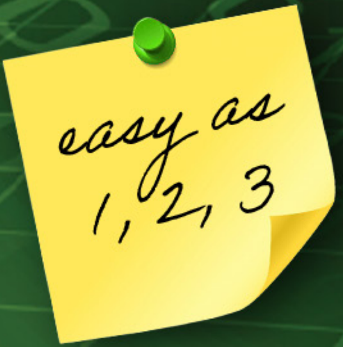
Sum Product

NEWSLETTER #123 - February 2023

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



It's easy as 123... as our newsletter continues into the new year unabated. And the updates seem to swing into full force with various updates for both Excel and Power BI – including a new way to connect Excel to Power BI!

Our normal features all remain primed and ready: you can check out our latest Beat the Boredom Challenge, plus there's also Charts & Dashboards, Visual Basics, Power Pivot Principles, Power Query Pointers and the A to Z of Excel functions series. We round out this month's newsletter with **CTRL + ALT + SHIFT** keyboard shortcuts and ruminate over what the heck 'Outdent' means...

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

Liam Bastick, Managing Director, SumProduct

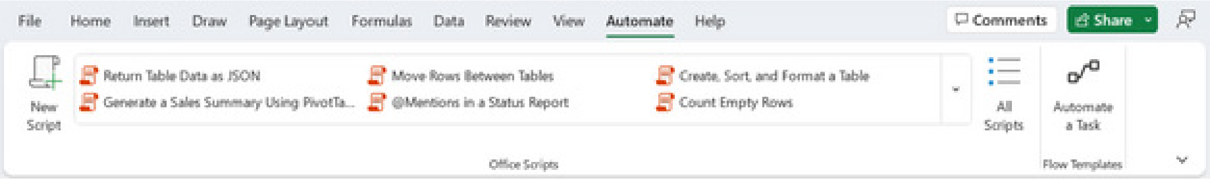
## Power Automate Now in Excel for Windows and Mac

The Automate tab is no longer just on the web browser. Since the beginning of the year, the Automate tab is now available for all eligible enterprise users in Excel for Windows and Mac. Previously, this tab was only available in Excel on the web.

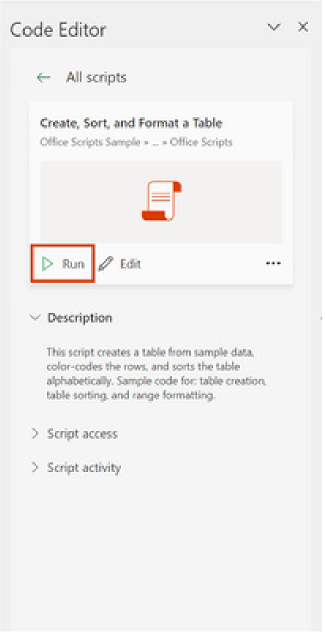
With this new tab, you may create and modify scripts that automate your repetitive tasks using Office Scripts. Furthermore, you can enhance your workbook by connecting popular applications like Microsoft Teams or SharePoint to build workflows with Power Automate. These may be combined, for example, to have Power Automate schedule your Office Script. Microsoft has confirmed that this tab represents the first stage of uniting automation solutions across platforms.

To view and run scripts:

- open any workbook in Excel for Windows or for Mac and select the Automate tab



- select a script from the gallery or from the 'All Scripts' task pane
- click the Run button on the script's detail page to run the script.

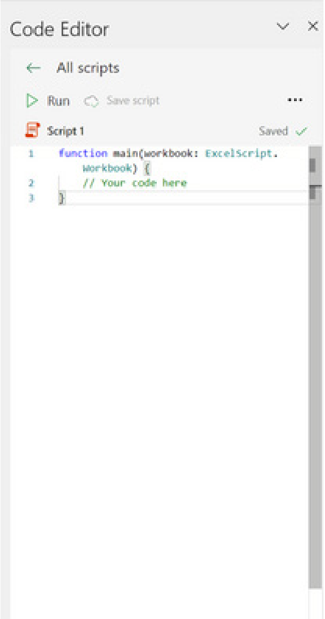

Code Editor
All scripts
Create, Sort, and Format a Table
Office Scripts Sample » ... » Office Scripts
Run
Edit
Description

This script creates a table from sample data, color-codes the rows, and sorts the table alphabetically. Sample code for: table creation, table sorting, and range formatting.

Script access
Script activity

Here's how to make a new script:

- open any workbook in Excel for Windows or for Mac and navigate to the Automate tab
- all the scripts in your workbook are available, as well as various samples from Microsoft
- make your own script by selecting the 'New Script' button

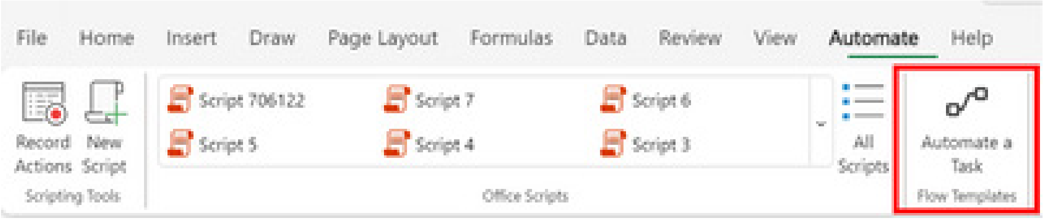


```
function main(workbook: ExcelScript.
    Workbook) {
    // Your code here
}
```

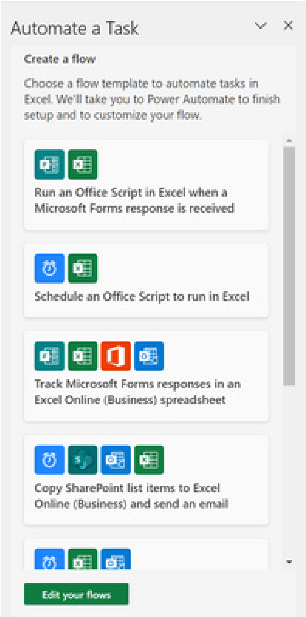
- to modify an existing script, select Edit on the script's details page, or select the pencil icon by hovering over any script in the 'All Scripts' task pane.

To connect your automations to other applications, proceed as follows:

- in Excel on the web, Excel for Windows or Excel for Mac, open an Excel workbook
- select **Automate -> Automate a Task**



- select the template you wish to use



Create a flow

Choose a flow template to automate tasks in Excel. We'll take you to Power Automate to finish setup and to customize your flow.

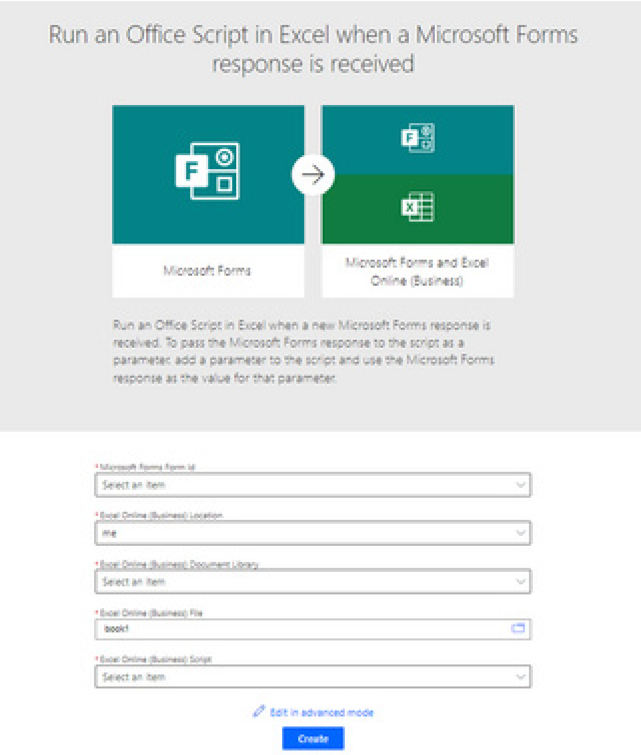
Run an Office Script in Excel when a Microsoft Forms response is received

Schedule an Office Script to run in Excel

Track Microsoft Forms responses in an Excel Online (Business) spreadsheet

Copy SharePoint list items to Excel Online (Business) and send an email

- sign in, provide the required information, and then select the Create button.



Run an Office Script in Excel when a Microsoft Forms response is received

Run an Office Script in Excel when a new Microsoft Forms response is received. To pass the Microsoft Forms response to the script as a parameter, add a parameter to the script and use the Microsoft Forms response as the value for that parameter.

* Microsoft Forms Form Id: Select an item

* Excel Online (Business) Location: me

* Excel Online (Business) Document Library: Select an item

* Excel Online (Business) File: book1

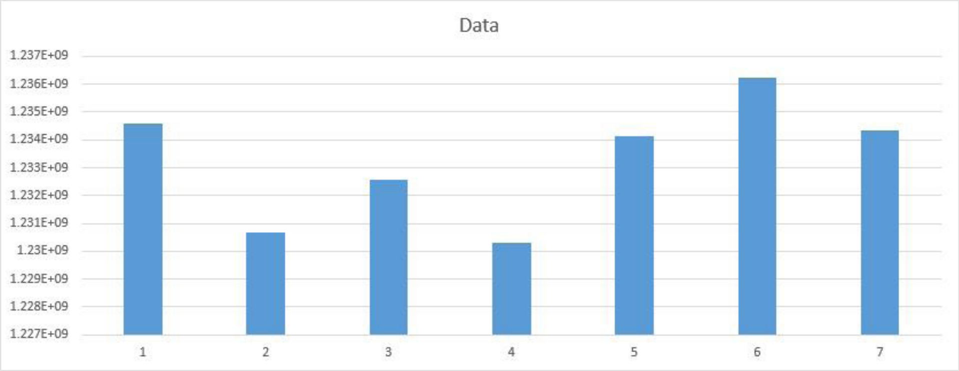
* Excel Online (Business) Script: Select an item

## 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…*

As we have shown numerous times, it's possible to introduce conditional number formatting into Excel cells using the basic number formatting options, *e.g.* to display numbers such as 10M, 650k and 120 using the same number format, reflecting the values 10,000,000, 650,000 and 120 respectively. We can have no more than three format options in any given cell though.

However, one client's request was to create a cell that would display four different number formats, reflecting billions (B), millions (M), thousands (K) and units. Since we can't display this in standard number formatting, we are forced to use conditional number formats, with conditions specifying when to apply target number formats.



The second part of the client's request was for the number format to continue onto a chart that the numbers were being reported. Therefore, our challenge for this month is as follows: can you create a chart to present a row of numbers, where the number format of the chart axis contains four or more different conditions?

Specific rules:

- It needs to change the number format into billions, millions, thousands or units depending on the numbers presented in the chart (at least four different conditions)
- No macros or user defined functions are allowed
- Conditional formatting is not allowed (not expecting that one??).

Sounds 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 the concept of dynamic charts.*

When we create lists or data in a workbook and make a report out of it, when we add or remove any data (*e.g.* add / remove rows or columns), it's possible that the whole report may become inaccurate. Excel has a solution for it, which is known as **Dynamic Tables**.

Why do we need Dynamic Tables or a Dynamic Range? The answer is because whenever a list or data range is updated or modified, it is not certain that the report will be changed as per the data change.

Basically, there are two main advantages of Dynamic Tables:

1. a dynamic range will update automatically, as per the data change(s)
2. PivotTables based on the dynamic table in Excel may be automatically updated when the pivot is refreshed.

So how do we create a Dynamic Tables in Excel? We may use either **Tables** or the **OFFSET** function.

### *Using Tables to Create Dynamic Tables*

Here, we'll select the Table range, then go to the Insert tab and choose Table, or **CTRL + T**:

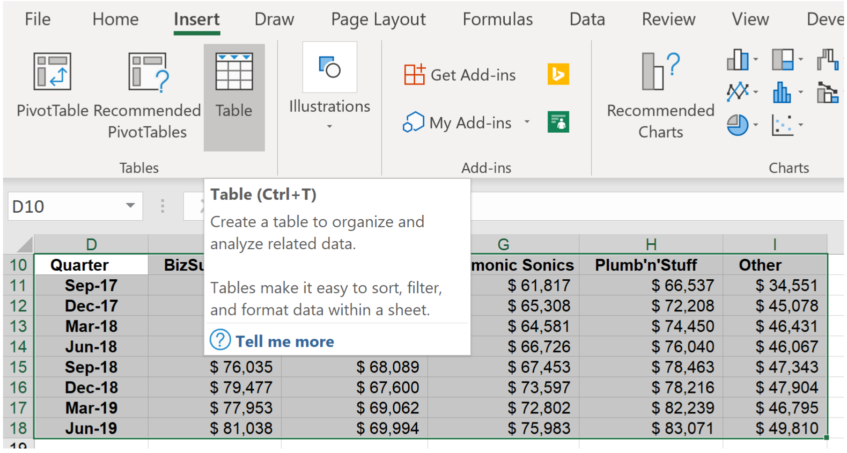


**Table (Ctrl+T)**

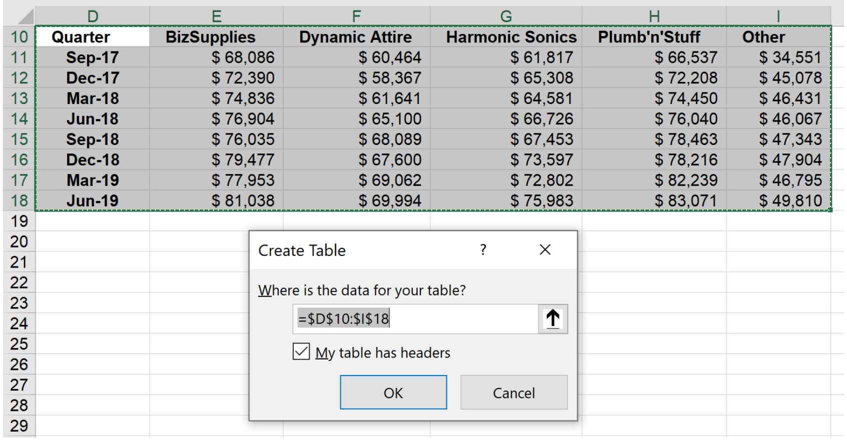
Create a table to organize and analyze related data.

Tables make it easy to sort, filter, and format data within a sheet.

Tell me more

| | D | E | F | G | H | I |
|---|---|---|---|---|---|---|
| 10 | Quarter | BizSu | | monic Sonics | Plumb'n'Stuff | Other |
| 11 | Sep-17 | | | $ 61,817 | $ 66,537 | $ 34,551 |
| 12 | Dec-17 | | | $ 65,308 | $ 72,208 | $ 45,078 |
| 13 | Mar-18 | | | $ 64,581 | $ 74,450 | $ 46,431 |
| 14 | Jun-18 | | | $ 66,726 | $ 76,040 | $ 46,067 |
| 15 | Sep-18 | $ 76,035 | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | $ 73,597 | $ 78,216 | $ 47,904 |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | $ 72,802 | $ 82,239 | $ 46,795 |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | $ 75,983 | $ 83,071 | $ 49,810 |

A 'Create Table' dialog will pop up. Since this data table has headers in the top row, you should check the box 'My table has headers'.



| | D | E | F | G | H | I |
|---|---|---|---|---|---|---|
| 10 | Quarter | BizSupplies | Dynamic Attire | Harmonic Sonics | Plumb'n'Stuff | Other |
| 11 | Sep-17 | $ 68,086 | $ 60,464 | $ 61,817 | $ 66,537 | $ 34,551 |
| 12 | Dec-17 | $ 72,390 | $ 58,367 | $ 65,308 | $ 72,208 | $ 45,078 |
| 13 | Mar-18 | $ 74,836 | $ 61,641 | $ 64,581 | $ 74,450 | $ 46,431 |
| 14 | Jun-18 | $ 76,904 | $ 65,100 | $ 66,726 | $ 76,040 | $ 46,067 |
| 15 | Sep-18 | $ 76,035 | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | $ 73,597 | $ 78,216 | $ 47,904 |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | $ 72,802 | $ 82,239 | $ 46,795 |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | $ 75,983 | $ 83,071 | $ 49,810 |

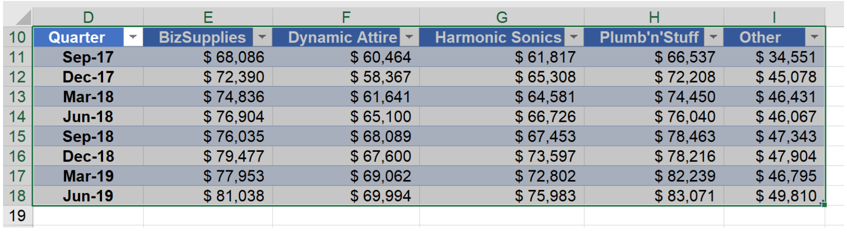
Create Table ? ×

Where is the data for your table?

=$D$10:$I$18

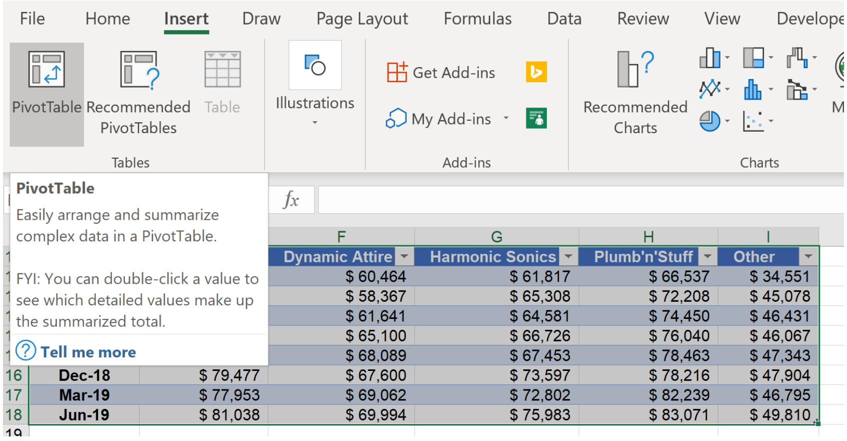
☑ My table has headers

The Table will now look like this:



| | D | E | F | G | H | I |
|---|---|---|---|---|---|---|
| 10 | Quarter | BizSupplies | Dynamic Attire | Harmonic Sonics | Plumb'n'Stuff | Other |
| 11 | Sep-17 | $ 68,086 | $ 60,464 | $ 61,817 | $ 66,537 | $ 34,551 |
| 12 | Dec-17 | $ 72,390 | $ 58,367 | $ 65,308 | $ 72,208 | $ 45,078 |
| 13 | Mar-18 | $ 74,836 | $ 61,641 | $ 64,581 | $ 74,450 | $ 46,431 |
| 14 | Jun-18 | $ 76,904 | $ 65,100 | $ 66,726 | $ 76,040 | $ 46,067 |
| 15 | Sep-18 | $ 76,035 | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | $ 73,597 | $ 78,216 | $ 47,904 |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | $ 72,802 | $ 82,239 | $ 46,795 |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | $ 75,983 | $ 83,071 | $ 49,810 |

Choosing the entire Table, we can then go to Insert tab and choose PivotTable.



PivotTable

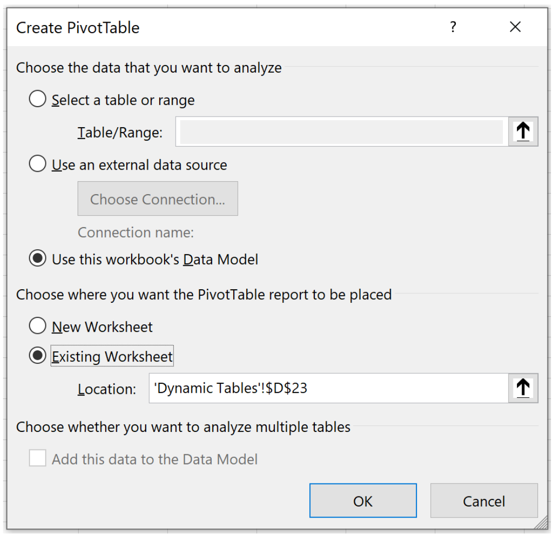
Easily arrange and summarize complex data in a PivotTable.

FYI: You can double-click a value to see which detailed values make up the summarized total.

Tell me more

| | | | F | G | H | I |
|---|---|---|---|---|---|---|
| | | | Dynamic Attire | Harmonic Sonics | Plumb'n'Stuff | Other |
| 1 | | | $ 60,464 | $ 61,817 | $ 66,537 | $ 34,551 |
| 1 | | | $ 58,367 | $ 65,308 | $ 72,208 | $ 45,078 |
| 1 | | | $ 61,641 | $ 64,581 | $ 74,450 | $ 46,431 |
| 1 | | | $ 65,100 | $ 66,726 | $ 76,040 | $ 46,067 |
| 1 | | | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | $ 73,597 | $ 78,216 | $ 47,904 |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | $ 72,802 | $ 82,239 | $ 46,795 |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | $ 75,983 | $ 83,071 | $ 49,810 |

The ‘Create PivotTable’ dialog will then appear. We will then load the PivotTable into the same worksheet, choose ‘Existing Worksheet’ and click on a cell location, here, we will use cell **D23**.



Create PivotTable

Choose the data that you want to analyze

○ Select a table or range

Table/Range:

○ 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

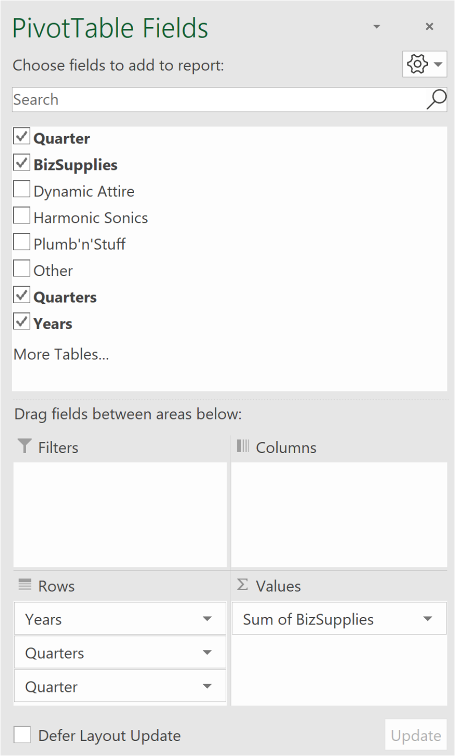
● Existing Worksheet

Location: 'Dynamic Tables'!$D$23

Choose whether you want to analyze multiple tables

□ Add this data to the Data Model

Choose **Quarter** as Rows and **BizSupplies** as Values:



PivotTable Fields

Choose fields to add to report:

☑ **Quarter**

☑ **BizSupplies**

□ Dynamic Attire

□ Harmonic Sonics

□ Plumb'n'Stuff

□ Other

☑ **Quarters**

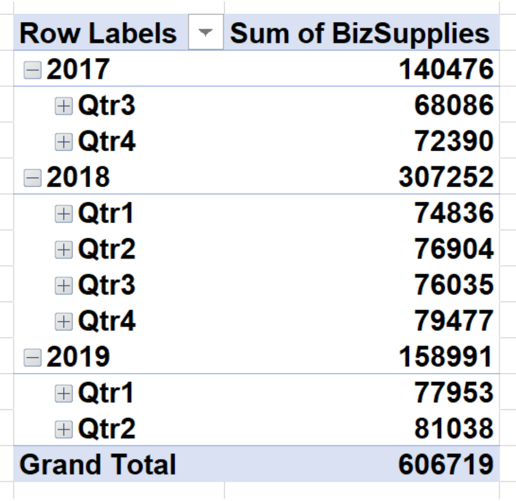
☑ **Years**

Drag fields between areas below:

| Rows | Values |
|---|---|
| Years | Sum of BizSupplies |
| Quarters | |
| Quarter | |

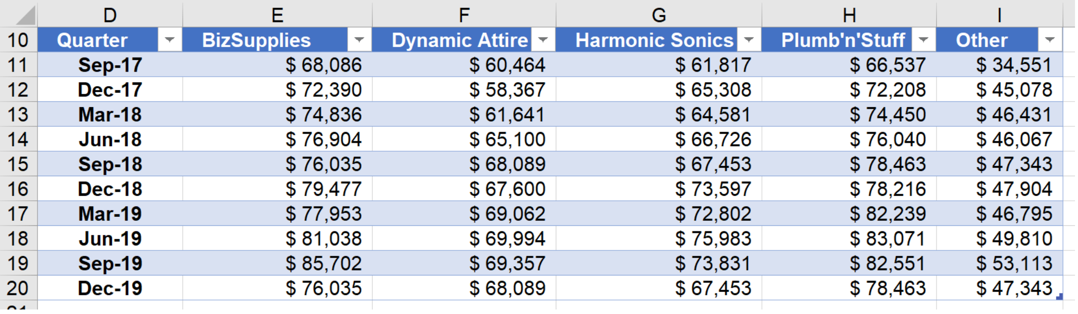
□ Defer Layout Update

This will produce the following PivotTable:



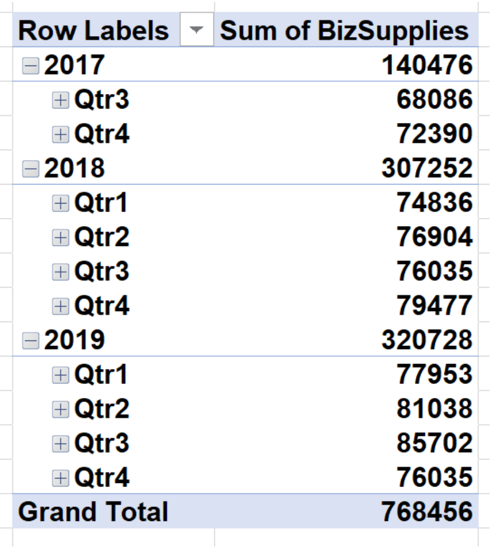
| Row Labels | Sum of BizSupplies |
|---|---|
| ⊟2017 | 140476 |
| ⊞Qtr3 | 68086 |
| ⊞Qtr4 | 72390 |
| ⊟2018 | 307252 |
| ⊞Qtr1 | 74836 |
| ⊞Qtr2 | 76904 |
| ⊞Qtr3 | 76035 |
| ⊞Qtr4 | 79477 |
| ⊟2019 | 158991 |
| ⊞Qtr1 | 77953 |
| ⊞Qtr2 | 81038 |
| Grand Total | 606719 |

Then in the existing Table, if we add new sales data (as seen, Sep-19 and Dec-19):



| | D | E | F | G | H | I |
|---|---|---|---|---|---|---|
| 10 | Quarter | BizSupplies | Dynamic Attire | Harmonic Sonics | Plumb'n'Stuff | Other |
| 11 | Sep-17 | $ 68,086 | $ 60,464 | $ 61,817 | $ 66,537 | $ 34,551 |
| 12 | Dec-17 | $ 72,390 | $ 58,367 | $ 65,308 | $ 72,208 | $ 45,078 |
| 13 | Mar-18 | $ 74,836 | $ 61,641 | $ 64,581 | $ 74,450 | $ 46,431 |
| 14 | Jun-18 | $ 76,904 | $ 65,100 | $ 66,726 | $ 76,040 | $ 46,067 |
| 15 | Sep-18 | $ 76,035 | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | $ 73,597 | $ 78,216 | $ 47,904 |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | $ 72,802 | $ 82,239 | $ 46,795 |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | $ 75,983 | $ 83,071 | $ 49,810 |
| 19 | Sep-19 | $ 85,702 | $ 69,357 | $ 73,831 | $ 82,551 | $ 53,113 |
| 20 | Dec-19 | $ 76,035 | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 |

The PivotTable will reflect newly added data, as seen under **2019: Qtr3** and **Qtr4**.

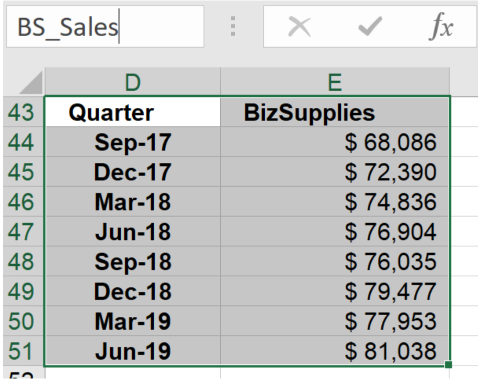


| Row Labels | Sum of BizSupplies |
|---|---|
| ⊟2017 | 140476 |
| ⊞Qtr3 | 68086 |
| ⊞Qtr4 | 72390 |
| ⊟2018 | 307252 |
| ⊞Qtr1 | 74836 |
| ⊞Qtr2 | 76904 |
| ⊞Qtr3 | 76035 |
| ⊞Qtr4 | 79477 |
| ⊟2019 | 320728 |
| ⊞Qtr1 | 77953 |
| ⊞Qtr2 | 81038 |
| ⊞Qtr3 | 85702 |
| ⊞Qtr4 | 76035 |
| Grand Total | 768456 |

Dynamic Tables are used to make sure that whenever a data range is updated or modified, the report associated with it will be changed accordingly. An alternative treatment involves **OFFSET**.

***Using the OFFSET function***

To begin, let’s select the data and give it a name in the Name box, *viz.*

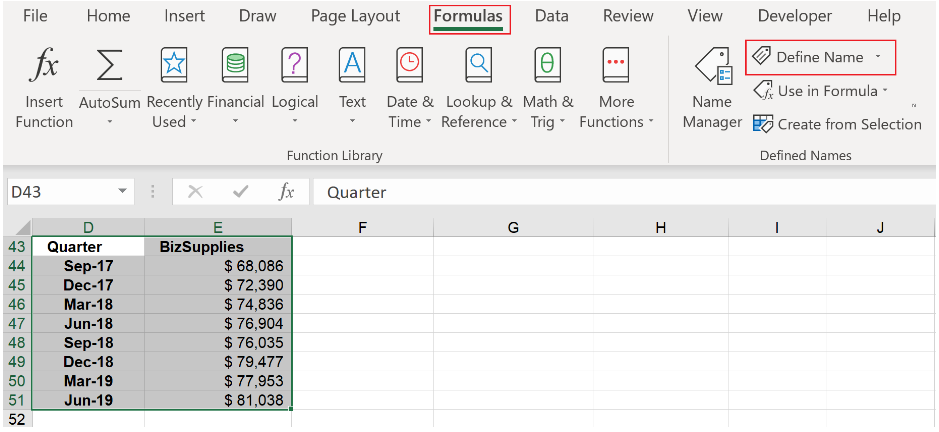


BS_Sales

| | D | E |
|---|---|---|
| 43 | Quarter | BizSupplies |
| 44 | Sep-17 | $ 68,086 |
| 45 | Dec-17 | $ 72,390 |
| 46 | Mar-18 | $ 74,836 |
| 47 | Jun-18 | $ 76,904 |
| 48 | Sep-18 | $ 76,035 |
| 49 | Dec-18 | $ 79,477 |
| 50 | Mar-19 | $ 77,953 |
| 51 | Jun-19 | $ 81,038 |

Now, whenever we refer to the dataset **BS_Sales**, it will point to the range **D43:E51** which contains the **BizSupplies** quarterly sales. If another row is added to the data, it will still point to the same range, as this range is static. However, we may use the **OFFSET** function to make this range dynamic.

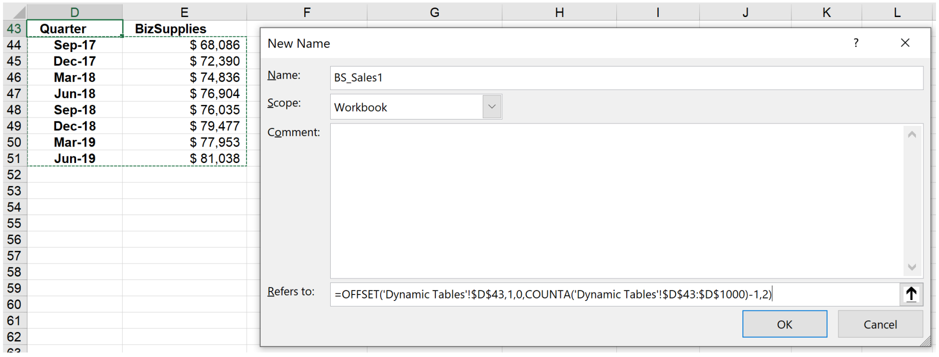
To do this, let’s select the range **D43:E51** and navigate to the Formulas tab on the Ribbon, where we may then ‘Define Name’.



D43 | Quarter

| | D | E |
|---|---|---|
| 43 | Quarter | BizSupplies |
| 44 | Sep-17 | $ 68,086 |
| 45 | Dec-17 | $ 72,390 |
| 46 | Mar-18 | $ 74,836 |
| 47 | Jun-18 | $ 76,904 |
| 48 | Sep-18 | $ 76,035 |
| 49 | Dec-18 | $ 79,477 |
| 50 | Mar-19 | $ 77,953 |
| 51 | Jun-19 | $ 81,038 |
| 52 | | |

The ‘New Name’ dialog will appear. I type the **OFFSET** formula into the ‘Refers to’ box as shown below:



New Name

Name: BS_Sales1

Scope: Workbook

Comment:

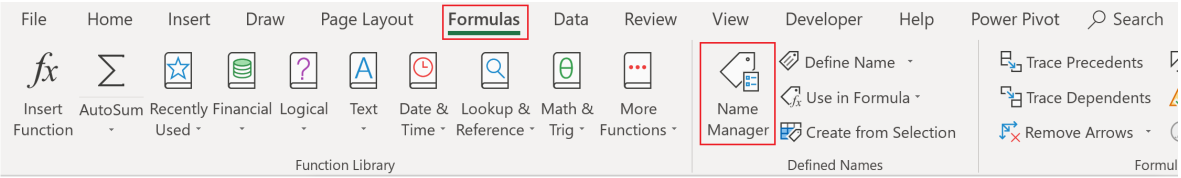
Refers to: =OFFSET('Dynamic Tables'!$D$43,1,0,COUNTA('Dynamic Tables'!$D$43:$D$1000)-1,2)

Let’s break down the **OFFSET** function formula:

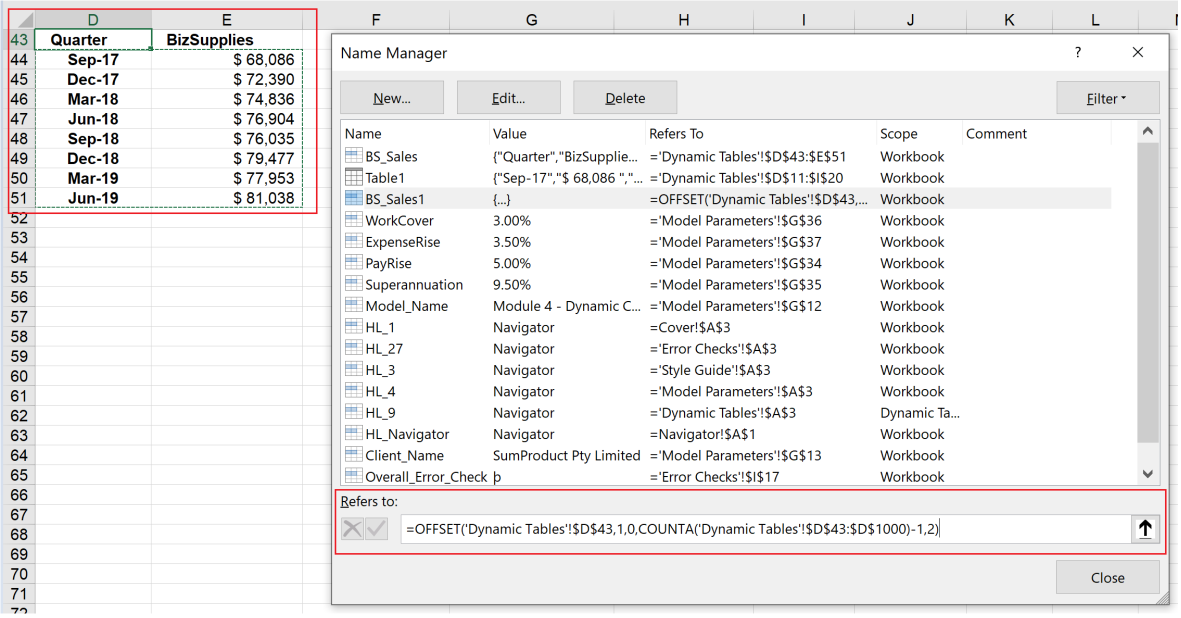
**=OFFSET('Dynamic Tables'!$D$43,1,0,COUNTA('Dynamic Tables'!$D$43:$D$1000)-1,2)**

- choose the starting cell, which, in this case, is **‘Dynamic Tables’!$D$43**
- we need to type **1,0** as it will count how many rows or columns to go
- we also need to count whatever the data is in Column **D**, *e.g.* range **D43:D1000** and use that as the number of rows, so use **COUNTA** function and select **‘Dynamic Tables’$D$43:$D$1000**
- since we do not want the first row (which is the **Quarter** header) to be counted, we have to subtract one [1]
- the number of columns will always be two [2].

