I know I want the page data, so I can filter the **Id** column to get everything beginning with 'Page':



	Id	Name	Kind	Data
1	Page001	Page001	Page	Table
2	Table001	Table001 (Page 1)	Table	Table
3	Table002	Table002 (Page 1)	Table	Table
4	Table003			
5	Table004			
6	Table005			
7	Page002			
8	Table006			
9	Table007			
10	Table008			
11	Table009			
12	Table010			
13	Table011			

Filter Rows

Apply one or more filter conditions to the rows in this table.

[✓ Basic] Advanced

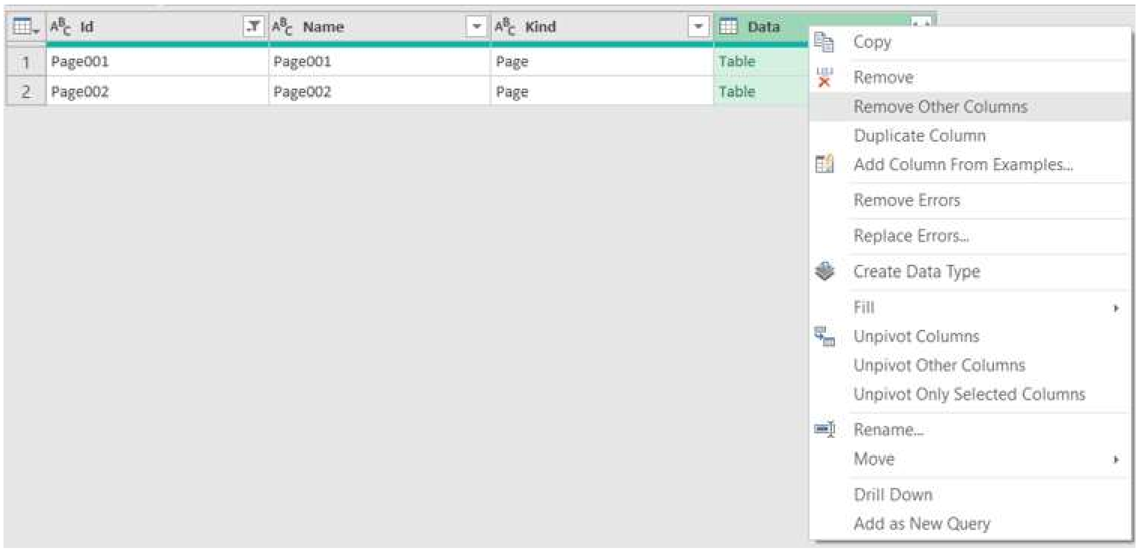
Keep rows where 'Id'

begins with | Page

[✓ And] Or

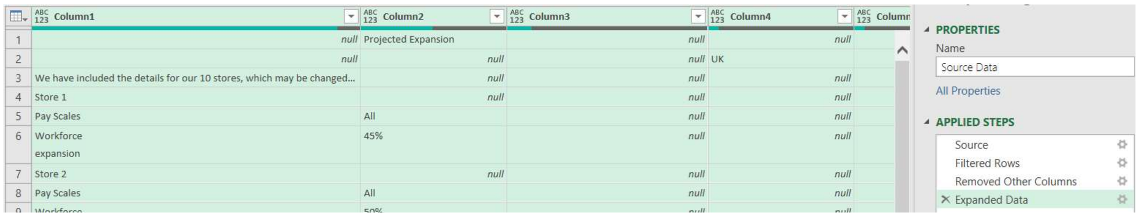
Enter or select a value

This gives me just the pages. I only need the **Data** column now, so I select it and opt to 'Remove Other Columns'.



	Id	Name	Kind	Data
1	Page001	Page001	Page	Table
2	Page002	Page002	Page	Table

I can then expand the data, which extracts to columns 1 to 7:



	Column1	Column2	Column3	Column4
1	null	Projected Expansion	null	null
2	null	null	null	UK
3	We have included the details for our 10 stores, which may be changed...	null	null	null
4	Store 1	null	null	null
5	Pay Scales	All	null	null
6	Workforce expansion	45%	null	null
7	Store 2	null	null	null
8	Pay Scales	All	null	null

PROPERTIES

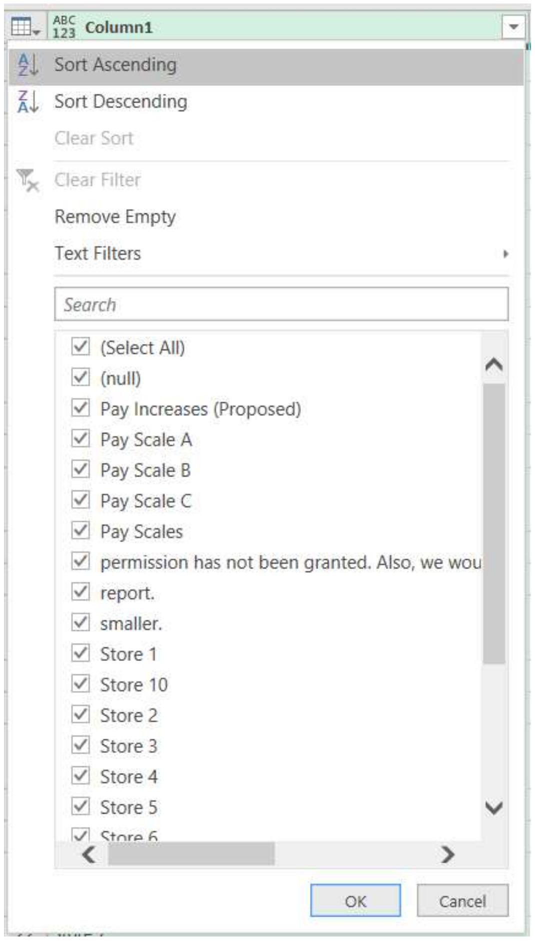
Name: Source Data

All Properties

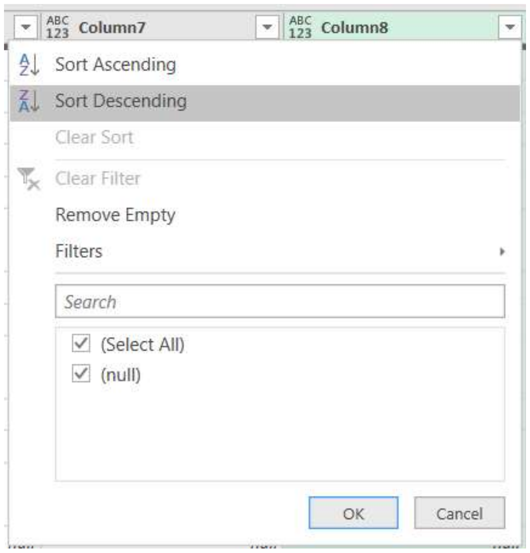
APPLIED STEPS

- Source
- Filtered Rows
- Removed Other Columns
- Expanded Data

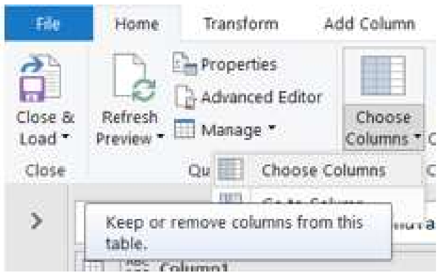
The key to making my transformations as immune to change as possible is to keep the data I need rather than delete the data I don't. Looking at the columns, the easiest way to see if there is any useful data in there is to use the filter icon; **Column1** is clearly very useful.



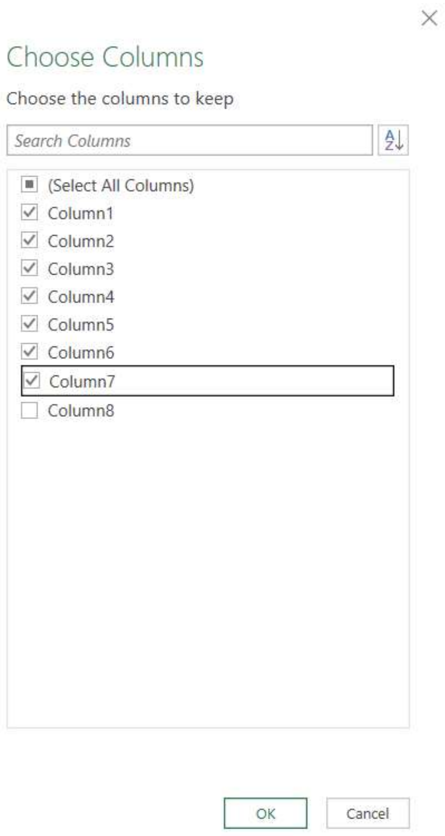
However, **Column8** is not:



However, rather than delete **Column8**, I should keep what I need. On the Home tab, there is an option to 'Choose Columns':



I can use this to specify columns I want to keep. It's much easier than selecting them all for large tables!

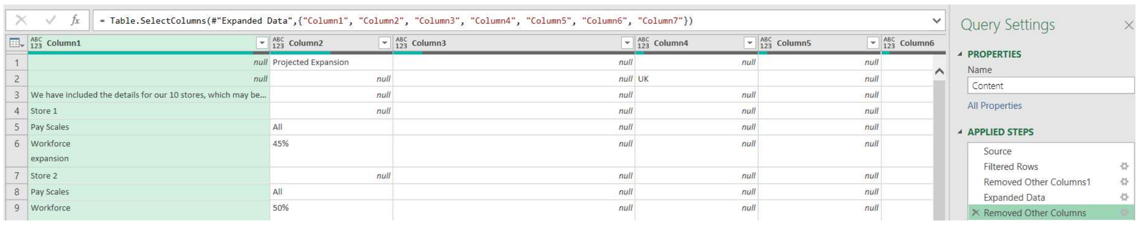


Choose Columns

Choose the columns to keep

- [■] (Select All Columns)
- [✓] Column1
- [✓] Column2
- [✓] Column3
- [✓] Column4
- [✓] Column5
- [✓] Column6
- [✓] Column7
- [ ] Column8

I choose to select the first seven [7] columns.



```
= Table.SelectColumns(#"Expanded Data",{"Column1", "Column2", "Column3", "Column4", "Column5", "Column6", "Column7"})
```

	Column1	Column2	Column3	Column4	Column5	Column6
1	null	Projected Expansion	null	null	null	
2	null	null	null	UK	null	
3	We have included the details for our 10 stores, which may be...	null	null	null	null	
4	Store 1	null	null	null	null	
5	Pay Scales	All	null	null	null	
6	Workforce expansion	45%	null	null	null	
7	Store 2	null	null	null	null	
8	Pay Scales	All	null	null	null	
9	Workforce	50%	null	null	null	

Query Settings

PROPERTIES

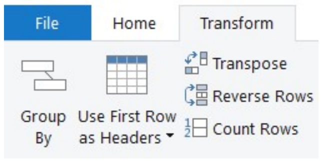
Name: Content

All Properties

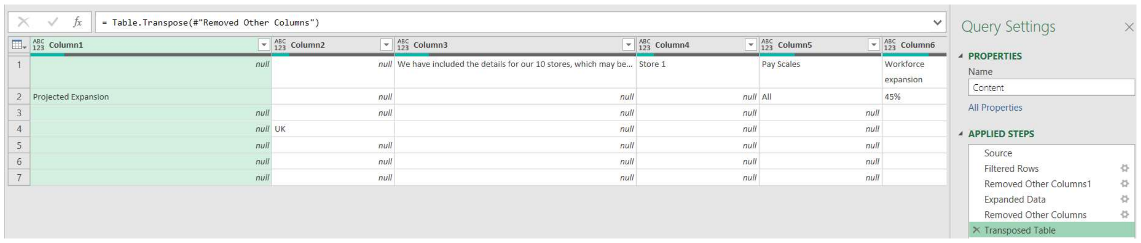
APPLIED STEPS

- Source
- Filtered Rows
- Removed Other Columns1
- Expanded Data
- Removed Other Columns

I can see that the heading data from the tables is in **Column1**, which suggests that transposing my data would be useful. I can do this from the Transform tab.



This swaps the rows and the columns and is much closer to the format I want to see.



```
= Table.Transpose(#"Removed Other Columns")
```

	Column1	Column2	Column3	Column4	Column5	Column6
1	null	null	We have included the details for our 10 stores, which may be...	Store 1	Pay Scales	Workforce expansion
2	Projected Expansion	null	null	null	All	45%
3	null	null	null	null	null	
4	null	UK	null	null	null	
5	null	null	null	null	null	
6	null	null	null	null	null	
7	null	null	null	null	null	

Query Settings

PROPERTIES

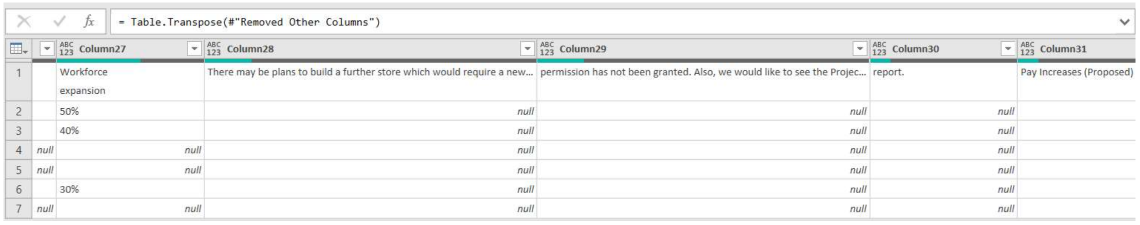
Name: Content

All Properties

APPLIED STEPS

- Source
- Filtered Rows
- Removed Other Columns1
- Expanded Data
- Removed Other Columns
- Transposed Table

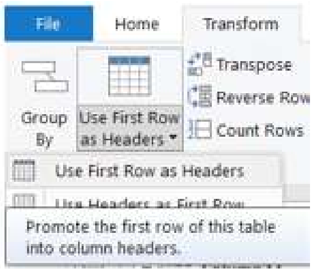
I can check the data in my columns again to see which ones I want to keep. However, it is clear that this time the column names will change with the extra text that is present in my source data.



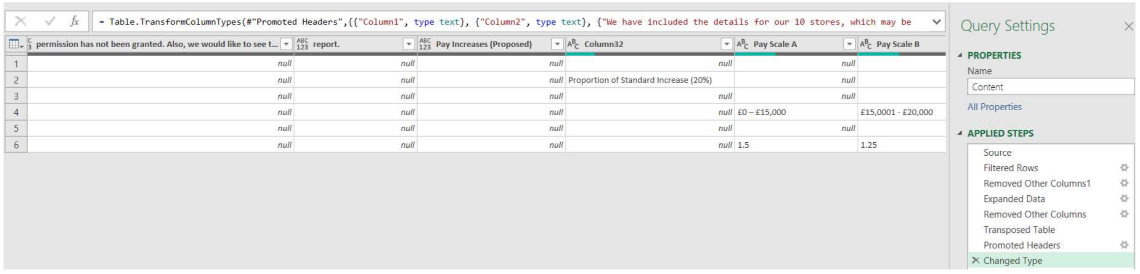
```
= Table.Transpose(#"Removed Other Columns")
```

		Column27	Column28	Column29	Column30	Column31
1		Workforce expansion	There may be plans to build a further store which would require a new...	permission has not been granted. Also, we would like to see the Projec...	report.	Pay Increases (Proposed)
2		50%	null	null	null	
3		40%	null	null	null	
4	null	null	null	null	null	
5	null	null	null	null	null	
6		30%	null	null	null	
7	null	null	null	null	null	

Before I decide which columns to keep, I need some way of identifying them. I will promote the first column to the column headings, which I can do from the Transform Tab.



I choose ‘Use First Row as Headers’:



```
= Table.TransformColumnTypes(#"Promoted Headers",{{"Column1", type text}, {"Column2", type text}, {"We have included the details for our 10 stores, which may be
```

	permission has not been granted. Also, we would like to see t...	report.	Pay Increases (Proposed)	Column32	Pay Scale A	Pay Scale B
1	null	null	null	null	null	
2	null	null	null	Proportion of Standard Increase (20%)	null	
3	null	null	null	null	null	
4	null	null	null	null	£0 – £15,000	£15,0001 - £20,000
5	null	null	null	null	null	
6	null	null	null	null	1.5	1.25

Query Settings

PROPERTIES

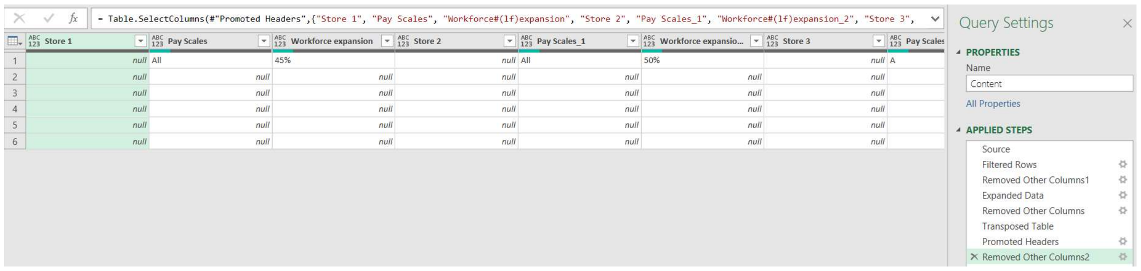
Name: Content

All Properties

APPLIED STEPS

- Source
- Filtered Rows
- Removed Other Columns1
- Expanded Data
- Removed Other Columns
- Transposed Table
- Promoted Headers
- Changed Type

Power Query has created a ‘Changed Type’ step, but this references column names, so I delete it. I can pick the columns I want to keep in the same way as I did earlier.



```
= Table.SelectColumns(#"Promoted Headers",{"Store 1", "Pay Scales", "Workforce#(lf)expansion", "Store 2", "Pay Scales_1", "Workforce#(lf)expansion_2", "Store 3",
```

	Store 1	Pay Scales	Workforce expansion	Store 2	Pay Scales_1	Workforce expansio...	Store 3	Pay Scales
1	null	All	45%	null	All	50%	null	A
2	null	null	null	null	null	null	null	
3	null	null	null	null	null	null	null	
4	null	null	null	null	null	null	null	
5	null	null	null	null	null	null	null	
6	null	null	null	null	null	null	null	

Query Settings

PROPERTIES

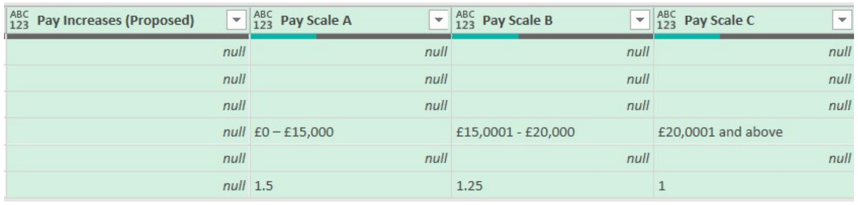
Name: Content

All Properties

APPLIED STEPS

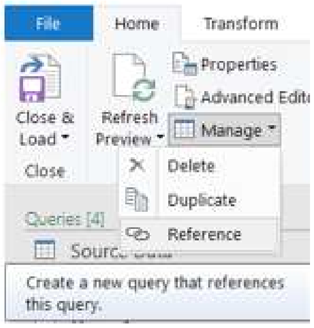
- Source
- Filtered Rows
- Removed Other Columns1
- Expanded Data
- Removed Other Columns
- Transposed Table
- Promoted Headers
- Removed Other Columns2

I have the data I want to keep, but there are two tables in here: the store data and the pay scales.

