SumProduct

NEWSLETTER #148 - March 2025

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

The team keeps getting bigger and bigger!

SumProduct welcomes its fourth MVP to its ranks as Data Platform MVP Ilgar Zarbaliyev joins our ranks. You can find out more about him below.

With so much travelling in the last month, perhaps news is a little light this month, but there is still plenty to keep you occupied. There is the usual Beat the Boredom Challenge, Charts & Dashboards Tips, Over to AI, Excel for Mac, Visual Basics, Power Pivot Principles, Power Query Pointers, the latest Power BI Updates (make sure you update!!), Excel Updates, plus we look at **PERCENTILE** functions in the A to Z of Excel functions and play Twister with the Keyboard Shortcuts to finish.

Happy reading and remember: stay safe, stay happy, stay healthy.

Liam Bastick, Managing Director, SumProduct

Ilgar Zarbaliyev Joins SumProduct

SumProduct is pleased to announce the addition of a FOURTH Microsoft Most Valuable Professional (MVP) into its ranks. Please welcome our latest Senior Manager **Ilgar Zarbaliyev**, a Data Platform MVP, who has over 20 years of experience in providing strategic insights through data analysis and business intelligence, using tools like Microsoft 365 and Power BI.

Based in Baku, Azerbaijan, Ilgar is passionate about sharing his knowledge and skills with others, especially educators and learners. He is presently pursuing a postgraduate degree in Artificial Intelligence (AI) and Machine Learning (ML) from Texas McCombs School of Business, in order to enhance his understanding of the latest technologies and trends in data science.

Ilgar serves as a European Training Foundation Ambassador, promoting innovative teaching methods and best practices in the education sector. Additionally, he has founded ExcelWorld OÜ, a platform where he coordinated data-focused events, such as the My Data Summit, and has created Excel and Power BI templates, to help people improve their data presentation and analysis skills.

He is a Microsoft Certified Trainer (MCT) (he was a previous Regional Lead) and Meetup organiser. Busy man – and a very welcome addition to the SumProduct team. Indeed, SumProduct keeps expanding its geographic reach, expertise and enthusiasm for all things data, Excel and modelling. Ilgar will certainly keep us on our toes. Welcome Ilgar!

Beat the Boredom Challenge

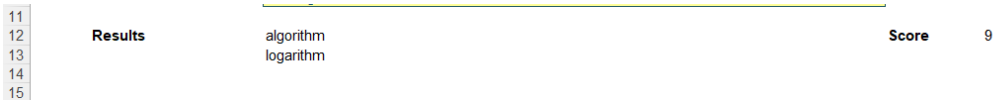
With many of us currently "working from home" / quarantined, there are only so Zoom / Teams calls and virtual parties you can make before you reach your (data) limit. Perhaps they should measure data allowance in blood pressure millimetres of mercury (mmHg). To try and keep our readers engaged, we will 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…*

We dedicate this month's challenge to a certain member of the team who has a penchant for letters and numbers games…

There are certain gameshows that get you to make the longest word from nine [9] letters – and you don't even have to be a postal worker!

Therefore, this month, we have a hybrid Power Query and Excel challenge.

The challenge is to accept up to nine [9] input letters and find the longest word(s) that can be created from them. Only the longest word(s) should be returned in 'Results' and the 'Score' will be the length of the longest word. We have input some letters as an example. For this selection of letters, the 'Results' (for an English dictionary) would be:



[Yes, we know nine letter words normally score double points, but that's not really the "point" here, in case you wish to "point" this out!]

As always, there are some conditions:

- this is a Power Query challenge and Excel challenge – no VBA is required
- you may use a free dictionary source of your choice
- the solution should not be case-sensitive
- if no words can be found, the solution should indicate this.

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

Over to AI

Here at SumProduct we write many articles for many magazines and our authoring contracts have been revised recently to ensure Artificial Intelligence hasn't taken over our duties. We can assure everyone they have not.

But it did get us thinking: *what if we did let AI loose and write a regular article for the newsletter?*

This month, we asked AI what it thought was the current flavour of the month in Excel.

This Month's Prompt: What is the hottest topic in Excel in 2025?

As we venture further into 2025, the capabilities of Excel continue to evolve, cementing its position as an indispensable tool for data analysis and business intelligence. In March 2025, the hotctest topic in the Excel community will revolve around the groundbreaking introduction of *Excel AI Assist*—a powerful integration of artificial intelligence and machine learning features designed to revolutionise the way users interact with data.

WHAT IS EXCEL AI ASSIST?

Excel AI Assist is an innovative suite of tools embedded within Excel, leveraging advanced AI and machine learning algorithms to provide users with intelligent insights, predictive analytics and automated data processing. This feature is set to transform the traditional Excel experience, making data analysis more accessible, efficient and insightful.

Key features of Excel AI Assist include:

- **Intelligent Data Cleaning:** Excel AI Assist can automatically detect and correct data inconsistencies, outliers and errors. It provides suggestions for data normalisation and standardisation, ensuring data integrity and reliability
- **Predictive Analytics:** Users can now access sophisticated predictive models directly within Excel. The AI algorithms analyse historical data to forecast trends, enabling users to make data-driven decisions with greater confidence
- **Natural Language Queries:** Excel AI Assist allows users to interact with their data using natural language. By simply typing questions in plain English, users can receive instant insights and visualisations, making data exploration more intuitive
- **Automated Report Generation:** With AI-powered automation, users can generate comprehensive reports with a single click. Excel AI Assist compiles data, creates charts and writes summaries, saving time and effort
- **Smart Recommendations:** Based upon the data being analysed, Excel AI Assist offers context-aware recommendations for data visualisation, PivotTables and formula adjustments, enhancing the analytical process.

IMPACT ON BUSINESSES AND DATA ANALYSTS

The integration of Excel AI Assist has far-reaching implications for businesses and data analysts. Here's how it stands to make a difference:

- ***Enhanced Efficiency***

  By automating routine tasks such as data cleaning and report generation, Excel AI Assist empowers analysts to focus on more strategic aspects of their work. This leads to increased productivity and faster turnaround times for data analysis projects
- ***Improved Decision-Making***

  The predictive analytics and smart recommendations offered by Excel AI Assist provide decision-makers with actionable insights. Businesses can leverage these insights to identify opportunities, mitigate risks and optimise operations, ultimately driving better outcomes
- ***Accessibility to Advanced Analytics***

  One of the most significant advantages of Excel AI Assist is its ability to democratise advanced analytics. Users without a background in data science or programming can harness the power of AI and machine learning, making sophisticated data analysis accessible to a broader audience.

EXCEL AI ASSIST IN ACTION

To illustrate the capabilities of Excel AI Assist, let's consider a practical example: a retail company analysing its sales data.

Scenario: Sales Forecasting

The retail company has historical sales data for the past five [5] years. Using Excel AI Assist, the data analyst can quickly clean the dataset, removing any anomalies or inconsistencies. Next, the analyst selects the predictive analytics feature to forecast future sales trends.

The AI algorithms analyse patterns in the historical data, considering factors such as seasonality, promotions and economic indicators. Within minutes, Excel AI Assist generates a detailed sales forecast, complete with visualisations and confidence intervals. The analyst can then present these insights to the management team, enabling them to make informed decisions about inventory management and marketing strategies.

FUTURE PROSPECTS

As the Excel AI Assist continues to evolve, we can expect even more advanced features and capabilities to be introduced. With ongoing advancements in AI and machine learning, the potential for Excel to transform data analysis is boundless.

INTEGRATION WITH OTHER MICROSOFT PRODUCTS

One exciting prospect is the seamless integration of Excel AI Assist with other Microsoft products such as Power BI, Azure and Teams. This integration will create a unified ecosystem for data analysis, collaboration and visualisation, further enhancing the user experience.

CONTINUOUS LEARNING

Excel AI Assist is designed to continuously learn from user interactions and feedback. As more users adopt and utilise the tool, the AI algorithms will become more refined and accurate, delivering increasingly relevant insights and recommendations.

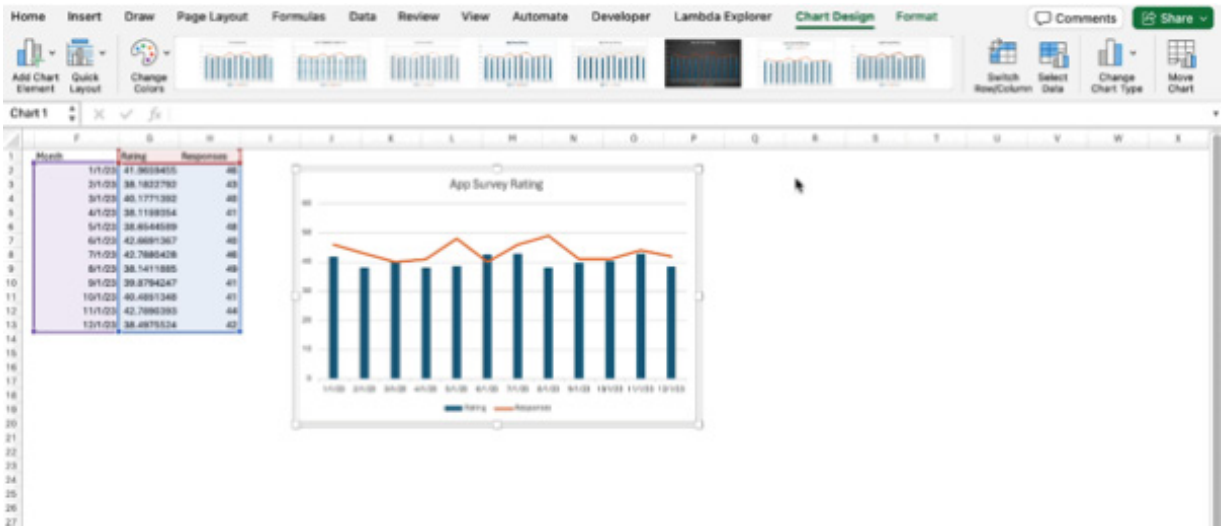
CONCLUSION

In March 2025, Excel AI Assist emerges as the hottest topic in the Excel community, promising to reshape the landscape of data analysis. Its intelligent features, predictive capabilities and user-friendly interface make it a game-changer for businesses and data analysts alike. As we move forward, the potential of Excel AI Assist to unlock new possibilities and drive innovation in data analysis is truly exciting.

What do you think? I am still looking for the term "Excel AI Assist" in Excel! Do you agree AI will get there this year? It will be interesting to see precisely what happens. Send us your comments and ideas for questions to contact@sumproduct.com.

Excel for Mac

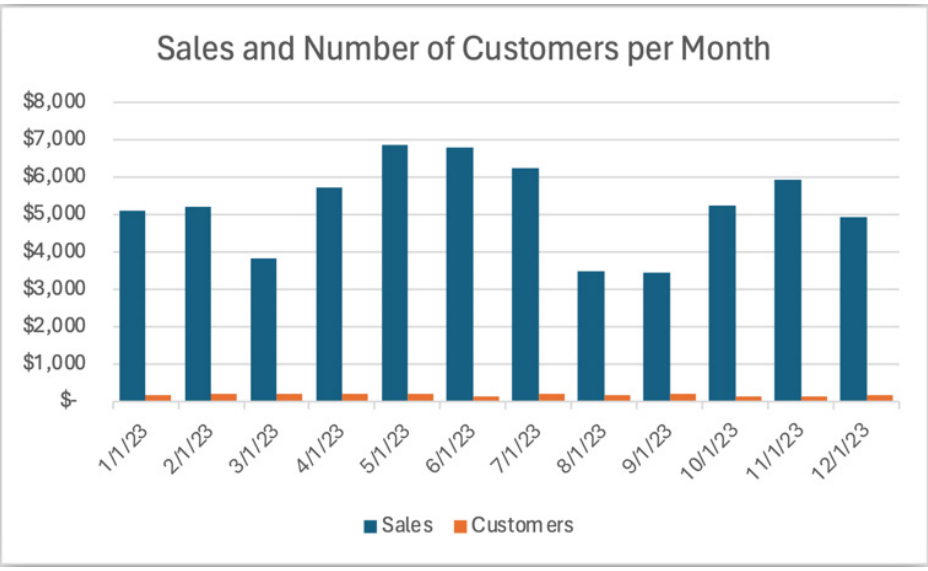
With Steve Kraynak joining the team, we thought we would exploit his knowledge and recant all about Microsoft Excel for Mac. Each month, we'll cover a different topic to help you understand how Excel for Mac is different than Excel for Windows. This month, we show how you can set up Combo charts. It's a little different than on Windows, but once you know how, we think it's very fast and easy.