If I go to the Formula tab and then click on 'Name Manager',



we may then click on the 'Refers to' field, where I can see the range of the Table:



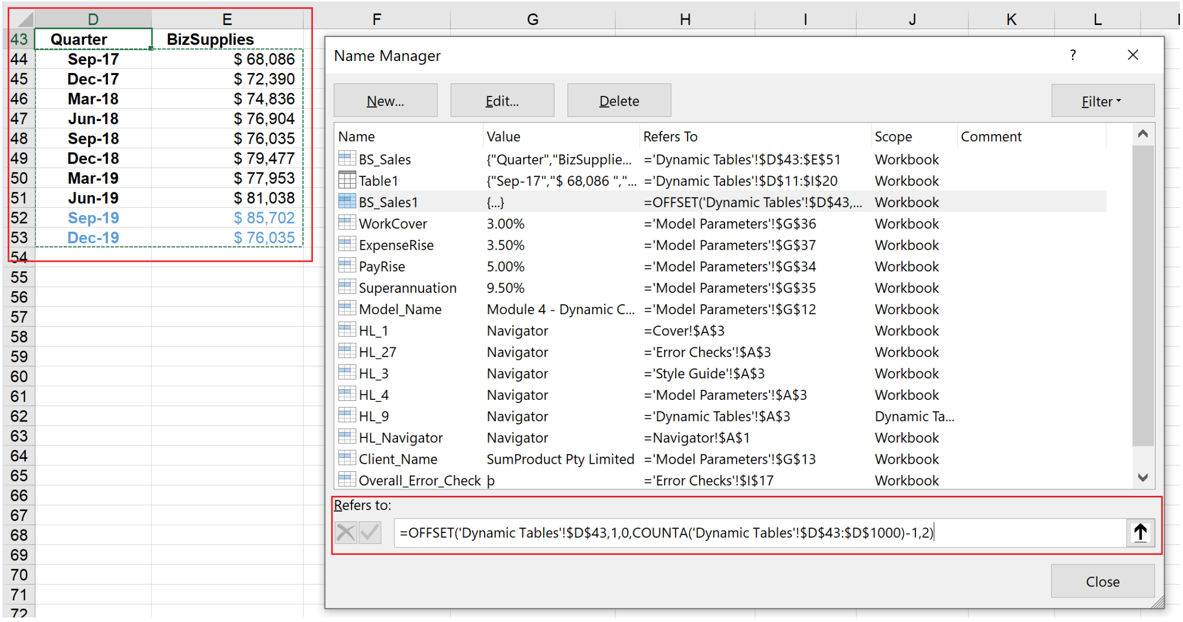
| | D | E |
|---|---|---|
| 43 | Quarter | BizSupplies |
| 44 | Sep-17 | $ 68,086 |
| 45 | Dec-17 | $ 72,390 |
| 46 | Mar-18 | $ 74,836 |
| 47 | Jun-18 | $ 76,904 |
| 48 | Sep-18 | $ 76,035 |
| 49 | Dec-18 | $ 79,477 |
| 50 | Mar-19 | $ 77,953 |
| 51 | Jun-19 | $ 81,038 |

| Name | Value | Refers To | Scope | Comment |
|---|---|---|---|---|
| BS_Sales | {"Quarter","BizSupplie... | ='Dynamic Tables'!$D$43:$E$51 | Workbook | |
| Table1 | {"Sep-17","$ 68,086 ","... | ='Dynamic Tables'!$D$11:$I$20 | Workbook | |
| BS_Sales1 | {...} | =OFFSET('Dynamic Tables'!$D$43,... | Workbook | |
| WorkCover | 3.00% | ='Model Parameters'!$G$36 | Workbook | |
| ExpenseRise | 3.50% | ='Model Parameters'!$G$37 | Workbook | |
| PayRise | 5.00% | ='Model Parameters'!$G$34 | Workbook | |
| Superannuation | 9.50% | ='Model Parameters'!$G$35 | Workbook | |
| Model_Name | Module 4 - Dynamic C... | ='Model Parameters'!$G$12 | Workbook | |
| HL_1 | Navigator | =Cover!$A$3 | Workbook | |
| HL_27 | Navigator | ='Error Checks'!$A$3 | Workbook | |
| HL_3 | Navigator | ='Style Guide'!$A$3 | Workbook | |
| HL_4 | Navigator | ='Model Parameters'!$A$3 | Workbook | |
| HL_9 | Navigator | ='Dynamic Tables'!$A$3 | Dynamic Ta... | |
| HL_Navigator | Navigator | =Navigator!$A$1 | Workbook | |
| Client_Name | SumProduct Pty Limited | ='Model Parameters'!$G$13 | Workbook | |
| Overall_Error_Check | þ | ='Error Checks'!$I$17 | Workbook | |

Refers to:

=OFFSET('Dynamic Tables'!$D$43,1,0,COUNTA('Dynamic Tables'!$D$43:$D$1000)-1,2)

If we add more data to the table (here as blue colour to make it clear) and then go back to **Formula -> 'Name Manager'** and click on the 'Refers to' field again, we will now see the range has been changed to cover the newly added data:



| | D | E |
|---|---|---|
| 43 | Quarter | BizSupplies |
| 44 | Sep-17 | $ 68,086 |
| 45 | Dec-17 | $ 72,390 |
| 46 | Mar-18 | $ 74,836 |
| 47 | Jun-18 | $ 76,904 |
| 48 | Sep-18 | $ 76,035 |
| 49 | Dec-18 | $ 79,477 |
| 50 | Mar-19 | $ 77,953 |
| 51 | Jun-19 | $ 81,038 |
| 52 | Sep-19 | $ 85,702 |
| 53 | Dec-19 | $ 76,035 |

| Name | Value | Refers To | Scope | Comment |
|---|---|---|---|---|
| BS_Sales | {"Quarter","BizSupplie... | ='Dynamic Tables'!$D$43:$E$51 | Workbook | |
| Table1 | {"Sep-17","$ 68,086 ","... | ='Dynamic Tables'!$D$11:$I$20 | Workbook | |
| BS_Sales1 | {...} | =OFFSET('Dynamic Tables'!$D$43,... | Workbook | |
| WorkCover | 3.00% | ='Model Parameters'!$G$36 | Workbook | |
| ExpenseRise | 3.50% | ='Model Parameters'!$G$37 | Workbook | |
| PayRise | 5.00% | ='Model Parameters'!$G$34 | Workbook | |
| Superannuation | 9.50% | ='Model Parameters'!$G$35 | Workbook | |
| Model_Name | Module 4 - Dynamic C... | ='Model Parameters'!$G$12 | Workbook | |
| HL_1 | Navigator | =Cover!$A$3 | Workbook | |
| HL_27 | Navigator | ='Error Checks'!$A$3 | Workbook | |
| HL_3 | Navigator | ='Style Guide'!$A$3 | Workbook | |
| HL_4 | Navigator | ='Model Parameters'!$A$3 | Workbook | |
| HL_9 | Navigator | ='Dynamic Tables'!$A$3 | Dynamic Ta... | |
| HL_Navigator | Navigator | =Navigator!$A$1 | Workbook | |
| Client_Name | SumProduct Pty Limited | ='Model Parameters'!$G$13 | Workbook | |
| Overall_Error_Check | þ | ='Error Checks'!$I$17 | Workbook | |

Refers to:

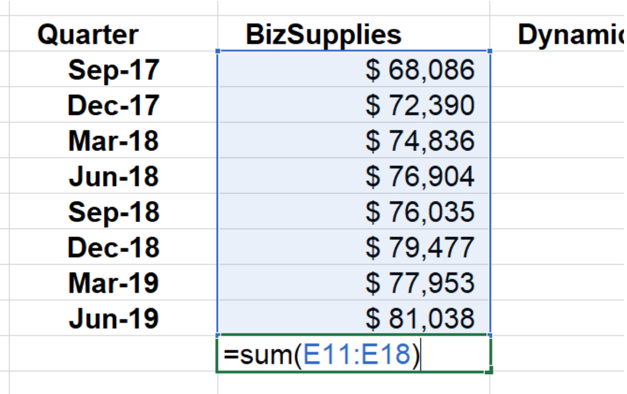
=OFFSET('Dynamic Tables'!$D$43,1,0,COUNTA('Dynamic Tables'!$D$43:$D$1000)-1,2)

The table has now become dynamic.

*Dynamic Named Ranges*

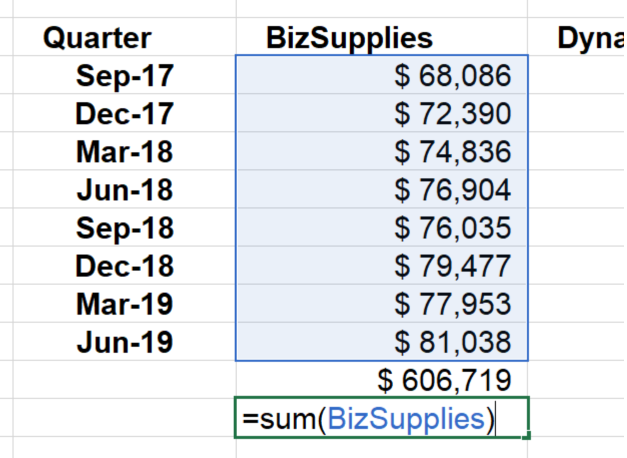
To recap, why do we need Dynamic Tables or Dynamic Ranges? The reason is whenever a list or data range is updated or modified, it is not certain that the report will be changed as per the data change. Dynamic Tables or Dynamic Range will help you avoid this problem.

A Dynamic Range in Excel is a named range. An Excel Dynamic Named range is one of the more powerful techniques in Excel. Let's look at the example below:



| Quarter | BizSupplies | Dynami |
|---|---|---|
| Sep-17 | $ 68,086 | |
| Dec-17 | $ 72,390 | |
| Mar-18 | $ 74,836 | |
| Jun-18 | $ 76,904 | |
| Sep-18 | $ 76,035 | |
| Dec-18 | $ 79,477 | |
| Mar-19 | $ 77,953 | |
| Jun-19 | $ 81,038 | |
| | =sum(E11:E18) | |

vs.



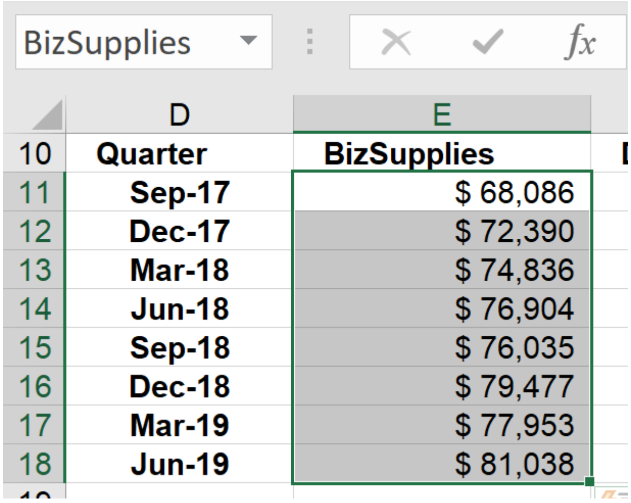
| Quarter | BizSupplies | Dyna |
|---|---|---|
| Sep-17 | $ 68,086 | |
| Dec-17 | $ 72,390 | |
| Mar-18 | $ 74,836 | |
| Jun-18 | $ 76,904 | |
| Sep-18 | $ 76,035 | |
| Dec-18 | $ 79,477 | |
| Mar-19 | $ 77,953 | |
| Jun-19 | $ 81,038 | |
| | $ 606,719 | |
| | =sum(BizSupplies) | |

In the first figure's summation, the range is from **E11:E18**, while in the second figure's total, it is showing as **BizSupplies**. The reason is that the range **B2:B10** has been named **BizSupplies**.

For a small set of data, it is not necessarily that a named range be applied. However, Dynamic Named Ranges make it much easier especially when you are working in larger set of data. Dynamic Named Ranges save us from going back to find and select targeted range of cells. Instead, we can just type the name which we have provided for that range.

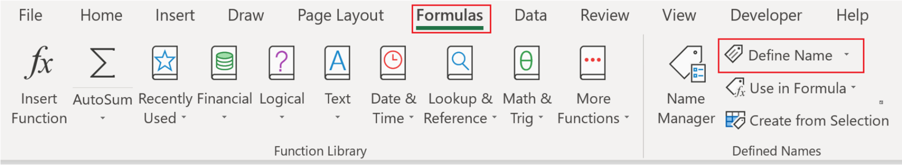
Creating named ranges in Excel is straightforward.

To do this, we may add a name for a range in the 'Name Box', after choosing the entire range that we wish to name:

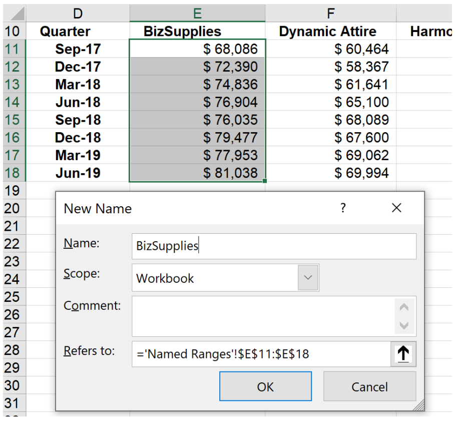


| | D | E |
|---|---|---|
| 10 | Quarter | BizSupplies |
| 11 | Sep-17 | $ 68,086 |
| 12 | Dec-17 | $ 72,390 |
| 13 | Mar-18 | $ 74,836 |
| 14 | Jun-18 | $ 76,904 |
| 15 | Sep-18 | $ 76,035 |
| 16 | Dec-18 | $ 79,477 |
| 17 | Mar-19 | $ 77,953 |
| 18 | Jun-19 | $ 81,038 |

Alternatively, we may also go to the Formulas tab on the Ribbon:



Then, the 'New Name' dialog will pop up,  Here, we may fill in the 'Name', and select the range in the 'Refers to' field:



| | D | E | F | |
|---|---|---|---|---|
| 10 | Quarter | BizSupplies | Dynamic Attire | Harmc |
| 11 | Sep-17 | $ 68,086 | $ 60,464 | |
| 12 | Dec-17 | $ 72,390 | $ 58,367 | |
| 13 | Mar-18 | $ 74,836 | $ 61,641 | |
| 14 | Jun-18 | $ 76,904 | $ 65,100 | |
| 15 | Sep-18 | $ 76,035 | $ 68,089 | |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | |

New Name

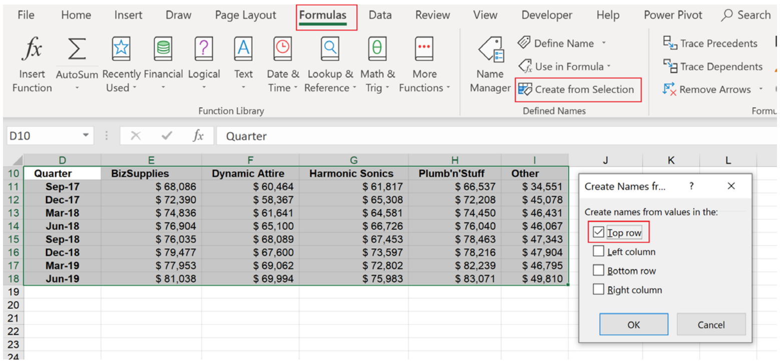
Name: BizSupplies

Scope: Workbook

Comment:

Refers to: ='Named Ranges'!$E$11:$E$18

I can also create named ranges automatically when my data contains a large number of columns, by first choosing the whole data set, then click **Formulas –> Name Manager –> Create from Selection**.  The 'Create Names from Selection' dialog will appear, where I may choose 'Top row' as the column name to use as the name for the range:



D10 Quarter

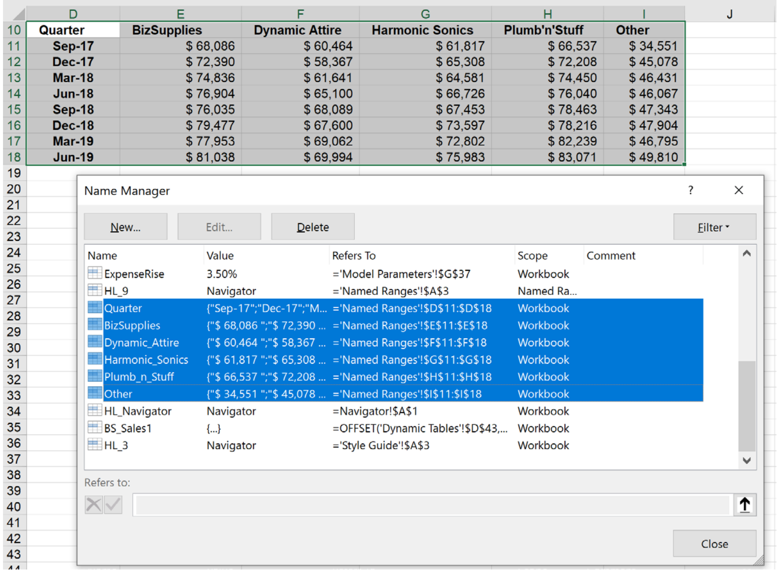
| | D | E | F | G | H | I |
|---|---|---|---|---|---|---|
| 10 | Quarter | BizSupplies | Dynamic Attire | Harmonic Sonics | Plumb'n'Stuff | Other |
| 11 | Sep-17 | $ 68,086 | $ 60,464 | $ 61,817 | $ 66,537 | $ 34,551 |
| 12 | Dec-17 | $ 72,390 | $ 58,367 | $ 65,308 | $ 72,208 | $ 45,078 |
| 13 | Mar-18 | $ 74,836 | $ 61,641 | $ 64,581 | $ 74,450 | $ 46,431 |
| 14 | Jun-18 | $ 76,904 | $ 65,100 | $ 66,726 | $ 76,040 | $ 46,067 |
| 15 | Sep-18 | $ 76,035 | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | $ 73,597 | $ 78,216 | $ 47,904 |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | $ 72,802 | $ 82,239 | $ 46,795 |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | $ 75,983 | $ 83,071 | $ 49,810 |

Create Names fr...

Create names from values in the:

☑ Top row
☐ Left column
☐ Bottom row
☐ Right column

After that, when I choose the **Formulas –> Name Manager**, the list of names should be displayed:



| | D | E | F | G | H | I |
|---|---|---|---|---|---|---|
| 10 | Quarter | BizSupplies | Dynamic Attire | Harmonic Sonics | Plumb'n'Stuff | Other |
| 11 | Sep-17 | $ 68,086 | $ 60,464 | $ 61,817 | $ 66,537 | $ 34,551 |
| 12 | Dec-17 | $ 72,390 | $ 58,367 | $ 65,308 | $ 72,208 | $ 45,078 |
| 13 | Mar-18 | $ 74,836 | $ 61,641 | $ 64,581 | $ 74,450 | $ 46,431 |
| 14 | Jun-18 | $ 76,904 | $ 65,100 | $ 66,726 | $ 76,040 | $ 46,067 |
| 15 | Sep-18 | $ 76,035 | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | $ 73,597 | $ 78,216 | $ 47,904 |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | $ 72,802 | $ 82,239 | $ 46,795 |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | $ 75,983 | $ 83,071 | $ 49,810 |

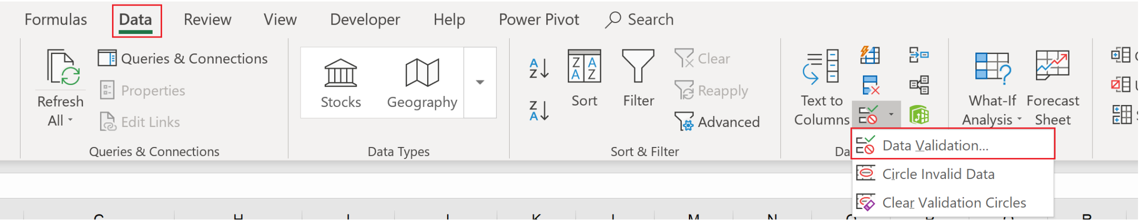
Name Manager

| Name | Value | Refers To | Scope | Comment |
|---|---|---|---|---|
| ExpenseRise | 3.50% | ='Model Parameters'!$G$37 | Workbook | |
| HL_9 | Navigator | ='Named Ranges'!$A$3 | Named Ra... | |
| Quarter | {"Sep-17";"Dec-17";"M... | ='Named Ranges'!$D$11:$D$18 | Workbook | |
| BizSupplies | {"$ 68,086 ";"$ 72,390 ... | ='Named Ranges'!$E$11:$E$18 | Workbook | |
| Dynamic_Attire | {"$ 60,464 ";"$ 58,367 ... | ='Named Ranges'!$F$11:$F$18 | Workbook | |
| Harmonic_Sonics | {"$ 61,817 ";"$ 65,308 ... | ='Named Ranges'!$G$11:$G$18 | Workbook | |
| Plumb_n_Stuff | {"$ 66,537 ";"$ 72,208 ... | ='Named Ranges'!$H$11:$H$18 | Workbook | |
| Other | {"$ 34,551 ";"$ 45,078 ... | ='Named Ranges'!$I$11:$I$18 | Workbook | |
| HL_Navigator | Navigator | =Navigator!$A$1 | Workbook | |
| BS_Sales1 | {...} | =OFFSET('Dynamic Tables'!$D$43,... | Workbook | |
| HL_3 | Navigator | ='Style Guide'!$A$3 | Workbook | |

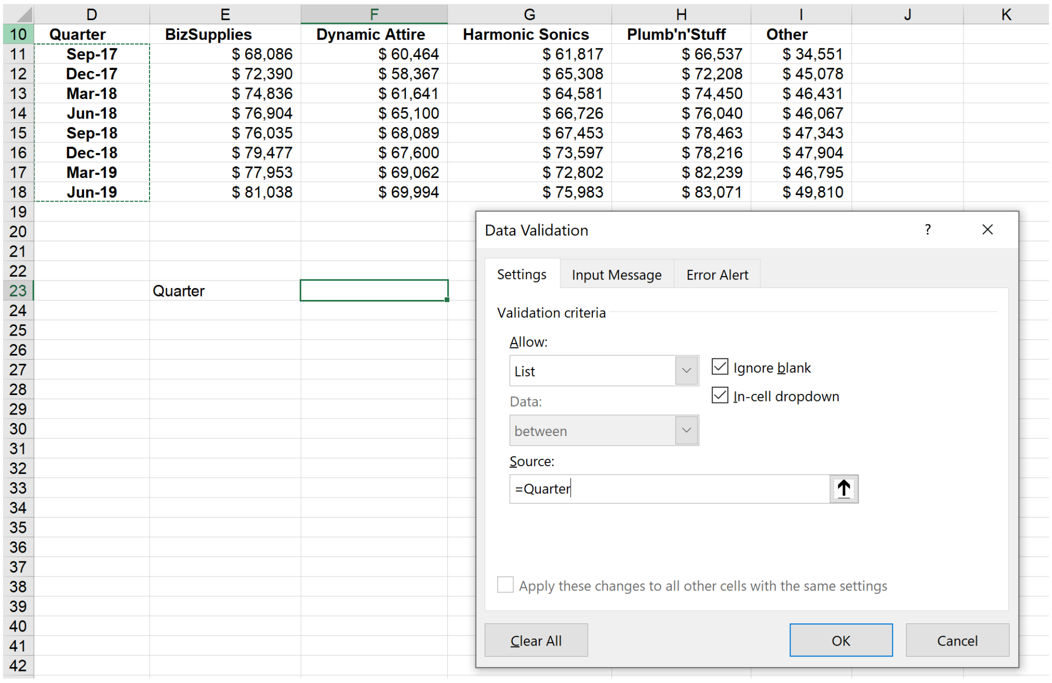
Refers to:

This is just the first step. In a simple named range, it will take only the predetermined range. In the case of data expansion, it will not expand dynamically. Therefore, we need to create a Dynamic Range in Excel. When we define an Excel Dynamic Named Range, as new items are added, the range will automatically expand.

To create this, let’s apply data validation to the named range that has already been set, navigating to the Data tab on the Ribbon and choosing ‘Data Validation’:



Change the validation criteria as in the below dialog:



| | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|
| 10 | Quarter | BizSupplies | Dynamic Attire | Harmonic Sonics | Plumb'n'Stuff | Other | | |
| 11 | Sep-17 | $ 68,086 | $ 60,464 | $ 61,817 | $ 66,537 | $ 34,551 | | |
| 12 | Dec-17 | $ 72,390 | $ 58,367 | $ 65,308 | $ 72,208 | $ 45,078 | | |
| 13 | Mar-18 | $ 74,836 | $ 61,641 | $ 64,581 | $ 74,450 | $ 46,431 | | |
| 14 | Jun-18 | $ 76,904 | $ 65,100 | $ 66,726 | $ 76,040 | $ 46,067 | | |
| 15 | Sep-18 | $ 76,035 | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 | | |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | $ 73,597 | $ 78,216 | $ 47,904 | | |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | $ 72,802 | $ 82,239 | $ 46,795 | | |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | $ 75,983 | $ 83,071 | $ 49,810 | | |
| 23 | | Quarter | | | | | | |

Data Validation

Settings | Input Message | Error Alert

Validation criteria

Allow: List

☑ Ignore blank

☑ In-cell dropdown

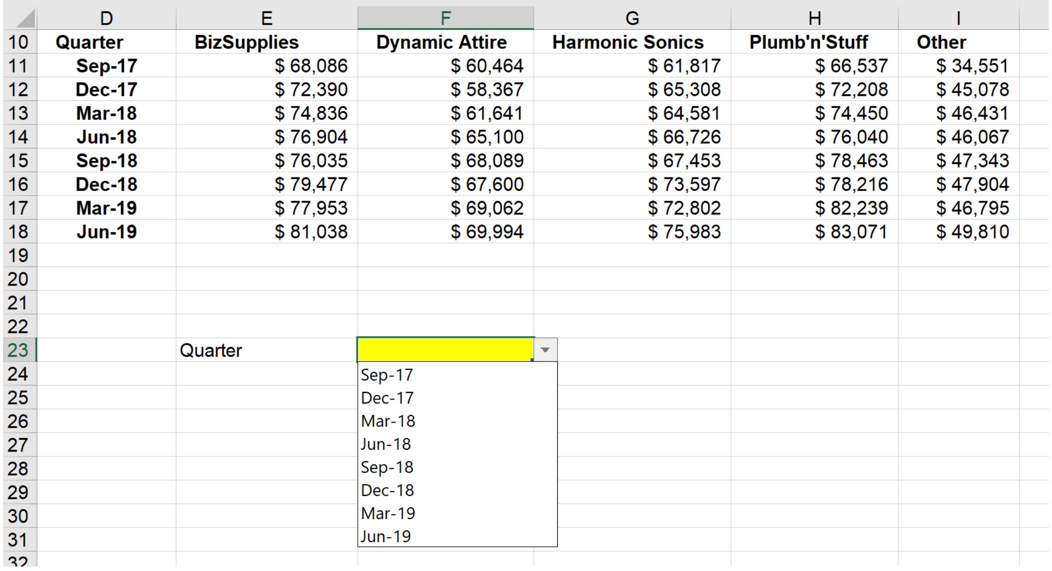
Data: between

Source: =Quarter

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

Clear All | OK | Cancel

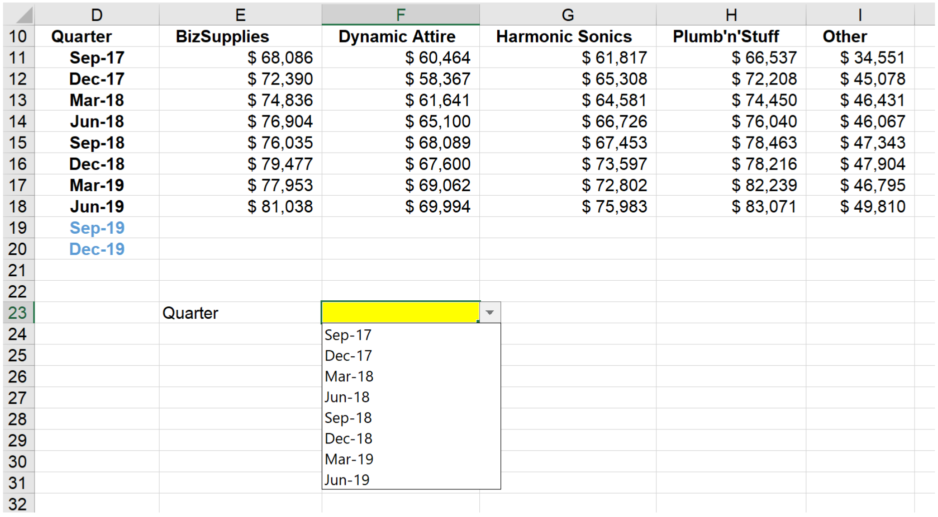
Now that the ‘Data Validation’ dialog for **Quarter** has been created, it displays all of the range in the **Quarter** column:



| | D | E | F | G | H | I |
|---|---|---|---|---|---|---|
| 10 | Quarter | BizSupplies | Dynamic Attire | Harmonic Sonics | Plumb'n'Stuff | Other |
| 11 | Sep-17 | $ 68,086 | $ 60,464 | $ 61,817 | $ 66,537 | $ 34,551 |
| 12 | Dec-17 | $ 72,390 | $ 58,367 | $ 65,308 | $ 72,208 | $ 45,078 |
| 13 | Mar-18 | $ 74,836 | $ 61,641 | $ 64,581 | $ 74,450 | $ 46,431 |
| 14 | Jun-18 | $ 76,904 | $ 65,100 | $ 66,726 | $ 76,040 | $ 46,067 |
| 15 | Sep-18 | $ 76,035 | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | $ 73,597 | $ 78,216 | $ 47,904 |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | $ 72,802 | $ 82,239 | $ 46,795 |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | $ 75,983 | $ 83,071 | $ 49,810 |
| 23 | | Quarter | | | | |

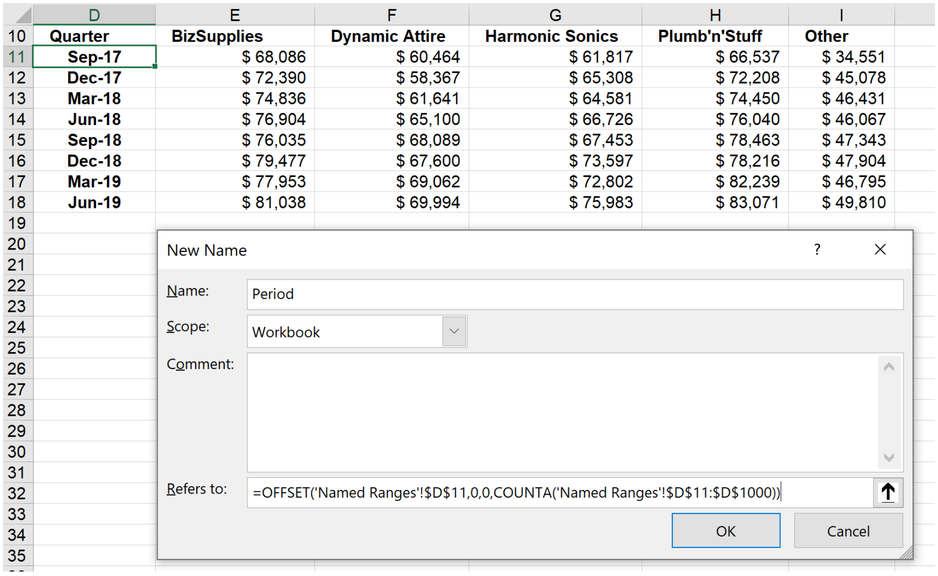
Sep-17
Dec-17
Mar-18
Jun-18
Sep-18
Dec-18
Mar-19
Jun-19

However, if we add more values to the column, these newly added values will not appear in the dropdown cell.  This is because the range is not dynamic.



| | D | E | F | G | H | I |
|---|---|---|---|---|---|---|
| 10 | Quarter | BizSupplies | Dynamic Attire | Harmonic Sonics | Plumb'n'Stuff | Other |
| 11 | Sep-17 | $ 68,086 | $ 60,464 | $ 61,817 | $ 66,537 | $ 34,551 |
| 12 | Dec-17 | $ 72,390 | $ 58,367 | $ 65,308 | $ 72,208 | $ 45,078 |
| 13 | Mar-18 | $ 74,836 | $ 61,641 | $ 64,581 | $ 74,450 | $ 46,431 |
| 14 | Jun-18 | $ 76,904 | $ 65,100 | $ 66,726 | $ 76,040 | $ 46,067 |
| 15 | Sep-18 | $ 76,035 | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | $ 73,597 | $ 78,216 | $ 47,904 |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | $ 72,802 | $ 82,239 | $ 46,795 |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | $ 75,983 | $ 83,071 | $ 49,810 |
| 19 | Sep-19 | | | | | |
| 20 | Dec-19 | | | | | |
| 21 | | | | | | |
| 22 | | | | | | |
| 23 | | Quarter | | | | |
| 24 | | | Sep-17 | | | |
| 25 | | | Dec-17 | | | |
| 26 | | | Mar-18 | | | |
| 27 | | | Jun-18 | | | |
| 28 | | | Sep-18 | | | |
| 29 | | | Dec-18 | | | |
| 30 | | | Mar-19 | | | |
| 31 | | | Jun-19 | | | |
| 32 | | | | | | |