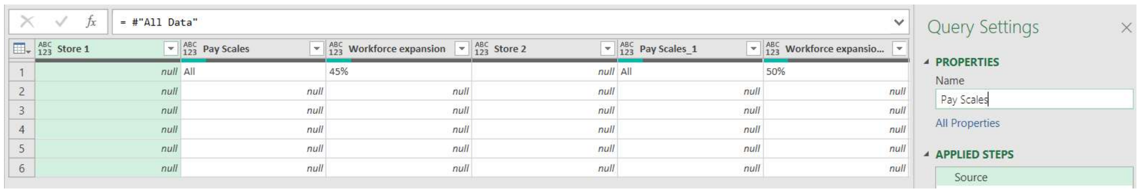


Pay Increases (Proposed)	Pay Scale A	Pay Scale B	Pay Scale C
null	null	null	null
null	null	null	null
null	null	null	null
null	£0 – £15,000	£15,0001 - £20,000	£20,0001 and above
null	null	null	null
null	1.5	1.25	1

I can keep this query, which I will call **All Data**, and make Reference queries: one for the store table and one for the pay scales table.   I can create reference queries from the 'Home' tab.



I call this Reference Query **Pay Scales**.



```
= #"All Data"
```

	Store 1	Pay Scales	Workforce expansion	Store 2	Pay Scales_1	Workforce expansio...
1	null	All	45%	null	All	50%
2	null	null	null	null	null	null
3	null	null	null	null	null	null
4	null	null	null	null	null	null
5	null	null	null	null	null	null
6	null	null	null	null	null	null

Query Settings

PROPERTIES

Name: Pay Scales

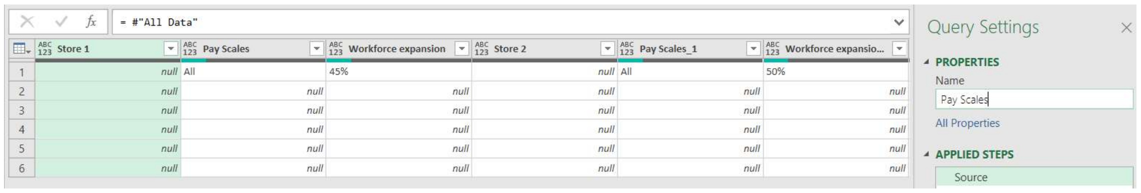
All Properties

APPLIED STEPS

- Source

I also create another Reference Query, **Stores.**

I start with **Pay Scales.**



```
= #"All Data"
```

	Store 1	Pay Scales	Workforce expansion	Store 2	Pay Scales_1	Workforce expansio...
1	null	All	45%	null	All	50%
2	null	null	null	null	null	null
3	null	null	null	null	null	null
4	null	null	null	null	null	null
5	null	null	null	null	null	null
6	null	null	null	null	null	null

Query Settings

PROPERTIES

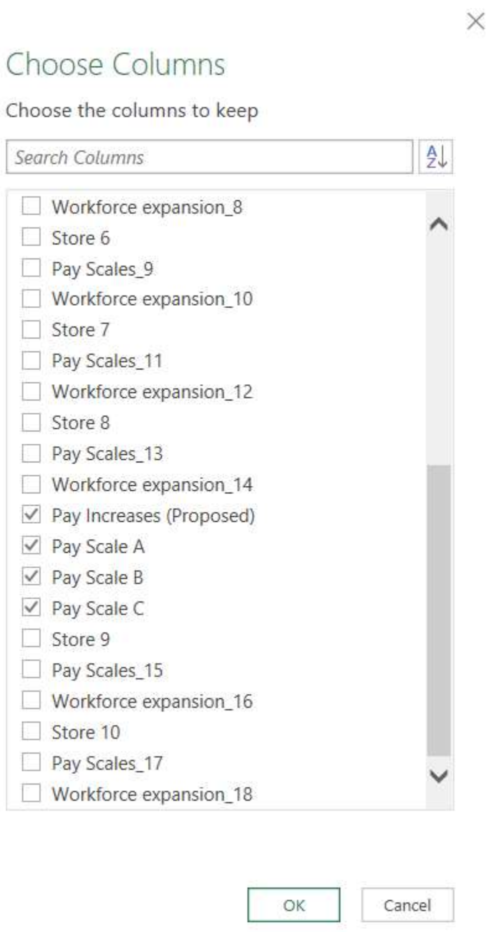
Name: Pay Scales

All Properties

APPLIED STEPS

- Source

I start by keeping the columns I will be needing for this table.

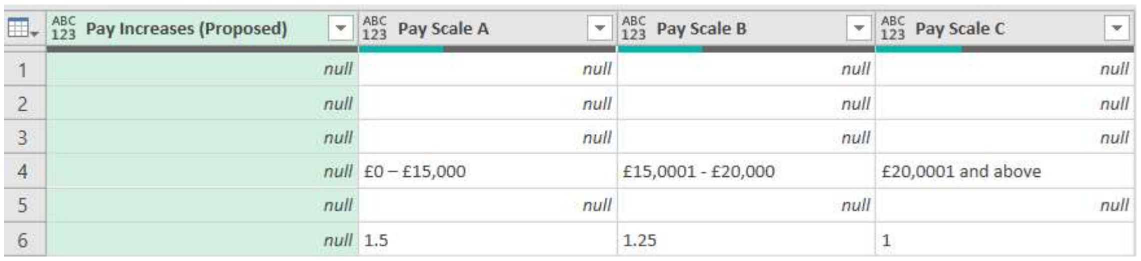

Choose Columns

Choose the columns to keep

Search Columns

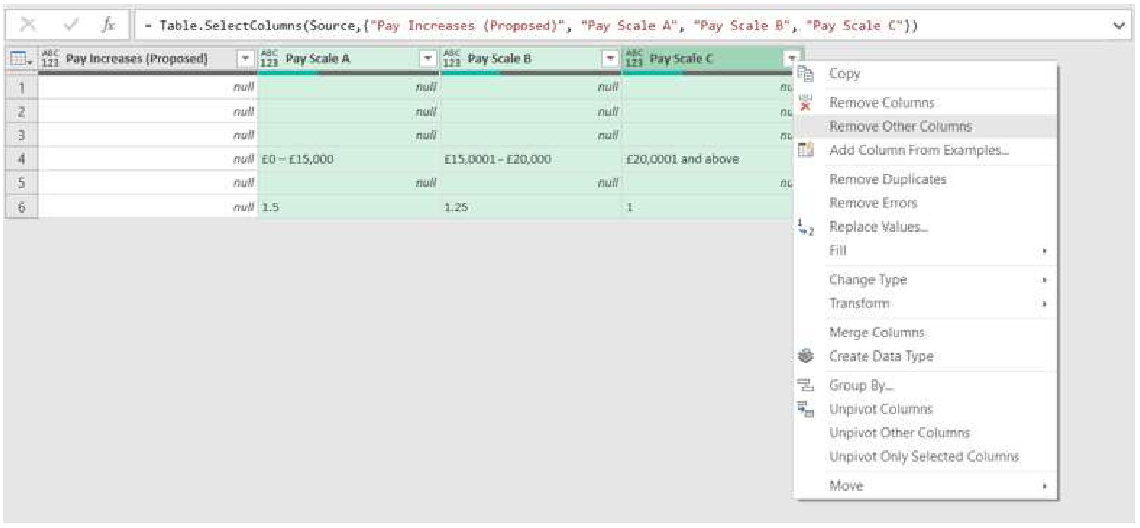
- [ ] Workforce expansion_8
- [ ] Store 6
- [ ] Pay Scales_9
- [ ] Workforce expansion_10
- [ ] Store 7
- [ ] Pay Scales_11
- [ ] Workforce expansion_12
- [ ] Store 8
- [ ] Pay Scales_13
- [ ] Workforce expansion_14
- [✓] Pay Increases (Proposed)
- [✓] Pay Scale A
- [✓] Pay Scale B
- [✓] Pay Scale C
- [ ] Store 9
- [ ] Pay Scales_15
- [ ] Workforce expansion_16
- [ ] Store 10
- [ ] Pay Scales_17
- [ ] Workforce expansion_18

This means I can concentrate on the data I need to transform for this table.



	Pay Increases (Proposed)	Pay Scale A	Pay Scale B	Pay Scale C
1	null	null	null	null
2	null	null	null	null
3	null	null	null	null
4	null	£0 – £15,000	£15,0001 - £20,000	£20,0001 and above
5	null	null	null	null
6	null	1.5	1.25	1

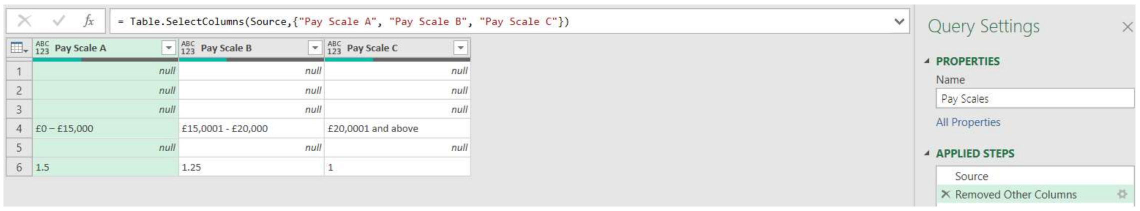
Having checked the data, I actually only need the pay scale columns, so I select them whilst holding down the **CRTL** key and click on 'Remove Other Columns'.



```
= Table.SelectColumns(Source,{"Pay Increases (Proposed)", "Pay Scale A", "Pay Scale B", "Pay Scale C"})
```

	Pay Increases (Proposed)	Pay Scale A	Pay Scale B	Pay Scale C
1	null	null	null	null
2	null	null	null	null
3	null	null	null	null
4	null	£0 – £15,000	£15,0001 - £20,000	£20,0001 and above
5	null	null	null	null
6	null	1.5	1.25	1

Power Query incorporates this into the existing 'Remove Other Columns' step.



```
= Table.SelectColumns(Source,{"Pay Scale A", "Pay Scale B", "Pay Scale C"})
```

	Pay Scale A	Pay Scale B	Pay Scale C
1	null	null	null
2	null	null	null
3	null	null	null
4	£0 – £15,000	£15,0001 - £20,000	£20,0001 and above
5	null	null	null
6	1.5	1.25	1

Query Settings

PROPERTIES

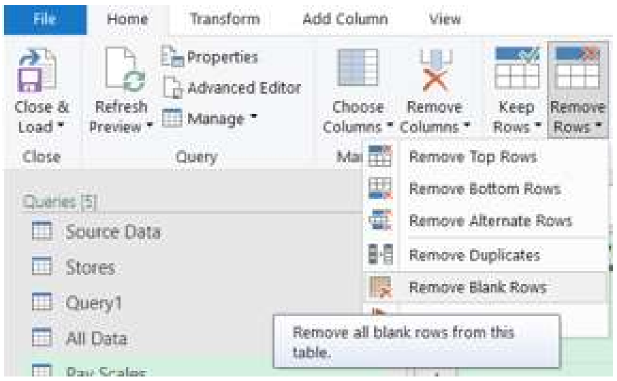
Name: Pay Scales

All Properties

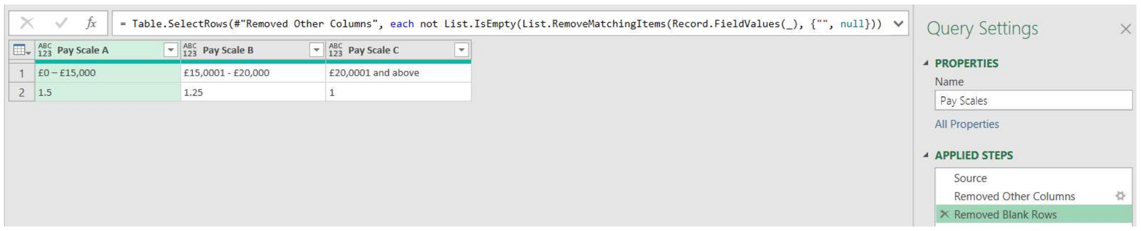
APPLIED STEPS

- Source
- Removed Other Columns

I can remove empty rows from the Home tab.



I can now look at how to transform my data from this into a useful table.



```
= Table.SelectRows(#"Removed Other Columns", each not List.IsEmpty(List.RemoveMatchingItems(Record.FieldValues(_), {"", null})))
```

	Pay Scale A	Pay Scale B	Pay Scale C
1	£0 – £15,000	£15,0001 - £20,000	£20,0001 and above
2	1.5	1.25	1

Query Settings

PROPERTIES

Name: Pay Scales

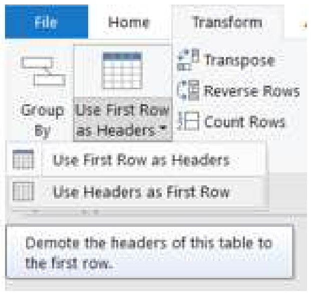
All Properties

APPLIED STEPS

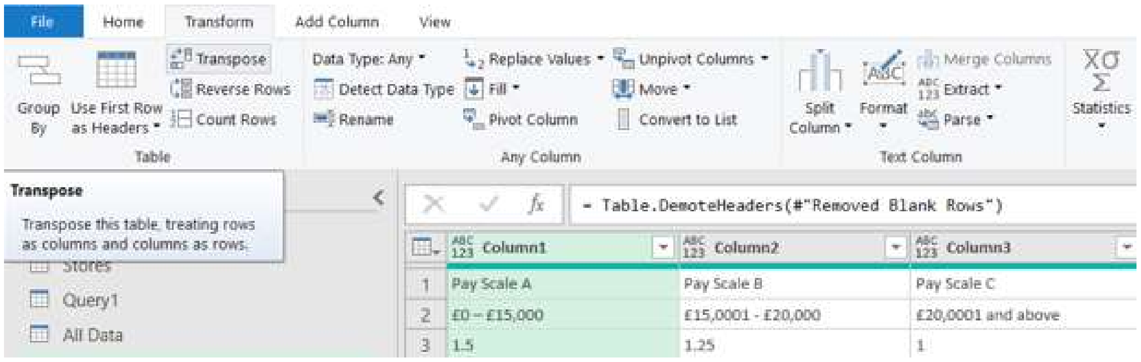
- Source
- Removed Other Columns
- Removed Blank Rows

Transposing Data

I want to transpose the data, but if I do this with some of the data I need in the column headings, I will lose it. First, I need to demote the column headings so that I have the information in a row. I can do this from the Home tab.



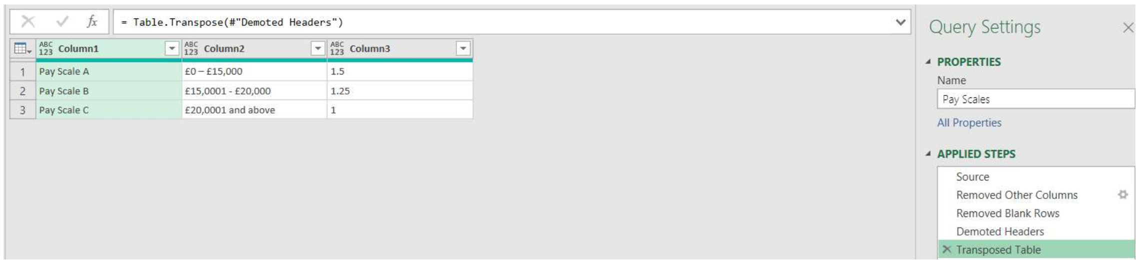
This creates a 'Change Type' step which I delete as I am not ready to decide column types yet. I am now ready to transpose my data, using the option on the Transform tab.



```
= Table.DemoteHeaders(#"Removed Blank Rows")
```

	Column1	Column2	Column3
1	Pay Scale A	Pay Scale B	Pay Scale C
2	£0 – £15,000	£15,0001 - £20,000	£20,0001 and above
3	1.5	1.25	1

My data is starting to take shape.



```
= Table.Transpose(#"Demoted Headers")
```

	Column1	Column2	Column3
1	Pay Scale A	£0 – £15,000	1.5
2	Pay Scale B	£15,0001 - £20,000	1.25
3	Pay Scale C	£20,0001 and above	1

Query Settings

PROPERTIES

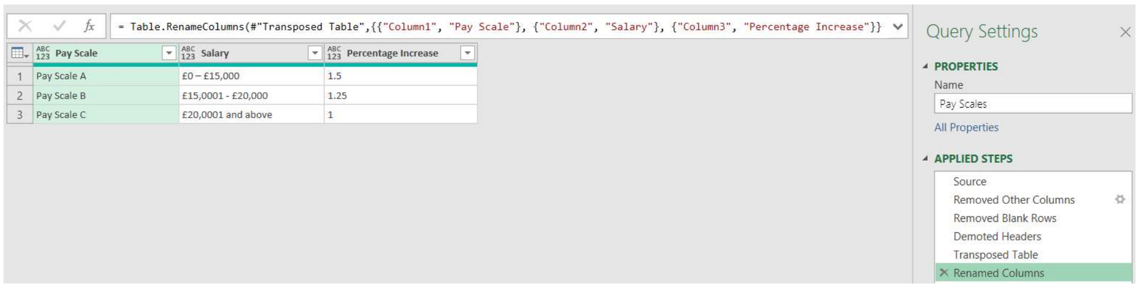
Name: Pay Scales

All Properties

APPLIED STEPS

- Source
- Removed Other Columns
- Removed Blank Rows
- Demoted Headers
- Transposed Table