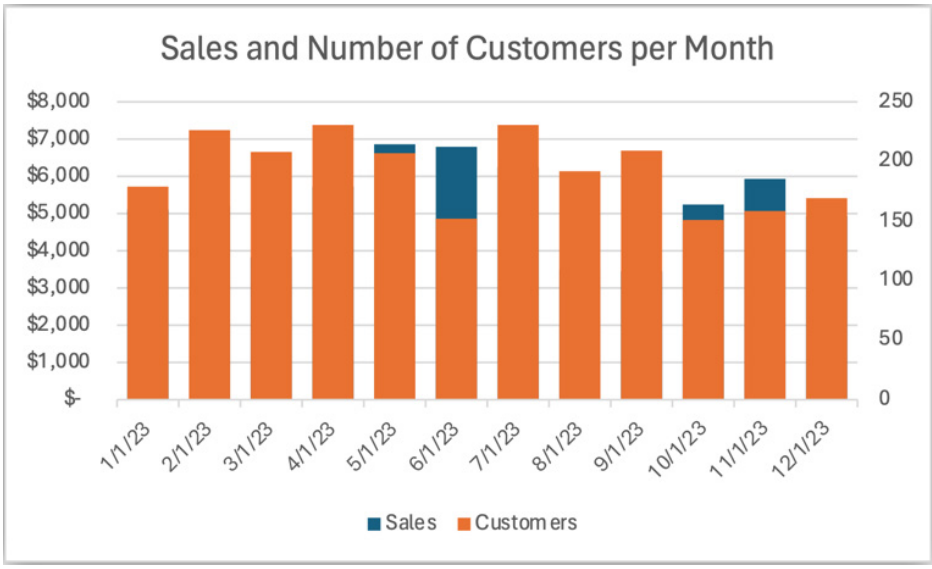
Combo Charts are just charts, but you combine two [2] or more chart types into a single visualisation. For example, you might want to show the volume of sales and the number of customers per month. It would be easy to create two Column or Line charts to show the data, but you might want to combine them to save space and show the relation of the data to each other. This is a good example of when you might want to use a Combo chart.

Using Combo Charts Effectively on Mac

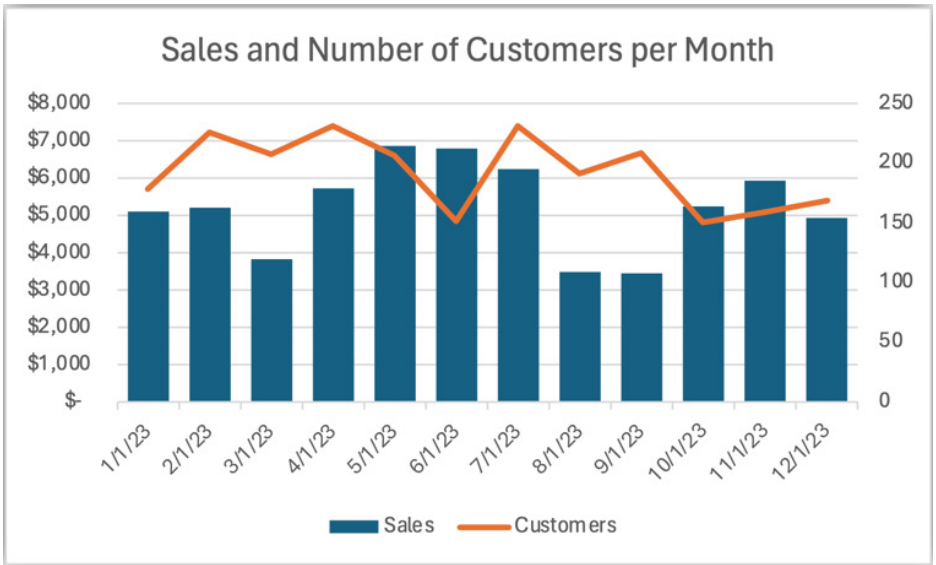
In our example, it might be confusing to show both volume of sales and the number of customers as columns in the same chart. In this case, the number of customers is much lower than the number of dollars, so the scale isn't good to compare the number of customers from one month to the next.



It's better to show the Customers on a secondary axis as shown below. However, that will cause the columns of Customers to Overlap the Sales, making it difficult to see the data.

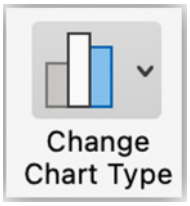


An alternative is to turn this into a Combo chart. For example, you could show the Customers as a line chart and keep Sales as a column chart. Then it's a bit easier to read. The Sales are no longer obscured by the Customers. The use of the line for Customers might give some viewers a hint that it's on a different scale than the dollars on the left axis.

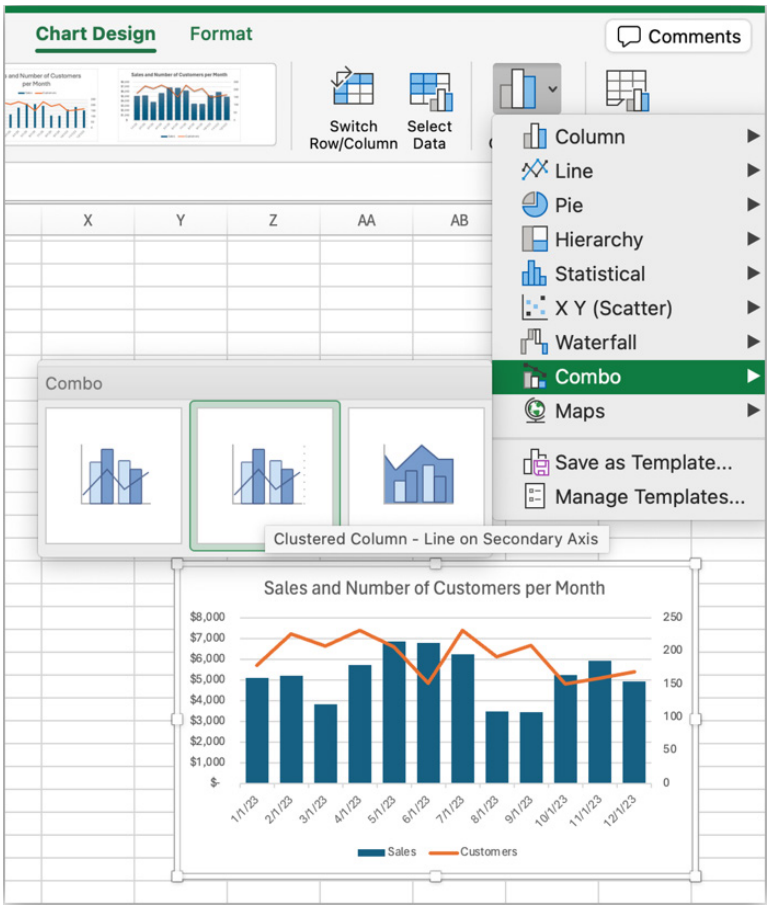


To make a chart this way is easy, but you need to know where to begin:

- Start by selecting the chart
- Then go to the 'Chart Design' tab of the ribbon and choose 'Change Chart Type'

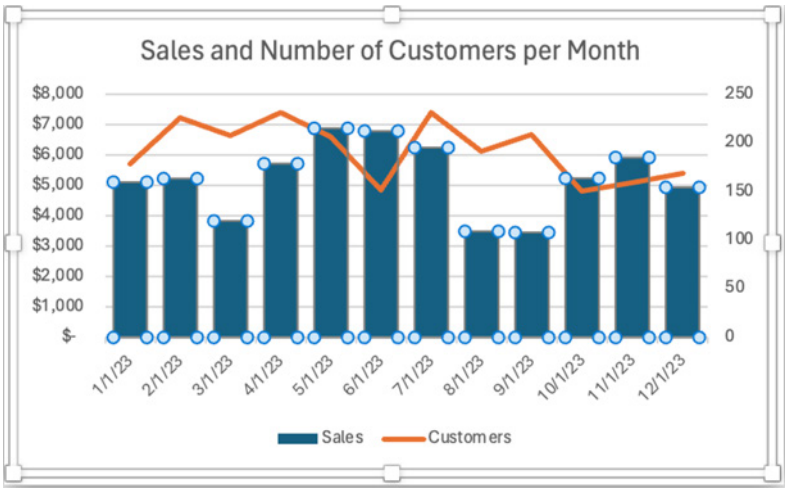


- Go to the Combo item on the menu and choose the desired combination. It shows three [3] options to combine a Column with other types. The second one is a Line on the secondary axis. Click it and your chart will be as we described above:

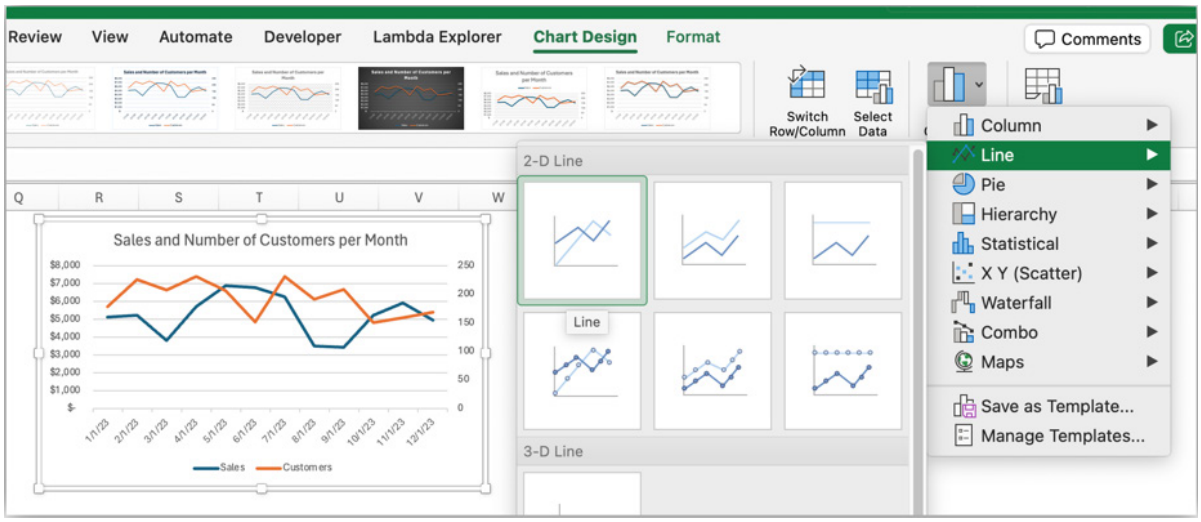


This is nice, but you may want the Sales to be a Line chart and Customers a Column chart. On Windows, this is accomplished using the 'Change Chart Type' dialog. This dialog doesn't exist on Mac, but there's still a way to accomplish the task

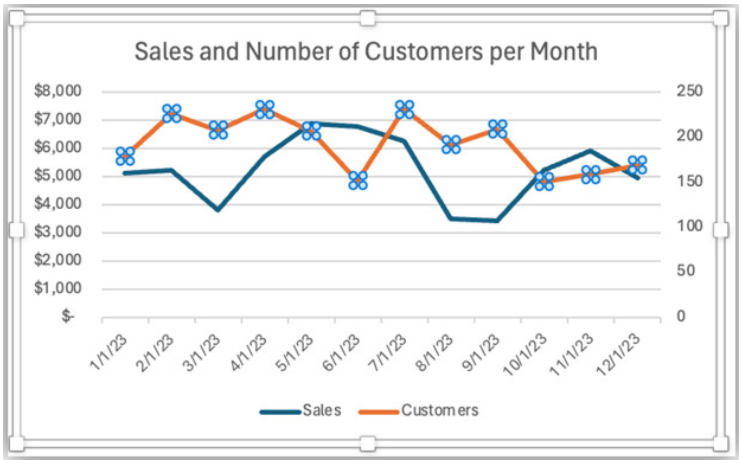
- First, select the series in your chart that you'd like to change. In our example, we'll select the Sales series. We see that it's selected because there are "selection handles" on the corners of the columns in the series. These are the blue dots in the screen shot below:



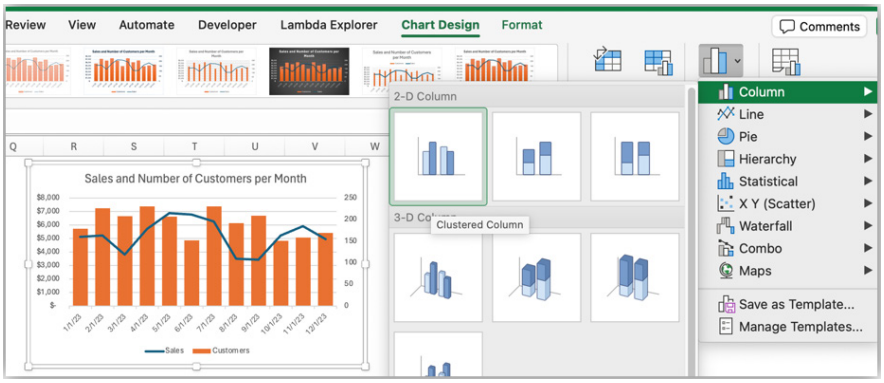
- Next, press the 'Change Chart Type' button on the 'Chart Design' tab of the Ribbon
- Choose Line since we want to set Sales to be a Line chart. At this point, both series are shown as Line charts



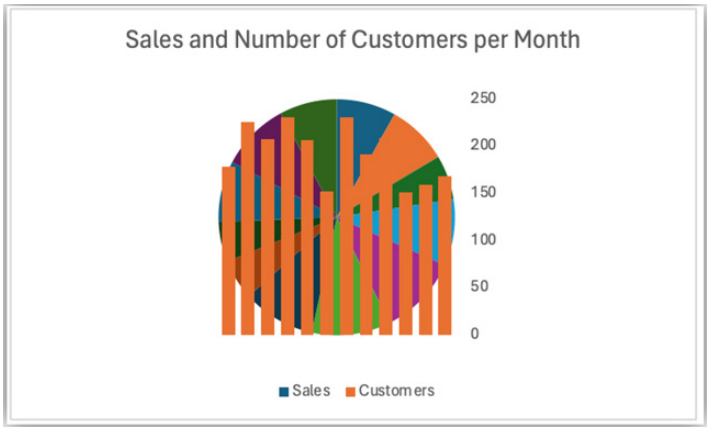
- Next, select the Customers series



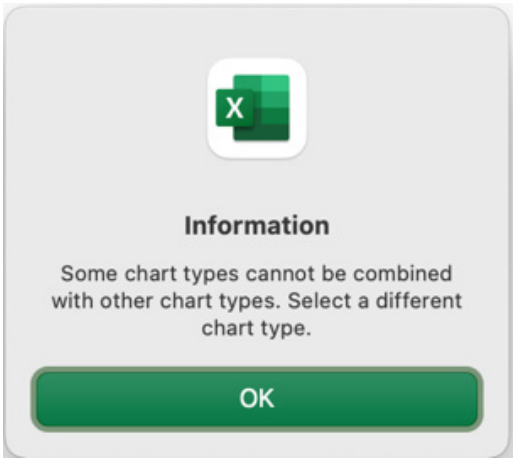
- Again, choose the 'Change Chart Type' menu, but this time, we'll choose **Column -> Clustered Column** as the chart type, since we wanted to swap from how we had it set earlier.