To fix and compare this, I create a new named range called **Period**, now using the **OFFSET** function as below, similar to the example provided with Dynamic Tables so that newly added values will be counted:



| | D | E | F | G | H | I |
|---|---|---|---|---|---|---|
| 10 | Quarter | BizSupplies | Dynamic Attire | Harmonic Sonics | Plumb'n'Stuff | Other |
| 11 | Sep-17 | $ 68,086 | $ 60,464 | $ 61,817 | $ 66,537 | $ 34,551 |
| 12 | Dec-17 | $ 72,390 | $ 58,367 | $ 65,308 | $ 72,208 | $ 45,078 |
| 13 | Mar-18 | $ 74,836 | $ 61,641 | $ 64,581 | $ 74,450 | $ 46,431 |
| 14 | Jun-18 | $ 76,904 | $ 65,100 | $ 66,726 | $ 76,040 | $ 46,067 |
| 15 | Sep-18 | $ 76,035 | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | $ 73,597 | $ 78,216 | $ 47,904 |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | $ 72,802 | $ 82,239 | $ 46,795 |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | $ 75,983 | $ 83,071 | $ 49,810 |

New Name

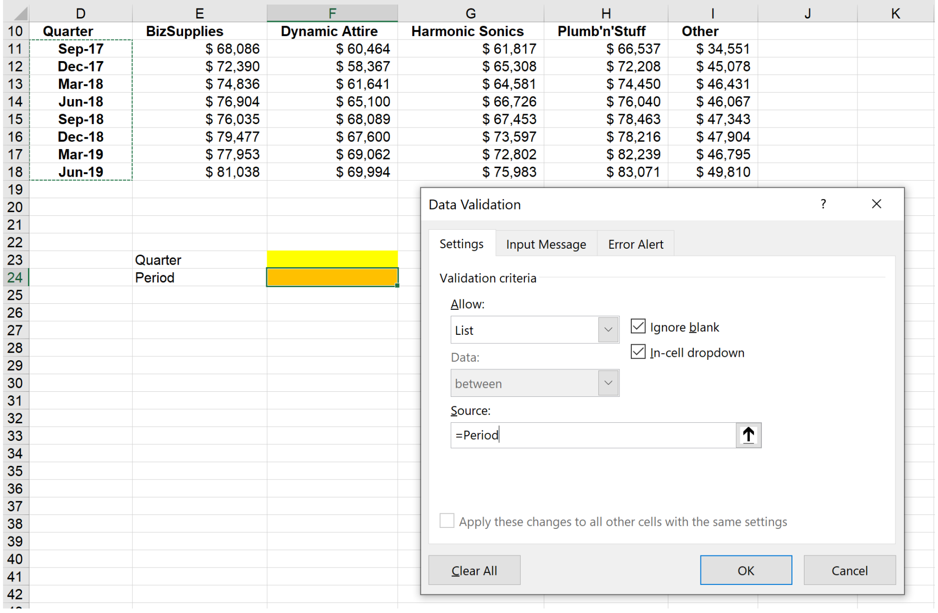
Name: Period

Scope: Workbook

Comment:

Refers to: =OFFSET('Named Ranges'!$D$11,0,0,COUNTA('Named Ranges'!$D$11:$D$1000))

Then, we call the 'Data Validation' dialog.  This time, the 'Source' will be the named range which has just been created using the **OFFSET** function, *i.e.* **Period**:



| | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|
| 10 | Quarter | BizSupplies | Dynamic Attire | Harmonic Sonics | Plumb'n'Stuff | Other | | |
| 11 | Sep-17 | $ 68,086 | $ 60,464 | $ 61,817 | $ 66,537 | $ 34,551 | | |
| 12 | Dec-17 | $ 72,390 | $ 58,367 | $ 65,308 | $ 72,208 | $ 45,078 | | |
| 13 | Mar-18 | $ 74,836 | $ 61,641 | $ 64,581 | $ 74,450 | $ 46,431 | | |
| 14 | Jun-18 | $ 76,904 | $ 65,100 | $ 66,726 | $ 76,040 | $ 46,067 | | |
| 15 | Sep-18 | $ 76,035 | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 | | |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | $ 73,597 | $ 78,216 | $ 47,904 | | |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | $ 72,802 | $ 82,239 | $ 46,795 | | |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | $ 75,983 | $ 83,071 | $ 49,810 | | |
| 23 | | Quarter | | | | | | |
| 24 | | Period | | | | | | |

Data Validation

Settings | Input Message | Error Alert

Validation criteria

Allow: List

☑ Ignore blank

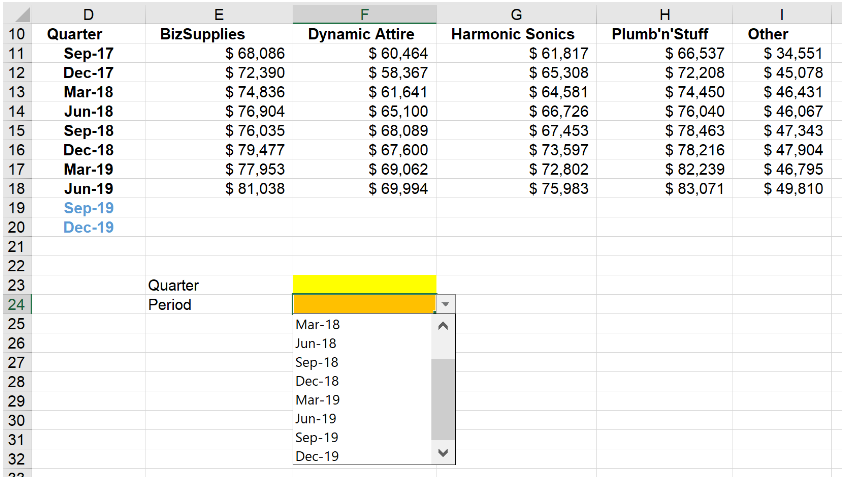
☑ In-cell dropdown

Data: between

Source: =Period

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

Here, we add values to the **Quarter** column. As new reports appear, we may check that in the **Period** validation, new values are now displayed:



| | D | E | F | G | H | I |
|---|---|---|---|---|---|---|
| 10 | Quarter | BizSupplies | Dynamic Attire | Harmonic Sonics | Plumb'n'Stuff | Other |
| 11 | Sep-17 | $ 68,086 | $ 60,464 | $ 61,817 | $ 66,537 | $ 34,551 |
| 12 | Dec-17 | $ 72,390 | $ 58,367 | $ 65,308 | $ 72,208 | $ 45,078 |
| 13 | Mar-18 | $ 74,836 | $ 61,641 | $ 64,581 | $ 74,450 | $ 46,431 |
| 14 | Jun-18 | $ 76,904 | $ 65,100 | $ 66,726 | $ 76,040 | $ 46,067 |
| 15 | Sep-18 | $ 76,035 | $ 68,089 | $ 67,453 | $ 78,463 | $ 47,343 |
| 16 | Dec-18 | $ 79,477 | $ 67,600 | $ 73,597 | $ 78,216 | $ 47,904 |
| 17 | Mar-19 | $ 77,953 | $ 69,062 | $ 72,802 | $ 82,239 | $ 46,795 |
| 18 | Jun-19 | $ 81,038 | $ 69,994 | $ 75,983 | $ 83,071 | $ 49,810 |
| 19 | Sep-19 | | | | | |
| 20 | Dec-19 | | | | | |
| 21 | | | | | | |
| 22 | | | | | | |
| 23 | | Quarter | | | | |
| 24 | | Period | | | | |

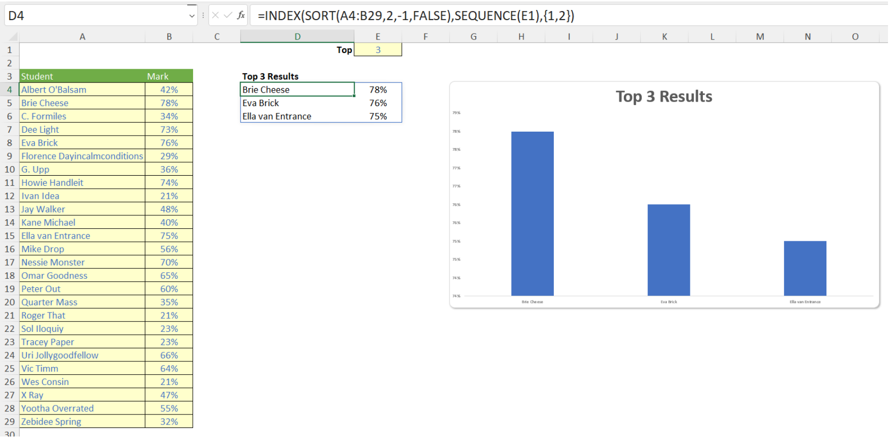
Mar-18
Jun-18
Sep-18
Dec-18
Mar-19
Jun-19
Sep-19
Dec-19

### *Dynamic Arrays*

It's a long article this month! We couldn't finish a discussion on dynamic charts without referring to November's news. In Excel for the web and Excel Desktop (Insiders Beta), charts should now respond to dynamic arrays. This will make things simpler.

For those who have access, you may now create a chart with a data source range aligned to the result of an array formula. The chart will now update to capture all data whenever the array recalculates, rather than being fixed to a specific number of data points. Yippee!

For example, consider the following:



D4 =INDEX(SORT(A4:B29,2,-1,FALSE),SEQUENCE(E1),{1,2})

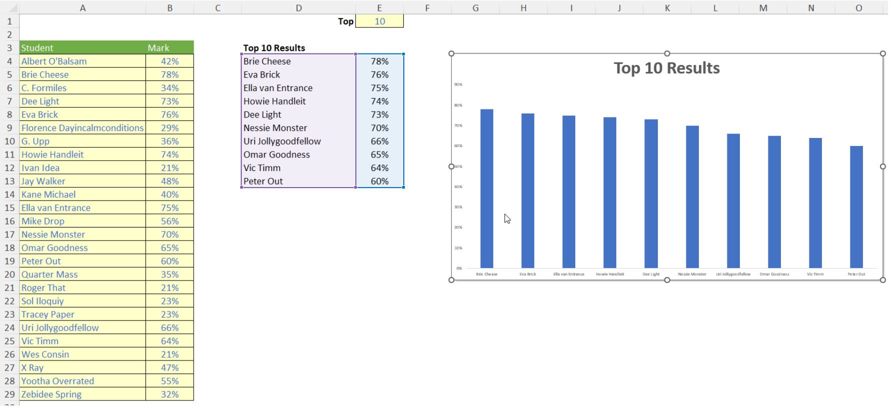
| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | | Top | 3 |
| 2 | | | | | |
| 3 | Student | Mark | | Top 3 Results | |
| 4 | Albert O'Balsam | 42% | | Brie Cheese | 78% |
| 5 | Brie Cheese | 78% | | Eva Brick | 76% |
| 6 | C. Formiles | 34% | | Ella van Entrance | 75% |
| 7 | Dee Light | 73% | | | |
| 8 | Eva Brick | 76% | | | |
| 9 | Florence Dayincalmconditions | 29% | | | |
| 10 | G. Upp | 36% | | | |
| 11 | Howie Handleit | 74% | | | |
| 12 | Ivan Idea | 21% | | | |
| 13 | Jay Walker | 48% | | | |
| 14 | Kane Michael | 40% | | | |
| 15 | Ella van Entrance | 75% | | | |
| 16 | Mike Drop | 56% | | | |
| 17 | Nessie Monster | 70% | | | |
| 18 | Omar Goodness | 65% | | | |
| 19 | Peter Out | 60% | | | |
| 20 | Quarter Mass | 35% | | | |
| 21 | Roger That | 21% | | | |
| 22 | Sol Iloquiy | 23% | | | |
| 23 | Tracey Paper | 23% | | | |
| 24 | Uri Jollygoodfellow | 66% | | | |
| 25 | Vic Timm | 64% | | | |
| 26 | Wes Consin | 21% | | | |
| 27 | X Ray | 47% | | | |
| 28 | Yootha Overrated | 55% | | | |
| 29 | Zebidee Spring | 32% | | | |

In cells **A4:B29** (purposely not placed in an Excel Table), we have entered the results of the Home Cookery & Poisoning (Joint Honours) vocational course. Cell **E1** contains an input number that specifies the top "**X**" students to chart, and the formula

**=INDEX(SORT(A4:B29,2,-1,FALSE),SEQUENCE(E1),{1,2})**

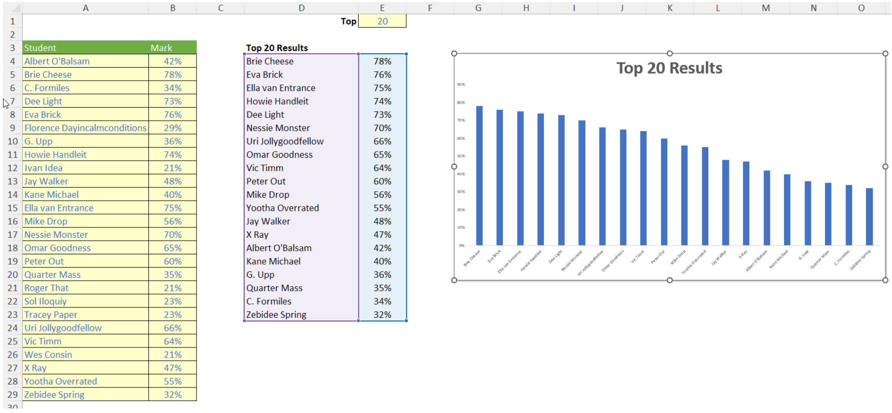
has been entered into cell **D4** as a dynamic array formula to summarise the said top **X** students and their respective marks.

Finally, a chart has been inserted linking to the dynamic range (cells **D4:E6** in the above illustration) in the usual way (*e.g.* **Insert -> Recommended Charts**). Nothing that exciting so far, but then, let's change the vale in cell **E1** to 10 *(say)*:



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | | Top | 10 |
| 2 | | | | | |
| 3 | Student | Mark | | Top 10 Results | |
| 4 | Albert O'Balsam | 42% | | Brie Cheese | 78% |
| 5 | Brie Cheese | 78% | | Eva Brick | 76% |
| 6 | C. Formiles | 34% | | Ella van Entrance | 75% |
| 7 | Dee Light | 73% | | Howie Handleit | 74% |
| 8 | Eva Brick | 76% | | Dee Light | 73% |
| 9 | Florence Dayincalmconditions | 29% | | Nessie Monster | 70% |
| 10 | G. Upp | 36% | | Uri Jollygoodfellow | 66% |
| 11 | Howie Handleit | 74% | | Omar Goodness | 65% |
| 12 | Ivan Idea | 21% | | Vic Timm | 64% |
| 13 | Jay Walker | 48% | | Peter Out | 60% |
| 14 | Kane Michael | 40% | | | |
| 15 | Ella van Entrance | 75% | | | |
| 16 | Mike Drop | 56% | | | |
| 17 | Nessie Monster | 70% | | | |
| 18 | Omar Goodness | 65% | | | |
| 19 | Peter Out | 60% | | | |
| 20 | Quarter Mass | 35% | | | |
| 21 | Roger That | 21% | | | |
| 22 | Sol Iloquiy | 23% | | | |
| 23 | Tracey Paper | 23% | | | |
| 24 | Uri Jollygoodfellow | 66% | | | |
| 25 | Vic Timm | 64% | | | |
| 26 | Wes Consin | 21% | | | |
| 27 | X Ray | 47% | | | |
| 28 | Yootha Overrated | 55% | | | |
| 29 | Zebidee Spring | 32% | | | |

or even 20:



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | | Top | 20 |
| 2 | | | | | |
| 3 | Student | Mark | | Top 20 Results | |
| 4 | Albert O'Balsam | 42% | | Brie Cheese | 78% |
| 5 | Brie Cheese | 78% | | Eva Brick | 76% |
| 6 | C. Formiles | 34% | | Ella van Entrance | 75% |
| 7 | Dee Light | 73% | | Howie Handleit | 74% |
| 8 | Eva Brick | 76% | | Dee Light | 73% |
| 9 | Florence Dayincalmconditions | 29% | | Nessie Monster | 70% |
| 10 | G. Upp | 36% | | Uri Jollygoodfellow | 66% |
| 11 | Howie Handleit | 74% | | Omar Goodness | 65% |
| 12 | Ivan Idea | 21% | | Vic Timm | 64% |
| 13 | Jay Walker | 48% | | Peter Out | 60% |
| 14 | Kane Michael | 40% | | Mike Drop | 56% |
| 15 | Ella van Entrance | 75% | | Yootha Overrated | 55% |
| 16 | Mike Drop | 56% | | Jay Walker | 48% |
| 17 | Nessie Monster | 70% | | X Ray | 47% |
| 18 | Omar Goodness | 65% | | Albert O'Balsam | 42% |
| 19 | Peter Out | 60% | | Kane Michael | 40% |
| 20 | Quarter Mass | 35% | | G. Upp | 36% |
| 21 | Roger That | 21% | | Quarter Mass | 35% |
| 22 | Sol Iloquiy | 23% | | C. Formiles | 34% |
| 23 | Tracey Paper | 23% | | Zebidee Spring | 32% |
| 24 | Uri Jollygoodfellow | 66% | | | |
| 25 | Vic Timm | 64% | | | |
| 26 | Wes Consin | 21% | | | |
| 27 | X Ray | 47% | | | |
| 28 | Yootha Overrated | 55% | | | |
| 29 | Zebidee Spring | 32% | | | |

More next month…

## Visual Basics

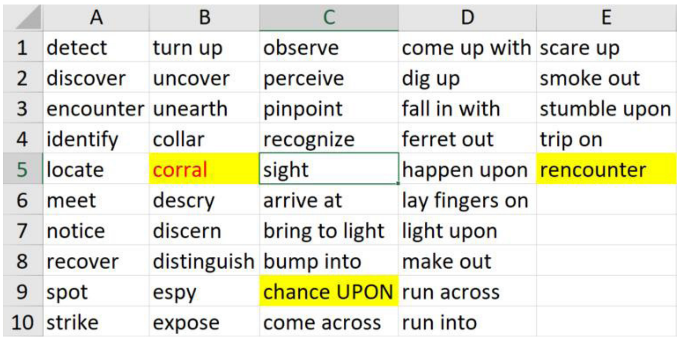
*We thought we'd run an elementary series going through the rudiments of Visual Basic for Applications (VBA) as a springboard for newer users. This month, we consider finding "types" of exact matches.*

Last month, we looked at using the **After**, **SearchDirection** and **SearchOrder** parameters of the **Find** method. But what if we needed an exact match? There are a few things we could match:

- matching the complete word
- matching the case of the word (*i.e.* the capitalisation)
- matching specific cell formatting.

Luckily, the **Find** method has parameters that can help us with that! For this month, we will look at the first option (can you guess what we might do next month..?).

The word field has been adjusted slightly to give us more interesting things to search (the changed squares are highlighted in yellow):

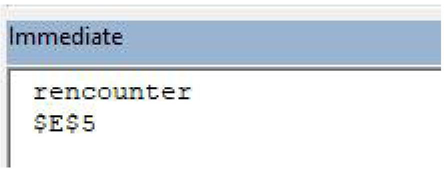


| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | detect | turn up | observe | come up with | scare up |
| 2 | discover | uncover | perceive | dig up | smoke out |
| 3 | encounter | unearth | pinpoint | fall in with | stumble upon |
| 4 | identify | collar | recognize | ferret out | trip on |
| 5 | locate | corral | sight | happen upon | rencounter |
| 6 | meet | descry | arrive at | lay fingers on | |
| 7 | notice | discern | bring to light | light upon | |
| 8 | recover | distinguish | bump into | make out | |
| 9 | spot | espy | chance UPON | run across | |
| 10 | strike | expose | come across | run into | |

If we were to search "encounter" from cell **C5** onwards, what would the result be? Let's give it a go:

```
Sub EncounterOfTheFirstKind()
    Dim searchRange As Range
    Set searchRange = Range("A1:E10")
    Dim foundrange As Range
    Set foundrange = searchRange.Find("encounter", After:=Range("C5"))

    If foundrange Is Nothing Then
        Debug.Print "not found!"
    Else
        Debug.Print foundrange
        Debug.Print foundrange.Address
    End If
End Sub
```

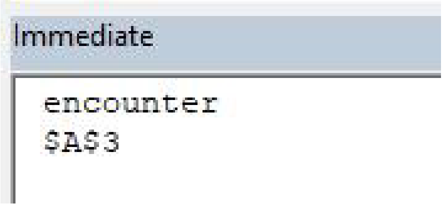


```
rencounter
$E$5
```

It technically found “encounter” albeit within “rencounter”. To force it to match the complete word, we would need to trigger the **LookAt** parameter. By defining it to be **xlWhole**, that will force it to look at the entire contents of the cell.

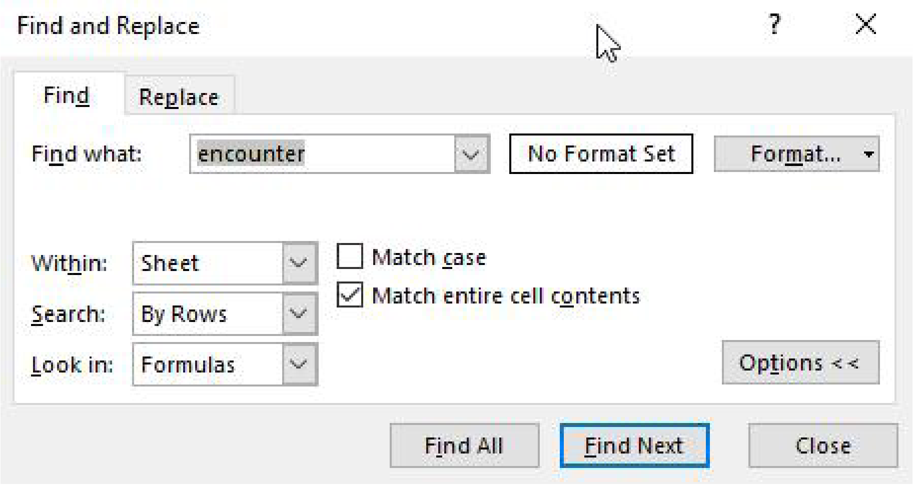
```
Sub FindWholeWord()
    Dim searchRange As Range
    Set searchRange = Range("A1:E10")
    Dim foundrange As Range
    Set foundrange = searchRange.Find("encounter", After:=Range("C5"), LookAt:=xlWhole)
    If foundrange Is Nothing Then
        Debug.Print "not found!"
    Else
        Debug.Print foundrange
        Debug.Print foundrange.Address
    End If
End Sub
```

This change will land on the expected answer of cell **A3**.



```
encounter
$A$3
```

Now, this is where things get interesting. If we were to run another search immediately after without the **LookAt:=xlWhole** portion, it would still maintain the parameter setting. This is because Excel saves this setting in the **Find** dialog. Opening the dialog up, you may view the advanced settings by clicking the "Options > >" button:



Find what: encounter

Within: Sheet

Search: By Rows

Look in: Formulas

□ Match case

☑ Match entire cell contents

There it is! Therefore, whenever doing a search using the **Find** method in VBA, ensure that you always set the parameters for each **Find** because otherwise it will lead to unexpected results.

Next newsletter, we’re going to keep **Find**ing things on a **Case-by-Case** basis...

Until then.

## Power Pivot Principles

*We continue our series on the Excel COM add-in, Power Pivot. This month, we discuss the **SUM** and **SUMX** functions.*

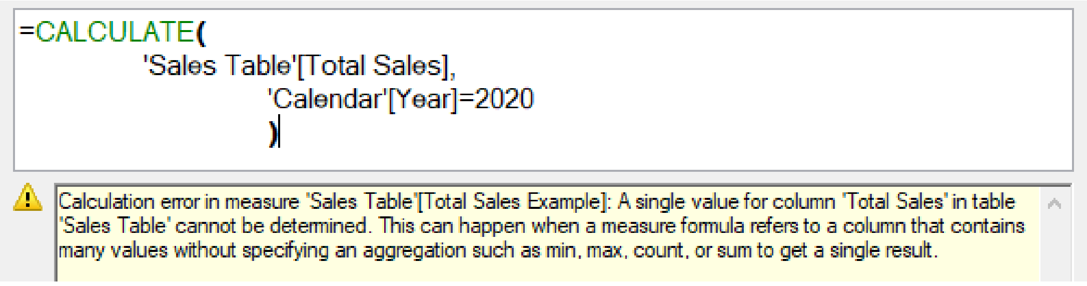
We've written over 50 articles in this newsletter on Power Pivot topics and it has just occurred to us that we have not formally covered the **SUM** function, despite using it on many occasions.

In short, the **SUM** function adds all the numbers referenced. The syntax is very straightforward:

**SUM(reference)**

The **SUM** function serves as an aggregation function, which allows us to use it again in other measures such as **CALCULATE**. For example, we will receive an error if we were to type this measure into the DAX editor:

```
=CALCULATE(
        'Sales Table'[Total Sales],
                'Calendar'[Year]=2020
                )
```



```
=CALCULATE(
        'Sales Table'[Total Sales],
                'Calendar'[Year]=2020
                )
```

Calculation error in measure 'Sales Table'[Total Sales Example]: A single value for column 'Total Sales' in table 'Sales Table' cannot be determined. This can happen when a measure formula refers to a column that contains many values without specifying an aggregation such as min, max, count, or sum to get a single result.

We have to create another measure with **SUM** to aggregate the column data before using it in our **CALCULATE** measure:

```
=SUM('Sales Table'[Total Sales])
```

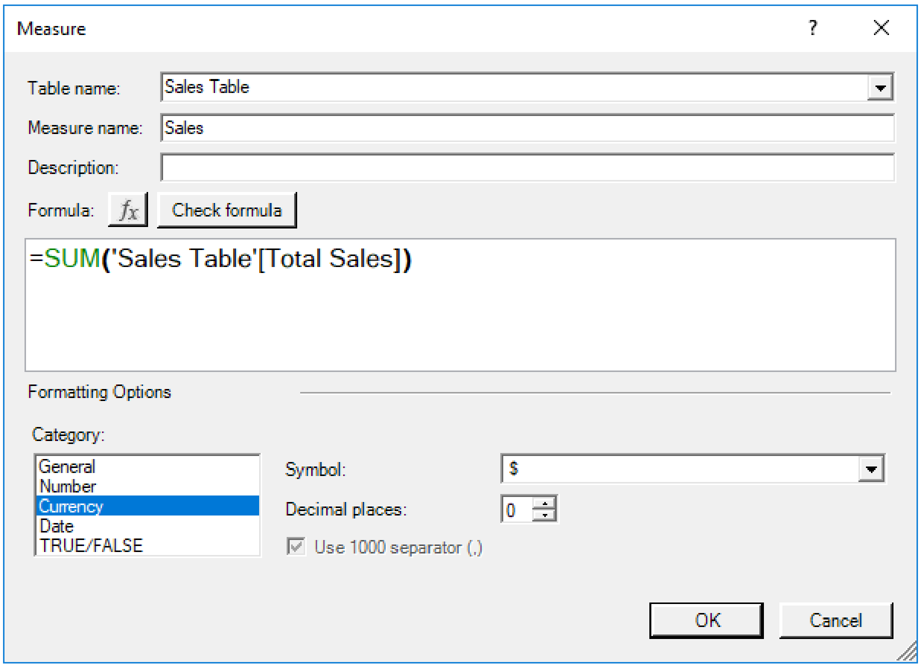


Table name: Sales Table

Measure name: Sales

Description:

```
=SUM('Sales Table'[Total Sales])
```

Formatting Options

Category:
- General
- Number
- Currency
- Date
- TRUE/FALSE

Symbol: $

Decimal places: 0

☑ Use 1000 separator (,)

We can now insert the **[Sales]** measure that into our **CALCULATE** measure:

```
=CALCULATE(
        [Sales],
                'Calendar'[Year]=2020
                )
```

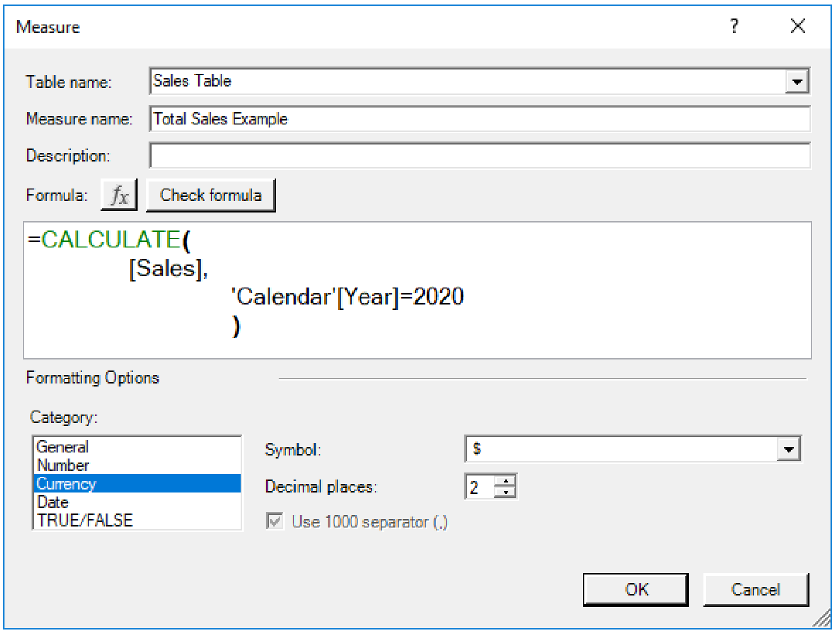


Table name: Sales Table

Measure name: Total Sales Example

Description:

```
=CALCULATE(
          [Sales],
                    'Calendar'[Year]=2020
                    )
```

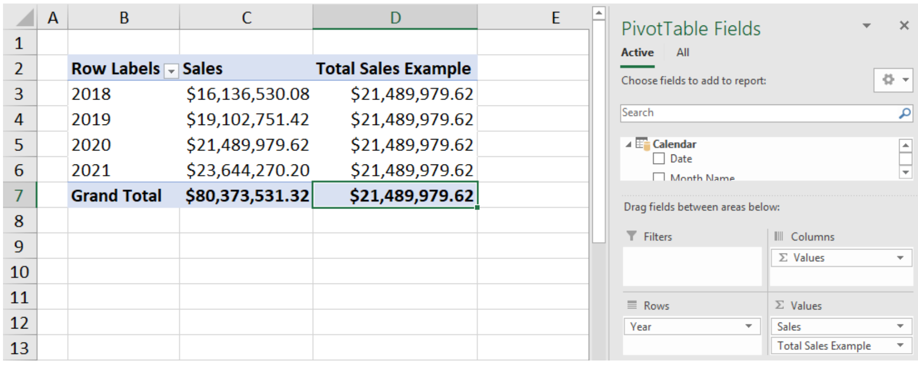
Category: General, Number, Currency (selected), Date, TRUE/FALSE

Symbol: $

Decimal places: 2

☑ Use 1000 separator (,)

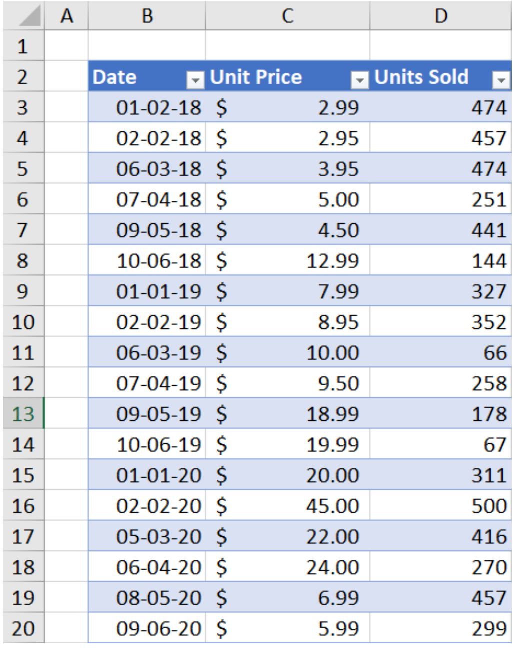
The resulting PivotTable would be as follows:



| Row Labels | Sales | Total Sales Example |
|---|---|---|
| 2018 | $16,136,530.08 | $21,489,979.62 |
| 2019 | $19,102,751.42 | $21,489,979.62 |
| 2020 | $21,489,979.62 | $21,489,979.62 |
| 2021 | $23,644,270.20 | $21,489,979.62 |
| **Grand Total** | **$80,373,531.32** | **$21,489,979.62** |

***Shortfalls***

The **SUM** function is great at aggregating or summing entire columns, however can it cope with summing the product of two columns at once? Consider the following data:



| Date | Unit Price | Units Sold |
|---|---|---|
| 01-02-18 | $ 2.99 | 474 |
| 02-02-18 | $ 2.95 | 457 |
| 06-03-18 | $ 3.95 | 474 |
| 07-04-18 | $ 5.00 | 251 |
| 09-05-18 | $ 4.50 | 441 |
| 10-06-18 | $ 12.99 | 144 |
| 01-01-19 | $ 7.99 | 327 |
| 02-02-19 | $ 8.95 | 352 |
| 06-03-19 | $ 10.00 | 66 |
| 07-04-19 | $ 9.50 | 258 |
| 09-05-19 | $ 18.99 | 178 |
| 10-06-19 | $ 19.99 | 67 |
| 01-01-20 | $ 20.00 | 311 |
| 02-02-20 | $ 45.00 | 500 |
| 05-03-20 | $ 22.00 | 416 |
| 06-04-20 | $ 24.00 | 270 |
| 08-05-20 | $ 6.99 | 457 |
| 09-06-20 | $ 5.99 | 299 |

Let's calculate the total revenue for our data. We will need to create three measures, **[Unit Price]**, **[Units Sold]** and **[Total Revenue]**. These can be created as follows:

**=SUM('Unit Price and Volume Sold'[Unit Price])**

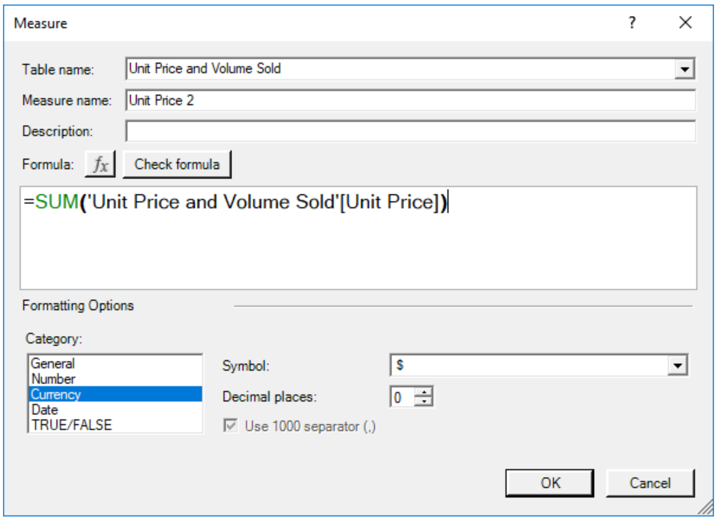


Table name: Unit Price and Volume Sold

Measure name: Unit Price 2

Description:

Formula:

```
=SUM('Unit Price and Volume Sold'[Unit Price])
```

Category: General, Number, Currency, Date, TRUE/FALSE (Currency selected)

Symbol: $

Decimal places: 0

☑ Use 1000 separator (,)

**=SUM('Unit Price and Volume Sold'[Units Sold])**

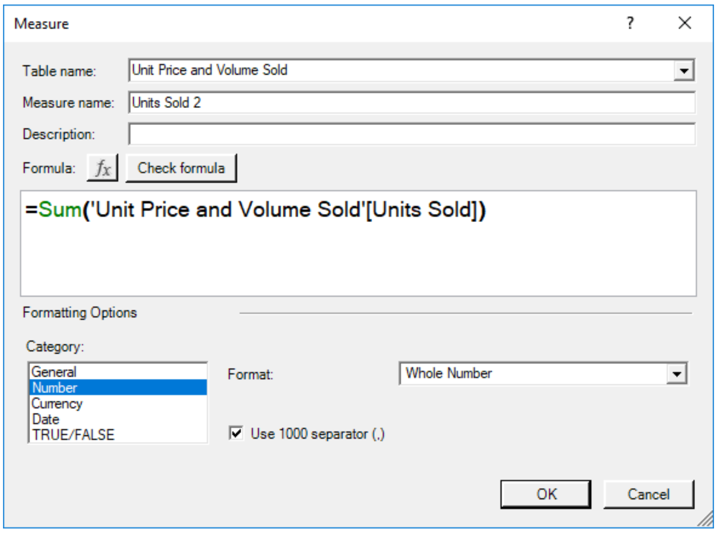


Table name: Unit Price and Volume Sold

Measure name: Units Sold 2

Description:

Formula:

```
=Sum('Unit Price and Volume Sold'[Units Sold])
```

Category: General, Number, Currency, Date, TRUE/FALSE (Number selected)

Format: Whole Number

☑ Use 1000 separator (,)

**=[Unit Price 2]*[Units Sold 2]**

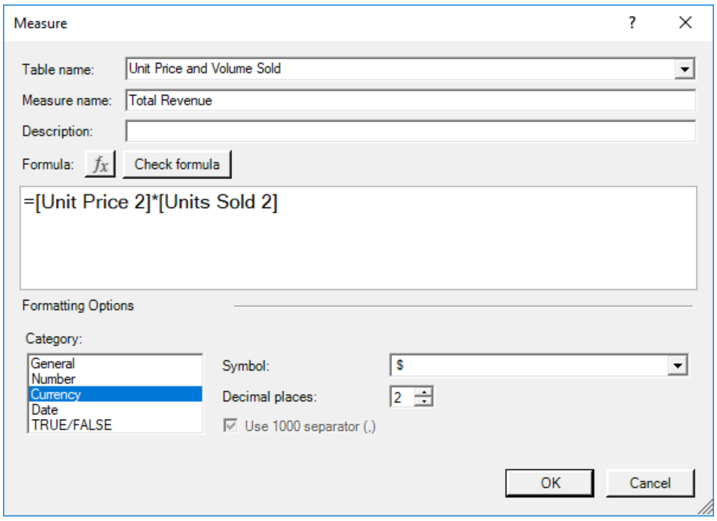


Table name: Unit Price and Volume Sold

Measure name: Total Revenue

Description:

Formula:

```
=[Unit Price 2]*[Units Sold 2]
```

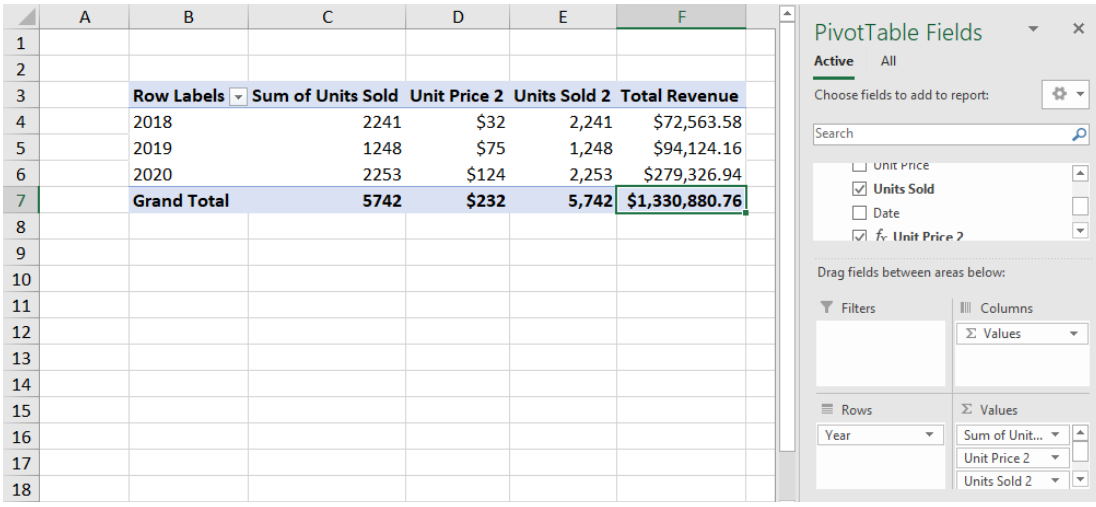
Category: General, Number, Currency, Date, TRUE/FALSE (Currency selected)

Symbol: $

Decimal places: 2

☑ Use 1000 separator (,)

The resulting PivotTable yields:



| Row Labels | Sum of Units Sold | Unit Price 2 | Units Sold 2 | Total Revenue |
|---|---|---|---|---|
| 2018 | 2241 | $32 | 2,241 | $72,563.58 |
| 2019 | 1248 | $75 | 1,248 | $94,124.16 |
| 2020 | 2253 | $124 | 2,253 | $279,326.94 |
| Grand Total | 5742 | $232 | 5,742 | $1,330,880.76 |

PivotTable Fields

Active All

Choose fields to add to report:

☐ Unit Price
☑ Units Sold
☐ Date
☑ fx Unit Price 2

Drag fields between areas below:

Filters

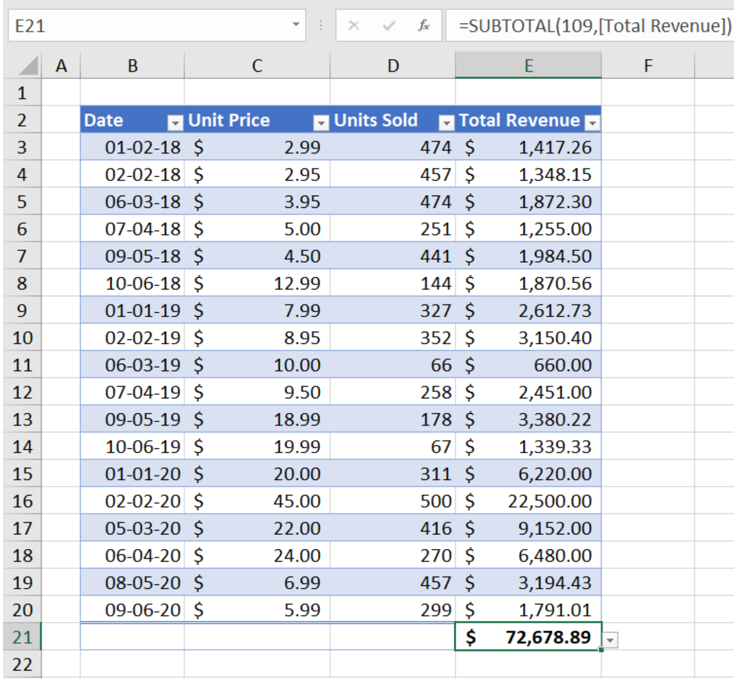
Columns: Σ Values

Rows: Year

Values: Sum of Unit... / Unit Price 2 / Units Sold 2

Strange, we didn't think we made that much money either…

A quick and dirty check in our original data table reveals that we only made $72,678.89:



E21 =SUBTOTAL(109,[Total Revenue])

| Date | Unit Price | Units Sold | Total Revenue |
|---|---|---|---|
| 01-02-18 | $ 2.99 | 474 | $ 1,417.26 |
| 02-02-18 | $ 2.95 | 457 | $ 1,348.15 |
| 06-03-18 | $ 3.95 | 474 | $ 1,872.30 |
| 07-04-18 | $ 5.00 | 251 | $ 1,255.00 |
| 09-05-18 | $ 4.50 | 441 | $ 1,984.50 |
| 10-06-18 | $ 12.99 | 144 | $ 1,870.56 |
| 01-01-19 | $ 7.99 | 327 | $ 2,612.73 |
| 02-02-19 | $ 8.95 | 352 | $ 3,150.40 |
| 06-03-19 | $ 10.00 | 66 | $ 660.00 |
| 07-04-19 | $ 9.50 | 258 | $ 2,451.00 |
| 09-05-19 | $ 18.99 | 178 | $ 3,380.22 |
| 10-06-19 | $ 19.99 | 67 | $ 1,339.33 |
| 01-01-20 | $ 20.00 | 311 | $ 6,220.00 |
| 02-02-20 | $ 45.00 | 500 | $ 22,500.00 |
| 05-03-20 | $ 22.00 | 416 | $ 9,152.00 |
| 06-04-20 | $ 24.00 | 270 | $ 6,480.00 |
| 08-05-20 | $ 6.99 | 457 | $ 3,194.43 |
| 09-06-20 | $ 5.99 | 299 | $ 1,791.01 |
| | | | **$ 72,678.89** |

Upon further investigation it appears that the two columns are being aggregated first, then multiplied with each other to produce the total revenue (*e.g.* $32 x 2,241 = $72,563.58). The actual total revenue for should be $9,747.77.

It appears that the **SUM** function is aggregating column values based on the row labels (years, grand total) and then multiplying the **Sum of # Units Sold** with the **Sum of Price Per Unit**. This approach will not correctly calculate the total revenue. We will need a function that iterates the calculation row by row, cue the **SUMX** function.

The **SUMX** function requires the following syntax to operate:

**SUMX(table, expression)**

We can use the **SUMX** function to create the following measure:

```
=SUMX(
        'Unit Price and Volume Sold',
        'Unit Price and Volume Sold'[# of Units Sold] * 'Unit Price and Volume Sold'[Price Per Unit]
        )
```

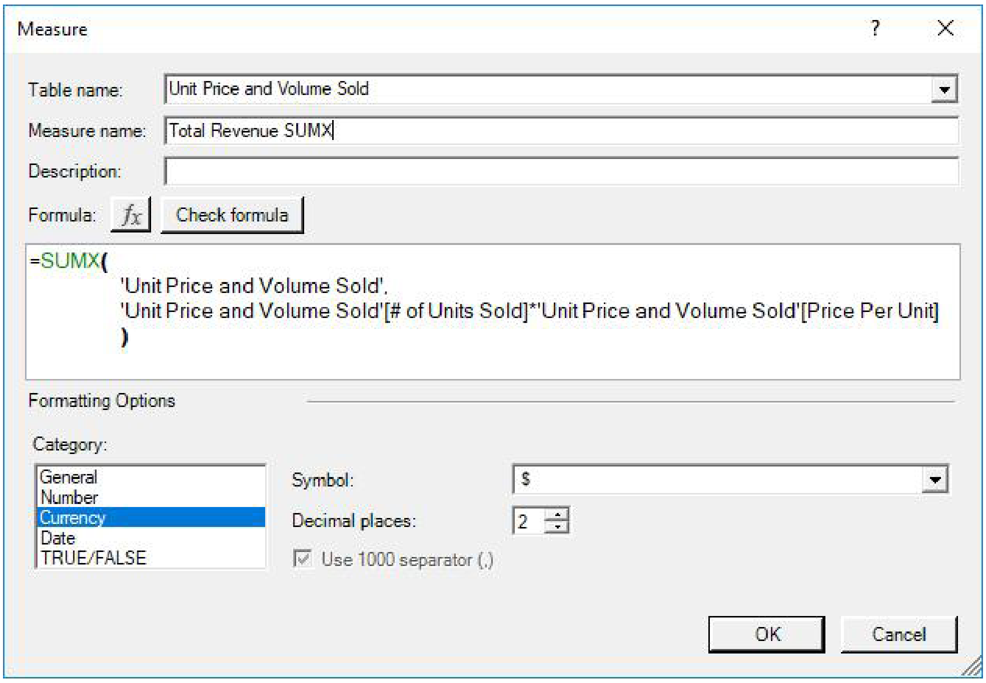


Table name: Unit Price and Volume Sold

Measure name: Total Revenue SUMX

Description:

```
=SUMX(
        'Unit Price and Volume Sold',
        'Unit Price and Volume Sold'[# of Units Sold]*'Unit Price and Volume Sold'[Price Per Unit]
        )
```

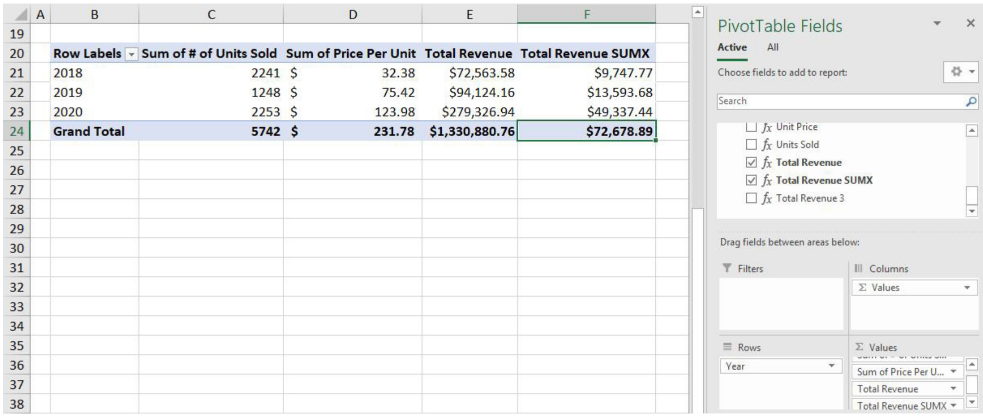
Category: General, Number, **Currency**, Date, TRUE/FALSE

Symbol: $

Decimal places: 2

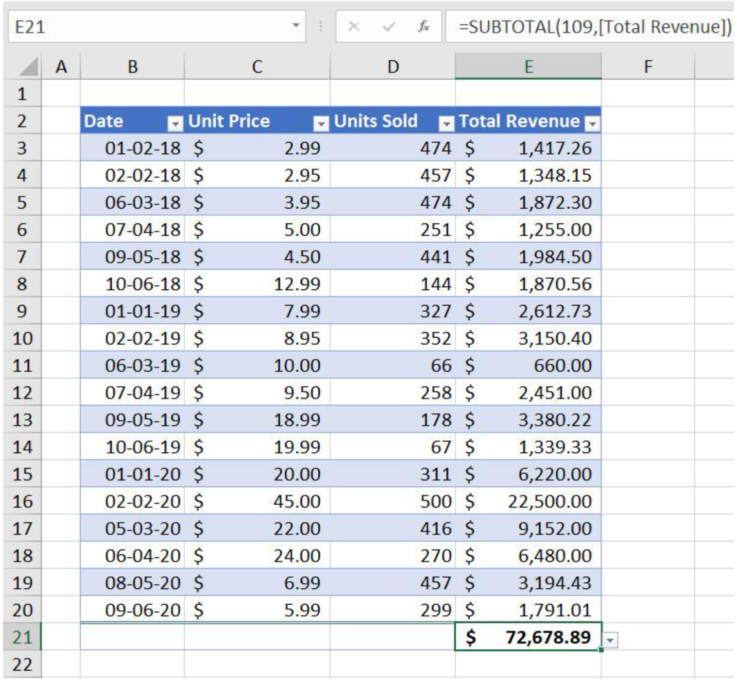
☑ Use 1000 separator (,)

The resulting PivotTable yields:



| Row Labels | Sum of # of Units Sold | Sum of Price Per Unit | Total Revenue | Total Revenue SUMX |
|---|---|---|---|---|
| 2018 | 2241 | $ 32.38 | $72,563.58 | $9,747.77 |
| 2019 | 1248 | $ 75.42 | $94,124.16 | $13,593.68 |
| 2020 | 2253 | $ 123.98 | $279,326.94 | $49,337.44 |
| **Grand Total** | **5742** | **$ 231.78** | **$1,330,880.76** | **$72,678.89** |

Just to check here’s our quick and dirty calculation:



E21 =SUBTOTAL(109,[Total Revenue])

| Date | Unit Price | Units Sold | Total Revenue |
|---|---|---|---|
| 01-02-18 | $ 2.99 | 474 | $ 1,417.26 |
| 02-02-18 | $ 2.95 | 457 | $ 1,348.15 |
| 06-03-18 | $ 3.95 | 474 | $ 1,872.30 |
| 07-04-18 | $ 5.00 | 251 | $ 1,255.00 |
| 09-05-18 | $ 4.50 | 441 | $ 1,984.50 |
| 10-06-18 | $ 12.99 | 144 | $ 1,870.56 |
| 01-01-19 | $ 7.99 | 327 | $ 2,612.73 |
| 02-02-19 | $ 8.95 | 352 | $ 3,150.40 |
| 06-03-19 | $ 10.00 | 66 | $ 660.00 |
| 07-04-19 | $ 9.50 | 258 | $ 2,451.00 |
| 09-05-19 | $ 18.99 | 178 | $ 3,380.22 |
| 10-06-19 | $ 19.99 | 67 | $ 1,339.33 |
| 01-01-20 | $ 20.00 | 311 | $ 6,220.00 |
| 02-02-20 | $ 45.00 | 500 | $ 22,500.00 |
| 05-03-20 | $ 22.00 | 416 | $ 9,152.00 |
| 06-04-20 | $ 24.00 | 270 | $ 6,480.00 |
| 08-05-20 | $ 6.99 | 457 | $ 3,194.43 |
| 09-06-20 | $ 5.99 | 299 | $ 1,791.01 |
| | | | **$ 72,678.89** |

We can see that the **SUMX** formula is correctly calculating the total revenue. This is because it is calculating the total revenue for each row item individually ($2.99 x 474 = $1,417.26, $2.95 x 457 = $1,348.15, *etc*.), unlike the **SUM** function. Therefore, we should use the **SUMX** function when we need Power Pivot to perform a calculation that needs to be iterated row by row.

For those of you thinking, why don't we just use a calculated column? Yes, we could, but we have to be wary of resource management: this may take up more memory than is required.

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 continue to look at some useful Boolean **Text** functions in **M**.*

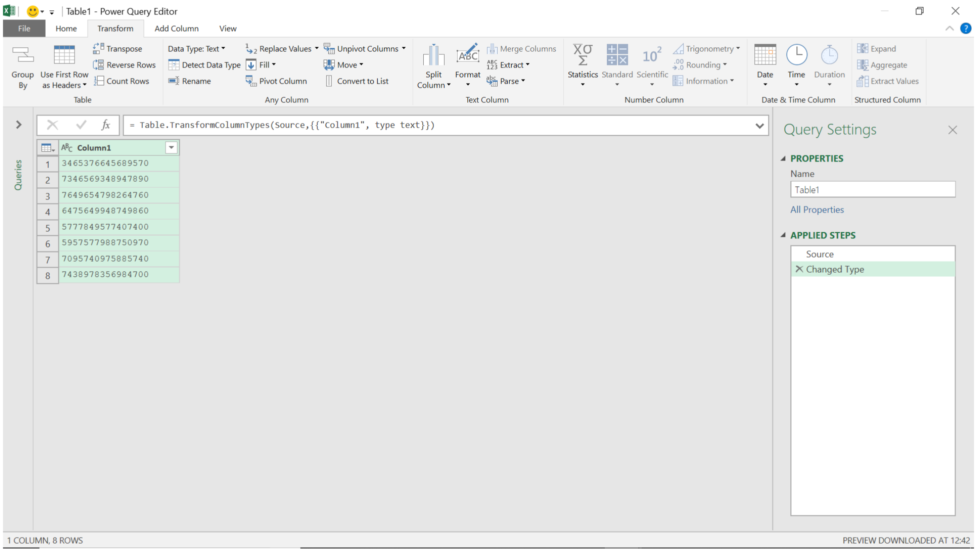
It's no secret that cleaning up data involves tidying up text strings so that we have consistent information to work with. There are a few **Text()** functions in **M** which are particularly useful. Having looked at true / false functions last month, this time I take a look at some **Text()** functions that can transform our data and we will give an example for each one.

### *Text.Insert*

**Text.Insert(text** as nullable text, **offset** as number, **newText** as text**)** as nullable text

This returns a text value with **newText** inserted into a **text** value starting at a zero-based offset.

This is a useful function for formatting a column to make it easier to read. In the next screenshot, we have a column of serial numbers. If different sections of the serial number have specific meanings (for example department or product type), then it makes sense to break down the number to make it easier to read. This also makes it clearer for other users who need to extract parts of the serial number to new columns.

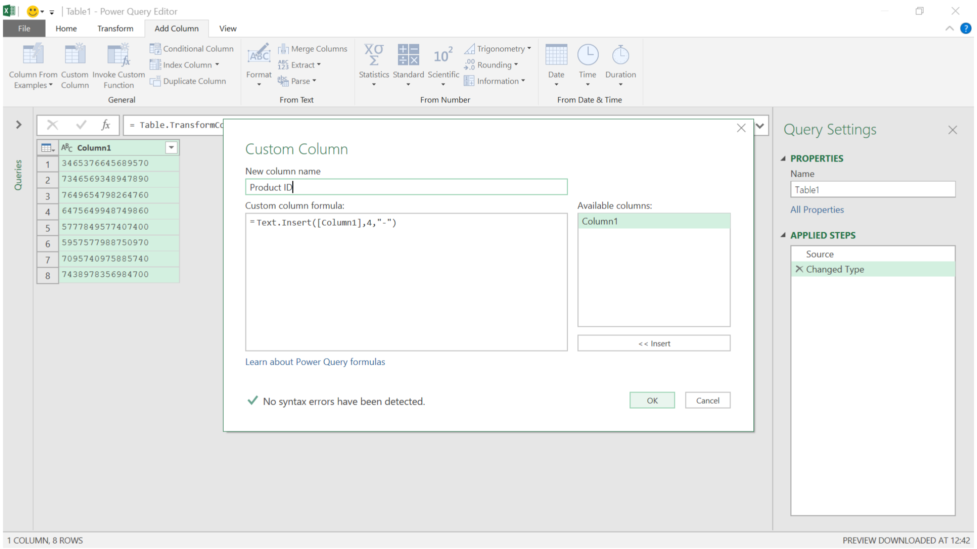


```
= Table.TransformColumnTypes(Source,{{"Column1", type text}})
```

| | Column1 |
|---|---|
| 1 | 3465376645689570 |
| 2 | 7346569348947890 |
| 3 | 7649654798264760 |
| 4 | 6475649948749860 |
| 5 | 5777849577407400 |
| 6 | 5957577988750970 |
| 7 | 7095740975885740 |
| 8 | 7438978356984700 |

Name: Table1

We may now use the **M Text.Insert()** formula to transform the column.



New column name: Product ID

Custom column formula:

```
= Text.Insert([Column1],4,"-")
```

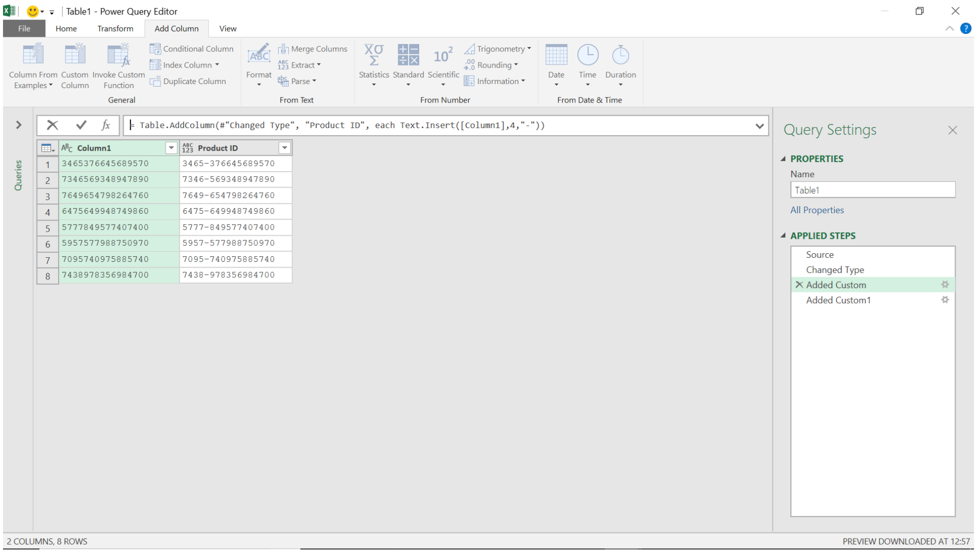
✓ No syntax errors have been detected.

| | Column1 |
|---|---|
| 1 | 3465376645689570 |
| 2 | 7346569348947890 |
| 3 | 7649654798264760 |
| 4 | 6475649948749860 |
| 5 | 5777849577407400 |
| 6 | 5957577988750970 |
| 7 | 7095740975885740 |
| 8 | 7438978356984700 |

Name: Table1

The formula used for our new column is

**= Text.Insert([Column1],4,"-")**



```
= Table.AddColumn(#"Changed Type", "Product ID", each Text.Insert([Column1],4,"-"))
```

| | Column1 | Product ID |
|---|---|---|
| 1 | 3465376645689570 | 3465-376645689570 |
| 2 | 7346569348947890 | 7346-569348947890 |
| 3 | 7649654798264760 | 7649-654798264760 |
| 4 | 6475649948749860 | 6475-649948749860 |
| 5 | 5777849577407400 | 5777-849577407400 |
| 6 | 5957577988750970 | 5957-577988750970 |
| 7 | 7095740975885740 | 7095-740975885740 |
| 8 | 7438978356984700 | 7438-978356984700 |

Query Settings

PROPERTIES

Name: Table1

All Properties

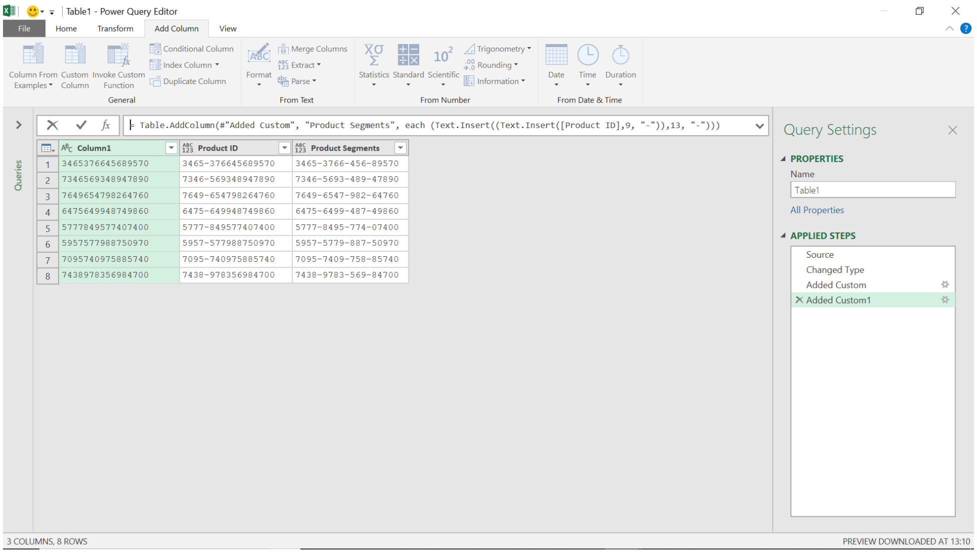
APPLIED STEPS

- Source
- Changed Type
- Added Custom
- Added Custom1

2 COLUMNS, 8 ROWS

PREVIEW DOWNLOADED AT 12:57

It is also possible to nest this function so that we insert multiple characters, as shown in the formula, to create column **Product Segments**.



```
= Table.AddColumn(#"Added Custom", "Product Segments", each (Text.Insert((Text.Insert([Product ID],9, "-")),13, "-")))
```

| | Column1 | Product ID | Product Segments |
|---|---|---|---|
| 1 | 3465376645689570 | 3465-376645689570 | 3465-3766-456-89570 |
| 2 | 7346569348947890 | 7346-569348947890 | 7346-5693-489-47890 |
| 3 | 7649654798264760 | 7649-654798264760 | 7649-6547-982-64760 |
| 4 | 6475649948749860 | 6475-649948749860 | 6475-6499-487-49860 |
| 5 | 5777849577407400 | 5777-849577407400 | 5777-8495-774-07400 |
| 6 | 5957577988750970 | 5957-577988750970 | 5957-5779-887-50970 |
| 7 | 7095740975885740 | 7095-740975885740 | 7095-7409-758-85740 |
| 8 | 7438978356984700 | 7438-978356984700 | 7438-9783-569-84700 |

Query Settings

PROPERTIES

Name: Table1

All Properties

APPLIED STEPS

- Source
- Changed Type
- Added Custom
- Added Custom1

3 COLUMNS, 8 ROWS

PREVIEW DOWNLOADED AT 13:10

In this case, the formula is given by

**= Text.Insert((Text.Insert([Product ID],9, "-")),13, "-"))**

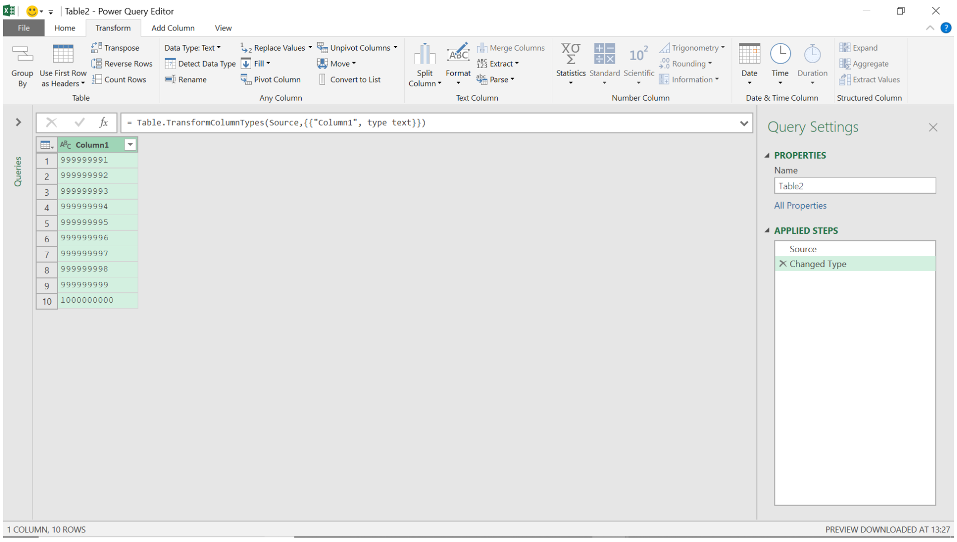
The **Product ID** has now been fully broken down into segments and is much easier to read and compare to other ID's.

### *Text.PadStart*

**Text.PadStart(text** as nullable text, **length** as number, **pad** as nullable text**)** as nullable text

This returns a **text** value padded at the start with **pad** to make it at least **length** characters.

This is a useful **M** function for presenting generated reference numbers correctly. Note that if no 'pad' character is specified, then the default is a space. In the next image, we have a list of generated reference numbers, but they all need to be 10 characters long.



```
= Table.TransformColumnTypes(Source,{{"Column1", type text}})
```

|  | Column1 |
|---|---|
| 1 | 999999991 |
| 2 | 999999992 |
| 3 | 999999993 |
| 4 | 999999994 |
| 5 | 999999995 |
| 6 | 999999996 |
| 7 | 999999997 |
| 8 | 999999998 |
| 9 | 999999999 |
| 10 | 1000000000 |

Name: Table2

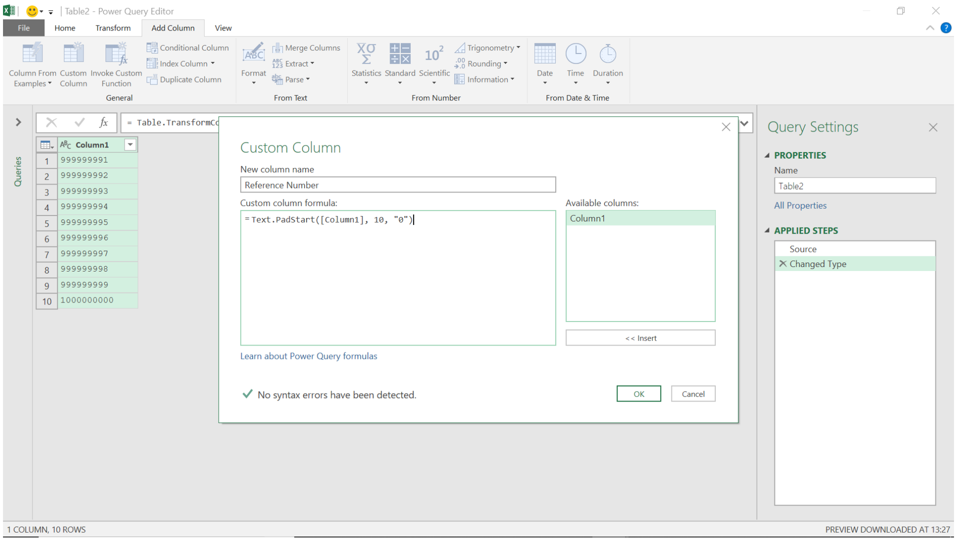
Applied steps: Source, Changed Type

1 COLUMN, 10 ROWS

PREVIEW DOWNLOADED AT 13:27

It’s important to make sure that Power Query hasn’t ‘helpfully’ converted the reference to a number – if it has, delete the automatically created ‘Changed Type’ step. In this case, we have converted the column to text.

We may now create a new column using **Text.PadStart()**.



New column name: Reference Number

Custom column formula:

```
= Text.PadStart([Column1], 10, "0")
```

Available columns: Column1

✓ No syntax errors have been detected.