I can rename the headings.



```
= Table.RenameColumns(#"Transposed Table",{{"Column1", "Pay Scale"}, {"Column2", "Salary"}, {"Column3", "Percentage Increase"}})
```

	Pay Scale	Salary	Percentage Increase
1	Pay Scale A	£0 – £15,000	1.5
2	Pay Scale B	£15,0001 - £20,000	1.25
3	Pay Scale C	£20,0001 and above	1

Query Settings

PROPERTIES

Name: Pay Scales

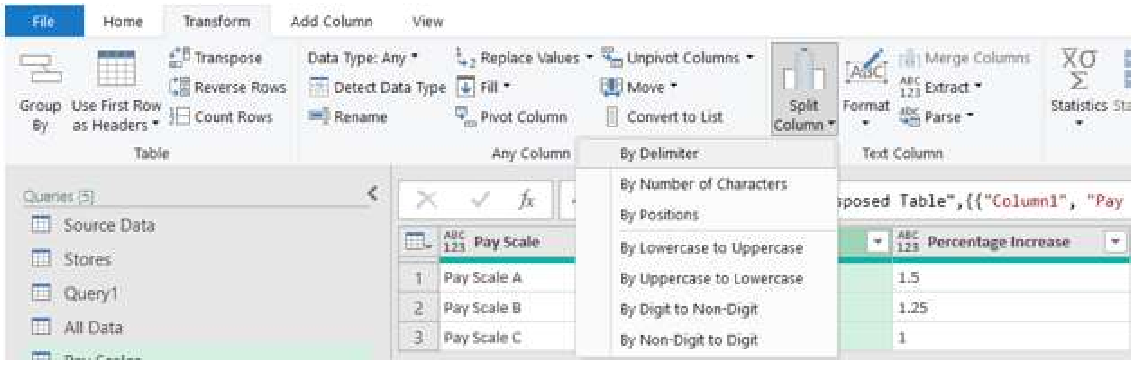
All Properties

APPLIED STEPS

- Source
- Removed Other Columns
- Removed Blank Rows
- Demoted Headers
- Transposed Table
- Renamed Columns

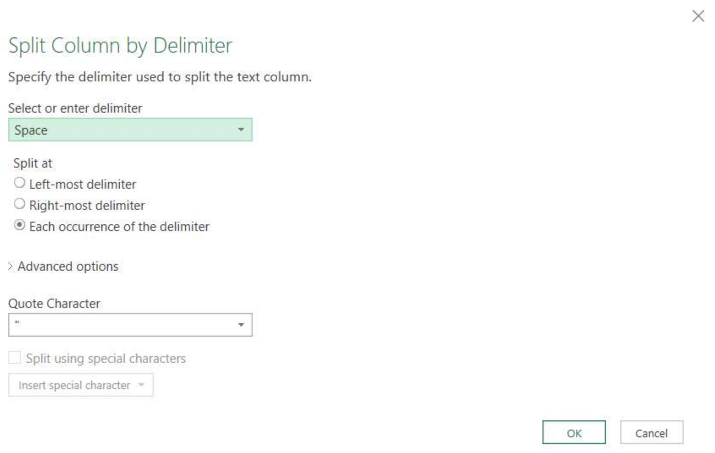
Splitting Columns

I want to show a start and end salary, rather than have the information in one column. I can split the Salary column from the Transform tab.



	Pay Scale	Percentage Increase
1	Pay Scale A	1.5
2	Pay Scale B	1.25
3	Pay Scale C	1

From the dropdown, I choose to split 'By Delimiter'; this brings up a dialog.



Split Column by Delimiter

Specify the delimiter used to split the text column.

Select or enter delimiter: Space

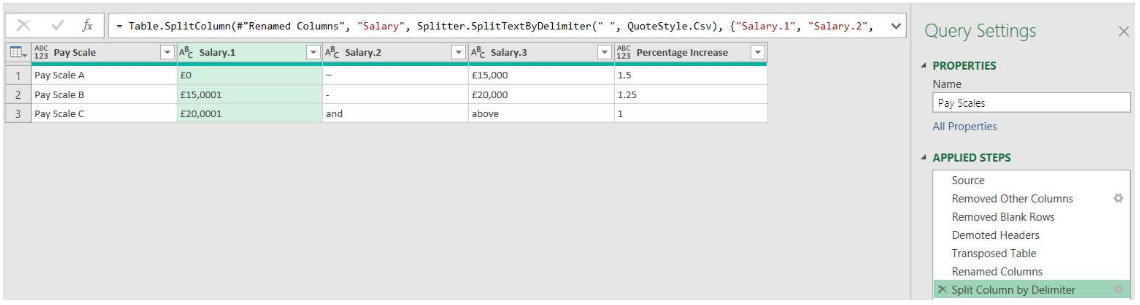
Split at
- ○ Left-most delimiter
- ○ Right-most delimiter
- ● Each occurrence of the delimiter

› Advanced options

Quote Character: "

☐ Split using special characters

I choose to split by space at each occurrence of a space, because this will give me a column with the lower and upper limit.



```
= Table.SplitColumn(#"Renamed Columns", "Salary", Splitter.SplitTextByDelimiter(" ", QuoteStyle.Csv), {"Salary.1", "Salary.2",
```

	Pay Scale	Salary.1	Salary.2	Salary.3	Percentage Increase
1	Pay Scale A	£0	–	£15,000	1.5
2	Pay Scale B	£15,0001	-	£20,000	1.25
3	Pay Scale C	£20,0001	and	above	1

Query Settings

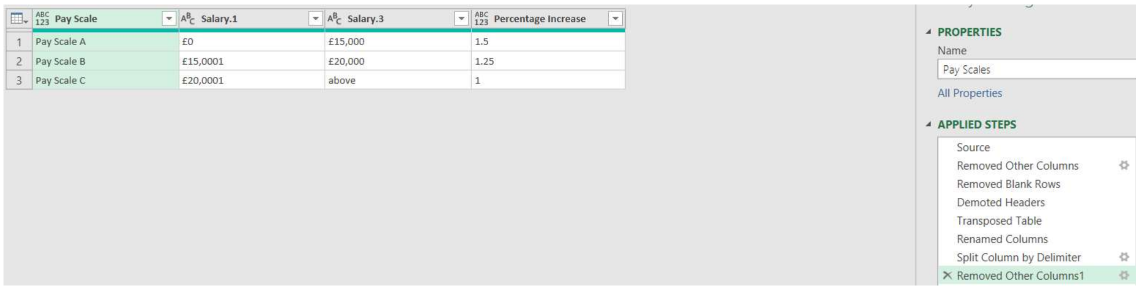
PROPERTIES

Name: Pay Scales

All Properties

APPLIED STEPS
- Source
- Removed Other Columns
- Removed Blank Rows
- Demoted Headers
- Transposed Table
- Renamed Columns
- Split Column by Delimiter

I delete the automated 'Changed Type' step again. I don't need Salary.2, so I can remove this. As usual, I do this by selecting the columns I want to keep and 'Remove Other Columns'.

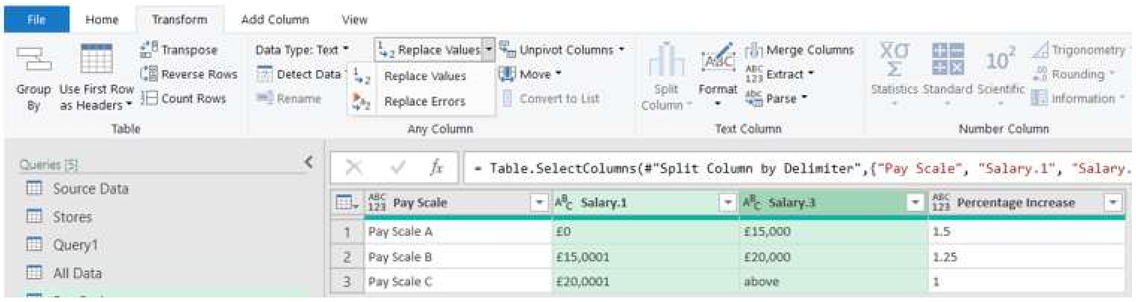


	Pay Scale	Salary.1	Salary.3	Percentage Increase
1	Pay Scale A	£0	£15,000	1.5
2	Pay Scale B	£15,0001	£20,000	1.25
3	Pay Scale C	£20,0001	above	1

PROPERTIES
Name
Pay Scales
All Properties

APPLIED STEPS
Source
Removed Other Columns
Removed Blank Rows
Demoted Headers
Transposed Table
Renamed Columns
Split Column by Delimiter
Removed Other Columns1

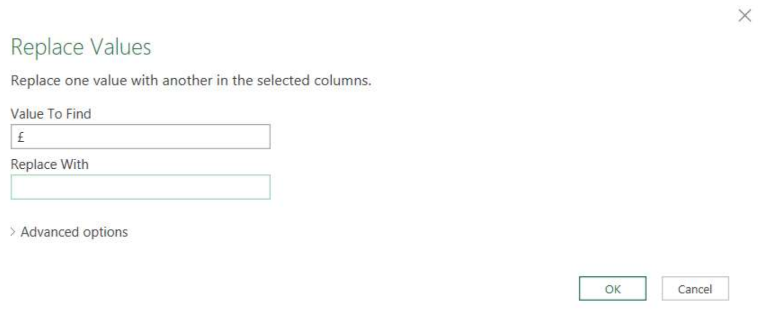
I want Salary.1 and Salary.2 to be numeric columns, so I need to remove the £ signs. I can do this by selecting the columns and replacing £ with blank. I start by using 'Replace Values' on the Transform tab.



= Table.SelectColumns(#"Split Column by Delimiter",{"Pay Scale", "Salary.1", "Salary.

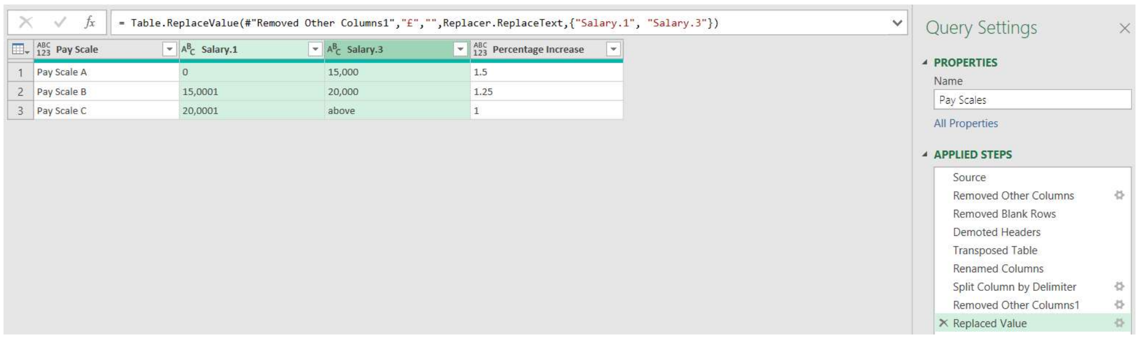
	Pay Scale	Salary.1	Salary.3	Percentage Increase
1	Pay Scale A	£0	£15,000	1.5
2	Pay Scale B	£15,0001	£20,000	1.25
3	Pay Scale C	£20,0001	above	1

This provides a dialog where I can enter the details.



Replace Values
Replace one value with another in the selected columns.
Value To Find
£
Replace With

This will remove the £ signs.



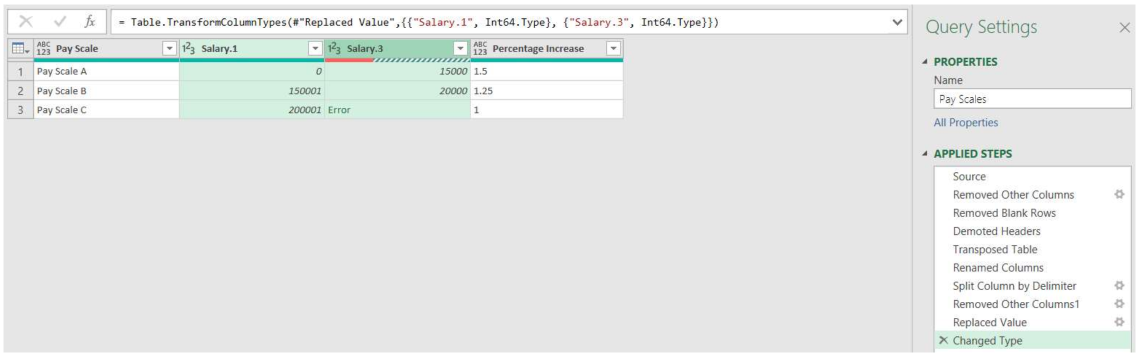
= Table.ReplaceValue(#"Removed Other Columns1","£","",Replacer.ReplaceText,{"Salary.1", "Salary.3"})

	Pay Scale	Salary.1	Salary.3	Percentage Increase
1	Pay Scale A	0	15,000	1.5
2	Pay Scale B	15,0001	20,000	1.25
3	Pay Scale C	20,0001	above	1

Query Settings
PROPERTIES
Name
Pay Scales
All Properties

APPLIED STEPS
Source
Removed Other Columns
Removed Blank Rows
Demoted Headers
Transposed Table
Renamed Columns
Split Column by Delimiter
Removed Other Columns1
Replaced Value

Next, I change both columns to whole numbers. I can do this from the Home tab or the Transform tab, or by using the right-click menu, and changing the data type.

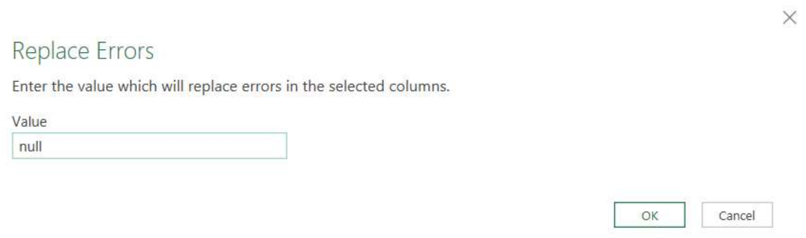


```
= Table.TransformColumnTypes(#"Replaced Value",{{"Salary.1", Int64.Type}, {"Salary.3", Int64.Type}})
```

	Pay Scale	Salary.1	Salary.3	Percentage Increase
1	Pay Scale A	0	15000	1.5
2	Pay Scale B	150001	20000	1.25
3	Pay Scale C	200001	Error	1

Query Settings
PROPERTIES
Name: Pay Scales
APPLIED STEPS: Source, Removed Other Columns, Removed Blank Rows, Demoted Headers, Transposed Table, Renamed Columns, Split Column by Delimiter, Removed Other Columns1, Replaced Value, Changed Type

I get an error, but I can use ‘Replace Values’ again; this time, I choose ‘Replace Errors’.



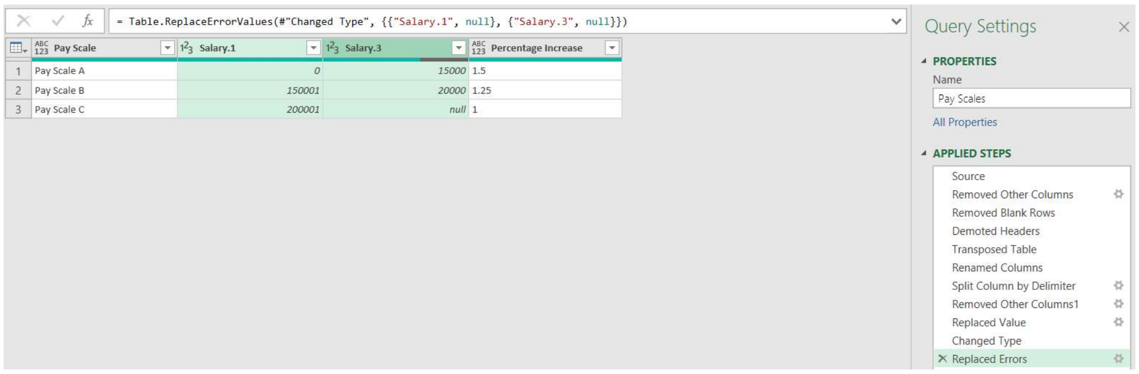
Replace Errors

Enter the value which will replace errors in the selected columns.

Value: null

OK Cancel

I want to replace it with *null*, not zero [0], since zero is the starting point for ‘Pay Scale A’.

