SumProduct

NEWSLETTER #119 - October 2022

www.sumproduct.com | www.sumproduct.com/thought

Excel Virtually Global 2022 is almost upon us

Have you checked out the website yet? If not, read all about it *(below)*.

Power Query gets another step closer with Excel for Mac and we fill you in on this important update below too, together with Excel's new **IMAGE** function.

As always, that's not all.

We also bring you another Beat the Boredom Challenge to keep you on your toes, together with Charts & Dashboards, Visual Basics, Power Pivot Principles, Power Query Pointers, Power BI Updates and Keyboard Shortcuts. Our A to Z of Excel Functions also details several key functions we use everyday in our work (such as **INFO** – er, maybe not – but **INDEX** and **INDIRECT** are certainly key topics for any modeller looking to make the most of their toolset).

As always, happy reading and remember: stay safe, stay happy, stay healthy.

Liam Bastick, Managing Director, SumProduct

Excel Virtually Global 2022

Not long to go now… Just a reminder, this 30+ hour virtual conference, for charity, presents many Excel, Data Platform and PowerPoint MVPs, together with other acknowledged experts from Microsoft and / or around the globe to present, answer questions and demonstrate the future of Excel and how it will make your life easier, personally and professionally.

Topics will include all things Excel, financial modelling, dynamic arrays, Power Pivot, Power Query and Power BI. It's not all in English either, with some sessions in Portuguese and Spanish too (at the time of writing).

Each session (including Q&A) will last no more than an hour and topics will cover all expertise levels, from novice to expert. Most presenters are well known in their spheres, and have written blogs, books and articles and / or present video sessions.

Most sessions will be recorded so you may watch them later with downloads aplenty – there are no medals for staying up to watch the entire event live! That's just as well, as it will run from midnight GMT / UTC on Monday 17 October into some time on Tuesday 18 October.

From your own favourite chair, bring a laptop, an inquisitive mind and your sense of humour. Be prepared to learn heaps. And remember, it's for charity – this year, there is no ticketing process: all we ask is an "honesty box" where you donate to your favourite charity.

For more details (*e.g.* times, speakers, sessions), please go to www.excelvirtuallyglobal.com. The links will go live nearer the time, so keep checking back as the program may change too.

Hopefully, we'll see you there!

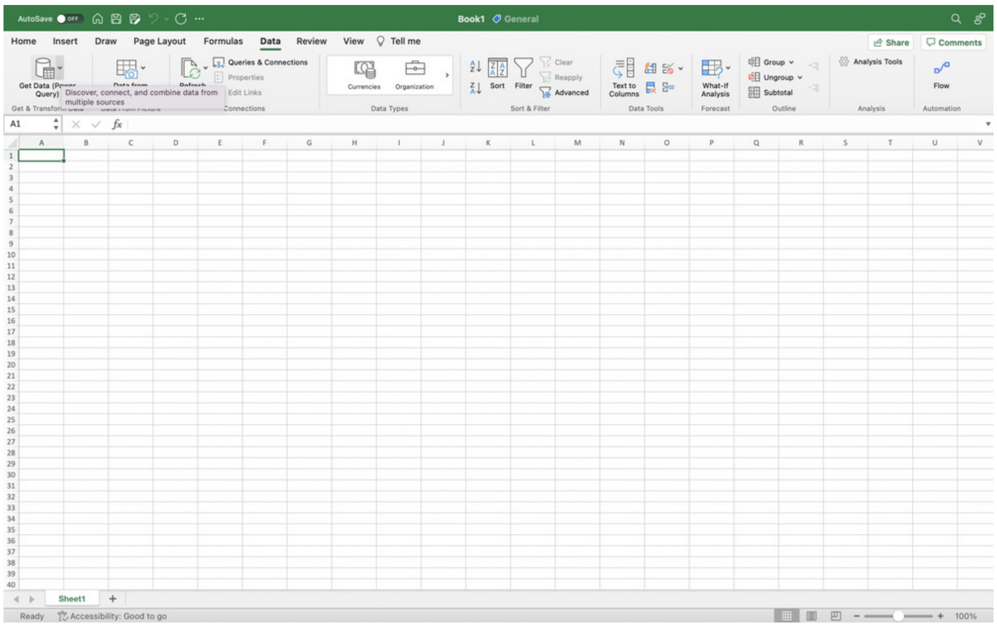
Shaping Data with the Power Query Editor in Excel for Mac

This feature is available to Insiders using either the Current Channel Preview or Beta Channel, running Version 16.64 (Build 22072501) or later. When we previously mentioned this back in the June newsletter, it was only available to Insiders on the Beta Channel.

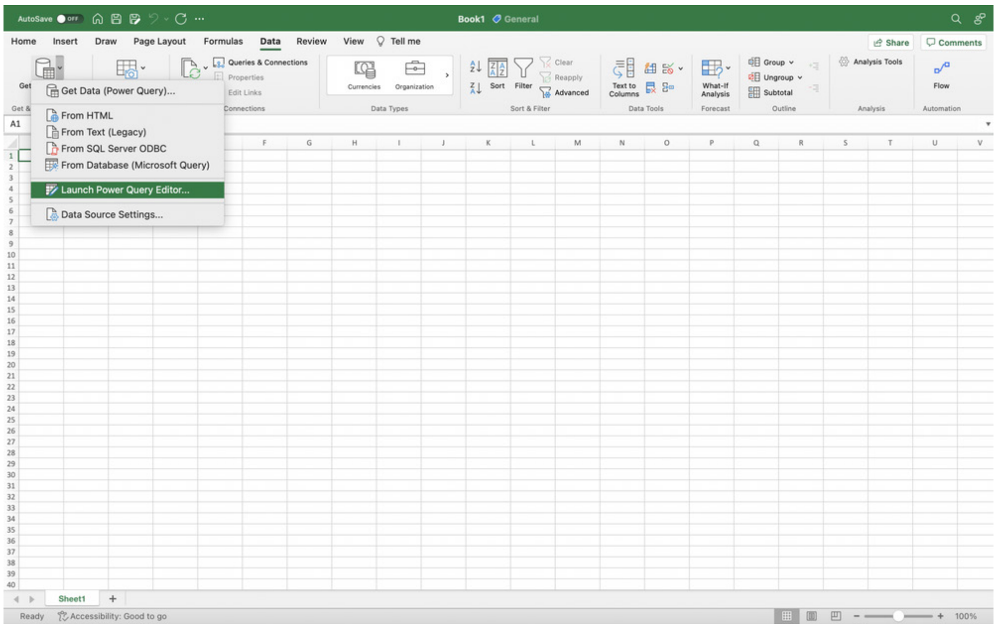
When Power Query was first released in Excel for Mac, there was hope that it wouldn't take long for the features to mirror their Window counterparts. It's taking time, but the capabilities are improving. Beginning with the ability to refresh data two years ago, importing data was introduced this time last year, so this does seem to be an annual event! As you may imagine, Mac users have desperately requested the ability to transform data using the Query Editor, which would provide users with the full Power Query experience in Excel for Mac.

Well… it's getting there! You may now clean and shape your data with hundreds of transformations available in Power Query Editor in Excel for Mac.

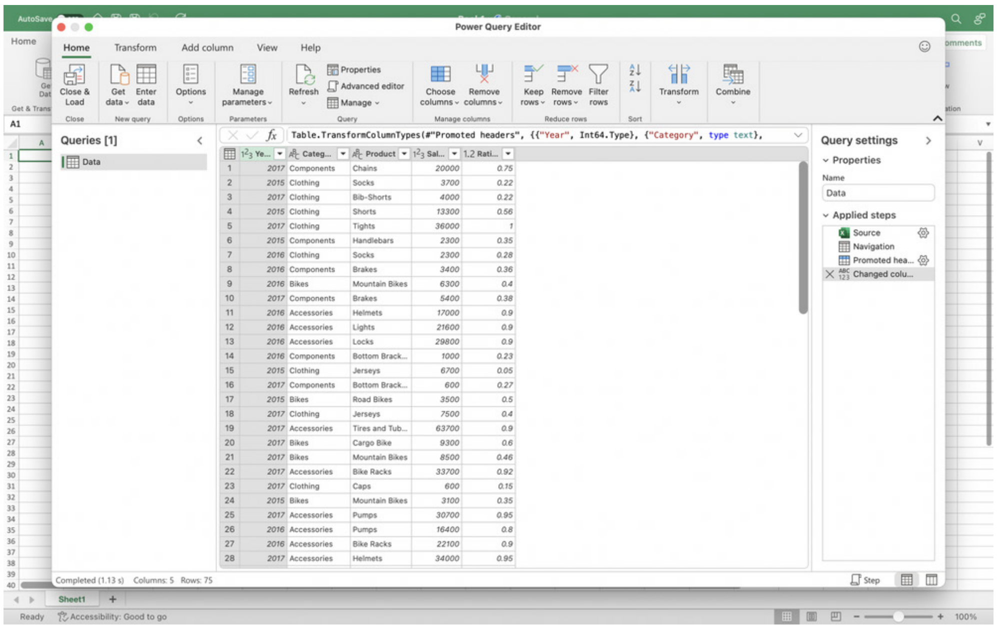
To access, on the Data tab, click the Get Data (Power Query) button.



Click Launch Power Query Editor to open the Query Editor.


Get Data (Power Query)...
From HTML
From Text (Legacy)
From SQL Server ODBC
From Database (Microsoft Query)
Launch Power Query Editor...
Data Source Settings...

You can shape and transform your data using the Query Editor similarly to Excel for Windows. When you're done, click the 'Close & Load' button on the Home tab.


Power Query Editor

```
Table.TransformColumnTypes(#"Promoted headers", {{"Year", Int64.Type}, {"Category", type text},
```

	Ye...	Categ...	Product	Sal...	Rati...
1	2017	Components	Chains	20000	0.75
2	2015	Clothing	Socks	3700	0.22
3	2017	Clothing	Bib-Shorts	4000	0.22
4	2015	Clothing	Shorts	13300	0.56
5	2017	Clothing	Tights	36000	1
6	2015	Components	Handlebars	2300	0.35
7	2016	Clothing	Socks	2300	0.28
8	2016	Components	Brakes	3400	0.36
9	2016	Bikes	Mountain Bikes	6300	0.4
10	2017	Components	Brakes	5400	0.38
11	2016	Accessories	Helmets	17000	0.9
12	2016	Accessories	Lights	21600	0.9
13	2016	Accessories	Locks	29800	0.9
14	2016	Components	Bottom Brack...	1000	0.23
15	2015	Clothing	Jerseys	6700	0.05
16	2017	Components	Bottom Brack...	600	0.27
17	2015	Bikes	Road Bikes	3500	0.5
18	2017	Clothing	Jerseys	7500	0.4
19	2017	Accessories	Tires and Tub...	63700	0.9
20	2017	Bikes	Cargo Bike	9300	0.6
21	2017	Bikes	Mountain Bikes	8500	0.46
22	2017	Accessories	Bike Racks	33700	0.92
23	2017	Clothing	Caps	600	0.15
24	2015	Bikes	Mountain Bikes	3100	0.35
25	2017	Accessories	Pumps	30700	0.95
26	2016	Accessories	Pumps	16400	0.8
27	2016	Accessories	Bike Racks	22100	0.9
28	2017	Accessories	Helmets	34000	0.95

Queries [1]: Data

Query settings — Properties — Name: Data — Applied steps: Source, Navigation, Promoted hea..., Changed colu...

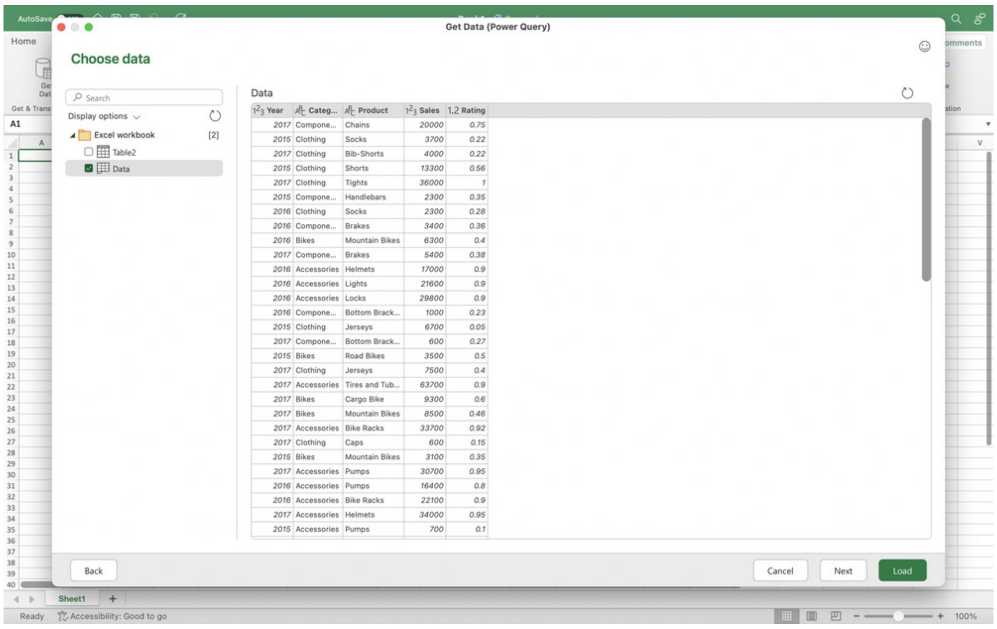
Completed (1.13 s) Columns: 5 Rows: 75

The newly imported data appears in a new sheet.

Supported data sources include:

- text, CSV, XLSX, XML and JSON files
- SharePoint, SharePoint Lists, SharePoint Folders and OData
- local tables, Tables and ranges
- Microsoft SQL Server.

You may also access the Query Editor from the data import flow by clicking the Get Data (Power Query) button, choosing a data source, and clicking the Next button.


Get Data (Power Query)

Choose data

Excel workbook [2]: Table2, Data

Data

Year	Categ...	Product	Sales	Rating
2017	Compone...	Chains	20000	0.75
2015	Clothing	Socks	3700	0.22
2017	Clothing	Bib-Shorts	4000	0.22
2015	Clothing	Shorts	13300	0.56
2017	Clothing	Tights	36000	1
2015	Compone...	Handlebars	2300	0.35
2016	Clothing	Socks	2300	0.28
2016	Compone...	Brakes	3400	0.36
2016	Bikes	Mountain Bikes	6300	0.4
2017	Compone...	Brakes	5400	0.38
2016	Accessories	Helmets	17000	0.9
2016	Accessories	Lights	21600	0.9
2016	Accessories	Locks	29800	0.9
2016	Compone...	Bottom Brack...	1000	0.23
2015	Clothing	Jerseys	6700	0.05
2017	Compone...	Bottom Brack...	600	0.27
2015	Bikes	Road Bikes	3500	0.5
2017	Clothing	Jerseys	7500	0.4
2017	Accessories	Tires and Tub...	63700	0.9
2017	Bikes	Cargo Bike	9300	0.6
2017	Bikes	Mountain Bikes	8500	0.46
2017	Accessories	Bike Racks	33700	0.92
2017	Clothing	Caps	600	0.15
2015	Bikes	Mountain Bikes	3100	0.35
2017	Accessories	Pumps	30700	0.95
2016	Accessories	Pumps	16400	0.8
2016	Accessories	Bike Racks	22100	0.9
2017	Accessories	Helmets	34000	0.95
2015	Accessories	Pumps	700	0.1

Beat the Boredom Challenge

With many of us currently "working from home" / quarantined, there are only so Zoom / Teams calls and virtual parties you can make before you reach your (data) limit. Perhaps they should measure data allowance in blood pressure millimetres of mercury (mmHg). To try and keep our readers engaged, we will continue to reproduce some of our popular ***Final Friday Fix*** *challenges from yesteryear in this and upcoming newsletters. One suggested solution may be found later in this newsletter. Here's this month's…*

…eno siht rof esrever otni niarb ruoy tup ot deen thgim uoY

Nearly two years ago, the Microsoft Research Blog discussed LAMBDA. Highlighting the power of this Office 365 function, they wrote the following:

> Reversing a string is beyond the built-in functions of Excel and could only previously be written outside the formula language, by using Visual Basic or JavaScript. Here is a definition of REVERSE as a recursive LAMBDA, which makes use of a couple of auxiliary functions—HEAD and TAIL—to compute the first character and everything but the first character, respectively.

Aside from the fact that there is a mistake in the formula for **REVERSE**, having one too many closed brackets (they really should use **FORMULATEXT!**), the Research Team has erroneously stated that "…reversing a string is beyond the built-in functions of Excel and could only previously *(sic)* be written outside the formula language…".

Spurred on by several of our readers, we vehemently disagree; so here's this month's challenge:

For All Versions of Excel	Mary Had A Little LAMBDA	ADBMAL elttiL A daH yraM

For **all** current versions of Excel (not just Office 365, Excel on the web and Insider / Beta variants), write a formula that will reverse the text in a cell (*i.e.* reverse the text string). You cannot use VBA, dynamic arrays, JavaScript, TypeScript or Office Scripts.

Sound easy? Try it. One solution just might be found later in this newsletter – but no reading ahead!

Charts and Dashboards

It's time to chart our progress with an introductory series into the world of creating charts and dashboards in Excel. This month, we look at PivotCharts.

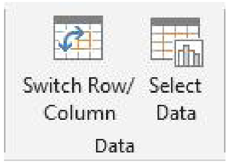
PivotCharts are complementary visual representations of data in PivotTables. They display any data currently in a PivotTable in a chart, with that they also inherit the flexibility and interactivity of PivotTables.

When a PivotChart is created, it will inherit the filters and sort abilities that are also available in the associated PivotTable. Changes made in the PivotChart will also be reflected in the PivotTable and *vice versa*.

The key benefit for a PivotChart is that it is able to summarise raw data or tabular data, i.e. data that has not been analysed whatsoever. We would be able to use a PivotChart just as we would a PivotTable to analyse data.

Below are a few differences between PivotCharts and ordinary charts:

- **row / column orientation:** switching the rows and columns around in a PivotChart cannot be performed through the 'Switch Row / Column' button on the Ribbon

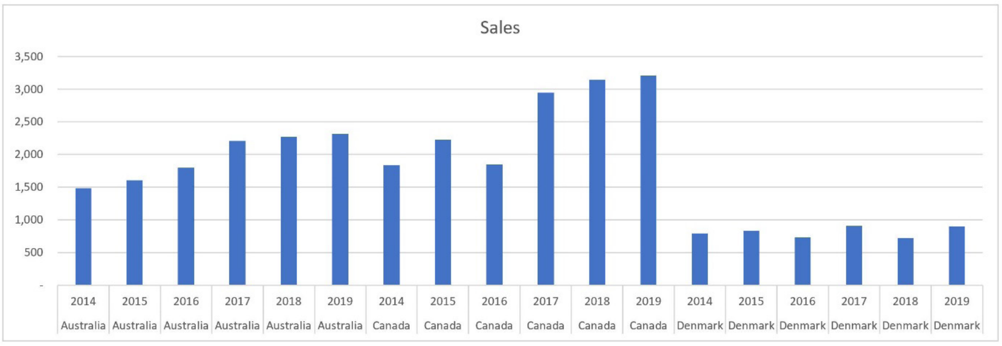
  

  It has to be undertaken through switching the fields from the Rows and Columns area in the PivotChart Fields pane
- **Chart Type limitations:** PivotCharts cannot be produced for XY (scatter charts), Stock or Bubble charts
- **source data:** PivotCharts draw data from their underlying PivotTable's data source, whereas standard charts are linked directly to worksheet cells
- **formatting:** most formatting elements in a PivotChart are preserved when a PivotChart is refreshed. However, do be wary that trendlines, error bars, trendlines and changes to data sets are not preserved.

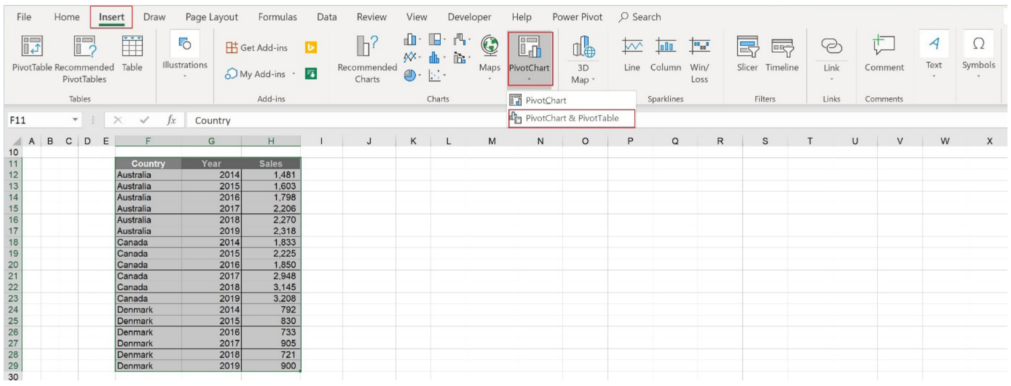
Let's move on to an example. Imagine creating a Bar chart with the following data:

Country	Year	Sales
Australia	2014	1,481
Australia	2015	1,603
Australia	2016	1,798
Australia	2017	2,206
Australia	2018	2,270
Australia	2019	2,318
Canada	2014	1,833
Canada	2015	2,225
Canada	2016	1,850
Canada	2017	2,948
Canada	2018	3,145
Canada	2019	3,208
Denmark	2014	792
Denmark	2015	830
Denmark	2016	733
Denmark	2017	905
Denmark	2018	721
Denmark	2019	900

We may create the following chart with the data above:

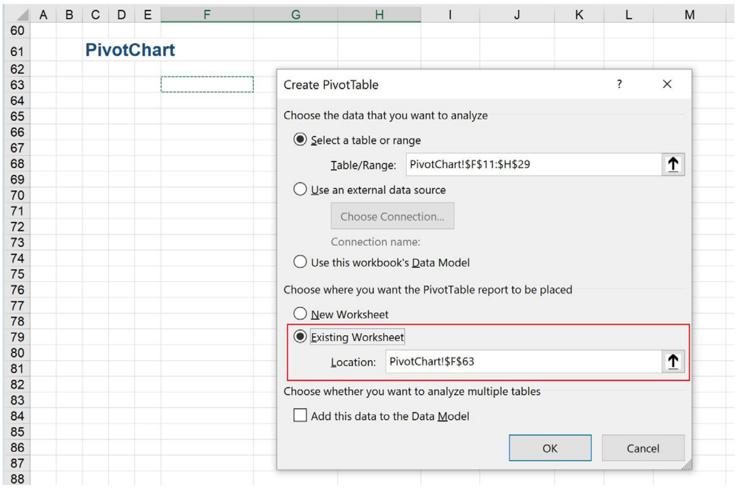


Let’s try again with a PivotChart. To create a PivotChart, highlight all the data and go to **Insert –> PivotChart –> PivotChart & PivotTable**.



Country	Year	Sales
Australia	2014	1,481
Australia	2015	1,603
Australia	2016	1,798
Australia	2017	2,206
Australia	2018	2,270
Australia	2019	2,318
Canada	2014	1,833
Canada	2015	2,225
Canada	2016	1,850
Canada	2017	2,948
Canada	2018	3,145
Canada	2019	3,208
Denmark	2014	792
Denmark	2015	830
Denmark	2016	733
Denmark	2017	905
Denmark	2018	721
Denmark	2019	900

A ‘Create PivotTable’ dialog will appear. Here, we may choose where to place the PivotTable report, either into a new worksheet or in the existing worksheet. In this example, the PivotTable report will be loaded to cell **F63** in the current worksheet:



Create PivotTable

Choose the data that you want to analyze

[✓ Select a table or range]

Table/Range: PivotChart!$F$11:$H$29

Use an external data source

Connection name:

Use this workbook's Data Model

Choose where you want the PivotTable report to be placed

New Worksheet

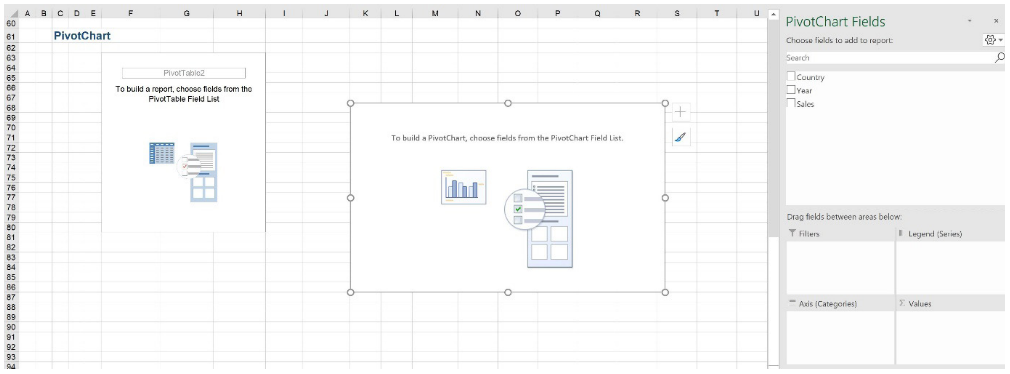
[✓ Existing Worksheet]

Location: PivotChart!$F$63

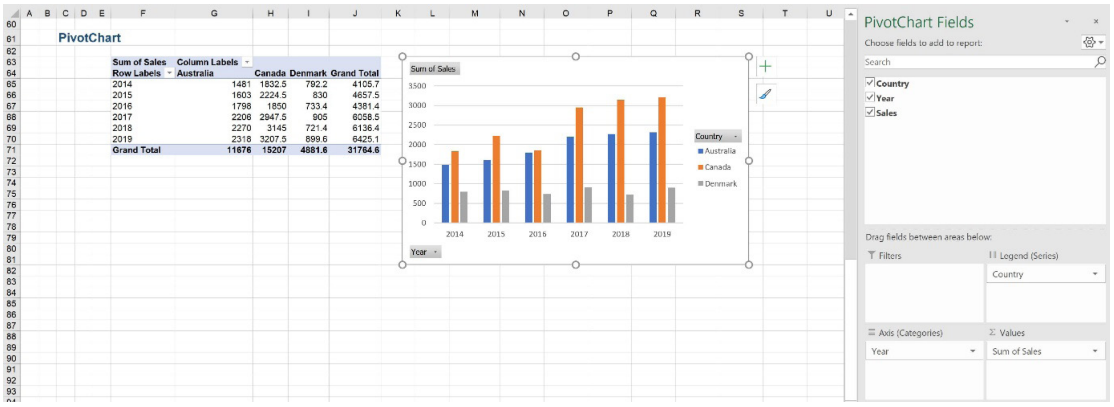
Choose whether you want to analyze multiple tables

Add this data to the Data Model

Clicking OK allows the PivotChart interface to be loaded:


PivotChart
PivotTable2
To build a report, choose fields from the PivotTable Field List
To build a PivotChart, choose fields from the PivotChart Field List.
PivotChart Fields
Choose fields to add to report:
Search
Country
Year
Sales
Drag fields between areas below:
Filters
Legend (Series)
Axis (Categories)
Values

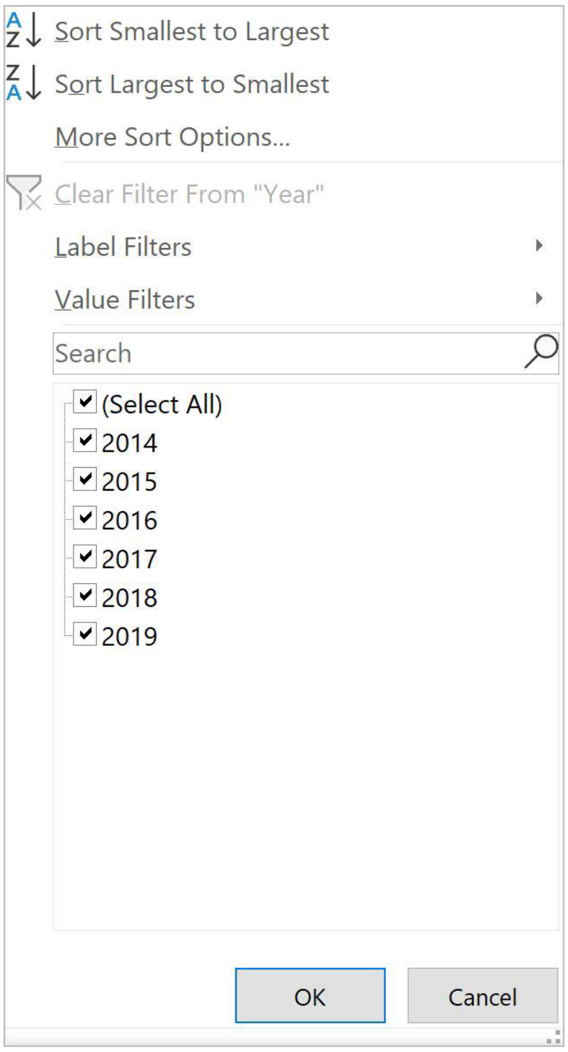
Dragging the fields to appropriate areas will result in an initial PivotChart and PivotTable being displayed:

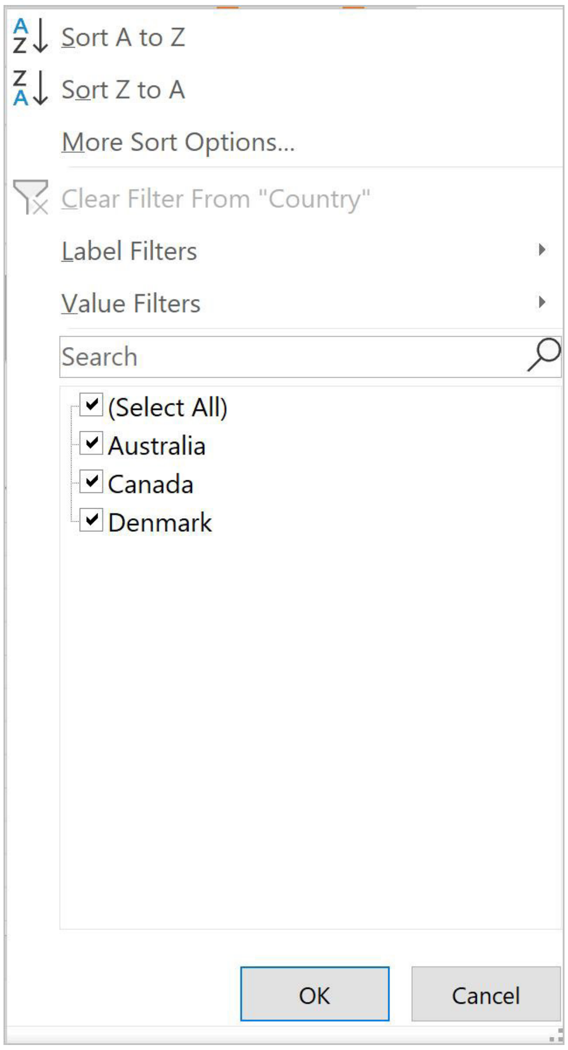


Sum of Sales	Column Labels			
Row Labels	Australia	Canada	Denmark	Grand Total
2014	1481	1832.5	792.2	4105.7
2015	1603	2224.5	830	4657.5
2016	1798	1850	733.4	4381.4
2017	2206	2947.5	905	6058.5
2018	2270	3145	721.4	6136.4
2019	2318	3207.5	899.6	6425.1
Grand Total	**11676**	**15207**	**4881.6**	**31764.6**

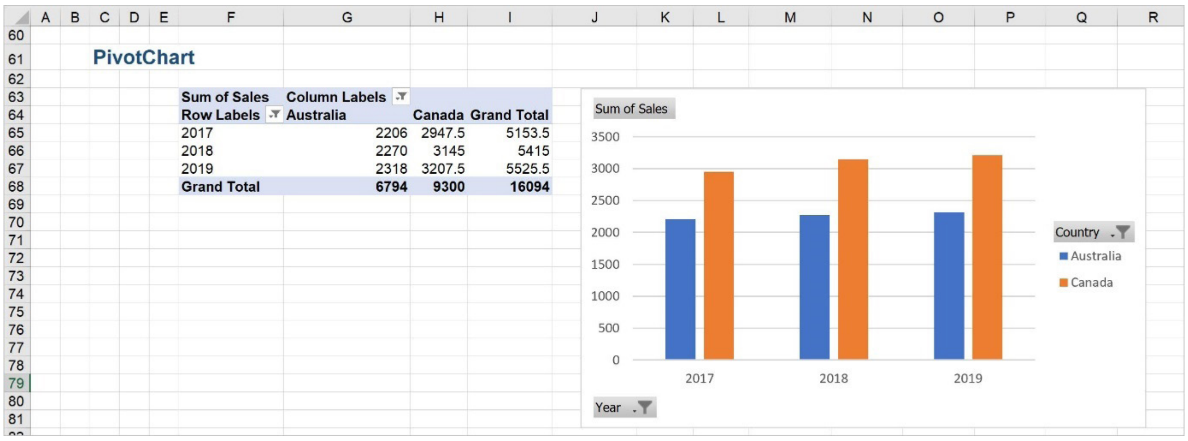
The PivotChart is able to produce a visual that readily summarises the data in an easily understandable way with few manual adjustments.

PivotCharts allow interactivity within the chart. For example, we may apply filters to the data by clicking on the downwards arrow next to ‘Year’ and ‘Country’ fields:





These filters will also apply to the PivotTable linked to the PivotChart. For example, if we choose to see only Australia's and Canada's sales in three most recent years reported (2017, 2018 and 2019), both the PivotChart and the related PivotTable are adjusted based upon the new filters. Further, note the filter icons in the 'Year' and 'Country' fields of PivotChart and PivotTable:



PivotChart

Sum of Sales	Column Labels		
Row Labels	Australia	Canada	Grand Total
2017	2206	2947.5	5153.5
2018	2270	3145	5415
2019	2318	3207.5	5525.5
Grand Total	6794	9300	16094

More next month…

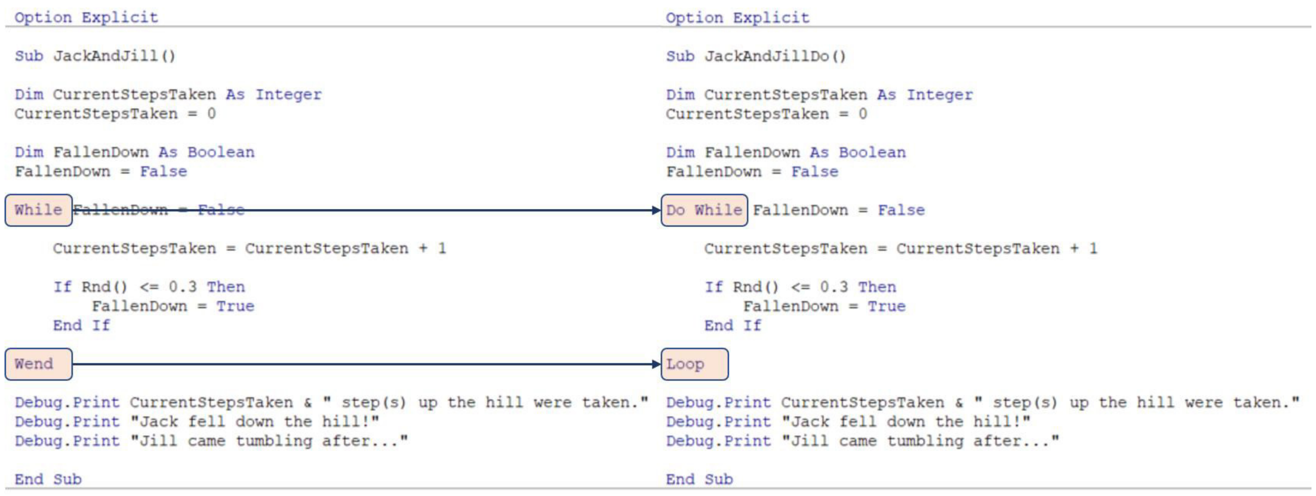
Visual Basics

We thought we'd run an elementary series going through the rudiments of Visual Basic for Applications (VBA) as a springboard for newer users. What should we 'Do' this month?

In last month's newsletter, we **While Wend**-ed down a loopy path, but now it is time to find better things to **Do**. **Do...Loop** loops are considered the upgraded alternative to **While Wend**. Let's have a look at how they work:

```
Do [{ While |Until } condition ]
[ statements ]
[ Exit Do ]
[ statements]
Loop
```

So how would the code change from a **While Wend** to a **Do...Loop** ? Simply replace the **While** with **Do While** and **Wend** with **Loop**.



```
Option Explicit

Sub JackAndJill()

Dim CurrentStepsTaken As Integer
CurrentStepsTaken = 0

Dim FallenDown As Boolean
FallenDown = False

While ~~FallenDown = False~~

    CurrentStepsTaken = CurrentStepsTaken + 1

    If Rnd() <= 0.3 Then
        FallenDown = True
    End If

Wend

Debug.Print CurrentStepsTaken & " step(s) up the hill were taken."
Debug.Print "Jack fell down the hill!"
Debug.Print "Jill came tumbling after..."

End Sub
```

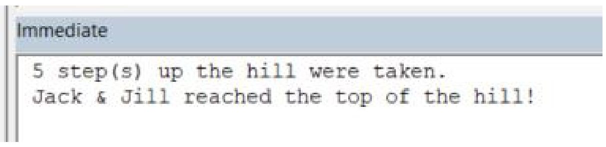
```
Option Explicit

Sub JackAndJillDo()

Dim CurrentStepsTaken As Integer
CurrentStepsTaken = 0

Dim FallenDown As Boolean
FallenDown = False

Do While FallenDown = False

    CurrentStepsTaken = CurrentStepsTaken + 1

    If Rnd() <= 0.3 Then
        FallenDown = True
    End If

Loop

Debug.Print CurrentStepsTaken & " step(s) up the hill were taken."
Debug.Print "Jack fell down the hill!"
Debug.Print "Jill came tumbling after..."

End Sub
```

Do...Loop is superior to **While Wend** for a multitude of reasons.

For one thing, **While Wend** has no ability to have an **Exit**. Let's return to our Jack and Jill example. What if the hill has five [5] steps? The condition for the loop could be altered to check **CurrentStepsTaken** before entering, but the **Do...Loop** has the flexibility to break out with an **Exit** statement. Here, the subroutine can check if Jack and Jill fall down after they take a step. I f they don't fall down when they reach the top of the hill, then the print statements need to be changed accordingly:

```
Sub JackAndJillExit()
Dim CurrentStepsTaken As Integer
CurrentStepsTaken = 0
Dim FallenDown As Boolean
FallenDown = False
Do While FallenDown = False
    CurrentStepsTaken = CurrentStepsTaken + 1
    If Rnd() <= 0.3 Then
        FallenDown = True
    End If
    If CurrentStepsTaken = 5 Then
        Exit Do
    End If
Loop
Debug.Print CurrentStepsTaken & " step(s) up the hill were taken."
If FallenDown Then
    Debug.Print "Jack fell down the hill!"
    Debug.Print "Jill came tumbling after..."
Else
    Debug.Print "Jack & Jill reached the top of the hill!"
End If
End Sub
```

This provides the following result:



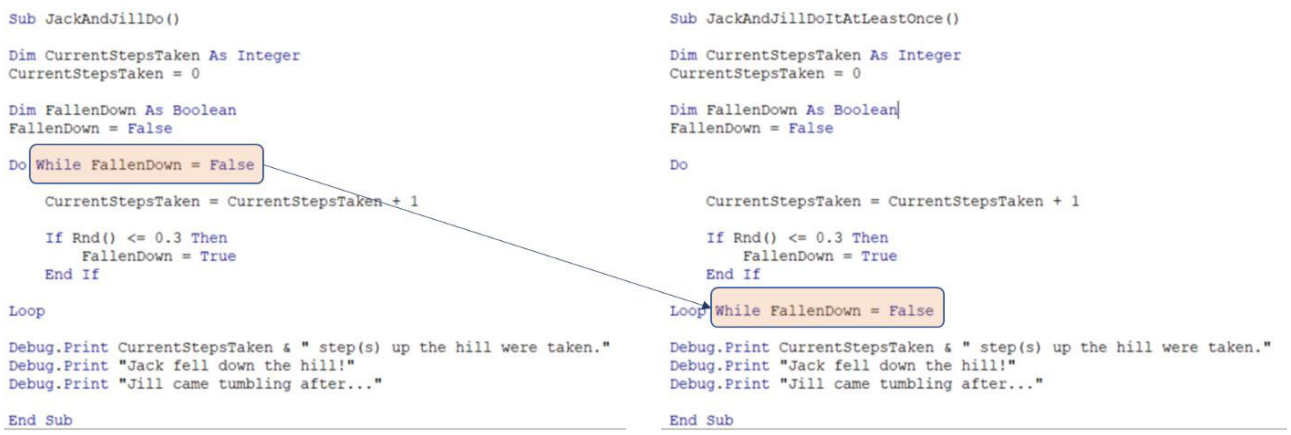
```
5 step(s) up the hill were taken.
Jack & Jill reached the top of the hill!
```

They didn't fall down when they reached the top, but because they had reached the top, we asked the program to exit as reaching the top was not part of our loop conditional expression.