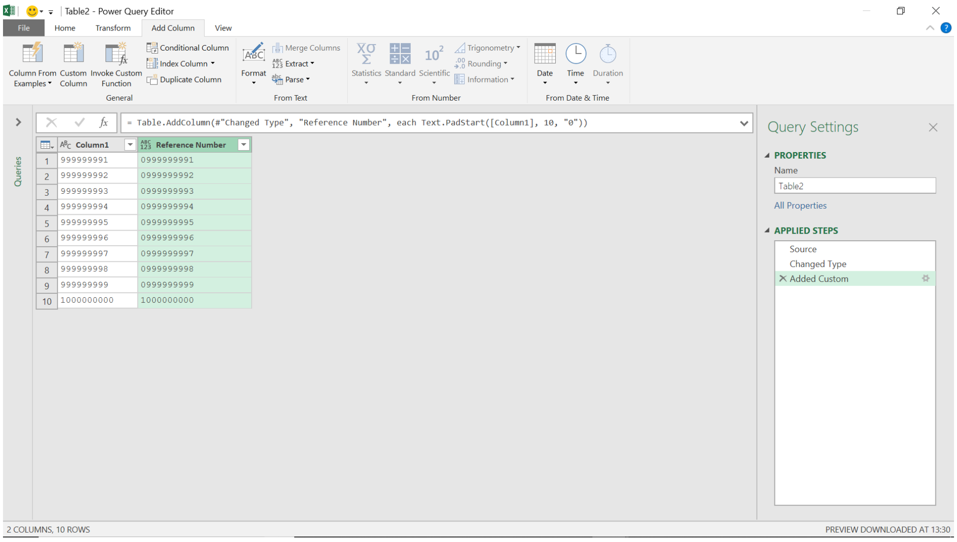
1 COLUMN, 10 ROWS

PREVIEW DOWNLOADED AT 13:27

The formula is

**= Text.PadStart([Column1],10,"0")**

We are expanding all our references to be 10 characters, by adding zeros at the beginning (zero filling).



```
= Table.AddColumn(#"Changed Type", "Reference Number", each Text.PadStart([Column1], 10, "0"))
```

|  | Column1 | Reference Number |
|---|---|---|
| 1 | 999999991 | 0999999991 |
| 2 | 999999992 | 0999999992 |
| 3 | 999999993 | 0999999993 |
| 4 | 999999994 | 0999999994 |
| 5 | 999999995 | 0999999995 |
| 6 | 999999996 | 0999999996 |
| 7 | 999999997 | 0999999997 |
| 8 | 999999998 | 0999999998 |
| 9 | 999999999 | 0999999999 |
| 10 | 1000000000 | 1000000000 |

Name: Table2

Applied steps: Source, Changed Type, Added Custom

2 COLUMNS, 10 ROWS

PREVIEW DOWNLOADED AT 13:30

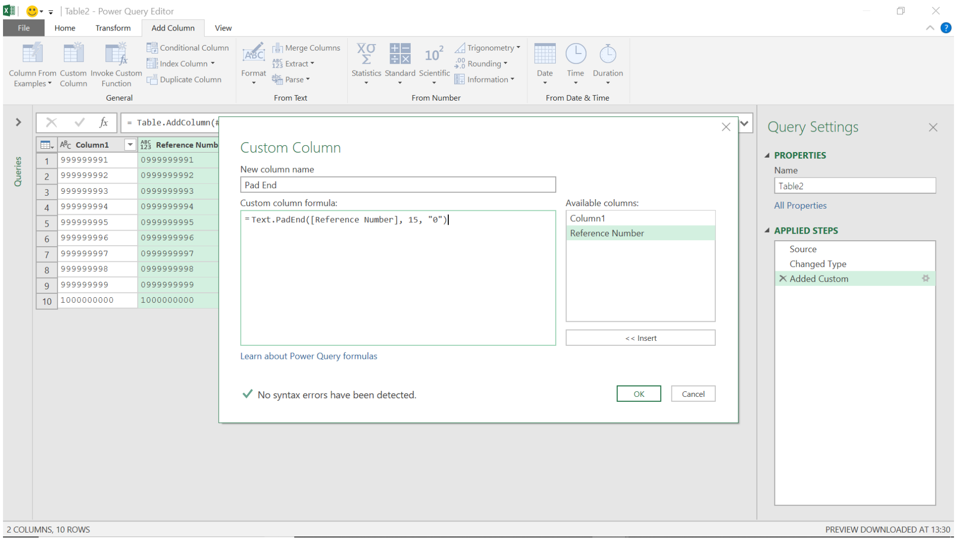
Our references are now presented correctly.

### *Text.PadEnd*

**Text.PadEnd(text** as nullable text, **length** as number, **pad** as nullable text**)** as nullable text

This returns a **text** value padded at the end with **pad** to make it at least **length** characters.

This is not an **M** function we use a great deal, but it's useful to know that text can be padded at either end. We will add zeros to the end of our reference number so that each number is 15 characters in length.



```
= Table.AddColumn(#
```

| | Column1 | Reference Numb |
|---|---|---|
| 1 | 999999991 | 0999999991 |
| 2 | 999999992 | 0999999992 |
| 3 | 999999993 | 0999999993 |
| 4 | 999999994 | 0999999994 |
| 5 | 999999995 | 0999999995 |
| 6 | 999999996 | 0999999996 |
| 7 | 999999997 | 0999999997 |
| 8 | 999999998 | 0999999998 |
| 9 | 999999999 | 0999999999 |
| 10 | 1000000000 | 1000000000 |

Custom Column

New column name: Pad End

Custom column formula:

```
= Text.PadEnd([Reference Number], 15, "0")
```

Available columns: Column1, Reference Number

✓ No syntax errors have been detected.

Name: Table2

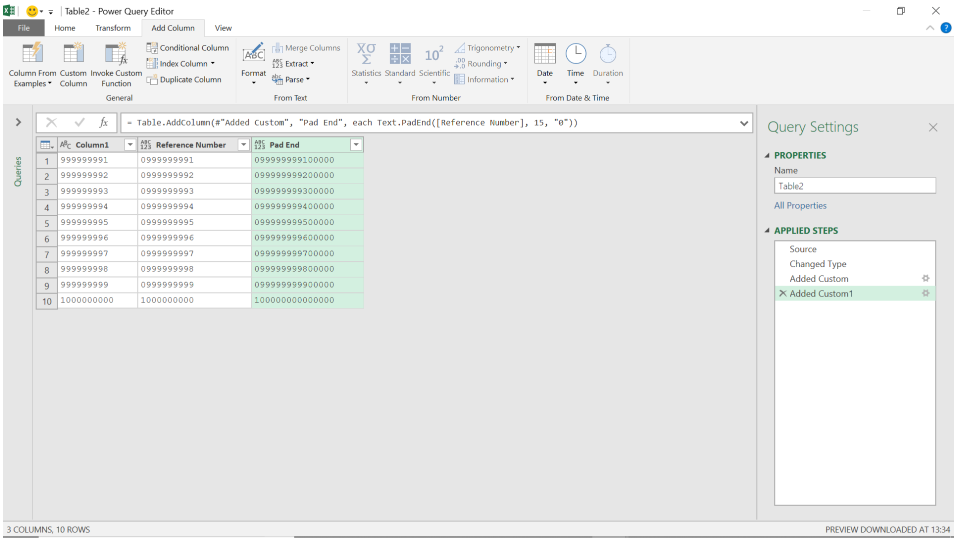
Applied steps: Source, Changed Type, Added Custom

2 COLUMNS, 10 ROWS

PREVIEW DOWNLOADED AT 13:30

The **M** formula used is

**= Text.PadEnd([Reference Number], 15, "0")**



```
= Table.AddColumn(#"Added Custom", "Pad End", each Text.PadEnd([Reference Number], 15, "0"))
```

| | Column1 | Reference Number | Pad End |
|---|---|---|---|
| 1 | 999999991 | 0999999991 | 099999999100000 |
| 2 | 999999992 | 0999999992 | 099999999200000 |
| 3 | 999999993 | 0999999993 | 099999999300000 |
| 4 | 999999994 | 0999999994 | 099999999400000 |
| 5 | 999999995 | 0999999995 | 099999999500000 |
| 6 | 999999996 | 0999999996 | 099999999600000 |
| 7 | 999999997 | 0999999997 | 099999999700000 |
| 8 | 999999998 | 0999999998 | 099999999800000 |
| 9 | 999999999 | 0999999999 | 099999999900000 |
| 10 | 1000000000 | 1000000000 | 100000000000000 |

Name: Table2

Applied steps: Source, Changed Type, Added Custom, Added Custom1

3 COLUMNS, 10 ROWS

PREVIEW DOWNLOADED AT 13:34

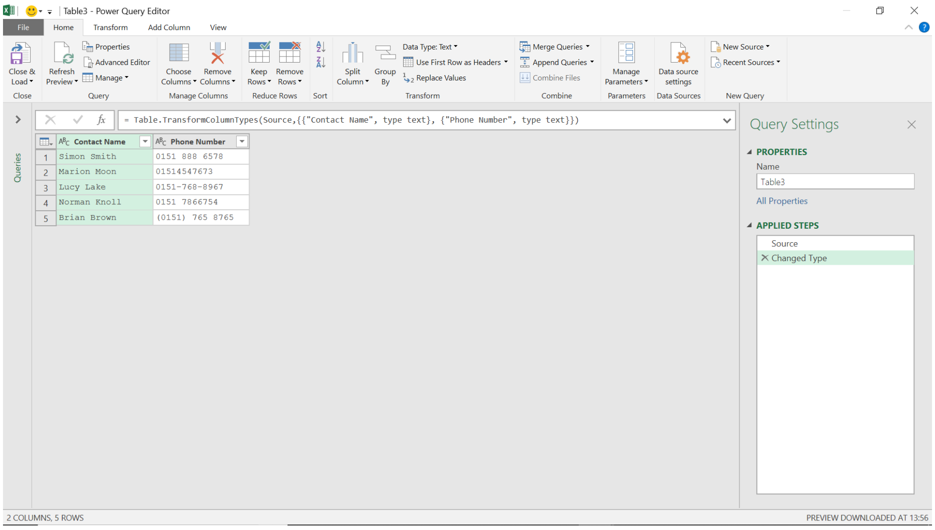
All the references are 15 characters long, and the zeros appear at the end of the references.

### *Text.Remove*

**Text.Remove(text** as nullable text, **removeChars** as any**)** as nullable text

This removes all occurrences of a character or list of characters from a **text** value. The **removeChars** parameter may be a character value or a list of character values.

This **M** function is very useful for getting rid of unwanted characters, so there is consistency. On the next image, we have a list of contacts where the phone numbers have not been recorded consistently:



```
= Table.TransformColumnTypes(Source,{{"Contact Name", type text}, {"Phone Number", type text}})
```

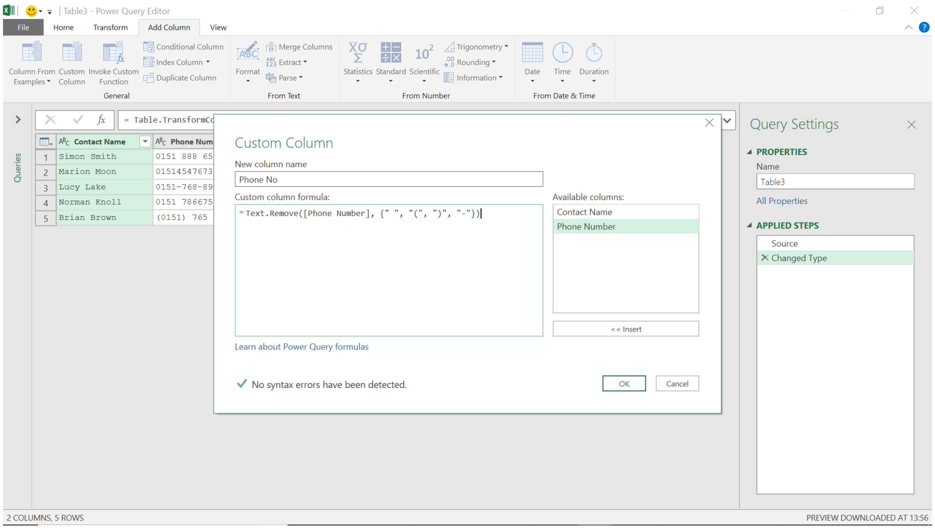
|   | Contact Name | Phone Number |
|---|---|---|
| 1 | Simon Smith | 0151 888 6578 |
| 2 | Marion Moon | 01514547673 |
| 3 | Lucy Lake | 0151-768-8967 |
| 4 | Norman Knoll | 0151 7866754 |
| 5 | Brian Brown | (0151) 765 8765 |

PROPERTIES
Name: Table3

APPLIED STEPS
- Source
- Changed Type

2 COLUMNS, 5 ROWS — PREVIEW DOWNLOADED AT 13:56

We need to remove spaces, brackets and hyphens so that the numbers are all in the same format.



New column name: Phone No

Custom column formula:

```
= Text.Remove([Phone Number], {" ", "(", ")", "-"})
```

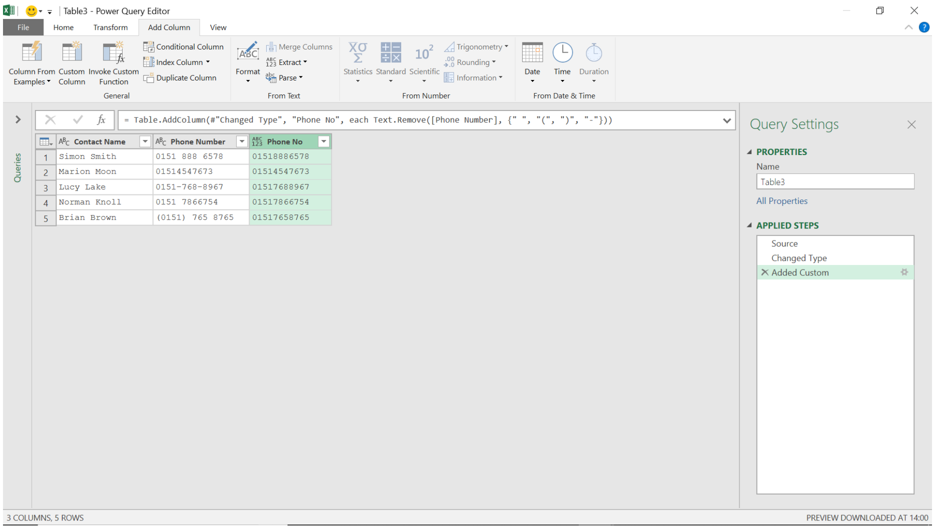
Available columns:
- Contact Name
- Phone Number

✓ No syntax errors have been detected.

The **M** formula used here is

**= Text.Remove([Phone Number], {" ", "(", ")", "-"})**

This should remove all spaces, brackets and hyphens.  We have specified a list by the use of the curly brackets or braces ({ }).



```
= Table.AddColumn(#"Changed Type", "Phone No", each Text.Remove([Phone Number], {" ", "(", ")", "-"}))
```

|   | Contact Name | Phone Number | Phone No |
|---|---|---|---|
| 1 | Simon Smith | 0151 888 6578 | 01518886578 |
| 2 | Marion Moon | 01514547673 | 01514547673 |
| 3 | Lucy Lake | 0151-768-8967 | 01517688967 |
| 4 | Norman Knoll | 0151 7866754 | 01517866754 |
| 5 | Brian Brown | (0151) 765 8765 | 01517658765 |

APPLIED STEPS
- Source
- Changed Type
- Added Custom

3 COLUMNS, 5 ROWS — PREVIEW DOWNLOADED AT 14:00

The phone numbers in the **Phone No** column are now displayed consistently.

Next month, we’ll look at some text functions that may be used to extract sections of a text string.

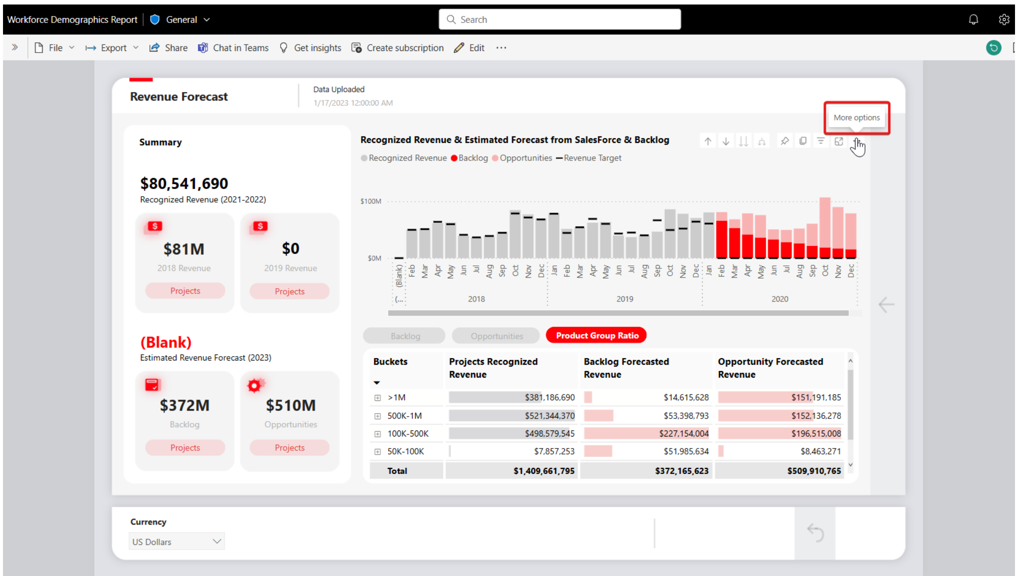
## Connected Excel Tables Now in Public Preview

This doesn’t appear to be part of the “formal” Power BI updates, but Microsoft has announced a new way to explore and analyse live Power BI data in Excel. Before now, you could only analyse live Power BI data in Excel using PivotTables (with the ‘Analyze in Excel’ feature) but with this feature, you can now use Excel tables to analyse live Power BI data too.

This is presently in Public Preview, with the capability being rolled out as we write. This new connected experience enables you to export refreshable data to Excel from a Power BI visual. The data loads into the spreadsheet grid so it’s easier to use for many users. The data is refreshable because the generated Excel workbook contains a live connection to Power BI, so you can refresh the data without leaving Excel, plus this option also allows you to export more data into Excel from Power BI.

Apparently, this new ‘Export to Excel’ feature is one of many planned new connected features aimed at improving user productivity by enabling you to self-serve data while keeping your data refreshed in Excel and reducing your reliance on *ad hoc* static data requests.

To use this feature, go to any report in Power BI Service (or https://www.powerbi.com), open ‘More options…’ on any Power BI visual and select ‘Export data’ from the menu. For example, in the following Workforce Demographics Report, let’s say you wish to analyse the ‘Recognized Revenue & Estimated Forecast from SalesForce & Backlog data’ *(sic)* in Excel:



Revenue Forecast

Data Uploaded
1/17/2023 12:00:00 AM

Summary

$80,541,690
Recognized Revenue (2021-2022)

$81M
2018 Revenue

$0
2019 Revenue

(Blank)
Estimated Revenue Forecast (2023)

$372M
Backlog

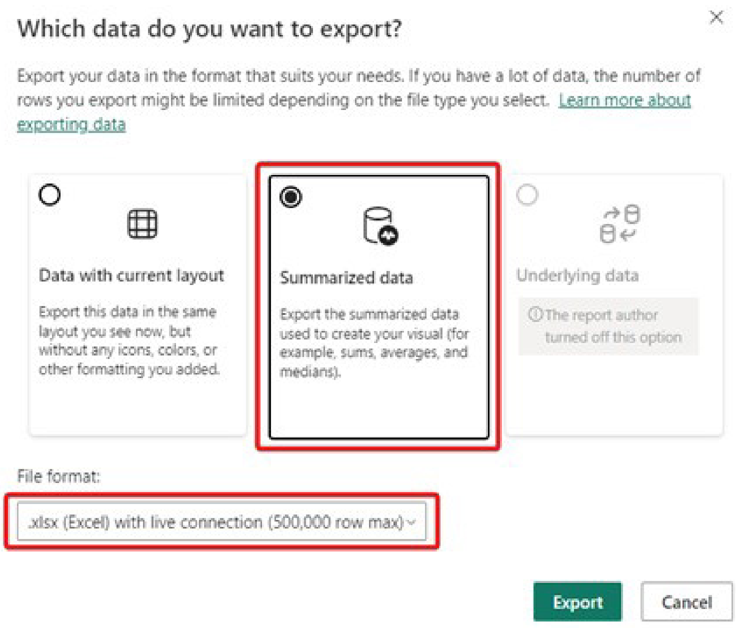
$510M
Opportunities

Recognized Revenue & Estimated Forecast from SalesForce & Backlog

| Buckets | Projects Recognized Revenue | Backlog Forecasted Revenue | Opportunity Forecasted Revenue |
|---|---|---|---|
| ⊞ >1M | $381,186,690 | $14,615,628 | $151,191,185 |
| ⊞ 500K-1M | $521,344,370 | $53,398,793 | $152,136,278 |
| ⊞ 100K-500K | $498,579,545 | $227,154,004 | $196,515,008 |
| ⊞ 50K-100K | $7,857,253 | $51,985,634 | $8,463,271 |
| **Total** | **$1,409,661,795** | **$372,165,623** | **$509,910,765** |

Currency: US Dollars

In the resulting dialog, select the ‘Summarized data’ card and you will see a new option under the ‘File format:’ dropdown menu, ‘.xlsx (Excel) with live connection’. Do note that in order to see the new .xlsx (Excel) with live connection option, you need to have Build permission for the underlying Power BI dataset.



Which data do you want to export?

Export your data in the format that suits your needs. If you have a lot of data, the number of rows you export might be limited depending on the file type you select. Learn more about exporting data

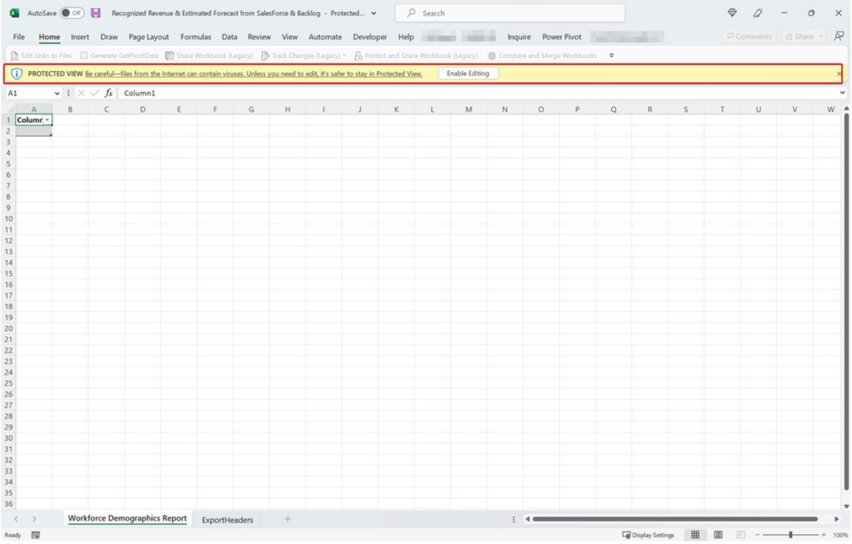
○ Data with current layout
Export this data in the same layout you see now, but without any icons, colors, or other formatting you added.

● Summarized data
Export the summarized data used to create your visual (for example, sums, averages, and medians).

○ Underlying data
ⓘ The report author turned off this option

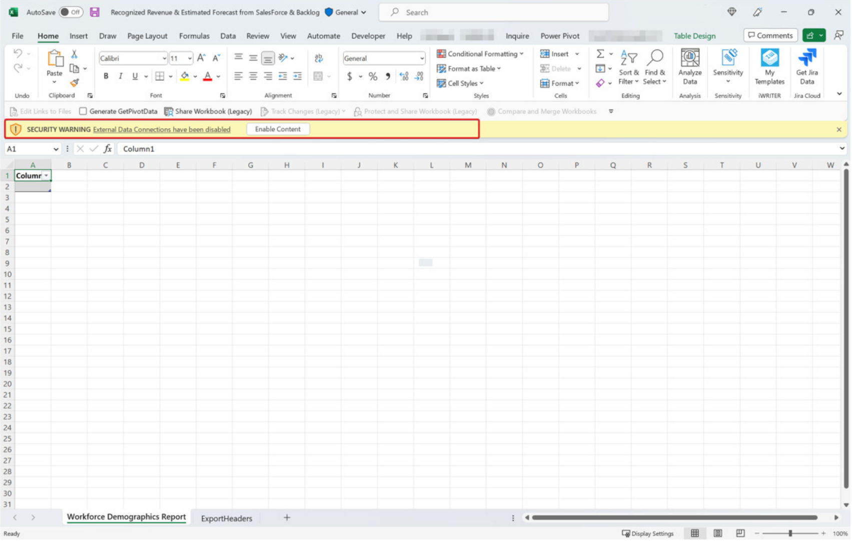
File format:
.xlsx (Excel) with live connection (500,000 row max)

After clicking Export, an Excel workbook containing the live Power BI data is downloaded to your computer. When you open the Excel workbook, it will be opened in read-only mode until you select the ‘Enable Editing’ button in the warning message.



PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

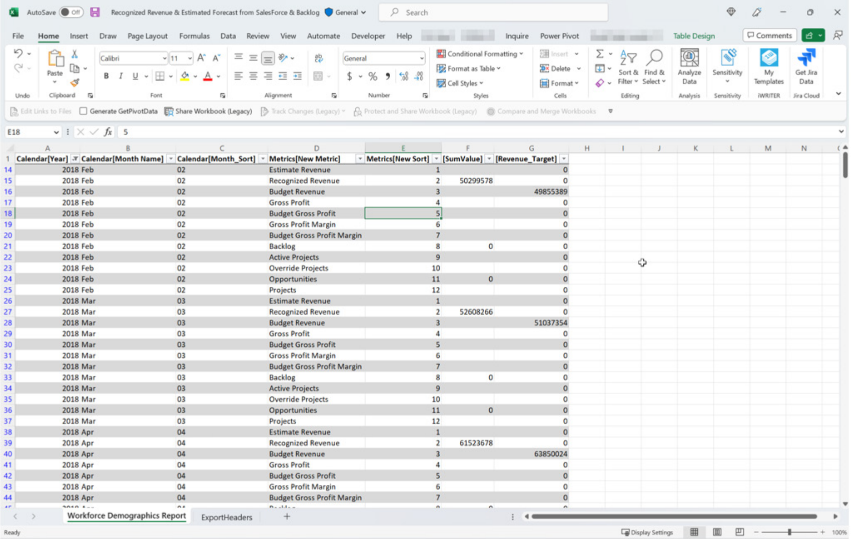
To load the data to the Excel grid, click 'Enable Content' and an Excel table is visible on the grid.



SECURITY WARNING External Data Connections have been disabled Enable Content

Once you click 'Enable Content' and the data is loaded to the Excel grid, anyone with whom the workbook is shared may view, but not refresh, the data. Before sharing the exported file with a colleague, you may need to open the file, pressing 'Enable Content', load the data into the Excel workbook and save the file. If you don't, the recipient will need to have Build permission on the underlying dataset to load the data when they open the file.

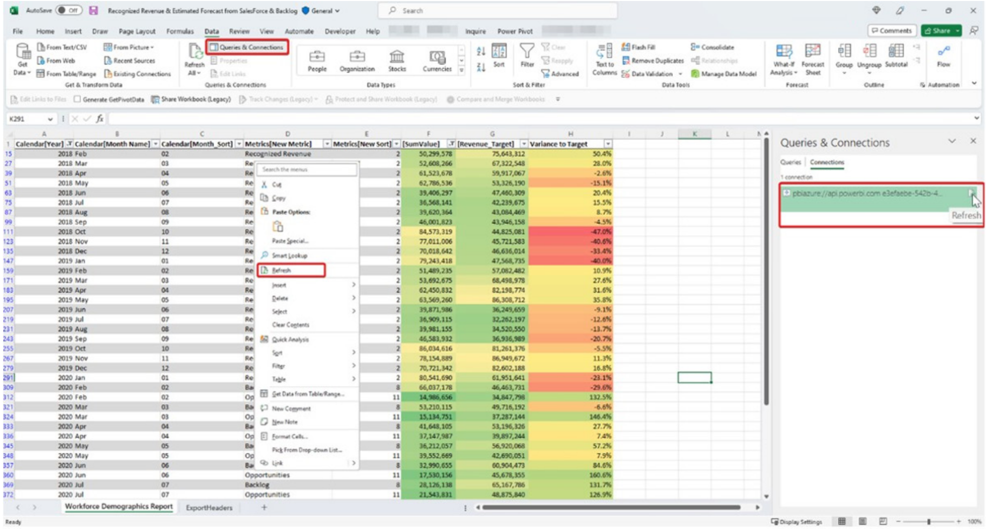
Once set up, you see the live Power BI data as an Excel Table and use your familiar Excel spreadsheet formulae to perform *ad hoc* analysis or apply formatting to the data.



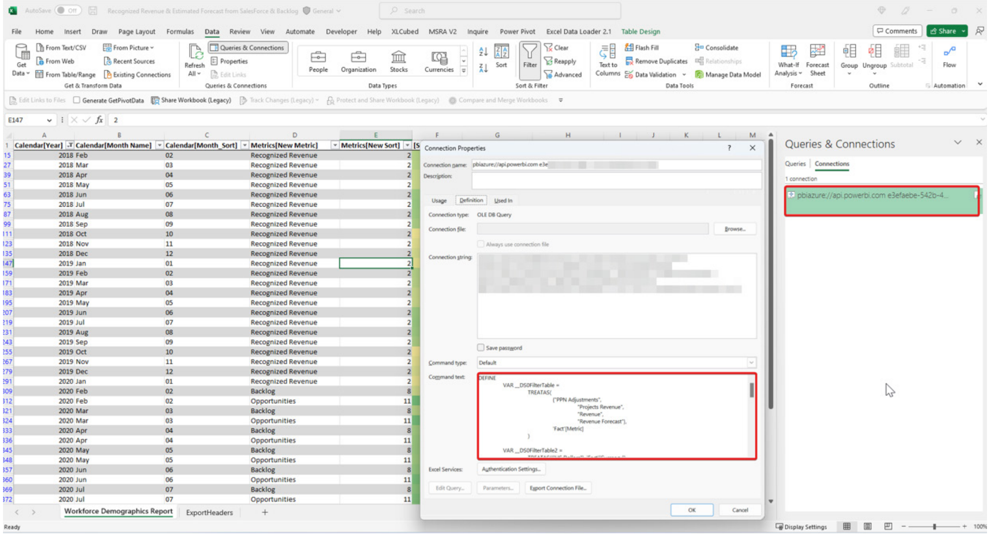
| 1 | Calendar[Year] | Calendar[Month Name] | Calendar[Month_Sort] | Metrics[New Metric] | Metrics[New Sort] | [SumValue] | [Revenue_Target] |
|---|---|---|---|---|---|---|---|
| 14 | 2018 | Feb | 02 | Estimate Revenue | 1 | | 0 |
| 15 | 2018 | Feb | 02 | Recognized Revenue | 2 | 50299578 | 0 |
| 16 | 2018 | Feb | 02 | Budget Revenue | 3 | | 49855389 |
| 17 | 2018 | Feb | 02 | Gross Profit | 4 | | 0 |
| 18 | 2018 | Feb | 02 | Budget Gross Profit | 5 | | 0 |
| 19 | 2018 | Feb | 02 | Gross Profit Margin | 6 | | 0 |
| 20 | 2018 | Feb | 02 | Budget Gross Profit Margin | 7 | | 0 |
| 21 | 2018 | Feb | 02 | Backlog | 8 | 0 | 0 |
| 22 | 2018 | Feb | 02 | Active Projects | 9 | | 0 |
| 23 | 2018 | Feb | 02 | Override Projects | 10 | | 0 |
| 24 | 2018 | Feb | 02 | Opportunities | 11 | 0 | 0 |
| 25 | 2018 | Feb | 02 | Projects | 12 | | 0 |
| 26 | 2018 | Mar | 03 | Estimate Revenue | 1 | | 0 |
| 27 | 2018 | Mar | 03 | Recognized Revenue | 2 | 52608266 | 0 |
| 28 | 2018 | Mar | 03 | Budget Revenue | 3 | | 51037354 |
| 29 | 2018 | Mar | 03 | Gross Profit | 4 | | 0 |
| 30 | 2018 | Mar | 03 | Budget Gross Profit | 5 | | 0 |
| 31 | 2018 | Mar | 03 | Gross Profit Margin | 6 | | 0 |
| 32 | 2018 | Mar | 03 | Budget Gross Profit Margin | 7 | | 0 |
| 33 | 2018 | Mar | 03 | Backlog | 8 | 0 | 0 |
| 34 | 2018 | Mar | 03 | Active Projects | 9 | | 0 |
| 35 | 2018 | Mar | 03 | Override Projects | 10 | | 0 |
| 36 | 2018 | Mar | 03 | Opportunities | 11 | 0 | 0 |
| 37 | 2018 | Mar | 03 | Projects | 12 | | 0 |
| 38 | 2018 | Apr | 04 | Estimate Revenue | 1 | | 0 |
| 39 | 2018 | Apr | 04 | Recognized Revenue | 2 | 61523678 | 0 |
| 40 | 2018 | Apr | 04 | Budget Revenue | 3 | | 63850024 |
| 41 | 2018 | Apr | 04 | Gross Profit | 4 | | 0 |
| 42 | 2018 | Apr | 04 | Budget Gross Profit | 5 | | 0 |
| 43 | 2018 | Apr | 04 | Gross Profit Margin | 6 | | 0 |
| 44 | 2018 | Apr | 04 | Budget Gross Profit Margin | 7 | | 0 |

It should be noted that the ExportHeaders tab shows the filters applied to the Power BI visual from which the data was exported from. This tab is always visible on any new workbook of data exported from a Power BI visual.

You can view the Power BI connection in the Excel workbook by clicking 'Queries & Connections' under the Data tab in Excel. The Power BI connection is visible in the right pane, and you can update your Power BI data by clicking Refresh in the pane or right-clicking on the Excel Table and selecting Refresh. Any formatting applied to the Excel Table is preserved after the data is refreshed.


Queries & Connections
Queries
Connections
1 connection
pbiazure://api.powerbi.com e3efaebe-542b-4...
Refresh
Workforce Demographics Report
ExportHeaders

If you are feeling a little more sophisticated, you can view the DAX statement behind the Excel Table under 'Connection Properties' in Excel:


Queries & Connections
Connections
pbiazure://api.powerbi.com e3efaebe-542b-4...
Workforce Demographics Report
ExportHeaders

Connection Properties

Connection name: pbiazure://api.powerbi.com e3e[redacted]

Description:

Usage | Definition | Used In

Connection type: OLE DB Query

Connection file: Browse...

□ Always use connection file

Connection string: [redacted]

□ Save password

Command type: Default

Command text:

```
DEFINE
        VAR __DS0FilterTable =
                TREATAS(
                        {"PPN Adjustments",
                                "Projects Revenue",
                                "Revenue",
                                "Revenue Forecast"},
                        'Fact'[Metric]
                )

        VAR __DS0FilterTable2 =
```

Excel Services: Authentication Settings...

Edit Query... Parameters... Export Connection File...

OK Cancel

The new .xlsx (Excel) with live connection option supports up to 500,000 rows of data. This is a substantial increase over the 150,000 rows supported in static export scenarios. According to Microsoft, increasing the number of rows in export has been one of the top requests regarding exporting. Furthermore, since the export is live and connected, export users can be more efficient because they don't need to recreate their analysis from scratch. Instead, they can save a copy, refresh their data, and immediately start their analysis.

The general requirements for using this feature are as follows:

- the 'Allow XMLA endpoints and Analyze in Excel with on-premises datasets' tenant setting has to be enabled
- you must have Build permissions to the Power BI dataset or have at least a Contributor role in a Power BI workspace
- you must have a Power BI license, such as Free, Pro or Premium Per User (PPU)
- this feature is available for use in both Excel Desktop and Excel for the web.