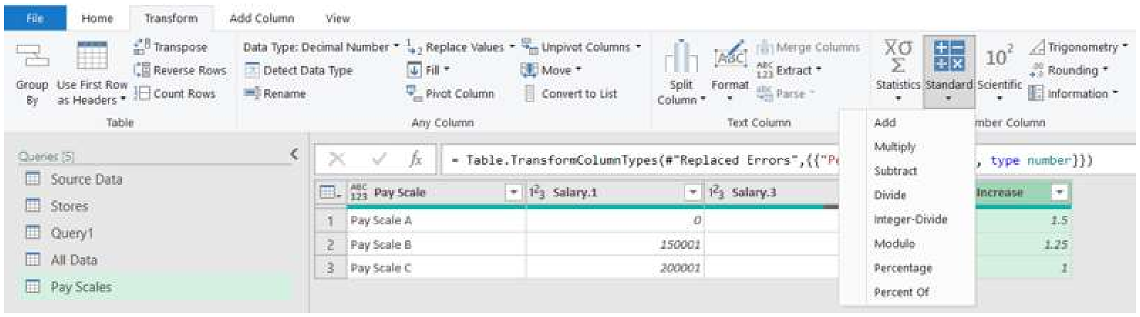


```
= Table.ReplaceErrorValues(#"Changed Type", {{"Salary.1", null}, {"Salary.3", null}})
```

	Pay Scale	Salary.1	Salary.3	Percentage Increase
1	Pay Scale A	0	15000	1.5
2	Pay Scale B	150001	20000	1.25
3	Pay Scale C	200001	null	1

Query Settings
PROPERTIES
Name: Pay Scales
APPLIED STEPS: Source, Removed Other Columns, Removed Blank Rows, Demoted Headers, Transposed Table, Renamed Columns, Split Column by Delimiter, Removed Other Columns1, Replaced Value, Changed Type, Replaced Errors

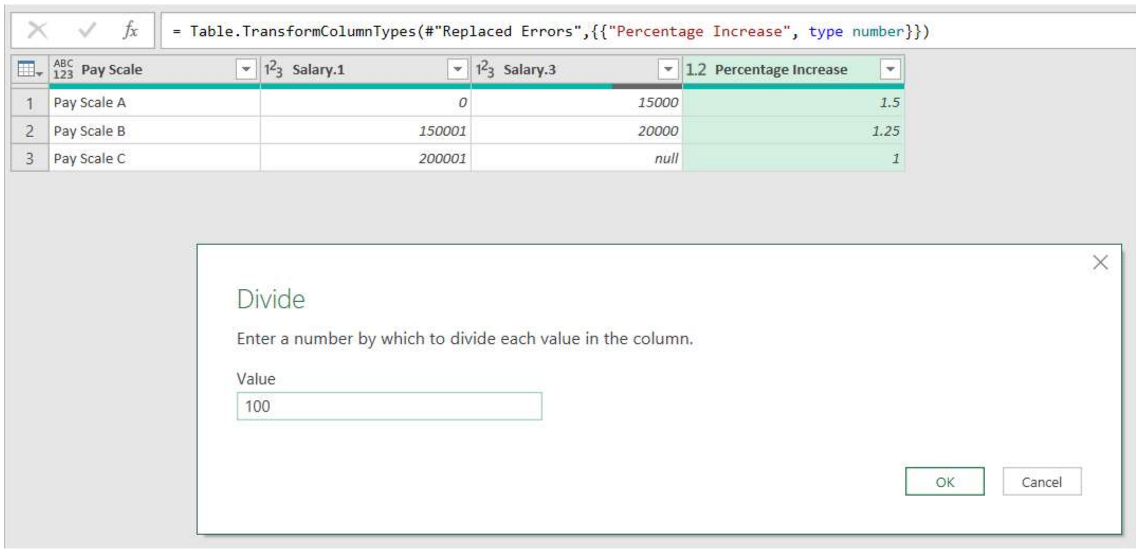
Next, I need to transform the Percentage Increase column. If I make it the data type Percentage, I will get values of over 100: I need to divide the values by 100 first, which I can do from the Transform tab, but first I change the data type to ‘Decimal Number’. This will allow me to access the ‘Standard’ dropdown.



```
= Table.TransformColumnTypes(#"Replaced Errors",{{"Pe ... , type number}})
```

	Pay Scale	Salary.1	Salary.3	Percentage Increase
1	Pay Scale A	0		1.5
2	Pay Scale B	150001		1.25
3	Pay Scale C	200001		1

I choose ‘Divide’ and enter 100 in the dialog.



```
= Table.TransformColumnTypes(#"Replaced Errors",{{"Percentage Increase", type number}})
```

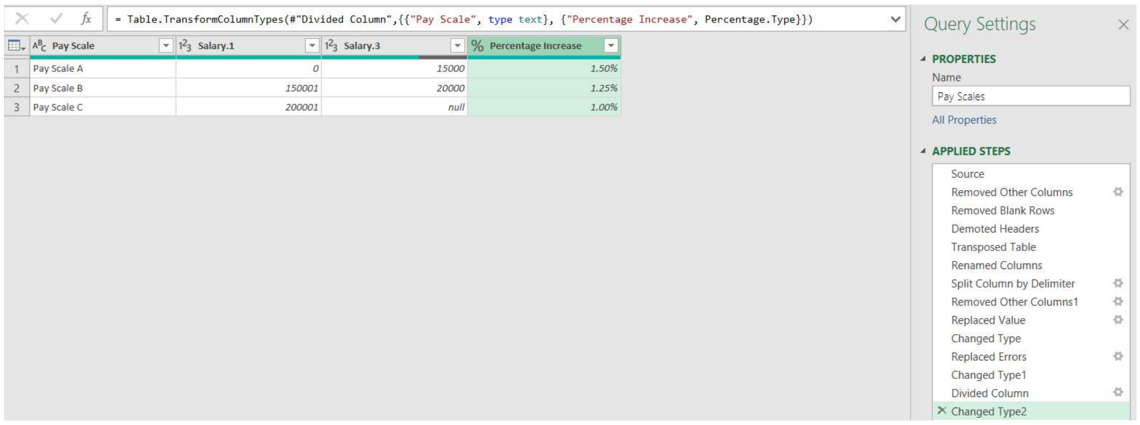
	Pay Scale	Salary.1	Salary.3	Percentage Increase
1	Pay Scale A	0	15000	1.5
2	Pay Scale B	150001	20000	1.25
3	Pay Scale C	200001	null	1

Divide

Enter a number by which to divide each value in the column.

Value: 100

I can now set the correct data types for all of the columns.



```
= Table.TransformColumnTypes(#"Divided Column",{{"Pay Scale", type text}, {"Percentage Increase", Percentage.Type}})
```

	Pay Scale	Salary.1	Salary.3	Percentage Increase
1	Pay Scale A	0	15000	1.50%
2	Pay Scale B	150001	20000	1.25%
3	Pay Scale C	200001	null	1.00%

Query Settings

PROPERTIES

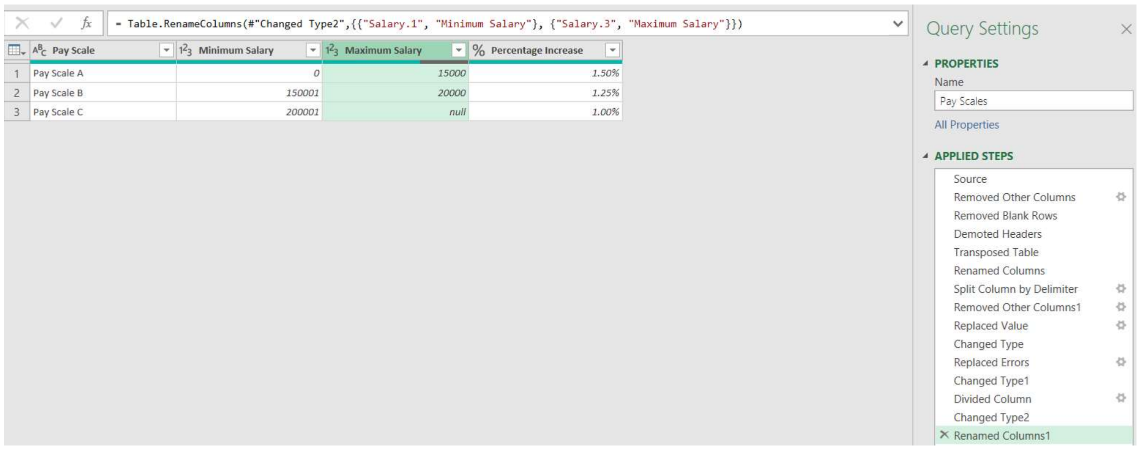
Name: Pay Scales

All Properties

APPLIED STEPS

- Source
- Removed Other Columns
- Removed Blank Rows
- Demoted Headers
- Transposed Table
- Renamed Columns
- Split Column by Delimiter
- Removed Other Columns1
- Replaced Value
- Changed Type
- Replaced Errors
- Changed Type1
- Divided Column
- Changed Type2

I rename the salary columns, and my table is ready to ‘Close & Load’.



```
= Table.RenameColumns(#"Changed Type2",{{"Salary.1", "Minimum Salary"}, {"Salary.3", "Maximum Salary"}})
```

	Pay Scale	Minimum Salary	Maximum Salary	Percentage Increase
1	Pay Scale A	0	15000	1.50%
2	Pay Scale B	150001	20000	1.25%
3	Pay Scale C	200001	null	1.00%

Query Settings

PROPERTIES

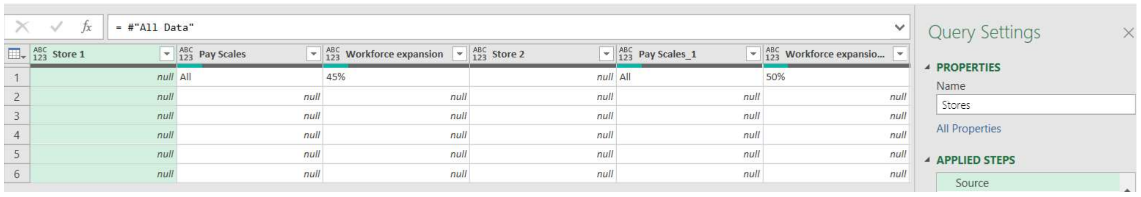
Name: Pay Scales

All Properties

APPLIED STEPS

- Source
- Removed Other Columns
- Removed Blank Rows
- Demoted Headers
- Transposed Table
- Renamed Columns
- Split Column by Delimiter
- Removed Other Columns1
- Replaced Value
- Changed Type
- Replaced Errors
- Changed Type1
- Divided Column
- Changed Type2
- Renamed Columns1

Next, I turn to the **Stores** table.



= #"All Data"

	Store 1	Pay Scales	Workforce expansion	Store 2	Pay Scales_1	Workforce expansio...
1	null	All	45%	null	All	50%
2	null	null	null	null	null	null
3	null	null	null	null	null	null
4	null	null	null	null	null	null
5	null	null	null	null	null	null
6	null	null	null	null	null	null

Query Settings

PROPERTIES

Name

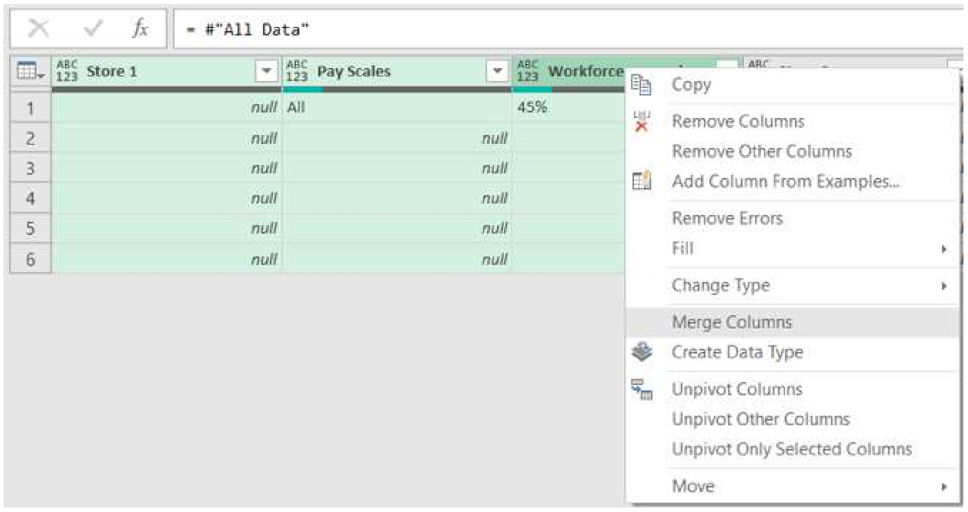
Stores

All Properties

APPLIED STEPS

Source

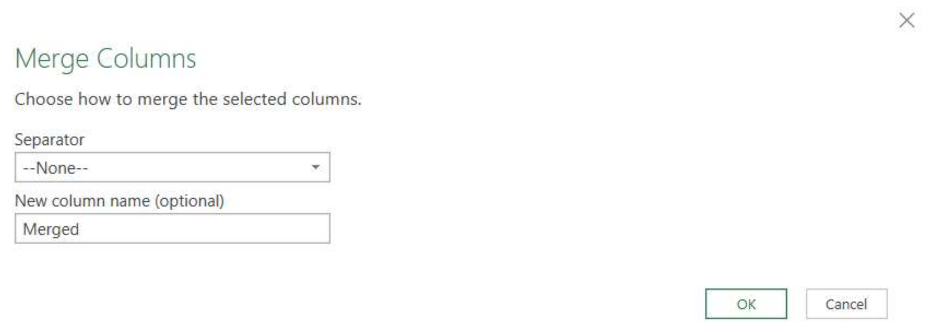
I have 10 stores in my table, and I need to perform the same transformations for each one. The first step I will take is to merge the columns I need for **Store 1**. I select **Store 1**, **Pay Scales** and **Workforce expansion** whilst holding down the **CTRL** key. When I right-click, I have the option to 'Merge Columns'.



= #"All Data"

	Store 1	Pay Scales	Workforce
1	null	All	45%
2	null	null	
3	null	null	
4	null	null	
5	null	null	
6	null	null	

Clicking on this option reveals a dialog:



Merge Columns

Choose how to merge the selected columns.

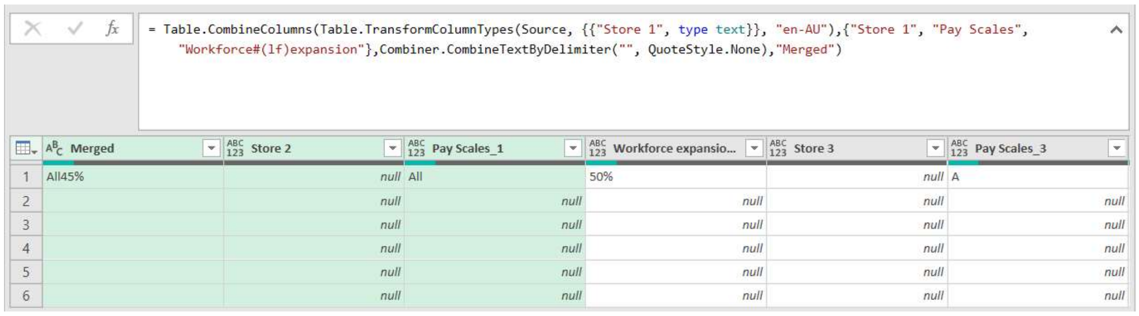
Separator

--None--

New column name (optional)

Merged

I choose not to use a separator, since I can split by non-numeric and numeric characters later. I want to call my new column **Store 1**, but Power Query won't let me do this as this is one of the names of the original columns, so for now I take the default **Merged**.



```
= Table.CombineColumns(Table.TransformColumnTypes(Source, {{"Store 1", type text}}, "en-AU"),{"Store 1", "Pay Scales",
    "Workforce#(lf)expansion"},Combiner.CombineTextByDelimiter("", QuoteStyle.None),"Merged")
```

	Merged	Store 2	Pay Scales_1	Workforce expansio...	Store 3	Pay Scales_3
1	All45%	null	All	50%	null	A
2		null	null	null	null	null
3		null	null	null	null	null
4		null	null	null	null	null
5		null	null	null	null	null
6		null	null	null	null	null

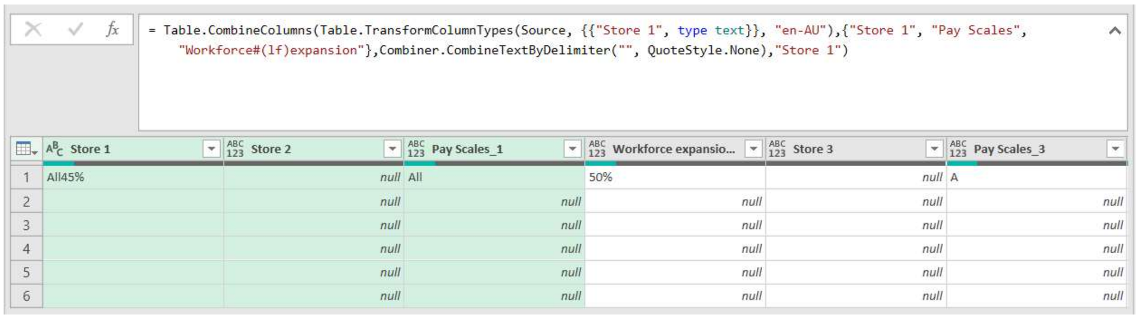
I can adjust the step created from:

```
Table.CombineColumns(Table.TransformColumnTypes(Source, {{"Store 1", type text}}, "en-
AU"),{"Store 1", "Pay Scales",
"Workforce#(lf)expansion"},Combiner.CombineTextByDelimiter("", QuoteStyle.None),"Merged")
```

to:

```
Table.CombineColumns(Table.TransformColumnTypes(Source, {{"Store 1", type text}}, "en-
AU"),{"Store 1", "Pay Scales",
"Workforce#(lf)expansion"},Combiner.CombineTextByDelimiter("", QuoteStyle.None),"Store 1")
```

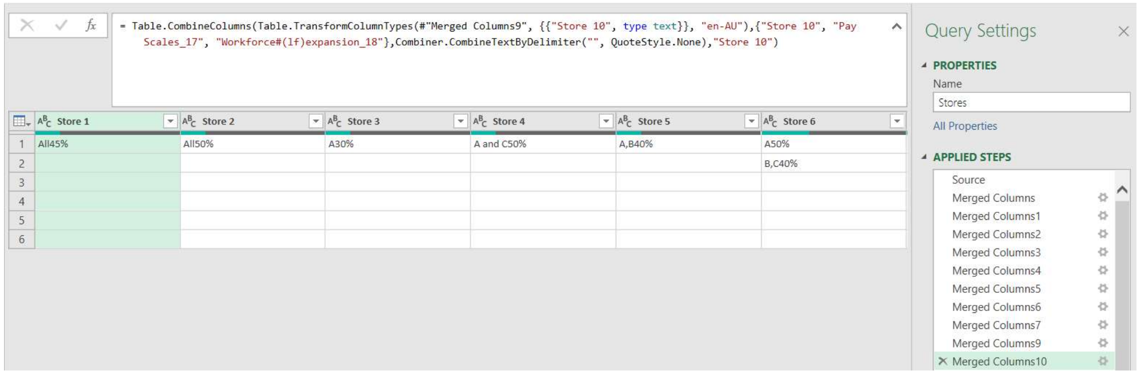
This is then accepted with no issues:



```
= Table.CombineColumns(Table.TransformColumnTypes(Source, {{"Store 1", type text}}, "en-AU"),{"Store 1", "Pay Scales",
    "Workforce#(lf)expansion"},Combiner.CombineTextByDelimiter("", QuoteStyle.None),"Store 1")
```

	Store 1	Store 2	Pay Scales_1	Workforce expansio...	Store 3	Pay Scales_3
1	All45%	null	All	50%	null	A
2		null	null	null	null	null
3		null	null	null	null	null
4		null	null	null	null	null
5		null	null	null	null	null
6		null	null	null	null	null

I repeat this for the other stores.



```
= Table.CombineColumns(Table.TransformColumnTypes(#"Merged Columns9", {{"Store 10", type text}}, "en-AU"),{"Store 10", "Pay
    Scales_17", "Workforce#(lf)expansion_18"},Combiner.CombineTextByDelimiter("", QuoteStyle.None),"Store 10")
```

	Store 1	Store 2	Store 3	Store 4	Store 5	Store 6
1	All45%	All50%	A30%	A and C50%	A,B40%	A50%
2						B,C40%
3						
4						
5						
6						