The second reason that **Do…Loop** is superior is that **While Wend** loops check for the condition prior to running, whereas with **Do…Loop** the condition may be checked at the end. So you might be thinking, why would you want to test the condition at the end?

For an example, let's consider Jack and Jill will always take at least one step onto the hill before falling. They won't fall if they haven't moved! How this is written in code is by simply moving the "**While** condition" part of the **Do** statement next to **Loop**. The syntax changes to:

```
Do
[ statements ]
[ Exit Do ]
[ statements]
Loop [{ While |Until } condition ]
```



```
Sub JackAndJillDo()

Dim CurrentStepsTaken As Integer
CurrentStepsTaken = 0

Dim FallenDown As Boolean
FallenDown = False

Do While FallenDown = False

    CurrentStepsTaken = CurrentStepsTaken + 1

    If Rnd() <= 0.3 Then
        FallenDown = True
    End If

Loop

Debug.Print CurrentStepsTaken & " step(s) up the hill were taken."
Debug.Print "Jack fell down the hill!"
Debug.Print "Jill came tumbling after..."

End Sub
```

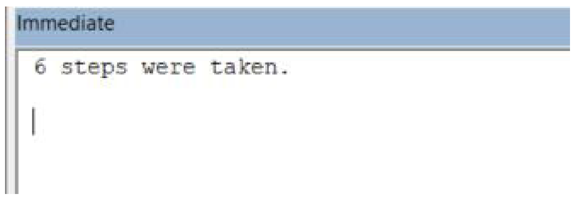
```
Sub JackAndJillDoItAtLeastOnce()

Dim CurrentStepsTaken As Integer
CurrentStepsTaken = 0

Dim FallenDown As Boolean
FallenDown = False

Do

    CurrentStepsTaken = CurrentStepsTaken + 1

    If Rnd() <= 0.3 Then
        FallenDown = True
    End If

Loop While FallenDown = False

Debug.Print CurrentStepsTaken & " step(s) up the hill were taken."
Debug.Print "Jack fell down the hill!"
Debug.Print "Jill came tumbling after..."

End Sub
```

This will execute the statements within the loop to run at least once, before checking the condition and exiting if applicable.

Imagine we have Jack and Jill sitting pretty on the top of the hill having already taken five steps. Let's set them on a loop walking, despite having reached the top:

```
Sub DoItAtLeastOnce()
Dim CurrentStepsTaken As Integer
CurrentStepsTaken = 5
Do
    CurrentStepsTaken = CurrentStepsTaken + 1
Loop While CurrentStepsTaken < 5
Debug.Print CurrentStepsTaken & " steps were taken."
End Sub
```

This results in:

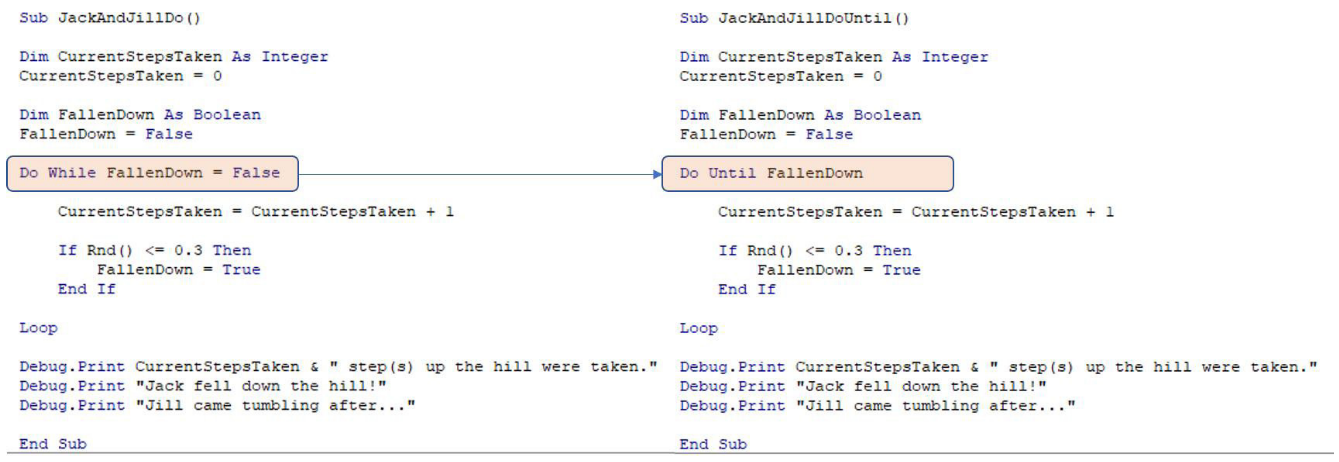


```
6 steps were taken.
```

The third and final reason consideration of **Do…Loop** being superior to **While Wend** is that we have the option of **Until**. What effect does this achieve? This essentially reverses the value of the condition to be tested.

While executes the block of code when the condition is true and keeps executing that until the condition becomes false. Once the condition becomes false, the loop is terminated. **Until** does the opposite. It executes the block of code when the condition is false and keep executing that until the condition becomes true. Again, once the condition becomes true, the until loop is terminated.

In our Jack and Jill example, the condition is that Jack and Jill have not fallen down. Our variable that determines that **FallenDown** = False in order to use **Do While**. Knowing that changing the keyword While to Until essentially flips the Boolean test condition requirement, we would simply check against the condition **FallenDown** = True. However, do note that our **FallenDown** variable is already a Boolean type. We don't even need to use the condition **FallenDown** = True because it is superfluous. We can convert our original loop as follows:



```
Sub JackAndJillDo()

Dim CurrentStepsTaken As Integer
CurrentStepsTaken = 0

Dim FallenDown As Boolean
FallenDown = False

Do While FallenDown = False

    CurrentStepsTaken = CurrentStepsTaken + 1

    If Rnd() <= 0.3 Then
        FallenDown = True
    End If

Loop

Debug.Print CurrentStepsTaken & " step(s) up the hill were taken."
Debug.Print "Jack fell down the hill!"
Debug.Print "Jill came tumbling after..."

End Sub
```

```
Sub JackAndJillDoUntil()

Dim CurrentStepsTaken As Integer
CurrentStepsTaken = 0

Dim FallenDown As Boolean
FallenDown = False

Do Until FallenDown

    CurrentStepsTaken = CurrentStepsTaken + 1

    If Rnd() <= 0.3 Then
        FallenDown = True
    End If

Loop

Debug.Print CurrentStepsTaken & " step(s) up the hill were taken."
Debug.Print "Jack fell down the hill!"
Debug.Print "Jill came tumbling after..."

End Sub
```

More next time.

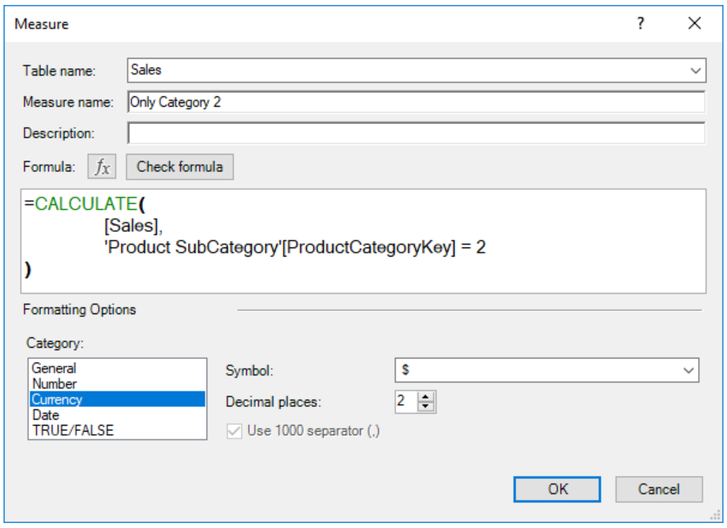
Power Pivot Principles

We continue our series on the Excel COM add-in, Power Pivot. This month, we consider how to stop the ***CALCULATE*** *function from overwriting row filters.*

Let's say we wish to create a measure to calculate the total sales of all products classified as Product Category 2.

Let’s try this with the **CALCULATE** function:

```
=CALCULATE(
        [Sales],
        'Product SubCategory'[ProductCategoryKey] = 2
)
```



Measure

Table name: Sales

Measure name: Only Category 2

Description:

Formula:

```
=CALCULATE(
            [Sales],
            'Product SubCategory'[ProductCategoryKey] = 2
)
```

Formatting Options

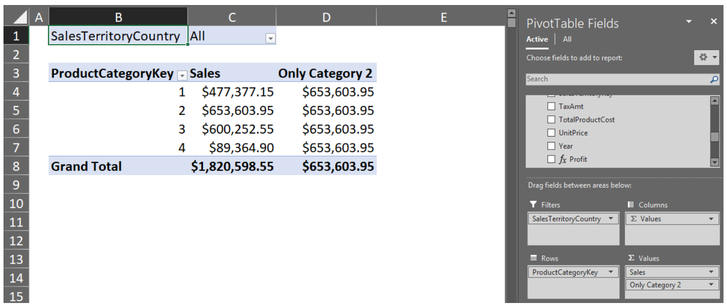
Category: Currency

Symbol: $

Decimal places: 2

Use 1000 separator (,)

The **Only Category 2** measure yields:



SalesTerritoryCountry | All

ProductCategoryKey	Sales	Only Category 2
1	$477,377.15	$653,603.95
2	$653,603.95	$653,603.95
3	$600,252.55	$653,603.95
4	$89,364.90	$653,603.95
Grand Total	$1,820,598.55	$653,603.95

PivotTable Fields

Filters: SalesTerritoryCountry

Columns: Σ Values

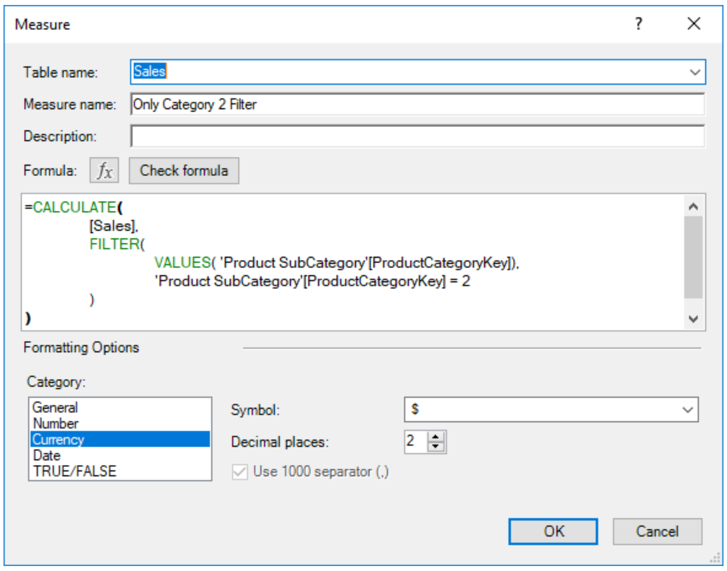
Rows: ProductCategoryKey

Values: Sales, Only Category 2

Active
All
Choose fields to add to report:
Search
TaxAmt
TotalProductCost
UnitPrice
Year
Profit
Drag fields between areas below:

In this case, the **CALCULATE** measure overrides the row context in cells **C4:C7** to be **ProductCategoryKey** = 2 instead of their respective values (*i.e.* 1, 2, 3 or 4). Let’s try adding the VALUES function into a measure:

```
=CALCULATE(
    [Sales],
    FILTER(
            VALUES( 'Product SubCategory'[ProductCategoryKey]),
            'Product SubCategory'[ProductCategoryKey] = 2
    )
)
```



Measure

Table name: Sales

Measure name: Only Category 2 Filter

Description:

Formula:

```
=CALCULATE(
            [Sales],
            FILTER(
                        VALUES( 'Product SubCategory'[ProductCategoryKey]),
                        'Product SubCategory'[ProductCategoryKey] = 2
            )
)
```

Formatting Options

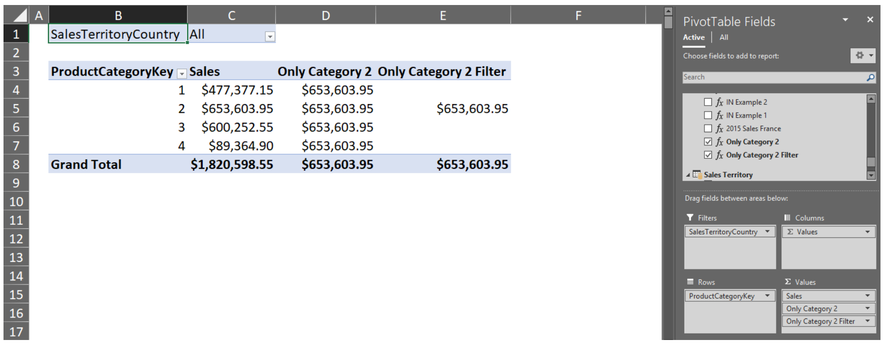
Category: Currency

Symbol: $

Decimal places: 2

Use 1000 separator (,)

This measure yields:



SalesTerritoryCountry: All

ProductCategoryKey	Sales	Only Category 2	Only Category 2 Filter
1	$477,377.15	$653,603.95	
2	$653,603.95	$653,603.95	$653,603.95
3	$600,252.55	$653,603.95	
4	$89,364.90	$653,603.95	
Grand Total	**$1,820,598.55**	**$653,603.95**	**$653,603.95**

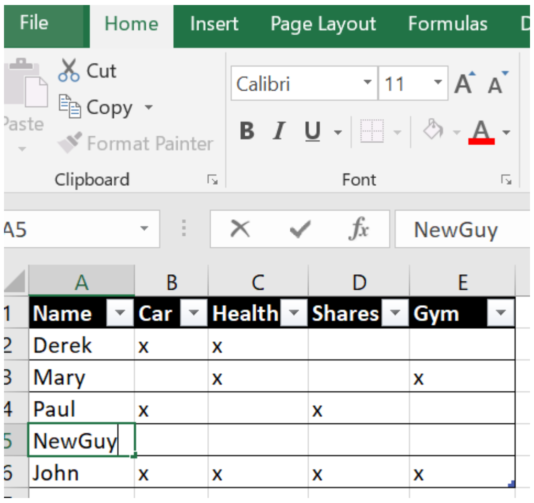
Voilà! Using the **VALUES** function we have managed to alter the **CALCULATE** function's behaviour and stop it from overriding the row context values. Our data no longer has the same values for each Product Category!

More *Power Pivot Principles* next month.

Power Query Pointers

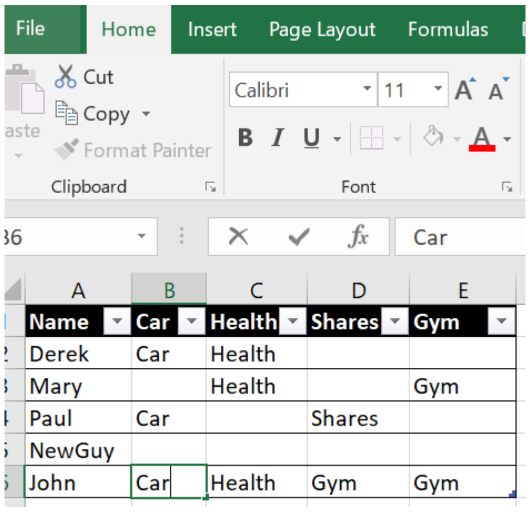
*Each month we'll reproduce one of our articles on Power Query (Excel 2010 and 2013) / Get & Transform (Office 365, Excel 2016 and 2019) from **www.sumproduct.com/blog**. If you wish to read more in the meantime, simply check out our Blog section each Wednesday. This month, we look at a more comprehensive way to convert a check box into a name.*

Continuing last newsletter's topic, the aim here is to take the table below and transform it into a more useful layout.



Name	Car	Health	Shares	Gym
Derek	x	x		
Mary		x		x
Paul	x		x	
NewGuy				
John	x	x	x	x

We will be using the same simple table showing the usual imaginary salespeople and the benefits they may access. Let's have the same goal as last month; we want to end up with the layout below:

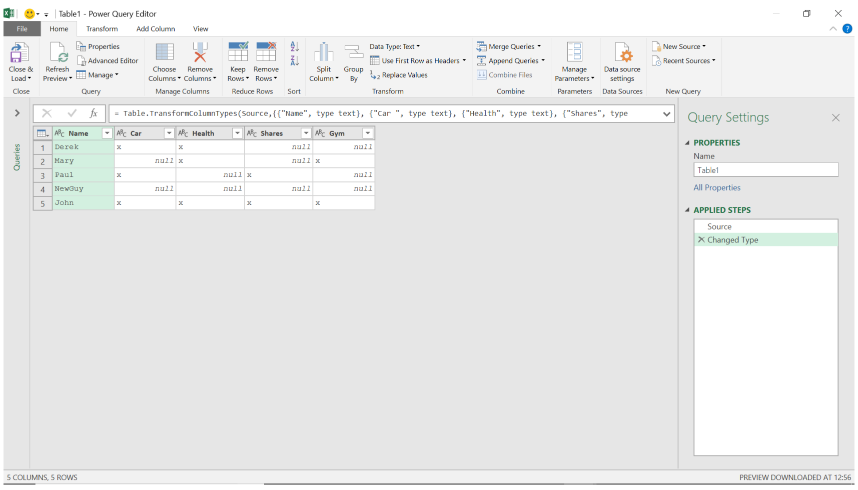


Name	Car	Health	Shares	Gym
Derek	Car	Health		
Mary		Health		Gym
Paul	Car		Shares	
NewGuy				
John	Car	Health	Gym	Gym

Last time, we looked at a way to achieve this on a small scale, by adding new columns. This time, we will look at a more complex version that will work for any number of columns.

In order to construct my table in the format that is required, let's first break it up into its constituent parts, that is, the Salesperson **Name** and Benefit types (*i.e.* a list of all the headings). We will also create a list of pairings between salespeople and benefits received. Finally, we can recreate the table, not forgetting that we will also need to show which benefits are *not* received (this is particularly appropriate for poor 'NewGuy'!).

Let's create a query using 'From Table' on the 'Get and Transform' section of the 'Data' tab:



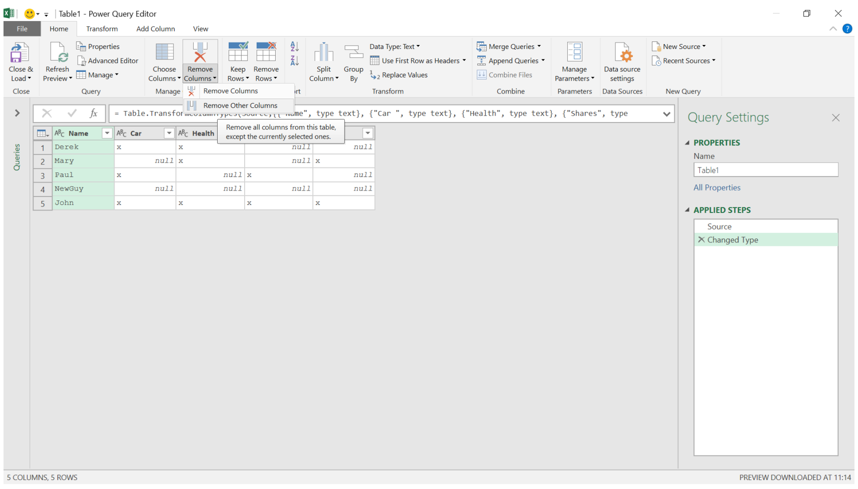
```
= Table.TransformColumnTypes(Source,{{"Name", type text}, {"Car ", type text}, {"Health", type text}, {"Shares", type
```

	Name	Car	Health	Shares	Gym
1	Derek	x	x	null	null
2	Mary	null	x	null	x
3	Paul	x	null	x	null
4	NewGuy	null	null	null	null
5	John	x	x	x	x

Query Settings
PROPERTIES
Name: Table1
APPLIED STEPS
Source
Changed Type

5 COLUMNS, 5 ROWS
PREVIEW DOWNLOADED AT 12:56

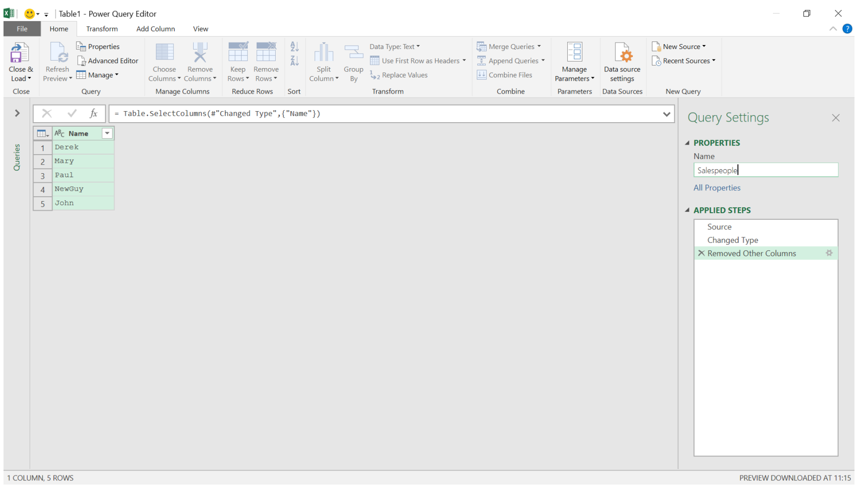
The first step is to obtain a list of salespeople, so select the **Name** column and choose to 'Remove other Columns' from the 'Remove Columns' dropdown in the 'Manage Columns' section on the 'Home' tab. Note that in most cases, the functions used in this example are also available by right-clicking on the appropriate column.



Remove all columns from this table, except the currently selected ones.

5 COLUMNS, 5 ROWS
PREVIEW DOWNLOADED AT 11:14

We have our first query, which we will call **'Salespeople'**.



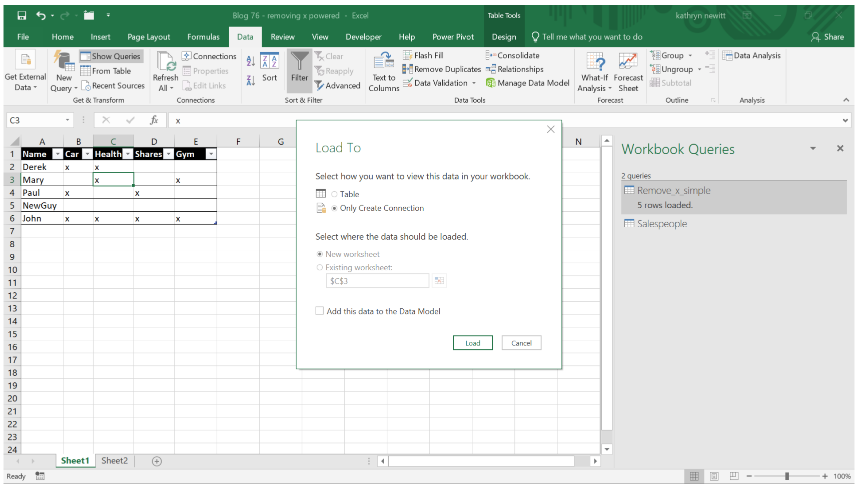
```
= Table.SelectColumns(#"Changed Type",{"Name"})
```

	Name
1	Derek
2	Mary
3	Paul
4	NewGuy
5	John

Query Settings
PROPERTIES
Name: Salespeople
APPLIED STEPS
Source
Changed Type
Removed Other Columns

1 COLUMN, 5 ROWS
PREVIEW DOWNLOADED AT 11:15

Save this query as a 'Connection Only', since we will be using it to build the final query.



Name	Car	Health	Shares	Gym
Derek	x	x		
Mary		x		x
Paul	x		x	
NewGuy				
John	x	x	x	x

Load To

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

Table [✓ Only Create Connection]

Select where the data should be loaded.

[✓ New worksheet] Existing worksheet: $C$3

Add this data to the Data Model

Load Cancel

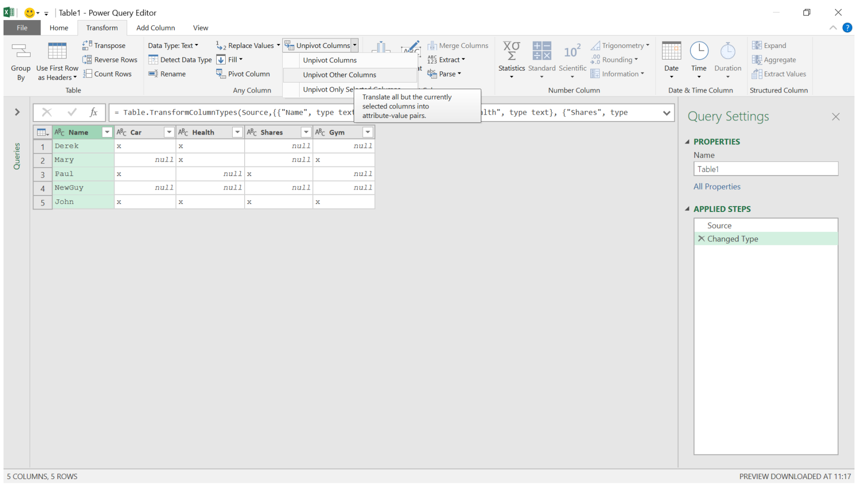
Workbook Queries

2 queries

Remove_x_simple
5 rows loaded.

Salespeople

Now we require the next part of the data: a list of pairs to show each benefit a salesperson receives. Let's start the query in the same way as before from the table, but this time we will need to unpivot the data. On the 'Transform' tab, having selected the **Name** column, we may choose to 'Unpivot Other Columns' from the 'Unpivot Columns' drop down in the 'Any Column' section.



```
= Table.TransformColumnTypes(Source,{{"Name", type text ... lth", type text}, {"Shares", type
```

Unpivot Columns
Unpivot Other Columns
Unpivot Only Selected Columns

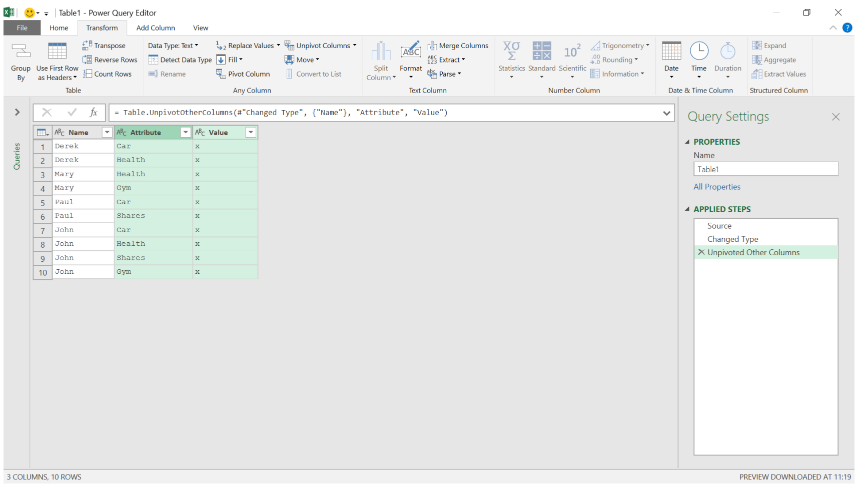
Translate all but the currently selected columns into attribute-value pairs.

	Name	Car	Health	Shares	Gym
1	Derek	x	x	null	null
2	Mary	null	x	null	x
3	Paul	x	null	x	null
4	NewGuy	null	null	null	null
5	John	x	x	x	x

Query Settings
PROPERTIES
Name
Table1
APPLIED STEPS
Source
Changed Type

5 COLUMNS, 5 ROWS PREVIEW DOWNLOADED AT 11:17

This gives us a list of salespeople and each benefit they receive. Note that 'NewGuy' is not shown because no values existed in the unpivoted columns for them. All the *null* cells, which had no '**x**' in them, have been removed as part of the unpivoting process.



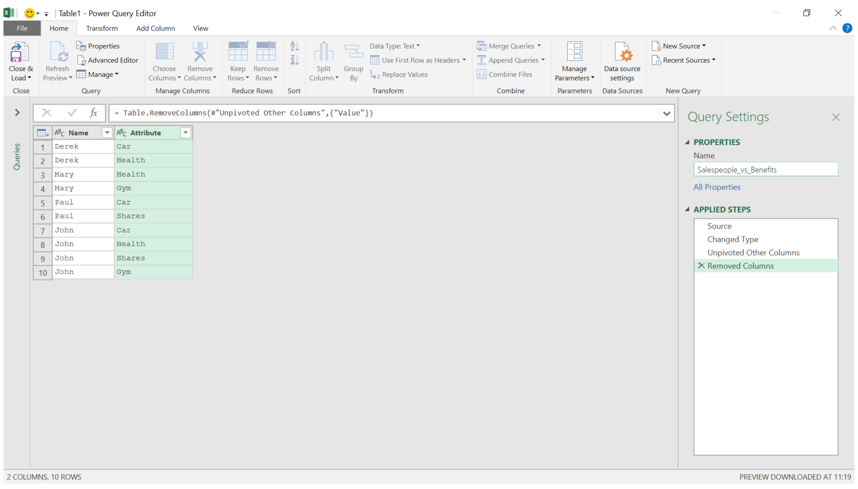
```
= Table.UnpivotOtherColumns(#"Changed Type", {"Name"}, "Attribute", "Value")
```

	Name	Attribute	Value
1	Derek	Car	x
2	Derek	Health	x
3	Mary	Health	x
4	Mary	Gym	x
5	Paul	Car	x
6	Paul	Shares	x
7	John	Car	x
8	John	Health	x
9	John	Shares	x
10	John	Gym	x

Query Settings
PROPERTIES
Name
Table1
APPLIED STEPS
Source
Changed Type
Unpivoted Other Columns

3 COLUMNS, 10 ROWS PREVIEW DOWNLOADED AT 11:19

We may now delete the **Value** column and rename the query (**Salespeople_vs_Benefits**), which we will also 'Close & Load' to 'Connection Only'.



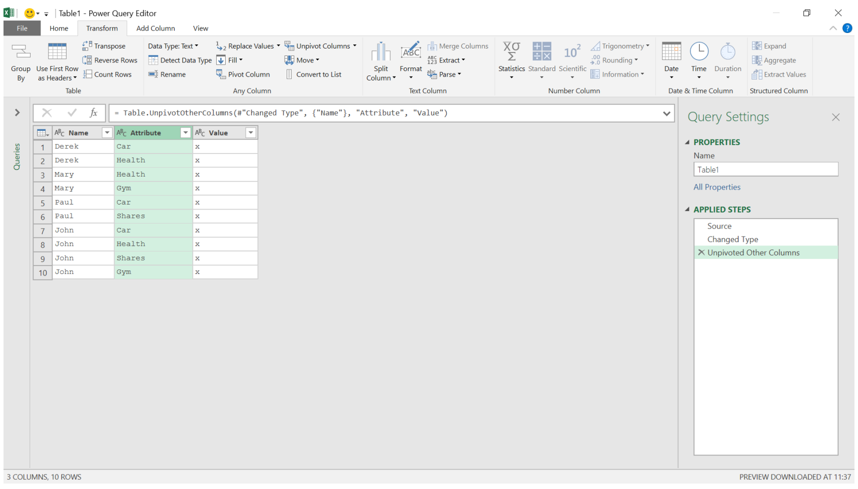
```
= Table.RemoveColumns(#"Unpivoted Other Columns",{"Value"})
```

	Name	Attribute
1	Derek	Car
2	Derek	Health
3	Mary	Health
4	Mary	Gym
5	Paul	Car
6	Paul	Shares
7	John	Car
8	John	Health
9	John	Shares
10	John	Gym

Name: Salespeople_vs_Benefits

Applied steps:
- Source
- Changed Type
- Unpivoted Other Columns
- Removed Columns

Thus, we have a list of salespeople and a list of benefits that each salesperson receives. We're now going to construct a list of the benefits that are available. Let's start again from the original table, and unpivot everything except the name once more.



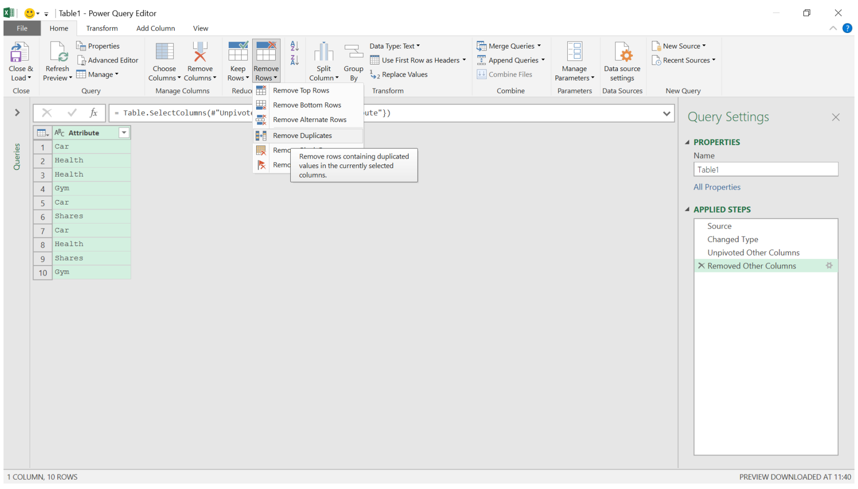
```
= Table.UnpivotOtherColumns(#"Changed Type", {"Name"}, "Attribute", "Value")
```

	Name	Attribute	Value
1	Derek	Car	x
2	Derek	Health	x
3	Mary	Health	x
4	Mary	Gym	x
5	Paul	Car	x
6	Paul	Shares	x
7	John	Car	x
8	John	Health	x
9	John	Shares	x
10	John	Gym	x

Name: Table1

Applied steps:
- Source
- Changed Type
- Unpivoted Other Columns

This time, we're going to only keep the **Attribute** column, and 'Remove Duplicates' from 'Remove Rows' dropdown on the 'Home Menu', *viz*.



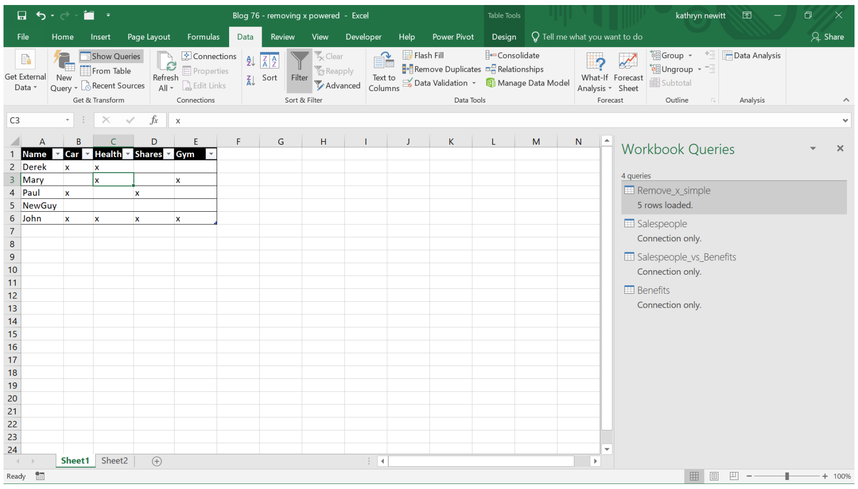
```
= Table.SelectColumns(#"Unpivote   ute"})
```

	Attribute
1	Car
2	Health
3	Health
4	Gym
5	Car
6	Shares
7	Car
8	Health
9	Shares
10	Gym

Name: Table1

Applied steps:
- Source
- Changed Type
- Unpivoted Other Columns
- Removed Other Columns

We now have a list of benefits, which we'll also save as a 'Connection Only'. Let's call this query '**Benefits**'.



	A	B	C	D	E
1	Name	Car	Health	Shares	Gym
2	Derek	x	x		
3	Mary		x		x
4	Paul	x		x	
5	NewGuy				
6	John	x	x	x	x

Workbook Queries

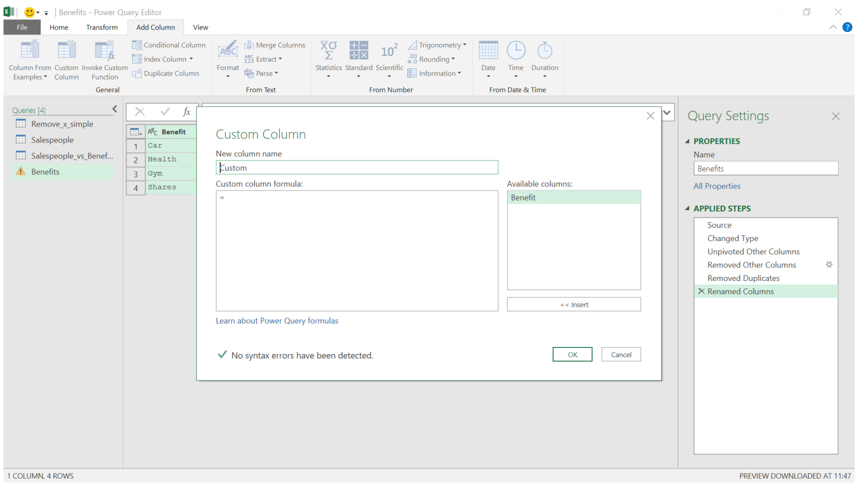
4 queries

- Remove_x_simple — 5 rows loaded.
- Salespeople — Connection only.
- Salespeople_vs_Benefits — Connection only.
- Benefits — Connection only.

Now we have our building blocks; we may now create our final query.

Let's start with our '**Benefits**' query. We need to pull the salespeople back into this query, so we shall add a custom column. The method we'll use is a cross join, which we have discussed in a previous newsletter.

Let's add a 'Custom Column' from the 'Add Column' tab. It's good to open the Query pane to the left of the screen so that you may see the names our other queries, *viz*.



Queries [4]: Remove_x_simple, Salespeople, Salespeople_vs_Benef..., Benefits

	Benefit
1	Car
2	Health
3	Gym
4	Shares

Custom Column

New column name: Custom

Custom column formula: =

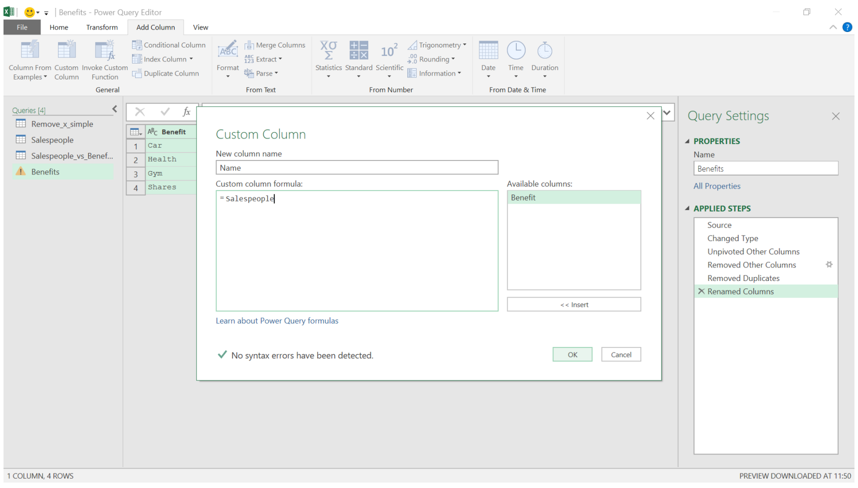
Available columns: Benefit

No syntax errors have been detected.