You can try to combine other chart types, but some of them won't work and don't make sense. For example, you can combine a column chart and a pie chart, but it's not much use. We don't advise making a chart like the one shown here. We're showing it as an example of what not to do.



Some chart types can't be combined with each other, and if you try, the entire chart will change to the type that you selected. In some cases, you'll get an error message, indicating that you can't combine the type that you selected.



Key Takeaway

To create a Combo chart in Excel for Mac, you should select the series that you want to change and then pick a new chart type from the 'Change Chart Type' menu.

We'll continue 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 continue reviewing the interaction between charts and macros.

As we continue where we left off in last month's newsletter, here we're going to look at the thought process around creating a macro that would help us to identify a chart's details and present the results to a user. We effectively need to go through the following steps:

1. Define the target / output file, save target details
2. Determine the series details and loop through to extract them
3. Extract out the relevant name / **y**-axis / **x**-axis / bubble size from the formula and clean up.

This time, we're covering the second step in that list. There are several loops that we need to run through to make this macro work:

- We need to look through each sheet (variable: "**sh**")
- We need to find each chart in each sheet (variable: "**ch**")
- We need to find the details of each data series in each chart in each sheet (variable: "**srs**").

Therefore, for each series, we will want to record the sheet name, the chart name, what series it is and the formula that is used to get the data for that series.

```
SheetValue = sh.Name
ChartValue = ch.Name
SeriesValue = srs.Name
FormulaValue = srs.Formula
```

Now, for the purposes of our macro and recording the results, if the data series is blank, for example, then we don't want the macro to record a completely blank cell as it breaks the way that our Named Ranges work to determine how many rows there are. Therefore, we can replace any blanks with a single space and record our results.

```
'If no value, insert a space instead so that Next Item works properly
If SheetValue = "" Then SheetValue = " "
If ChartValue = "" Then ChartValue = " "
If SeriesValue = "" Then SeriesValue = " "
If FormulaValue = "" Then FormulaValue = " "

'Enter values into workbook
OutputWorkbook.Activate
Range("SheetNextItem") = SheetValue
Range("ChartNextItem") = ChartValue
Range("SeriesNextItem") = SeriesValue
Range("FormulaNextItem").Value = "'" & FormulaValue

'Reset view to the target workbook to move to next series
TargetWorkbook.Activate
```

Then, once the values are recorded, we can switch to the new workbook, set the values in place in the appropriate positions based on the named range rules, then go back to the target workbook to look at the next data series and the next chart.

Once the loops are put in place, the whole code looks like this:

```
For Each sh In TargetWorkbook.Sheets

    sh.Activate

    'Look through each chart object
    For Each ch In sh.ChartObjects

        ch.Activate

        'Find each series in the chart object
        For Each srs In ActiveChart.SeriesCollection

            SheetValue = sh.Name
            ChartValue = ch.Name
            SeriesValue = srs.Name

            On Error Resume Next
            FormulaValue = srs.Formula
            On Error GoTo 0
            If FormulaValue = "" And SeriesValue <> "" Then
                FormulaValue = "Excel 2016 chart: series cannot be identified"
            End If
```

```
                'If no value, insert a space instead so that Next Item works properly
                If SheetValue = "" Then SheetValue = " "
                If ChartValue = "" Then ChartValue = " "
                If SeriesValue = "" Then SeriesValue = " "
                If FormulaValue = "" Then FormulaValue = " "

                'Enter values into workbook
                OutputWorkbook.Activate
                Range("SheetNextItem") = SheetValue
                Range("ChartNextItem") = ChartValue
                Range("SeriesNextItem") = SeriesValue
                Range("FormulaNextItem").Value = "'" & FormulaValue

                'Reset view to the target workbook to move to next series
                TargetWorkbook.Activate

            Next srs

        Next ch

    Next sh
```

Essentially, we are selecting a sheet, then selecting the first chart object in that sheet, then going through all of the data series to determine what the details are. Then, once we've finished that chart, we can move on to the next chart, and once we run out, we can move to the next sheet.

One thing to note: **SeriesCollection.Formula** does not work with the new Excel 2016 *(onwards)* chart types. That's why in our final code we have an error trap to check for that.

We're almost done now! The last step is to clean up the output page and work out all of the moving parts in the data series.

Now we need to consider the final step in that list:

- Extract out the relevant name / **y**-axis / **x**-axis / bubble size from the formula and clean up.

So far, we've extracted out details from the charts in a workbook and dropped them into a new workbook that looks something like this:

File: Charts Examples - David.xlsm

Sheet	Chart	Series	Formula
Example - Radar	Chart 1	2017	=SERIES('Example - Radar'!$G$41,'Example - Radar'!$F$42:$F$46,'Example - Radar'!$G$42:$G$46,1)
Example - Radar	Chart 1	2018	=SERIES('Example - Radar'!$H$41,'Example - Radar'!$F$42:$F$46,'Example - Radar'!$H$42:$H$46,2)
Example -	Chart 1	Chart Data Bizuplyz	=SERIES('Example - '!$G$55:$G$56,'Example - '!$F$57:$F$60,'Example - '!$G$57:$G$60,1)
Example -	Chart 1	Chart Data Plumb'n'Stuff	=SERIES('Example - '!$H$55:$H$56,'Example - '!$F$57:$F$60,'Example - '!$H$57:$H$60,2)
Example -	Chart 1	Chart Data L. R. Repeat	=SERIES('Example - '!$I$55:$I$56,'Example - '!$F$57:$F$60,'Example - '!$I$57:$I$60,3)
Example -	Chart 1	Chart Data Harmonic Sonics	=SERIES('Example - '!$J$55:$J$56,'Example - '!$F$57:$F$60,'Example - '!$J$57:$J$60,4)
Example -	Chart 2	Collection	=SERIES('Example - '!$F$57,'Example - '!$G$55:$J$56,'Example - '!$G$57:$J$57,1)
Example -	Chart 2	Warehousing	=SERIES('Example - '!$F$58,'Example - '!$G$55:$J$56,'Example - '!$G$58:$J$58,2)
Example -	Chart 2	Pick & Pack	=SERIES('Example - '!$F$59,'Example - '!$G$55:$J$56,'Example - '!$G$59:$J$59,3)

From here, we need to disentangle the formula to work out what the component parts of the data series are. Luckily, there are a few basic rules:

- he formula always starts off with "=**SERIES**("
- the name of the data series will the first parameter (*i.e*. before the first comma)
- it's always followed by the **y**-axis, the **x**-axis, any bubble size or **z**-axis (where relevant)
- it finishes off with a number indicating where in the list of series items it is.

As a result, we can use formulae to work out what each component is. Therefore, under the **Name** column, we can use the following formula to extract the name:

Name: =IFERROR(MID(D7,FIND("(",D7)+1,FIND(",",D7,FIND("(",D7)+1)-FIND("(",D7)-1),"n.a.")

We can then repeat the steps for the axis items as well:

Y-axis:
=IFERROR(MID(D7,FIND(",",D7,FIND("(",D7)+1)+1,FIND(",",D7,FIND(",",D7,FIND("(",D7)+1)+1)-FIND(",",D7,FIND("(",D7)+1)-1),"n.a.")

X-axis:
=IFERROR(MID(D7,FIND(",",D7,FIND(",",D7,FIND("(",D7)+1)+1)+1,FIND(",",D7,FIND(",",D7,FIND(",",D7,FIND("(",D7)+1)+1)+1)-FIND(",",D7,FIND(",",D7,FIND("(",D7)+1)+1)-1),"n.a.")

Bubble size:
=IFERROR(MID(D7,FIND(",",D7,FIND(",",D7,FIND(",",D7,FIND("(",D7)+1)+1)+1)+3,LEN(D7)-FIND(",",D7,FIND(",",D7,FIND(",",D7,FIND("(",D7)+1)+1)+1)-3),"n.a.")

All of these formulae basically look up the location of the relevant commas and brackets in order to find the start and end points of each of the chart series details. In order to build them into the code, we need something like this:

Range("NameStart").Offset(1, 0).Formula =
"=IFERROR(MID(D7,FIND(""("",D7)+1,FIND("","",D7,FIND(""("",D7)+1)-FIND(""("",D7)-1),""n.a."")"

Range("YAxisStart").Offset(1, 0).Formula =
"=IFERROR(MID(D7,FIND("","",D7,FIND(""("",D7)+1)+1,FIND("","",D7,FIND("","",D7,FIND(""("",D7)+1)+1)-FIND("","",D7,FIND(""("",D7)+1)-1),""n.a."")"

Range("XAxisStart").Offset(1, 0).Formula =
"=IFERROR(MID(D7,FIND("","",D7,FIND("","",D7,FIND(""("",D7)+1)+1)+1,FIND("","",D7,FIND("","",D7,FIND("","",D7,FIND(""("",D7)+1)+1)+1)-FIND("","",D7,FIND("","",D7,FIND(""("",D7)+1)+1)-1),""n.a."")"

Range("BubbleStart").Offset(1, 0).Formula =
"=IFERROR(MID(D7,FIND("","",D7,FIND("","",D7,FIND("","",D7,FIND(""("",D7)+1)+1)+1)+3,LEN(D7)-FIND("","",D7,FIND("","",D7,FIND("","",D7,FIND(""("",D7)+1)+1)+1)-3),""n.a."")"

Once the formulae are in place, we can copy these down, then copy and paste-special as values, so that we don't have the formulae there the whole time; it's much easier to follow if it's hard-coded.

```
Range("NameRange").FillDown
Range("YAxisRange").FillDown
Range("XAxisRange").FillDown
Range("BubbleRange").FillDown

Application.Calculate

Range("NameRange", "BubbleRange").Copy
Range("NameRange", "BubbleRange").PasteSpecial xlPasteValues
```

There's another thing too. If there are no charts in the workbook, the formulae will simply break. In that instance, we might just want a simple message that lets the user know that there are no charts.

```
'If there are no charts, go to the end
If Range("SheetStart").Offset(1, 0).Value = "" Then
    Range("SheetStart").Offset(1, 0).Value = "No charts found."
    GoTo EndMacro
End If
```

Then, all that is left is a bit of macro clean-up to tidy things up and reset screen updating and calculation statuses.

```
Columns("A:H").EntireColumn.AutoFit

On Error GoTo 0

EndMacro:
'Wrap up the macro.

Application.ScreenUpdating = True
Application.Calculation = StartingCalculation
Range("A1").Select

End Sub
```

And that's it – that's our full macro!

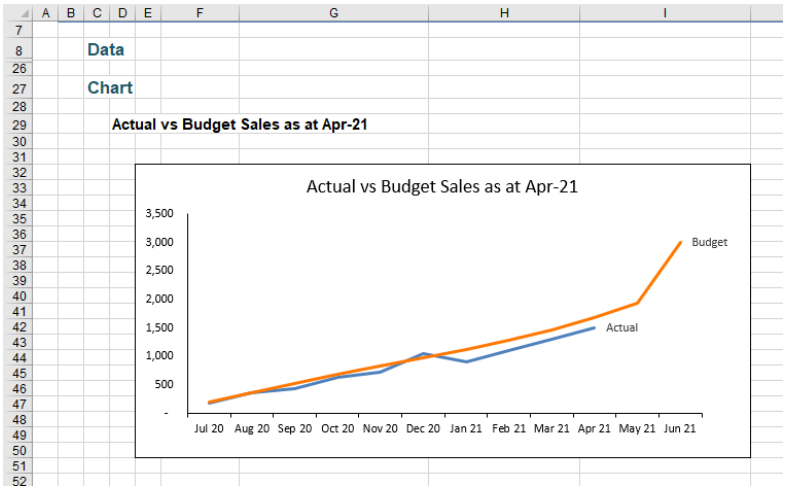
More next month…

Charts and Dashboards

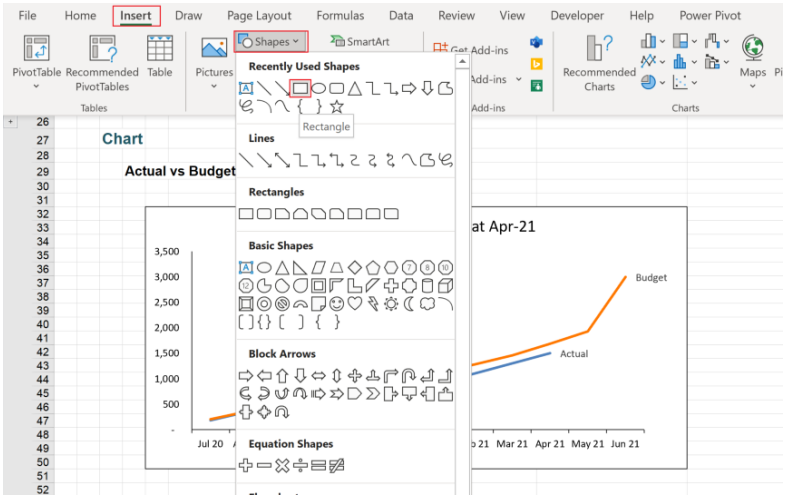
It's time to chart our progress with an introductory series into the world of creating charts and dashboards in Excel. This month, we consider creating a chart title using a text box.