## Power BI Updates

The December update was released too late to make January's newsletter, but for completeness, we detail it here. This update saw a variety of new DAX functions, updated slicer type formatting and brought Metrics to the Windows application.

The full list is as follows:

***Reporting***

- Slicer type formatting moved to Format pane

***Modelling***

- Making it easier to do comparison calculations

***Data connectivity and preparation***

- Anaplan (connector update)
- Azure Databricks, Databricks (connector update)
- CData Connect Cloud (new connector)
- Cosmos DB V2 (new connector)
- Dremio Cloud (connector update)
- Google BigQuery (Azure AD) (new connector)

***Service***

- Power BI org app Multiple Audiences will be Generally Available later in 2023
- Announcing the deprecation of ‘Getting Started’ in the Expanded View of the Power BI Service
- Upcoming changes to the ‘Get Data’ experience in the Power BI Service

***Mobile***

- Track your Metrics on the Windows app

***Developers***

- Dynamically setting data chunk size

***Visualisations***

- New visuals in AppSource
- Update on Charticulator status.

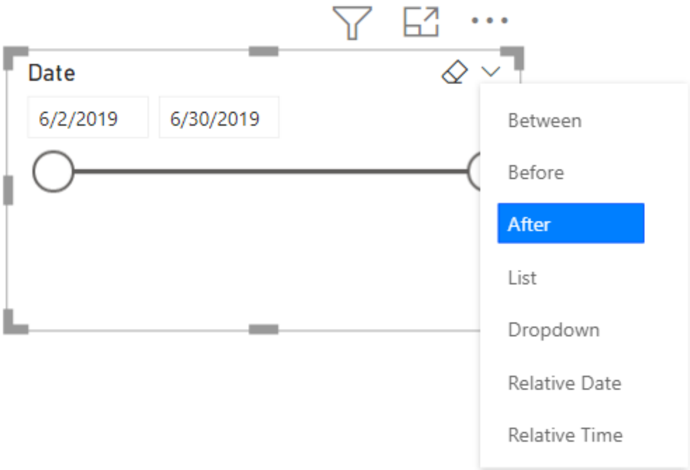
Let’s look at each in turn.

***Slicer type formatting moved to Format pane***

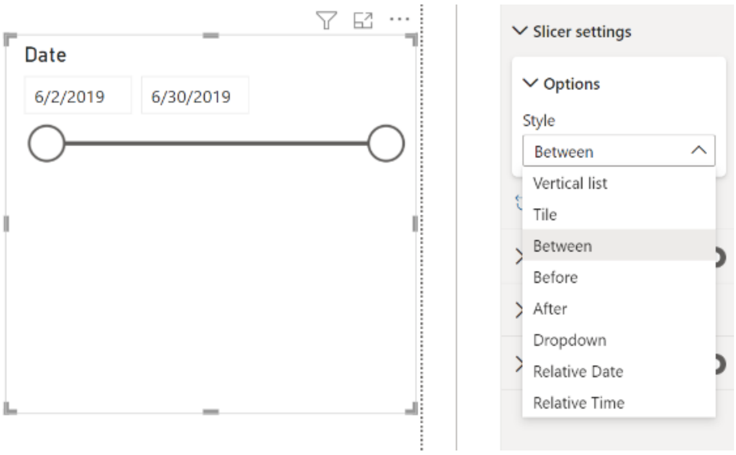
Previously, to change a slicer’s type, for example changing from relative date to a slider, these settings were only available in the visual header and only on hover. In addition, to change a slicer to ‘horizontal’ required users to first choose ‘list’ from the visual header to see the option in the Format pane and then use said pane to swap the orientation.

Now, these settings live in one place in the Format pane making it easier to discover and change between slicer types consistently. And for the record, do note that the aforementioned ‘horizontal’ has now been renamed to ‘tile’ based upon user feedback.

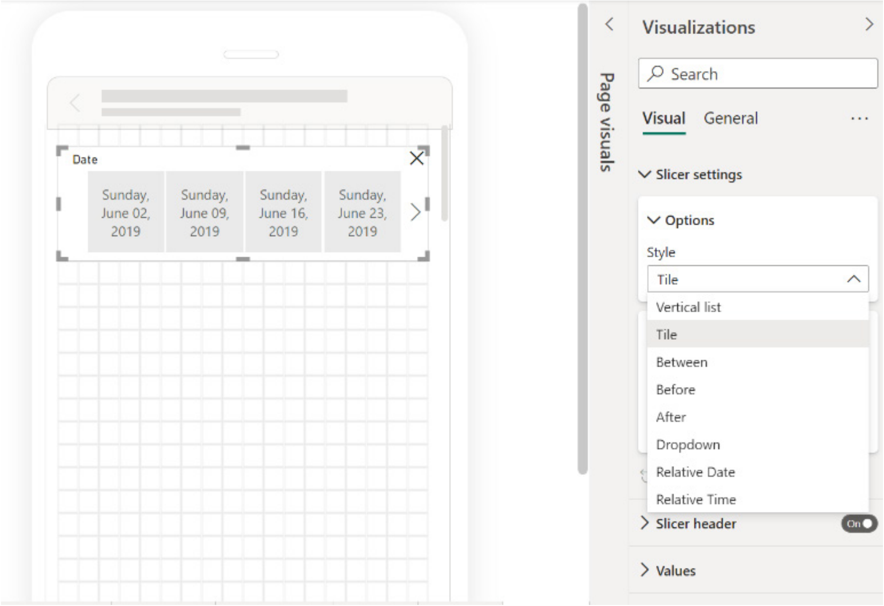
Previously:



Now:



Another added benefit with this change is the new Mobile formatting options now have access to this setting too. Users can quickly update their Mobile layout slicers to use tile to be more mobile friendly.



***Making it easier to do comparison calculations***

This update saw Microsoft introduce multiple new functions for DAX, targeted at making it easier to do comparison calculations in Power BI. The new functions were as follows:

- **INDEX** retrieves a result using absolute positioning
- **OFFSET** retrieves a result using relative positioning
- **WINDOW** retrieves a slice of results using absolute or relative positioning.

These functions also come with two helper functions called **ORDERBY** and **PARTITIONBY**. By our reckoning, that now takes the total number of DAX functions to 357.

These functions will make it easier to perform calculations such as:

- comparing values against a baseline or finding another specific entry (using **INDEX**)
- comparing values against a previous value (using **OFFSET**)
- adding a running total, moving average or similar calculations that rely on selecting a range of values (using **WINDOW**).

If you are familiar with the SQL language, you will note that these functions are very similar to SQL window functions. The functions released in this update perform a calculation across a set of table rows that are in one way or another related to the current row. These functions are different from SQL window functions, because of the DAX evaluation context concept, which will determine what is the "current row". Moreover, the functions introduced will not return a value but rather a set of rows which can be used together with **CALCULATE** or an aggregation function like **SUMX** (see elsewhere in this month's newsletter) to calculate a value.

Further, it should be noted that this group of functions is not pushed to the data source, but rather they are executed in the DAX engine. Additionally, Microsoft has stated it has seen much better performance using these functions compared to existing DAX expression to achieve the same result, especially when the calculation requires sorting by non-continuous columns.

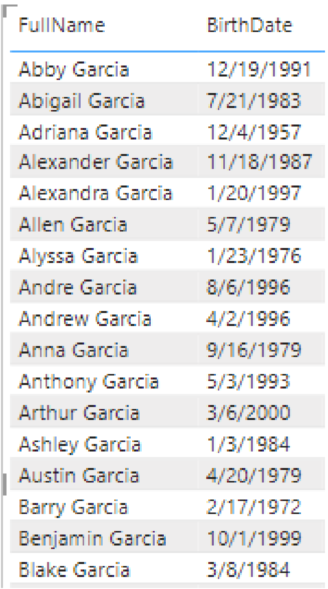
The DAX required to perform these calculations is simpler than the DAX required without them. However, while these new functions are very powerful and flexible, they still require a fair amount of complexity to make them work correctly. That is because Microsoft opted for high flexibility for these functions. However, the Powers that Be have stated that they recognise there is a need for easier to use functions that sacrifice some of the flexibility in favour of easier DAX. With this in mind, these functions now released should be seen as "a stepping stone, a building block if you will" towards Microsoft's goal to make DAX easier.

Let's take a look at them now.

**INDEX**

**INDEX** allows you to perform comparison calculations by retrieving a row that is in an absolute position. This will be most useful for comparing values against a certain baseline or another specific entry.

Consider the following example.  Below is a table of customer names and birth dates whose last name is "Garcia":

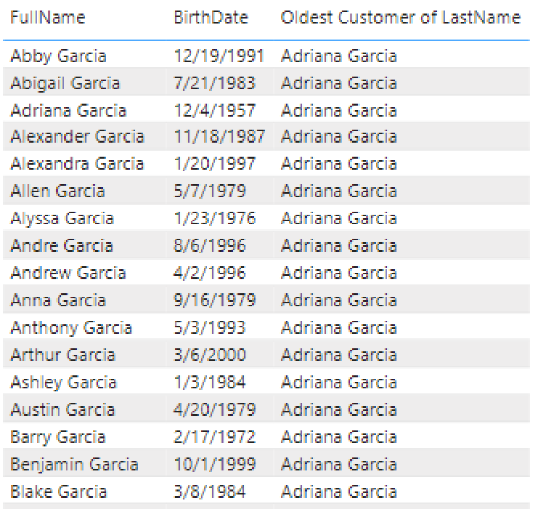


| FullName | BirthDate |
|---|---|
| Abby Garcia | 12/19/1991 |
| Abigail Garcia | 7/21/1983 |
| Adriana Garcia | 12/4/1957 |
| Alexander Garcia | 11/18/1987 |
| Alexandra Garcia | 1/20/1997 |
| Allen Garcia | 5/7/1979 |
| Alyssa Garcia | 1/23/1976 |
| Andre Garcia | 8/6/1996 |
| Andrew Garcia | 4/2/1996 |
| Anna Garcia | 9/16/1979 |
| Anthony Garcia | 5/3/1993 |
| Arthur Garcia | 3/6/2000 |
| Ashley Garcia | 1/3/1984 |
| Austin Garcia | 4/20/1979 |
| Barry Garcia | 2/17/1972 |
| Benjamin Garcia | 10/1/1999 |
| Blake Garcia | 3/8/1984 |

Now, imagine you wanted to find the oldest customer for each last name.  Therefore, for the last name "Garcia" that would be Adriana Garcia, born December 4th, 1957.  You can add the following calculated column on the **DimCustomer** table to achieve this goal and return the name:

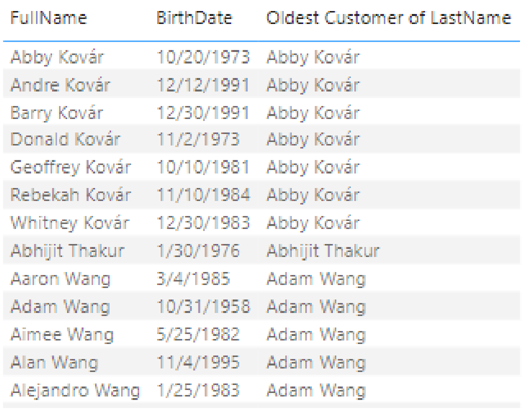
**Oldest Customer of LastName = SELECTCOLUMNS(INDEX(1, DimCustomer, ORDERBY([BirthDate]), PARTITIONBY([LastName])), [FullName])**

This would return the following result:



| FullName | BirthDate | Oldest Customer of LastName |
|---|---|---|
| Abby Garcia | 12/19/1991 | Adriana Garcia |
| Abigail Garcia | 7/21/1983 | Adriana Garcia |
| Adriana Garcia | 12/4/1957 | Adriana Garcia |
| Alexander Garcia | 11/18/1987 | Adriana Garcia |
| Alexandra Garcia | 1/20/1997 | Adriana Garcia |
| Allen Garcia | 5/7/1979 | Adriana Garcia |
| Alyssa Garcia | 1/23/1976 | Adriana Garcia |
| Andre Garcia | 8/6/1996 | Adriana Garcia |
| Andrew Garcia | 4/2/1996 | Adriana Garcia |
| Anna Garcia | 9/16/1979 | Adriana Garcia |
| Anthony Garcia | 5/3/1993 | Adriana Garcia |
| Arthur Garcia | 3/6/2000 | Adriana Garcia |
| Ashley Garcia | 1/3/1984 | Adriana Garcia |
| Austin Garcia | 4/20/1979 | Adriana Garcia |
| Barry Garcia | 2/17/1972 | Adriana Garcia |
| Benjamin Garcia | 10/1/1999 | Adriana Garcia |
| Blake Garcia | 3/8/1984 | Adriana Garcia |

In the example above, we showed only customers whose last name is "Garcia".  However, the same calculated column works on a set that has more than one last name:



| FullName | BirthDate | Oldest Customer of LastName |
|---|---|---|
| Abby Kovár | 10/20/1973 | Abby Kovár |
| Andre Kovár | 12/12/1991 | Abby Kovár |
| Barry Kovár | 12/30/1991 | Abby Kovár |
| Donald Kovár | 11/2/1973 | Abby Kovár |
| Geoffrey Kovár | 10/10/1981 | Abby Kovár |
| Rebekah Kovár | 11/10/1984 | Abby Kovár |
| Whitney Kovár | 12/30/1983 | Abby Kovár |
| Abhijit Thakur | 1/30/1976 | Abhijit Thakur |
| Aaron Wang | 3/4/1985 | Adam Wang |
| Adam Wang | 10/31/1958 | Adam Wang |
| Aimee Wang | 5/25/1982 | Adam Wang |
| Alan Wang | 11/4/1995 | Adam Wang |
| Alejandro Wang | 1/25/1983 | Adam Wang |

As you can see in the screenshots above, the full name of the oldest person with that last name is returned.  That's because we instructed **INDEX** to retrieve the first result when ordering by birth date, by specifying one [1].  By default, the ordering for the columns passed into **OrderBy** is ascending.  If we had specified two [2], we would have retrieved the name of the second oldest person with the last name instead, and so on.

Had we specified -1 or changed the sort order we would have returned the youngest person instead:

**Youngest Customer of LastName = SELECTCOLUMNS(INDEX(1, DimCustomer, ORDERBY([BirthDate], DESC), PARTITIONBY([LastName])), [FullName])**

is equivalent to:

**Youngest Customer of LastName = SELECTCOLUMNS(INDEX(-1, DimCustomer, ORDERBY([BirthDate]), PARTITIONBY([LastName])), [FullName])**

Notice that **INDEX** relies on two other new helper functions called **ORDERBY** and **PARTITIONBY**.

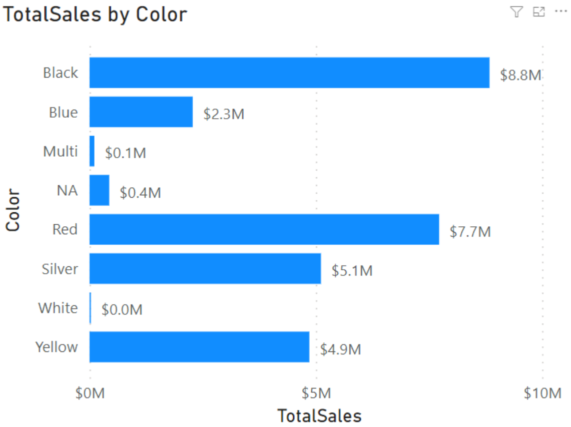
### ORDERBY and PARTITIONBY

- These helper functions may only be used in functions that accept an **orderBy** or **partitionBy** parameter, which are the functions introduced above:
- the **PARTITIONBY** function defines the columns that will be used to partition the rows on which these functions operate
- the **ORDERBY** function defines the columns that determine the sort order within each of a window function's partitions specified by **PARTITIONBY**.

### OFFSET

**OFFSET** allows you to perform comparison calculations more readily by retrieving a row that is in a relative position from your current position. This will be most useful for comparing across something else than time, for example across regions, cities or products. For date comparisons, for example, comparing the sales for this quarter against the same quarter last year there are dedicated Time Intelligence functions in DAX already. That doesn't mean you cannot use **OFFSET** to do the same, but it is not the immediate scenario.

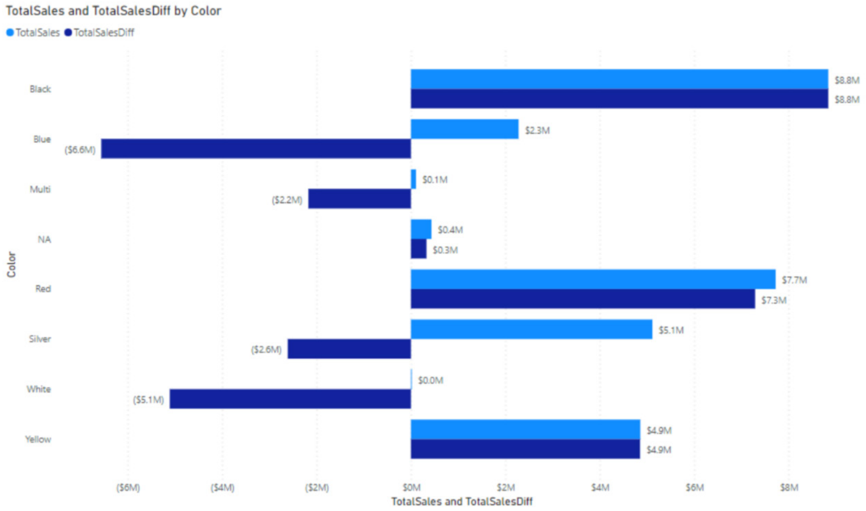
So what is the scenario for **OFFSET**? Consider the following. Here's a Bar chart that shows total sales by product colour:



Now, let's say you wanted to compare how well each colour is doing against the colour above it in the chart. You could write a complicated DAX statement for that, or you can now use **OFFSET** to accomplish this goal more simply, *viz.*

**TotalSalesDiff = IF(NOT ISBLANK([TotalSales]), [TotalSales] - CALCULATE([TotalSales], OFFSET(-1, FILTER(ALLSELECTED(DimProduct[Color]), NOT ISBLANK([TotalSales])))))**

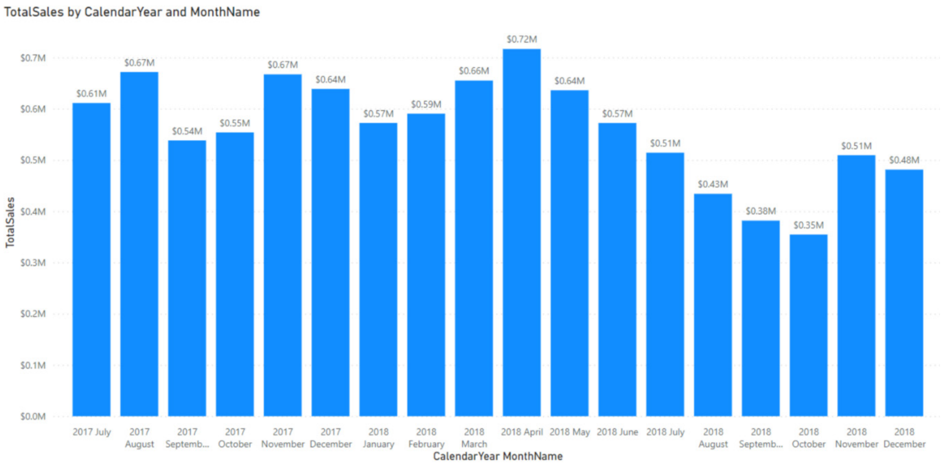
This will return the following result:



As you can see the newly added bars calculate the difference for each colour compared to the one just above it in the chart. That's because the DAX formula specified -1 for the first parameter to **OFFSET**. If we had specified -2 we would have made the comparison against the colour above each colour, but skipping the one right above it, so effectively the sales for the grey colour would have been compared against the sales for products that were black, *etc.*

**WINDOW**

**WINDOW** allows you to perform calculations that rely on ranges of results ("windows"), such as a moving average or a running sum. For example, the below Column chart shows total sales by year and month:

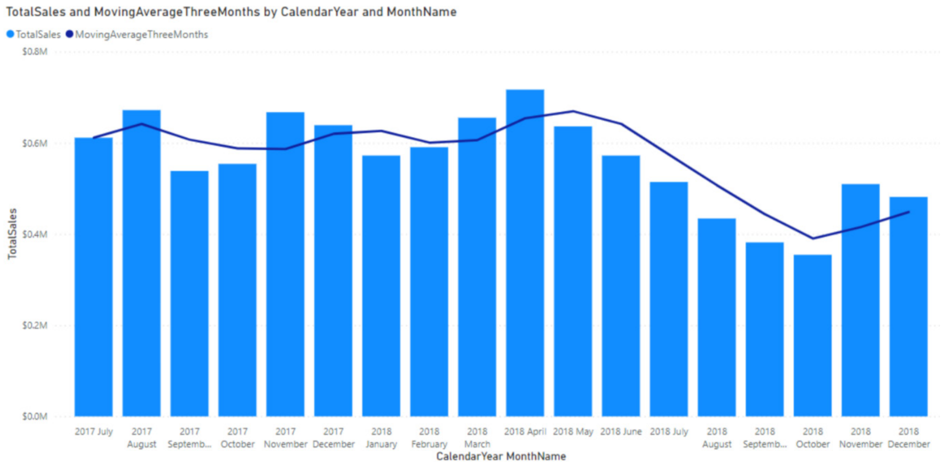


Now, let's say you wanted to add a moving average for the last three months of sales including the current. For example, for September 2017, you would expect the result to be the average sales of July, August and September in 2017 and for February 2018, we would expect the result to be the average sales for December 2017, January 2018 and February 2018.

To meet this requirement, you could write a complicated DAX statement or you can now use **WINDOW** to accomplish this goal using a simpler DAX statement:

**MovingAverageThreeMonths = AVERAGEX(WINDOW(-2, 0, ALLSELECTED(DimDate[CalendarYear], DimDate[MonthName], DimDate[MonthNumberOfYear]), ORDERBY(DimDate[CalendarYear], ASC, DimDate[MonthNumberOfYear], ASC)), [TotalSales])**

This will return the following result:



The newly added line correctly calculates the average sales over three months (including the current month). This release on a so-called 'relative window': the first parameter to **WINDOW** is set to -2, which means that the start of the range is set two months before to the current month (if that exists). The end of the range is inclusive and set to zero [0], which means the current month. Absolute windows are available as well, as both the start and end of the range can be defined in relative or absolute terms. Notice that **WINDOW** relies on two other new functions called **ORDERBY** and **PARTITIONBY** too.

***Anaplan (connector update)***

This version of Power BI connector for Anaplan includes backend changes for compatibility with ongoing Anaplan infrastructure updates. There is no change to user facing connector features.

***Azure Databricks, Databricks (connector update)***

The Azure Databricks and Databricks connectors now support native queries.

### *CData Connect Cloud (new connector)*

CData Connect Cloud brings real-time data access to hundreds of new cloud applications, databases and APIs from within Power BI.  CData Connect Cloud helps everyone can access the data they need.  Real-time data connectivity in the cloud means no installation, delays or complex data pipelines.  This allows you to take advantage of the most current data available to make real-time data driven decisions using Power BI.

### *Cosmos DB V2 (new connector)*

The new V2 connector will support querying the Cosmos DB transactional store in both DirectQuery and Import modes.  The DirectQuery mode will enable query pushdown, including aggregations to the Cosmos DB container when a filter on partition key is specified.

The DirectQuery mode in the V2 connector will be helpful in scenarios where Cosmos DB Container data is large and it is not feasible to import it all to Power BI cache in the Import mode.  It will also be helpful in user scenarios where real-time reporting with the latest Cosmos DB data is a requirement.  In addition to supporting DirectQuery mode, the V2 connector includes performance optimisations related to query pushdown and data serialisation.

It should also be noted that due to a known issue that is presently being fixed and deployed, support for this feature in Premium and end-to-end cloud refresh may not work until the very near future.

### *Dremio Cloud (connector update)*

This release contains a fix that allows query folding on data containing Decomal data types to be pushed down to Dremio.  This improves performance by reducing the volume of data Power BI is required to read.

### *Google BigQuery (Azure AD) (new connector)*

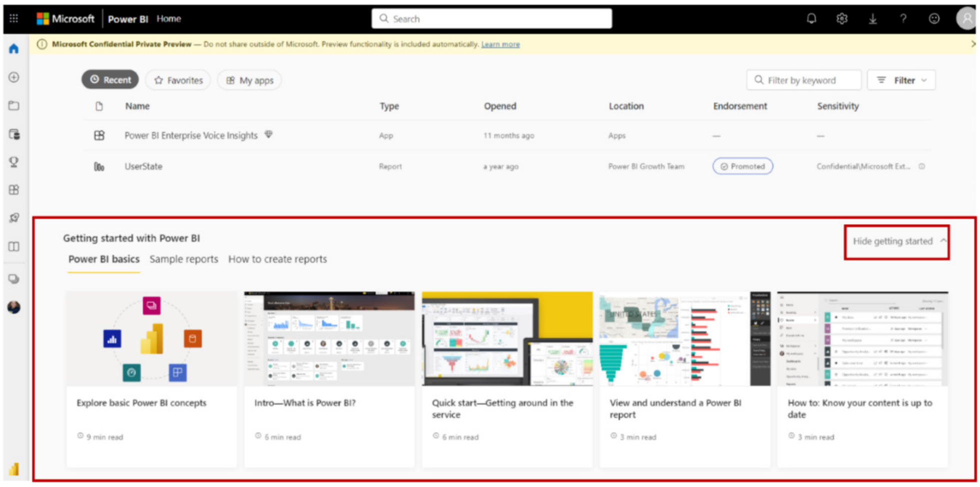
This update sees the release of a new Google BigQuery connector leveraging Azure Active Directory authentication.  Users can use Azure Active Directory-based Single Sign-On through Power BI Service and Gateway using this connector.

### *Power BI org app Multiple Audiences will be Generally Available later in 2023*

Following Microsoft's last announcement in August of launching the public Preview of Multiple Audiences for Power BI org apps, at the time of this update they further announced that this feature will be Generally Available early in 2023.  As stated, with Multiple Audiences for an app Power BI App author can create Multiple Audience groups within the same app and assign different permissions to each group.

### *Announcing the deprecation of 'Getting Started' in the Expanded View of the Power BI Service*

Since mid-December, the ability to view 'Getting Started' content through the expanded view from Power BI Service will be retired.



Microsoft Confidential Private Preview — Do not share outside of Microsoft. Preview functionality is included automatically. Learn more

| Name | Type | Opened | Location | Endorsement | Sensitivity |
|---|---|---|---|---|---|
| Power BI Enterprise Voice Insights | App | 11 months ago | Apps | — | — |
| UserState | Report | a year ago | Power BI Growth Team | Promoted | Confidential\Microsoft Ext... |

Getting started with Power BI

Explore basic Power BI concepts — 9 min read

Intro—What is Power BI? — 6 min read

Quick start—Getting around in the service — 6 min read

View and understand a Power BI report — 3 min read

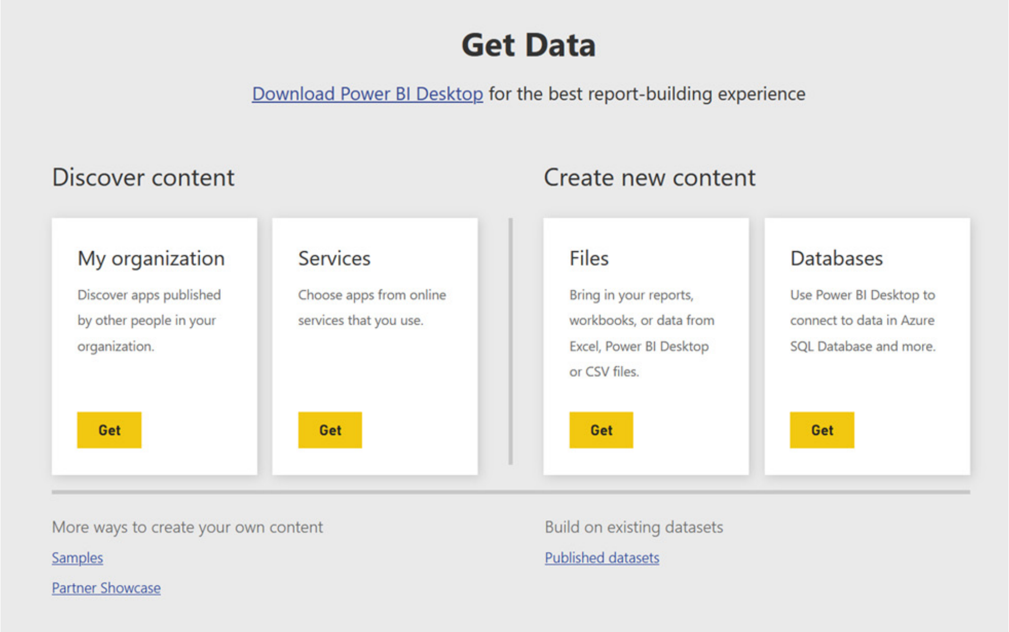
How to: Know your content is up to date — 3 min read

Power BI Service's expanded view currently gives you the option to view 'Getting Started' content at the end of your Home Page.  From the 'Getting Started' section, you can use information on how to get started using Power BI, tips and tricks on how to create / utilise reports and dashboards, *etc*.  Due to low usage from Power BI users, support for the 'Getting Started' section will be removed and you will no longer be able to access the content above through this area.  Retirement of the 'Getting Started' section will open Power BI Service's expanded view Homepage for new additions in the future.

Although you will not be able to view the 'Getting Started' content directly through Power BI Service, the content is still publicly available. If you are a new Power BI user, some of the content from the 'Getting Started' section will be available to you through your 'Recommended' carousel.  If you are not a new user, the 'Getting Started' content is still publicly available to you through Microsoft Learn.

***Upcoming changes to the 'Get Data' experience in the Power BI Service***

As mentioned in the last update, Microsoft is in the process of removing the older 'Get Data' page in the Power BI service in favour of new, comparable features available within workspaces. The change that will remove the entry points to the old 'Get Data' page, which is shown before reference, will be slowly rolling out early in 2023.



Get Data

Download Power BI Desktop for the best report-building experience

Discover content

My organization

Discover apps published by other people in your organization.

Get

Services

Choose apps from online services that you use.

Get

Create new content

Files

Bring in your reports, workbooks, or data from Excel, Power BI Desktop or CSV files.

Get

Databases

Use Power BI Desktop to connect to data in Azure SQL Database and more.

Get

More ways to create your own content

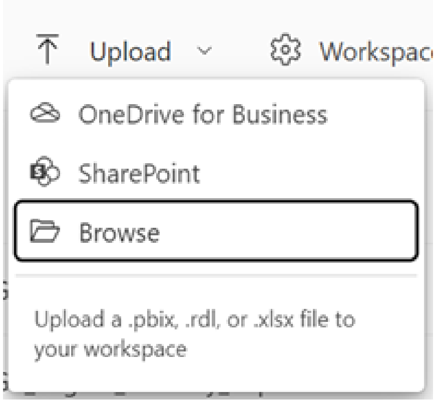
Samples

Partner Showcase

Build on existing datasets

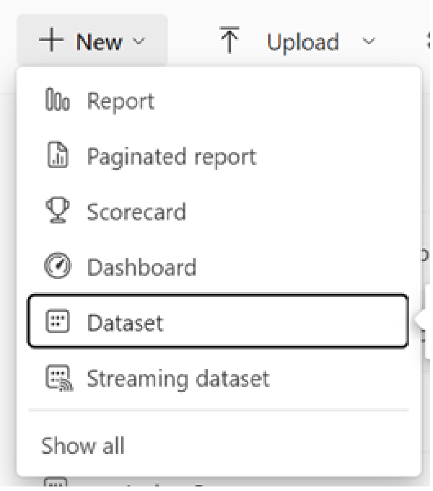
Published datasets

Going forward, you'll be able to access comparable features within workspaces. If you want to upload a file to Power BI, such as a .pbix, .xlsx, or .rdl files to your workspace, you can use the Upload option that was released in November. This option lets you upload files from your local computer or connect to files on OneDrive or a SharePoint site. With this change, you'll no longer be able to connect to files on personal OneDrive accounts.

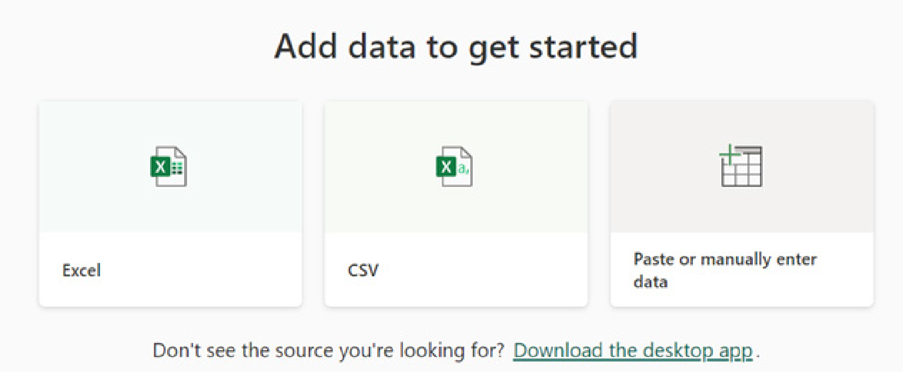


Upload a .pbix, .rdl, or .xlsx file to your workspace

If instead you want to create a dataset from Excel or CSV data, you can now access that functionality through the **New -> Dataset** option in the workspace you want to create the dataset in.



At the same time as the older 'Get Data' experience is removed, Power BI will also be updating this Dataset option to take you to a new page with options to create a dataset off an Excel, CSV or pasting in data. Once you select the file, the behaviour used to generate the dataset is the same as previously used on the 'Get Data' page. Once the dataset is created, you'll be taken to the dataset's details page in the Data hub.



Add data to get started

Excel

CSV

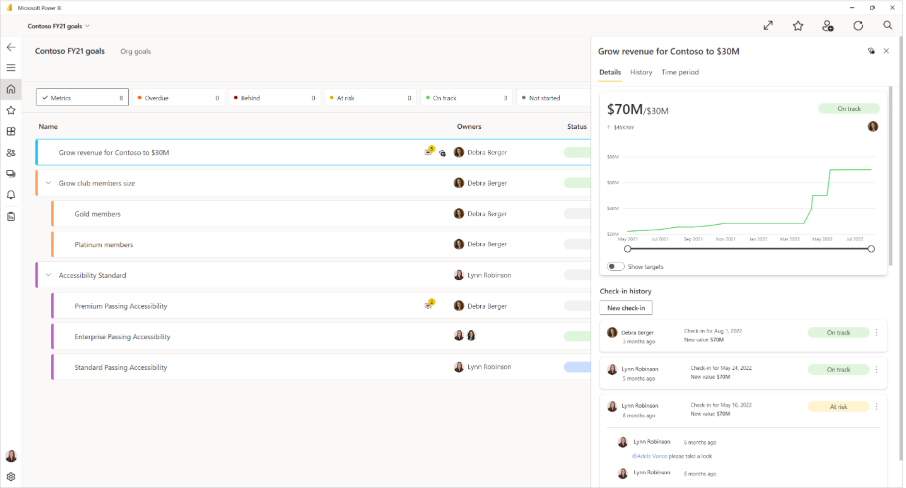
Paste or manually enter data

Don't see the source you're looking for? Download the desktop app.