APPLIED STEPS: Source, Changed Type, Unpivoted Other Columns, Removed Other Columns, Removed Duplicates, Renamed Columns

1 COLUMN, 4 ROWS — PREVIEW DOWNLOADED AT 11:47

We'll add the salespeople first, so let's use the '**Salespeople**' query:



Custom Column

New column name: Name

Custom column formula: = Salespeople

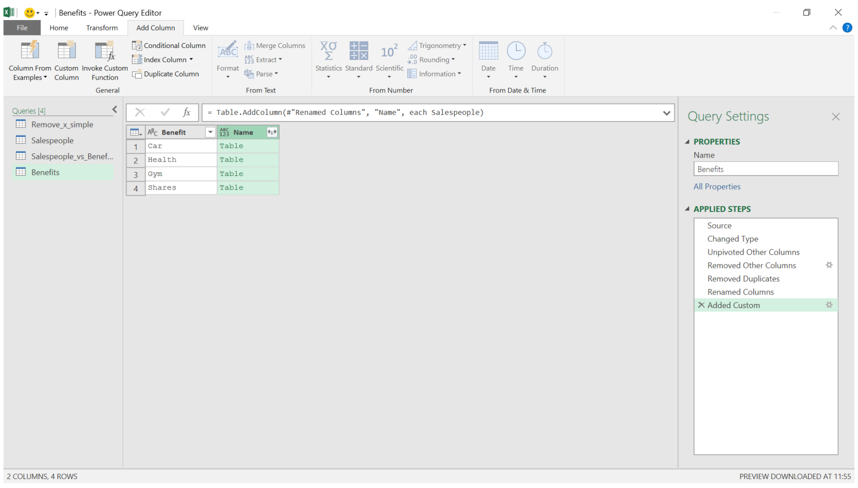
Available columns: Benefit

No syntax errors have been detected.

1 COLUMN, 4 ROWS — PREVIEW DOWNLOADED AT 11:50

This initially gives us a table in each row.

This initially gives us a table in each row.



Queries [4]
- Remove_x_simple
- Salespeople
- Salespeople_vs_Benef...
- Benefits

```
= Table.AddColumn(#"Renamed Columns", "Name", each Salespeople)
```

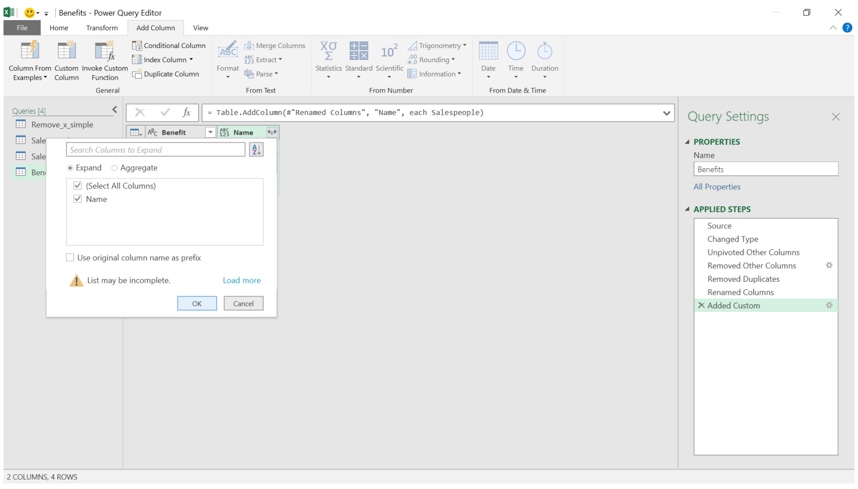
	Benefit	Name
1	Car	Table
2	Health	Table
3	Gym	Table
4	Shares	Table

Name: Benefits

APPLIED STEPS:
- Source
- Changed Type
- Unpivoted Other Columns
- Removed Other Columns
- Removed Duplicates
- Renamed Columns
- Added Custom

2 COLUMNS, 4 ROWS   PREVIEW DOWNLOADED AT 11:55

We can expand the table using the icon next to the column heading.



```
= Table.AddColumn(#"Renamed Columns", "Name", each Salespeople)
```

Expand [✓] / Aggregate

- ☑ (Select All Columns)
- ☑ Name

☐ Use original column name as prefix

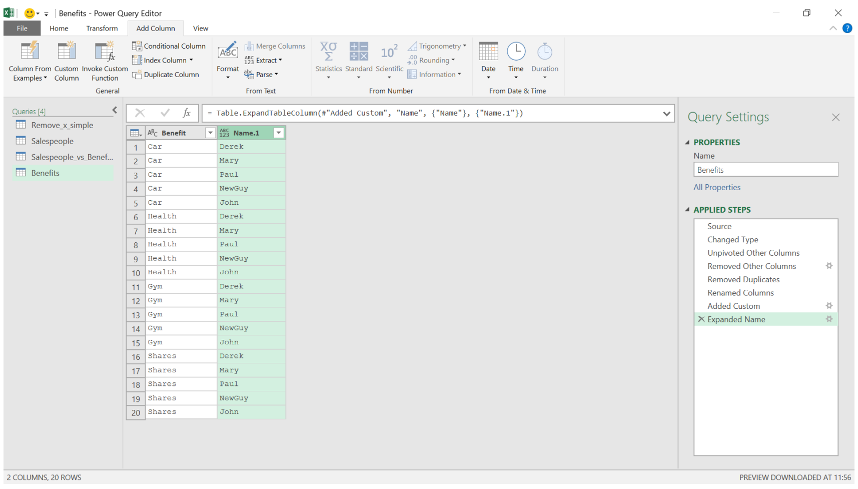
List may be incomplete.

Name: Benefits

APPLIED STEPS:
- Source
- Changed Type
- Unpivoted Other Columns
- Removed Other Columns
- Removed Duplicates
- Renamed Columns
- Added Custom

2 COLUMNS, 4 ROWS

Presumably, the salespeople would like this (especially NewGuy!), but the data now shows all the possible combinations available. We need to change this to show those that are actually received!



Queries [4]
- Remove_x_simple
- Salespeople
- Salespeople_vs_Benef...
- Benefits

```
= Table.ExpandTableColumn(#"Added Custom", "Name", {"Name"}, {"Name.1"})
```

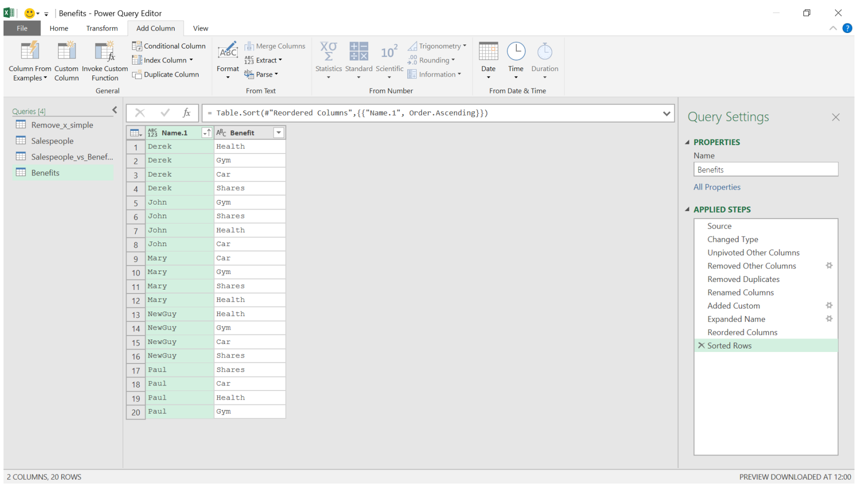
	Benefit	Name.1
1	Car	Derek
2	Car	Mary
3	Car	Paul
4	Car	NewGuy
5	Car	John
6	Health	Derek
7	Health	Mary
8	Health	Paul
9	Health	NewGuy
10	Health	John
11	Gym	Derek
12	Gym	Mary
13	Gym	Paul
14	Gym	NewGuy
15	Gym	John
16	Shares	Derek
17	Shares	Mary
18	Shares	Paul
19	Shares	NewGuy
20	Shares	John

Name: Benefits

APPLIED STEPS:
- Source
- Changed Type
- Unpivoted Other Columns
- Removed Other Columns
- Removed Duplicates
- Renamed Columns
- Added Custom
- Expanded Name

2 COLUMNS, 20 ROWS   PREVIEW DOWNLOADED AT 11:56

Let's start to get our data ready to resemble the table I want by putting **Name** first and sorting on **Name**.



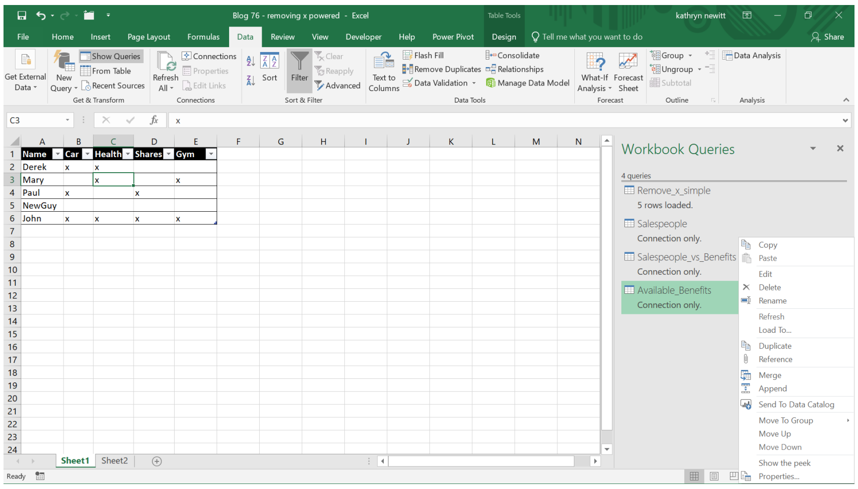
```
= Table.Sort(#"Reordered Columns",{{"Name.1", Order.Ascending}})
```

	Name.1	Benefit
1	Derek	Health
2	Derek	Gym
3	Derek	Car
4	Derek	Shares
5	John	Gym
6	John	Shares
7	John	Health
8	John	Car
9	Mary	Car
10	Mary	Gym
11	Mary	Shares
12	Mary	Health
13	NewGuy	Health
14	NewGuy	Gym
15	NewGuy	Car
16	NewGuy	Shares
17	Paul	Shares
18	Paul	Car
19	Paul	Health
20	Paul	Gym

APPLIED STEPS: Source, Changed Type, Unpivoted Other Columns, Removed Other Columns, Removed Duplicates, Renamed Columns, Added Custom, Expanded Name, Reordered Columns, Sorted Rows

2 COLUMNS, 20 ROWS — PREVIEW DOWNLOADED AT 12:00

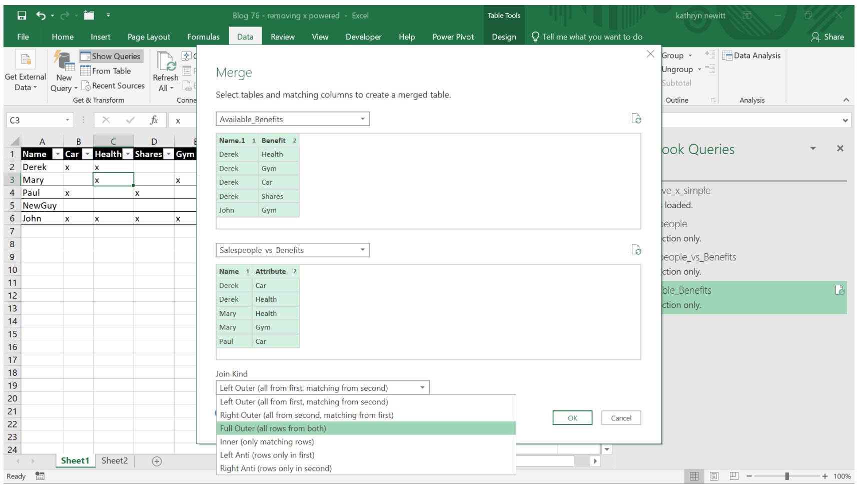
We'll save this query as '**Available_Benefits**'. Next, we are going to create a new query by merging some of our existing queries.



	A	B	C	D	E
1	Name	Car	Health	Shares	Gym
2	Derek	x	x		
3	Mary		x		x
4	Paul	x		x	
5	NewGuy				
6	John	x	x	x	x

Workbook Queries
4 queries
Remove_x_simple
5 rows loaded.
Salespeople
Connection only.
Salespeople_vs_Benefits
Connection only.
Available_Benefits
Connection only.

We'll access the Merge option by right-clicking on the '**Available_ Benefits**' query. We want to merge '**Available_Benefits**' with '**Salespeople_vs_Benefits**', linking all the possible benefits to the ones that are actually received. This will allow us to keep 'NewGuy', as he is in the '**Available_Benefits**' table.


Blog 76 - removing x powered - Excel
Table Tools
kathryn newitt
Data
Design
Tell me what you want to do
Share
OK
Cancel

Merge

Select tables and matching columns to create a merged table.

Available_Benefits

Name.1	Benefit
Derek	Health
Derek	Gym
Derek	Car
Derek	Shares
John	Gym

Salespeople_vs_Benefits

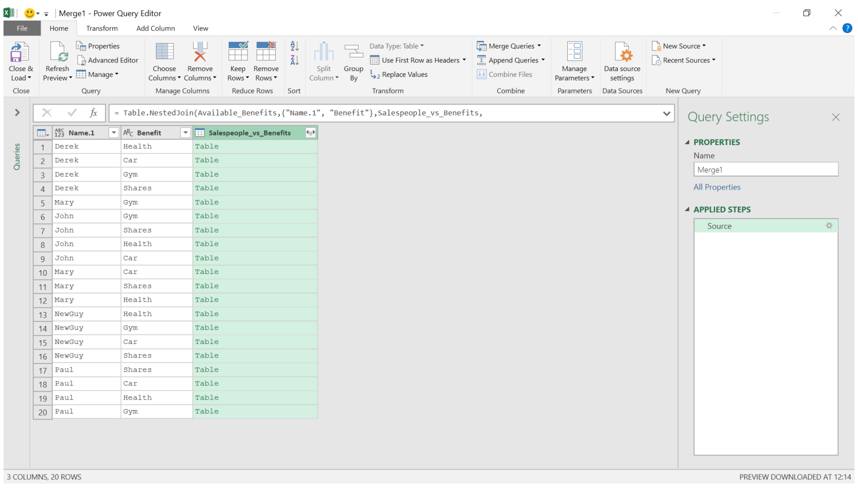
Name	Attribute
Derek	Car
Derek	Health
Mary	Health
Mary	Gym
Paul	Car

Join Kind

Left Outer (all from first, matching from second)

- Left Outer (all from first, matching from second)
- Right Outer (all from second, matching from first)
- Full Outer (all rows from both)
- Inner (only matching rows)
- Left Anti (rows only in first)
- Right Anti (rows only in second)

We wish to join by matching both of our columns, and we need to include everyone in the first query (so that 'NewGuy' doesn't get left behind) and all the data in the second query (as that tells us who gets the benefits). Let's choose a 'Full Outer' join, combining all the available data from both tables:



Merge1 - Power Query Editor

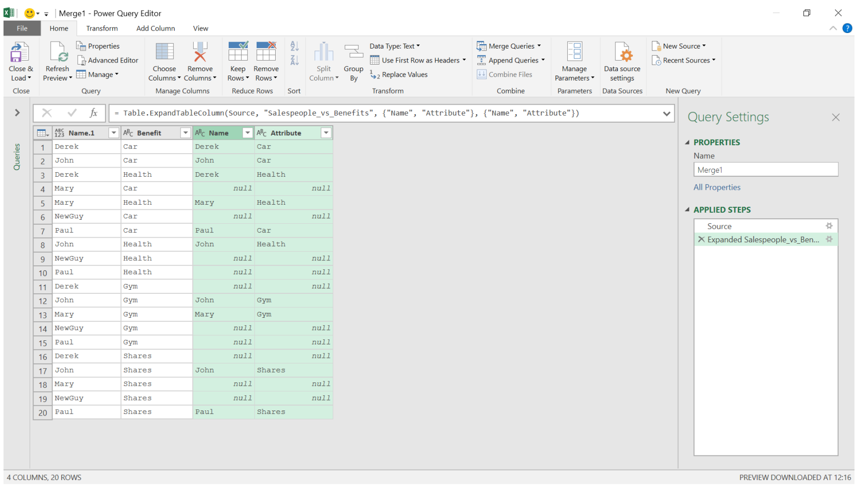
```
= Table.NestedJoin(Available_Benefits,{"Name.1", "Benefit"},Salespeople_vs_Benefits,
```

	Name.1	Benefit	Salespeople_vs_Benefits
1	Derek	Health	Table
2	Derek	Car	Table
3	Derek	Gym	Table
4	Derek	Shares	Table
5	Mary	Gym	Table
6	John	Gym	Table
7	John	Shares	Table
8	John	Health	Table
9	John	Car	Table
10	Mary	Car	Table
11	Mary	Shares	Table
12	Mary	Health	Table
13	NewGuy	Health	Table
14	NewGuy	Gym	Table
15	NewGuy	Car	Table
16	NewGuy	Shares	Table
17	Paul	Shares	Table
18	Paul	Car	Table
19	Paul	Health	Table
20	Paul	Gym	Table

Query Settings: Name: Merge1; Applied Steps: Source

3 COLUMNS, 20 ROWS   PREVIEW DOWNLOADED AT 12:14

We may expand the **Salespeople_vs_Benefits** column to show all the columns in that table.



Merge1 - Power Query Editor

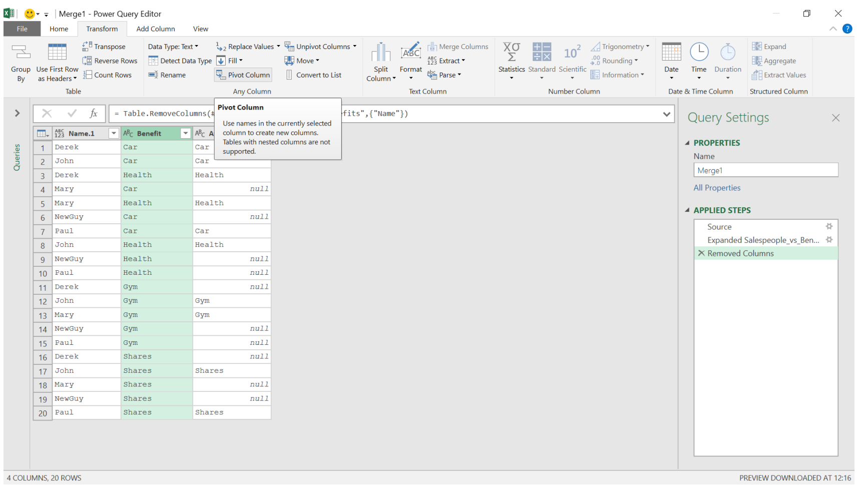
```
= Table.ExpandTableColumn(Source, "Salespeople_vs_Benefits", {"Name", "Attribute"}, {"Name", "Attribute"})
```

	Name.1	Benefit	Name	Attribute
1	Derek	Car	Derek	Car
2	John	Car	John	Car
3	Derek	Health	Derek	Health
4	Mary	Car	null	null
5	Mary	Health	Mary	Health
6	NewGuy	Car	null	null
7	Paul	Car	Paul	Car
8	John	Health	John	Health
9	NewGuy	Health	null	null
10	Paul	Health	null	null
11	Derek	Gym	null	null
12	John	Gym	John	Gym
13	Mary	Gym	Mary	Gym
14	NewGuy	Gym	null	null
15	Paul	Gym	null	null
16	Derek	Shares	null	null
17	John	Shares	John	Shares
18	Mary	Shares	null	null
19	NewGuy	Shares	null	null
20	Paul	Shares	Paul	Shares

Query Settings: Name: Merge1; Applied Steps: Source, Expanded Salespeople_vs_Ben...

4 COLUMNS, 20 ROWS   PREVIEW DOWNLOADED AT 12:16

Now you can see that we're starting to get the data in the format we'd like. We just need to remove the **Name** column and pivot the **Benefit** column so that they become the headings. We may do this by selecting the **Benefit** column and choosing the 'Pivot' option in the 'Any Column' section of the Transform tab.



Merge1 - Power Query Editor

```
= Table.RemoveColumns(#  fits",{"Name"})
```

Pivot Column

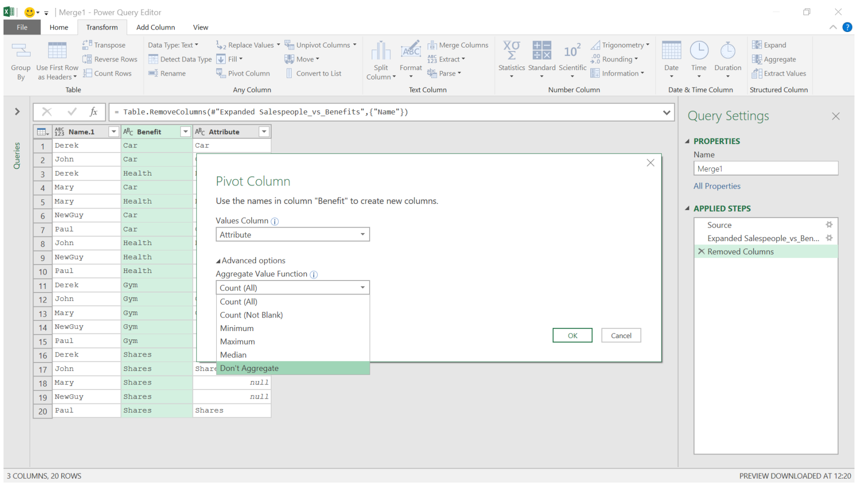
Use names in the currently selected column to create new columns. Tables with nested columns are not supported.

	Name.1	Benefit	A
1	Derek	Car	Car
2	John	Car	Car
3	Derek	Health	Health
4	Mary	Car	null
5	Mary	Health	Health
6	NewGuy	Car	null
7	Paul	Car	Car
8	John	Health	Health
9	NewGuy	Health	null
10	Paul	Health	null
11	Derek	Gym	null
12	John	Gym	Gym
13	Mary	Gym	Gym
14	NewGuy	Gym	null
15	Paul	Gym	null
16	Derek	Shares	null
17	John	Shares	Shares
18	Mary	Shares	null
19	NewGuy	Shares	null
20	Paul	Shares	Shares

Query Settings: Name: Merge1; Applied Steps: Source, Expanded Salespeople_vs_Ben..., Removed Columns

4 COLUMNS, 20 ROWS   PREVIEW DOWNLOADED AT 12:16

The values need to come from the **Attributes** column, as that holds the logic for the benefits, and we don't want Power Query to apply any aggregation.



```
= Table.RemoveColumns(#"Expanded Salespeople_vs_Benefits",{"Name"})
```

	Name.1	Benefit	Attribute
1	Derek	Car	Car
2	John	Car	
3	Derek	Health	
4	Mary	Car	
5	Mary	Health	
6	NewGuy	Car	
7	Paul	Car	
8	John	Health	
9	NewGuy	Health	
10	Paul	Health	
11	Derek	Gym	
12	John	Gym	
13	Mary	Gym	
14	NewGuy	Gym	
15	Paul	Gym	
16	Derek	Shares	
17	John	Shares	Share
18	Mary	Shares	null
19	NewGuy	Shares	null
20	Paul	Shares	Shares

Pivot Column

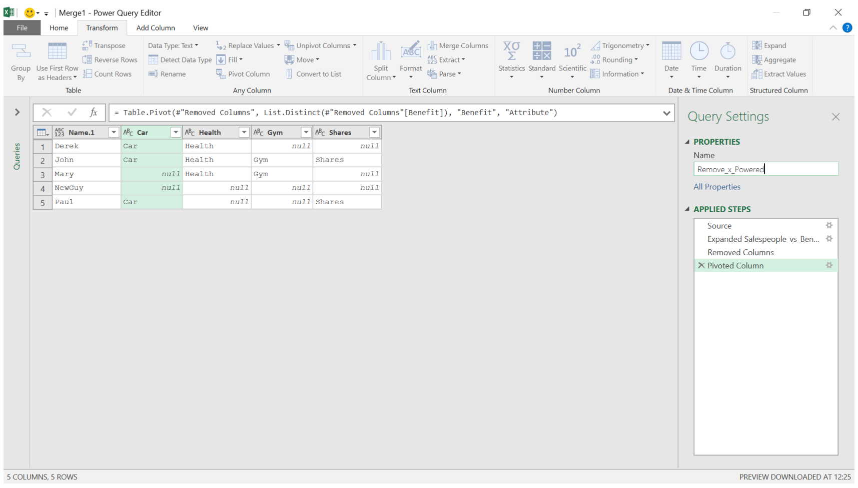
Use the names in column "Benefit" to create new columns.

Values Column: Attribute

Advanced options

Aggregate Value Function: Count (All)

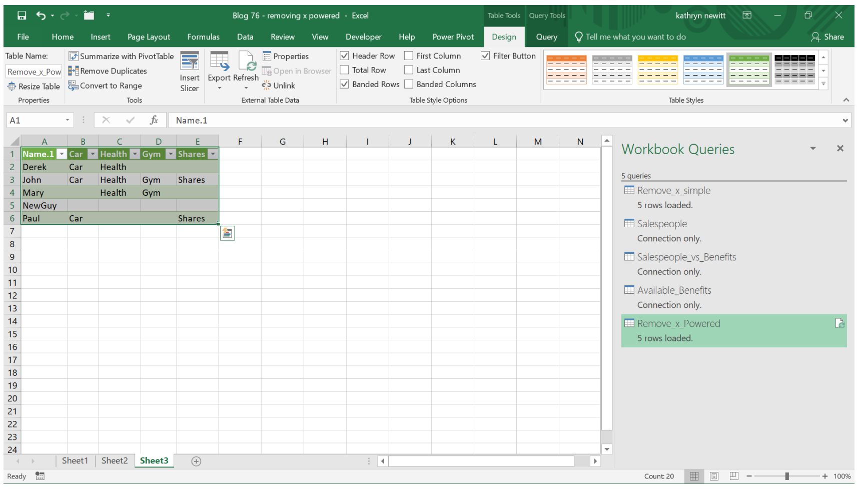
Our results are now looking good!



```
= Table.Pivot(#"Removed Columns", List.Distinct(#"Removed Columns"[Benefit]), "Benefit", "Attribute")
```

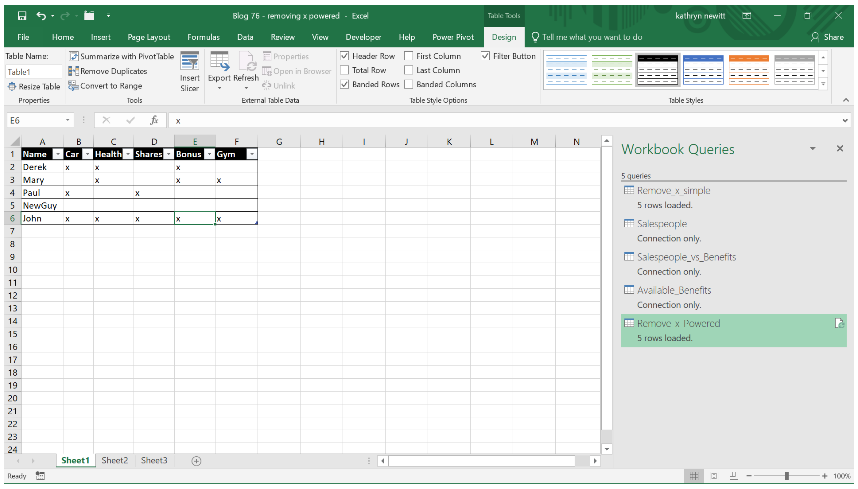
	Name.1	Car	Health	Gym	Shares
1	Derek	Car	Health	null	null
2	John	Car	Health	Gym	Shares
3	Mary	null	Health	Gym	null
4	NewGuy	null	null	null	null
5	Paul	Car	null	null	Shares

We may now load this data to our worksheet:



	A	B	C	D	E
1	Name.1	Car	Health	Gym	Shares
2	Derek	Car	Health		
3	John	Car	Health	Gym	Shares
4	Mary		Health	Gym	
5	NewGuy				
6	Paul	Car			Shares

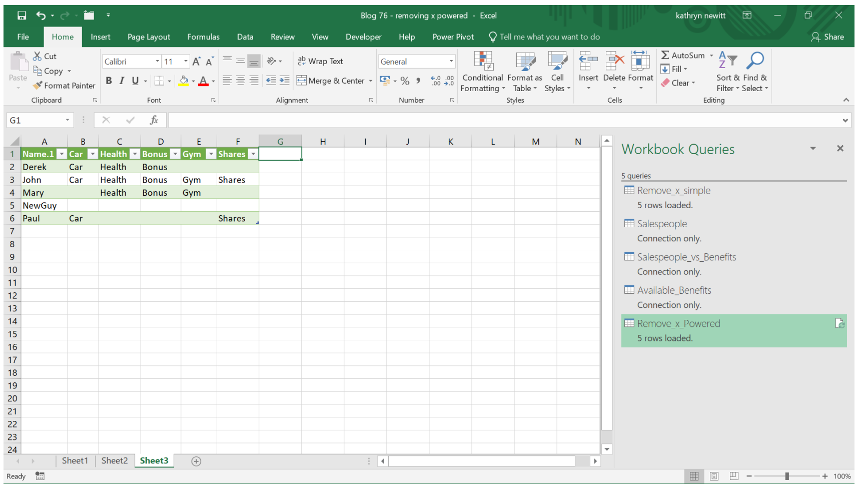
Finally, we want to see what happens if we add a new benefit and refresh our query.



Name	Car	Health	Shares	Bonus	Gym
Derek	x	x		x	
Mary		x		x	x
Paul	x		x		
NewGuy					
John	x	x	x	x	x

Workbook Queries
5 queries
Remove_x_simple
5 rows loaded.
Salespeople
Connection only.
Salespeople_vs_Benefits
Connection only.
Available_Benefits
Connection only.
Remove_x_Powered
5 rows loaded.

A new bonus scheme has arrived, but not everyone is eligible! Let's refresh the query.



Name.1	Car	Health	Bonus	Gym	Shares
Derek	Car	Health	Bonus		
John	Car	Health	Bonus	Gym	Shares
Mary		Health	Bonus	Gym	
NewGuy					
Paul	Car				Shares

Workbook Queries
5 queries
Remove_x_simple
5 rows loaded.
Salespeople
Connection only.
Salespeople_vs_Benefits
Connection only.
Available_Benefits
Connection only.
Remove_x_Powered
5 rows loaded.

Derek, John and Mary are clearly on the bonus scheme. As we have broken the query down and reconstructed it, any changes to benefits or staff will be reflected when refreshed.

More next month.

Power BI Updates

As always, there is yet another variety of new features. This month's updates include features such as an hierarchical axis by default, translations for composite models, mobile formatting options in General Availability and cross tenants' datasets sharing. The full list is as follows:

Reporting

- Hierarchical axis by default
- Improved display name for summarised fields
- Conditional formatting for data labels
- Translations support for Composite models on Power BI Datasets and Analysis Services in Preview
- Mobile formatting options now Generally Available
- Information protection update

Data connectivity and preparation

- Dremio (Connector Update)
- Profisee (New Connector)
- Starburst Enterprise (Connector Update)

Service

- Cross-tenant dataset sharing in Preview
- Auto-generate reports on existing datasets
- Discoverability feature for B2B content

Mobile applications

- Power BI fonts now available on mobile devices (iOS and Android)
- Windows app: upgraded browsing experience with WebView2

Embedded Analytics

- Power BI component for Vue.js
- Power BI and Jupyter integration updates (delayed until October 2022)
- Export paginated reports by a service principal with a Power BI dataset as a data source

Visualisations

- New visuals in AppSource
- Shielded HTML Viewer by Nova Silva
- Drill Down Combo PRO by ZoomCharts
- Intelligent Narratives by Arria NLG

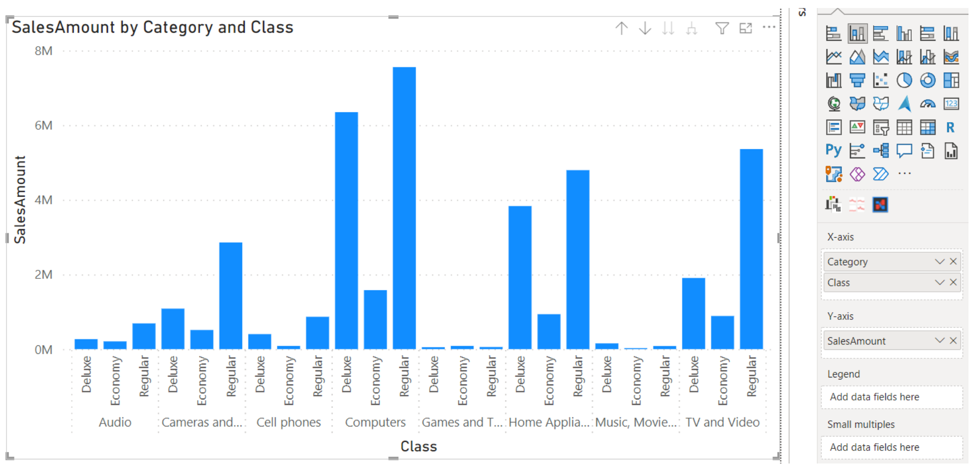
Other

- Paginated reports data Preview.

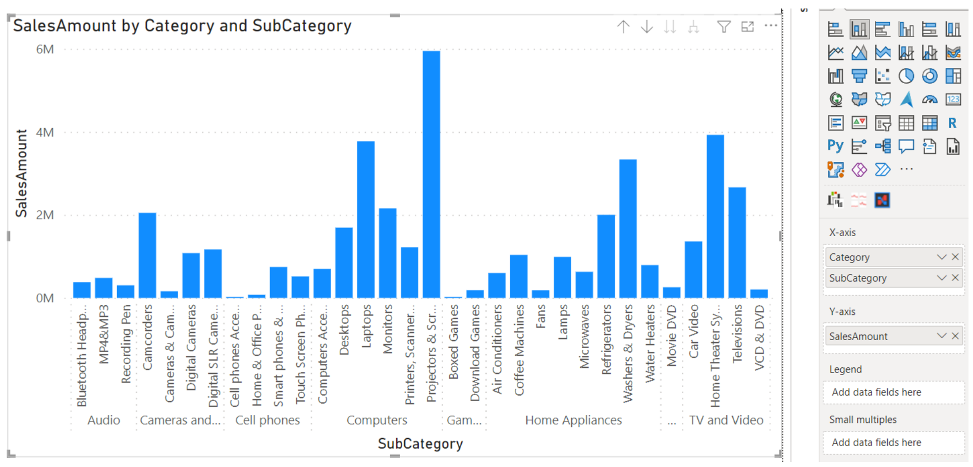
Let's now go through each in turn.

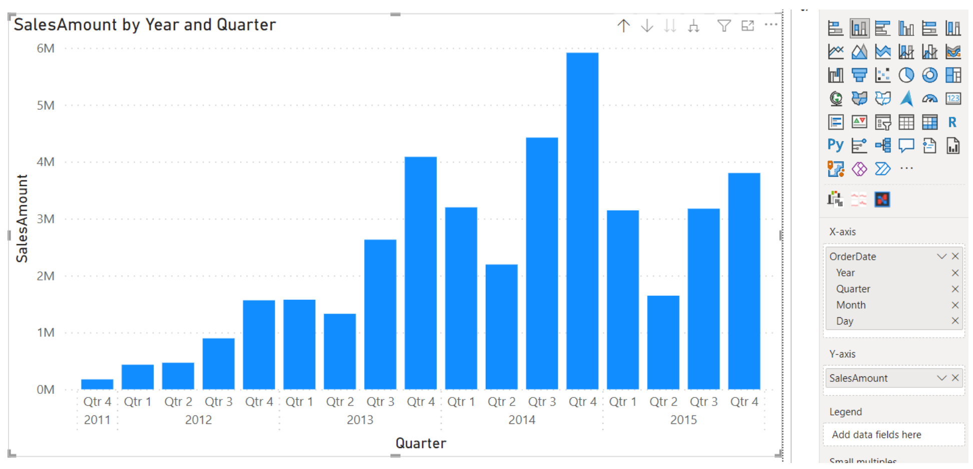
Hierarchical axis by default

One of Power BI's Cartesian charts' more useful capabilities is the option to visualise multiple categorical fields in an hierarchical **x**-axis.



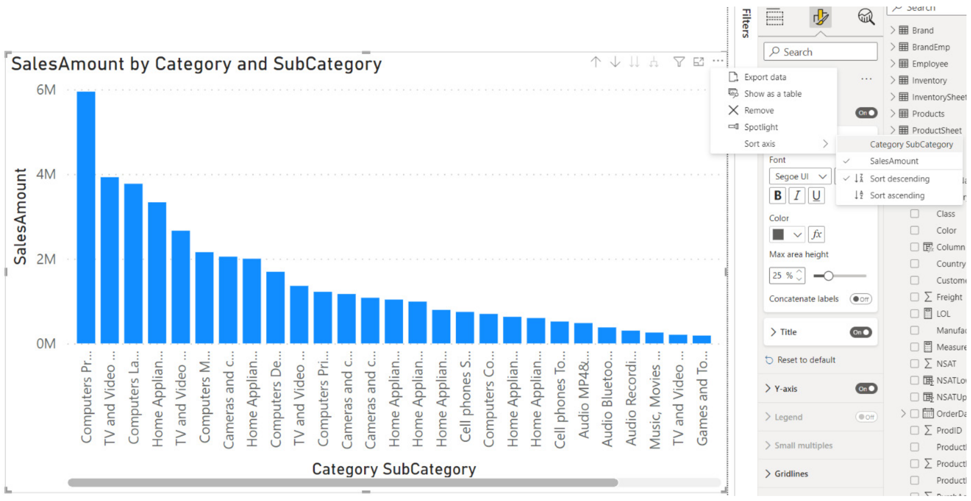
The grouped **x**-axis is an organised way to add an additional dimension to the data in your charts, especially when dealing with subcategories that are specific to individual category fields or sequential subcategories such as more granular dates.





A different form of visualising these additional dimensions, like using a legend or small multiples, would not be nearly as clear as the hierarchical **x**-axis. However, before this release, formatting a chart to use the hierarchical **x**-axis was a multi-step process involving:

- adding a new field to the X-axis field well and knowing to expand the visual down to the next level, or else nothing would change
- turning off the concatenate labels option in the Formatting pane, which would often still result in nothing happening because the user also had to sort the visual by the axis fields rather than by value fields, otherwise categories and subcategories would not be grouped properly.



This latest update adjusts the behaviour to ensure that the hierarchy axis is automatically turned on when a user drags multiple fields into the x-axis field well of charts which support the feature. Also, Microsoft has turned the concatenate labels option off by default in the Formatting pane. Power BI will now auto-expand charts down to the bottom of the hierarchy when you add fields to the **x**-axis field well and will also sort on category by default once you drill down.

The exact changes in logic are as follows:

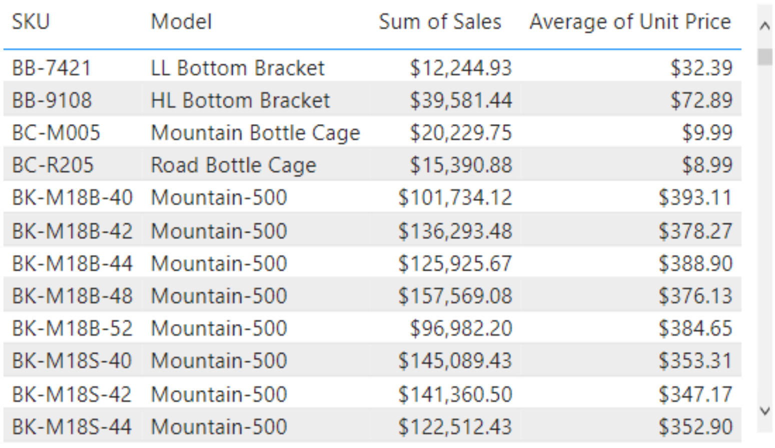
Behaviour	Defaults before update	Defaults after update
Concatenate labels option	On by default	Off by default
Adding new fields to the **x**-axis field well	Adding new fields will not change which fields are shown on the visual (user must manually expand all)	Expand to lowest level when: • the visual supports hierarchy axis • the axis is a categorical or datetime axis (note that numerical fields be come categorical when they're not the top level of a hierarchy)
Sort behaviour	Sort by measure when the user has not explicitly set a sort	Sort by category when: • the visual supports hierarchy axis • when the visual is expanded down in any way • user has not explicitly set a sort • an authoring action is taken (like adding or removing a field; notably, not manually expanding the visual) Otherwise, sort by measure (or by user-set sort).