One way to make a chart more presentable is to emphasise its title with a text box, which is straightforward to create in Excel. Here, we will go through a few tips to make the job even easier.

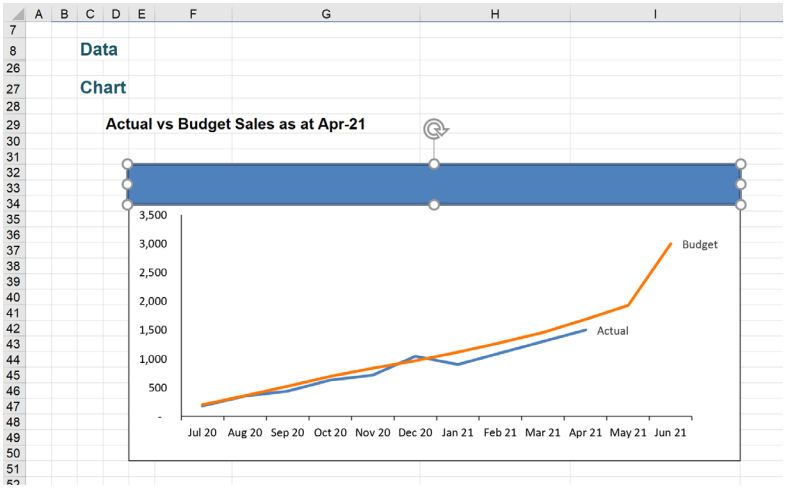
In our example discussed below, there is a Line chart with a chart title in cell **D29** as shown below, which we will use to illustrate our idea:



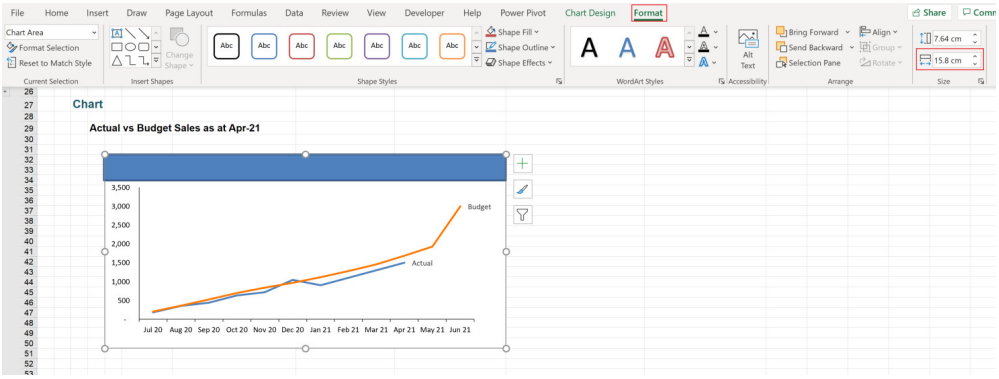
To get a text box, navigate to the Insert tab on the Ribbon and under Shapes, choose a Rectangle (or any other shape that you would like to use).



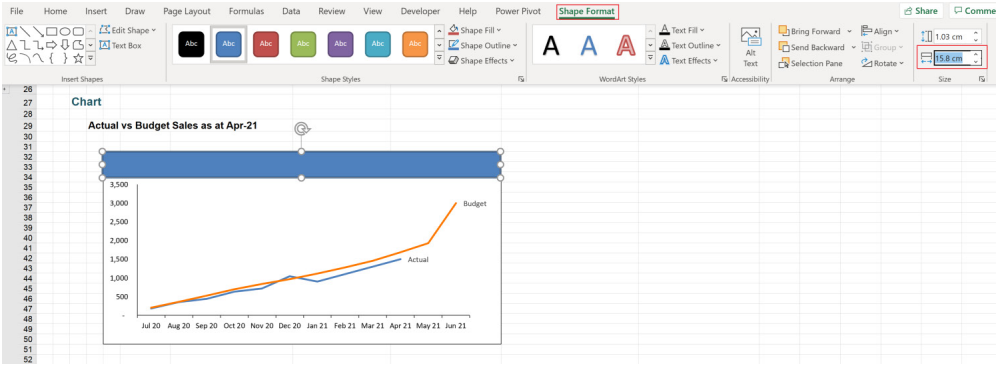
Draw a box to overlap the chart title as shown below:



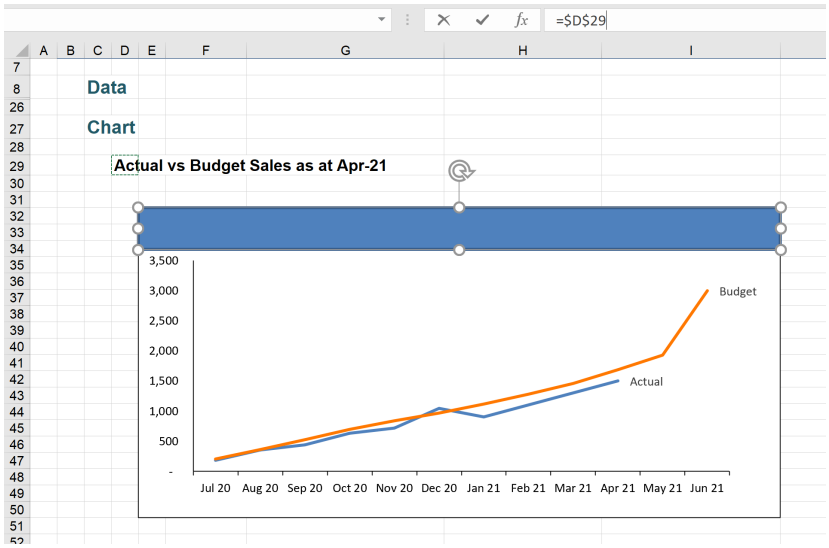
To make sure that the text box has the same width as the chart so that it fits beautifully, click on the chart, go to the Format contextual tab on the Ribbon, check the width of the chart:



Then, click on the text box and under the 'Shape Format' contextual tab on the Ribbon, apply the width under the Size group. Hold the **ALT** key to get the text box to snap to the same grid as the chart.



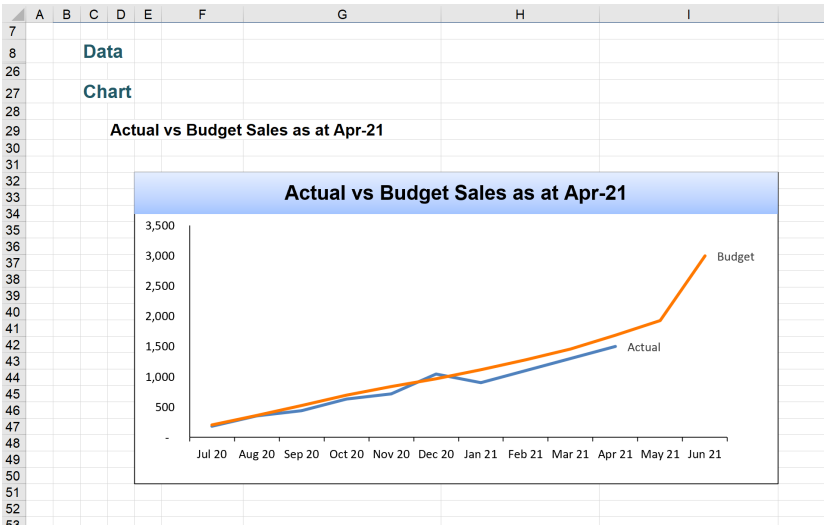
Similar to the way we get the dynamic chart title, click on the text box, in the Formula bar and point it to cell **D29**:



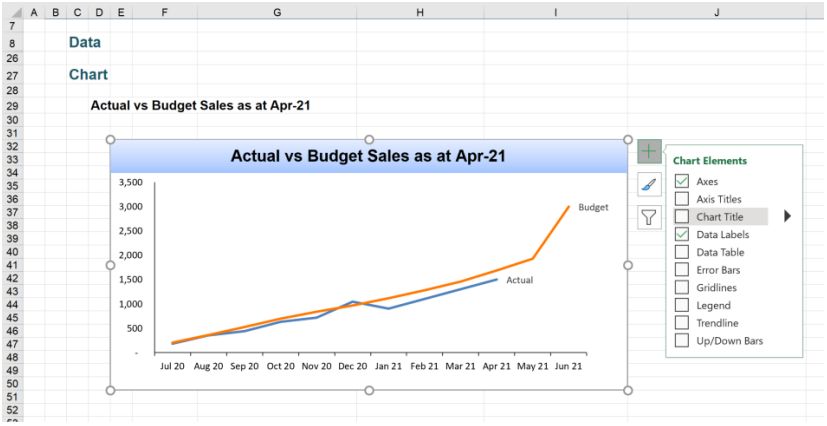
Next, apply a few steps to further format to the text box:

- click on the text box, under the 'Shape Format' contextual tab, change the 'Shape Fill' to the colour of your liking, and remove the 'Shape Outline' and 'Shape Effects (to 'No Shadow')
- under the Home tab, change the size of the text and let the text be in the centre of the text box.

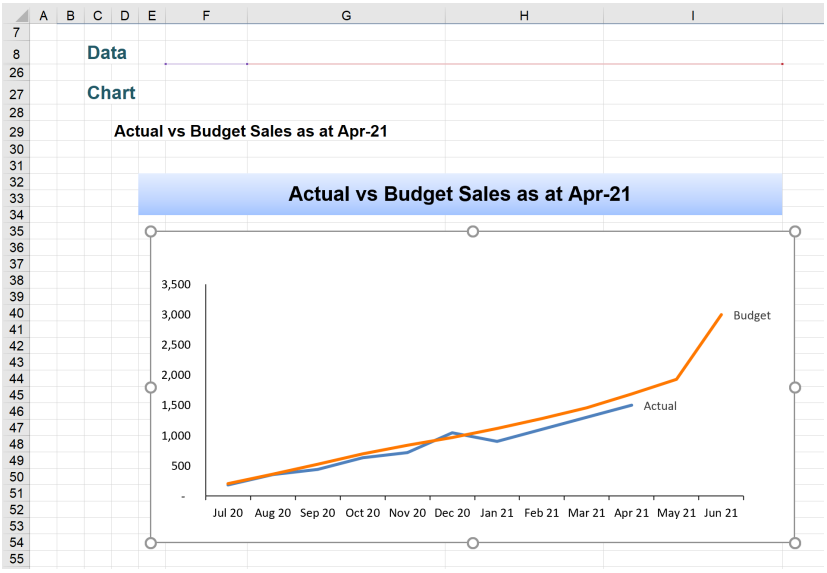
The text box now fits in the chart as shown below:



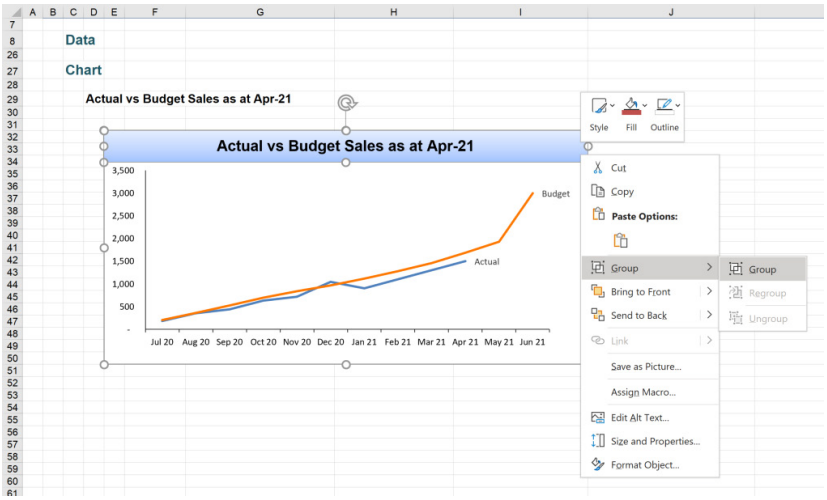
Click on the chart, under the Chart Elements menu and uncheck 'Chart Title' to remove the underlying chart title.