Query Settings

PROPERTIES

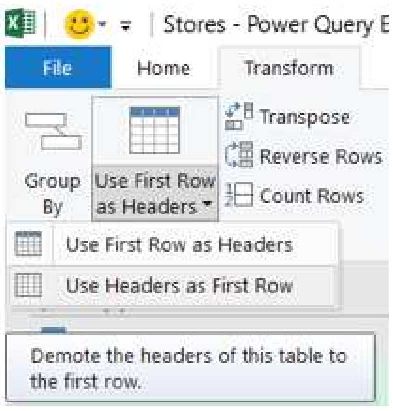
Name: Stores

All Properties

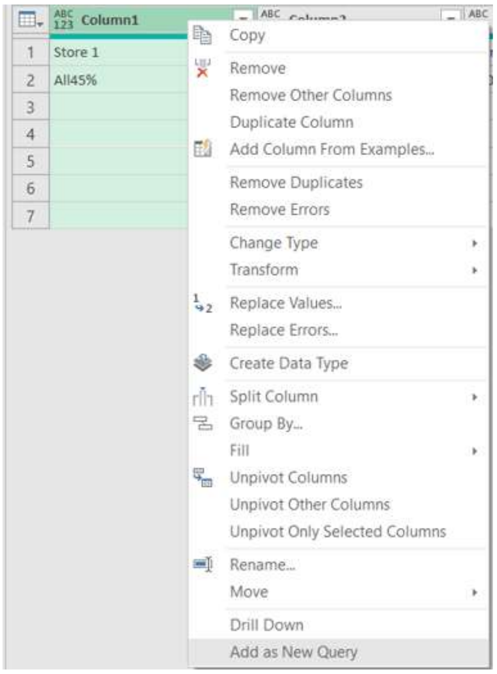
APPLIED STEPS

- Source
- Merged Columns
- Merged Columns1
- Merged Columns2
- Merged Columns3
- Merged Columns4
- Merged Columns5
- Merged Columns6
- Merged Columns7
- Merged Columns9
- Merged Columns10

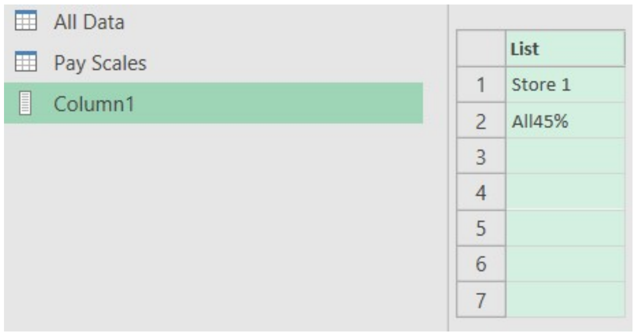
I could keep working on all the stores together, but it would soon become a large table. Instead, I am going to work on the stores separately. Before I do this, I need to make sure that the store name is included in the data, so I demote headers using the option from the Transform tab:



My data is now safely stored in the rows, and I can create a query from a column. I right-click with **Column1** selected:



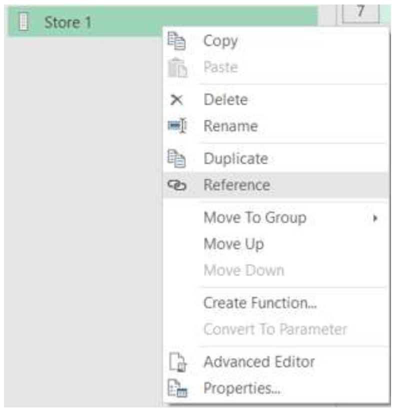
This creates a new List Query.



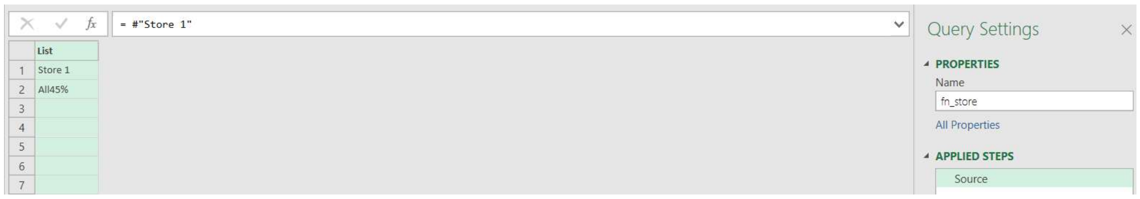
	List
1	Store 1
2	All45%
3	
4	
5	
6	
7	

I will rename my List Query to **Store 1** to reflect the data.

What I want now, is to create a series of steps that will work on any of the stores. I need a function. I start by taking a reference copy of **Store 1.**



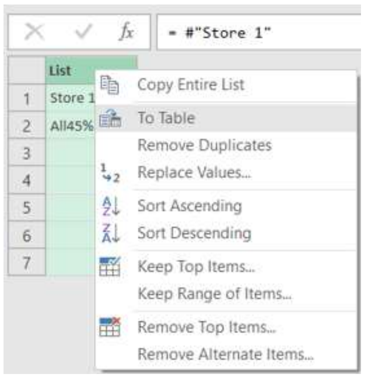
I call my new query **fn_store**.


```
= #"Store 1"
```

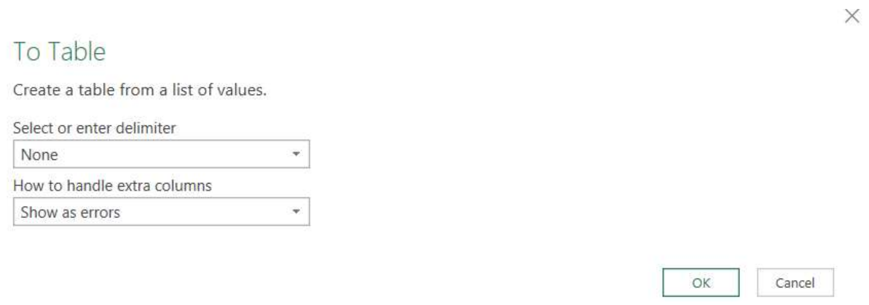
	List
1	Store 1
2	All45%
3	
4	
5	
6	
7	

Name: fn_store

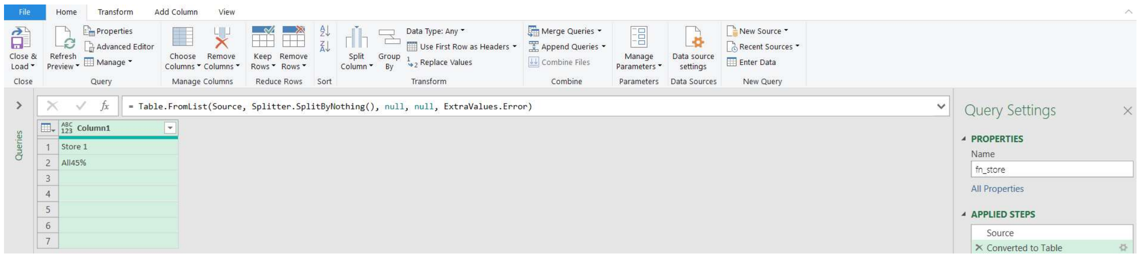
The first step is to convert this list to a table. I can do this from the List tab or I may otherwise right-click.



A dialog appears where I shall accept the defaults:

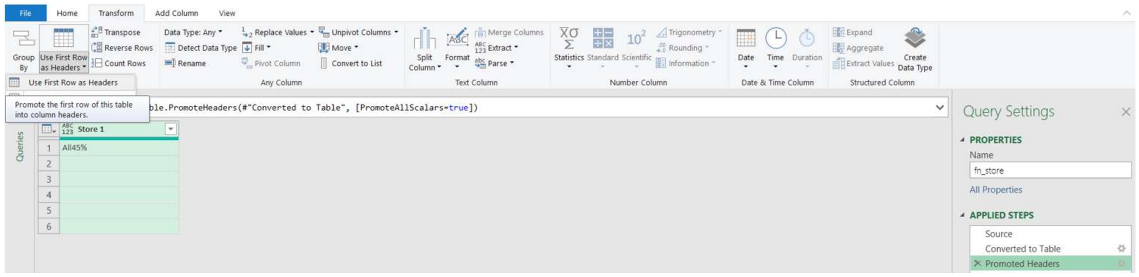

To Table
Create a table from a list of values.
Select or enter delimiter
None
How to handle extra columns
Show as errors
OK
Cancel

This gives me access to the other tabs.



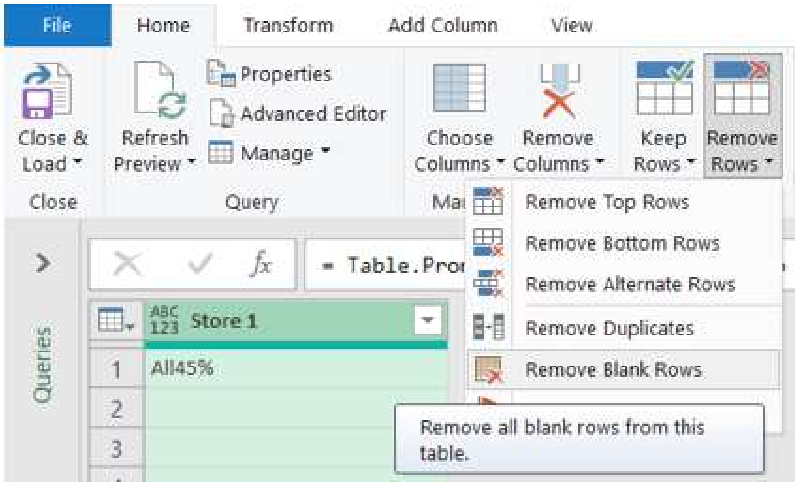
```
= Table.FromList(Source, Splitter.SplitByNothing(), null, null, ExtraValues.Error)
```

I start by promoting the first row to headers using the option on the Transform Tab.

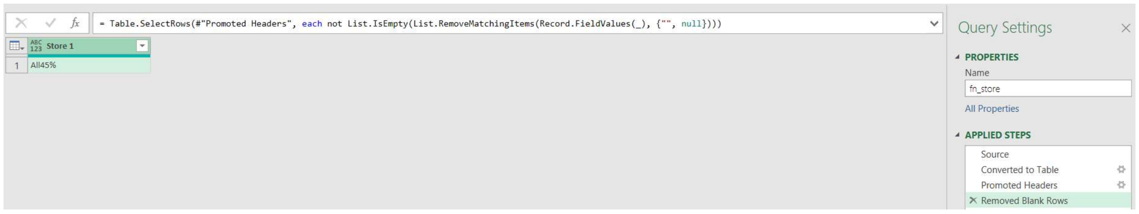


```
ble.PromoteHeaders(#"Converted to Table", [PromoteAllScalars=true])
```

I am ready to create the transformations required and convert this query. I start by getting rid of any empty rows. I can do this by using 'Remove Blank Rows' from the Home tab:



This simplifies my table: for this store it is one row, but for other stores there may be more.



```
= Table.SelectRows(#"Promoted Headers", each not List.IsEmpty(List.RemoveMatchingItems(Record.FieldValues(_), {"", null})))
```

	Store 1
1	All45%

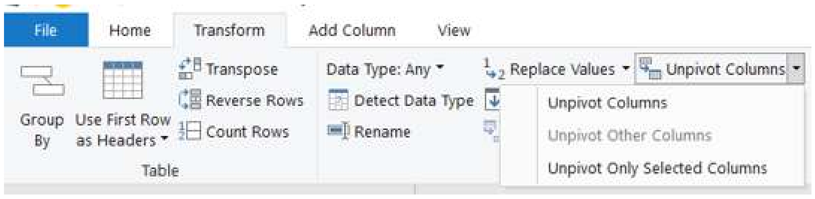
Query Settings

PROPERTIES

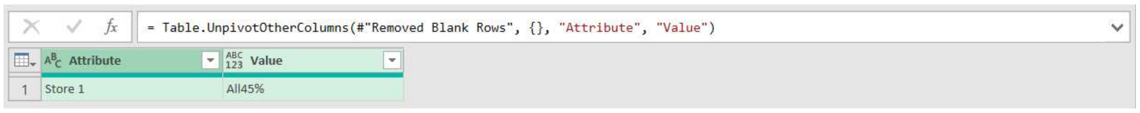
Name: fn_store

APPLIED STEPS: Source, Converted to Table, Promoted Headers, Removed Blank Rows

I need the store information to be a separate column. I can do this by employing the 'Unpivot Column' feature from the Transform tab.



I now have a column with the store data, and one for the pay scales and percentage increase of workforce. I will rename these columns later.



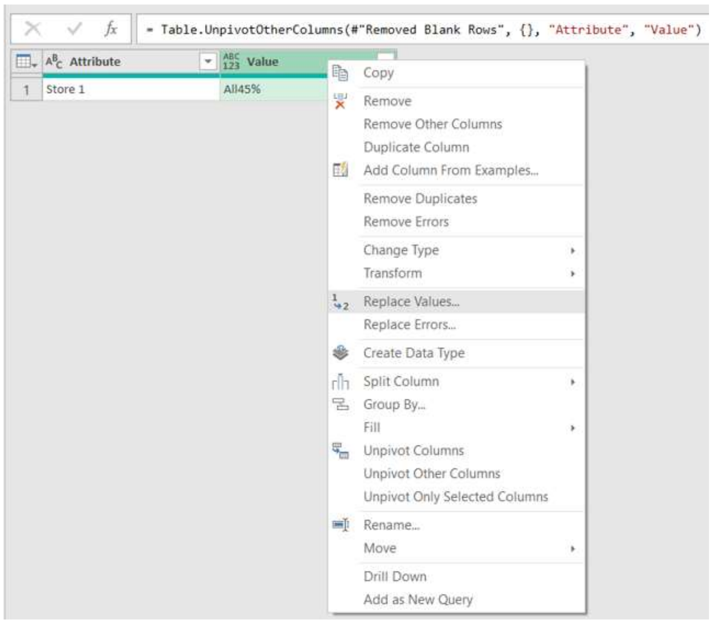
```
= Table.UnpivotOtherColumns(#"Removed Blank Rows", {}, "Attribute", "Value")
```

	Attribute	Value
1	Store 1	All45%

I need to keep in mind that this function needs to cope with the data for all the stores. This means I need to carry out some replacements:

- 'All' needs to be replaced by 'A,B,C'
- ' and ' needs to be replaced by ','

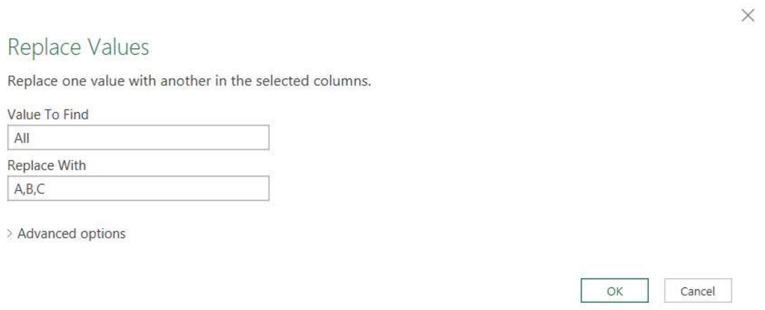
I can then use comma (,) as my delimiter to split up the pay scales. I begin with 'Replace Values' on the 'Transform' tab, or else by right-clicking:



```
= Table.UnpivotOtherColumns(#"Removed Blank Rows", {}, "Attribute", "Value")
```

	Attribute	Value
1	Store 1	All45%

First I replace 'All':



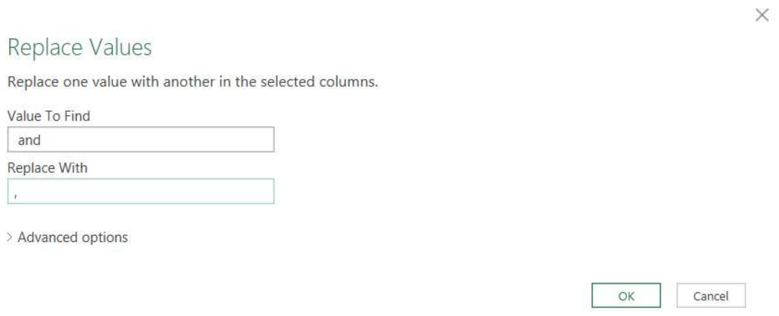
Replace Values

Replace one value with another in the selected columns.

Value To Find: All

Replace With: A,B,C

Then I replace ' and ':



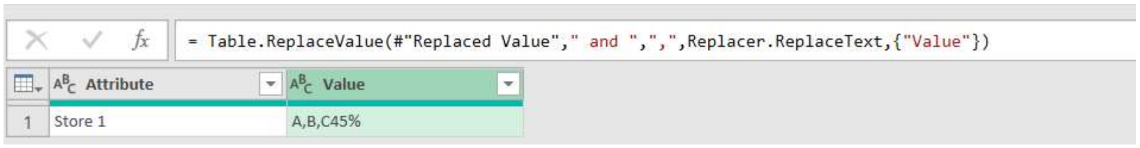
Replace Values

Replace one value with another in the selected columns.