This only changes how Power BI responds to authoring actions, so existing reports should not be affected, and you should be able to recreate any previous sort or expand state after this release as well.

It is understood that adding new conditions to default behaviour will create some inconsistencies in what happens when you perform an action. That said, Microsoft hopes that they will feel intuitive as you create new charts and that they will save you clicks, Formatting pane navigation and internet troubleshooting.

Improved display name for summarised fields

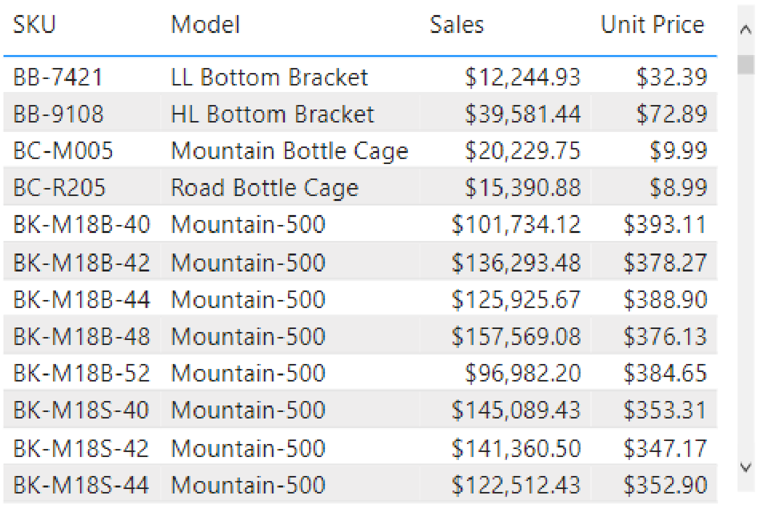
To improve comprehension for end users and new creators, this update sees the display name for summarised (summarized) fields to include the default or selected aggregation:



SKU	Model	Sum of Sales	Average of Unit Price
BB-7421	LL Bottom Bracket	$12,244.93	$32.39
BB-9108	HL Bottom Bracket	$39,581.44	$72.89
BC-M005	Mountain Bottle Cage	$20,229.75	$9.99
BC-R205	Road Bottle Cage	$15,390.88	$8.99
BK-M18B-40	Mountain-500	$101,734.12	$393.11
BK-M18B-42	Mountain-500	$136,293.48	$378.27
BK-M18B-44	Mountain-500	$125,925.67	$388.90
BK-M18B-48	Mountain-500	$157,569.08	$376.13
BK-M18B-52	Mountain-500	$96,982.20	$384.65
BK-M18S-40	Mountain-500	$145,089.43	$353.31
BK-M18S-42	Mountain-500	$141,360.50	$347.17
BK-M18S-44	Mountain-500	$122,512.43	$352.90

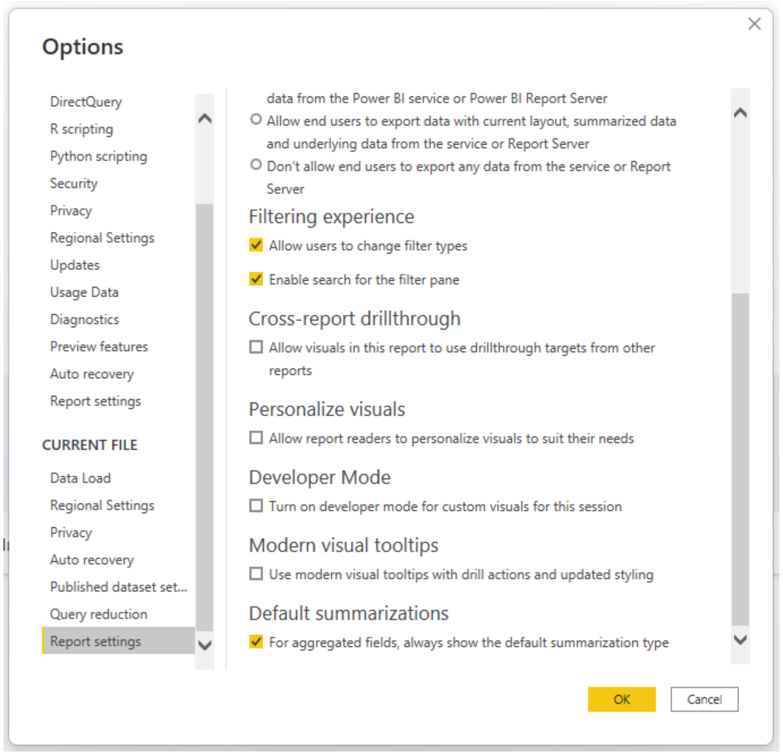
Previously, default aggregates would be dropped from the display name, and this applies to all aggregates not just the 'sum' aggregation. This is in response to the fact that it has been noted that dropping the aggregate leads to users misinterpreting what aggregate is being applied, if any.

For example, in the table *(below)*, the **Sales** and **Unit Price** columns do not indicate to users how they are being aggregated. This may cause users to think both columns have a sum aggregation applied, which is not correct for the **Unit Price** column.



SKU	Model	Sales	Unit Price
BB-7421	LL Bottom Bracket	$12,244.93	$32.39
BB-9108	HL Bottom Bracket	$39,581.44	$72.89
BC-M005	Mountain Bottle Cage	$20,229.75	$9.99
BC-R205	Road Bottle Cage	$15,390.88	$8.99
BK-M18B-40	Mountain-500	$101,734.12	$393.11
BK-M18B-42	Mountain-500	$136,293.48	$378.27
BK-M18B-44	Mountain-500	$125,925.67	$388.90
BK-M18B-48	Mountain-500	$157,569.08	$376.13
BK-M18B-52	Mountain-500	$96,982.20	$384.65
BK-M18S-40	Mountain-500	$145,089.43	$353.31
BK-M18S-42	Mountain-500	$141,360.50	$347.17
BK-M18S-44	Mountain-500	$122,512.43	$352.90

This improvement will be available on all new reports by default. If you want to enable this behaviour for existing reports, you can navigate to **File -> Options and settings -> Options -> Default summarization** and enable the setting: **For aggregated fields, always show the default summarization type**.



Options

data from the Power BI service or Power BI Report Server
○ Allow end users to export data with current layout, summarized data and underlying data from the service or Report Server
○ Don't allow end users to export any data from the service or Report Server

Filtering experience
☑ Allow users to change filter types
☑ Enable search for the filter pane

Cross-report drillthrough
☐ Allow visuals in this report to use drillthrough targets from other reports

Personalize visuals
☐ Allow report readers to personalize visuals to suit their needs

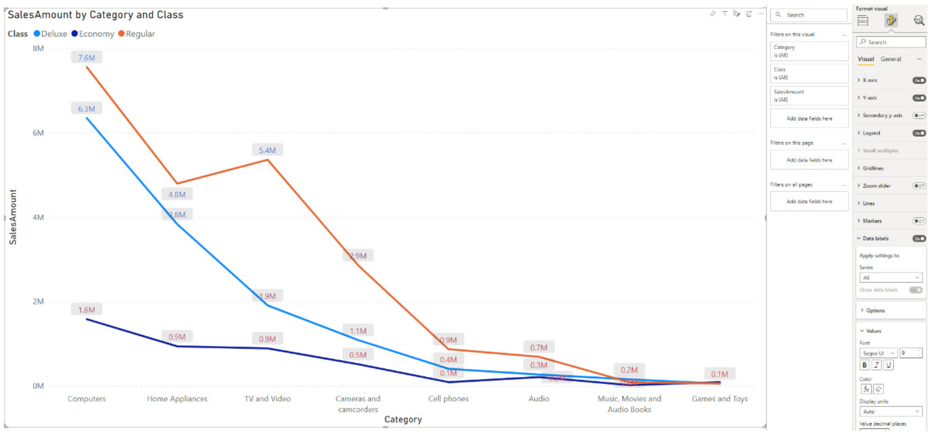
Developer Mode
☐ Turn on developer mode for custom visuals for this session

Modern visual tooltips
☐ Use modern visual tooltips with drill actions and updated styling

Default summarizations
☑ For aggregated fields, always show the default summarization type

Conditional formatting for data labels

Last month, Microsoft released an update to conditional formatting for data labels. This allowed them to apply to each individual data point rather than all of them together. At the time, this improvement was limited to visuals without a field in the Legend field well. Now, this update has brought that same behaviour to those cases as well.



Since this interaction has now been improved, Microsoft has returned the conditional formatting button to the Formatting pane (yay!) for visuals with legend fields. Keep in mind that the conditional formatting rules for data labels that you've already set on current reports will still not be affected until you reapply the rule and publish the report again.

Translations support for Composite models on Power BI Datasets and Analysis Services in Preview

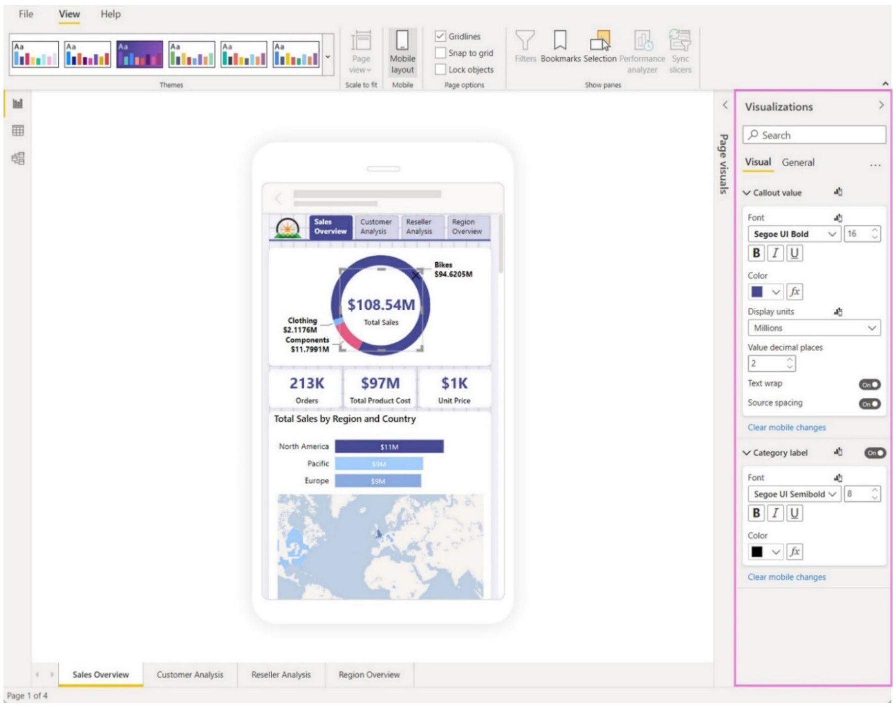
This month sees translations now supported for composite models on Power BI datasets and Analysis Services. Translations enable you to show the tables and columns in your dataset in the language the end user prefers. Up until now, if the source had any translations defined, they would be ignored when building a composite model on Power BI datasets or an Analysis Services model.

However, starting with this release, any translations defined in the source will be exposed in the composite model as well so the table and column names will be translated to the user's language, the same as in the source datasets.

You can also add translations in your composite model and to make sure your local translations are not overwritten by the translations defined in the source, you can set the 'Altered' property on the corresponding object to 'true'. If you currently have translations defined in your composite model, Power BI will set the 'Altered' property automatically once you open the model in this month's Desktop release. In addition, you can use external tools to accomplish the same effect.

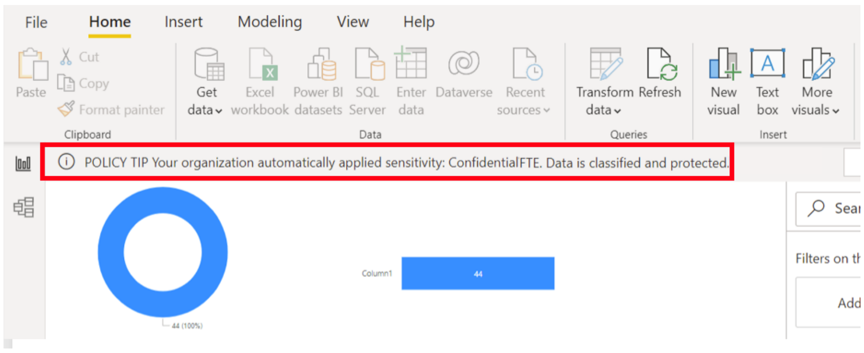
Mobile formatting options now Generally Available

Mobile formatting options are now Generally Available. These options allow you to style and format visuals in mobile-optimised layout without affecting their formatting in web layout. This gives you greater flexibility and opens up a world of design possibilities for creating reports better optimised for phone viewing.



Information protection update

Using Power BI Desktop, you can build reports on a dataset in the Power BI service by creating a live connection to a dataset using either a connection string or the Get Data experience. If the dataset has a sensitivity label, Power BI will automatically apply the live dataset's sensitivity label to the PBIX file to maintain the data's classification and protection as it leaves the Power BI Service.



POLICY TIP Your organization automatically applied sensitivity: ConfidentialFTE. Data is classified and protected.

Dremio (Connector Update)

The Dremio connectors have been updated. This release contains a fix to provide more accurate error reporting when a connection error occurs. It also contains a fix that prevents Dremio Software from displaying virtual datasets (VDSs) in reports in some cases.

Profisee (New Connector)

The Profisee Connector for Power BI makes it easy to access clean, complete and accurate data to help organisations accelerate both operationalised Business Intelligence and *ad hoc* analytics — even if they aren't familiar with master data management (MDM) or the Profisee platform. With just a few clicks, users may load data into Power BI from Profisee, allowing them to easily model and leverage data directly in Power BI.

Starburst Enterprise (Connector Update)

The Starburst Enterprise connector has been updated. It has now added:

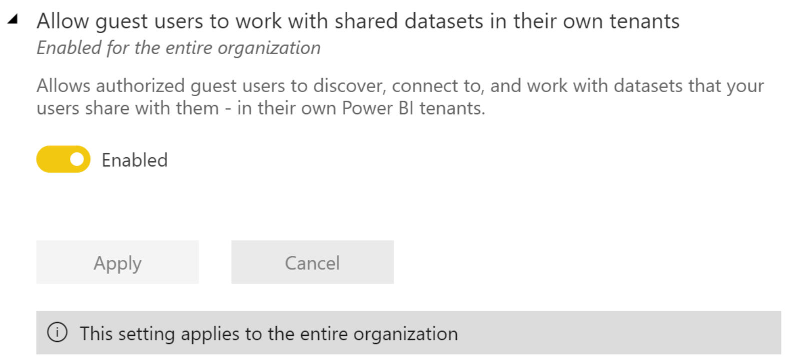
- support for new authentication methods using OAuth 2.0 Personal Access Token (JWT)
- support for advanced filtering with the built-in regular expression filtering feature in Power BI Desktop
- an optional **Catalog** field in the connection dialog. Selecting a catalogue with this field reduces the amount of metadata fetched from the cluster
- an optional safe metadata read field in connection dialog. When enabled, queries will not fail on faulty catalogues, but performance will be slower (by default, this option is disabled).

Cross-tenant dataset sharing in Preview

The new cross-tenant dataset sharing capability is now available in public Preview. This capability allows customers to share their Power BI datasets with external users in a way that allows these them to access the datasets in their own Power BI tenant. This is in-place sharing, meaning that from their own tenant, users may discover and connect to live data from providers. They can then work with these shared datasets in their own tenant and create composite models by meshing the shared datasets with their own internal data. The composite model may then be published on the Power BI Service for reporting purposes.

With in-place sharing, the shared data remains in the provider tenant. Users query the shared datasets directly in the source data systems. Also, they can connect to external datasets using the Direct Query method and build their own composite models and reports on top of the shared data, eliminating the need to manually transfer data between organisations. These composite models built on top of external datasets may also be analysed in Excel. This capability is particularly useful when you want to share data with external partners, customers, subsidiaries, vendors, consultants and other business partners.

Recognising the need to provide governance over external sharing, the dataset sharing capability has two tenant settings that enable Power BI administrators either to disable cross-tenant data sharing entirely or to control which users and / or user groups are allowed to share datasets across tenants. These controls enable organisations to better manage dataset access and sharing in line with their data policies.

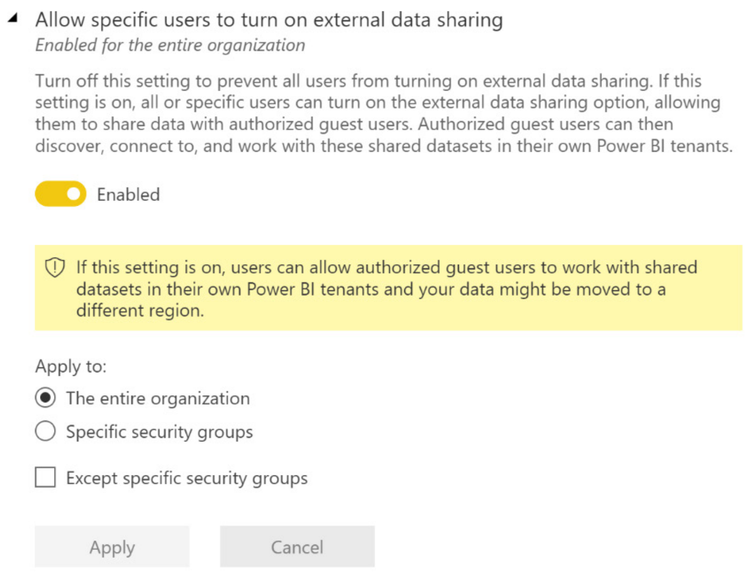

Allow guest users to work with shared datasets in their own tenants

Enabled for the entire organization

Allows authorized guest users to discover, connect to, and work with datasets that your users share with them - in their own Power BI tenants.

Enabled

This setting applies to the entire organization



Allow specific users to turn on external data sharing

Enabled for the entire organization

Turn off this setting to prevent all users from turning on external data sharing. If this setting is on, all or specific users can turn on the external data sharing option, allowing them to share data with authorized guest users. Authorized guest users can then discover, connect to, and work with these shared datasets in their own Power BI tenants.

Enabled

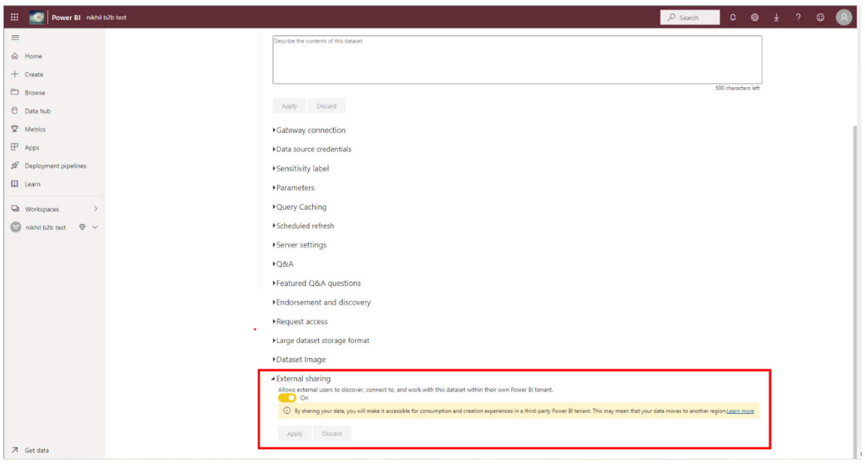
If this setting is on, users can allow authorized guest users to work with shared datasets in their own Power BI tenants and your data might be moved to a different region.

Apply to:

[● The entire organization]
○ Specific security groups

☐ Except specific security groups

When external data sharing is enabled, to initiate dataset sharing with external users, the specified dataset owners need to go to the settings of the specific dataset to be shared and enable external sharing there *(see below)*. This provides additional granular control over which datasets can or cannot be shared externally.



Describe the contents of this dataset.

500 characters left

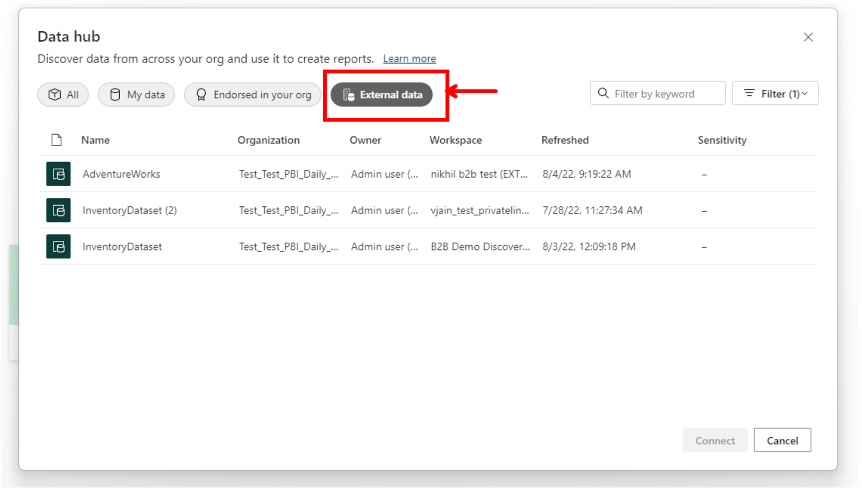
- Gateway connection
- Data source credentials
- Sensitivity label
- Parameters
- Query Caching
- Scheduled refresh
- Server settings
- Q&A
- Featured Q&A questions
- Endorsement and discovery
- Request access
- Large dataset storage format
- Dataset Image
- External sharing

Allows external users to discover, connect to, and work with this dataset within their own Power BI tenant.

On

By sharing your data, you will make it accessible for consumption and creation experiences in a third-party Power BI tenant. This may mean that your data moves to another region. Learn more

Once external sharing is enabled in the dataset settings, data providers may share the datasets externally just as they would with internal users. External users need to have a registered Azure Active Directory (Azure AD) guest account in the provider tenant for them to be able to access the shared datasets. The external users may then discover these datasets in Power BI Desktop, *viz*.



Data hub

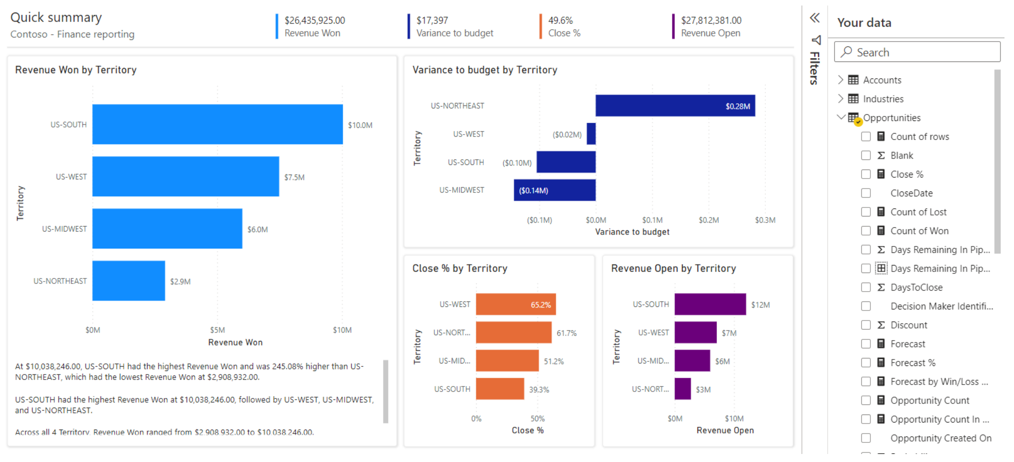
Discover data from across your org and use it to create reports. Learn more

Name	Organization	Owner	Workspace	Refreshed	Sensitivity
AdventureWorks	Test_Test_PBI_Daily_...	Admin user (...	nikhil b2b test (EXT...	8/4/22, 9:19:22 AM	–
InventoryDataset (2)	Test_Test_PBI_Daily_...	Admin user (...	vjain_test_privatelin...	7/28/22, 11:27:34 AM	–
InventoryDataset	Test_Test_PBI_Daily_...	Admin user (...	B2B Demo Discover...	8/3/22, 12:09:18 PM	–

External users can connect to the external dataset and then build composite models by adding either other external datasets or their own internal datasets. These new datasets may then be published to the Power BI Service. Once published, the external users can access the new datasets in the Power BI Service in their own tenant and build further reporting on top of them. These reports can be shared with other users in their own organisation, provided that those users also have the requisite Azure AD guest credentials on the original provider tenant.

Auto-generate reports on existing datasets

In Power BI Service, you may now quickly create an automatically generated report off the majority of datasets in the Power BI Service. This makes it easy for you to explore all the datasets you have access to and can be helpful for jumpstarting report creation in the Power BI Service.

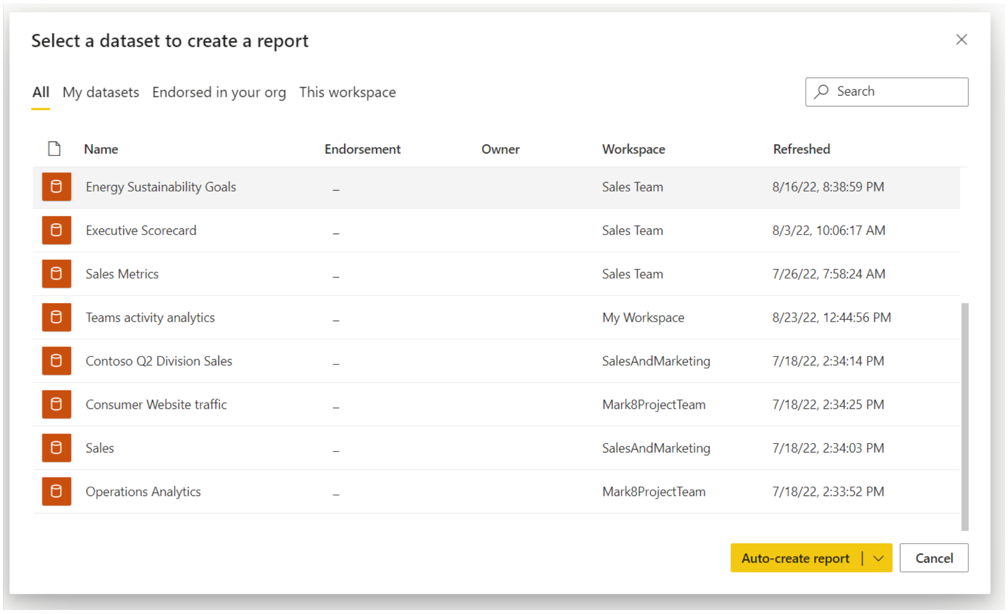


At $10,038,246.00, US-SOUTH had the highest Revenue Won and was 245.08% higher than US-NORTHEAST, which had the lowest Revenue Won at $2,908,932.00.

US-SOUTH had the highest Revenue Won at $10,038,246.00, followed by US-WEST, US-MIDWEST, and US-NORTHEAST.

Across all 4 Territory, Revenue Won ranged from $2,908,932.00 to $10,038,246.00.

On the Create page, if you select the 'Pick a published dataset' option, you'll be able to pick if you want to auto-create a report or, through the split button, if you want to start from a blank report.



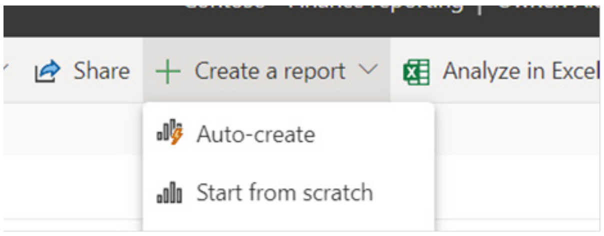
Select a dataset to create a report

All | My datasets | Endorsed in your org | This workspace | Search

Name	Endorsement	Owner	Workspace	Refreshed
Energy Sustainability Goals	–		Sales Team	8/16/22, 8:38:59 PM
Executive Scorecard	–		Sales Team	8/3/22, 10:06:17 AM
Sales Metrics	–		Sales Team	7/26/22, 7:58:24 AM
Teams activity analytics	–		My Workspace	8/23/22, 12:44:56 PM
Contoso Q2 Division Sales	–		SalesAndMarketing	7/18/22, 2:34:14 PM
Consumer Website traffic	–		Mark8ProjectTeam	7/18/22, 2:34:25 PM
Sales	–		SalesAndMarketing	7/18/22, 2:34:03 PM
Operations Analytics	–		Mark8ProjectTeam	7/18/22, 2:33:52 PM

Auto-create report | Cancel

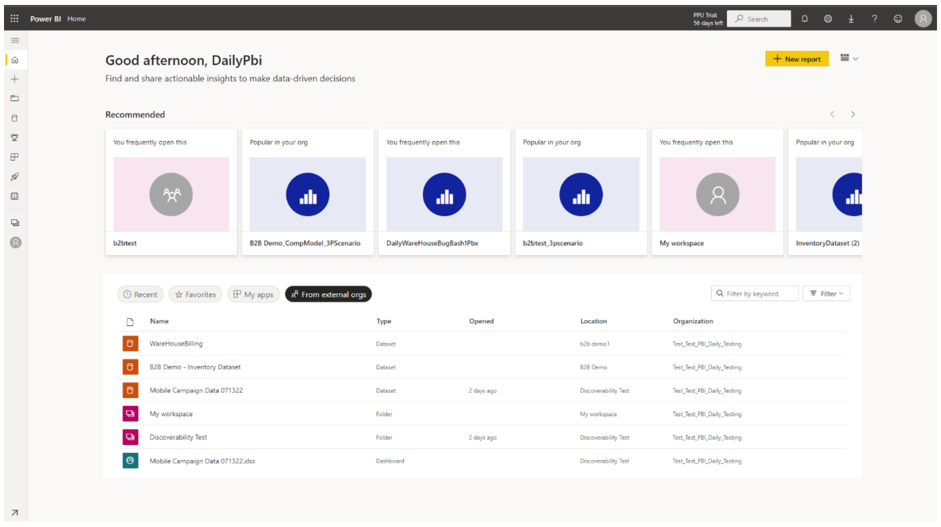
You can also access these options in the Datahub, through the Create a report dropdown, where you can access both the auto-create and from-scratch options.



There are still a few types of datasets Power BI doesn't yet support for auto-generated reports. In these cases, you'll just have the 'blank report' option available to you.

Discoverability feature for B2B content

The latest updates also introduce another new feature which will make Business to Business (B2B) sharing easy for the consumers. Power BI users who are guest users in any other tenant, will now see a new tab on their home page (in home tenant) called 'From external orgs'. This interface will list all artifacts that you have access to as a guest user (shared with you from external tenants). The interface will allow you filter and sort through to find the content easily. You may also see which organisation is sharing a specific document with you. When you click on an artifact on this screen, a new window will open and take you to the relevant provider tenant for that artifact access.



Name	Type	Opened	Location	Organization
WareHouseBilling	Dataset		b2b demo1	Test_Test_PBI_Daily_Testing
B2B Demo - Inventory Dataset	Dataset		B2B Demo	Test_Test_PBI_Daily_Testing
Mobile Campaign Data 071322	Dataset	2 days ago	Discoverability Test	Test_Test_PBI_Daily_Testing
My workspace	Folder		My workspace	Test_Test_PBI_Daily_Testing
Discoverability Test	Folder	2 days ago	Discoverability Test	Test_Test_PBI_Daily_Testing
Mobile Campaign Data 071322.xlsx	Dashboard		Discoverability Test	Test_Test_PBI_Daily_Testing

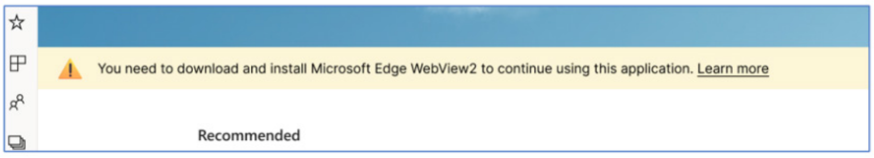
Power BI fonts now available on mobile devices (iOS and Android)

Up until now, a Power BI report created with a font that was not available in the mobile device would display in the Power BI mobile app using the mobile device's default font, rather than the font used by the report creator.

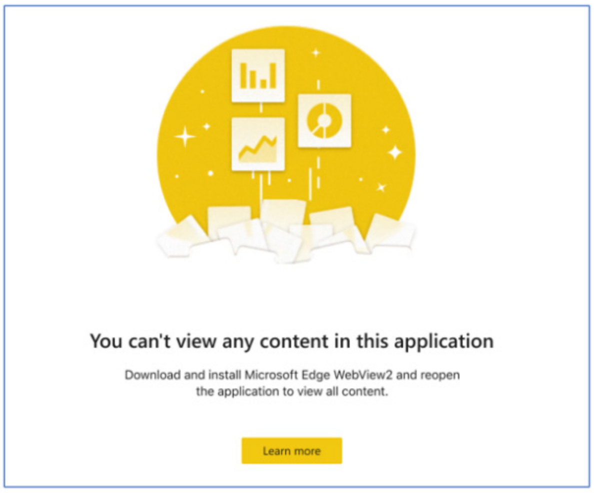
As of this update, the Power BI mobile applications now support the full range of fonts available in Power BI Desktop. This means that in the mobile app, the report will look exactly as you designed it, with the same fonts that you chose when you created the report in Desktop.

Windows app: upgraded browsing experience with WebView2

The Power BI application for Windows is now aligned with the overall Power BI migration to Microsoft Edge WebView2, for faster performance and a better browsing experience all around. To support WebView2, the minimum operating system (OS) required by the Power BI Windows app has changed to Windows 10 version 17763. App upgrades will not be available for Windows devices running on earlier versions.



You need to download and install Microsoft Edge WebView2 to continue using this application. Learn more



You can't view any content in this application

Download and install Microsoft Edge WebView2 and reopen the application to view all content.

If you see one of these messages, you will need to install WebView2.

Power BI component for Vue.js

There is now a new library is available for embedding Power BI content in Vue.js applications. This new library makes it easy to embed Power BI content and leverage Power BI embedded capabilities such as bootstrapping and phased embedding for better performance, applying styles to the embedded component, setting event handlers, and more.

The library is publicly available on npm and GitHub and also includes a demo application which you can run in just a few easy steps.

Power BI and Jupyter integration updates (delayed until October 2022)

The release of this feature has been delayed until later this month.

Shortly, there will be some improvements to the Power BI integration in Jupyter notebooks, making it easier to embed Power BI reports in Jupyter notebooks.

The Powerbi-jupyter library is a Python IPyWidget that brings Power BI embedded capabilities to Jupyter notebooks, allowing users to tell complete stories with their data in their Jupyter notebooks. The library supports embedding existing reports for both viewing and editing, and creating new reports based on data available in Power BI.

The update includes changes to the available authentication methods and the report embedding process, minimising the complexity, and making the embedding process simple and quick.

Export paginated reports by a service principal with a Power BI dataset as a data source

It is now possible to export paginated reports with Power BI datasets as a data source when the caller is a service principal, given that all downstream data sources are SSO-disabled.

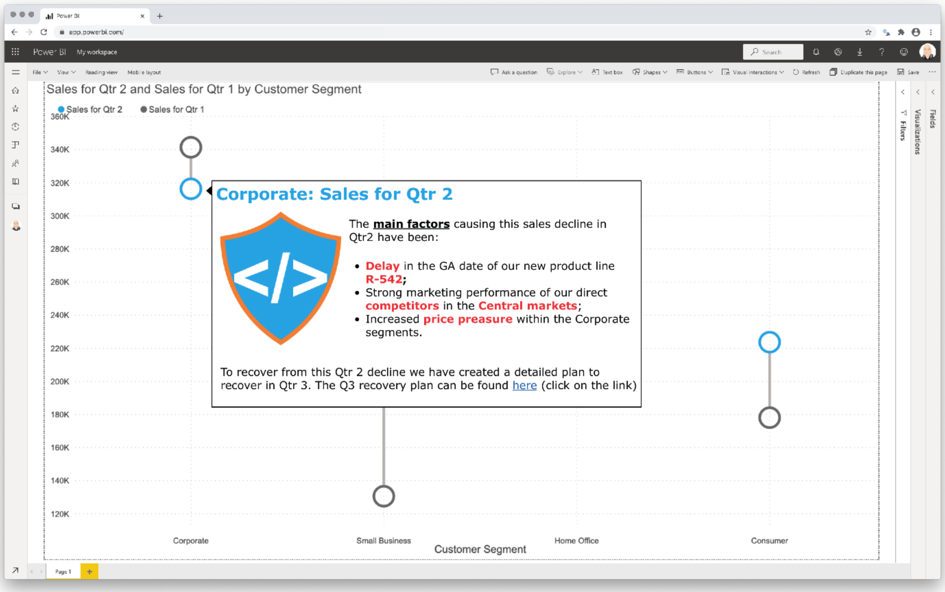
New visuals in AppSource

The following are new visuals this update:

- Calendar by Datanau
- SuperTables by Apps for Power BI
- Selection Slicer by Walnut Innovation.

Shielded HTML Viewer by Nova Silva

All HTML rendered through this Shielded HTML Viewer is sanitised to keep your data safe. This first HTML Viewer Certified by Microsoft now is part of the first visuals you can buy and manage directly through Microsoft.



Corporate: Sales for Qtr 2

The **main factors** causing this sales decline in Qtr2 have been:

- **Delay** in the GA date of our new product line **R-542**;
- Strong marketing performance of our direct **competitors** in the **Central markets**;
- Increased **price preasure** within the Corporate segments.

To recover from this Qtr 2 decline we have created a detailed plan to recover in Qtr 3. The Q3 recovery plan can be found here (click on the link)

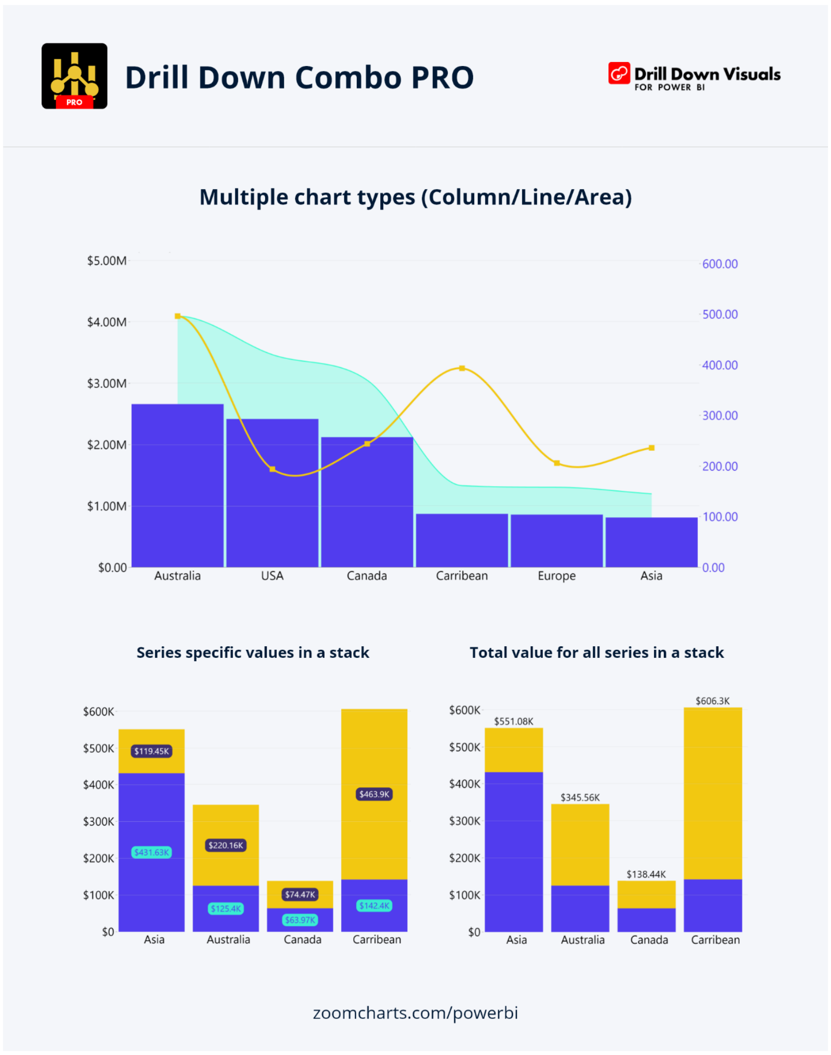
Microsoft announced this new functionality in July 2022. It is currently in Preview and requires Power BI Desktop July 2022 or later. Advantages include:

- obtain fine grained control of the assignment of licences
- (re)assign each licence to a user or group of users whenever you require
- purchase a licence for a single month or a whole year
- make use of the optional auto-renewal feature
- no longer need to enter (or update) a licence key: you manage your visual licences like you manage any other Microsoft licence.