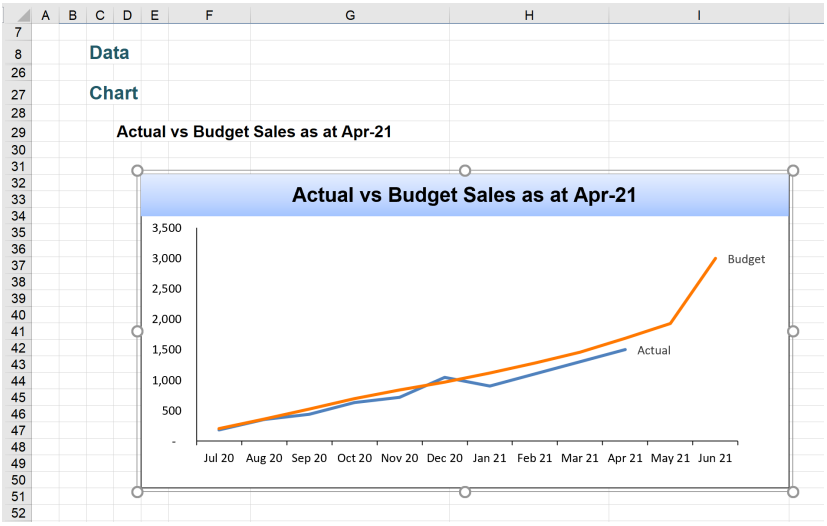
We are almost done. However, if we need to move the chart, it no longer has the text box title on it, since they are not attached together yet.



To attach the chart title text box to the chart, click on the text box, hold the **CTRL** key and click on the chart, then right-click, choose **Group -> Group**.



Now, the chart and the text box are grouped and will move together.



More next month.

Power Pivot Principles

*We continue our series on the Excel COM add-in, Power Pivot. This month, we look at an alternative method of calculating cumulative sums without using the **EARLIER** function discussed in last month's newsletter.*

Last month, we looked at the **EARLIER** function and used it to create a cumulative column. As a reminder, we used the following measure to calculate the cumulative total:

```
=CALCULATE(
        SUM(SaleDataDates[Total Sales]),
        FILTER('SaleDataDates',
        SaleDataDates[Date] <= EARLIER(SaleDataDates[Date])
        )
)
```

```
[Cumul...  fx  =CALCULATE(
                    SUM([Total Sales]),
                    FILTER('SaleDataJul1',
                    SaleDataJul1[Date] <= EARLIER(SaleDataJul1[Date])
                    )
               )
```

	Trans...	Date	Total Sales	Cumulative Total Sales
1	10001	01-Jul-18	$353.00	$353.00
2	10002	02-Jul-18	$446.00	$799.00
3	10003	03-Jul-18	$383.00	$1,182.00
4	10004	04-Jul-18	$283.00	$1,465.00
5	10005	05-Jul-18	$285.00	$1,750.00
6	10006	06-Jul-18	$446.00	$2,196.00
7	10007	07-Jul-18	$323.00	$2,519.00
8	10008	08-Jul-18	$299.00	$2,818.00

The alternative method of calculating a cumulative total involves the use of variables. When writing the new measure, we begin by defining a variable:

= VAR SaleDate = SaleDataDates[Date]

Evaluating this variable alone will result in this calculated column:

```
[VAR Sale Date]  fx  =
                     VAR SaleDate = SaleDataDates[Date]

                     Return
                     SaleDate
```

	Date	Total Sales	Cumulative Sales	VAR Sale Date
1	1/07/2018	$490.00	$490.00	1/07/2018
2	2/07/2018	$475.30	$965.30	2/07/2018
3	3/07/2018	$451.54	$1,416.84	3/07/2018
4	4/07/2018	$465.08	$1,881.92	4/07/2018
5	5/07/2018	$483.68	$2,365.60	5/07/2018
6	6/07/2018	$498.19	$2,863.80	6/07/2018
7	7/07/2018	$518.12	$3,381.92	7/07/2018
8	8/07/2018	$492.22	$3,874.13	8/07/2018

As you can see, the variable evaluates to the current row's Sales date. This is the same evaluation result as the **EARLIER** function in:

EARLIER(SaleDataDates[Date])

Using the Table name as in the example, the next step is to replace the **EARLIER** function segment with the variable:

```
= VAR SaleDate = SaleDataJul1[Date]

Return
CALCULATE(
        SUM([Total Sales]),
        FILTER('SaleDataJul1',
        SaleDataJul1[Date] <= SaleDate
        )
)
```

The logic of the measure remains the same as the measure with the **EARLIER** function, and we have been able to replicate the results without the **EARLIER** function:

```
[Cumulative S... =
VAR SaleDate = SaleDataJul1[Date]

Return
CALCULATE(
        SUM([Cumulative Sales]),
        FILTER('SaleDataJul1',
        SaleDataJul1[Date] <= SaleDate
        )
)
```

	Transaction ID	Date	Cumulative Total Sales EARLIER Funct	Cumulative Sales VAR
1	10001	1/07/2018	$353.00	$353.00
2	10002	2/07/2018	$799.00	$799.00
3	10003	3/07/2018	$1,182.00	$1,182.00
4	10004	4/07/2018	$1,465.00	$1,465.00
5	10005	5/07/2018	$1,750.00	$1,750.00
6	10006	6/07/2018	$2,196.00	$2,196.00
7	10007	7/07/2018	$2,519.00	$2,519.00
8	10008	8/07/2018	$2,818.00	$2,818.00
9	10009	9/07/2018	$3,174.00	$3,174.00
10	10010	10/07/2018	$3,609.00	$3,609.00
11	10011	11/07/2018	$3,927.00	$3,927.00
12	10012	12/07/2018	$4,285.00	$4,285.00
13	10013	13/07/2018	$4,698.00	$4,698.00
14	10014	14/07/2018	$5,044.00	$5,044.00
15	10015	15/07/2018	$5,493.00	$5,493.00
16	10016	16/07/2018	$5,982.00	$5,982.00

Wonder if this method works with numeric columns as well?

```
[Cumulative A... =
VAR ProductID = SaleTbl[Product ID]

Return
CALCULATE(
        SUM([Amount Sold]),
        FILTER('SaleTbl',
        SaleTbl[Product ID] <= ProductID)
        )
```

	Product ID	Price	Amount Sold	Sale Amount	Cumulative Amt Sold	Cumulative Amt Sold VAR
1	10022	$184.00	28	$5,152.00	28	28
2	10023	$307.00	79	$24,253.00	107	107
3	10024	$96.00	75	$7,200.00	182	182
4	10025	$215.00	37	$7,955.00	219	219
5	10026	$20.00	41	$820.00	260	260
6	10027	$462.00	40	$18,480.00	300	300
7	10028	$391.00	21	$8,211.00	321	321
8	10029	$360.00	38	$13,680.00	359	359
9	10030	$220.00	47	$10,340.00	406	406
10	10031	$496.00	56	$27,776.00	462	462
11	10032	$206.00	92	$18,952.00	554	554
12	10033	$164.00	47	$7,708.00	601	601

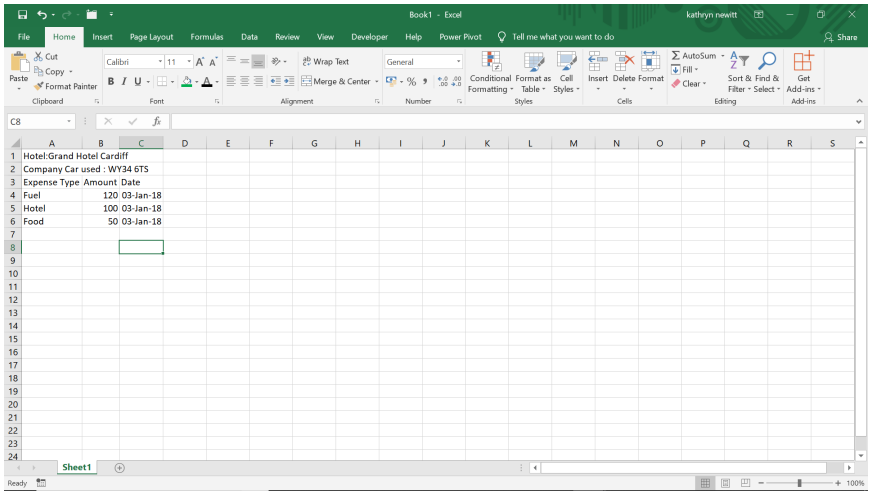
Yes, it does! There you have it: an alternative method of calculating cumulative totals for different columns.

That's it for this month; more next time.

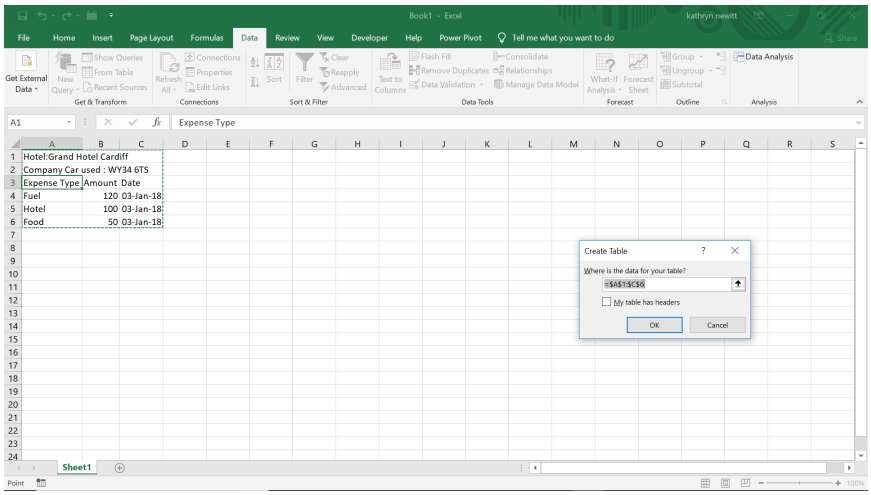
Power Pivot Principles

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 discuss how to remove a given number of rows before retaining useful data.

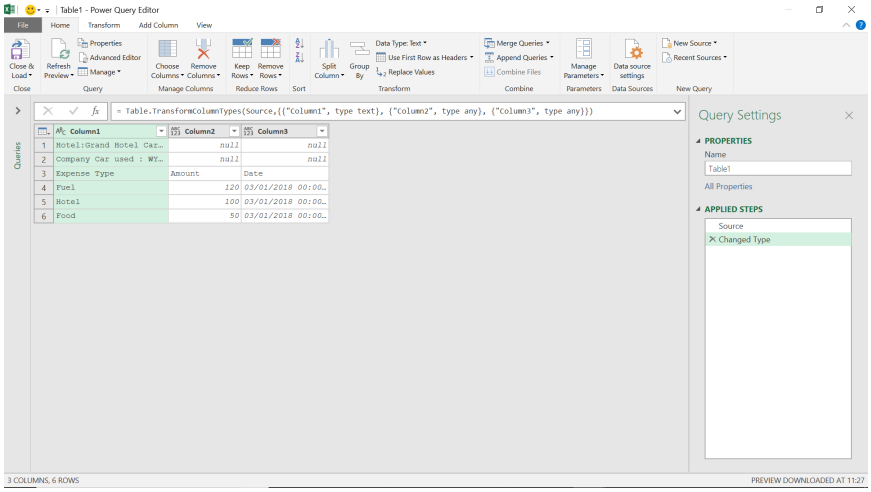
Yet again, our fictional salespeople have been imaginative with their expense forms! We have a form from John, who has decided to add some information that is not necessary for our calculations.



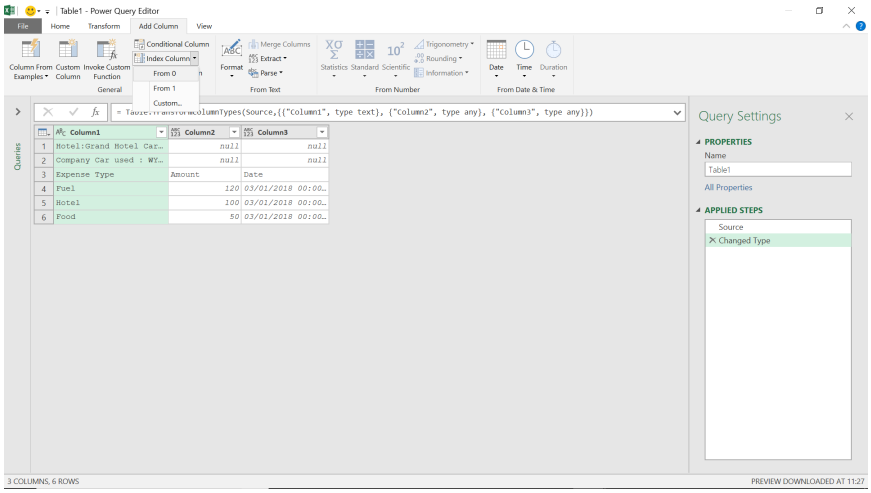
All we need is the expense data – we don't need to know what car and hotel are involved. However, John likes to use the sheet for any notes that he wants to write and the anount of comment lines may vary. Therefore, our first step is to create a new query by using the 'From Table' option on the 'Get & Transform' section on the 'Data' tab:



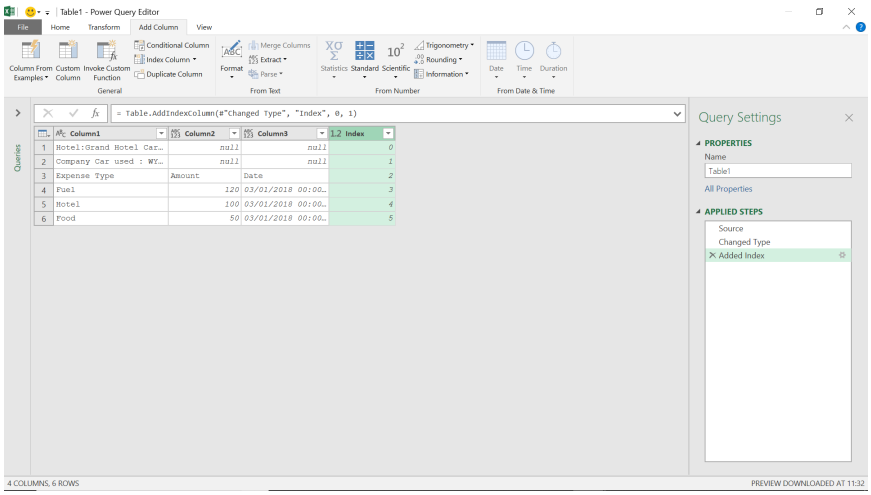
The area of our table is correctly identified, and though we do have headers, they are not at the top, so we leave the 'My table has headers' box unchecked.