These changes will be rolling out slowly, so you may not see the update immediately.

### *Track your Metrics on the Windows app*

Now you can monitor your Power BI metrics and scorecards and even check in progress, directly from the Windows application.



Microsoft Power BI

Contoso FY21 goals

Contoso FY21 goals    Org goals

✓ Metrics 8 · ● Overdue 0 · ● Behind 0 · ● At risk 0 · ● On track 3 · ● Not started

| Name | Owners | Status |
|---|---|---|
| Grow revenue for Contoso to $30M | Debra Berger | |
| Grow club members size | Debra Berger | |
| Gold members | Debra Berger | |
| Platinum members | Debra Berger | |
| Accessibility Standard | Lynn Robinson | |
| Premium Passing Accessibility | Debra Berger | |
| Enterprise Passing Accessibility | | |
| Standard Passing Accessibility | Lynn Robinson | |

Grow revenue for Contoso to $30M

Details    History    Time period

$70M/$30M    On track

↑ $45KYoY

Show targets

Check-in history

New check-in

Debra Berger, 3 months ago — Check-in for Aug 1, 2022, New value $70M — On track

Lynn Robinson, 5 months ago — Check-in for May 24, 2022, New value $70M — On track

Lynn Robinson, 6 months ago — Check-in for May 16, 2022, New value $70M — At risk

Lynn Robinson, 6 months ago: @Adele Vance please take a look

Lynn Robinson, 6 months ago

### *Dynamically setting data chunk size*

This update has enhanced the 'fetch more data' API to allow report authors to set the data chunk size dynamically by using the new property **dataReductionCustomization**. This is available with the new 5.2 API release.

### *New visuals in AppSource*

The following are new visuals within this update:

- 100% Clustered Stacked Bar Chart (Standard)
- 100% Clustered Stacked Column Chart (Standard)
- 3DBI
- Advanced Line Chart (Standard)
- Aimplan Planning and Reporting Visual
- Bubble Chart with Categorical Data (Standard)
- Definitive Logic Advanced Gantt Chart
- Dual X-axis Bar Chart (Standard)
- Dual X-Axis Combo Chart (Standard)
- Dual Y-Axis Column Chart (Standard)
- Dual Y-Axis Combo Chart (Standard)
- Excalibur
- GANTT by Lingaro
- Horizontal Bullet Chart (Standard)
- kpi emoji
- Likert Scale (Standard)
- Lollipop Bar Chart (Standard)
- Lollipop Column Chart (Standard)
- Merged Bar Chart (Standard)
- Multiple Vertical Line Chart (Standard)
- OneTax-PowerBIvisual
- Overlapping Bar (Standard)
- Overlapping Column (Standard)
- Pie Chart with Full Legend Label (Standard)
- Population Pyramid (Standard)
- swColorMap_twoLevels
- Vertical Bullet Chart (Standard)
- verticalText by sio2Graphs.

*Update on Charticulator status*

Finally, Microsoft has provided a status update on the Charticulator custom visual and the visual creation tool it's built upon. The Microsoft Research team which built the tool has moved on to new projects and ownership of the technology is being transitioned to a team in Power BI. Microsoft will continue to fix major bugs and actively maintain the quality of the current tool, although feature-level development has been put on hold for now.

That's it for this month: more anon.

## New Features for Excel

This release sees the announcement of 'Formula Suggestions' and 'Formula by Example' for Excel web users – a couple of exciting capabilities designed to help save you time and learn more about Excel formulae as you use them. Also for web users are suggested links, the **IMAGE** function and a new Search bar in the Queries pane. For Windows users, a new keyboard shortcut is available to open the Power Query editor (wow), and Insiders users on Windows can now get data from dynamic arrays and create nested Power Query data types to better organise data.

The full list is as follows:

*Excel for the web*

- Formula Suggestions
- Formula by Example
- Suggested Links
- Add Search bar in Queries pane
- **IMAGE** function

*Excel for Windows*

- Add keyboard shortcut to open the Power Query editor
- Create nested Power Query data types (Insiders)
- Add Get Data from Dynamic Arrays (Insiders)
- **IMAGE** function

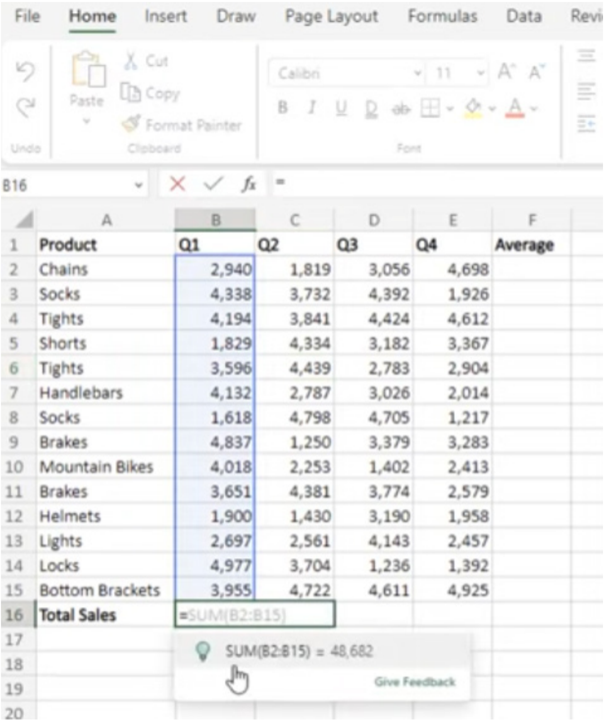
*Excel for Mac*

- **IMAGE** function.

Let's plough through.

*Formula Suggestions*

After you type the "=" sign in a cell or the Formula bar, Excel will automatically suggest a formula based upon contextual insights from your data. Formulae that can be suggested use **SUM**, **AVERAGE**, **COUNT**, **COUNTA**, **MAX** and **MIN**. Presently, there is support for the English language in Excel for the web only. This feature is rolling out to production for web users.



=

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Product | Q1 | Q2 | Q3 | Q4 | Average |
| 2 | Chains | 2,940 | 1,819 | 3,056 | 4,698 | |
| 3 | Socks | 4,338 | 3,732 | 4,392 | 1,926 | |
| 4 | Tights | 4,194 | 3,841 | 4,424 | 4,612 | |
| 5 | Shorts | 1,829 | 4,334 | 3,182 | 3,367 | |
| 6 | Tights | 3,596 | 4,439 | 2,783 | 2,904 | |
| 7 | Handlebars | 4,132 | 2,787 | 3,026 | 2,014 | |
| 8 | Socks | 1,618 | 4,798 | 4,705 | 1,217 | |
| 9 | Brakes | 4,837 | 1,250 | 3,379 | 3,283 | |
| 10 | Mountain Bikes | 4,018 | 2,253 | 1,402 | 2,413 | |
| 11 | Brakes | 3,651 | 4,381 | 3,774 | 2,579 | |
| 12 | Helmets | 1,900 | 1,430 | 3,190 | 1,958 | |
| 13 | Lights | 2,697 | 2,561 | 4,143 | 2,457 | |
| 14 | Locks | 4,977 | 3,704 | 1,236 | 1,392 | |
| 15 | Bottom Brackets | 3,955 | 4,722 | 4,611 | 4,925 | |
| 16 | Total Sales | =SUM(B2:B15) | | | | |
| 17 | | | | | | |
| 18 | | | | | | |
| 19 | | | | | | |
| 20 | | | | | | |

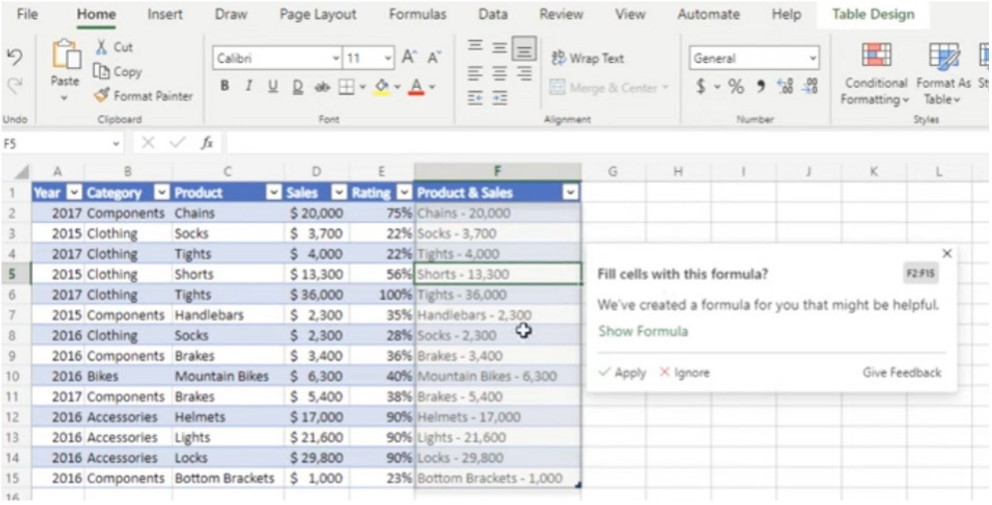
SUM(B2:B15) = 48,682

Give Feedback

***Formula by Example***

Also coming to Excel for the web, as you are performing manual and repetitive data entry in a column, Excel will now suggest you to fill the entire column with a formula when a pattern has been identified. This is similar to Flash Fill, but now, instead of static text - now formulae can be suggested. It's like dynamic Power Query functionality without needing to refresh.

Simply start typing in the column and wait for the dialog to pop up:



| | Year | Category | Product | Sales | Rating | Product & Sales |
|---|---|---|---|---|---|---|
| 2 | 2017 | Components | Chains | $ 20,000 | 75% | Chains - 20,000 |
| 3 | 2015 | Clothing | Socks | $ 3,700 | 22% | Socks - 3,700 |
| 4 | 2017 | Clothing | Tights | $ 4,000 | 22% | Tights - 4,000 |
| 5 | 2015 | Clothing | Shorts | $ 13,300 | 56% | Shorts - 13,300 |
| 6 | 2017 | Clothing | Tights | $ 36,000 | 100% | Tights - 36,000 |
| 7 | 2015 | Components | Handlebars | $ 2,300 | 35% | Handlebars - 2,300 |
| 8 | 2016 | Clothing | Socks | $ 2,300 | 28% | Socks - 2,300 |
| 9 | 2016 | Components | Brakes | $ 3,400 | 36% | Brakes - 3,400 |
| 10 | 2016 | Bikes | Mountain Bikes | $ 6,300 | 40% | Mountain Bikes - 6,300 |
| 11 | 2017 | Components | Brakes | $ 5,400 | 38% | Brakes - 5,400 |
| 12 | 2016 | Accessories | Helmets | $ 17,000 | 90% | Helmets - 17,000 |
| 13 | 2016 | Accessories | Lights | $ 21,600 | 90% | Lights - 21,600 |
| 14 | 2016 | Accessories | Locks | $ 29,800 | 90% | Locks - 29,800 |
| 15 | 2016 | Components | Bottom Brackets | $ 1,000 | 23% | Bottom Brackets - 1,000 |

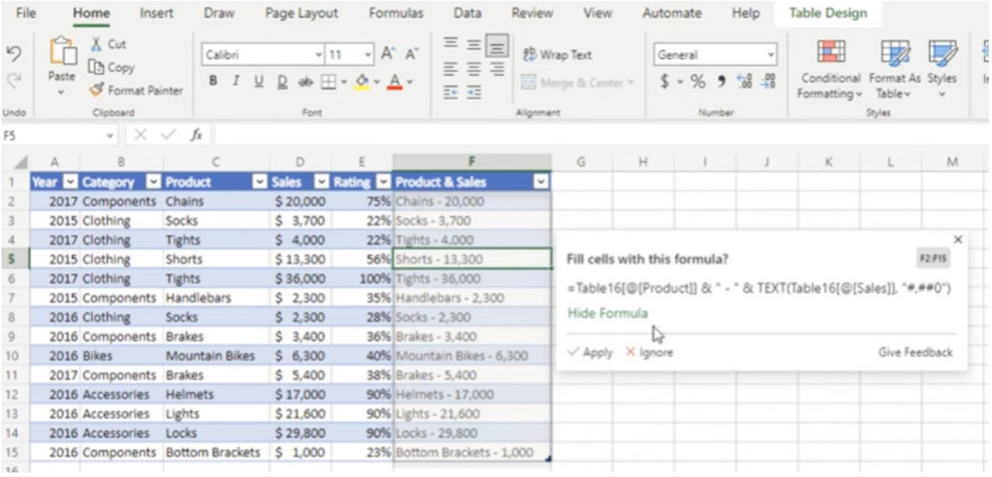
Fill cells with this formula? F2:F15

We've created a formula for you that might be helpful.

Show Formula

✓ Apply ✕ Ignore Give Feedback

You may then review the formula suggested:



| | Year | Category | Product | Sales | Rating | Product & Sales |
|---|---|---|---|---|---|---|
| 2 | 2017 | Components | Chains | $ 20,000 | 75% | Chains - 20,000 |
| 3 | 2015 | Clothing | Socks | $ 3,700 | 22% | Socks - 3,700 |
| 4 | 2017 | Clothing | Tights | $ 4,000 | 22% | Tights - 4,000 |
| 5 | 2015 | Clothing | Shorts | $ 13,300 | 56% | Shorts - 13,300 |
| 6 | 2017 | Clothing | Tights | $ 36,000 | 100% | Tights - 36,000 |
| 7 | 2015 | Components | Handlebars | $ 2,300 | 35% | Handlebars - 2,300 |
| 8 | 2016 | Clothing | Socks | $ 2,300 | 28% | Socks - 2,300 |
| 9 | 2016 | Components | Brakes | $ 3,400 | 36% | Brakes - 3,400 |
| 10 | 2016 | Bikes | Mountain Bikes | $ 6,300 | 40% | Mountain Bikes - 6,300 |
| 11 | 2017 | Components | Brakes | $ 5,400 | 38% | Brakes - 5,400 |
| 12 | 2016 | Accessories | Helmets | $ 17,000 | 90% | Helmets - 17,000 |
| 13 | 2016 | Accessories | Lights | $ 21,600 | 90% | Lights - 21,600 |
| 14 | 2016 | Accessories | Locks | $ 29,800 | 90% | Locks - 29,800 |
| 15 | 2016 | Components | Bottom Brackets | $ 1,000 | 23% | Bottom Brackets - 1,000 |

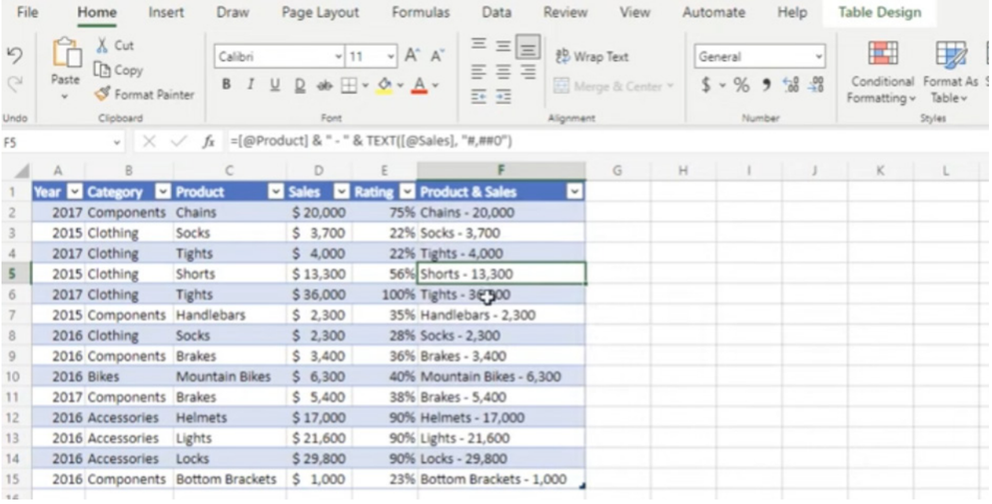
Fill cells with this formula? F2:F15

```
=Table16[@[Product]] & " - " & TEXT(Table16[@[Sales]], "#,##0")
```

Hide Formula

✓ Apply ✕ Ignore Give Feedback

Clicking Apply will insert the suggested formula into the range.



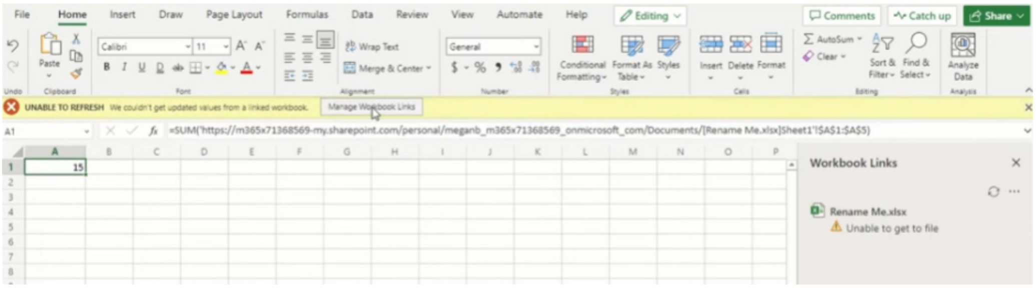
```
=[@Product] & " - " & TEXT([@Sales], "#,##0")
```

| | Year | Category | Product | Sales | Rating | Product & Sales |
|---|---|---|---|---|---|---|
| 2 | 2017 | Components | Chains | $ 20,000 | 75% | Chains - 20,000 |
| 3 | 2015 | Clothing | Socks | $ 3,700 | 22% | Socks - 3,700 |
| 4 | 2017 | Clothing | Tights | $ 4,000 | 22% | Tights - 4,000 |
| 5 | 2015 | Clothing | Shorts | $ 13,300 | 56% | Shorts - 13,300 |
| 6 | 2017 | Clothing | Tights | $ 36,000 | 100% | Tights - 36,000 |
| 7 | 2015 | Components | Handlebars | $ 2,300 | 35% | Handlebars - 2,300 |
| 8 | 2016 | Clothing | Socks | $ 2,300 | 28% | Socks - 2,300 |
| 9 | 2016 | Components | Brakes | $ 3,400 | 36% | Brakes - 3,400 |
| 10 | 2016 | Bikes | Mountain Bikes | $ 6,300 | 40% | Mountain Bikes - 6,300 |
| 11 | 2017 | Components | Brakes | $ 5,400 | 38% | Brakes - 5,400 |
| 12 | 2016 | Accessories | Helmets | $ 17,000 | 90% | Helmets - 17,000 |
| 13 | 2016 | Accessories | Lights | $ 21,600 | 90% | Lights - 21,600 |
| 14 | 2016 | Accessories | Locks | $ 29,800 | 90% | Locks - 29,800 |
| 15 | 2016 | Components | Bottom Brackets | $ 1,000 | 23% | Bottom Brackets - 1,000 |

***Suggested Links***

Also new to Excel for the web, Suggested Links will allow a new Cloud workbook store data which will detect when an external link to a Cloud workbook is broken. It will then suggest a new location in order to fix the broken link. This feature is currently rolling out to Production.

For example, here in Excel for the web, the software has detected an issue:

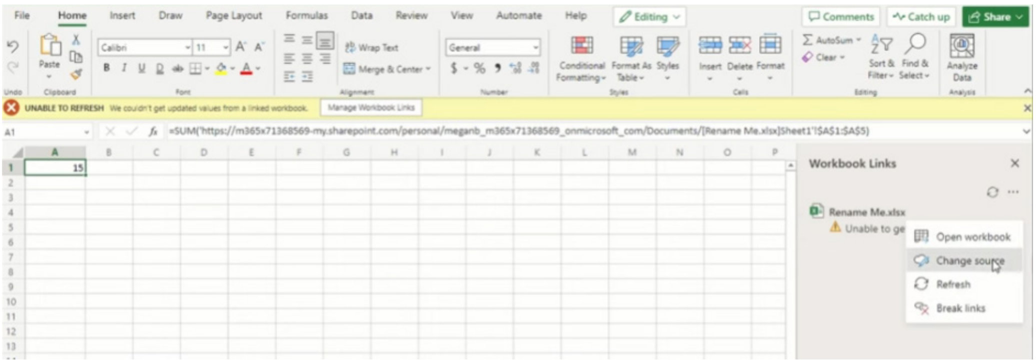

UNABLE TO REFRESH We couldn't get updated values from a linked workbook. Manage Workbook Links

=SUM('https://m365x71368569-my.sharepoint.com/personal/meganb_m365x71368569_onmicrosoft_com/Documents/[Rename Me.xlsx]Sheet1'!$A$1:$A$5)

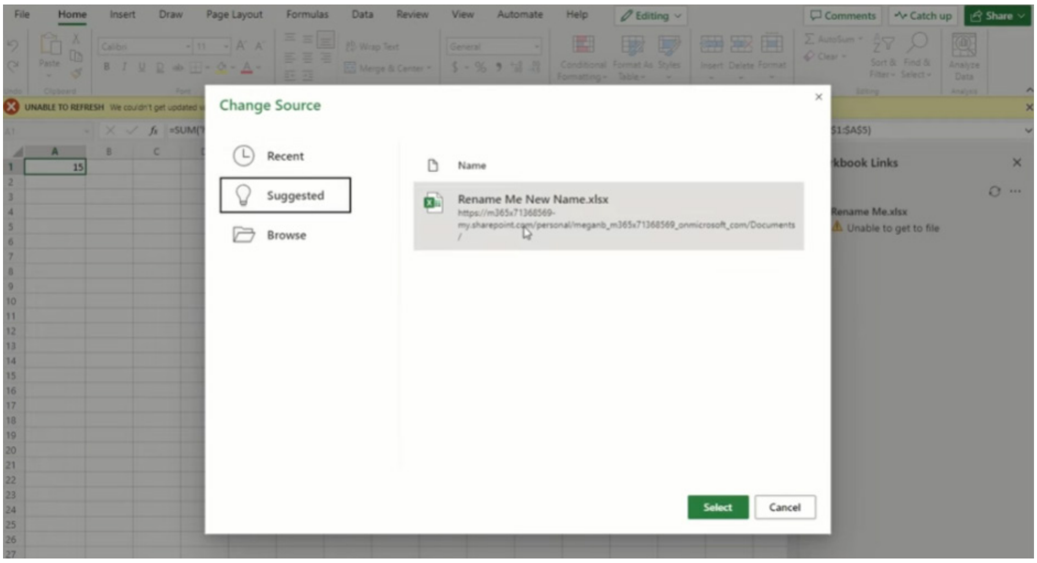
Workbook Links

Rename Me.xlsx
Unable to get to file

Clicking on the ellipsis (…) in the 'Workbook Links' pane will allow you to 'Change source' *viz.*

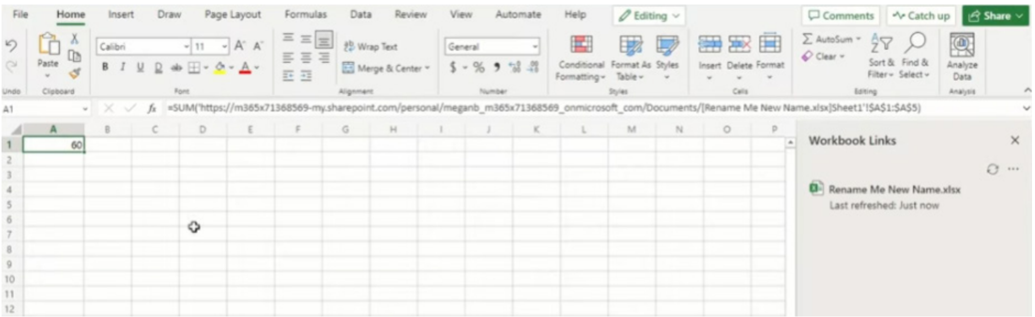


In the 'Change Source' dialog, selecting 'Suggested' prompts you with a suggested file to link to:


Change Source

Rename Me New Name.xlsx
https://m365x71368569-my.sharepoint.com/personal/meganb_m365x71368569_onmicrosoft_com/Documents/

If satisfied with the suggestion, clicking Select will change the source and rectify the error.

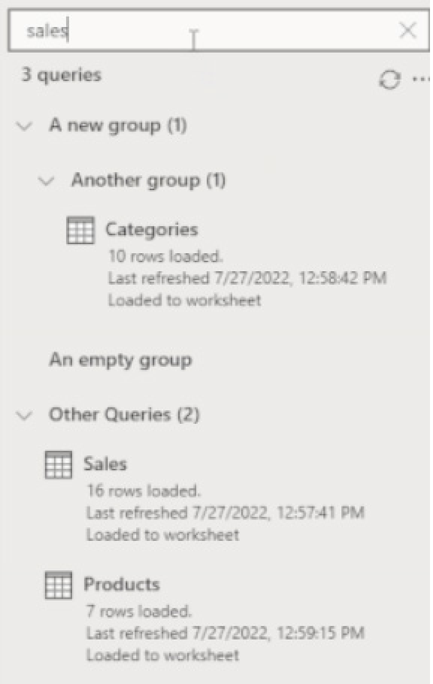

=SUM('https://m365x71368569-my.sharepoint.com/personal/meganb_m365x71368569_onmicrosoft_com/Documents/[Rename Me New Name.xlsx]Sheet1'!$A$1:$A$5)

Workbook Links

Rename Me New Name.xlsx
Last refreshed: Just now

### *Add Search bar in Queries pane*

Another new feature for Excel for the web is the addition of the Search bar in the Queries search pane:



sales

3 queries

A new group (1)

Another group (1)

Categories
10 rows loaded.
Last refreshed 7/27/2022, 12:58:42 PM
Loaded to worksheet

An empty group

Other Queries (2)

Sales
16 rows loaded.
Last refreshed 7/27/2022, 12:57:41 PM
Loaded to worksheet

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

### *IMAGE function*

The **IMAGE** function is now available in Excel for the web, Excel for Windows and Excel for Mac.

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])**

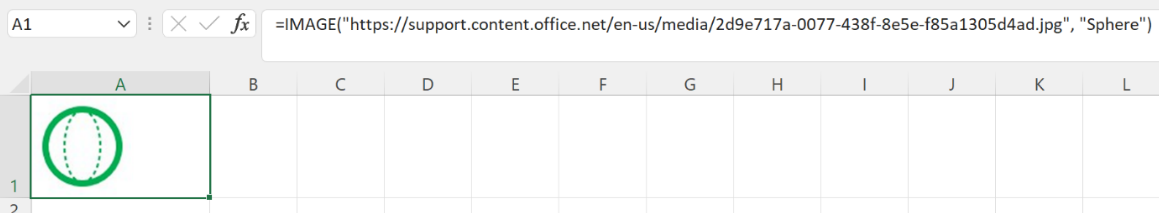
The arguments are as follows:

- **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")**

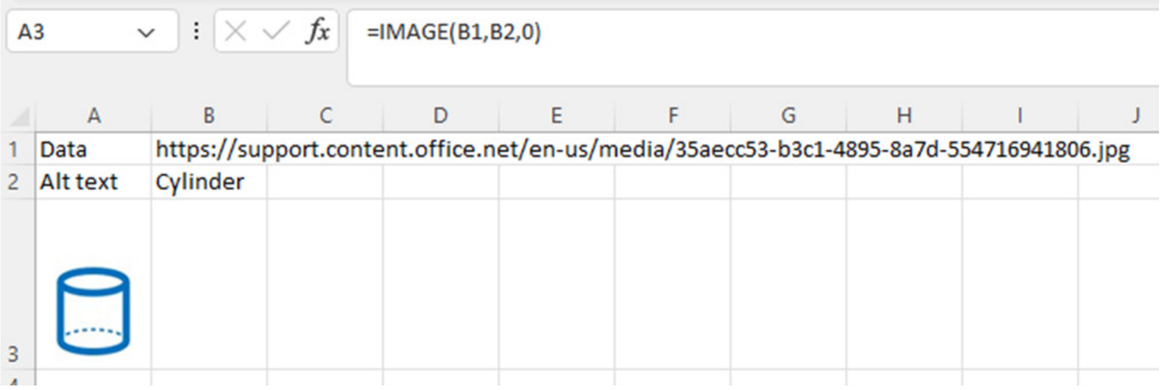


A1

=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**

As highlighted back in October’s newsletter, there are areas for improvement, such as:

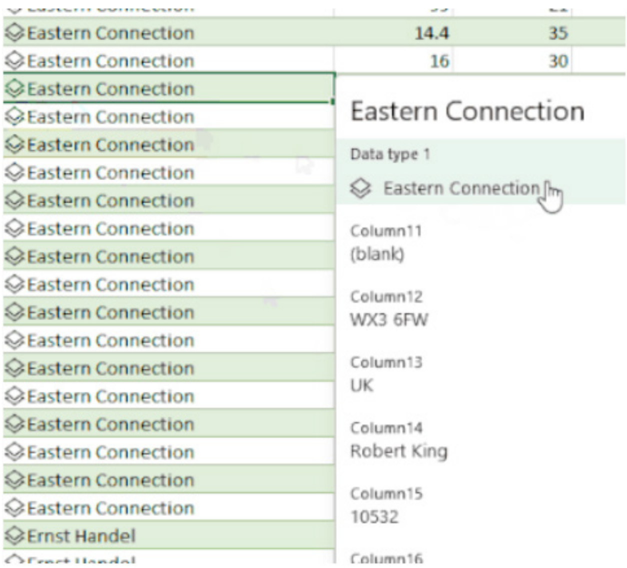
- if the URL to the image file you are using is pointing to a site that requires authentication, the image will not render
- zooming in and out with images in cells may distort the images
- moving between platforms (for example, Windows and Mac) may result in irregular image rendering.

***Add keyboard shortcut to open the Power Query editor***

You have simply no idea the debate that was triggered from this little gem. Finally, it was decided that **ALT + F12** (Windows 32-bit) and **Option + F12** (Mac) will open the Power Query editor.

***Create nested Power Query data types (Insiders)***

Data may now be organised better, by creating nested data types (Power Query Data Types with multiple levels). This feature is currently rolling out to Insiders users for Windows.



| | | |
|---|---|---|
| Eastern Connection | 14.4 | 35 |
| Eastern Connection | 16 | 30 |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Eastern Connection | | |
| Ernst Handel | | |

Eastern Connection

Data type 1
Eastern Connection

Column11
(blank)

Column12
WX3 6FW

Column13
UK

Column14
Robert King

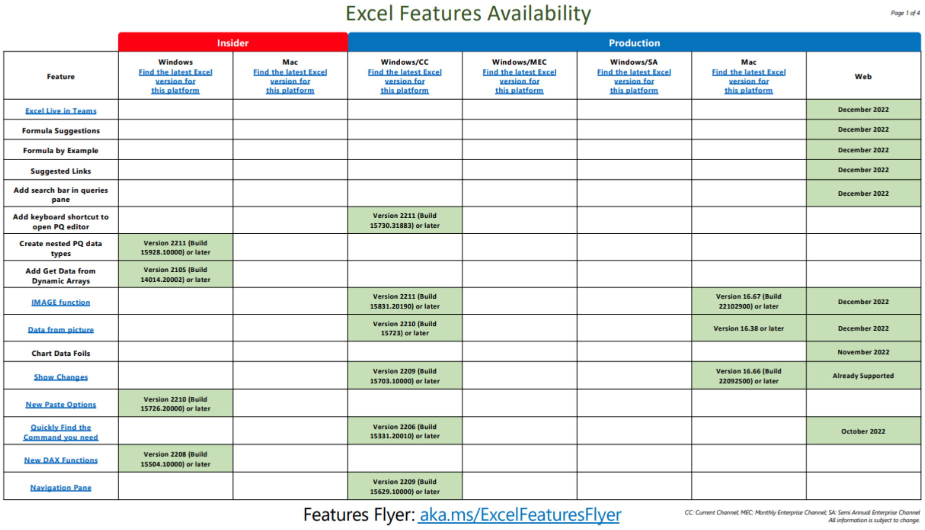
Column15
10532

Column16

### *Add Get Data from Dynamic Arrays (Insiders)*

‘Get Data from Table / Range’ now supports importing data from Dynamic Arrays. This means you may load said arrays into Power Query and transform your data, just as you can with static ranges or data from Excel Tables. This feature is currently rolling out to Windows Production.

The updated version of the grid with all the new features is fast becoming too complicated to show clearly here. Nonetheless, you can find the interactive links at aka.ms/ExcelFeaturesFlyer.



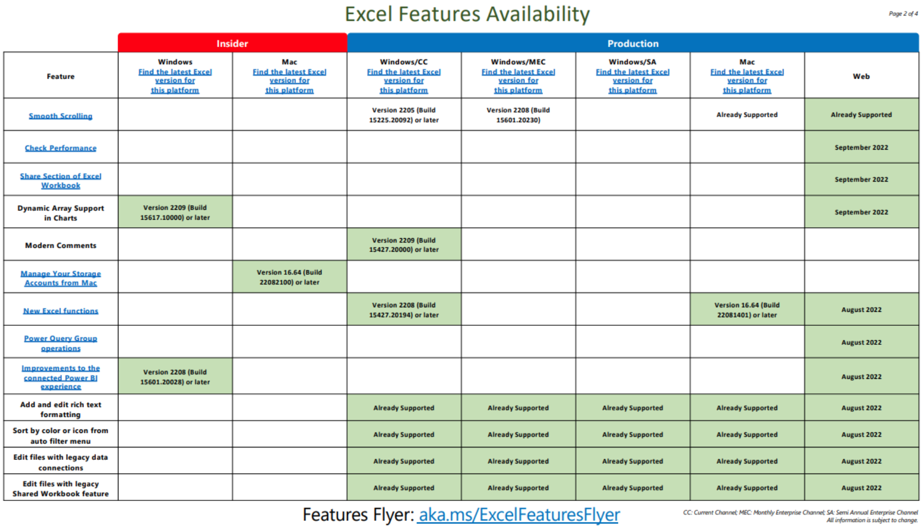
Excel Features Availability

Page 1 of 4

| | Insider | | Production | | | | |
|---|---|---|---|---|---|---|---|
| Feature | Windows<br>Find the latest Excel version for this platform | Mac<br>Find the latest Excel version for this platform | Windows/CC<br>Find the latest Excel version for this platform | Windows/MEC<br>Find the latest Excel version for this platform | Windows/SA<br>Find the latest Excel version for this platform | Mac<br>Find the latest Excel version for this platform | Web |
| Excel Live in Teams | | | | | | | December 2022 |
| Formula Suggestions | | | | | | | December 2022 |
| Formula by Example | | | | | | | December 2022 |
| Suggested Links | | | | | | | December 2022 |
| Add search bar in queries pane | | | | | | | December 2022 |
| Add keyboard shortcut to open PQ editor | | | Version 2211 (Build 15730.31883) or later | | | | |
| Create nested PQ data types | Version 2211 (Build 15928.10000) or later | | | | | | |
| Add Get Data from Dynamic Arrays | Version 2105 (Build 14014.20002) or later | | | | | | |
| IMAGE function | | | Version 2211 (Build 15831.20190) or later | | | Version 16.67 (Build 22102900) or later | December 2022 |
| Data from picture | | | Version 2210 (Build 15723) or later | | | Version 16.38 or later | December 2022 |
| Chart Data Foils | | | | | | | November 2022 |
| Show Changes | | | Version 2209 (Build 15703.10000) or later | | | Version 16.66 (Build 22092500) or later | Already Supported |
| New Paste Options | Version 2210 (Build 15726.20000) or later | | | | | | |
| Quickly Find the Command you need | | | Version 2206 (Build 15331.20010) or later | | | | October 2022 |
| New DAX Functions | Version 2208 (Build 15504.10000) or later | | | | | | |
| Navigation Pane | | | Version 2209 (Build 15629.10000) or later | | | | |

Features Flyer: aka.ms/ExcelFeaturesFlyer

CC: Current Channel; MEC: Monthly Enterprise Channel; SA: Semi Annual Enterprise Channel
All information is subject to change.