Try the Shielded HTML Viewer now on your own data by downloading it from the AppSource. All features are available for free to evaluate the Shielded HTML Viewer within Power BI Desktop.

Drill Down Combo PRO by ZoomCharts

Drill Down Combo PRO offers plenty of customisation options, enabling you to visualise data in new ways. This visual also features cross-chart filtering along with intuitive on-chart interactions, making the data exploration simpler for even the most inexperienced Power BI user.



The main features include:

- multiple chart types: create column, line, and area charts
- full customisation: modify **x** and **y** axes, the legend, outline and fill settings
- choose normal, 100% proportional or zero-based stacking
- set up to four [4] static and dynamic thresholds to demonstrate targets
- conditional formatting for advanced customisation
- customise multiple series simultaneously with series and value label defaults
- touch-input device friendly.

In particular, there are popular use cases to mention too:

- **Sales and Marketing:** sales strategies, results and campaign-by-campaign marketing metrics
- **Human Resources:** hiring, overtime and efficiency ratios by department
- **Accounting and Finance:** financial performance by region, office or business line
- **Manufacturing:** production and quality metrics.

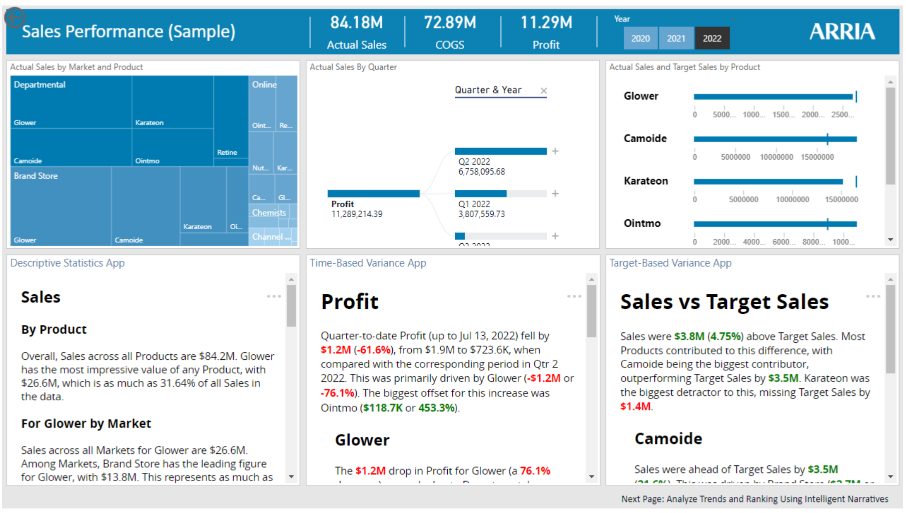
ZoomCharts Drill Down Visuals are known for interactive drilldowns, smooth animations and various customisation options. They support interactions, selections, custom and native ToolTips, filtering, bookmarks and context menu.

This may be downloaded from AppSource.

Intelligent Narratives by Arria NLG

Arria's custom visual for natural language generation has been improved. In the 3.4.0.0 release of Arria for Power BI, you'll find a redesigned UI, significantly increased control over the building of narratives, and formatting improvements for better readability.

Intelligent Narratives by Arria bring NLG to your dashboard, featuring a wide array of analysis types for generating plain-language reporting to complement your visuals. Arria's out-of-the-box narratives are user-configurable and require no coding.



Sales

By Product

Overall, Sales across all Products are $84.2M. Glower has the most impressive value of any Product, with $26.6M, which is as much as 31.64% of all Sales in the data.

For Glower by Market

Sales across all Markets for Glower are $26.6M. Among Markets, Brand Store has the leading figure for Glower, with $13.8M. This represents as much as

Profit

Quarter-to-date Profit (up to Jul 13, 2022) fell by $1.2M (-61.6%), from $1.9M to $723.6K, when compared with the corresponding period in Qtr 2 2022. This was primarily driven by Glower (-$1.2M or -76.1%). The biggest offset for this increase was Ointmo ($118.7K or 453.3%).

Glower

The $1.2M drop in Profit for Glower (a 76.1%

Sales vs Target Sales

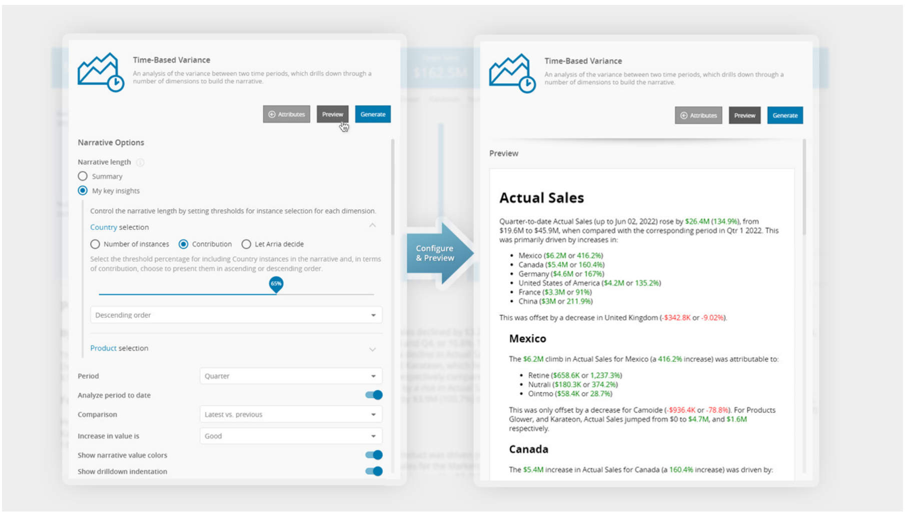
Sales were $3.8M (4.75%) above Target Sales. Most Products contributed to this difference, with Camoide being the biggest contributor, outperforming Target Sales by $3.5M. Karateon was the biggest detractor to this, missing Target Sales by $1.4M.

Camoide

Sales were ahead of Target Sales by $3.5M

You may drill down into all your dashboard's underlying data to tap into insights that you might otherwise miss. From key driver analysis to anomaly detection, trend analysis to correlations, you no longer have to sift through voluminous data to get the answers to your organisation's queries.

Choose the analysis type you need, then configure the narrative according to your business requirements. Narratives are generated instantly, based upon your selections.



Time-Based Variance

An analysis of the variance between two time periods, which drills down through a number of dimensions to build the narrative.

Narrative Options

Narrative length: Summary [✓ My key insights]

Control the narrative length by setting thresholds for instance selection for each dimension.

Country selection: Number of instances [✓ Contribution] Let Arria decide

Select the threshold percentage for including Country instances in the narrative and, in terms of contribution, choose to present them in ascending or descending order.

65%

Descending order

Product selection

Period: Quarter

Analyze period to date: ✓

Comparison: Latest vs. previous

Increase in value is: Good

Show narrative value colors: ✓

Show drilldown indentation: ✓

Preview

Actual Sales

Quarter-to-date Actual Sales (up to Jun 02, 2022) rose by $26.4M (134.9%), from $19.6M to $45.9M, when compared with the corresponding period in Qtr 1 2022. This was primarily driven by increases in:

- Mexico ($6.2M or 416.2%)
- Canada ($5.4M or 160.4%)
- Germany ($4.6M or 167%)
- United States of America ($4.2M or 135.2%)
- France ($3.3M or 91%)
- China ($3M or 211.9%)

This was offset by a decrease in United Kingdom (-$342.8K or -9.02%).

Mexico

The $6.2M climb in Actual Sales for Mexico (a 416.2% increase) was attributable to:

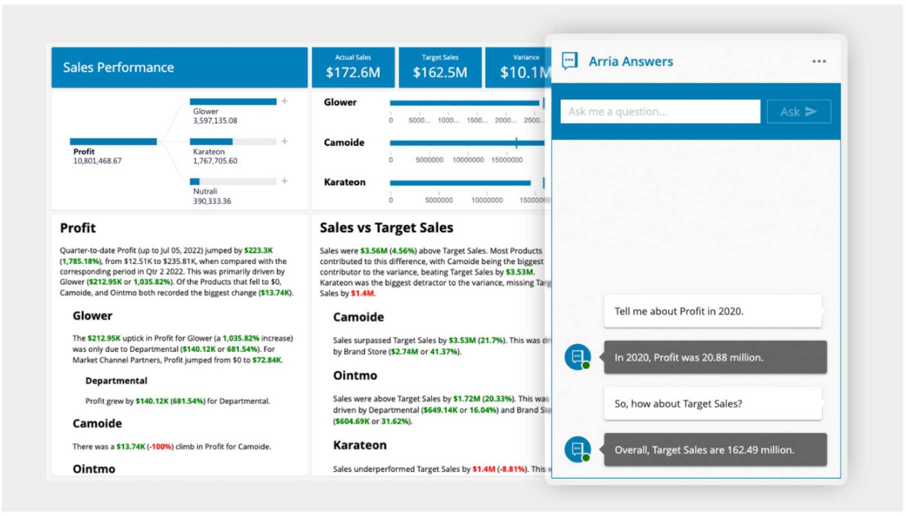
- Retine ($658.6K or 1,237.3%)
- Nutrali ($180.3K or 374.2%)
- Ointmo ($58.4K or 28.7%)

This was only offset by a decrease for Camoide (-$936.4K or -78.8%). For Products Glower, and Karateon, Actual Sales jumped from $0 to $4.7M, and $1.6M respectively.

Canada

The $5.4M increase in Actual Sales for Canada (a 160.4% increase) was driven by:

In addition to advanced Natural Language Generation (NLG), Arria's custom visual also provides Natural Language Queries (NLQs).



Profit

Quarter-to-date Profit (up to Jul 05, 2022) jumped by $223.3K (1,785.18%), from $12.51K to $235.81K, when compared with the corresponding period in Qtr 2 2022. This was primarily driven by Glower ($212.95K or 1,035.82%). Of the Products that fell to $0, Camoide, and Ointmo both recorded the biggest change ($13.74K).

Glower

The $212.95K uptick in Profit for Glower (a 1,035.82% increase) was only due to Departmental ($140.12K or 681.54%). For Market Channel Partners, Profit jumped from $0 to $72.84K.

Departmental

Profit grew by $140.12K (681.54%) for Departmental.

Camoide

There was a $13.74K (-100%) climb in Profit for Camoide.

Ointmo

Sales vs Target Sales

Sales were $3.56M (4.56%) above Target Sales. Most Products contributed to this difference, with Camoide being the biggest contributor to the variance, beating Target Sales by $3.53M. Karateon was the biggest detractor to the variance, missing Tar Sales by $1.4M.

Camoide

Sales surpassed Target Sales by $3.53M (21.7%). This was d by Brand Store ($2.74M or 41.37%).

Ointmo

Sales were above Target Sales by $1.72M (20.33%). This wa driven by Departmental ($649.14K or 16.04%) and Brand S ($604.69K or 31.62%).

Karateon

Sales underperformed Target Sales by $1.4M (-8.81%). This

Tell me about Profit in 2020.

In 2020, Profit was 20.88 million.

So, how about Target Sales?

Overall, Target Sales are 162.49 million.

Again, this may be downloaded from AppSource.

Paginated reports data Preview

Recently, Microsoft introduced 'data preview' as a new action on the dataset details page. 'Data preview' enables you to view selected tables and columns from the dataset and provides entry points for exporting the data to supported file formats, customising it as a formatted table, and saving it as a paginated report.

'Show query' is a new capability added to 'data preview'. It allows you to copy to the clipboard the DAX query used to create the table preview. This may be useful for those who want to reuse the DAX query for future actions.

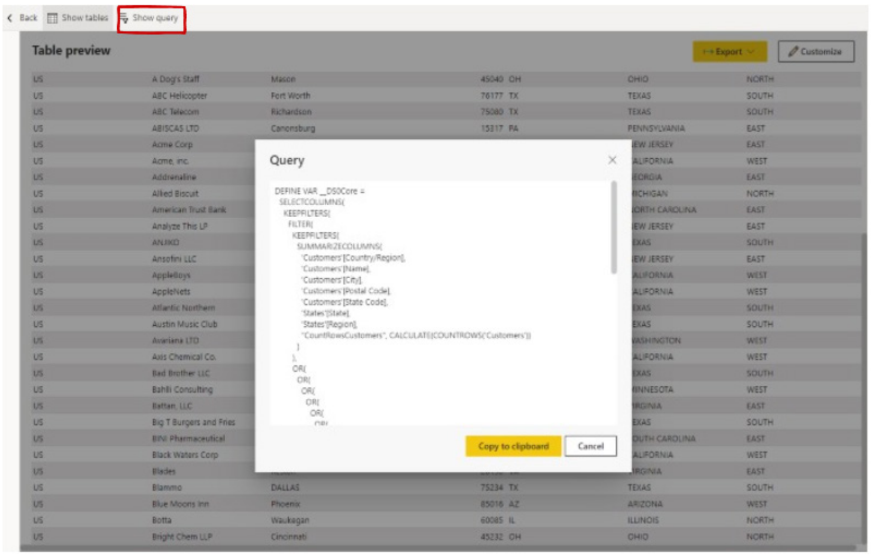


Table preview

Query

```
DEFINE VAR __DS0Core =
  SELECTCOLUMNS(
   KEEPFILTERS(
    FILTER(
     KEEPFILTERS(
      SUMMARIZECOLUMNS(
        'Customers'[Country/Region],
        'Customers'[Name],
        'Customers'[City],
        'Customers'[Postal Code],
        'Customers'[State Code],
        'States'[State],
        'States'[Region],
        "CountRowsCustomers", CALCULATE(COUNTROWS('Customers'))
      )
     ),
     OR(
      OR(
       OR(
        OR(
         OR(
```

US	A Dog's Staff	Mason	45040	OH	OHIO	NORTH
US	ABC Helicopter	Fort Worth	76177	TX	TEXAS	SOUTH
US	ABC Telecom	Richardson	75080	TX	TEXAS	SOUTH
US	ABISCAS LTD	Canonsburg	15317	PA	PENNSYLVANIA	EAST
US	Acme Corp				EW JERSEY	EAST
US	Acme, inc.				ALIFORNIA	WEST
US	Addrenaline				EORGIA	EAST
US	Allied Biscuit				ICHIGAN	NORTH
US	American Trust Bank				ORTH CAROLINA	EAST
US	Analyze This LP				EW JERSEY	EAST
US	ANJIKO				EXAS	SOUTH
US	Ansofini LLC				EW JERSEY	EAST
US	AppleBoys				ALIFORNIA	WEST
US	AppleNets				ALIFORNIA	WEST
US	Atlantic Northern				EXAS	SOUTH
US	Austin Music Club				EXAS	SOUTH
US	Avariana LTD				ASHINGTON	WEST
US	Axis Chemical Co.				ALIFORNIA	WEST
US	Bad Brother LLC				EXAS	SOUTH
US	Bahli Consulting				INNESOTA	WEST
US	Battan, LLC				IRGINIA	EAST
US	Big T Burgers and Fries				EXAS	SOUTH
US	BINI Pharmaceutical				OUTH CAROLINA	EAST
US	Black Waters Corp				ALIFORNIA	WEST
US	Blades				IRGINIA	EAST
US	Blammo	DALLAS	75234	TX	TEXAS	SOUTH
US	Blue Moons Inn	Phoenix	85016	AZ	ARIZONA	WEST
US	Botta	Waukegan	60085	IL	ILLINOIS	NORTH
US	Bright Chem LLP	Cincinnati	45232	OH	OHIO	NORTH

Column resizing, an additional capability, has been added to the data preview table. With a rendered table, a user will be able to resize the width of the columns through a drag handle. This functionality makes the table preview more readable, especially for long column input values.

To demonstrate, compare the old size

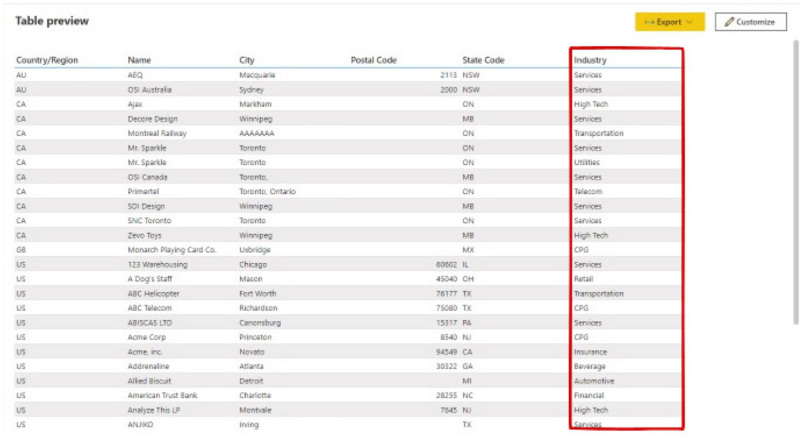


Table preview

Country/Region	Name	City	Postal Code	State Code	Industry
AU	AEQ	Macquarie	2113	NSW	Services
AU	OSI Australia	Sydney	2000	NSW	Services
CA	Ajax	Markham		ON	High Tech
CA	Decore Design	Winnipeg		MB	Services
CA	Montreal Railway	AAAAAAA		ON	Transportation
CA	Mr. Sparkle	Toronto		ON	Services
CA	Mr. Sparkle	Toronto		ON	Utilities
CA	OSI Canada	Toronto,		MB	Services
CA	Primertel	Toronto, Ontario		ON	Telecom
CA	SDI Design	Winnipeg		MB	Services
CA	SNC Toronto	Toronto		ON	Services
CA	Zevo Toys	Winnipeg		MB	High Tech
GB	Monarch Playing Card Co.	Uxbridge		MX	CPG
US	123 Warehousing	Chicago	60602	IL	Services
US	A Dog's Staff	Mason	45040	OH	Retail
US	ABC Helicopter	Fort Worth	76177	TX	Transportation
US	ABC Telecom	Richardson	75080	TX	CPG
US	ABISCAS LTD	Canonsburg	15317	PA	Services
US	Acme Corp	Princeton	8540	NJ	CPG
US	Acme, inc.	Novato	94549	CA	Insurance
US	Addrenaline	Atlanta	30322	GA	Beverage
US	Allied Biscuit	Detroit		MI	Automotive
US	American Trust Bank	Charlotte	28255	NC	Financial
US	Analyze This LP	Montvale	7645	NJ	High Tech
US	ANJIKO	Irving		TX	Services

with the new size:

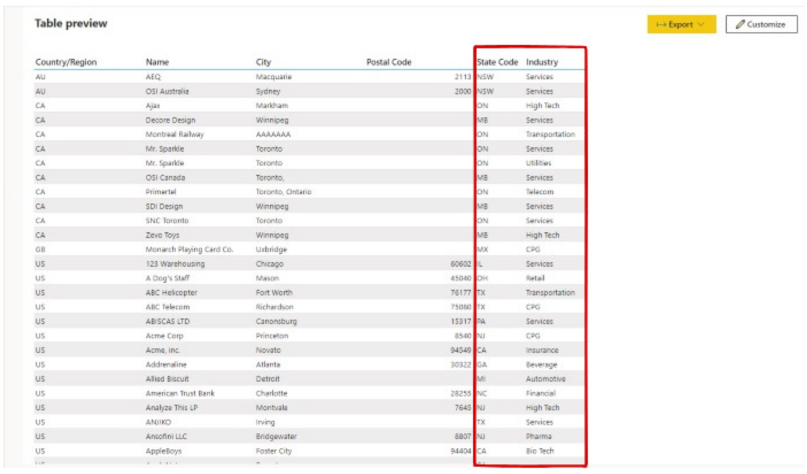


Table preview

Country/Region	Name	City	Postal Code	State Code	Industry
AU	AEQ	Macquarie	2113	NSW	Services
AU	OSI Australia	Sydney	2000	NSW	Services
CA	Ajax	Markham		ON	High Tech
CA	Decore Design	Winnipeg		MB	Services
CA	Montreal Railway	AAAAAAA		ON	Transportation
CA	Mr. Sparkle	Toronto		ON	Services
CA	Mr. Sparkle	Toronto		ON	Utilities
CA	OSI Canada	Toronto,		MB	Services
CA	Primertel	Toronto, Ontario		ON	Telecom
CA	SDI Design	Winnipeg		MB	Services
CA	SNC Toronto	Toronto		ON	Services
CA	Zevo Toys	Winnipeg		MB	High Tech
GB	Monarch Playing Card Co.	Uxbridge		MX	CPG
US	123 Warehousing	Chicago	60602	IL	Services
US	A Dog's Staff	Mason	45040	OH	Retail
US	ABC Helicopter	Fort Worth	76177	TX	Transportation
US	ABC Telecom	Richardson	75080	TX	CPG
US	ABISCAS LTD	Canonsburg	15317	PA	Services
US	Acme Corp	Princeton	8540	NJ	CPG
US	Acme, inc.	Novato	94549	CA	Insurance
US	Addrenaline	Atlanta	30322	GA	Beverage
US	Allied Biscuit	Detroit		MI	Automotive
US	American Trust Bank	Charlotte	28255	NC	Financial
US	Analyze This LP	Montvale	7645	NJ	High Tech
US	ANJIKO	Irving		TX	Services
US	Ansofini LLC	Bridgewater	8807	NJ	Pharma
US	AppleBoys	Foster City	94404	CA	Bio Tech

That's it for this month. See you in November!

New Features for Excel

It's been a busy month for Excel with recent new additions to Insider rolling out to the Current Channel, and the new IMAGE function too. But that's not even half of it. The full list of updates for this month is as follows:

Excel for the web

- New Excel functions
- Power Query Group operations
- Improvements to the connected Power BI experience
- Add and edit rich text formatting
- Sort by colour or icon from AutoFilter menu
- Edit files with legacy data connections
- Edit files with legacy Shared Workbook feature
- Delete chart elements
- Multiline Formula bar

Excel for Windows

- New Excel functions
- **IMAGE** function (Insiders Beta)
- Improvements to the connected Power BI experience (Insiders Beta)
- Show Changes (Insiders CC Preview)

Android

- New Excel functions
- **IMAGE** function (Insiders Beta)

iOS

- New Excel functions
- **IMAGE** function (Insiders Beta).

Yes, there might be a little repetition here. Yes, there might be a little repetition here. Yes, there might be a little repetition here. Yes, there might be a little repetition here. Yes, there might be a little repetition here…

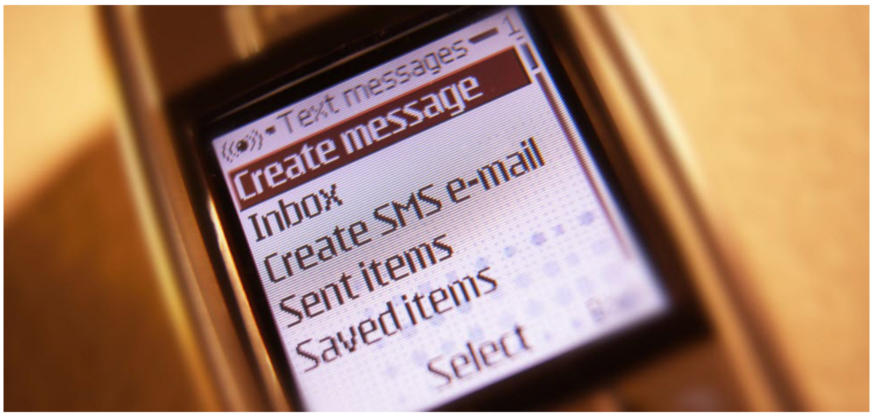
Let's plough through.

New Excel functions

Since Wednesday 17 August, 14 new Excel functions have been rolling out to all subscription users in the Current Channel for <u>all</u> endpoints. The full list of these functions (and what they do) is repeated here for information.

Around for a while now, the **updated** text functions were originally available to Beta channel users starting with build 15316.20000. Some changed, but all remain useful…

The TEXTBEFORE function



The **TEXTBEFORE** function returns the string of text that occurs before a given substring (*i.e.* a character or set of characters) in that string. It is the opposite of the **TEXTAFTER** function. **TEXTBEFORE** has the following syntax:

TEXTBEFORE(text, delimiter, [instance number], [match mode], [match end], [if not found])

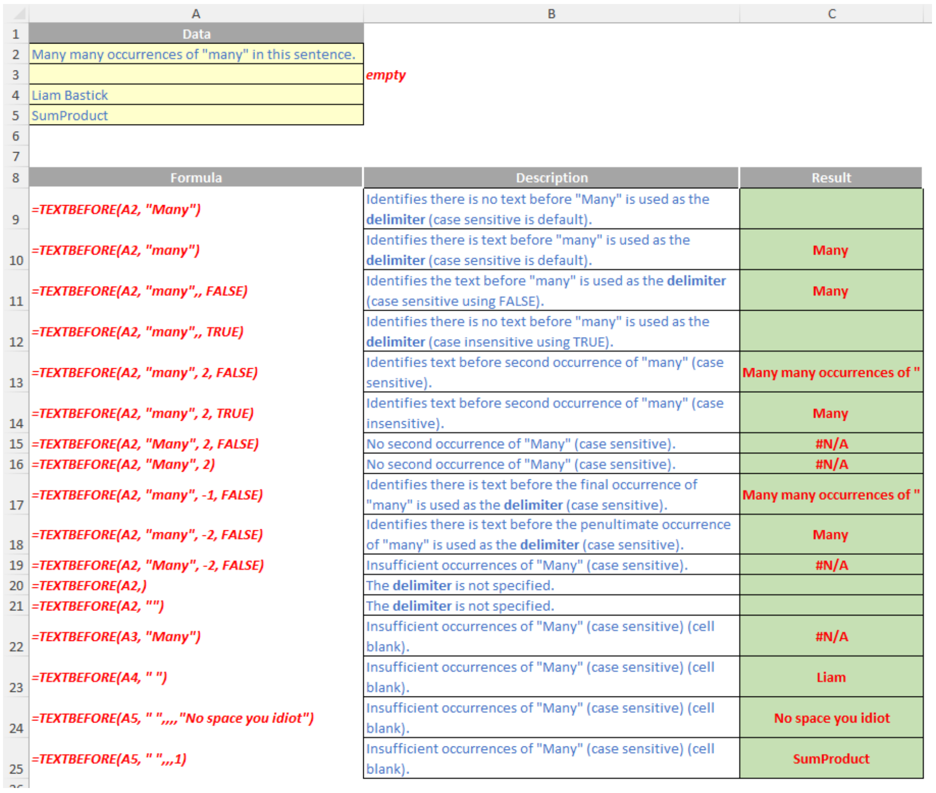
The **TEXTBEFORE** function has the following arguments:

- **text:** this is required and represents the text string you are searching within. Wildcard characters are not allowed
- **delimiter:** this is also required and represents the text in the **text** string that marks the point before which you wish to extract
- **instance number:** this is the first optional argument and denotes the **n**th instance of the delimiter before which you wish to extract. By default, this is equal to one [1]. If a negative number is used here, the function starts searching for the **delimiter** from the end rather than the beginning
- **match mode:** this is the second optional argument, and determines whether the text search is case-sensitive. The default is case-sensitive. If used, you should enter one of the following two [2] values:
  - **0:** case sensitive
  - **1:** case insensitive
- **match end:** also optional, this treats the end of the **text** as a **delimiter**. By default this is viewed as an exact match. If used, you should enter one of the following two [2] values:
  - **0:** do not match the **delimiter** against the end of the text
  - **1:** match the **delimiter** against the end of the **text**
- **if not found:** this too is an optional argument and provides the value should no match be found. If not used and no match is found, *#N/A* will be returned.

It should be further noted that:

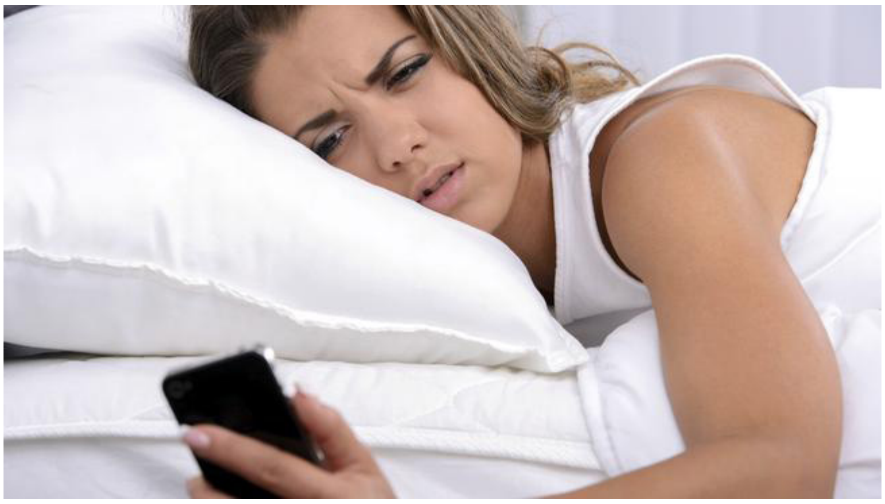
- when searching with an empty **delimiter** value, **TEXTBEFORE** matches immediately. It returns empty **text** when searching from the front (if **instance number** is positive) and the entire text when searching from the end (if **instance number** is negative)
- Excel returns an *#VALUE!* error if the **instance number** is zero (the default is one) or if the **instance number** is greater than the length of the **text**
- Excel returns an *#N/A* error if the **delimiter** does not occur within the text
- Excel returns an *#N/A* error if the **instance number** is greater than the number of occurrences of the **delimiter** within the **text**.

Please see the examples below:



	A	B	C
1	Data		
2	Many many occurrences of "many" in this sentence.		
3		*empty*	
4	Liam Bastick		
5	SumProduct		
6			
7			
8	Formula	Description	Result
9	*=TEXTBEFORE(A2, "Many")*	Identifies there is no text before "Many" is used as the **delimiter** (case sensitive is default).	
10	*=TEXTBEFORE(A2, "many")*	Identifies there is text before "many" is used as the **delimiter** (case sensitive is default).	Many
11	*=TEXTBEFORE(A2, "many",, FALSE)*	Identifies the text before "many" is used as the **delimiter** (case sensitive using FALSE).	Many
12	*=TEXTBEFORE(A2, "many",, TRUE)*	Identifies there is no text before "many" is used as the **delimiter** (case insensitive using TRUE).	
13	*=TEXTBEFORE(A2, "many", 2, FALSE)*	Identifies text before second occurrence of "many" (case sensitive).	Many many occurrences of "
14	*=TEXTBEFORE(A2, "many", 2, TRUE)*	Identifies text before second occurrence of "many" (case insensitive).	Many
15	*=TEXTBEFORE(A2, "Many", 2, FALSE)*	No second occurrence of "Many" (case sensitive).	#N/A
16	*=TEXTBEFORE(A2, "Many", 2)*	No second occurrence of "Many" (case sensitive).	#N/A
17	*=TEXTBEFORE(A2, "many", -1, FALSE)*	Identifies there is text before the final occurrence of "many" is used as the **delimiter** (case sensitive).	Many many occurrences of "
18	*=TEXTBEFORE(A2, "many", -2, FALSE)*	Identifies there is text before the penultimate occurrence of "many" is used as the **delimiter** (case sensitive).	Many
19	*=TEXTBEFORE(A2, "Many", -2, FALSE)*	Insufficient occurrences of "Many" (case sensitive).	#N/A
20	*=TEXTBEFORE(A2,)*	The **delimiter** is not specified.	
21	*=TEXTBEFORE(A2, "")*	The **delimiter** is not specified.	
22	*=TEXTBEFORE(A3, "Many")*	Insufficient occurrences of "Many" (case sensitive) (cell blank).	#N/A
23	*=TEXTBEFORE(A4, " ")*	Insufficient occurrences of "Many" (case sensitive) (cell blank).	Liam
24	*=TEXTBEFORE(A5, " ",,,,"No space you idiot")*	Insufficient occurrences of "Many" (case sensitive) (cell blank).	No space you idiot
25	*=TEXTBEFORE(A5, " ",,,1)*	Insufficient occurrences of "Many" (case sensitive) (cell blank).	SumProduct

The TEXTAFTER function



The **TEXTAFTER** function returns the string of text that occurs after a given substring (*i.e.* a character or set of characters) in that string. It is the opposite of the **TEXTBEFORE** function. **TEXTAFTER** has the following syntax:

TEXTAFTER(text, delimiter, [instance number], [match mode], [match end], [if not found])

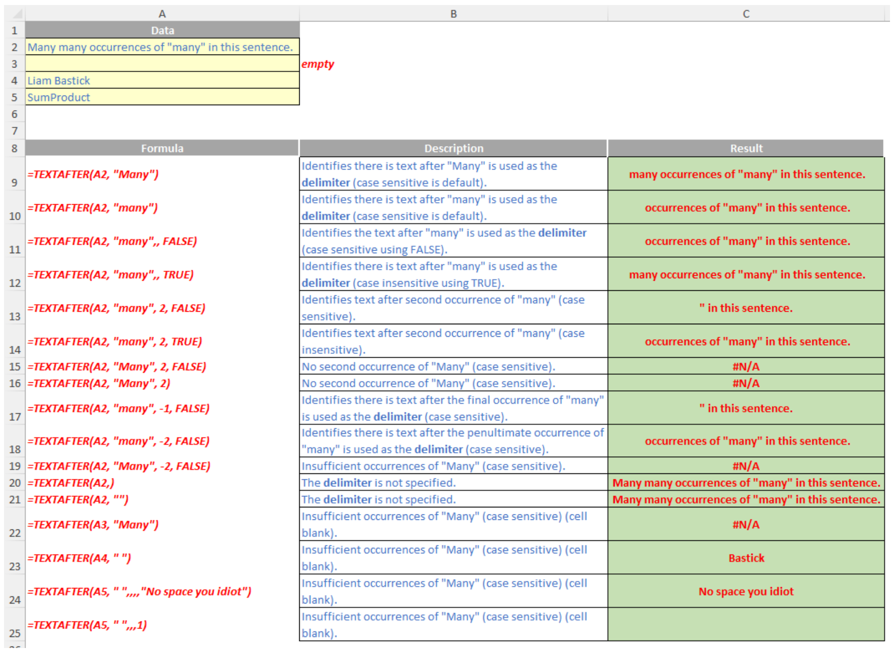
The **TEXTAFTER** function has the following arguments:

- **text:** this is required and represents the text string you are searching within. Wildcard characters are not allowed
- **delimiter:** this is also required and represents the text in the **text** string that marks the point after which you wish to extract
- **instance number:** this is the first optional argument and denotes the **n**th instance of the **delimiter** after which you wish to extract. By default, this is equal to one [1]. If a negative number is used here, the function starts searching for the **delimiter** from the end rather than the beginning
- **match mode:** this is the second optional argument, and determines whether the **text** search is case-sensitive. The default is case-sensitive. If used, you should enter one of the following two [2] values:
  - **0:** case sensitive
  - **1:** case insensitive
- **match end:** also optional, this treats the end of the **text** as a **delimiter**. By default this is viewed as an exact match. If used, you should enter one of the following two [2] values:
  - **0:** do not match the **delimiter** against the end of the **text**
  - **1:** match the **delimiter** against the end of the **text**
- **if not found:** this too is an optional argument and provides the value should no match be found. If not used and no match is found, *#N/A* will be returned.

It should be further noted that:

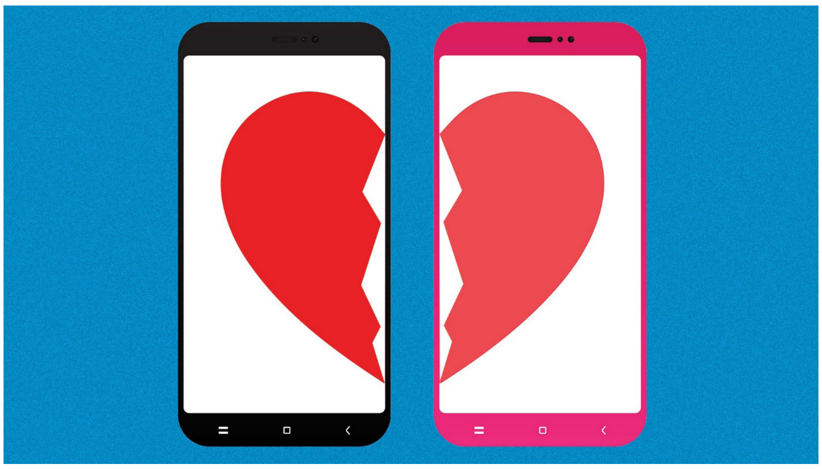
- when searching with an empty **delimiter** value, **TEXTAFTER** matches immediately. It returns the entire **text** when searching from the front (if **instance number** is positive) and empty **text** when searching from the end (if **instance number** is negative)
- Excel returns an *#VALUE!* error if the **instance number** is zero (the default is one) or if the **instance number** is greater than the length of the **text**
- Excel returns an *#N/A* error if the **delimiter** does not occur within the **text**
- Excel returns an *#N/A* error if the **instance number** is greater than the number of occurrences of the **delimiter** within the **text**.

Please see relevant examples below:



	A	B	C
1	Data		
2	Many many occurrences of "many" in this sentence.		
3		*empty*	
4	Liam Bastick		
5	SumProduct		
6			
7			
8	Formula	Description	Result
9	*=TEXTAFTER(A2, "Many")*	Identifies there is text after "Many" is used as the **delimiter** (case sensitive is default).	**many occurrences of "many" in this sentence.**
10	*=TEXTAFTER(A2, "many")*	Identifies there is text after "many" is used as the **delimiter** (case sensitive is default).	**occurrences of "many" in this sentence.**
11	*=TEXTAFTER(A2, "many",, FALSE)*	Identifies the text after "many" is used as the **delimiter** (case sensitive using FALSE).	**occurrences of "many" in this sentence.**
12	*=TEXTAFTER(A2, "many",, TRUE)*	Identifies there is text after "many" is used as the **delimiter** (case insensitive using TRUE).	**many occurrences of "many" in this sentence.**
13	*=TEXTAFTER(A2, "many", 2, FALSE)*	Identifies text after second occurrence of "many" (case sensitive).	**" in this sentence.**
14	*=TEXTAFTER(A2, "many", 2, TRUE)*	Identifies text after second occurrence of "many" (case insensitive).	**occurrences of "many" in this sentence.**
15	*=TEXTAFTER(A2, "Many", 2, FALSE)*	No second occurrence of "Many" (case sensitive).	**#N/A**
16	*=TEXTAFTER(A2, "Many", 2)*	No second occurrence of "Many" (case sensitive).	**#N/A**
17	*=TEXTAFTER(A2, "many", -1, FALSE)*	Identifies there is text after the final occurrence of "many" is used as the **delimiter** (case sensitive).	**" in this sentence.**
18	*=TEXTAFTER(A2, "many", -2, FALSE)*	Identifies there is text after the penultimate occurrence of "many" is used as the **delimiter** (case sensitive).	**occurrences of "many" in this sentence.**
19	*=TEXTAFTER(A2, "Many", -2, FALSE)*	Insufficient occurrences of "Many" (case sensitive).	**#N/A**
20	*=TEXTAFTER(A2,)*	The **delimiter** is not specified.	**Many many occurrences of "many" in this sentence.**
21	*=TEXTAFTER(A2, "")*	The **delimiter** is not specified.	**Many many occurrences of "many" in this sentence.**
22	*=TEXTAFTER(A3, "Many")*	Insufficient occurrences of "Many" (case sensitive) (cell blank).	**#N/A**
23	*=TEXTAFTER(A4, " ")*	Insufficient occurrences of "Many" (case sensitive) (cell blank).	**Bastick**
24	*=TEXTAFTER(A5, " ",,,,"No space you idiot")*	Insufficient occurrences of "Many" (case sensitive) (cell blank).	**No space you idiot**
25	*=TEXTAFTER(A5, " ",,,1)*	Insufficient occurrences of "Many" (case sensitive) (cell blank).	