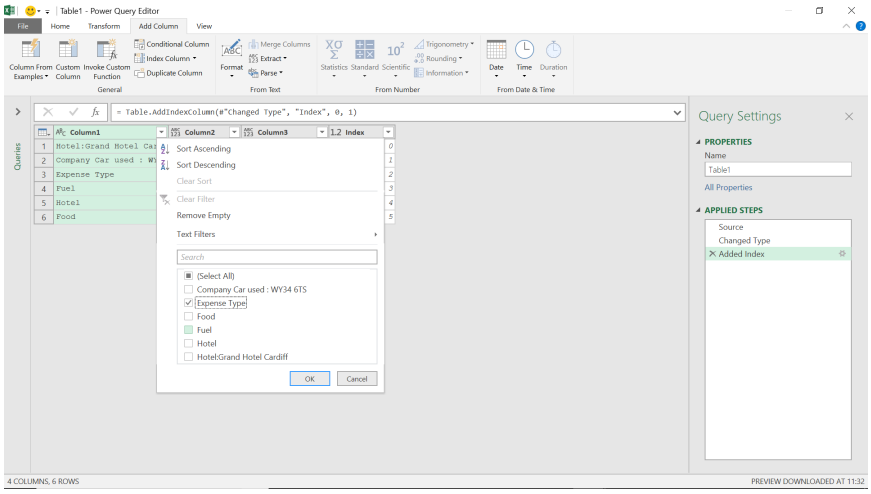
The transformation we are about to perform must work for any number of 'useless' rows at the top, so we will need to do more than delete them manually. First, we determine where our header row is. We are going to create an **Index** column from the 'Add Column' tab.



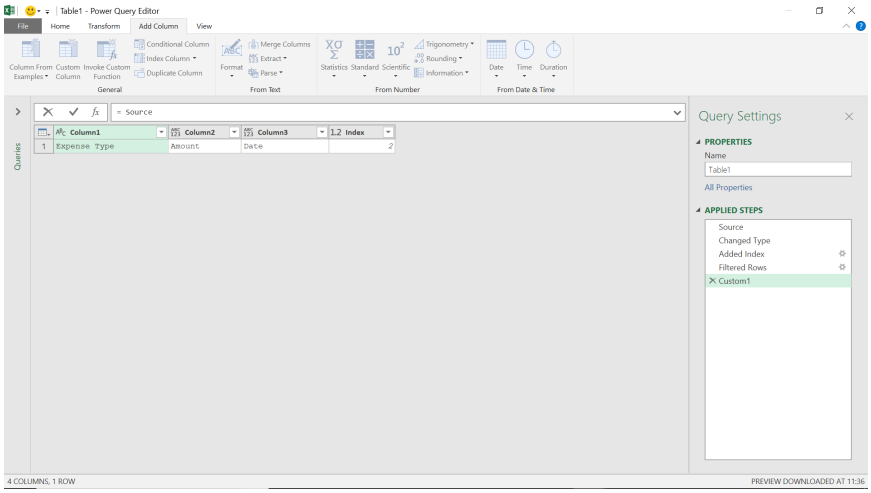
Let’s choose to start at zero [0] and create a new column:



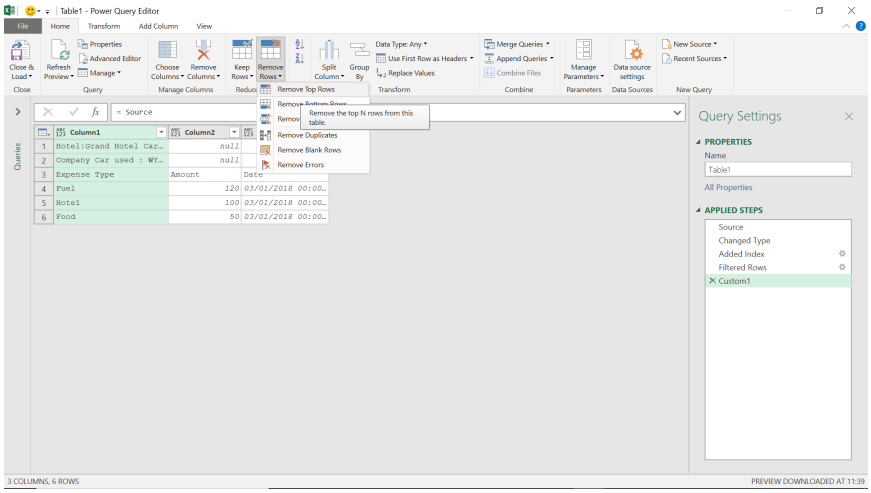
We filter the data to give the index number of our header row.



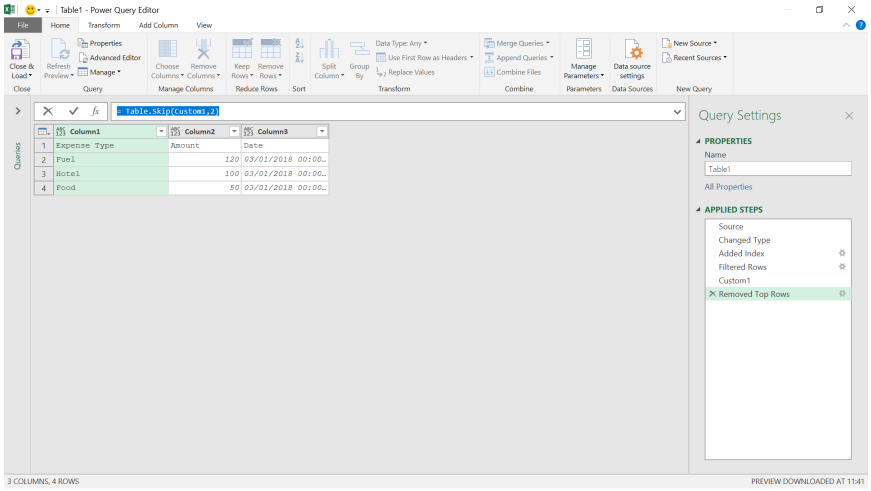
This gives us just the header row. Since Power Query does not change the original data on the Excel worksheet, we may create another source step which points back to our original data.



Now, we delete our first two [2] rows. This time, we will manually delete two [2] rows, but next time it could be more or less than that, so this is just the starting point so get at the **M** code we will need to use.



From the 'Remove Rows' option on the 'Home' tab, we choose to 'Remove Top Rows', and for now, we will choose to delete two [2] rows.



We need to amend the **M** code for this step to cope with any number of rows.

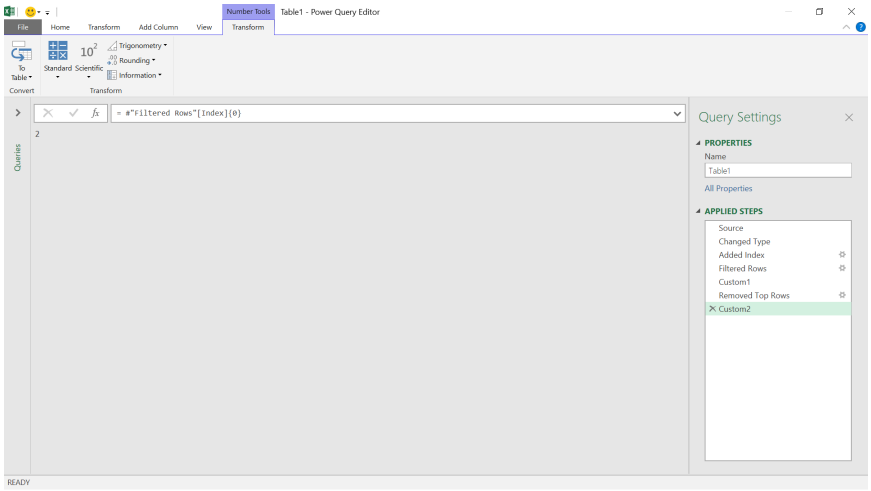
= Table.Skip(Custom1,2)

This is using the **M** function **Table.Skip()**. This function returns a table that does not contain the first row or rows of the table.

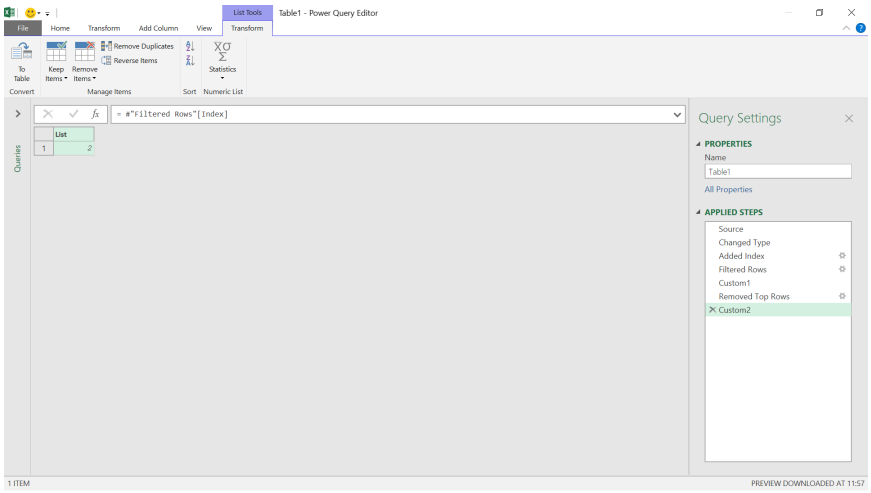
Table.Skip(table as table**, optional countOrCondition** as any**)** as table

where **table** is the table to modify and the optional parameter **countOrCondition** tells the function how many rows to delete.

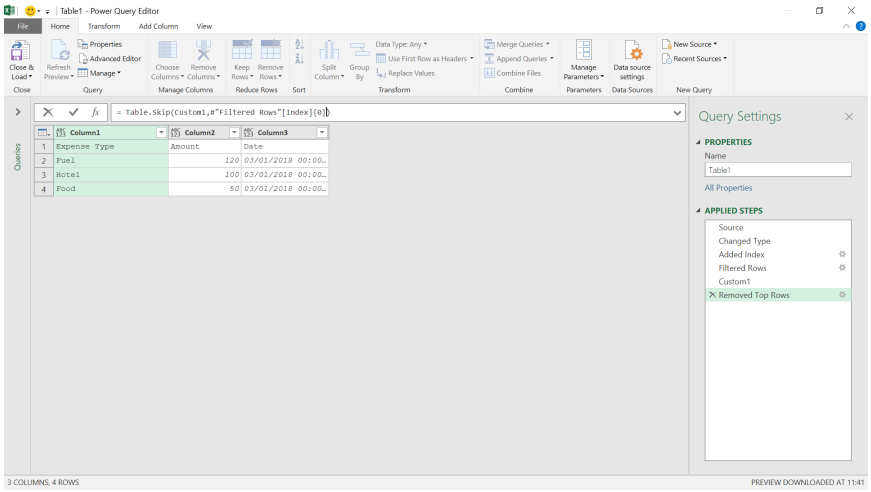
Currently, we have a count (2), but we need to amend the **Table.Skip()** step to use a condition instead, and that condition will be based on the location of my header row. We need to incorporate the '**Filtered Rows**' step and use that instead of the value two [2]. In order to get the index from the 'Filtered Rows' step, we can extract a column from it.