Value To Find: and

Replace With: ,

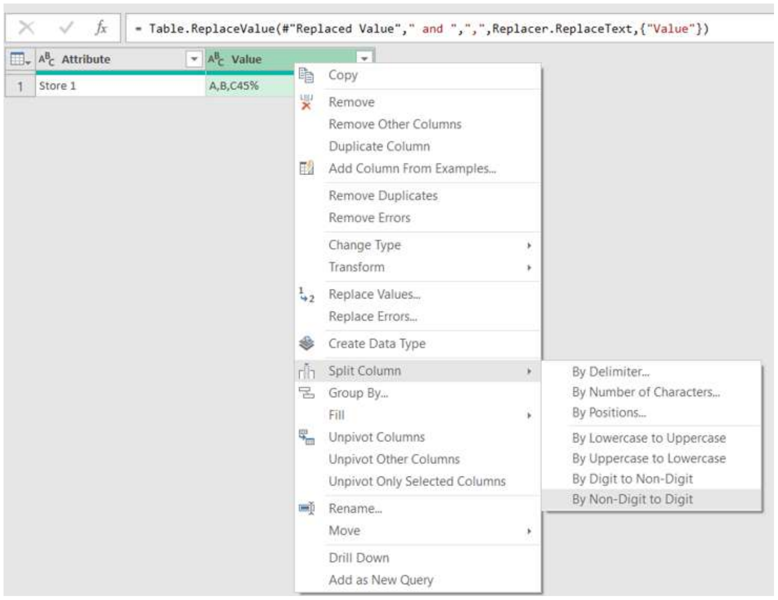
My data is now ready to be split:



```
= Table.ReplaceValue(#"Replaced Value"," and ",",",Replacer.ReplaceText,{"Value"})
```

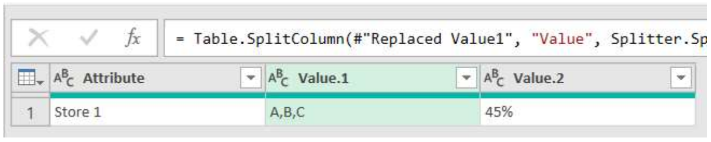
	Attribute	Value
1	Store 1	A,B,C45%

I am going to split the column into pay scales and increase percentage. I can do this by right-clicking, and choosing to 'Split Column' 'By Non-Digit to Digit':



```
= Table.ReplaceValue(#"Replaced Value"," and ",",",Replacer.ReplaceText,{"Value"})
```

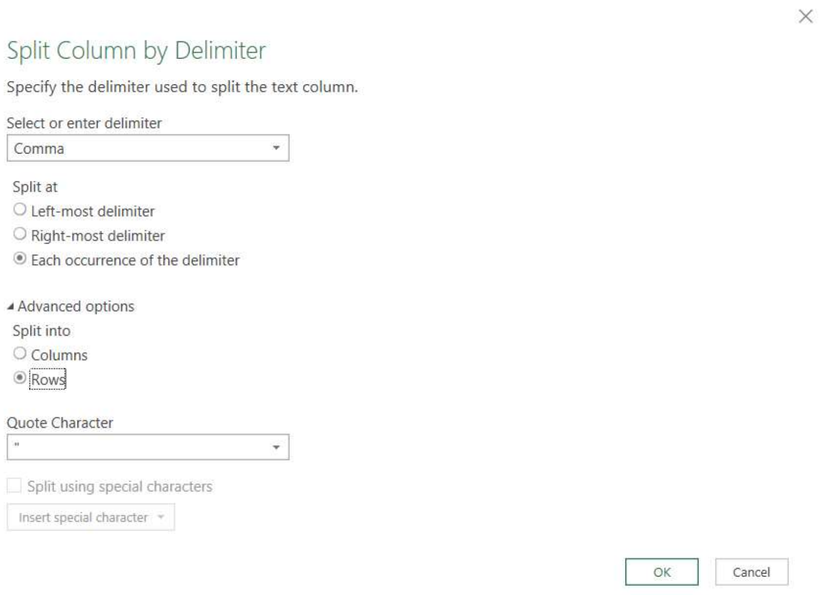
This gives me an extra column.



```
= Table.SplitColumn(#"Replaced Value1", "Value", Splitter.Sp
```

	Attribute	Value.1	Value.2
1	Store 1	A,B,C	45%

Now I can split up the pay scales: I will do this using Comma as a Delimiter, and I will choose to 'Split into Rows' in the advanced options.



Split Column by Delimiter

Specify the delimiter used to split the text column.

Select or enter delimiter: Comma

Split at
- ○ Left-most delimiter
- ○ Right-most delimiter
- ● Each occurrence of the delimiter

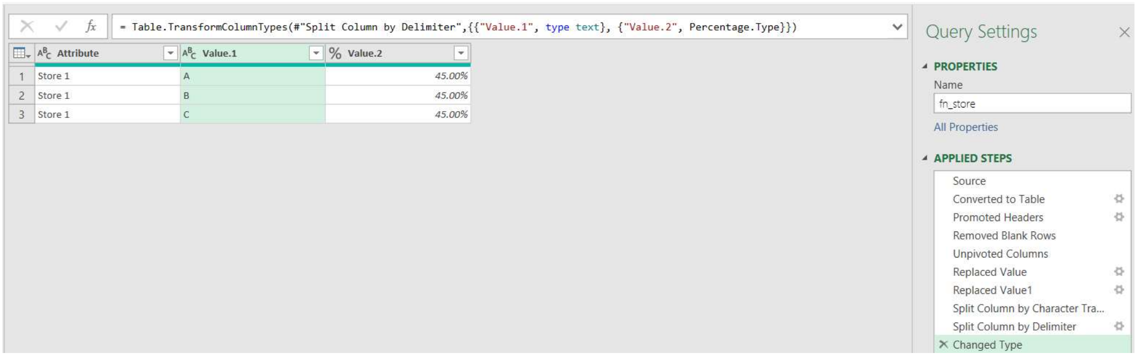
Advanced options

Split into
- ○ Columns
- ● Rows

Quote Character: "

Split using special characters

This gives me more rows:



```
= Table.TransformColumnTypes(#"Split Column by Delimiter",{{"Value.1", type text}, {"Value.2", Percentage.Type}})
```

	Attribute	Value.1	Value.2
1	Store 1	A	45.00%
2	Store 1	B	45.00%
3	Store 1	C	45.00%

Query Settings

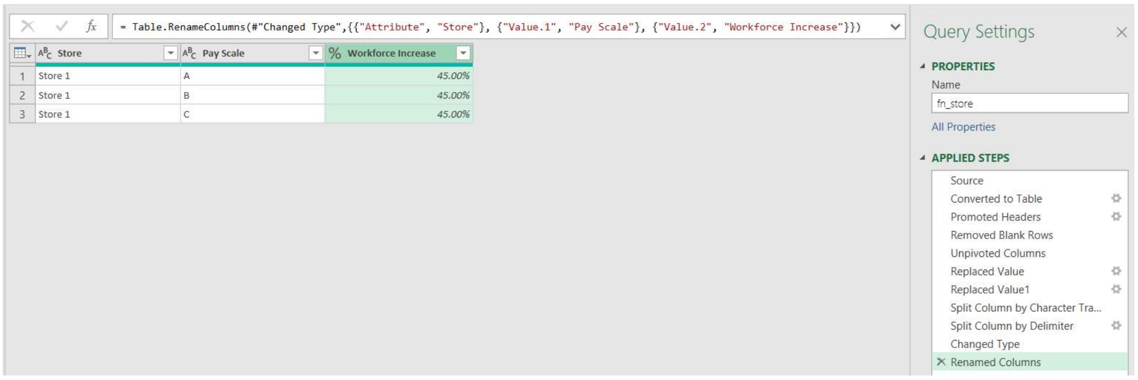
PROPERTIES

Name: fn_store

All Properties

APPLIED STEPS
- Source
- Converted to Table
- Promoted Headers
- Removed Blank Rows
- Unpivoted Columns
- Replaced Value
- Replaced Value1
- Split Column by Character Tra...
- Split Column by Delimiter
- Changed Type

I keep the automated 'Changed Type' step and rename the columns.



```
= Table.RenameColumns(#"Changed Type",{{"Attribute", "Store"}, {"Value.1", "Pay Scale"}, {"Value.2", "Workforce Increase"}})
```

	Store	Pay Scale	Workforce Increase
1	Store 1	A	45.00%
2	Store 1	B	45.00%
3	Store 1	C	45.00%

Query Settings

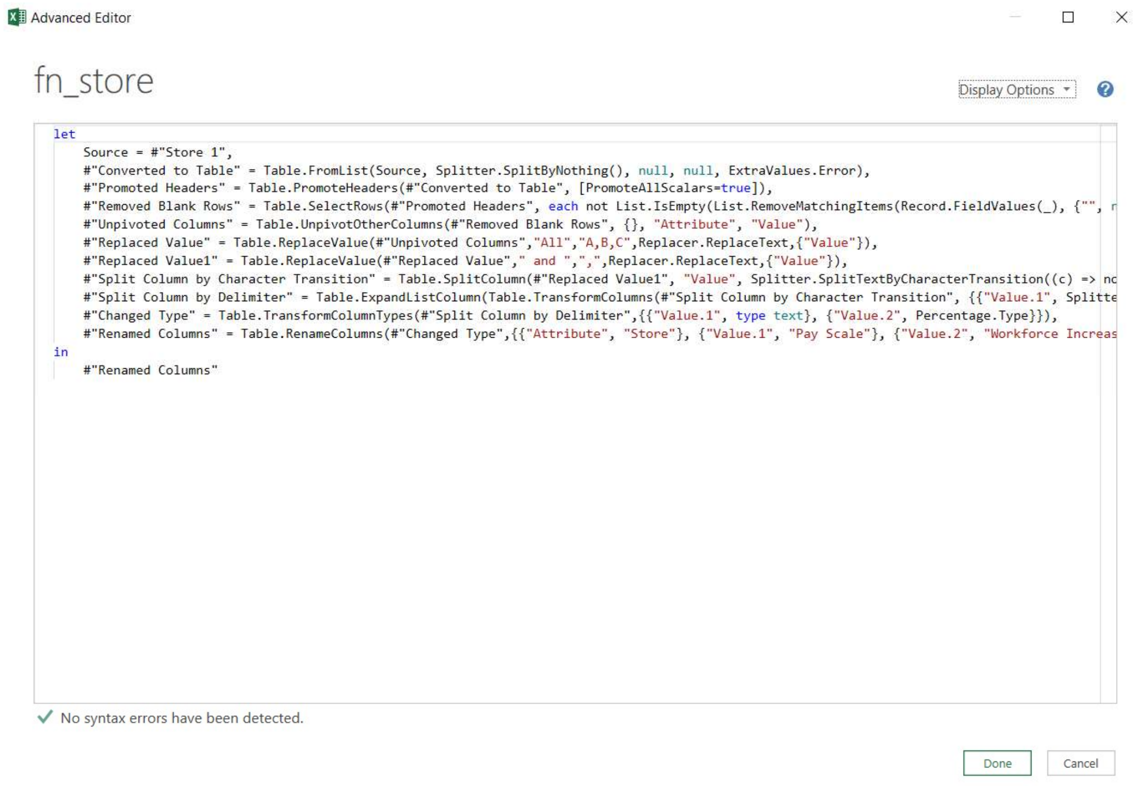
PROPERTIES

Name: fn_store

All Properties

APPLIED STEPS
- Source
- Converted to Table
- Promoted Headers
- Removed Blank Rows
- Unpivoted Columns
- Replaced Value
- Replaced Value1
- Split Column by Character Tra...
- Split Column by Delimiter
- Changed Type
- Renamed Columns

My query is ready to be converted to a function. I need to parameterise, so I view the **M** code in the Advanced Editor, available from the Home tab.



```
let
    Source = #"Store 1",
    #"Converted to Table" = Table.FromList(Source, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    #"Promoted Headers" = Table.PromoteHeaders(#"Converted to Table", [PromoteAllScalars=true]),
    #"Removed Blank Rows" = Table.SelectRows(#"Promoted Headers", each not List.IsEmpty(List.RemoveMatchingItems(Record.FieldValues(_), {"", r
    #"Unpivoted Columns" = Table.UnpivotOtherColumns(#"Removed Blank Rows", {}, "Attribute", "Value"),
    #"Replaced Value" = Table.ReplaceValue(#"Unpivoted Columns","All","A,B,C",Replacer.ReplaceText,{"Value"}),
    #"Replaced Value1" = Table.ReplaceValue(#"Replaced Value"," and ",",",Replacer.ReplaceText,{"Value"}),
    #"Split Column by Character Transition" = Table.SplitColumn(#"Replaced Value1", "Value", Splitter.SplitTextByCharacterTransition((c) => nc
    #"Split Column by Delimiter" = Table.ExpandListColumn(Table.TransformColumns(#"Split Column by Character Transition", {{"Value.1", Splitte
    #"Changed Type" = Table.TransformColumnTypes(#"Split Column by Delimiter",{{"Value.1", type text}, {"Value.2", Percentage.Type}}),
    #"Renamed Columns" = Table.RenameColumns(#"Changed Type",{{"Attribute", "Store"}, {"Value.1", "Pay Scale"}, {"Value.2", "Workforce Increas
in
    #"Renamed Columns"
```

The only line I need to change is the Source step. I am going to introduce a parameter **p_store**, which will receive any of the store columns as a list. The **M** code before the 'let' statement will be:

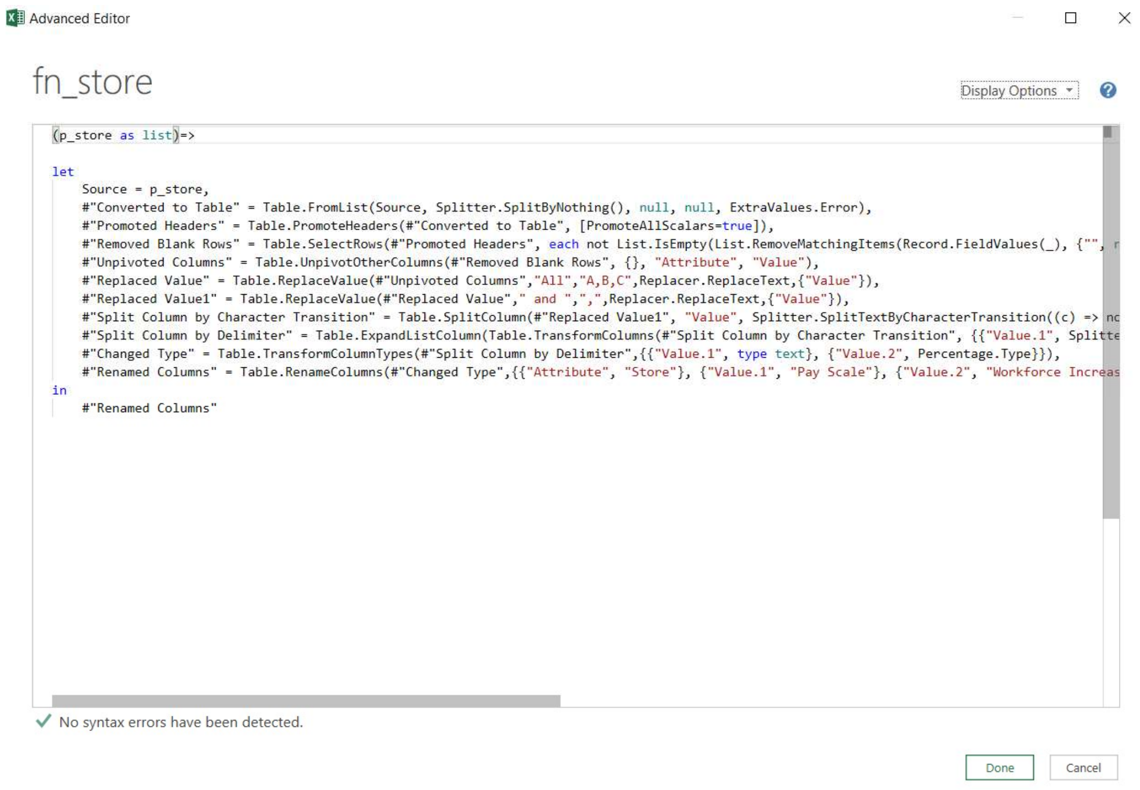
```
(p_store as list) =>
```

and the Source step will change from:

```
Source = #"Store 1",
```

to:

```
Source = p_store,
```



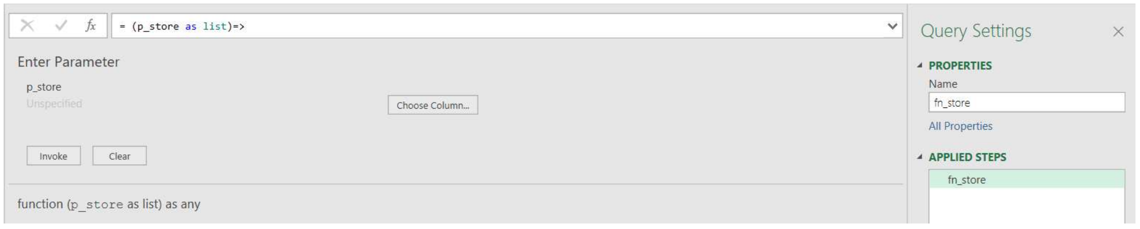
Advanced Editor

fn_store

```
(p_store as list)=>

let
    Source = p_store,
    #"Converted to Table" = Table.FromList(Source, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    #"Promoted Headers" = Table.PromoteHeaders(#"Converted to Table", [PromoteAllScalars=true]),
    #"Removed Blank Rows" = Table.SelectRows(#"Promoted Headers", each not List.IsEmpty(List.RemoveMatchingItems(Record.FieldValues(_), {"", r
    #"Unpivoted Columns" = Table.UnpivotOtherColumns(#"Removed Blank Rows", {}, "Attribute", "Value"),
    #"Replaced Value" = Table.ReplaceValue(#"Unpivoted Columns","All","A,B,C",Replacer.ReplaceText,{"Value"}),
    #"Replaced Value1" = Table.ReplaceValue(#"Replaced Value"," and ",",",Replacer.ReplaceText,{"Value"}),
    #"Split Column by Character Transition" = Table.SplitColumn(#"Replaced Value1", "Value", Splitter.SplitTextByCharacterTransition((c) => nc
    #"Split Column by Delimiter" = Table.ExpandListColumn(Table.TransformColumns(#"Split Column by Character Transition", {{"Value.1", Splitte
    #"Changed Type" = Table.TransformColumnTypes(#"Split Column by Delimiter",{{"Value.1", type text}, {"Value.2", Percentage.Type}}),
    #"Renamed Columns" = Table.RenameColumns(#"Changed Type",{{"Attribute", "Store"}, {"Value.1", "Pay Scale"}, {"Value.2", "Workforce Increas
in
    #"Renamed Columns"
```

✓ No syntax errors have been detected.