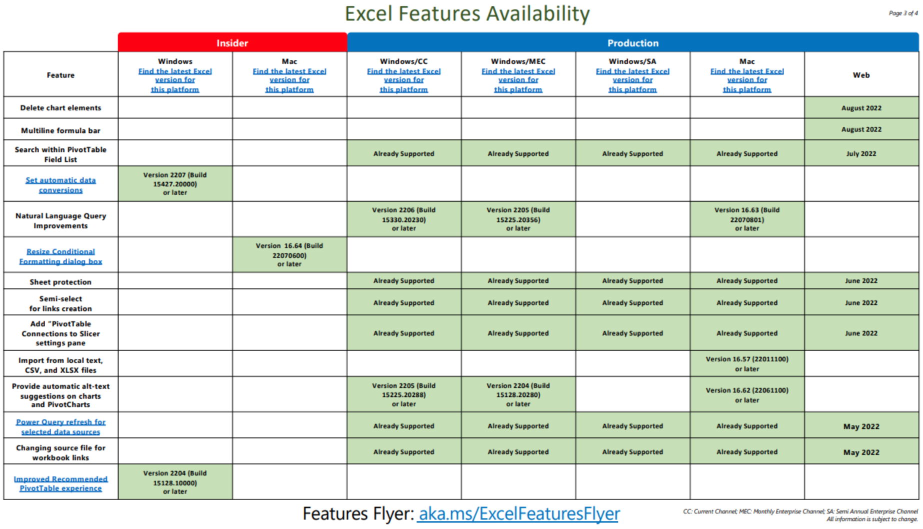
Excel Features Availability

Page 2 of 4

| | Insider | | Production | | | | |
|---|---|---|---|---|---|---|---|
| Feature | Windows<br>Find the latest Excel version for this platform | Mac<br>Find the latest Excel version for this platform | Windows/CC<br>Find the latest Excel version for this platform | Windows/MEC<br>Find the latest Excel version for this platform | Windows/SA<br>Find the latest Excel version for this platform | Mac<br>Find the latest Excel version for this platform | Web |
| Smooth Scrolling | | | Version 2205 (Build 15225.20092) or later | Version 2208 (Build 15601.20230) | | Already Supported | Already Supported |
| Check Performance | | | | | | | September 2022 |
| Share Section of Excel Workbook | | | | | | | September 2022 |
| Dynamic Array Support in Charts | Version 2209 (Build 15617.10000) or later | | | | | | September 2022 |
| Modern Comments | | | Version 2209 (Build 15427.20000) or later | | | | |
| Manage Your Storage Accounts from Mac | | Version 16.64 (Build 22082100) or later | | | | | |
| New Excel functions | | | Version 2208 (Build 15427.20194) or later | | | Version 16.64 (Build 22081401) or later | August 2022 |
| 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 | | | Already Supported | Already Supported | Already Supported | Already Supported | August 2022 |
| Sort by color or icon from auto filter menu | | | Already Supported | Already Supported | Already Supported | Already Supported | August 2022 |
| Edit files with legacy data connections | | | Already Supported | Already Supported | Already Supported | Already Supported | August 2022 |
| Edit files with legacy Shared Workbook feature | | | Already Supported | Already Supported | Already Supported | Already Supported | August 2022 |

Features Flyer: aka.ms/ExcelFeaturesFlyer

CC: Current Channel; MEC: Monthly Enterprise Channel; SA: Semi Annual Enterprise Channel
All information is subject to change.



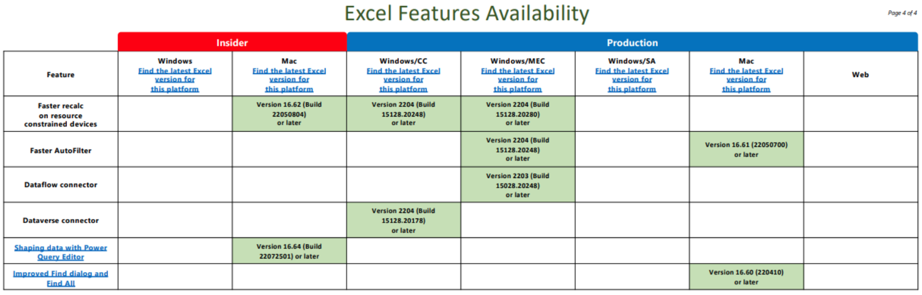
Excel Features Availability

Page 3 of 4

| | Insider | | Production | | | | |
|---|---|---|---|---|---|---|---|
| Feature | Windows<br>Find the latest Excel version for this platform | Mac<br>Find the latest Excel version for this platform | Windows/CC<br>Find the latest Excel version for this platform | Windows/MEC<br>Find the latest Excel version for this platform | Windows/SA<br>Find the latest Excel version for this platform | Mac<br>Find the latest Excel version for this platform | Web |
| Delete chart elements | | | | | | | August 2022 |
| Multiline formula bar | | | | | | | August 2022 |
| Search within PivotTable Field List | | | Already Supported | Already Supported | Already Supported | Already Supported | 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 | | | | | |
| Sheet protection | | | Already Supported | Already Supported | Already Supported | Already Supported | June 2022 |
| Semi-select for links creation | | | Already Supported | Already Supported | Already Supported | Already Supported | June 2022 |
| Add "PivotTable Connections to Slicer settings pane | | | Already Supported | Already Supported | Already Supported | Already Supported | 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 | | | Already Supported | Already Supported | Already Supported | Already Supported | May 2022 |
| Changing source file for workbook links | | | Already Supported | Already Supported | Already Supported | Already Supported | May 2022 |
| Improved Recommended PivotTable experience | Version 2204 (Build 15128.10000) or later | | | | | | |

Features Flyer: aka.ms/ExcelFeaturesFlyer

CC: Current Channel; MEC: Monthly Enterprise Channel; SA: Semi Annual Enterprise Channel
All information is subject to change.



Excel Features Availability

Page 4 of 4

| | Insider | | Production | | | | |
|---|---|---|---|---|---|---|---|
| Feature | Windows<br>Find the latest Excel version for this platform | Mac<br>Find the latest Excel version for this platform | Windows/CC<br>Find the latest Excel version for this platform | Windows/MEC<br>Find the latest Excel version for this platform | Windows/SA<br>Find the latest Excel version for this platform | Mac<br>Find the latest Excel version for this platform | Web |
| 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 16.64 (Build 22072501) or later | | | | | |
| Improved Find dialog and Find All | | | | | | Version 16.60 (220410) or later | |

More next month, we're sure.

## The A to Z of Excel Functions: ISO.CEILING

This function returns a number that is rounded up to the nearest integer or to the nearest multiple of significance.  Regardless of the sign of the number, the number is rounded up.  However, if the number or the significance is zero (0), zero is returned.

The **ISO.CEILING** function employs the following syntax to operate:

**ISO.CEILING(number, [significance])**

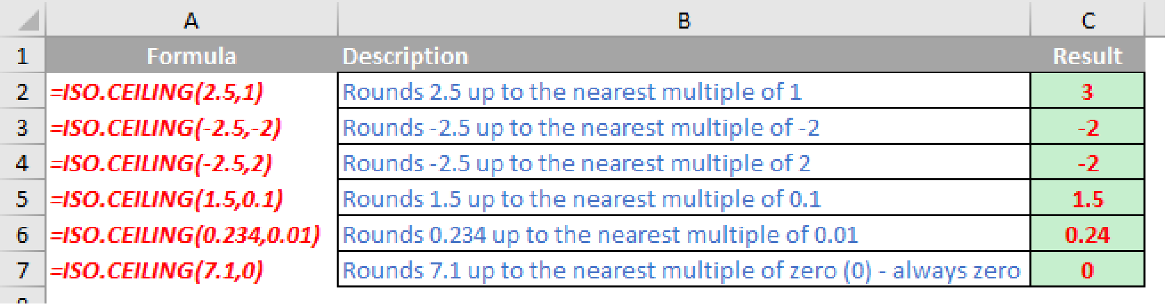
The **ISO.CEILING** function has the following arguments:

- **number:** this is required and represents the value you wish to round
- **significance:** this argument is optional.  This is the multiple used for rounding.

It should be further noted that:

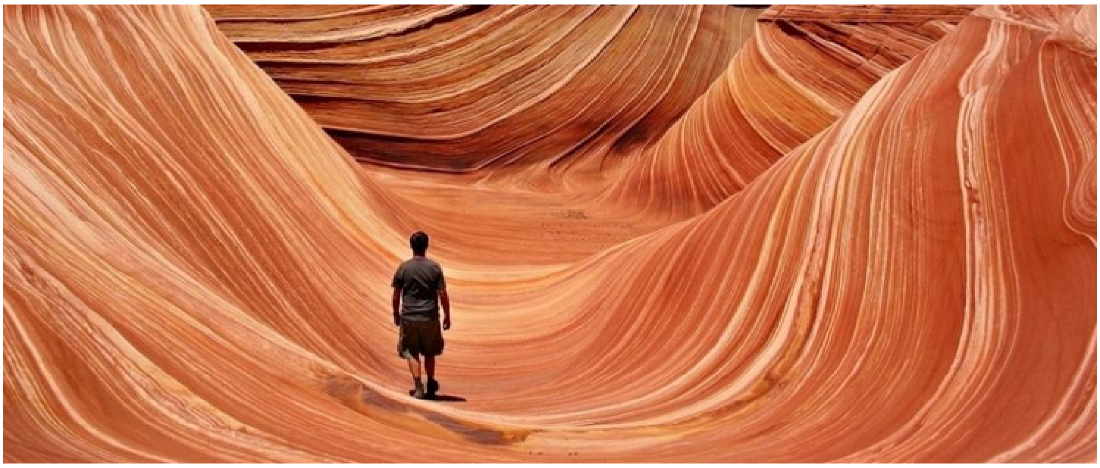
- if either argument is nonnumeric, **ISO.CEILING** returns the *#VALUE!* error value
- this function should not be confused with **CEILING**, which may round up or down, depending upon the arguments used
- this function is not recognised by Excel's AutoComplete.  Do not be alarmed if it is not offered as a suggested function; it is still a current and valid function in Excel
- if **significance** is omitted, its default value is one (1)
- the absolute value of the multiple is used, so that the **ISO.CEILING** function returns the mathematical ceiling irrespective of the signs of number and significance
- some modellers believe **CEILING.MATH** replaced **ISO.CEILING** after initial beta testing, but this is not the case.  These two functions are similar, but different.

Please see our example below:



| | A | B | C |
|---|---|---|---|
| 1 | Formula | Description | Result |
| 2 | =ISO.CEILING(2.5,1) | Rounds 2.5 up to the nearest multiple of 1 | 3 |
| 3 | =ISO.CEILING(-2.5,-2) | Rounds -2.5 up to the nearest multiple of -2 | -2 |
| 4 | =ISO.CEILING(-2.5,2) | Rounds -2.5 up to the nearest multiple of 2 | -2 |
| 5 | =ISO.CEILING(1.5,0.1) | Rounds 1.5 up to the nearest multiple of 0.1 | 1.5 |
| 6 | =ISO.CEILING(0.234,0.01) | Rounds 0.234 up to the nearest multiple of 0.01 | 0.24 |
| 7 | =ISO.CEILING(7.1,0) | Rounds 7.1 up to the nearest multiple of zero (0) - always zero | 0 |

## The A to Z of Excel Functions: ISODD



At the time of writing, there are 12 **IS** functions, *i.e.* functions that give rise to a TRUE or FALSE value depending upon whether a certain condition is met:

1. **ISBLANK(reference):** checks whether the **reference** is to an empty cell
2. **ISERR(value):** checks whether the **value** is an error (*e.g. #REF!, #DIV/0!, #NULL!*). This check specifically excludes *#N/A*
3. **ISERROR(value):** checks whether the **value** is an error (*e.g. #REF!, #DIV/0!, #NULL!*). This is probably the most commonly used of these functions in financial modelling
4. **ISEVEN(number):** checks to see if the **number** is even
5. **ISFORMULA(reference):** checks to see whether the **reference** is to a cell containing a formula
6. **ISLOGICAL(value):** checks to see whether the **value** is a logical (TRUE or FALSE) value
7. **ISNA(value):** checks to see whether the **value** is *#N/A*. This gives us the rather crude identity **ISERR + ISNA = ISERROR**
8. **ISNONTEXT(value):** checks whether the **value** is not text (*N.B.* blank cells are not text)
9. **ISNUMBER(Value):** checks whether the **value** is a number
10. **ISODD(number):** checks to see if the **number** is odd. Personally, I find the number 46 very odd, but Excel doesn't
11. **ISREF(value):** checks whether the **value** is a reference
12. **ISTEXT(value):** checks whether the **value** is text.

We covered many of these last month. As stated above, the **ISODD** function checks whether a number is odd. It has the following syntax:

**ISODD(number)**

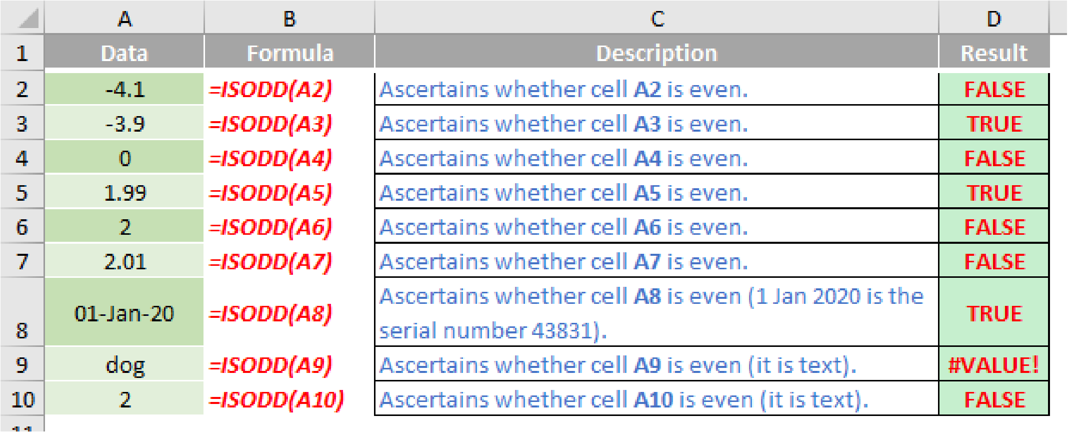
The **ISODD** function has the following argument:

- **number:** this is required and represents the **number** for which you wish to determine whether it is odd. If **number** is not an integer, it is truncated (*i.e.* not rounded, simply ended).

It should be further noted that:

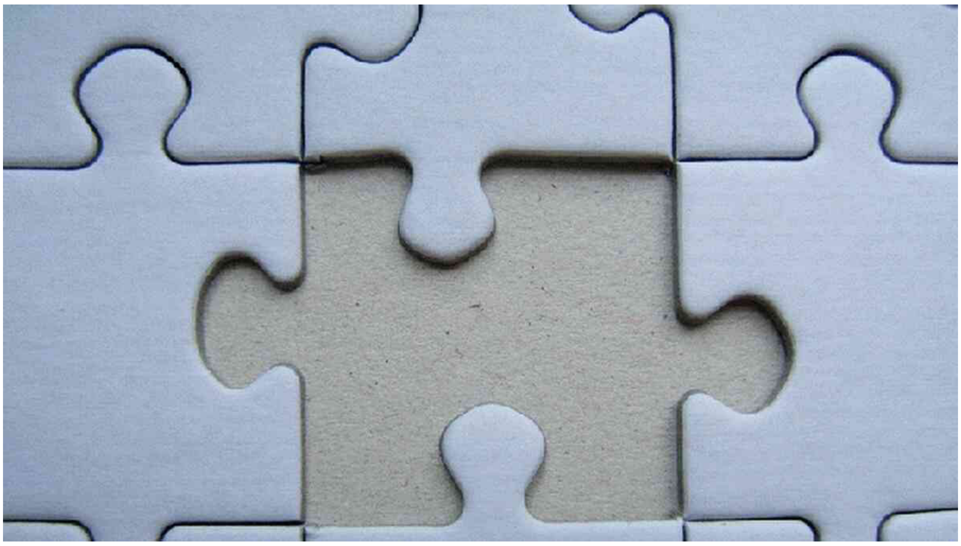
- if **number** is nonnumeric, **ISODD** returns the *#VALUE!* error value.

Please see more examples below:



| | A | B | C | D |
|---|---|---|---|---|
| 1 | Data | Formula | Description | Result |
| 2 | -4.1 | *=ISODD(A2)* | Ascertains whether cell **A2** is even. | FALSE |
| 3 | -3.9 | *=ISODD(A3)* | Ascertains whether cell **A3** is even. | TRUE |
| 4 | 0 | *=ISODD(A4)* | Ascertains whether cell **A4** is even. | FALSE |
| 5 | 1.99 | *=ISODD(A5)* | Ascertains whether cell **A5** is even. | TRUE |
| 6 | 2 | *=ISODD(A6)* | Ascertains whether cell **A6** is even. | FALSE |
| 7 | 2.01 | *=ISODD(A7)* | Ascertains whether cell **A7** is even. | FALSE |
| 8 | 01-Jan-20 | *=ISODD(A8)* | Ascertains whether cell **A8** is even (1 Jan 2020 is the serial number 43831). | TRUE |
| 9 | dog | *=ISODD(A9)* | Ascertains whether cell **A9** is even (it is text). | #VALUE! |
| 10 | 2 | *=ISODD(A10)* | Ascertains whether cell **A10** is even (it is text). | FALSE |

## The A to Z of Excel Functions: ISOMITTED



The more observant amongst you will note we have not included this in our list of the 12 **IS** functions. This is because this doesn't necessarily play nicely in Excel on its own like the others (it likes to play with **LAMBDA**). This new function checks whether the value is missing, and returns either TRUE (value is missing) or FALSE (value is not missing) accordingly. The syntax is simple:

**ISOMITTED(argument)**

where:

- **argument** is a required parameter, and is the value you want to test, which may be based upon a **LAMBDA**.

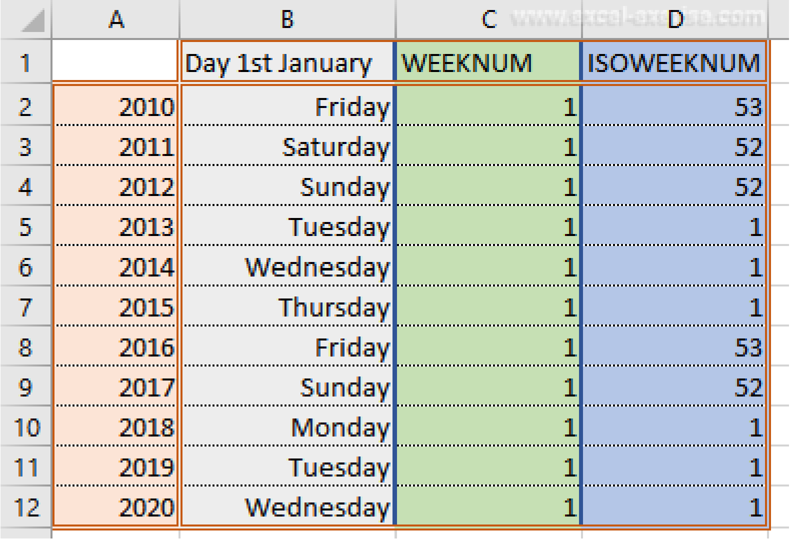
The example

**=LAMBDA(arg1, [arg2], IF(ISOMITTED(arg2), arg1, arg2))**

will return the value of **arg1** if **arg2** is omitted (this is why **arg2** is in square brackets as it is an optional argument); otherwise, it will return the value of **arg2**.

Simply put, this lambda will return the value of **arg1** if **arg2** is omitted; otherwise, it will return the value of **arg2**. I'd like to have called this lambda function **Jason** as it sort of checks if his **arg**s are naught, but sadly it's not Friday 13th…

# The A to Z of Excel Functions: ISOWEEKNUM



| | A | B | C | D |
|---|---|---|---|---|
| 1 | | Day 1st January | WEEKNUM | ISOWEEKNUM |
| 2 | 2010 | Friday | 1 | 53 |
| 3 | 2011 | Saturday | 1 | 52 |
| 4 | 2012 | Sunday | 1 | 52 |
| 5 | 2013 | Tuesday | 1 | 1 |
| 6 | 2014 | Wednesday | 1 | 1 |
| 7 | 2015 | Thursday | 1 | 1 |
| 8 | 2016 | Friday | 1 | 53 |
| 9 | 2017 | Sunday | 1 | 52 |
| 10 | 2018 | Monday | 1 | 1 |
| 11 | 2019 | Tuesday | 1 | 1 |
| 12 | 2020 | Wednesday | 1 | 1 |

The ISO week date system is effectively a leap week calendar system that is part of the ISO 8601 date and time standard issued by the International Organization for Standardization (ISO) since 1988 (last revised in 2004) and, before that, it was defined in ISO (R) 2015 since 1971. It is used (mainly) in government and business for fiscal years, as well as in timekeeping. This was previously known as "Industrial date coding". The system specifies a week year atop the Gregorian calendar by defining a notation for ordinal weeks of the year.

An ISO week-numbering year (also called ISO year informally) has 52 or 53 full weeks. The extra week is sometimes referred to as a leap week, although ISO 8601 does not use this term.

Weeks start with Monday. Each week's year is the Gregorian year in which the Thursday falls. The first week of the year, hence, always contains 4 January. ISO week year numbering therefore slightly deviates from the Gregorian for some days close to 1 January.

This function returns the number of the ISO week number of the year for a given date.

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

**ISOWEEKNUM(date)**

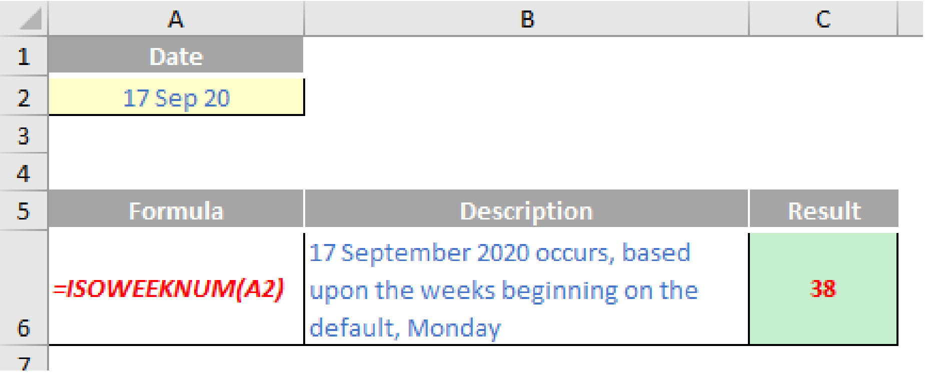
The **ISOWEEKNUM** function has the following arguments:

- **date:** this is required and represents the date-time code used by Excel for date and time calculations.

It should be further noted that:

- Microsoft Excel stores dates as sequential numbers so they can be used in calculations. By default, January 1, 1900 is serial number 1, and January 1, 2008 is serial number 39,448 because it is 39,447 days after January 1, 1900
- if the date argument is not a valid number, **ISOWEEKNUM** returns the *#NUM!* error value
- if the date argument is not a valid date type, **ISOWEEKNUM** returns the *#VALUE!* error value.

Please see our final example for this month below:



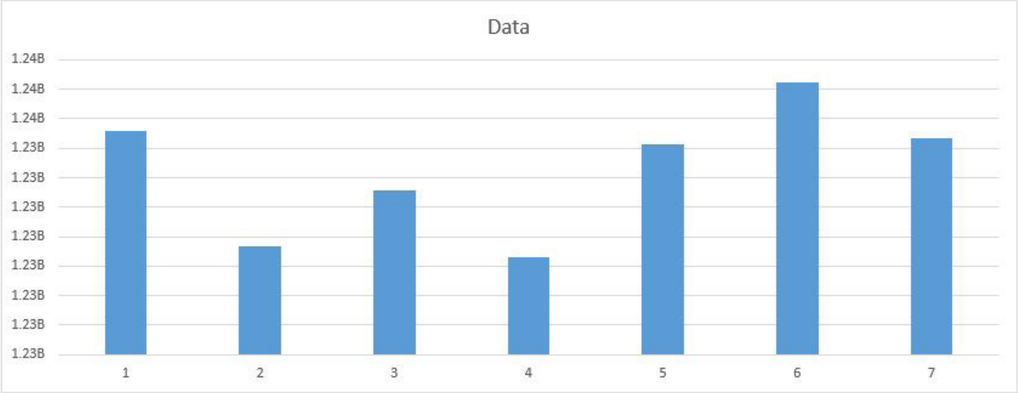
| | A | B | C |
|---|---|---|---|
| 1 | Date | | |
| 2 | 17 Sep 20 | | |
| 3 | | | |
| 4 | | | |
| 5 | Formula | Description | Result |
| 6 | *=ISOWEEKNUM(A2)* | 17 September 2020 occurs, based upon the weeks beginning on the default, Monday | **38** |
| 7 | | | |

More Excel Functions next month.

# Beat the Boredom Suggested Solution

Earlier this newsletter, we asked if you could create a chart that would have the number format of the chart axis dynamically change when the values being charted increase or decrease.

Chart axes will, by default, take on the number formatting of the cells in the first data series that it uses. This means that if you set the number formatting in the data cells, you can have up to three different number formats built into the cell. However, if you use conditional formatting to get the fourth format, the conditional formatting doesn't flow onto the chart!



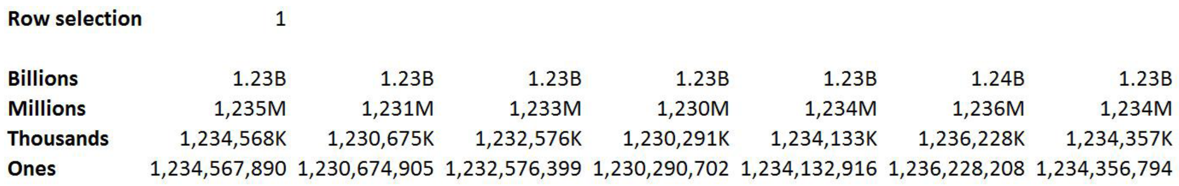
We mentioned some specific rules:

- It needs to change the number format into billions, millions, thousands or units depending on the numbers presented in the chart (at least four different conditions)
- No macros or user defined functions are allowed
- Conditional formatting is not allowed (not expecting that one??).

The last condition might have caught you out as we have shown previously how to create multiple custom number formats using conditional (number) formatting. So how did you fare?

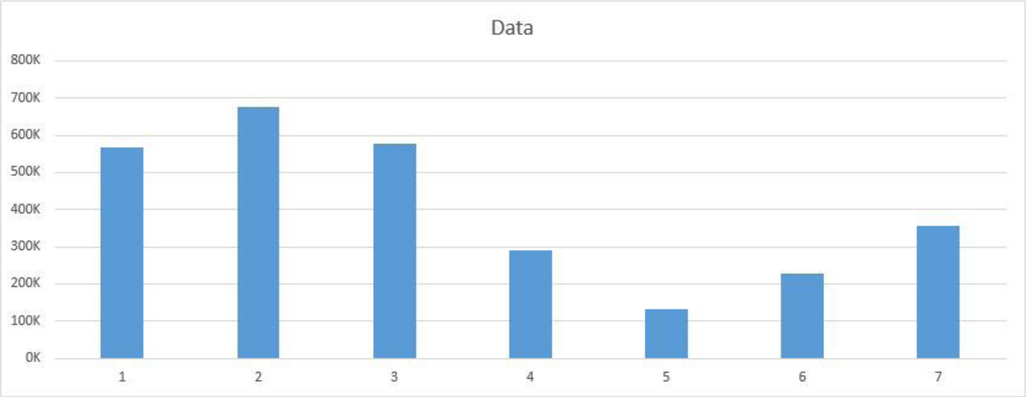
### *Suggested Solution*

The trick here is to take advantage of one of the perennial oft frowned upon functions in Excel. The **OFFSET** function, amongst its other neat features, can be used inside a named range to create a dynamic range that can select different rows depending on a condition that we set up. Therefore, if we set up four [4] rows, each with their own number format, then we can create a named range that chooses which row to get the chart data from, *viz.*



| **Row selection** | 1 | | | | | | |
|---|---|---|---|---|---|---|---|
| **Billions** | 1.23B | 1.23B | 1.23B | 1.23B | 1.23B | 1.24B | 1.23B |
| **Millions** | 1,235M | 1,231M | 1,233M | 1,230M | 1,234M | 1,236M | 1,234M |
| **Thousands** | 1,234,568K | 1,230,675K | 1,232,576K | 1,230,291K | 1,234,133K | 1,236,228K | 1,234,357K |
| **Ones** | 1,234,567,890 | 1,230,674,905 | 1,232,576,399 | 1,230,290,702 | 1,234,132,916 | 1,236,228,208 | 1,234,356,794 |

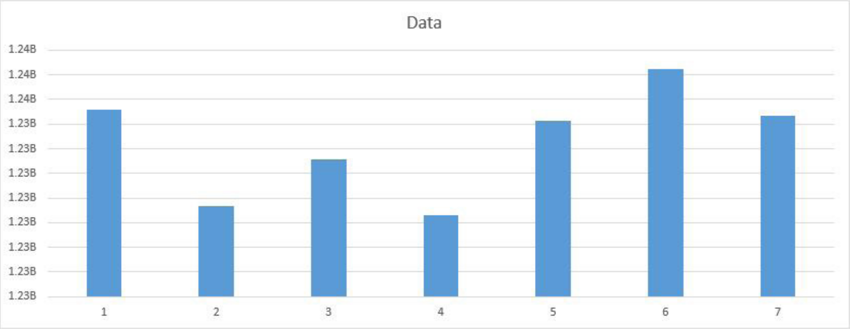
When you change the numbers, the named range will automatically pick up the correct row, and the chart axis will update in kind:



## Beat the Boredom Suggested Solution

Earlier this newsletter, we asked if you could create a chart that would have the number format of the chart axis dynamically change when the values being charted increase or decrease.

Chart axes will, by default, take on the number formatting of the cells in the first data series that it uses. This means that if you set the number formatting in the data cells, you can have up to three different number formats built into the cell. However, if you use conditional formatting to get the fourth format, the conditional formatting doesn't flow onto the chart!



We mentioned some specific rules:

- It needs to change the number format into billions, millions, thousands or units depending on the numbers presented in the chart (at least four different conditions)
- No macros or user defined functions are allowed
- Conditional formatting is not allowed (not expecting that one??).

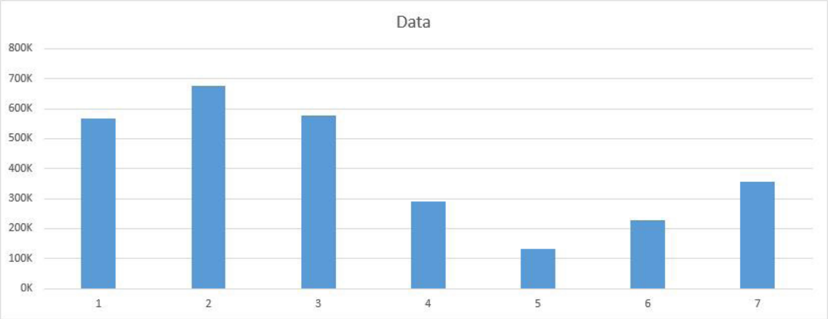
The last condition might have caught you out as we have shown previously how to create multiple custom number formats using conditional (number) formatting. So how did you fare?

***Suggested Solution***

The trick here is to take advantage of one of the perennial oft frowned upon functions in Excel. The **OFFSET** function, amongst its other neat features, can be used inside a named range to create a dynamic range that can select different rows depending on a condition that we set up. Therefore, if we set up four [4] rows, each with their own number format, then we can create a named range that chooses which row to get the chart data from, *viz.*

| **Row selection** | 1 | | | | | | |
|---|---|---|---|---|---|---|---|
| **Billions** | 1.23B | 1.23B | 1.23B | 1.23B | 1.23B | 1.24B | 1.23B |
| **Millions** | 1,235M | 1,231M | 1,233M | 1,230M | 1,234M | 1,236M | 1,234M |
| **Thousands** | 1,234,568K | 1,230,675K | 1,232,576K | 1,230,291K | 1,234,133K | 1,236,228K | 1,234,357K |
| **Ones** | 1,234,567,890 | 1,230,674,905 | 1,232,576,399 | 1,230,290,702 | 1,234,132,916 | 1,236,228,208 | 1,234,356,794 |

When you change the numbers, the named range will automatically pick up the correct row, and the chart axis will update in kind:



| **Row selection** | 3 | | | | | | |
|---|---|---|---|---|---|---|---|
| **Billions** | 0.00B | 0.00B | 0.00B | 0.00B | 0.00B | 0.00B | 0.00B |
| **Millions** | 1M | 1M | 1M | 0M | 0M | 0M | 0M |
| **Thousands** | 568K | 675K | 576K | 291K | 133K | 228K | 357K |
| **Ones** | 567,890 | 674,905 | 576,399 | 290,702 | 132,916 | 228,208 | 356,794 |

This is our attempt to find a workaround to get conditional number formatting into the chart axis. By no means are we claiming this is the only way to do it – if you have a more elegant solution, please let us know!

Until next time.

## Upcoming SumProduct Training Courses - COVID-19 update

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

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

| Location | Course | Date | Date | Duration | Duration |
|---|---|---|---|---|---|
| Online (Australia) | Power Pivot, Power Query and Power BI | 20 - 22 Mar 2023 | 09:00-17:00 AEDT | (-1 day) 22:00-17:00 GMT | 3 Days |
| Online (Australia) | Excel Tips and Tricks | 27 Mar 2023 | 09:00-17:00 AEDT | (-1 day) 22:00-17:00 GMT | 1 Day |
| Online (Australia) | Financial Modelling | 28 - 29 Mar 2023 | 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 at the **CTRL** and **ALT** keys:

| Keystroke | What it does |
|---|---|
| CTRL + ALT + SHIFT + F2 | Print |
| CTRL + ALT + SHIFT + F4 | Close application |
| CTRL + ALT + SHIFT + F9 | Recalculation: full rebuild |
| CTRL + ALT + SHIFT + TAB | Outdent *(it's a word apparently – look it up!)* |

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