The TEXTSPLIT function



The **TEXTSPLIT** function is intended to work like the Text to Columns button on the Data tab of the Ribbon, almost like the “inverse” of the **TEXTJOIN** function. It allows you to split a given text across rows or down columns. **TEXTSPLIT** has the following syntax:

TEXTSPLIT(text, [column delimiter], [row delimiter], [ignore empty], [match mode], [pad with])

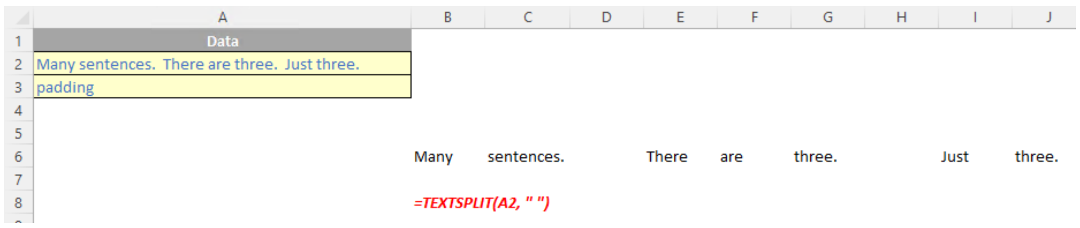
The **TEXTSPLIT** function has the following arguments:

- **text:** this is required and represents the text string you wish to split
- **column delimiter:** this is optional and denotes one or more characters that specify where to spill the text across columns
- **row delimiter:** this is optional and denotes one or more characters that specify where to spill the text down rows
- **ignore empty:** another optional argument, you should specify TRUE to create an empty cell when two delimiters are used. This argument defaults to FALSE, which means don't create an empty cell
- **match mode:** this is the second optional argument, and determines whether the text search is case-sensitive. The default is case-sensitive. If used, you should enter one of the following two [2] values:
    - o **0:** case sensitive
    - o **1:** case insensitive
- **pad with:** not to be confused with Pad Thai, this final optional argument “pads” the resulting text range where cells would otherwise be blank. The default is *N/A*.

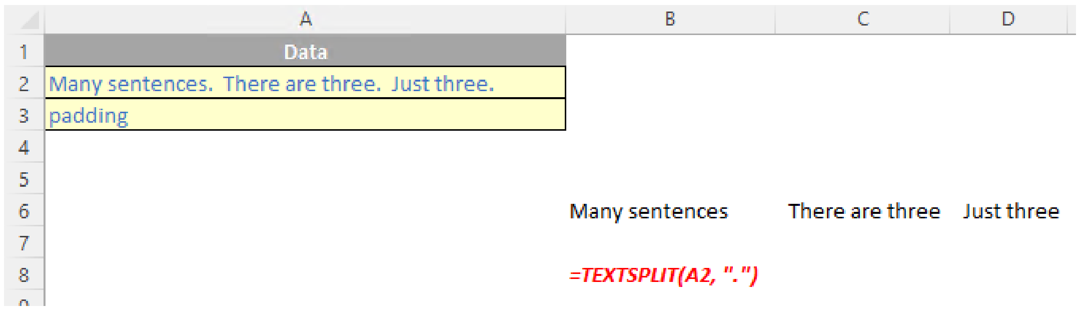
Please note that you will need to update any **TEXTSPLIT** functions previously written if you used the **pad with** argument, as this argument has moved from the fifth to the sixth argument.

If there is more than one delimiter (row or column), then an array constant must be used. For example, to split by both a comma (,) and a period (full stop, .), use **=TEXTSPLIT(text, {",", "."})**.

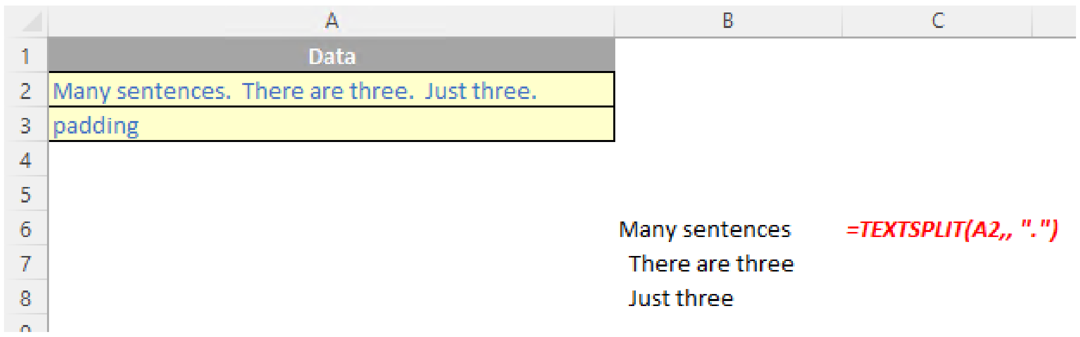
Just for a change, some more examples:



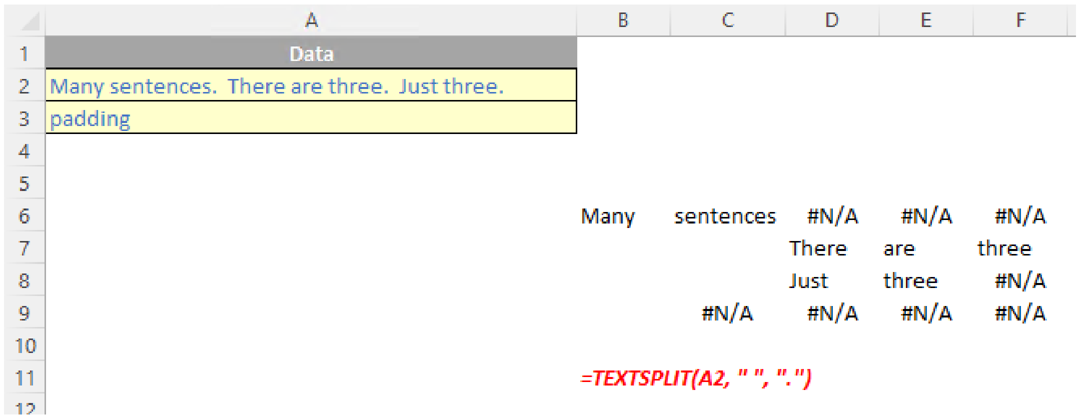
	A	B	C	D	E	F	G	H	I	J
1	Data									
2	Many sentences. There are three. Just three.									
3	padding									
4										
5										
6		Many	sentences.		There	are	three.		Just	three.
7										
8		*=TEXTSPLIT(A2, " ")*								



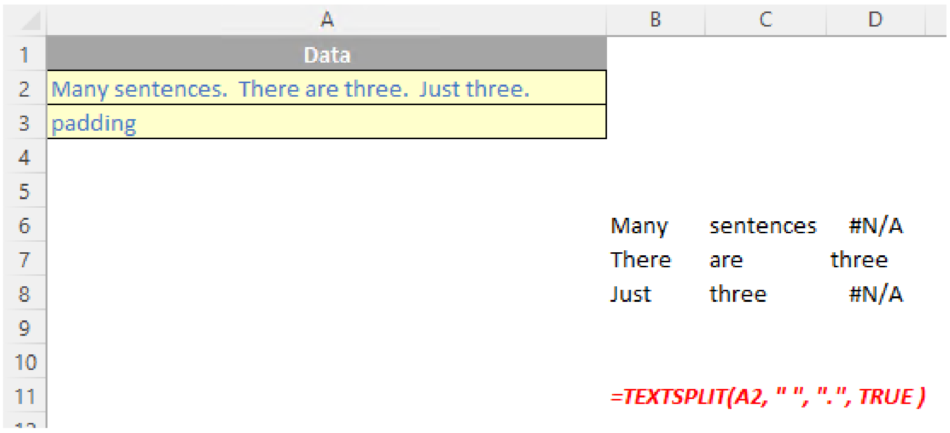
	A	B	C	D
1	Data			
2	Many sentences. There are three. Just three.			
3	padding			
4				
5				
6		Many sentences	There are three	Just three
7				
8		*=TEXTSPLIT(A2, ".")*		



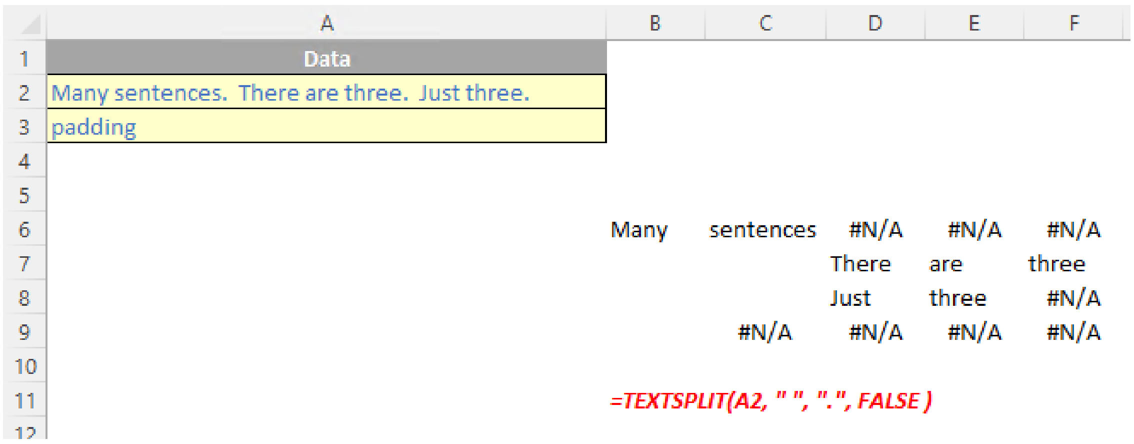
	A	B	C
1	Data		
2	Many sentences. There are three. Just three.		
3	padding		
4			
5			
6		Many sentences	*=TEXTSPLIT(A2,, ".")*
7		There are three	
8		Just three	



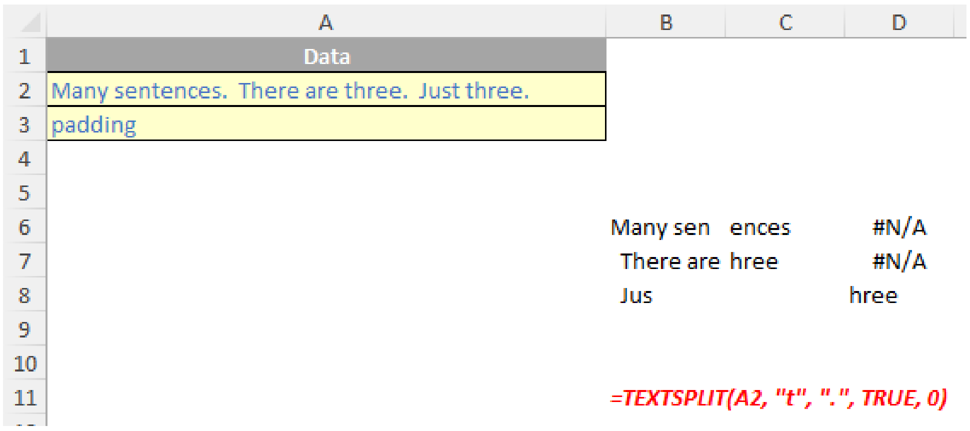
	A	B	C	D	E	F
1	Data					
2	Many sentences. There are three. Just three.					
3	padding					
4						
5						
6		Many	sentences	#N/A	#N/A	#N/A
7				There	are	three
8				Just	three	#N/A
9			#N/A	#N/A	#N/A	#N/A
10						
11		*=TEXTSPLIT(A2, " ", ".")*				



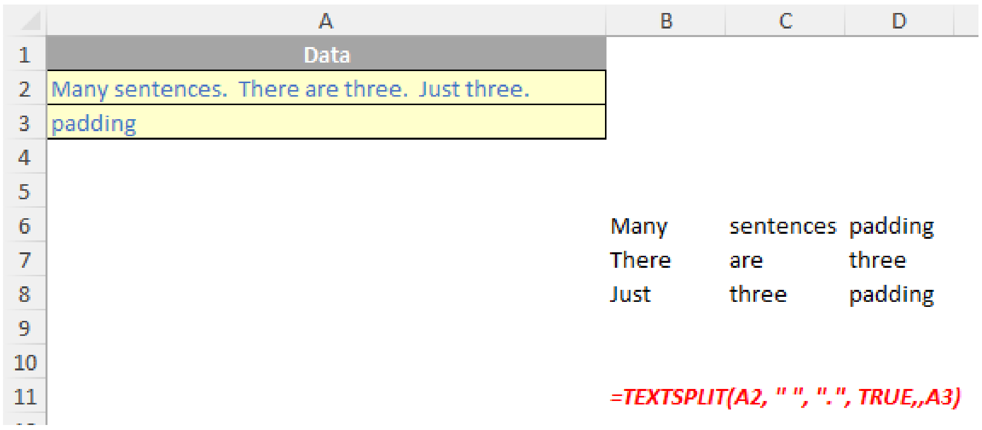
	A	B	C	D
1	Data			
2	Many sentences. There are three. Just three.			
3	padding			
4				
5				
6		Many	sentences	#N/A
7		There	are	three
8		Just	three	#N/A
9				
10				
11		=TEXTSPLIT(A2, " ", ".", TRUE)		



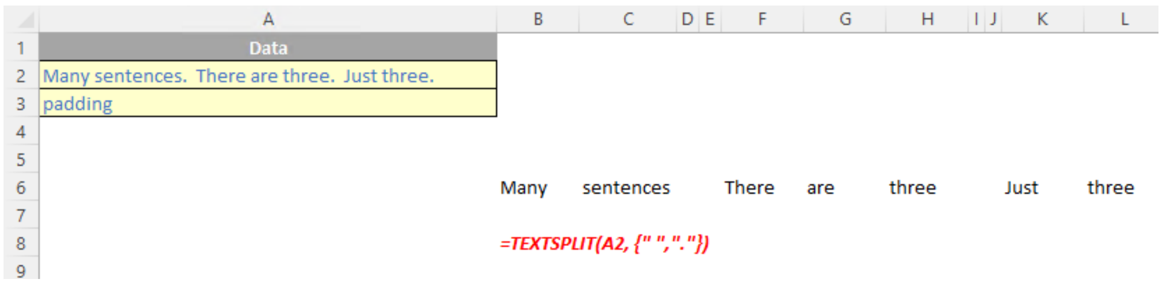
	A	B	C	D	E	F
1	Data					
2	Many sentences. There are three. Just three.					
3	padding					
4						
5						
6		Many	sentences	#N/A	#N/A	#N/A
7				There	are	three
8				Just	three	#N/A
9			#N/A	#N/A	#N/A	#N/A
10						
11		=TEXTSPLIT(A2, " ", ".", FALSE)				



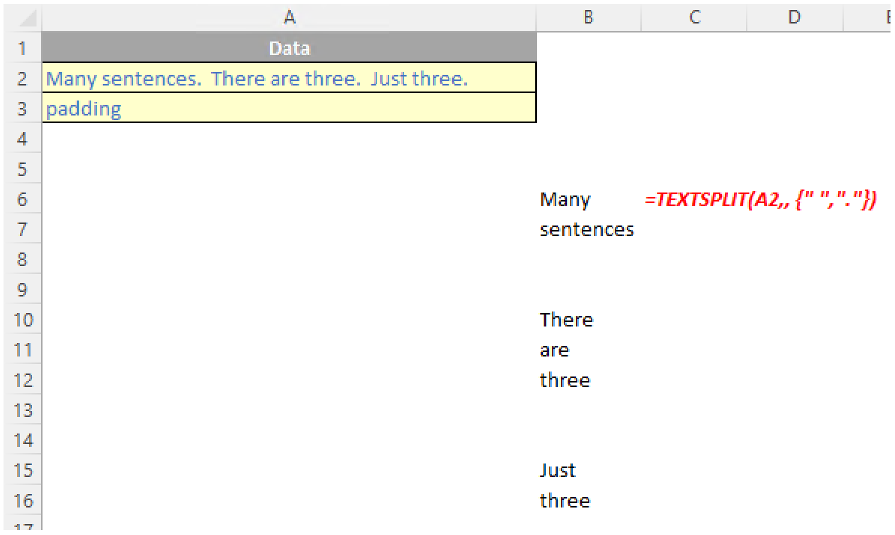
	A	B	C	D
1	Data			
2	Many sentences. There are three. Just three.			
3	padding			
4				
5				
6		Many sen	ences	#N/A
7		There are	hree	#N/A
8		Jus		hree
9				
10				
11		=TEXTSPLIT(A2, "t", ".", TRUE, 0)		



	A	B	C	D
1	Data			
2	Many sentences. There are three. Just three.			
3	padding			
4				
5				
6		Many	sentences	padding
7		There	are	three
8		Just	three	padding
9				
10				
11		=TEXTSPLIT(A2, " ", ".", TRUE,,A3)		



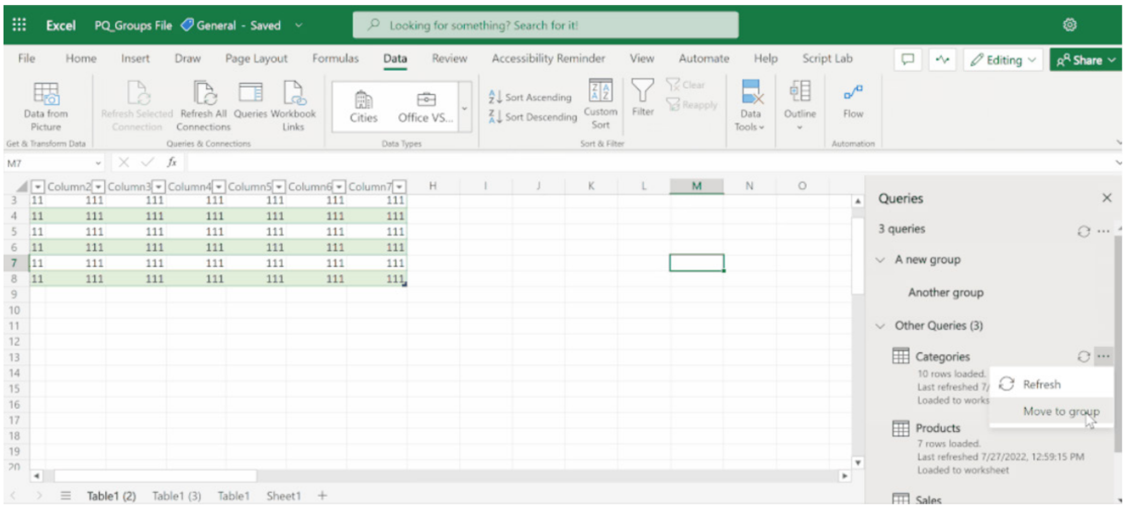
	A	B	C	D	E	F	G	H	I	J	K	L
1	Data											
2	Many sentences. There are three. Just three.											
3	padding											
4												
5												
6		Many	sentences			There	are	three			Just	three
7												
8		=TEXTSPLIT(A2, {" ","."})										
9												



	A	B	C	D
1	Data			
2	Many sentences. There are three. Just three.			
3	padding			
4				
5				
6		Many	=TEXTSPLIT(A2,, {" ","."})	
7		sentences		
8				
9				
10		There		
11		are		
12		three		
13				
14				
15		Just		
16		three		

Power Query Group operations

You can now perform various operations on Power Query groups in Excel for the web to better organise your queries and easily consume the data on the queries pane. This has now been extended to all Office 365 subscribers with Business or Enterprise plans. We detailed this last month, but below is a reminder.



Queries

3 queries

A new group

Another group

Other Queries (3)

Categories
10 rows loaded.
Last refreshed 7/
Loaded to works

Refresh

Move to group

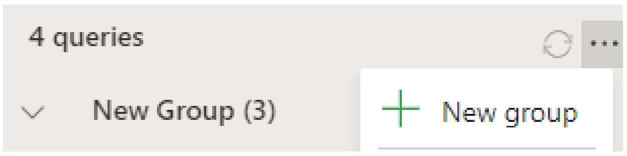
Products
7 rows loaded.
Last refreshed 7/27/2022, 12:59:15 PM
Loaded to worksheet

Sales

To begin, open the Queries pane, select the Data tab and then choose Queries.

Create a new group

- Select the top-level button and then choose ‘New group’

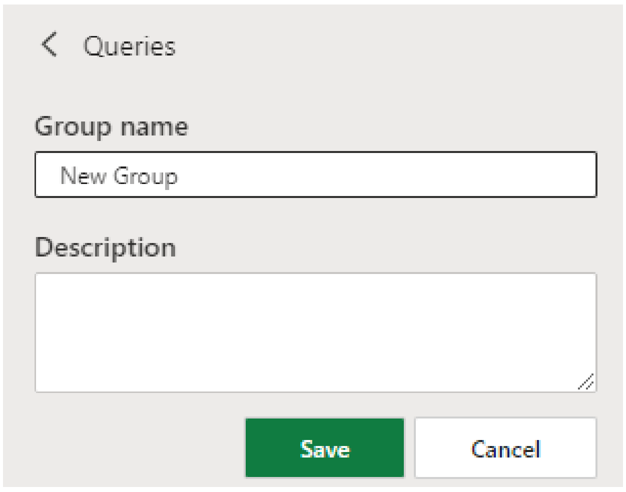


4 queries

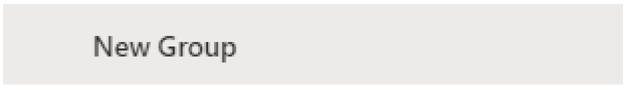
New Group (3)

New group

- Give the new group a name; you can also enter a description should you wish
- When you are done, choose Save



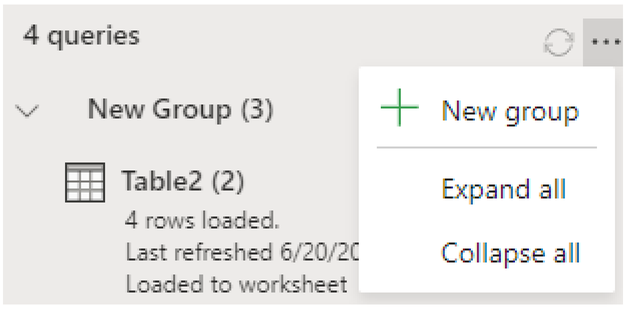
- You will see the new group was created:



You should note that until you move queries into groups, all queries will be listed under the 'Other queries' group.

Expand / collapse all groups

- Select the top-level button and then choose 'Expand all' / 'Collapse all' as required

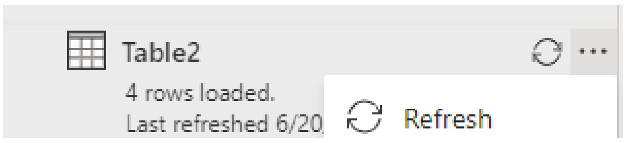


- You will see the result of the respective action in the Queries pane.

Again, you should note that you can Expand or Collapse any group by using the arrow next to the group name.

Refresh a group

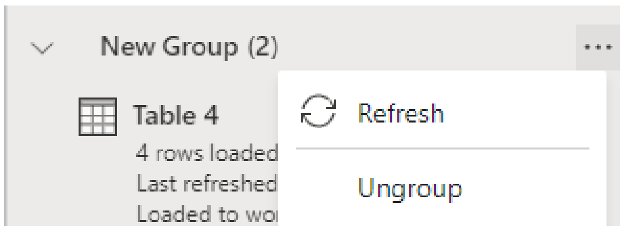
- If the group contains queries, you can refresh all of them at once, the same way you can refresh individual queries
- To do this, select the action menu button and then choose Refresh.



Obviously, refresh will work for supported data sources only.

Ungroup the queries within a group

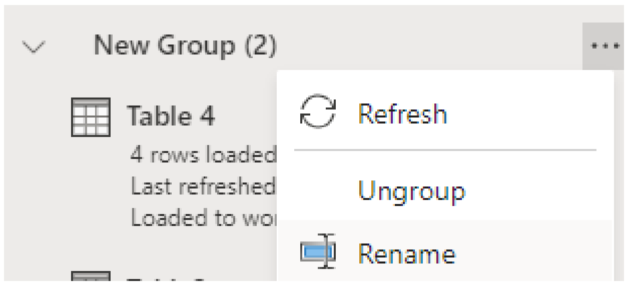
- If you want to reorganise your queries and take them out of the group, you can use this operation
- To do this, select the action menu button and then choose Ungroup



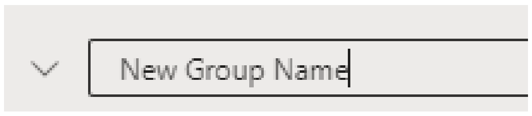
You should note that this action will only delete the group, not the queries in this group.

To rename a group

- To do this, select the action menu button and then choose Rename

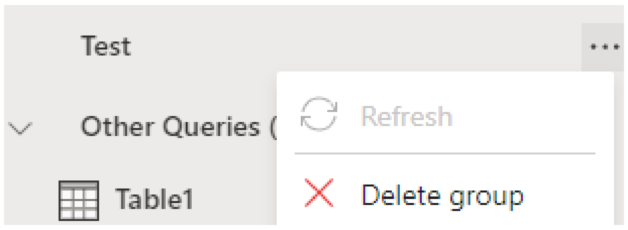


- Enter the new name, then press anywhere to save changes.



To delete a group

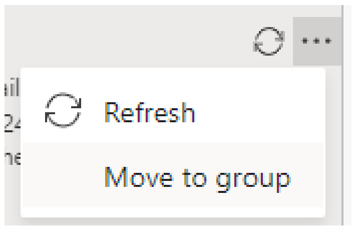
- To do this, select the action menu button and then choose 'Delete group'.



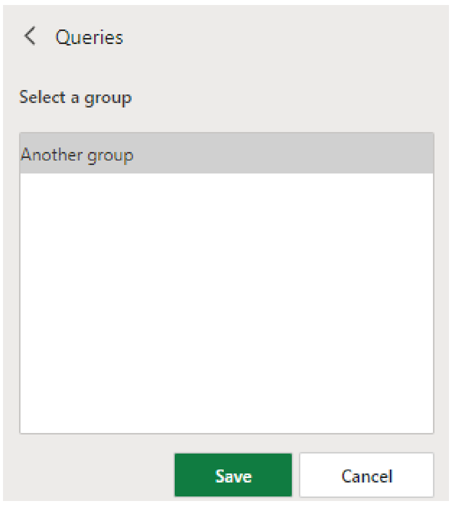
At this stage, delete is supported for empty groups only. Microsoft has stated it is working on adding support for deleting groups that include queries.

Move a query or group into another group

- To do this, select the action menu button and then choose 'Move to group…'

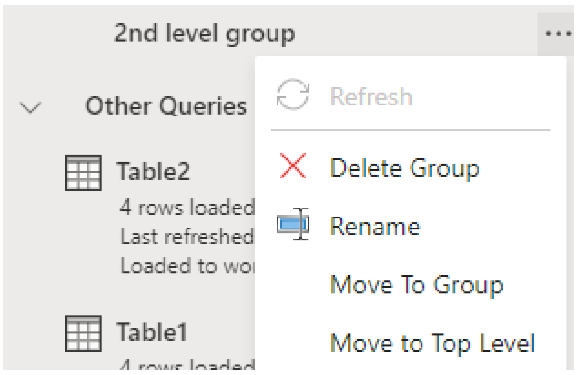


- Select the item you want to move it to and then select Save.



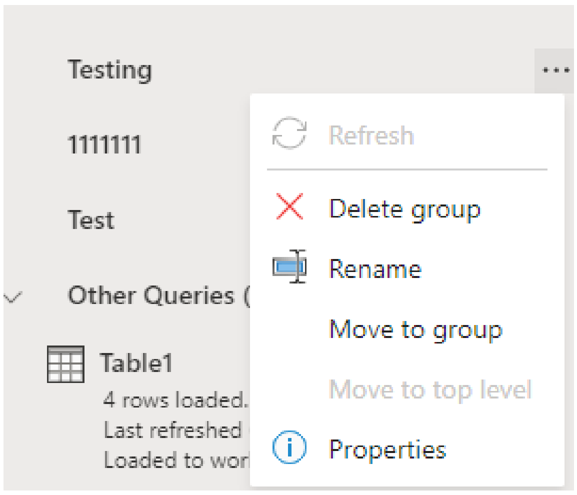
Move a group to the top level

- If a group is nested within one or more groups, you can quickly move it to the top level
- To do this, select the action menu button and then choose 'Move to top level'.

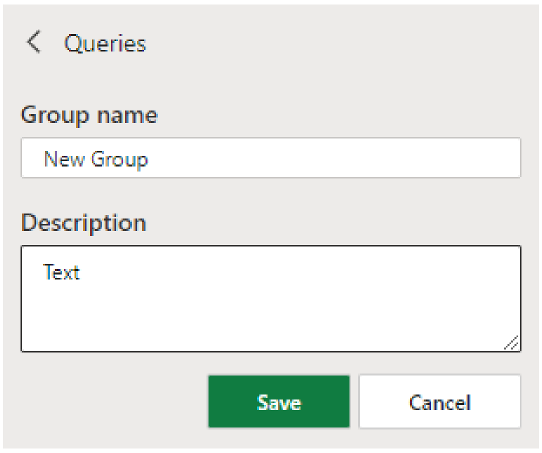


Edit a group's properties

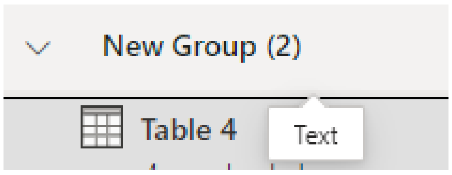
- To do this, select the action menu button and then choose Properties



- You can use this to rename the group or change its description
- When you are done, select Save



- You will be able to see the group description by hovering over it.



Apparently, Microsoft's future plans include Query operations, amongst other ideas. The end goal is to release the full Power Query Editor experience to Excel for the Web *eventually*.

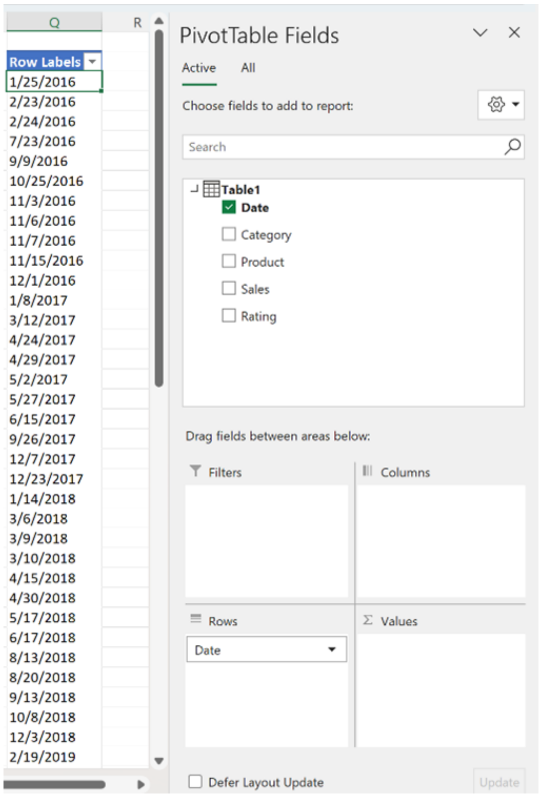
Improvements to the connected Power BI experience

In both Excel for the web and Excel for Windows, you can now create a PivotTable that connects directly to a Power BI dataset. This allows you to analyse data between platforms seamlessly. With this set of updates, Microsoft has improved the experience of analysing data in PivotTables.

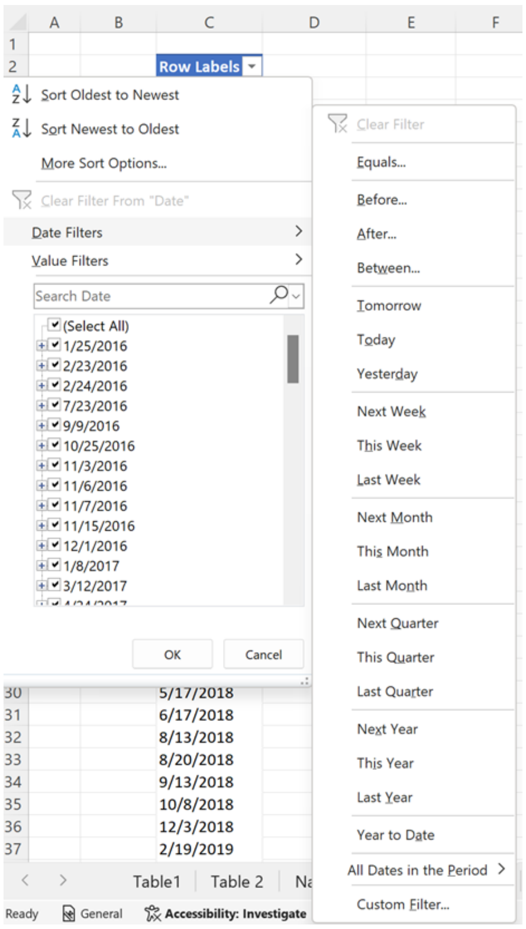
One upgrade that has been made is to enable proper date support for Power BI-connected PivotTables. Previously, dates were recognised as text strings. Now, they are date-and-time objects, which allows for accurate time-based filtering and sorting of data.

For date support:

- create a PivotTable connected to a Power BI dataset
- if the dataset has a date field that you'd like to use in your analysis, drag the field into an area of the PivotTable fields list (*e.g.* rows, columns, filters)



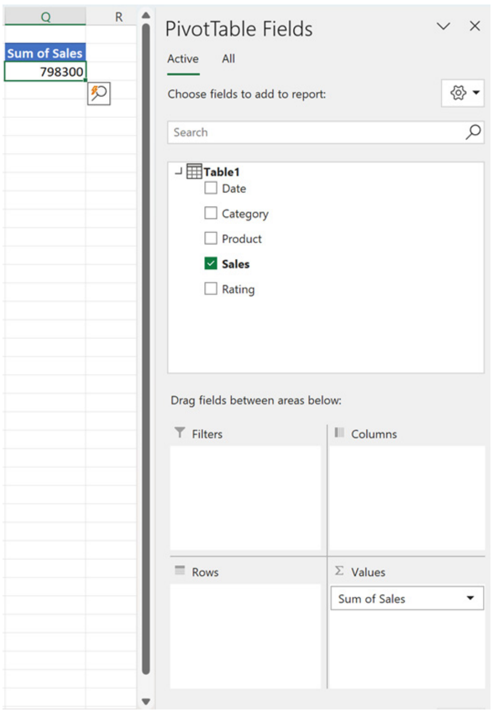
- Filter or sort the field in your PivotTable as desired.



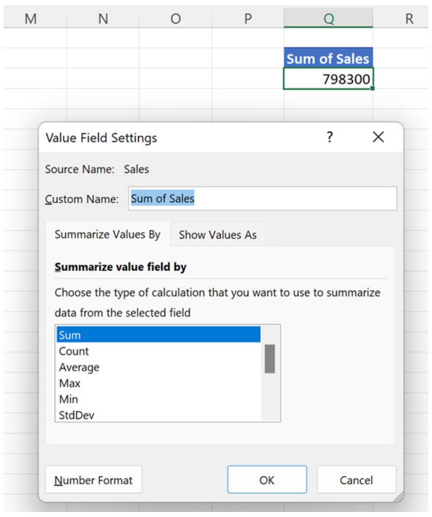
That's not all. Support for implicit measures has now been enabled in Power BI–connected PivotTables. These now support drag-and-drop aggregations (*e.g.* sum, average, distinct count) of fields without previously needed pre-defined measures (*i.e.* explicit measures) in the underlying Power BI datasets.

For drag and drop aggregation support:

- create a PivotTable connected to a Power BI dataset
- drag any field into the 'Values' area of the PivotTable fields list



- create aggregations on the field as desired.


Sum of Sales
798300

Value Field Settings

Source Name: Sales

Custom Name: Sum of Sales

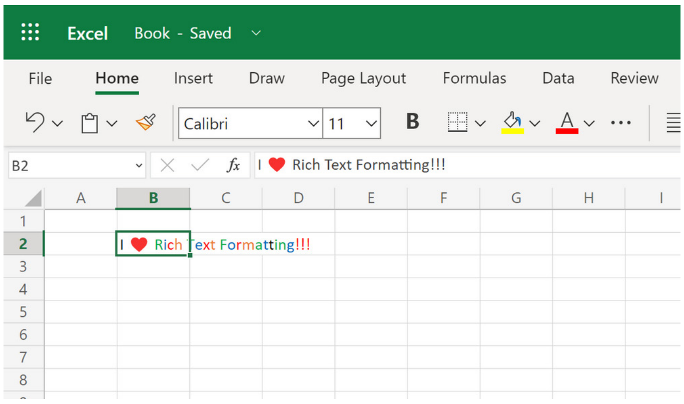
Summarize value field by

Choose the type of calculation that you want to use to summarize data from the selected field

The above will be rolled out to all users of Excel for the web and to Insiders running Beta Channel Version 2208 (Build 15601.20028) or later for Windows in due course.

Add and edit rich text formatting

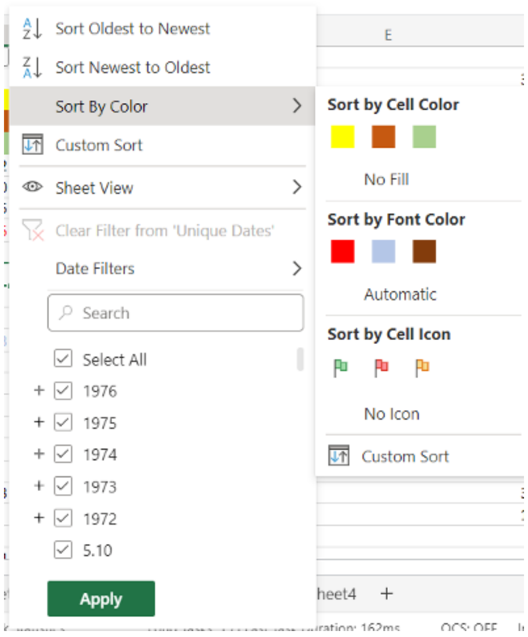
This one I use all the time! Also for Excel for the web, rich text formatting allows you to add formatting to only part of the text within a cell. You can use the Ribbon or shortcuts to add the formatting, *e.g.*



B2 | I ❤ Rich Text Formatting!!!

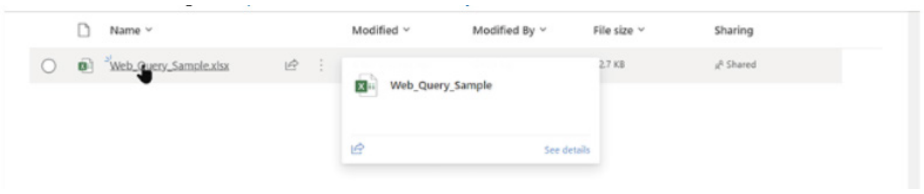
Sort by colour or icon from AutoFilter menu

Sorting is now easier and more convenient with new sort-by-colour or icon options in Excel for the web:



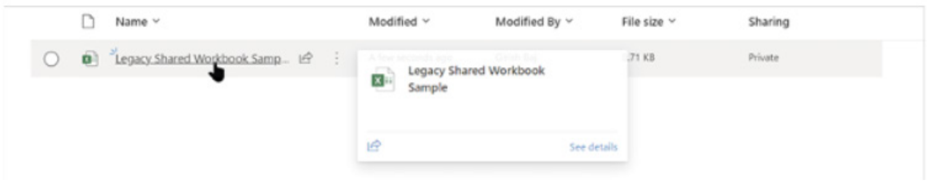
Edit files with legacy data connections

You can now edit files that contain legacy data connections like Text Queries, Web Queries, Query Table or SharePoint Lists in Excel for the web. This enables interaction with the workbook and access to the previously stored data from these connections, but without interaction, modification or refreshing of the connections themselves. Microsoft recommends switching to import data via Power Query to connect and refresh data.