As soon as I click 'Done', Power Query recognises that my query is now a function.



```
= (p_store as list)=>
```

Enter Parameter

p_store
Unspecified

function (p_store as list) as any

Query Settings

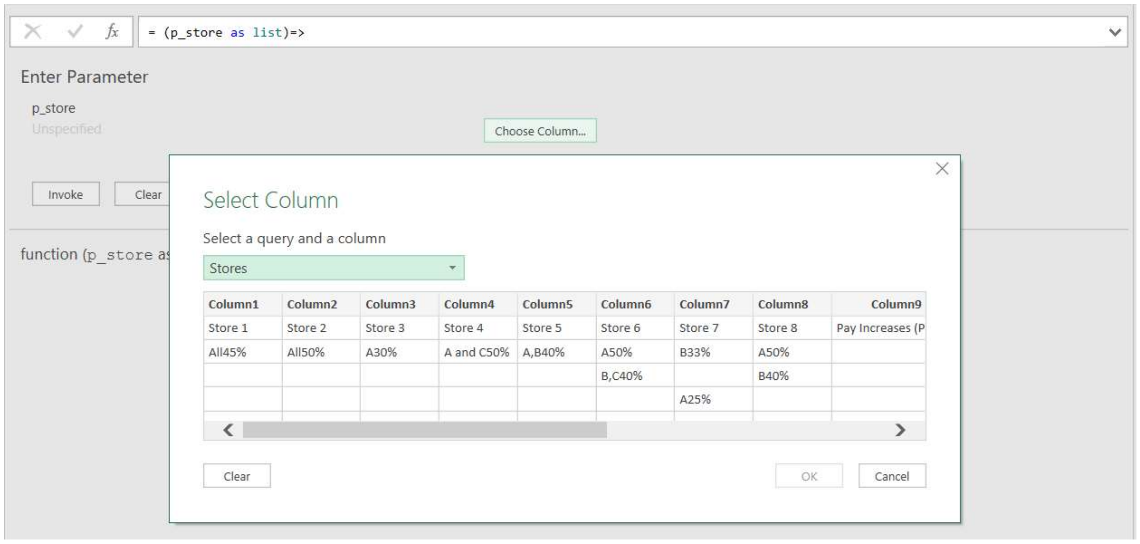
PROPERTIES

Name: fn_store

APPLIED STEPS

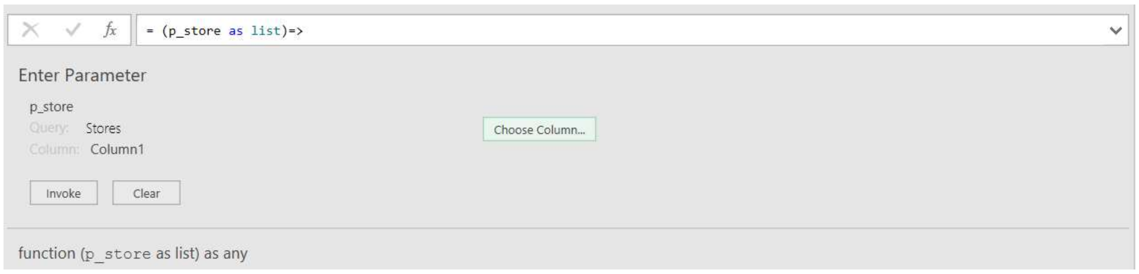
fn_store

It is prompting me for a column (field). This means that this function will now create a table from any of the store columns in the **Stores** query.

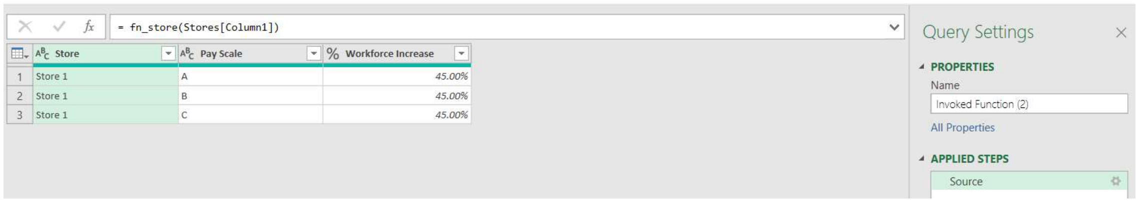


Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9
Store 1	Store 2	Store 3	Store 4	Store 5	Store 6	Store 7	Store 8	Pay Increases (P
All45%	All50%	A30%	A and C50%	A,B40%	A50%	B33%	A50%	
					B,C40%		B40%	
						A25%		

I can test the function on **Column1**:

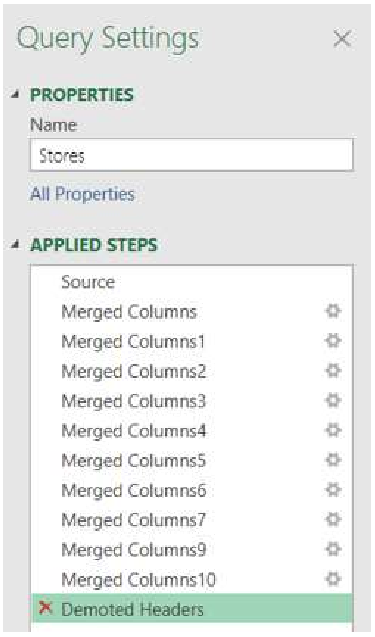


Invoking this query will give me a table:



	Store	Pay Scale	Workforce Increase
1	Store 1	A	45.00%
2	Store 1	B	45.00%
3	Store 1	C	45.00%

In the Stores query, I need to make sure I only have the columns I need. To do this, I will remove the ‘Demote Headers’ step. I can reapply it before I invoke the function.



Query Settings

PROPERTIES

Name

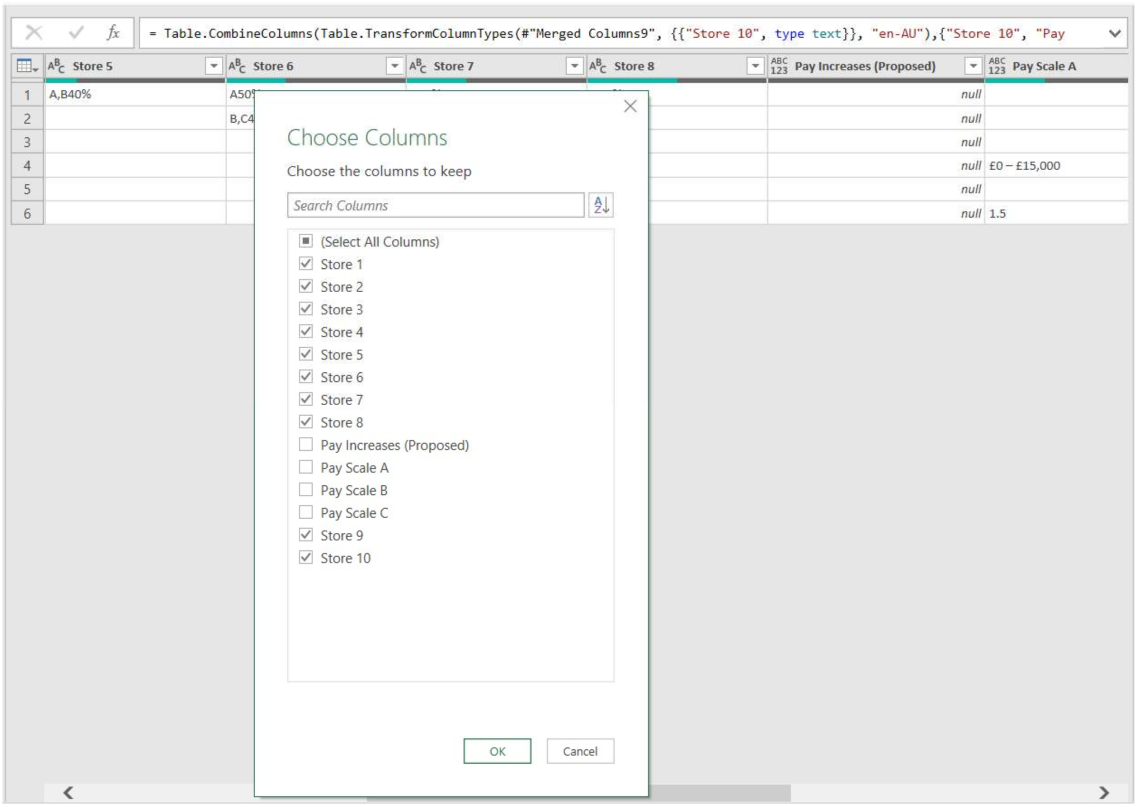
Stores

All Properties

APPLIED STEPS

- Source
- Merged Columns
- Merged Columns1
- Merged Columns2
- Merged Columns3
- Merged Columns4
- Merged Columns5
- Merged Columns6
- Merged Columns7
- Merged Columns9
- Merged Columns10
- Demoted Headers

This means I have the store names in the headings to choose:



```
= Table.CombineColumns(Table.TransformColumnTypes(#"Merged Columns9", {{"Store 10", type text}}, "en-AU"),{"Store 10", "Pay
```

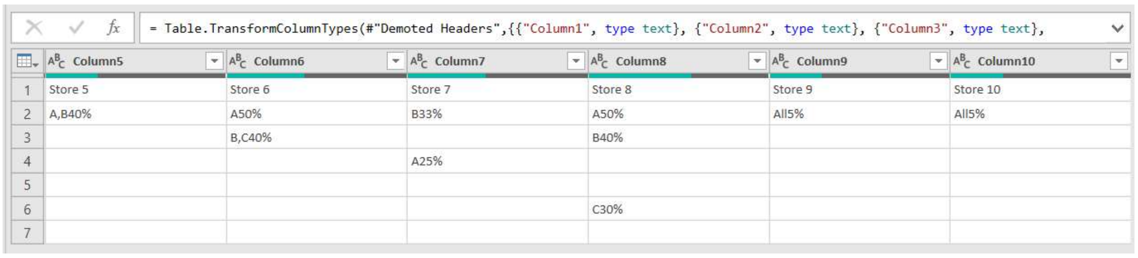
	Store 5	Store 6	Store 7	Store 8	Pay Increases (Proposed)	Pay Scale A
1	A,B40%	A50			null	
2		B,C4			null	
3					null	
4					null	£0 – £15,000
5					null	
6					null	1.5

Choose Columns

Choose the columns to keep

- (Select All Columns)
- [✓] Store 1
- [✓] Store 2
- [✓] Store 3
- [✓] Store 4
- [✓] Store 5
- [✓] Store 6
- [✓] Store 7
- [✓] Store 8
- [ ] Pay Increases (Proposed)
- [ ] Pay Scale A
- [ ] Pay Scale B
- [ ] Pay Scale C
- [✓] Store 9
- [✓] Store 10

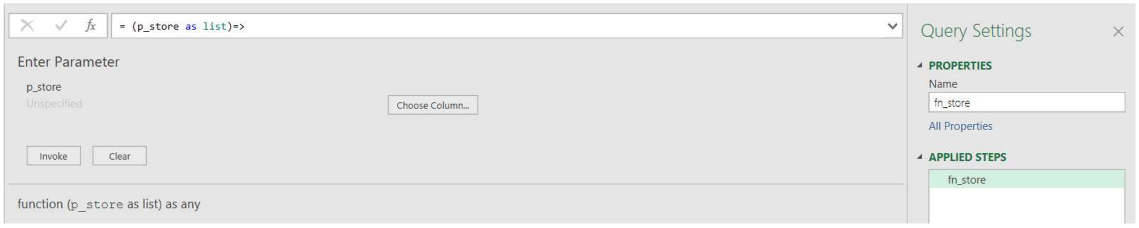
This gives me the store columns, and I can demote the headers again:



```
= Table.TransformColumnTypes(#"Demoted Headers",{{"Column1", type text}, {"Column2", type text}, {"Column3", type text},
```

	Column5	Column6	Column7	Column8	Column9	Column10
1	Store 5	Store 6	Store 7	Store 8	Store 9	Store 10
2	A,B40%	A50%	B33%	A50%	All5%	All5%
3		B,C40%		B40%		
4			A25%			
5						
6				C30%		
7						

Stores is ready to use as the Source for fn_store. I go back to fn_store:



```
= (p_store as list)=>
```

Enter Parameter

p_store
Unspecified

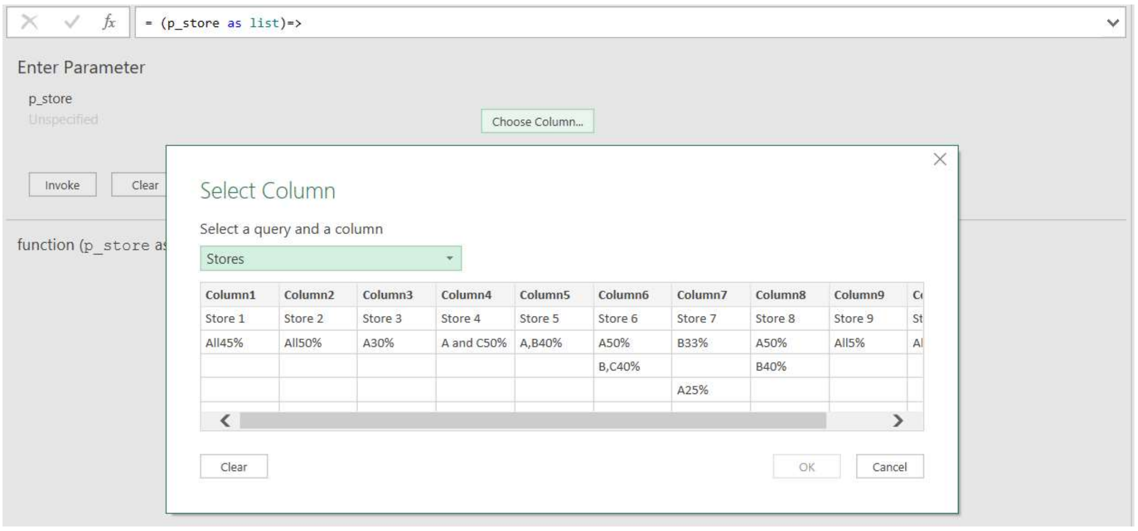
Choose Column...

Invoke Clear

function (p_store as list) as any

Query Settings
PROPERTIES
Name
fn_store
All Properties
APPLIED STEPS
fn_store

I need to provide a column from the **Stores** query.



```
= (p_store as list)=>
```

Enter Parameter

p_store
Unspecified

Select Column

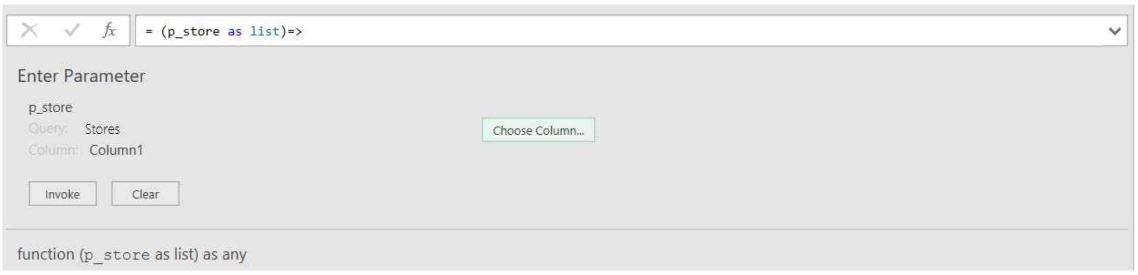
Select a query and a column

Stores

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9
Store 1	Store 2	Store 3	Store 4	Store 5	Store 6	Store 7	Store 8	Store 9
All45%	All50%	A30%	A and C50%	A,B40%	A50%	B33%	A50%	All5%
					B,C40%		B40%	
						A25%		

Clear OK Cancel

I use **Column1**:



```
= (p_store as list)=>
```

Enter Parameter

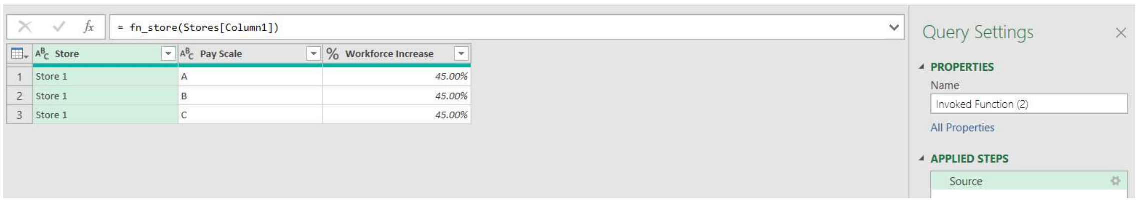
p_store
Query: Stores
Column: Column1

Choose Column...

Invoke Clear

function (p_store as list) as any

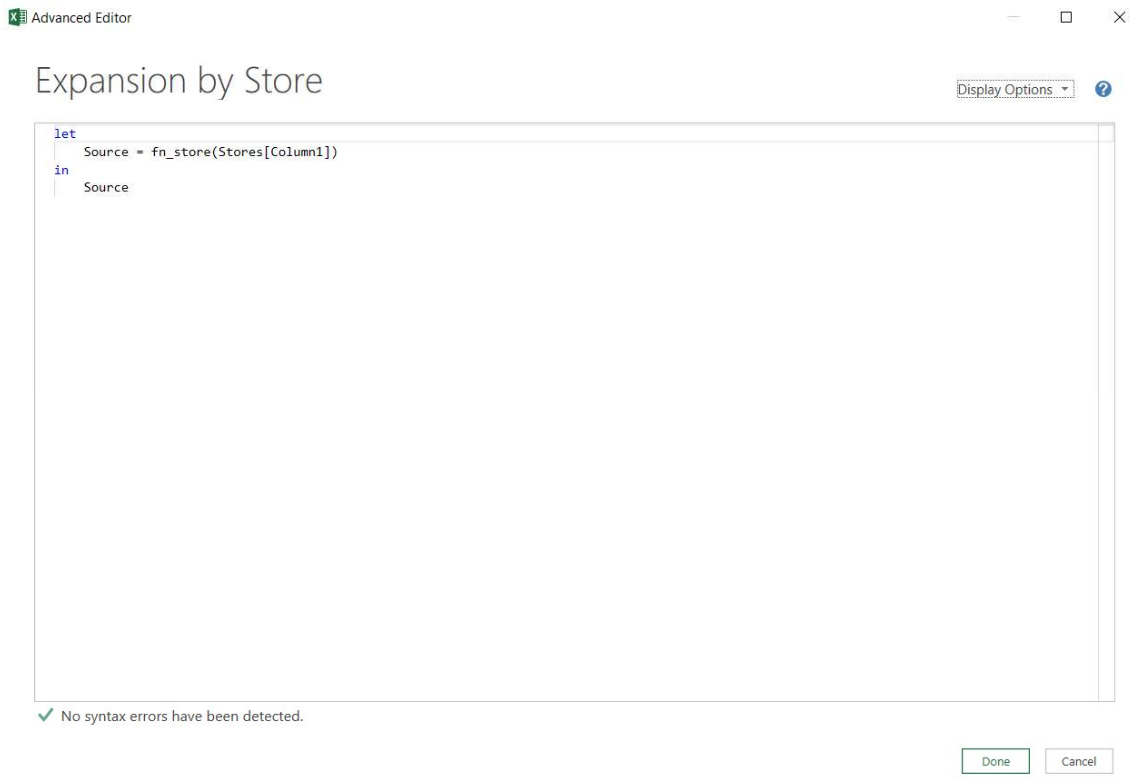
Invoking this query will give me a table:



```
= fn_store(Stores[Column1])
```