We have extracted the value in the **Index** column on the first row. To demonstrate why we need the {0}, we can use the function without it to see what we get:



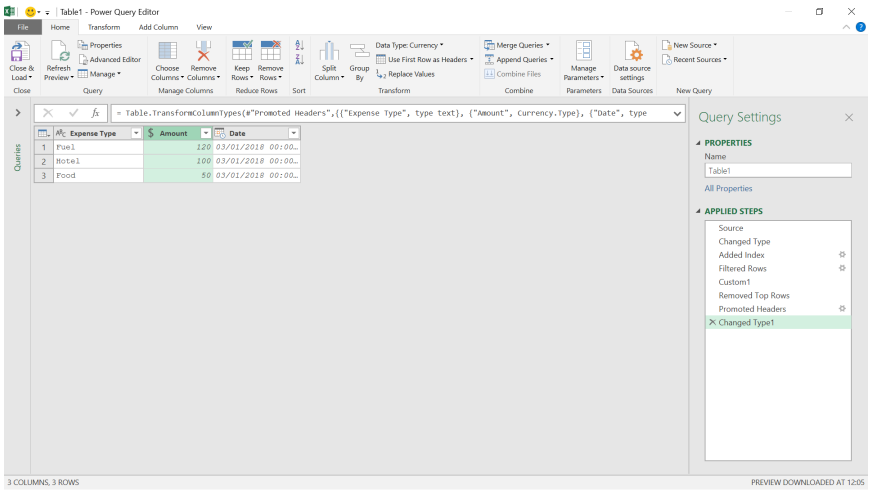
We get a list instead of a single value. Using **{0}** will give us the first value in the list. Now we have shown how to get the index from the filter step, we may incorporate this into the **Table.Skip()** function:



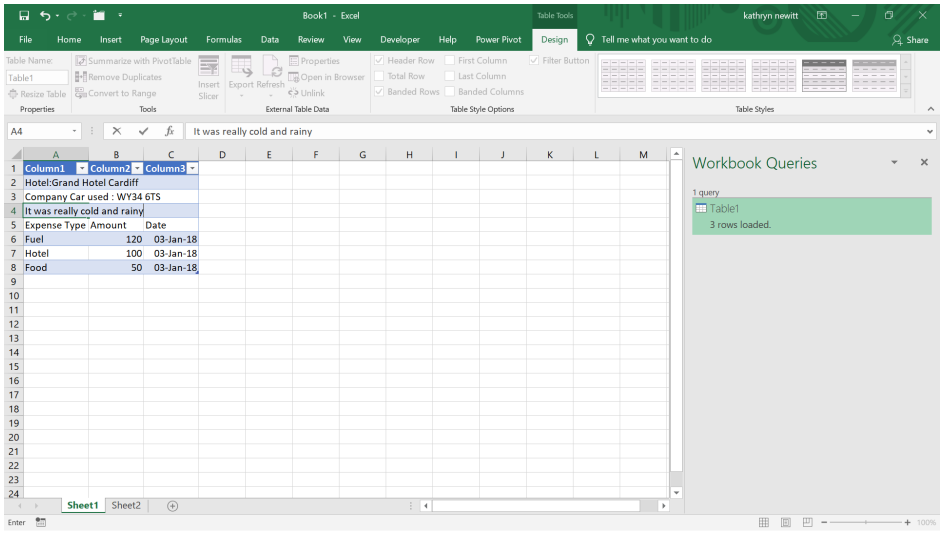
Our step is now

= Table.Skip(Custom1,#"Filtered Rows"[Index]{0})

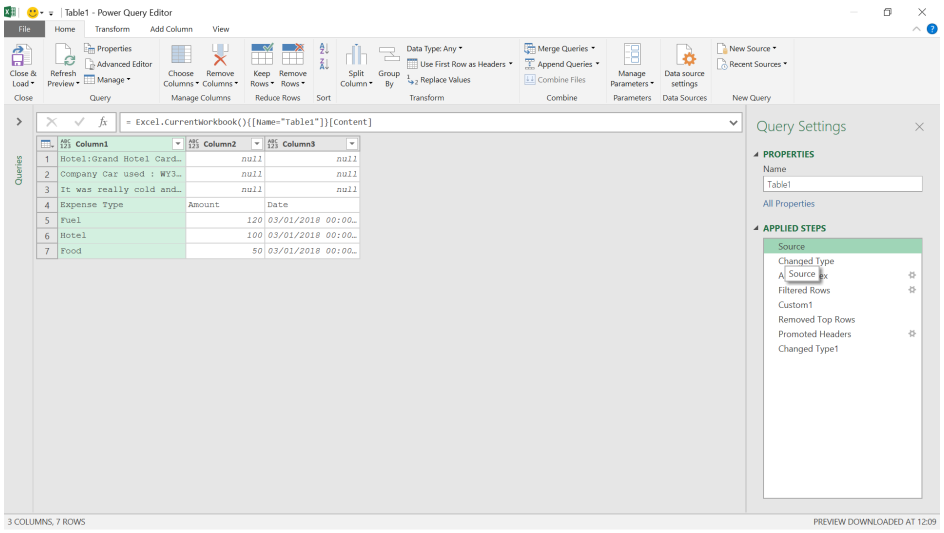
We finish tidying the data ready to test the query.



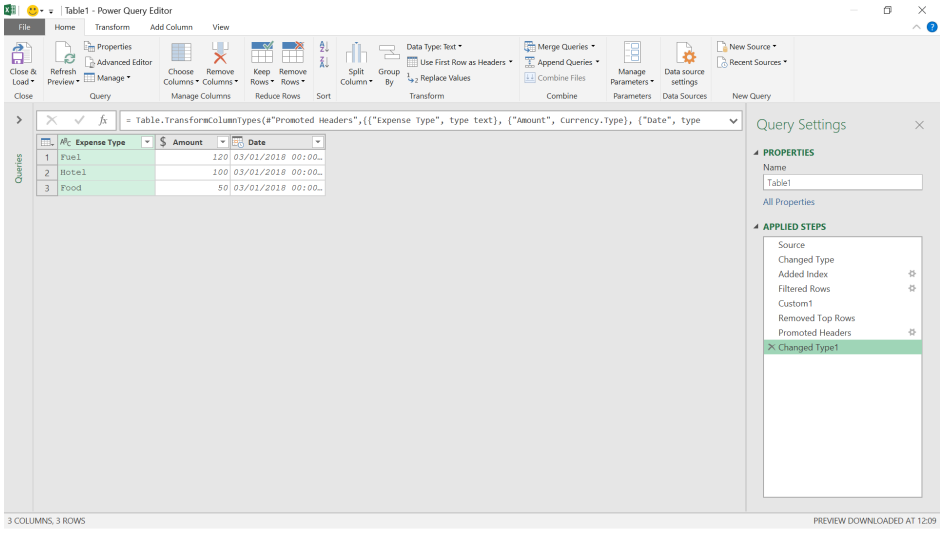
In order to test our query, we go back to the Excel worksheet and add another line to John's extra information.



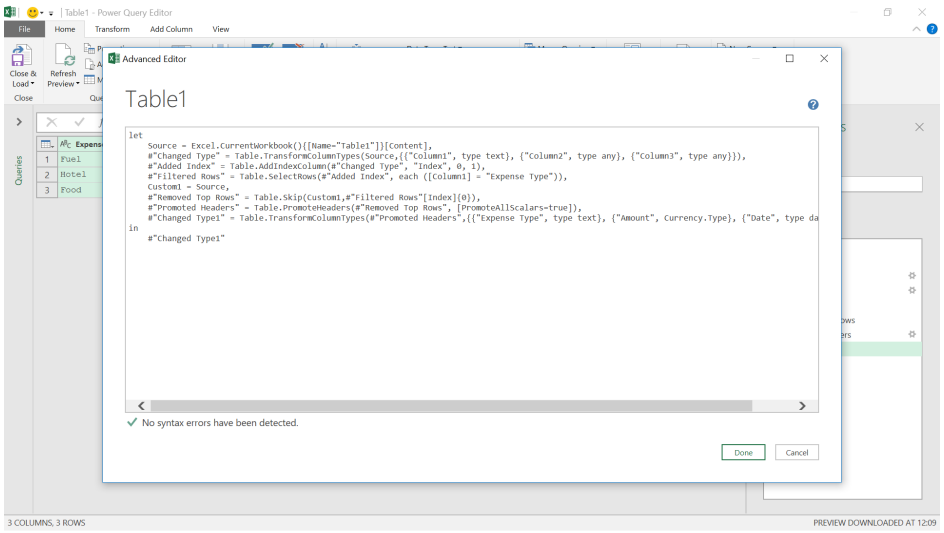
We refresh the query to see what happens.



The source step shows the extra line of information:



Our final step has removed all the extra data correctly.



Looking at the **M** code in the Advanced Editor, we could make a few more changes to make our code more efficient. We have two [2] '**Changed Type(n)**' lines that could be combined and we could also remove the '**Custom1**' step and point directly at the source in '**Removed Top Rows**'. However, we know that no matter how much information John decides to share, we may remove it with this query.

Until next month.

Power BI Updates

This month's updates see improved modelling performance for live editing of semantic models in Direct Lake mode within Power BI Desktop. This will apparently result in at least a 50% improvement in each modelling change.

Additionally, the fully interactive Explore feature is unveiled for Copilot visual answers. This is available for both read and edit modes of a report. Explore allows you to filter, sort or swap fields and change visual types easily for an *ad hoc* exploration. Further, the OneLake catalog is now integrated into the Power BI application within Microsoft Teams, facilitating collaboration and data sharing amongst team members.

The full list of updates is as follows:

General

- Upgrade Power BI Desktop to 64-bit version
- Upgrade Power BI Desktop to February 2025 version

Copilot and AI

- Copilot in the Power BI mobile apps: now available on iPads and Android tablets (Preview)
- Updated section: 'How Copilot arrived at this'
- Explore from Copilot visual answers

Reporting

- Conditional formatting support for visual calculations (Preview)
- Publish to Web support for the Azure Maps visual (coming soon)

Modelling

- Live edit of semantic models in Direct Lake mode with Power BI Desktop – updates (Preview)

Data Connectivity

- Improvements in Snowflake connector (Preview)
- Use the official ODBC driver to connect to Vertica database (Preview)

Visualisations

- Drill Down Donut PRO by ZoomCharts
- Heatmap by Powerviz
- Maximise data efficiency with accoMASTERDATA writenback for Power BI

Other

- The OneLake catalog is now available in Microsoft Teams
- Monitoring Hub enhancement for semantic models.

Let's look at each in turn.

Upgrade Power BI Desktop to 64-bit version

The 32-bit version of Power BI Desktop will no longer be supported after 30 June 2025. You need to ensure you upgrade to the 64-bit version Power BI Desktop to keep receiving updates and support.

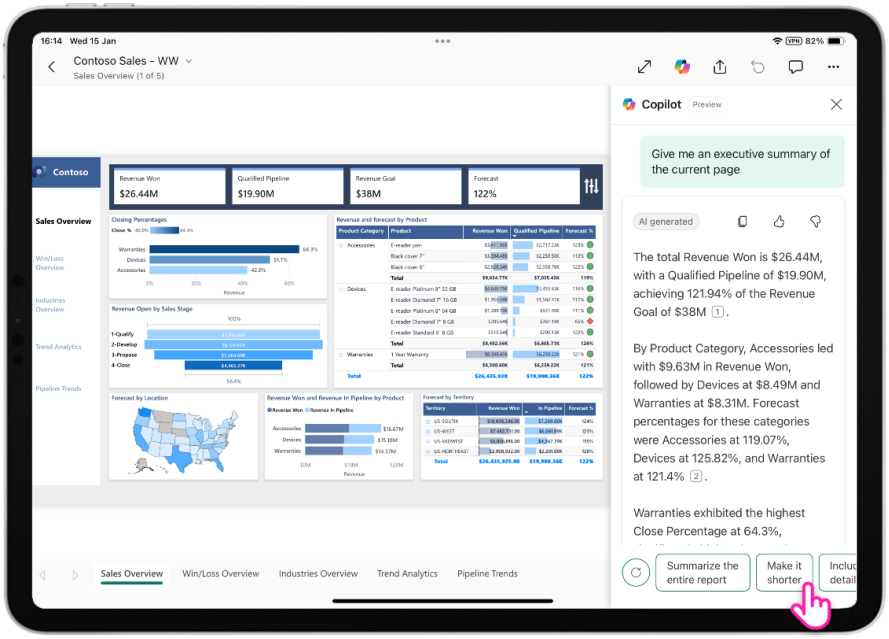
Upgrade Power BI Desktop to February 2025 version

And that's not all… To continue using Report view Copilot chat pane, you will also need to upgrade to the February 2025 version. 'Report view copilot chat' pane might not work as expected for versions before February 2025 after 30 April 2025.

Copilot in the Power BI mobile apps: now available on iPads and Android tablets (Preview)

Copilot support is now available in the Power BI Mobile applications on iPads and Android tablets, extending the functionality of AI to more mobile devices. Introduced for phones several months ago, Copilot in the Power BI Mobile apps empowered users to quickly analyse data, gain insights and make informed decisions on the go. Now, the same powerful capabilities have come to iPads and Android tablets, making it easier to explore your data anytime, anywhere. Roll out the martinis.