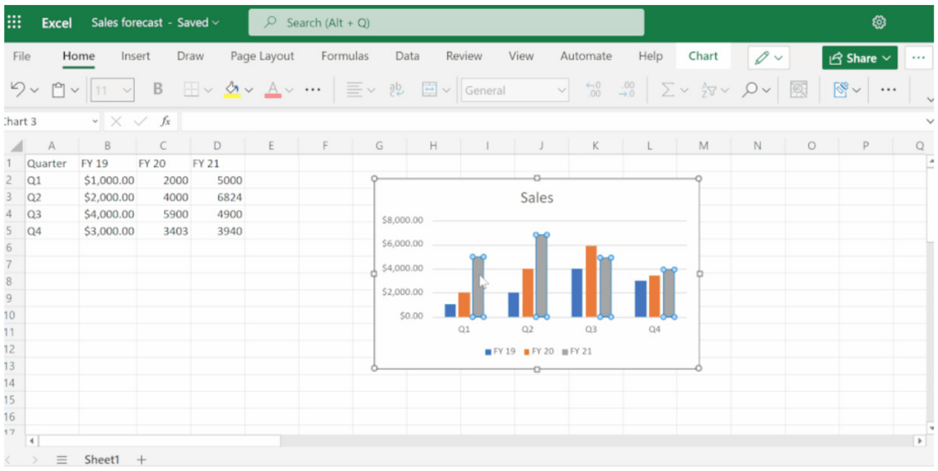
Edit files with legacy Shared Workbook feature

You can now edit files that use the legacy shared workbook feature in Excel for the web, with a provision for one-click turn-off of the legacy feature, thus allowing you to interact and collaborate with the workbook.



Delete chart elements

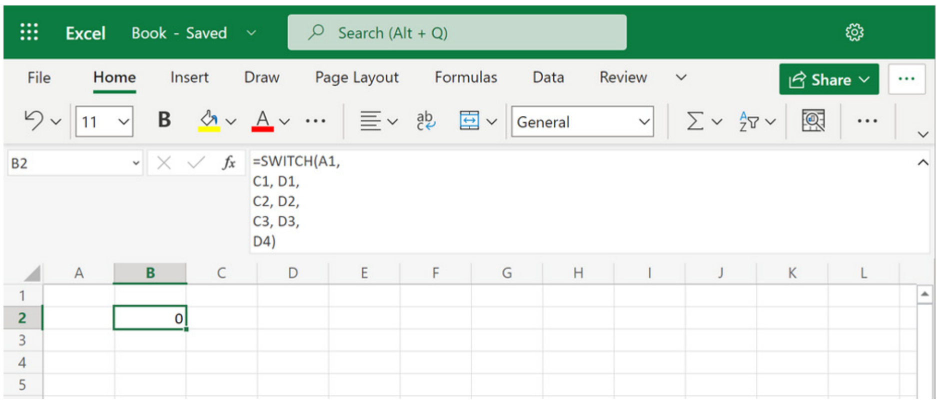
Deleting a data series from charts in Excel for the web is now easier by selecting a series and pressing the delete / backspace key to remove it.



Quarter	FY 19	FY 20	FY 21
Q1	$1,000.00	2000	5000
Q2	$2,000.00	4000	6824
Q3	$4,000.00	5900	4900
Q4	$3,000.00	3403	3940

Multiline Formula bar

And there is more for Excel for the web! Users may now expand and collapse the Formula bar by using the chevron or manually resizing it. This capability Improves the readability of long formulae or text, *viz.*



```
=SWITCH(A1,
C1, D1,
C2, D2,
C3, D3,
D4)
```

IMAGE function (Insiders Beta)

Your images can now be part of a worksheet in pretty much any version of Excel (either for the web, or some Insiders Beta variant), instead of floating on top of the cells. You may move and resize cells, sort and filter, and work with images within an Excel table. This improvement unlocks and facilitates many new scenarios, such as tracking inventories, creating employee dashboards or building games and brackets – something I know you are all especially keen to do!

The **IMAGE** function inserts images into cells from a source location, along with alternative text. Its syntax is as follows:

IMAGE(source, [alt_text], [sizing], [height], [width])

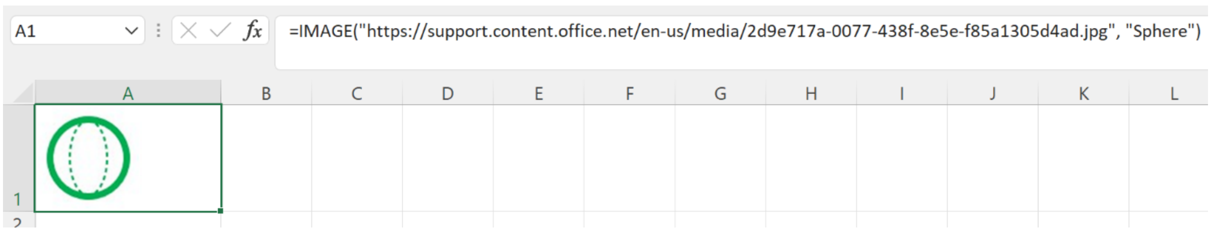
The arguments are as follows:

- **source** is required, and represents the URL path of the image file, using an *https* protocol (it should be noted that supported file formats include BMP, JPG, JPEG, GIF, TIFF, PNG, ICO and WEBP). Upon tinkering, cell references within the workbook appear to be recognised too
- **alt_text** is the first optional argument. This is the alternative text that describes the image (for accessibility purposes)
- **sizing** is also an optional parameter and specifies the image dimensions. There are several possible values:
  - **0:** fit the image in the cell and maintain its aspect ratio (default)
  - **1:** fill the cell with the image and ignore its aspect ratio
  - **2:** maintain the original image size, which may exceed the cell boundary
  - **3:** customise the image size by using the **height** and **width** arguments *(see below)*
- **height** and **width** are optional arguments. These define the height and width respectively of the image only when using **sizing** option 3 *(see above)*.

Examples

You may insert a sphere into a cell by typing

=IMAGE("https://support.content.office.net/en-us/media/2d9e717a-0077-438f-8e5e-f85a1305d4ad.jpg", "Sphere")

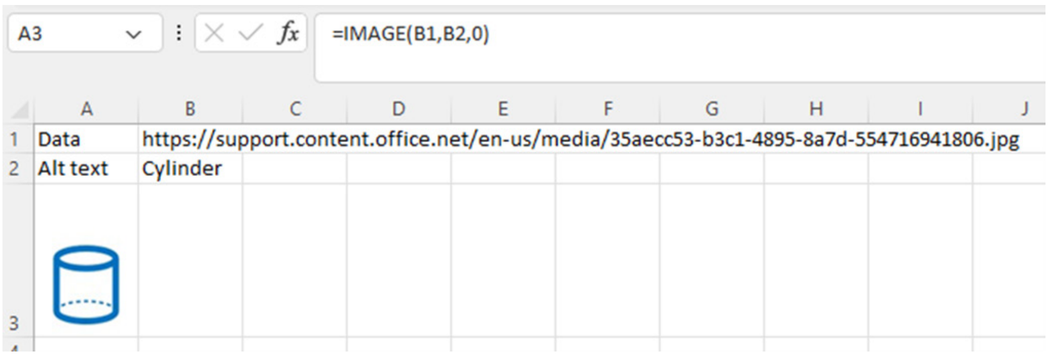


A1

=IMAGE("https://support.content.office.net/en-us/media/2d9e717a-0077-438f-8e5e-f85a1305d4ad.jpg", "Sphere")

Alternatively, you may insert a cylinder into a cell as follows:

- Copy and pasting the following URL into cell **B1**: https://support.content.office.net/en-us/media/35aecc53-b3c1-4895-8a7d-554716941806.jpg
- Type "Cylinder" in cell **B2**
- Enter the formula **=IMAGE(B1,B2,0)** in cell **A3** and pressing the **ENTER** key.



A3

=IMAGE(B1,B2,0)

	A	B
1	Data	https://support.content.office.net/en-us/media/35aecc53-b3c1-4895-8a7d-554716941806.jpg
2	Alt text	Cylinder
3		

Known issues

These include:

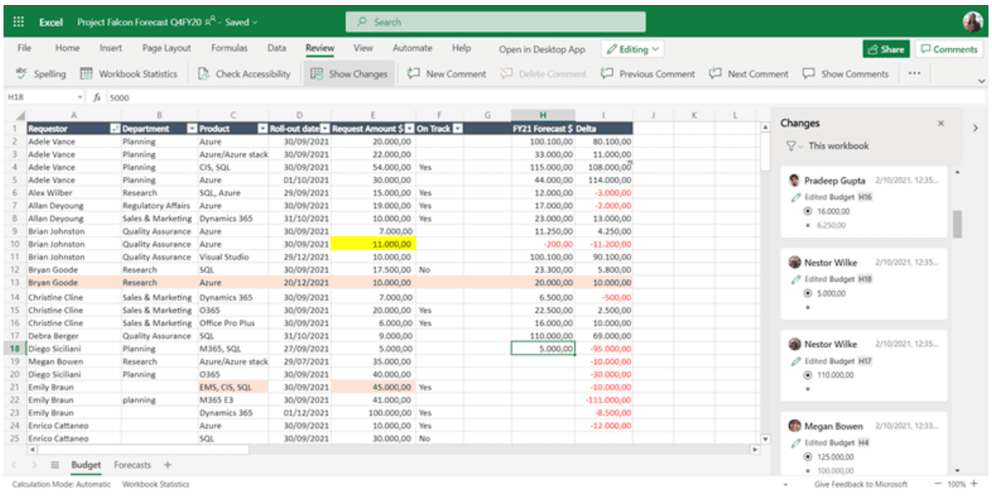
- if the URL to the image file you are using is pointing to a site that requires authentication, the image will not render
- zooming in and out with images in cells may distort the images
- moving between platforms (for example, Windows and Mac) may result in irregular image rendering.

The **IMAGE** function is available to Insiders running the following Beta Channel builds:

- **Windows:** Version 2209 (Build 15608.10000) or later
- **Mac:** Version 16.65 (Build 22080701) or later
- **iOS:** Version 2.65 (Build 22080701) or later
- **Android:** 16.0.15608.10000 or later.

Show Changes (Insiders CC Preview)

Show Changes in Excel lets you see exactly what edits were made to your workbooks, so you can confidently allow others to collaborate on your work. This is for all versions of Excel, subject to Preview. You can see details of who changed what, where and when, along with the previous value of the cell for quick reversion.



	Requestor	Department	Product	Roll-out date	Request Amount $	On Track		FY21 Forecast $	Delta
2	Adele Vance	Planning	Azure	30/09/2021	20.000,00			100.100,00	80.100,00
3	Adele Vance	Planning	Azure/Azure stack	30/09/2021	22.000,00			33.000,00	11.000,00
4	Adele Vance	Planning	CIS, SQL	30/09/2021	54.000,00	Yes		115.000,00	108.000,00
5	Adele Vance	Planning	Azure	01/10/2021	30.000,00			44.000,00	114.000,00
6	Alex Wilber	Research	SQL, Azure	29/09/2021	15.000,00	Yes		12.000,00	-3.000,00
7	Allan Deyoung	Regulatory Affairs	Azure	30/09/2021	19.000,00	Yes		17.000,00	-2.000,00
8	Allan Deyoung	Sales & Marketing	Dynamics 365	31/10/2021	10.000,00	Yes		23.000,00	13.000,00
9	Brian Johnston	Quality Assurance	Azure	30/09/2021	7.000,00			11.250,00	4.250,00
10	Brian Johnston	Quality Assurance	Azure	30/09/2021	11.000,00			-200,00	-11.200,00
11	Brian Johnston	Quality Assurance	Visual Studio	29/12/2021	10.000,00			100.100,00	90.100,00
12	Bryan Goode	Research	SQL	30/09/2021	17.500,00	No		23.300,00	5.800,00
13	Bryan Goode	Research	Azure	20/12/2021	10.000,00			20.000,00	10.000,00
14	Christine Cline	Sales & Marketing	Dynamics 365	30/09/2021	7.000,00			6.500,00	-500,00
15	Christine Cline	Sales & Marketing	O365	30/09/2021	20.000,00	Yes		22.500,00	2.500,00
16	Christine Cline	Sales & Marketing	Office Pro Plus	30/09/2021	6.000,00	Yes		16.000,00	10.000,00
17	Debra Berger	Quality Assurance	SQL	31/10/2021	9.000,00			110.000,00	69.000,00
18	Diego Siciliani	Planning	M365, SQL	27/09/2021	5.000,00			5.000,00	-95.000,00
19	Megan Bowen	Research	Azure/Azure stack	29/07/2021	35.000,00				-10.000,00
20	Diego Siciliani	Planning	O365	30/09/2021	40.000,00				-30.000,00
21	Emily Braun		EMS, CIS, SQL	30/09/2021	45.000,00	Yes			-10.000,00
22	Emily Braun	planning	M365 E3	30/09/2021	41.000,00				-111.000,00
23	Emily Braun		Dynamics 365	01/12/2021	100.000,00	Yes			-8.500,00
24	Enrico Cattaneo		Azure	30/09/2021	10.000,00	Yes			-12.000,00
25	Enrico Cattaneo		SQL	30/09/2021	30.000,00	No			

Changes

This workbook

Pradeep Gupta 2/10/2021, 12:35...
Edited Budget H16
16.000,00
6.250,00

Nestor Wilke 2/10/2021, 12:35...
Edited Budget H18
5.000,00

Nestor Wilke 2/10/2021, 12:35...
Edited Budget H17
110.000,00

Megan Bowen 2/10/2021, 12:33...
Edited Budget H4
125.000,00
100.000,00

You can narrow down the list of changes by selecting any sheet, range, or individual cell, to see all changes that were made, including bulk edits. You can see past changes for up to 60 days.

To view changes for the entire workbook:

- in the Review tab on the Ribbon, select 'Show Changes'
- changes are shown in the pane with the most recent changes on top, in the order the changes were made
- you can see who made edits, exactly where in the workbook, when, and what they changed
- you can also see 'Changes made at once' by clicking on 'See changes' in a bulk card.

You should note that 'Show Changes' tracks changes made from Windows Desktop, Excel on the Mac, iPad, iOS and Android phones. You can see past changes for up to 60 days.

To filter and see changes for a subset or range

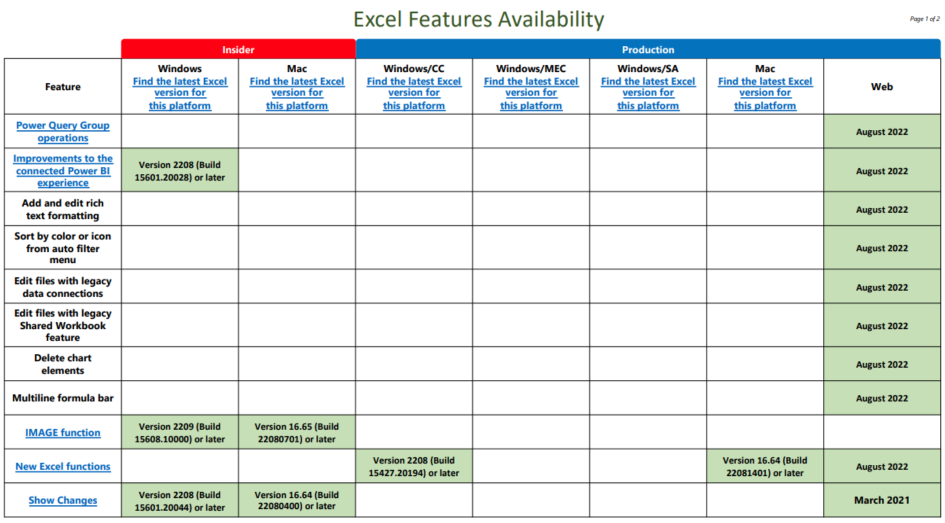
- select any sheet, range or single cell
- right-click to open the context menu and select 'Show Changes'.

If you or others edit the workbook while this pane is open, select the 'See New Changes' button to update the changes. The notification lets you stay in control when you want to refresh the list and not be distracted when reviewing changes.

To filter changes from the Changes pane:

- at the top of the pane, select the Filter icon
- select Range or Sheet to filter the changes shown, and then do the following:
    - o to show a range or cell's changes, select Range and enter the range or cell in the text box
    - o select the arrow icon next to the text box to confirm.

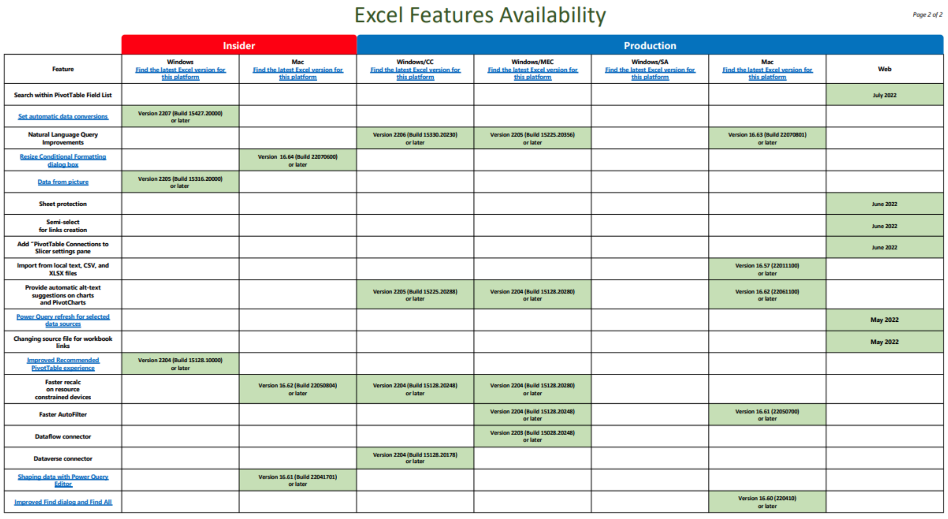
Microsoft has released an 'Excel Features Availability' flyer to detail all of these items:



Excel Features Availability

Page 1 of 2

	Insider		Production				
Feature	Windows Find the latest Excel version for this platform	Mac Find the latest Excel version for this platform	Windows/CC Find the latest Excel version for this platform	Windows/MEC Find the latest Excel version for this platform	Windows/SA Find the latest Excel version for this platform	Mac Find the latest Excel version for this platform	Web
Power Query Group operations							August 2022
Improvements to the connected Power BI experience	Version 2208 (Build 15601.20028) or later						August 2022
Add and edit rich text formatting							August 2022
Sort by color or icon from auto filter menu							August 2022
Edit files with legacy data connections							August 2022
Edit files with legacy Shared Workbook feature							August 2022
Delete chart elements							August 2022
Multiline formula bar							August 2022
IMAGE function	Version 2209 (Build 15608.10000) or later	Version 16.65 (Build 22080701) or later					
New Excel functions			Version 2208 (Build 15427.20194) or later			Version 16.64 (Build 22081401) or later	August 2022
Show Changes	Version 2208 (Build 15601.20044) or later	Version 16.64 (Build 22080400) or later					March 2021


Excel Features Availability

Page 2 of 2

	Insider		Production				
Feature	Windows Find the latest Excel version for this platform	Mac Find the latest Excel version for this platform	Windows/CC Find the latest Excel version for this platform	Windows/MEC Find the latest Excel version for this platform	Windows/SA Find the latest Excel version for this platform	Mac Find the latest Excel version for this platform	Web
Search within PivotTable Field List							July 2022
Set automatic data conversions	Version 2207 (Build 15427.20000) or later						
Natural Language Query Improvements			Version 2206 (Build 15330.20230) or later	Version 2205 (Build 15225.20356) or later		Version 16.63 (Build 22070801) or later	
Resize Conditional Formatting dialog box		Version 16.64 (Build 22070600) or later					
Data from picture	Version 2205 (Build 15316.20000) or later						
Sheet protection							June 2022
Semi-select for links creation							June 2022
Add "PivotTable Connections to Slicer settings pane							June 2022
Import from local text, CSV, and XLSX files						Version 16.57 (22011100) or later	
Provide automatic alt-text suggestions on charts and PivotCharts			Version 2205 (Build 15225.20288) or later	Version 2204 (Build 15128.20280) or later		Version 16.62 (22061100) or later	
Power Query refresh for selected data sources							May 2022
Changing source file for workbook links							May 2022
Improved Recommended PivotTable experience	Version 2204 (Build 15128.10000) or later						
Faster recalc on resource constrained devices		Version 16.62 (Build 22050804) or later	Version 2204 (Build 15128.20248) or later	Version 2204 (Build 15128.20280) or later			
Faster AutoFilter				Version 2204 (Build 15128.20248) or later		Version 16.61 (22050700) or later	
Dataflow connector				Version 2203 (Build 15028.20248) or later			
Dataverse connector			Version 2204 (Build 15128.20178) or later				
Shaping data with Power Query Editor		Version 16.61 (Build 22041701) or later					
Improved Find dialog and Find All						Version 16.60 (220410) or later	

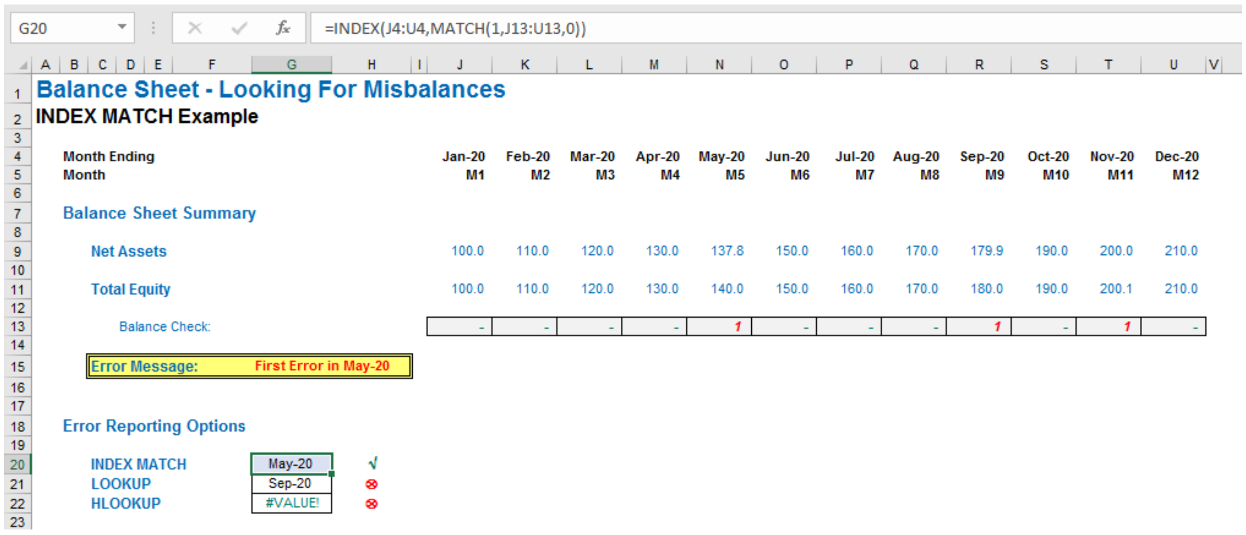
You can find the updated version of this grid with the interactive links at aka.ms/ExcelFeaturesFlyer.

The A to Z of Excel Functions: INDEX

INDEX with its best friend **MATCH** – as a combination – are two of the most useful functions at a modeller's disposal. They provide a versatile lookup in a way that **LOOKUP**, **HLOOKUP** and **VLOOKUP** simply cannot. The best way to illustrate this point is by means of an example.

Here is a common problem. Imagine you have built a financial model and your Balance Sheet – ahem! – contains misbalances. You need to fix it. Now I am sure you have never had this mistake yourself, but you have "close friends" that have encountered this feast of fun: solving Balance Sheet errors can take a *long* while. One of the first things modellers will do is locate the first period (in ascending order) that has such an error, as identifying the issue in this period may often solve the problem for all periods.

Consider the following example:


G20 =INDEX(J4:U4,MATCH(1,J13:U13,0))

Balance Sheet - Looking For Misbalances

INDEX MATCH Example

	Jan-20	Feb-20	Mar-20	Apr-20	May-20	Jun-20	Jul-20	Aug-20	Sep-20	Oct-20	Nov-20	Dec-20
Month Ending	Jan-20	Feb-20	Mar-20	Apr-20	May-20	Jun-20	Jul-20	Aug-20	Sep-20	Oct-20	Nov-20	Dec-20
Month	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
Balance Sheet Summary												
Net Assets	100.0	110.0	120.0	130.0	137.8	150.0	160.0	170.0	179.9	190.0	200.0	210.0
Total Equity	100.0	110.0	120.0	130.0	140.0	150.0	160.0	170.0	180.0	190.0	200.1	210.0
Balance Check:	-	-	-	-	1	-	-	-	1	-	1	-

Error Message: First Error in May-20

Error Reporting Options

INDEX MATCH	May-20	✓
LOOKUP	Sep-20	⊗
HLOOKUP	#VALUE!	⊗

This is a common modelling query. The usual suspects, **LOOKUP** and **HLOOKUP** / **VLOOKUP** do not work here:

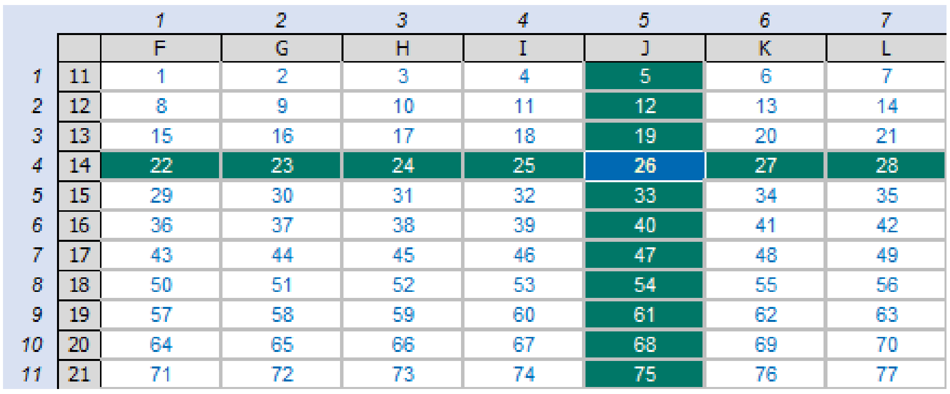
- **LOOKUP(lookup_value, lookup_vector, [result_vector])** gives the wrong date as the balance checks are not in strict ascending order (*i.e.* ascending alphanumerically with no duplicates); *whilst*
- **HLOOKUP(lookup_value, table_array, row_index_number, [range_ lookup])** gives *#VALUE!* since the first row must contain the data to be 'looked up', but the Balance Check is in row 13 in our example above, whereas the dates we need to return are in row 4 – hence we get a syntax error.

There is a solution, however: **INDEX MATCH**. They form a highly versatile tag team, but are worth introducing individually.

INDEX

Essentially, **INDEX(array, row_number, [column_number])** returns a value or the reference to a value from within a table or range (list). For example, **INDEX({7,8,9,10,11,12},3)** returns the third item in the list {7,8,9,10,11,12}, *i.e.* 9. This could have been a range: **INDEX(A1:A10,5)** gives the value in cell **A5**, *etc.*

INDEX can work in two dimensions as well (hence the **column_number** reference). Consider the following example:



		1	2	3	4	5	6	7
		F	G	H	I	J	K	L
1	11	1	2	3	4	5	6	7
2	12	8	9	10	11	12	13	14
3	13	15	16	17	18	19	20	21
4	14	22	23	24	25	26	27	28
5	15	29	30	31	32	33	34	35
6	16	36	37	38	39	40	41	42
7	17	43	44	45	46	47	48	49
8	18	50	51	52	53	54	55	56
9	19	57	58	59	60	61	62	63
10	20	64	65	66	67	68	69	70
11	21	71	72	73	74	75	76	77

INDEX(F11:L21,4,5) returns the value in the fourth row, fifth column of the table array **F11:L21** (clearly 26 in the above illustration).

MATCH

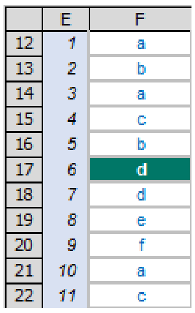
MATCH(lookup_value, lookup_vector, [match_type]) returns the relative position of an item in an array that (approximately) matches a specified value. It is not case sensitive.

The third argument, **match_type**, does not have to be entered, but for many situations, I strongly recommend that it is specified. It allows one of three values:

- **match_type 1 [default if omitted]:** finds the largest value less than or equal to the **lookup_value** – but the **lookup_vector** must be in strict ascending order, limiting flexibility
- **match_type 0:** probably the most useful setting, **MATCH** will find the position of the first value that matches **lookup_value** exactly. The **lookup_array** can have data in any order and even allows duplicates
- **match_type -1:** finds the smallest value greater than or equal to the **lookup_value** – but the **lookup_array** must be in strict descending order, again limiting flexibility.

When using **MATCH**, if there is no (approximate) match, *#N/A* is returned (this may also occur if data is not correctly sorted depending upon **match_type**).

MATCH is fairly straightforward to use:

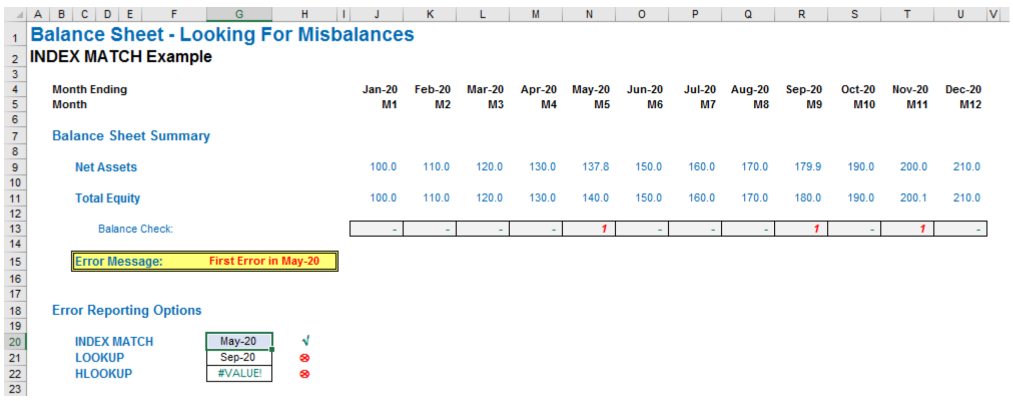


	E	F
12	1	a
13	2	b
14	3	a
15	4	c
16	5	b
17	6	d
18	7	d
19	8	e
20	9	f
21	10	a
22	11	c

In the figure above, **MATCH("d",F12:F22,0)** gives a value of 6, being the relative position of the first 'd' in the range. Note that having **match_type** 0 here is important. The data contains duplicates and is not sorted alphanumerically. Consequently, using **match_type** 1 and -1 would give the wrong answer: 7 and *#N/A* respectively.

INDEX MATCH

Whilst useful functions in their own right, combined they form a highly versatile partnership. Consider the original problem:



Balance Sheet - Looking For Misbalances
INDEX MATCH Example

	Jan-20	Feb-20	Mar-20	Apr-20	May-20	Jun-20	Jul-20	Aug-20	Sep-20	Oct-20	Nov-20	Dec-20
Month Ending	Jan-20	Feb-20	Mar-20	Apr-20	May-20	Jun-20	Jul-20	Aug-20	Sep-20	Oct-20	Nov-20	Dec-20
Month	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
Balance Sheet Summary												
Net Assets	100.0	110.0	120.0	130.0	137.8	150.0	160.0	170.0	179.9	190.0	200.0	210.0
Total Equity	100.0	110.0	120.0	130.0	140.0	150.0	160.0	170.0	180.0	190.0	200.1	210.0
Balance Check:	-	-	-	-	1	-	-	-	1	-	1	-

Error Message: First Error in May-20

Error Reporting Options

INDEX MATCH	May-20	√
LOOKUP	Sep-20	⊗
HLOOKUP	#VALUE!	⊗

MATCH(1,J13:O13,0) equals 5, *i.e.* the first period the balance sheet does not balance in is Period 5. But we can do better than that. **INDEX(J4:O4,5)** equals May-20, so combining the two functions:

INDEX(J4:O4,MATCH(1,J13:O13,0))

equals May-20 in one step. This process of stepping out two calculations and then inserting one into another is often referred to as "staggered development". No, this is not how you construct a financial model late in the evening after having the odd drink or two!

Do note how flexible this combination really is. We do not need to specify an order for the lookup range, we can have duplicates and the value to be returned does not have to be in a row / column below / to the right of the lookup range (indeed, it can be in another workbook never mind another worksheet!). With a little practice, the above technique can be extended to match items on a case sensitive basis, use multiple criteria and even 'grade'.

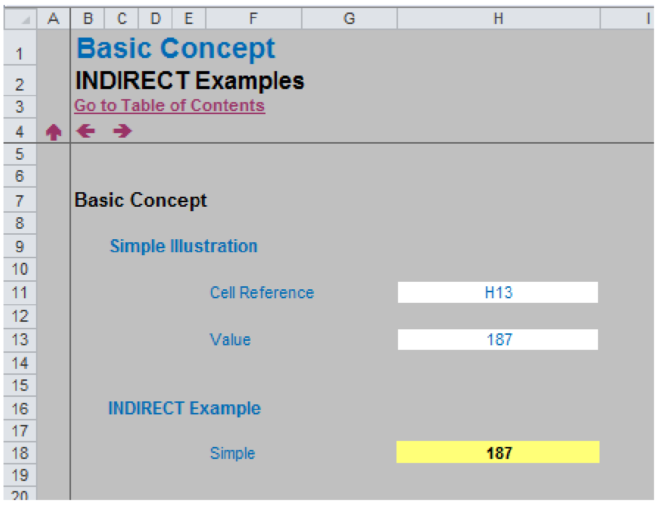
The A to Z of Excel Functions: INDIRECT

Excel's **INDIRECT** function allows the creation of a formula by referring to the contents of a cell, rather than the cell reference itself.

The **INDIRECT(ref_text,[a1])** function syntax has two arguments:

- **ref_text:** this is a required reference to a cell that contains an **A1**-style reference, an R1C1-style reference, a name defined as a reference or a reference to a cell as a text string. If **ref_text** is not a valid cell reference, **INDIRECT** returns the *#REF!* error value. If **ref_text** refers to another workbook (an external reference), the other workbook must be open. If the source workbook is not open, **INDIRECT** again returns the *#REF!* error value.
- **[a1]** This is optional (hence the square brackets) and represents a logical value that specifies what type of reference is contained in the cell ref_text. If **a1** is TRUE or omitted, **ref_text** is interpreted as an A1-style reference. If **a1** is FALSE, **ref_text** is interpreted as an R1C1-style reference. Most modellers seldom consider this alternative referencing approach, which is not without its merits *(see below)*.

Essentially, **INDIRECT** works as follows:


Basic Concept
INDIRECT Examples
Go to Table of Contents

Basic Concept

Simple Illustration

Cell Reference	H13
Value	187

INDIRECT Example

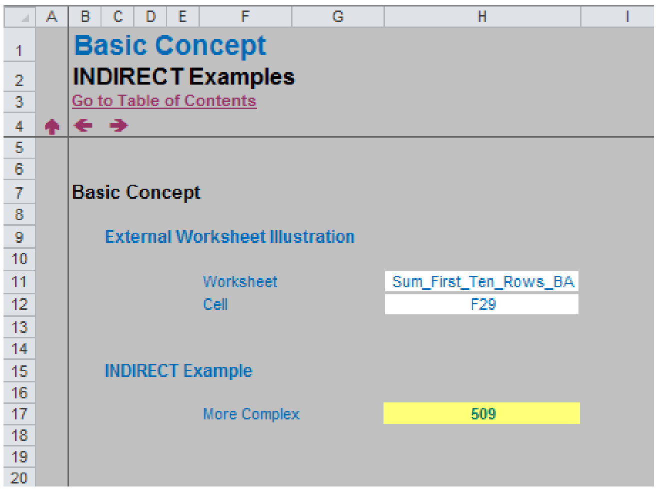
Simple	187

In the above example, the formula in cell **H18** (the yellow cell) is

=INDIRECT(H11).

With only one argument in this function, **INDIRECT** assumes the **A1**-cell notation (*e.g.* the cell in the third row fourth column is cell **D3**). Note that the value in cell **H11** is **H13**, this formula returns the value / contents of cell **H13**, *i.e.* 187.

This idea can be extended: the value indirectly referred to does not need to be in the same worksheet (or even workbook) as follows:


Basic Concept
INDIRECT Examples
Go to Table of Contents

Basic Concept

External Worksheet Illustration

Worksheet	Sum_First_Ten_Rows_BA
Cell	F29

INDIRECT Example

More Complex	509

The formula in the yellow-coloured cell (**H17**) uses concatenation:

=INDIRECT("'"&H11&"'!"&H12).

This formula is difficult to read. Let's make it clearer. **H11** in my example is **Sum_First_Ten_Rows_BA**, which is the name of another worksheet in the workbook and F29 is the cell to be linked to. In other words, this formula becomes

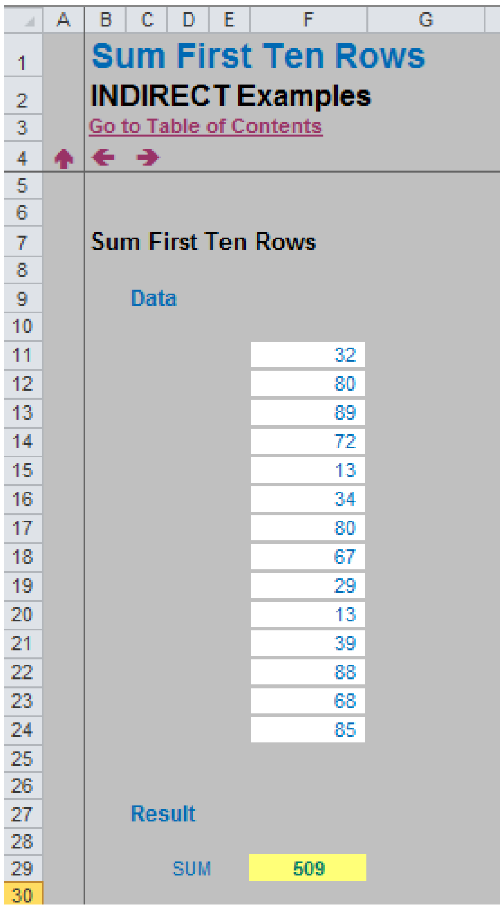
='Sum_First_Ten_Rows_BA'!F29,

which will make more sense to end users. The point is, however, the value in cell **H11** can be changed so that the formula suddenly links to a completely different worksheet.

Eagle-eyed readers may note that worksheet names without spaces do not need apostrophes. Whilst this is true, I include them here so that the formula will work in general.

Advantages of INDIRECT

INDIRECT has useful properties that may be exploited. For example, consider the following illustration:



Sum First Ten Rows

INDIRECT Examples

Go to Table of Contents

Sum First Ten Rows

Data

Row	F
11	32
12	80
13	89
14	72
15	13
16	34
17	80
18	67
19	29
20	13
21	39
22	88
23	68
24	85

Result

SUM 509

Imagine you wanted to sum the first ten values in this list. The obvious formula to use would be

=SUM(F11:F20).

However, what happens if someone inadvertently inserts or removes rows in this range? If a row were to be inserted the formula would automatically update to **=SUM(F11:F21)**, which most of the time would be what would be required. On occasion, though, it might be important that only the first ten values are still summed. **INDIRECT** can ensure this happen, *viz.*

=SUM(INDIRECT("F11:F20")).