	Store	Pay Scale	Workforce Increase
1	Store 1	A	45.00%
2	Store 1	B	45.00%
3	Store 1	C	45.00%

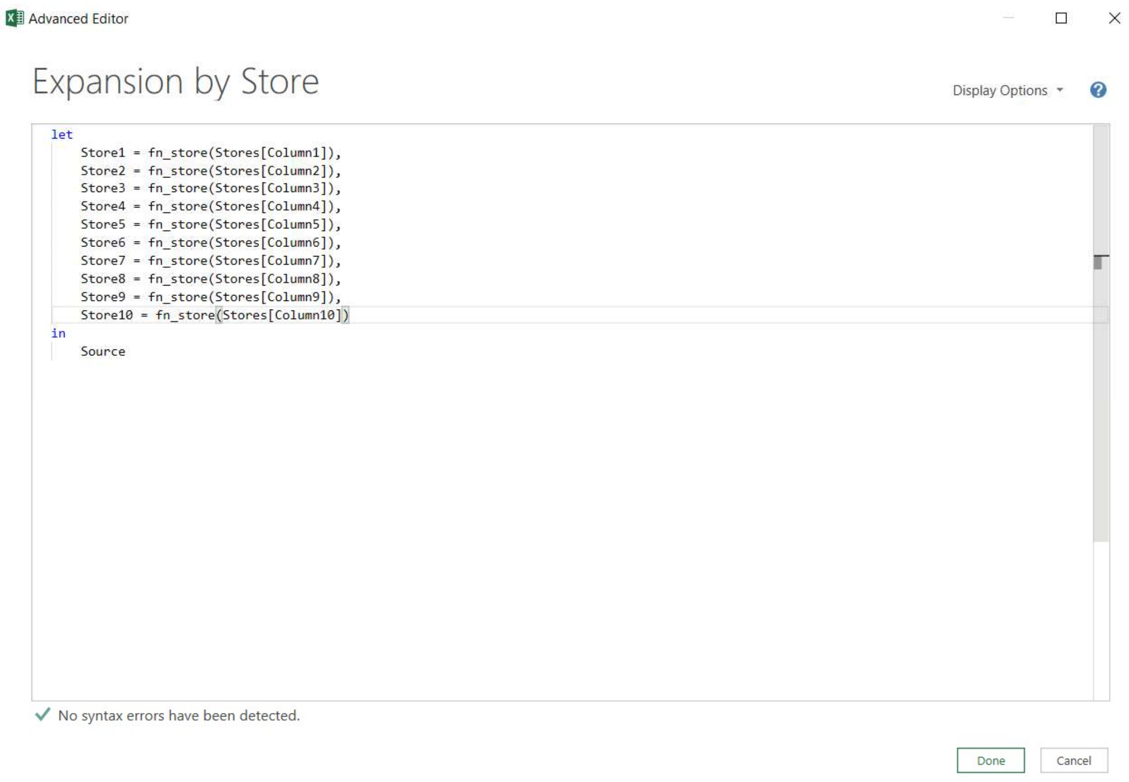
I rename this table **Expansion by Store**. I am going to use the **M** code already generated as a basis for this query.



```
let
    Source = fn_store(Stores[Column1])
in
    Source
```

No syntax errors have been detected.

I take the **M** code created for **Column1**, and replicate it for 10 columns. I have renamed the steps to make it clear what I am planning. I will rename the final 'in' statement when I have finished:



Expansion by Store

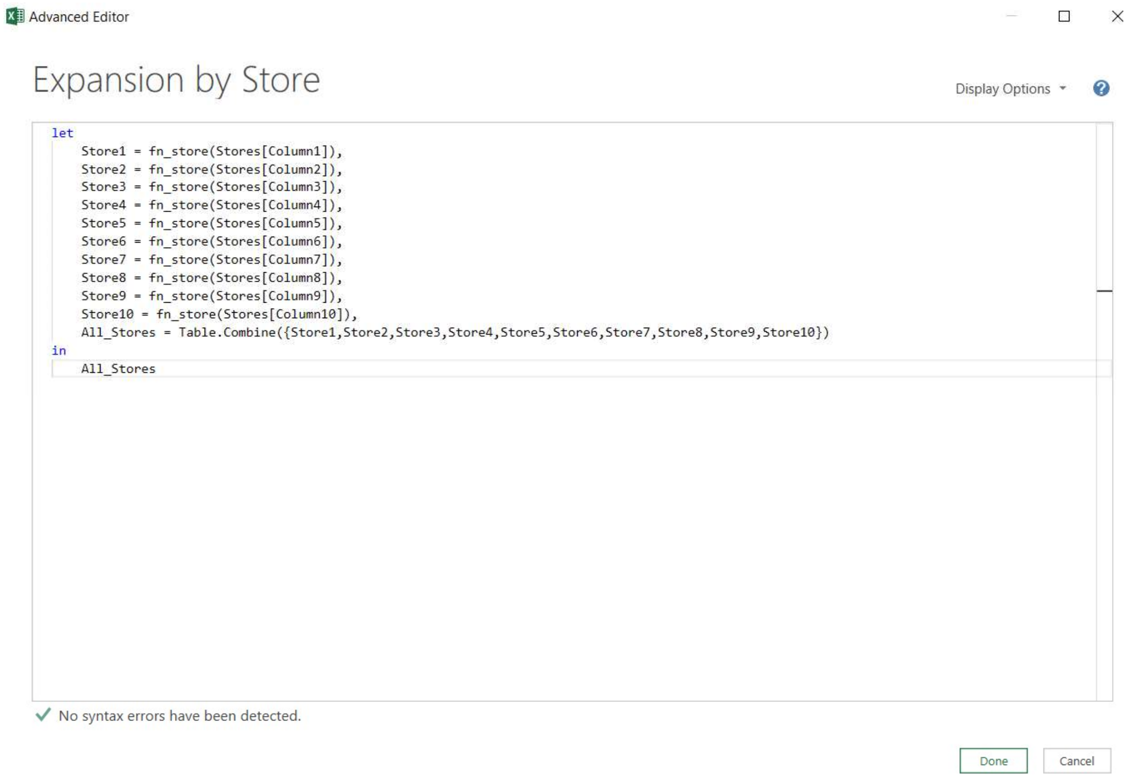
```
let
    Store1 = fn_store(Stores[Column1]),
    Store2 = fn_store(Stores[Column2]),
    Store3 = fn_store(Stores[Column3]),
    Store4 = fn_store(Stores[Column4]),
    Store5 = fn_store(Stores[Column5]),
    Store6 = fn_store(Stores[Column6]),
    Store7 = fn_store(Stores[Column7]),
    Store8 = fn_store(Stores[Column8]),
    Store9 = fn_store(Stores[Column9]),
    Store10 = fn_store(Stores[Column10])
in
    Source
```

✓ No syntax errors have been detected.

I have created a table for each store; now I just need to append them. The **M** code function to append is:

`Table.Combine({table1, table2,….}).`

I add this line to the **M** code in the Advanced Editor.



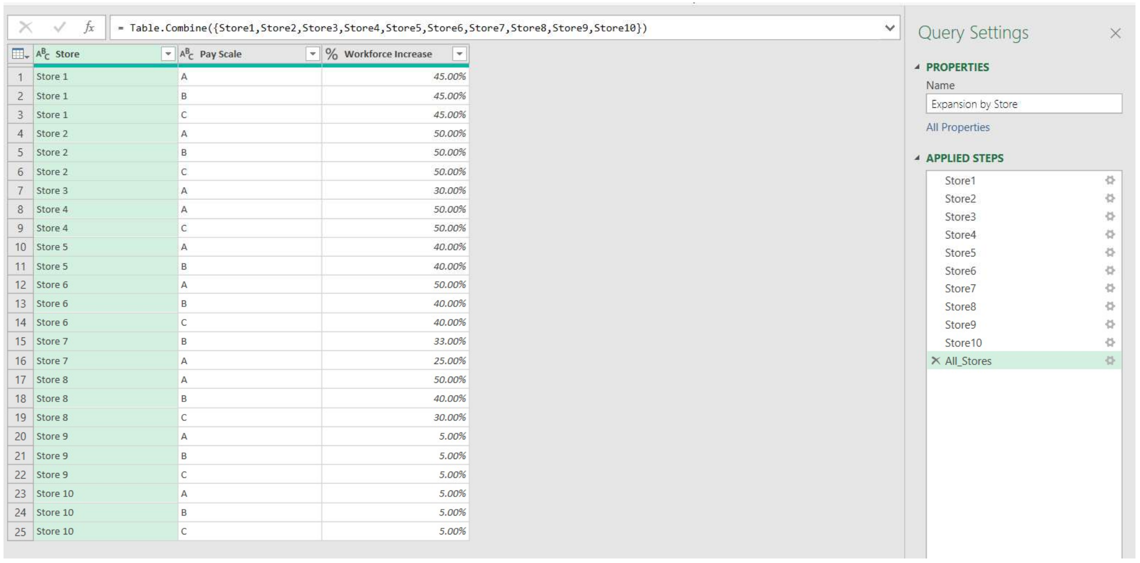
Advanced Editor

Expansion by Store

```
let
    Store1 = fn_store(Stores[Column1]),
    Store2 = fn_store(Stores[Column2]),
    Store3 = fn_store(Stores[Column3]),
    Store4 = fn_store(Stores[Column4]),
    Store5 = fn_store(Stores[Column5]),
    Store6 = fn_store(Stores[Column6]),
    Store7 = fn_store(Stores[Column7]),
    Store8 = fn_store(Stores[Column8]),
    Store9 = fn_store(Stores[Column9]),
    Store10 = fn_store(Stores[Column10]),
    All_Stores = Table.Combine({Store1,Store2,Store3,Store4,Store5,Store6,Store7,Store8,Store9,Store10})
in
    All_Stores
```

No syntax errors have been detected.

I click 'Done' to see the results:



```
= Table.Combine({Store1,Store2,Store3,Store4,Store5,Store6,Store7,Store8,Store9,Store10})
```

	Store	Pay Scale	Workforce Increase
1	Store 1	A	45.00%
2	Store 1	B	45.00%
3	Store 1	C	45.00%
4	Store 2	A	50.00%
5	Store 2	B	50.00%
6	Store 2	C	50.00%
7	Store 3	A	30.00%
8	Store 4	A	50.00%
9	Store 4	C	50.00%
10	Store 5	A	40.00%
11	Store 5	B	40.00%
12	Store 6	A	50.00%
13	Store 6	B	40.00%
14	Store 6	C	40.00%
15	Store 7	B	33.00%
16	Store 7	A	25.00%
17	Store 8	A	50.00%
18	Store 8	B	40.00%
19	Store 8	C	30.00%
20	Store 9	A	5.00%
21	Store 9	B	5.00%
22	Store 9	C	5.00%
23	Store 10	A	5.00%
24	Store 10	B	5.00%
25	Store 10	C	5.00%

Query Settings

PROPERTIES

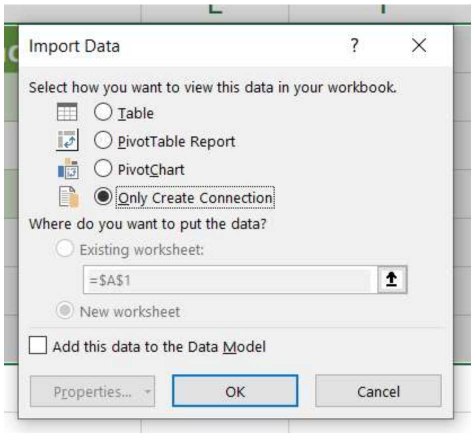
Name: Expansion by Store

All Properties

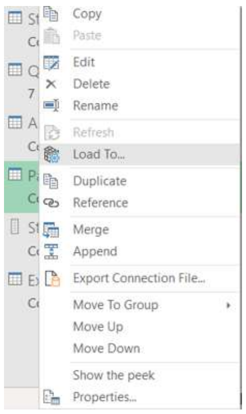
APPLIED STEPS

- Store1
- Store2
- Store3
- Store4
- Store5
- Store6
- Store7
- Store8
- Store9
- Store10
- All_Stores

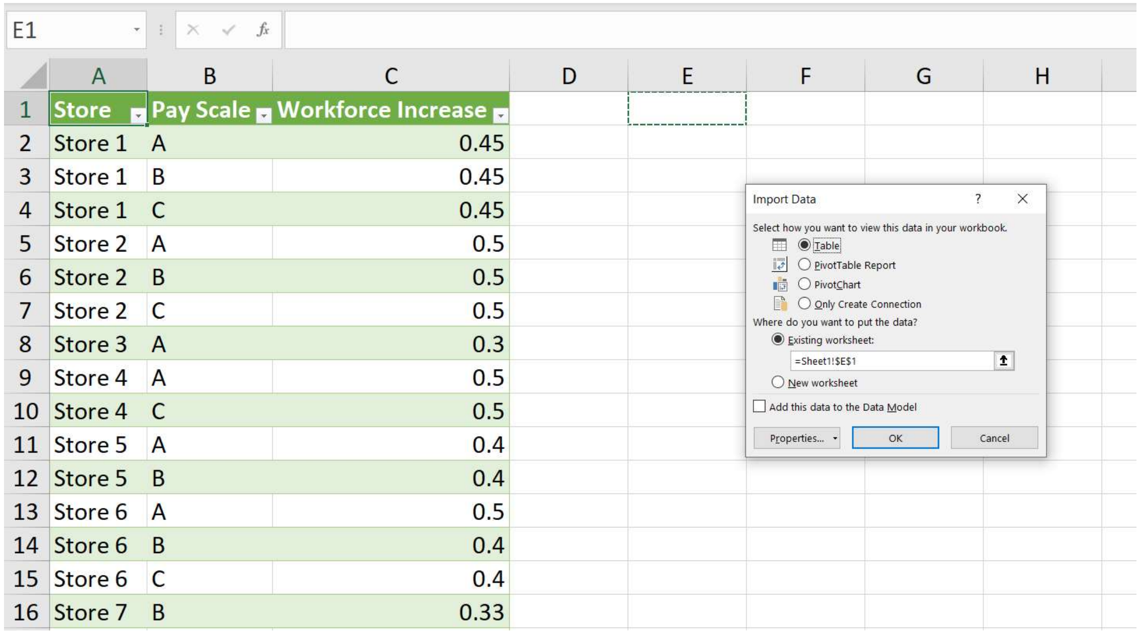
I have the results in the format I wanted for all stores. I can now 'Close & Load'. I will choose 'Close & Load to…' so that I can load to 'Connection Only' to begin with to avoid loading all queries.



I can then right-click on the queries I want to load, and position them together:



I load the tables onto a worksheet:



E1

	A	B	C
1	Store	Pay Scale	Workforce Increase
2	Store 1	A	0.45
3	Store 1	B	0.45
4	Store 1	C	0.45
5	Store 2	A	0.5
6	Store 2	B	0.5
7	Store 2	C	0.5
8	Store 3	A	0.3
9	Store 4	A	0.5
10	Store 4	C	0.5
11	Store 5	A	0.4
12	Store 5	B	0.4
13	Store 6	A	0.5
14	Store 6	B	0.4
15	Store 6	C	0.4
16	Store 7	B	0.33

Import Data

Select how you want to view this data in your workbook.

[✓ Table] PivotTable Report PivotChart Only Create Connection

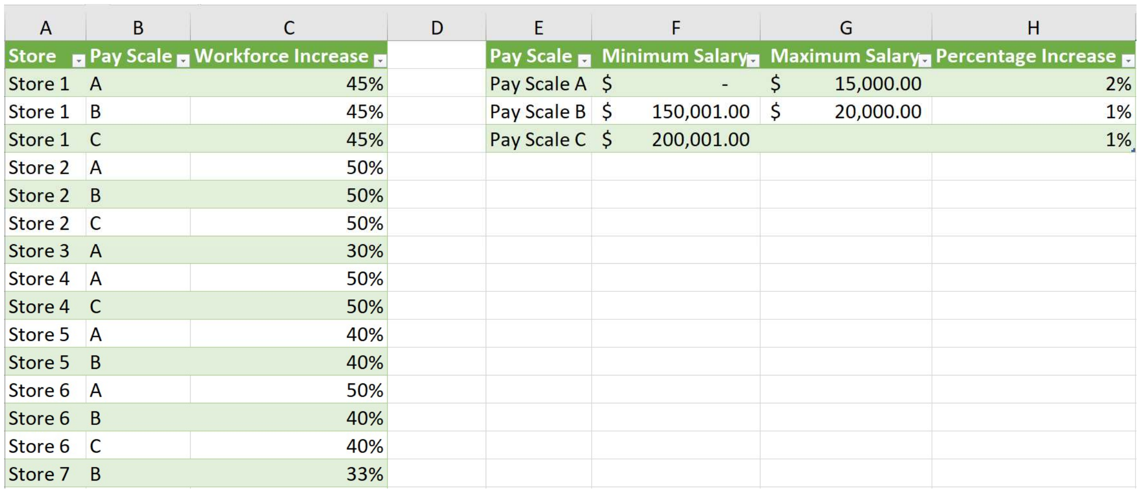
Where do you want to put the data?

[✓ Existing worksheet:] =Sheet1!$E$1

New worksheet

☐ Add this data to the Data Model

I check the data types and the report is ready.



Store	Pay Scale	Workforce Increase
Store 1	A	45%
Store 1	B	45%
Store 1	C	45%
Store 2	A	50%
Store 2	B	50%
Store 2	C	50%
Store 3	A	30%
Store 4	A	50%
Store 4	C	50%
Store 5	A	40%
Store 5	B	40%
Store 6	A	50%
Store 6	B	40%
Store 6	C	40%
Store 7	B	33%

Pay Scale	Minimum Salary	Maximum Salary	Percentage Increase
Pay Scale A	$ -	$ 15,000.00	2%
Pay Scale B	$ 150,001.00	$ 20,000.00	1%
Pay Scale C	$ 200,001.00		1%

Further Reading

The course may consider topics not covered in this document. Moreover, you may wish to learn more. SumProduct presently has weekly blogs on Power Query which may be found here:

https://www.sumproduct.com/blog

Past articles may be located here:

https://www.sumproduct.com/thought/power-query-blog-series

Find out more at contact@sumproduct.com or training@sumproduct.com as required. We are happy to help!

Liam Bastick
+61 421 610 852
liam.bastick@sumproduct.com

November 2024