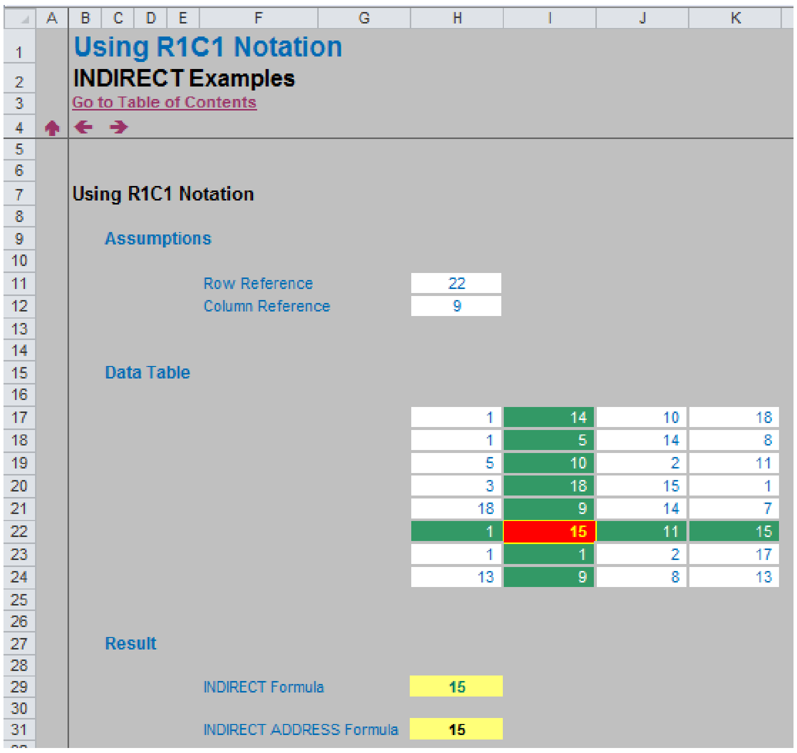
Note that when a reference is typed in like this it should be included in inverted commas as displayed. Using this formula will maintain the integrity of the referencing as required.

So far, all of my examples use the **A1**-style of cell referencing. However, using the R1C1 (row / column approach where the third row fourth column would be (3,4)) has benefits too. When this method is used, I often use **INDIRECT** in conjunction with the **ADDRESS** function.

The **ADDRESS(row_number, column_number)** function syntax has the following arguments:

- **row_number** is a numeric value that specifies the row number to use in the cell reference
- **column_number** is also required and is a numeric value that specifies the column number to use in the cell reference.

Therefore **=ADDRESS(3,4)** is **$D$3**.



Using R1C1 Notation

INDIRECT Examples

Go to Table of Contents

Using R1C1 Notation

Assumptions

Row Reference	22
Column Reference	9

Data Table

1	14	10	18
1	5	14	8
5	10	2	11
3	18	15	1
18	9	14	7
1	15	11	15
1	1	2	17
13	9	8	13

Result

INDIRECT Formula	15
INDIRECT ADDRESS Formula	15

Just using **INDIRECT** in the example above, cell **H29** uses concatenation once more:

=INDIRECT("R"&H11&"C"&H12,FALSE).

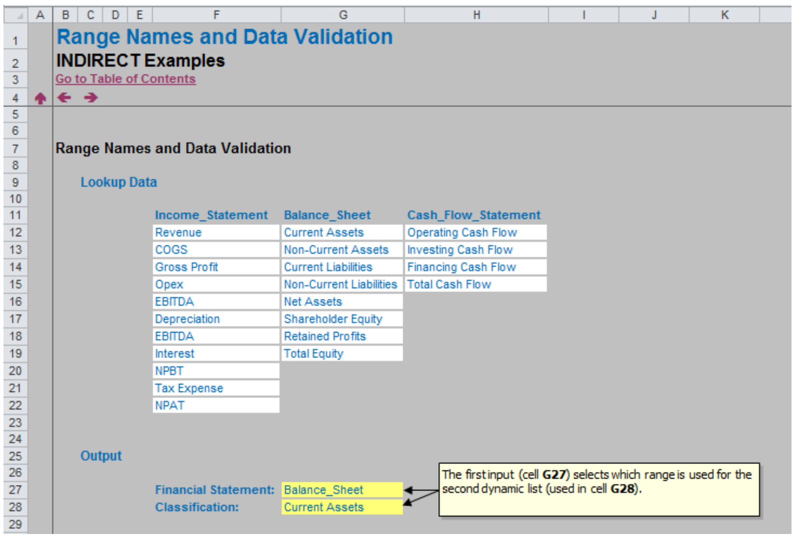
The second argument (**FALSE**) is necessary to recognise the R1C1 notation. Here, this formula reduces to **=R22C9**, which is the 22nd row, ninth column, i.e. delivers the value in cell **I22** (highlighted in red).

This formula can be difficult to understand for the uninitiated. Now, we're not saying the following is necessarily simpler, merely it is an alternative. The second formula in cell **H31** is

=INDIRECT(ADDRESS(H11,H12)).

This does not need the R1C1 notation as **ADDRESS(H11,H12)** equals cell **$I$22**.

Another advantage is in generating dynamic data validation lists.



Range Names and Data Validation

INDIRECT Examples

Go to Table of Contents

Range Names and Data Validation

Lookup Data

Income_Statement	Balance_Sheet	Cash_Flow_Statement
Revenue	Current Assets	Operating Cash Flow
COGS	Non-Current Assets	Investing Cash Flow
Gross Profit	Current Liabilities	Financing Cash Flow
Opex	Non-Current Liabilities	Total Cash Flow
EBITDA	Net Assets	
Depreciation	Shareholder Equity	
EBITDA	Retained Profits	
Interest	Total Equity	
NPBT		
Tax Expense		
NPAT		

Output

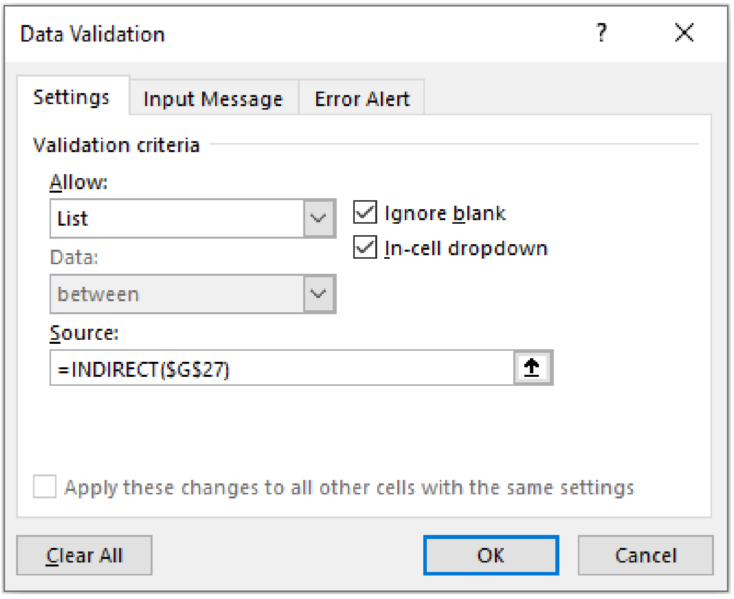
Financial Statement:	Balance_Sheet
Classification:	Current Assets

The first input (cell **G27**) selects which range is used for the second dynamic list (used in cell **G28**).

Here, we want to select a classification category in cell **G28**, based upon the financial statement selected in cell **G27** (*e.g.* Balance Sheet, Current Assets).

The trick here is <u>not</u> to include spaces in the names of the financial statements.

Then, first of all, in my illustration above, we've have named cells **F12:F22 Income_Statement**, cells **G12:G19 Balance Sheet** and cells **H12:H15 Cash_Flow_Statement**. Cells **F11:H11** have been used to construct a data validation list in cell **G27** and then the data validation list in cell **G28** has used the **INDIRECT** function in the 'Source:' field as follows:


Data Validation

Settings | Input Message | Error Alert

Validation criteria

Allow: List

☑ Ignore blank

☑ In-cell dropdown

Data: between

Source: =INDIRECT($G$27)

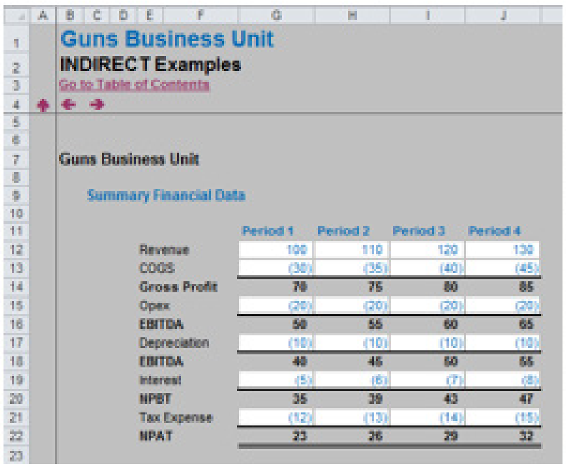
☐ Apply these changes to all other cells with the same settings

As a different financial statement is selected in cell **G27**, so the list will update in cell **G28** (but only once the data validation list is activated, which is an Excel limitation).

One last – and key – example: how often do you seek a summary sheet which selects data from one of several similarly constructed datasheets?

We have seen all sorts of weird and wonderful formulae to perform this common requirement, but **INDIRECT** is by far one of the simplest approaches available.

Consider the following file, which has several similar worksheets:


Guns Business Unit

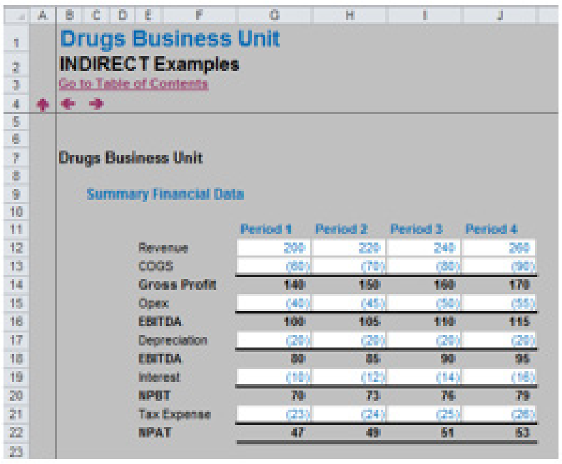
INDIRECT Examples

Go to Table of Contents

Guns Business Unit

Summary Financial Data

	Period 1	Period 2	Period 3	Period 4
Revenue	100	110	120	130
COGS	(30)	(35)	(40)	(45)
Gross Profit	70	75	80	85
Opex	(20)	(20)	(20)	(20)
EBITDA	50	55	60	65
Depreciation	(10)	(10)	(10)	(10)
EBITDA	40	45	50	55
Interest	(5)	(6)	(7)	(8)
NPBT	35	39	43	47
Tax Expense	(12)	(13)	(14)	(15)
NPAT	23	26	29	32


Drugs Business Unit

INDIRECT Examples

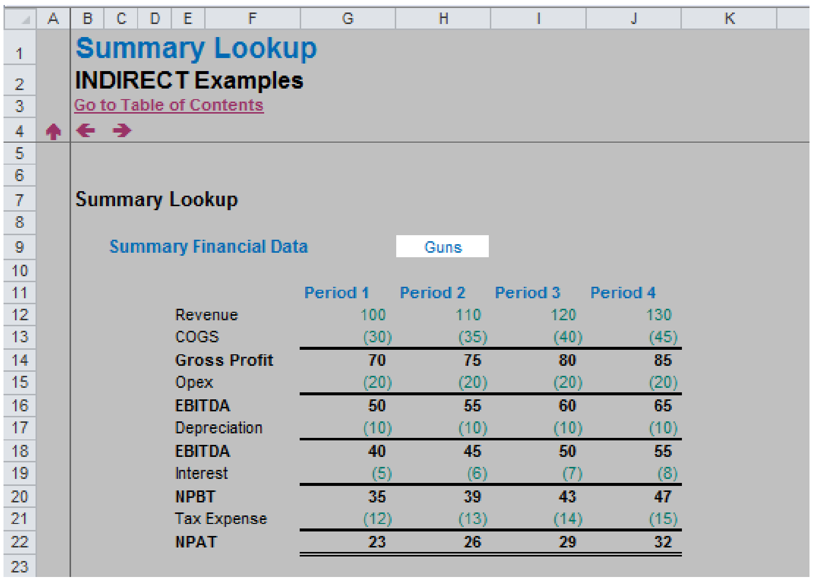
Go to Table of Contents

Drugs Business Unit

Summary Financial Data

	Period 1	Period 2	Period 3	Period 4
Revenue	200	220	240	260
COGS	(60)	(70)	(80)	(90)
Gross Profit	140	150	160	170
Opex	(40)	(45)	(50)	(55)
EBITDA	100	105	110	115
Depreciation	(20)	(20)	(20)	(20)
EBITDA	80	85	90	95
Interest	(10)	(12)	(14)	(16)
NPBT	70	73	76	79
Tax Expense	(23)	(24)	(25)	(26)
NPAT	47	49	51	53

We might require a summary sheet:


Summary Lookup

INDIRECT Examples

Go to Table of Contents

Summary Lookup

Summary Financial Data Guns

	Period 1	Period 2	Period 3	Period 4
Revenue	100	110	120	130
COGS	(30)	(35)	(40)	(45)
Gross Profit	70	75	80	85
Opex	(20)	(20)	(20)	(20)
EBITDA	50	55	60	65
Depreciation	(10)	(10)	(10)	(10)
EBITDA	40	45	50	55
Interest	(5)	(6)	(7)	(8)
NPBT	35	39	43	47
Tax Expense	(12)	(13)	(14)	(15)
NPAT	23	26	29	32

In this example, we have called our similar worksheets **Guns_BA** and **Drugs_BA**. The BA here refers to "Blank Assumptions" but it could mean "Basically Anything", *i.e.* the worksheet names contain more than just the business unit name.

With cell **H9** named **Selection**, the formula used in the calculations is simply

=INDIRECT("'"&Selection&"_BA'!RC",FALSE).

(the necessary apostrophes are coloured in red in this above formula).

However, as well as apostrophes and concatenation this formula uses a neat trick. The second argument must be **FALSE** (*i.e.* the formula assumes the R1C1 notation). When this is selected the **RC** in the above formula means use the row and column reference of the cell this formula is in. This avoids unnecessary hard code and generates a formula that changes reference depending upon the formula location in the worksheet. For example, in cell G12, the formula reduces to **='Guns_BA'!G12** and in cell **J21**, it reduces to **='Guns_BA'!J21**, *etc.*

Very useful!

Disadvantages of INDIRECT

Not all modellers embrace this useful function. There are three key issues with **INDIRECT**:

1. **INDIRECT** encourages the use of hard code in formulae. This should always be a last resort as this leads to a potential lack of transparency and flexibility in a model
2. **INDIRECT** is a difficult function to review / audit, as the cell(s) it refers to is not the ultimate location of the value used in the formula. Excel's in-built auditing tools are of limited use and the formulae can be highly confusing and 'clunky'
3. If **ref_text** refers to a cell range outside the row limit of 1,048,576 or the column limit of 16,384 (**XFD**), **INDIRECT** returns an *#REF!* error. However, this behaviour is different in earlier versions of Excel, which ignored the exceeded limit and returned an often meaningless value instead. This can lead to compatibility issues between versions of Excel (admittedly, these earlier versions are outdated, no longer supported, and therefore, should not be used).

Ultimately, the use of **INDIRECT** becomes a subject "horses for courses" issue. If modellers are knowledgeable and wary of its limitations, **INDIRECT** can simplify and resolve many common modelling problems. Just be careful out there.

The A to Z of Excel Functions: INFO

The **INFO** function returns information about the current operating environment. It employs the following syntax to operate:

INFO(type_text).

The **INFO** function has the following argument:

- **type_text:** this is required and represents the specification of what type of information you want returned.

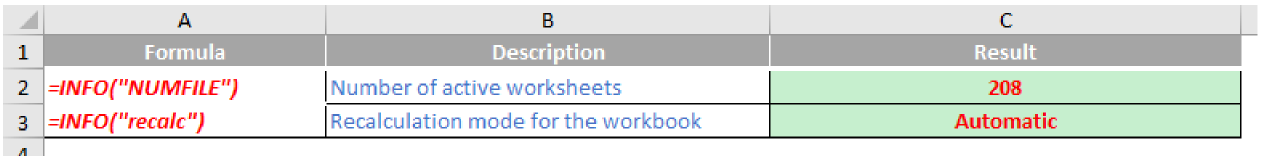
type_text	Returns
"directory"	Path of the current directory or folder
"numfile"	Number of active worksheets in the open workbooks
"origin"	Returns the absolute cell reference of the top and leftmost cell visible in the window, based on the current scrolling position, as text prepended with "$A:". This value is intended for Lotus 1-2-3 release 3.x compatibility. The actual value returned depends on the current reference style setting. Using **D9** as an example, the return value would be: • **A1** reference style **"$D$9"** • **R1C1** reference style **"R9C4"**
"osversion"	Current operating system version, as text

type_text	Returns
"recalc"	Current recalculation mode; returns "Automatic" or "Manual"
"release"	Version of Microsoft Excel, as text
"system"	Name of the operating environment: • Macintosh = "mac" • Windows = "pcdos"

It should be noted that:

- in previous versions of Microsoft Excel, the **"memavail"**, **"memused"** and **"totmem" type_text** values returned memory information. These **type_text** values are no longer supported and now return an *#N/A* error value instead.

Please see our example below:



	A	B	C
1	Formula	Description	Result
2	=INFO("NUMFILE")	Number of active worksheets	208
3	=INFO("recalc")	Recalculation mode for the workbook	Automatic

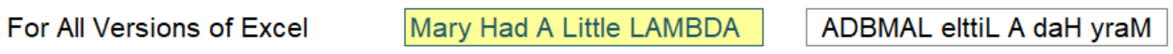
More Excel Functions next month.

Beat the Boredom Suggested Solution

If you are one of those people that reads the newsletter from back to front, then this might be the challenge for you! We shall *reverse* judgment.

The Challenge

So this month's challenge was as follows:



For All Versions of Excel | Mary Had A Little LAMBDA | ADBMAL elttiL A daH yraM

For **all** current versions of Excel (not just Office 365, Excel on the web and Insider / Beta variants), write a formula that will reverse the text in a cell (*i.e.* reverse the text string). You cannot use VBA, dynamic arrays, JavaScript, TypeScript or Office Scripts.

Suggested Solution

It's always dangerous to state that you can't do something in Excel. No disrespect is intended upon the authors of the Microsoft Research article, but we have often found it to be the case that saying such a thing may demonstrate a lack of Excel vocabulary and / or imagination. After all, necessity is often the mother of invention!

Having stated you might need an extended function knowledge and innovation, let's fall back on brute force and ignorance (I have the latter in abundance). We are going to need several functions.

The first one, **LEN(text)**, gives the length of **text** in terms of the number of characters contained within the **text** string (including blank spaces), *e.g.*

=LEN("Mary Had a Little LAMBDA")

has a value of 24, being the total number of characters in the text string, including the total number of spaces between words. You might not be sure where this is going at this point, but don't worry: all will become clear shortly. First though, let's introduce the next function, **INDIRECT**, in a little more detail.

The INDIRECT Function

Excel's **INDIRECT** function allows the creation of a formula by referring to the contents of the argument (cell), rather than the cell reference itself.

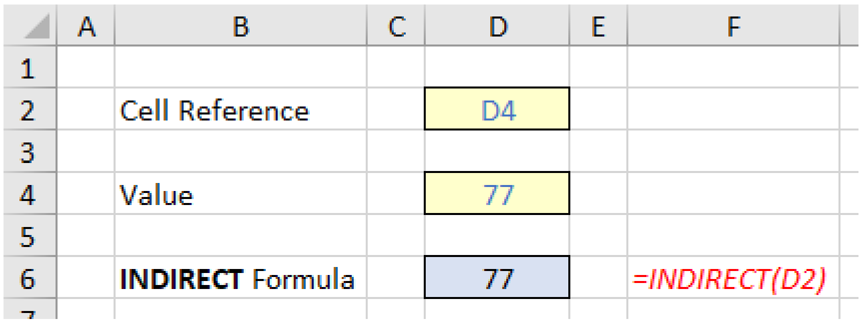
INDIRECT(ref_text, [a1])

The function syntax has two arguments:

1. **ref_text:** this is a required reference to a cell that contains an **A1**-style reference, an R1C1-style reference, a name defined as a reference or a reference to a cell as a text string. If ref_text is not a valid cell reference, **INDIRECT** returns the *#REF!* error value.

   If **ref_text** refers to another workbook (an external reference), the other workbook must be open. If the source workbook is not open, **INDIRECT** again returns the *#REF!* error value
2. **[a1]:** this is optional (hence the square brackets) and represents a logical value that specifies what type of reference is contained in the cell **ref_text**. If **a1** is TRUE or omitted, **ref_text** is interpreted as an A1-style reference. If **a1** is **FALSE**, **ref_text** is interpreted as an R1C1-style reference.

Essentially, **INDIRECT** works as follows:



	A	B	C	D	E	F
1						
2		Cell Reference		D4		
3						
4		Value		77		
5						
6		**INDIRECT** Formula		77		=INDIRECT(D2)

In the above example, the formula in cell **D6** (the blue cell) is

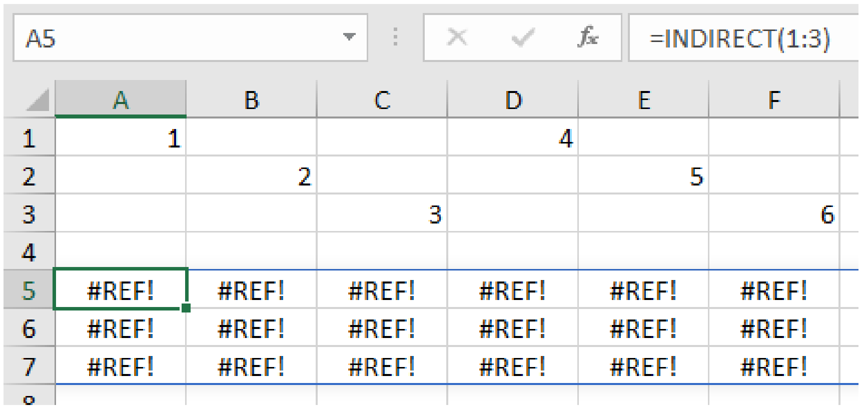
=INDIRECT(D2).

With only one argument in this function, **INDIRECT** assumes the **A1** cell notation (*e.g.* the cell in the third row fourth column is cell **D3**). Note that the value in cell **D2** is **D4**, so this formula returns the value / contents of cell **D4**, *i.e.* 77.

This idea can be extended: the value indirectly referred to does not need to be in the same worksheet (or even workbook), or may even be of another type altogether.

Returning to the Suggested Solution

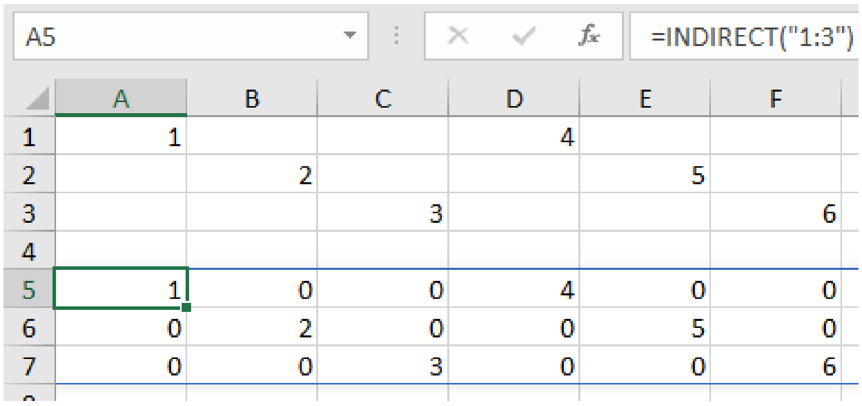
Let's consider **=INDIRECT(1)**. This would return *#REF!* because the intermediary reference 1 is meaningless in Excel. Yes, it constitutes a value, but **INDIRECT** is looking for a cell / range reference. You might think **=INDIRECT(1:3)** would reference rows 1 to 3 in Excel, but you would be wrong:



A5 =INDIRECT(1:3)

	A	B	C	D	E	F
1	1			4		
2		2			5	
3			3			6
4						
5	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!
6	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!
7	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!

This is incorrect syntax. It needs to be **=INDIRECT("1:3")** because **INDIRECT** is expecting to convert text to a reference it can resolve, *viz.*

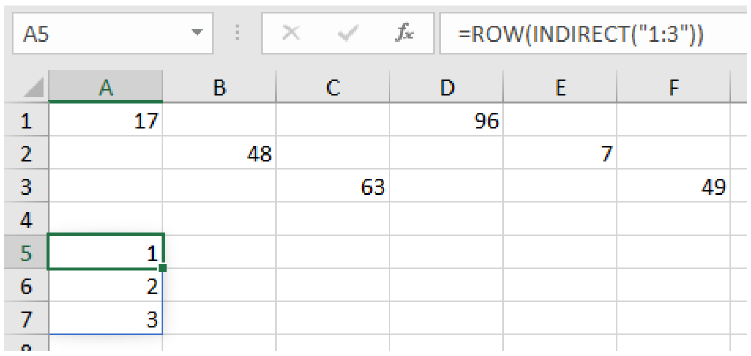


A5 =INDIRECT("1:3")

	A	B	C	D	E	F
1	1			4		
2		2			5	
3			3			6
4						
5	1	0	0	4	0	0
6	0	2	0	0	5	0
7	0	0	3	0	0	6

Hang on a minute: this is *spilling* – we are using dynamic arrays, and these are not allowed for the purposes of the challenge. Well, yes and no; you may use **=INDIRECT("1:3")** in all current versions of Excel, it's just this formula cannot be *rendered* in all versions. But that's not what we want in any case.

Now, let’s extend the idea. Let’s consider **=ROW(INDIRECT(“1:3”))**:

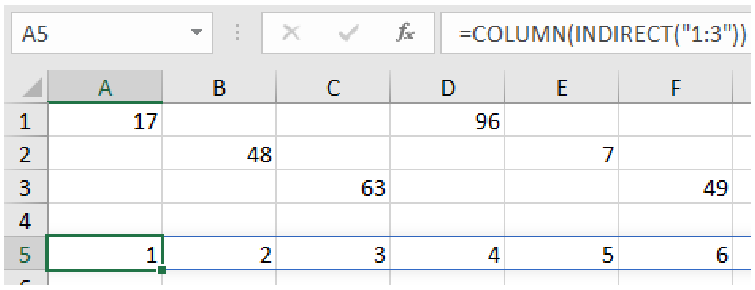


A5 =ROW(INDIRECT("1:3"))

	A	B	C	D	E	F
1	17			96		
2		48			7	
3			63			49
4						
5	1					
6	2					
7	3					

This example has deliberately changed the occasional values in rows 1 to 3 to demonstrate that this has *nothing* to do whatsoever with the contents of rows 1 to 3. It’s generating the numbers 1, 2 and 3 as **ROW(1)** equals 1, **ROW(2)** equals 2 and **ROW(3)** equals 3. The **ROW** function has essentially passed the range argument “1:3” to whatever function or operation it has been set up for.

Just for completion and in case you are wondering, we cannot use **COLUMN**:



A5 =COLUMN(INDIRECT("1:3"))

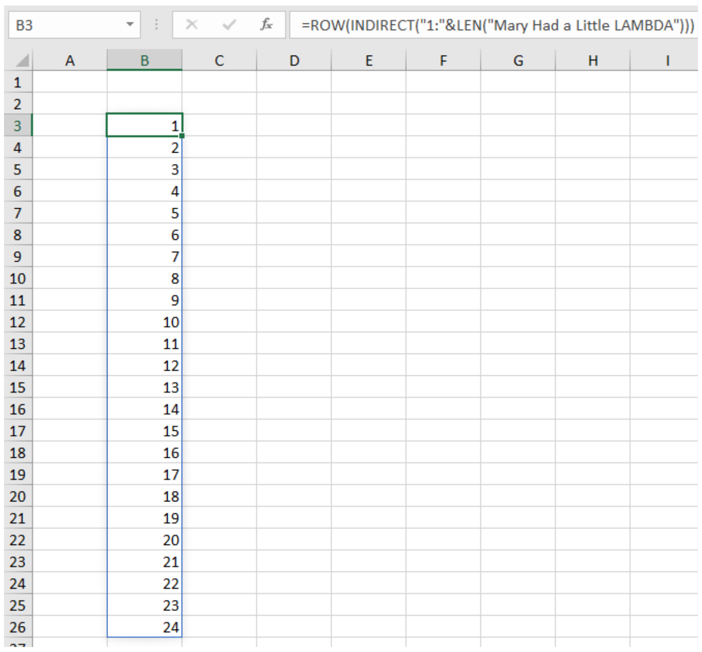
	A	B	C	D	E	F
1	17			96		
2		48			7	
3			63			49
4						
5	1	2	3	4	5	6

This would always extend to 16,384 columns (!).

If we combine this result with **LEN** from earlier,

=ROW(INDIRECT("1:"&LEN("Mary Had a Little LAMBDA")))

I would get

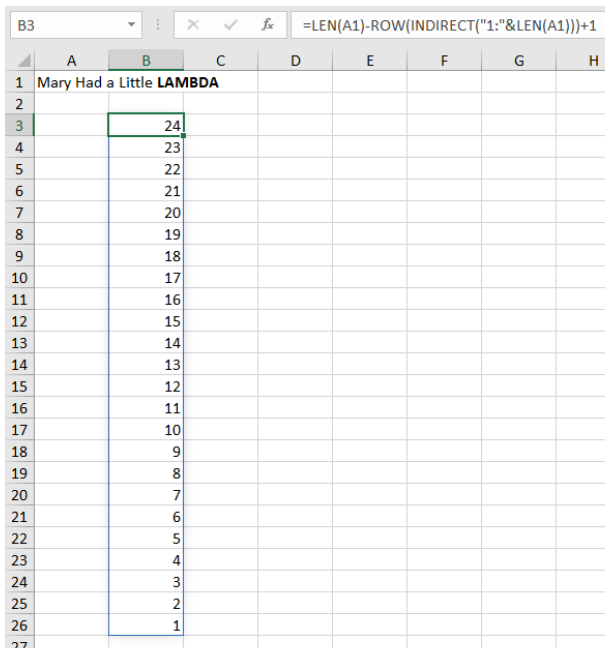


B3 =ROW(INDIRECT("1:"&LEN("Mary Had a Little LAMBDA")))

	A	B	C	D	E	F	G	H	I
1									
2									
3		1							
4		2							
5		3							
6		4							
7		5							
8		6							
9		7							
10		8							
11		9							
12		10							
13		11							
14		12							
15		13							
16		14							
17		15							
18		16							
19		17							
20		18							
21		19							
22		20							
23		21							
24		22							
25		23							
26		24							

(The **&** operator simply joins arguments together.)

It should be noted that this list of the numbers 1 to 24 might not display in all versions of Excel, but the calculation engine would recognise the list. We may even reverse the order:


B3 =LEN(A1)-ROW(INDIRECT("1:"&LEN(A1)))+1

	A	B	C	D	E	F	G	H
1	Mary Had a Little **LAMBDA**							
2								
3		24						
4		23						
5		22						
6		21						
7		20						
8		19						
9		18						
10		17						
11		16						
12		15						
13		14						
14		13						
15		12						
16		11						
17		10						
18		9						
19		8						
20		7						
21		6						
22		5						
23		4						
24		3						
25		2						
26		1						

The formula

=LEN(A1)-ROW(INDIRECT("1:"&LEN(A1)))+1

takes the list generated earlier (**ROW(INDIRECT("1:"&LEN(A1)))**) and subtracts it from the length of the text string in cell **A1** and adds one [1]. The first number would be 24 – 1 + 1 = 24, the second would be 24 – 2 + 1 = 23, the third would be 24 – 3 + 1 = 22, and so on.

Now we're ready to introduce my next function.

The MID Function

The **MID** function returns a specific number of characters from a text string, starting at the position you cite, based upon the number of characters you specify.

The **MID** function employs the following syntax to operate:

MID(text, start_number, number_of_characters)

The **MID** function has the following arguments:

- **text:** this is required and represents the text string that contains the characters you want to extract
- **start_number:** this is also required and specifies the position of the first character you want to extract from text. The first character in text has **start_number 1**, and so on
- **number_of_characters:** this argument is mandatory too and specifies the number of characters you want MID to return from the **text**.

It should be further noted that:

- if **start_number** is greater than the length of **text**, **MID** returns "" (empty text)
- if **start_number** is less than the length of **text**, but **start_number** plus number_of_characters exceeds the length of text, **MID** returns the characters up to the end of **text**
- if **start_number** is less than 1, **MID** returns the *#VALUE!* error value
- if **number_of_characters** is negative, **MID** returns the *#VALUE!* error value.

As an example,

=MID("Mary Had a Little LAMBDA",12,6)

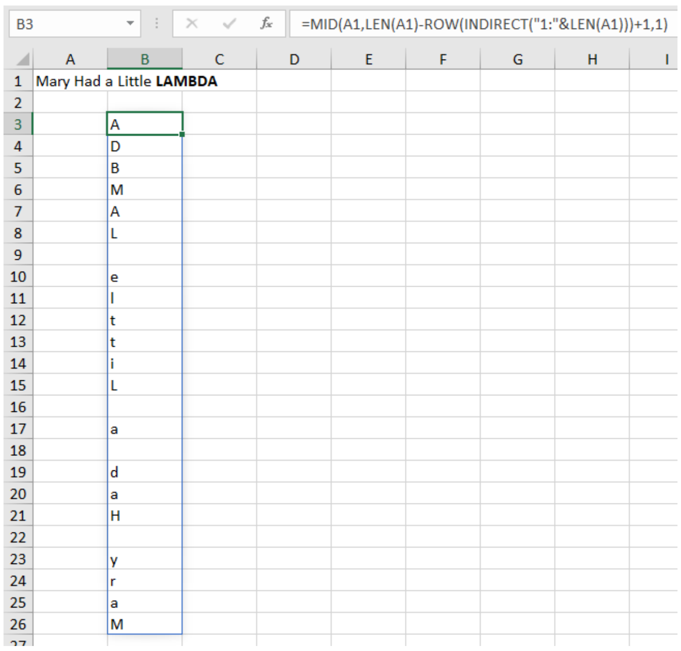
would return the text string "Little", being the sub-text string starting at the 12th character, consisting of six [6] characters.

Returning to the Suggested Solution

It starts to come together now. Consider

=MID(A1,LEN(A1)-ROW(INDIRECT("1:"&LEN(A1)))+1,1)

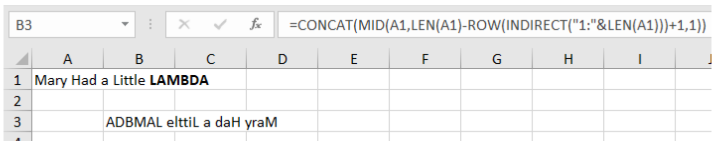
in our above example:



B3 =MID(A1,LEN(A1)-ROW(INDIRECT("1:"&LEN(A1)))+1,1)

	A	B
1	Mary Had a Little **LAMBDA**	
2		
3		A
4		D
5		B
6		M
7		A
8		L
9		
10		e
11		l
12		t
13		t
14		i
15		L
16		
17		a
18		
19		d
20		a
21		H
22		
23		y
24		r
25		a
26		M

Each cell takes the **n**th character in the reverse list from the text string in cell **A1**. We simply need to join it together.



B3 =CONCAT(MID(A1,LEN(A1)-ROW(INDIRECT("1:"&LEN(A1)))+1,1))

	A	B
1	Mary Had a Little **LAMBDA**	
2		
3		ADBMAL elttiL a daH yraM

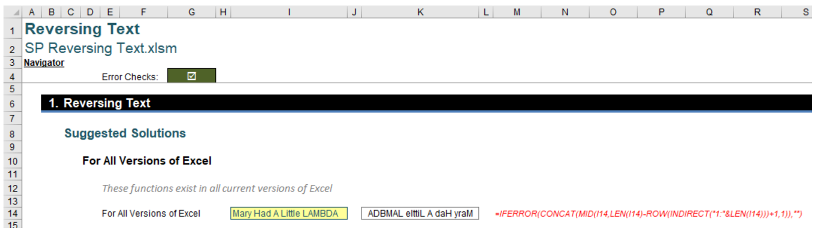
=CONCAT(MID(A1,LEN(A1)-ROW(INDIRECT("1:"&LEN(A1)))+1,1))

The **CONCAT** function replaces the old **CONCATENATE** function and combines the text from multiple ranges and / or text strings:

CONCAT(text1, [text2],…)

where **text1** is the text item to be joined and **text2** (onwards) are the additional items to be joined.

This is why you don't need to worry about spilling: we do not require our result to spill, and we are not using any dynamic array functions – as per the requirements.



Reversing Text
SP Reversing Text.xlsm
Navigator
Error Checks: ☑
1. Reversing Text
Suggested Solutions
For All Versions of Excel
These functions exist in all current versions of Excel
For All Versions of Excel | Mary Had A Little LAMBDA | ADBMAL elttiL A daH yraM | =IFERROR(CONCAT(MID(I14,LEN(I14)-ROW(INDIRECT("1:"&LEN(I14)))+1,1)),"")

Word to the Wise

In some versions of Excel, you may need to enter this formula using **CTRL + SHIFT + ENTER**, but even with older versions, this requirement should be unnecessary.

Until next time.

Location	Course	Date	Date	Duration	Duration
Online (Australia)	Excel Tips and Tricks	5 Oct 2022	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	1 Day
Online (Australia)	Financial Modelling	6 - 7 Oct 2022	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	2 Days
Online (Australia)	Power Pivot, Power Query and Power BI	9 - 11 Nov 2022	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	3 Days
Online (Australia)	Excel Tips and Tricks	16 Nov 2022	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	1 Day
Online (Australia)	Financial Modelling	17 - 18 Nov 2022	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	2 Days
Online (Australia)	Power Pivot, Power Query and Power BI	7 - 9 Dec 2022	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	3 Days
Online (Australia)	Excel Tips and Tricks	14 Dec 2022	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	1 Day
Online (Australia)	Financial Modelling	15 - 16 Dec 2022	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	2 Days

Key Strokes

Each newsletter, we'd like to introduce you to useful keystrokes you may or may not be aware of. This month, we look again at the **CTRL** and **SHIFT** keys, but this time combined with various special characters that Excel uses:

Keystroke	What it does
CTRL + SHIFT + -	Remove all borders
CTRL + SHIFT + ,	Fill down
CTRL + SHIFT + .	Fill right
CTRL + SHIFT + ;	Insert current time (in Edit mode)
CTRL + SHIFT + [	Select all precedent cells
CTRL + SHIFT + \	Select cells unequal to active cell
CTRL + SHIFT +]	Select all dependent cells
CTRL + SHIFT + `	General Number Format
CTRL + SHIFT + =	Insert cells

There are *c.*550 keyboard shortcuts in Excel. For a comprehensive list, please download our Excel file at www.sumproduct.com/thought/keyboard-shortcuts. Also, check out our new daily **Excel Tip of the Day** feature on the www.sumproduct.com homepage.

Our Services

We have undertaken a vast array of assignments over the years, including:

- **Business planning**
- **Building three-way integrated financial statement projections**
- **Independent expert reviews**
- **Key driver analysis**
- **Model reviews / audits for internal and external purposes**
- **M&A work**
- **Model scoping**
- **Power BI, Power Query & Power Pivot**
- **Project finance**
- **Real options analysis**
- **Refinancing / restructuring**
- **Strategic modelling**
- **Valuations**
- **Working capital management**

If you require modelling assistance of any kind, please do not hesitate to contact us at contact@sumproduct.com.

Link to Others

These newsletters are not intended to be closely guarded secrets. Please feel free to forward this newsletter to anyone you think might be interested in converting to "the SumProduct way".

If you have received a forwarded newsletter and would like to receive future editions automatically, please subscribe by completing our newsletter registration process found at the foot of any www.sumproduct.com web page.

Any Questions?

If you have any tips, comments or queries for future newsletters, we'd be delighted to hear from you. Please drop us a line at newsletter@sumproduct.com.

Training

SumProduct offers a wide range of training courses, aimed at finance professionals and budding Excel experts. Courses include Excel Tricks & Tips, Financial Modelling 101, Introduction to Forecasting and M&A Modelling.

Check out our more popular courses in our training brochure:

Drop us a line at training@sumproduct.com for a copy of the brochure or download it directly from www.sumproduct.com/training.

Sydney Address:	SumProduct Pty Ltd, Suite 803, Level 8, 276 Pitt Street, Sydney NSW 2000
New York Address:	SumProduct Pty Ltd, 48 Wall Street, New York, NY, USA 10005
London Address:	SumProduct Pty Ltd, Office 7, 3537 Ludgate Hill, London, EC4M 7JN, UK
Melbourne Address:	SumProduct Pty Ltd, Ground Floor, 470 St Kilda Road, Melbourne, VIC 3004
Registered Address:	SumProduct Pty Ltd, Level 14, 440 Collins Street, Melbourne, VIC 3000

contact@sumproduct.com
www.sumproduct.com
+61 3 9020 2071