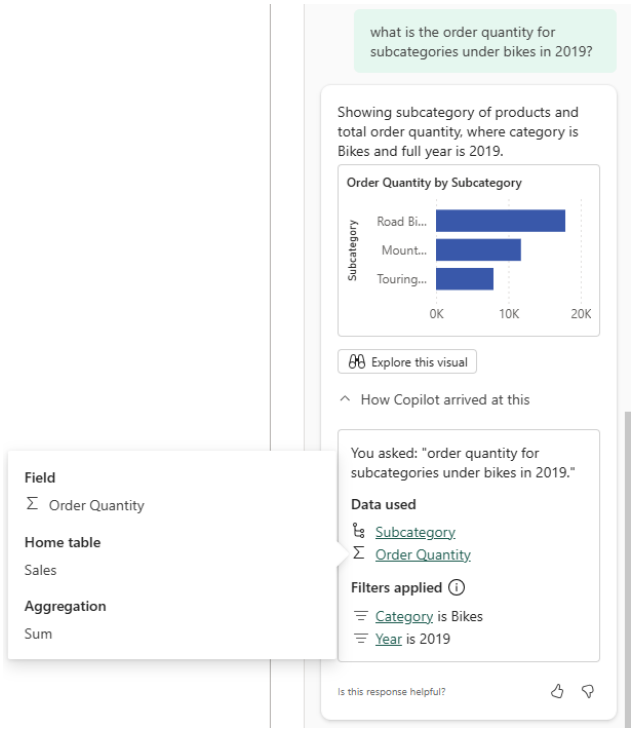
Getting started with Copilot on your mobile app is easy: simply tap the Copilot button located in the report header (for reports that meet Copilot requirements in Power BI). From there, you can choose whether to receive a summary or uncover insights. Copilot will deliver a response based on your request, which you can then copy, share or continue interacting with. Suggestions at the bottom of the screen help you refine your request or create new ones, making it simple to explore your data further.



Updated section: 'How Copilot arrived at this'

Microsoft has responded to the feedback that users would like more transparency regarding how Copilot generates visual answers. This month sees the previously-named 'show reasoning' section now called 'How Copilot arrived at this'.

When receiving a visual answer from Copilot, you may expand this new section to assess what data (fields / measures) and filters were used to generate the answer. You can also click on the field to see more information like the aggregation or home table. This is especially helpful when there could be multiple fields with the same name, like having multiple Dates in your data, verifying Copilot picked the correct one.



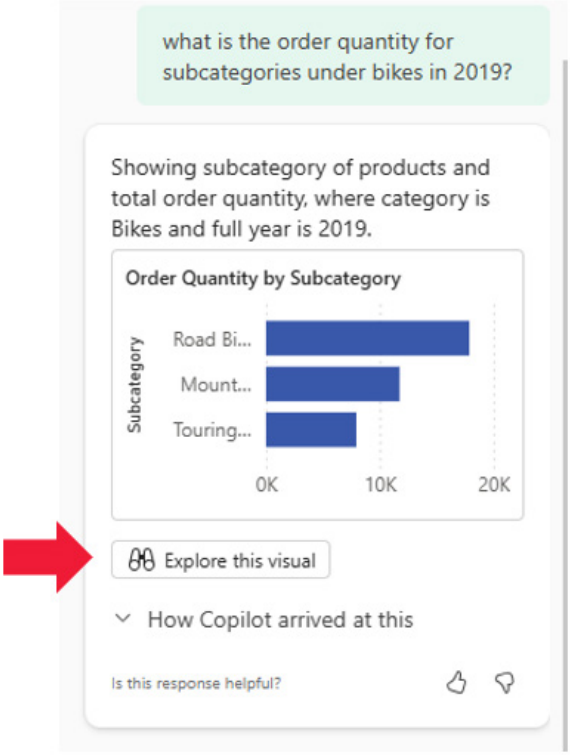
As a reminder, Copilot does not currently incorporate existing filters from the report when generating a visual answer. This list only contains filters that were used at the time of generating the answer.

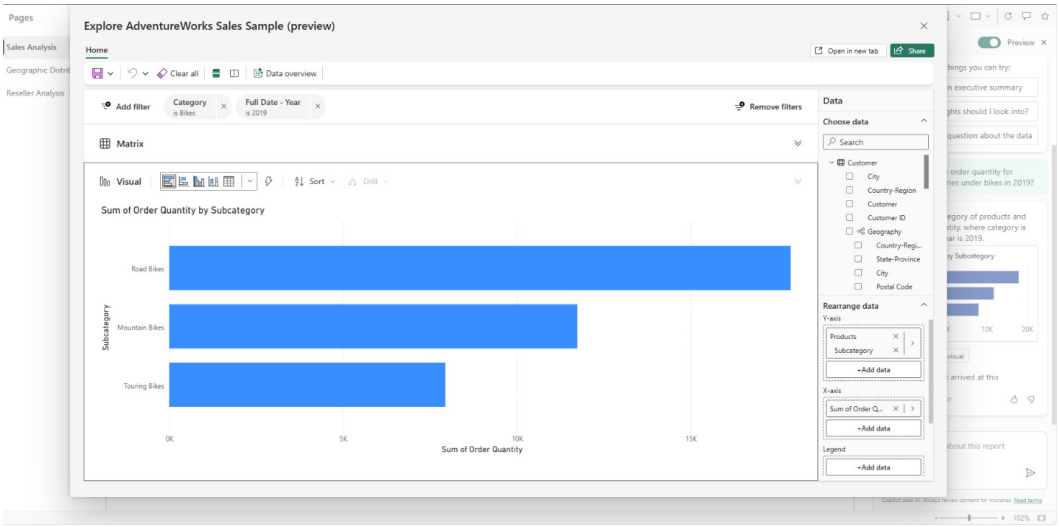
This is a step towards increasing transparency in how Copilot understands and generates responses. More soon, we think.

Explore from Copilot visual answers

Sometimes in Copilot, you may wish to do more with the visual answer, such as drill down, filter it further or maybe even swap it to a different visual type. Until now, the only way to take action on the Copilot generated visual was to add it to the page. This of course was limited to 'edit mode' only. However, this month Microsoft has replaced the previous 'expand view' action with a fully interactive Explore feature. This may be used in both read and edit modes of a report.

With Explore, quick actions are now possible from Copilot. Simply open the Explore experience to filter, sort, swap fields or change visual types easily for an *ad hoc* exploration.





You can choose to save the new explore visual to a report or an exploration to return to. It should be noted that saving the new visual to the original report is not yet available and that Explore does have certain limitations: some users may still see the 'expand view' button even if requirements to use Explore are not met.

Conditional formatting support for visual calculations (Preview)

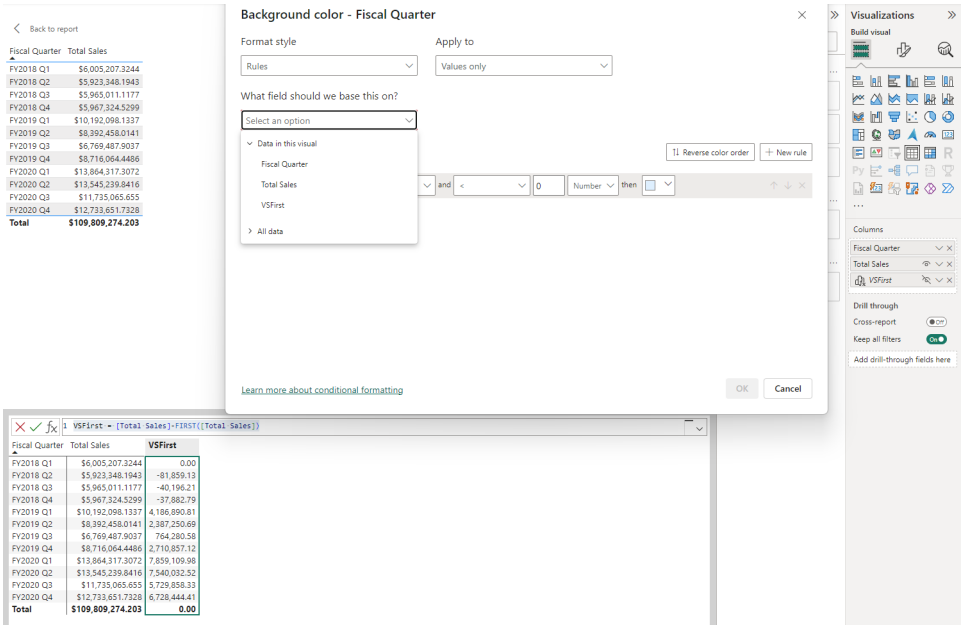
This month finally sees greater support for conditional formatting. You may now use visual calculations to set up conditional formatting rules for columns and measures on your visuals. Also, you can now apply condition formatting rules to visual calculations.

Once you open the 'Conditional formatting' dialog, you will notice there is a new section that shows the data that's on your visual. This section includes any column, measure and visual calculations on your visual and will include any hidden fields as well.

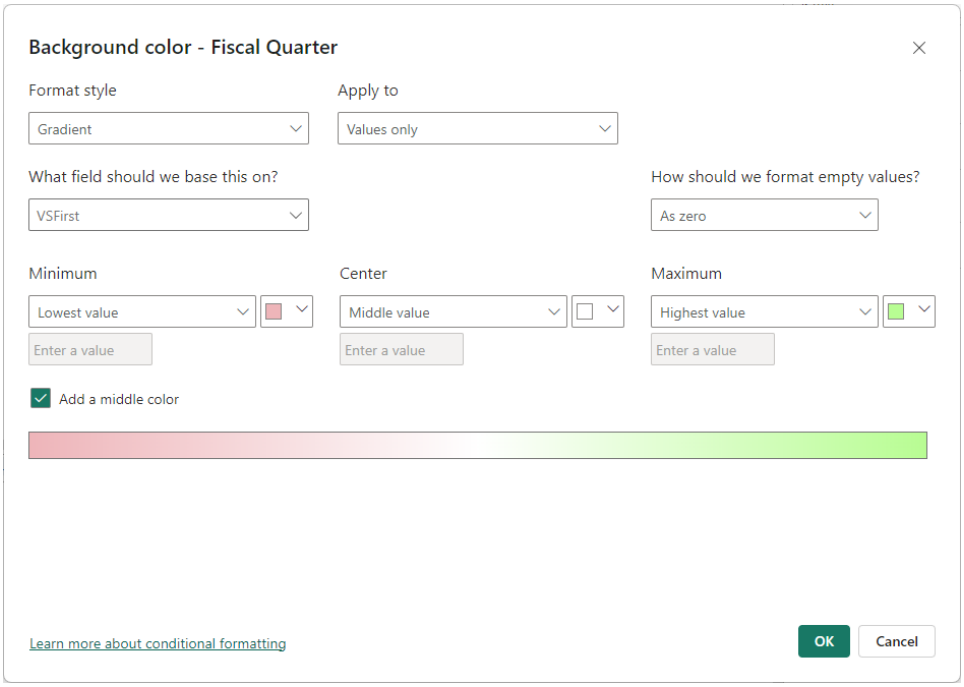
For example, here we have a simple visual that shows sales by quarter using a **Total Sales** measure and the **Fiscal Quarter** column. We have added a visual calculation that for each quarter calculates the difference with the first quarter:

VSFirst = [Total Sales] - FIRST([Total Sales])

We have hidden the **VSFirst** calculation but we will use it to set a conditional background colour for the fiscal quarter. In the 'Conditional formatting' dialog, we see the data in this visual section:



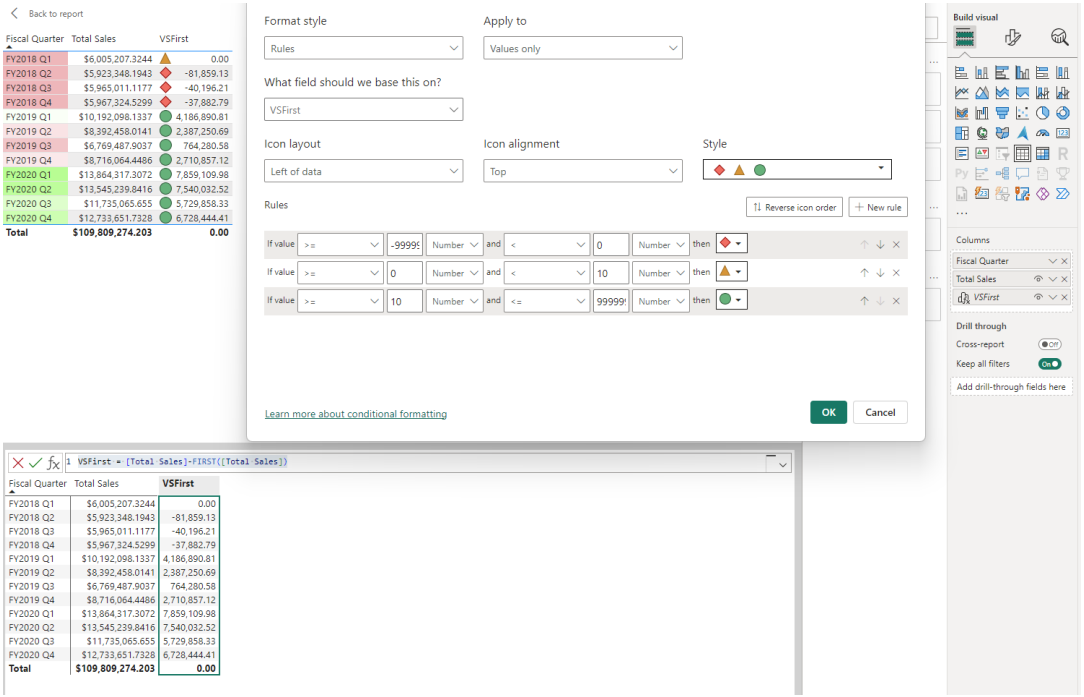
All your model data is still accessible, it is in the 'All data' section. Next, select the **VSFirst** visual calculation and set up the rule as usual:



The conditional formatting is applied:

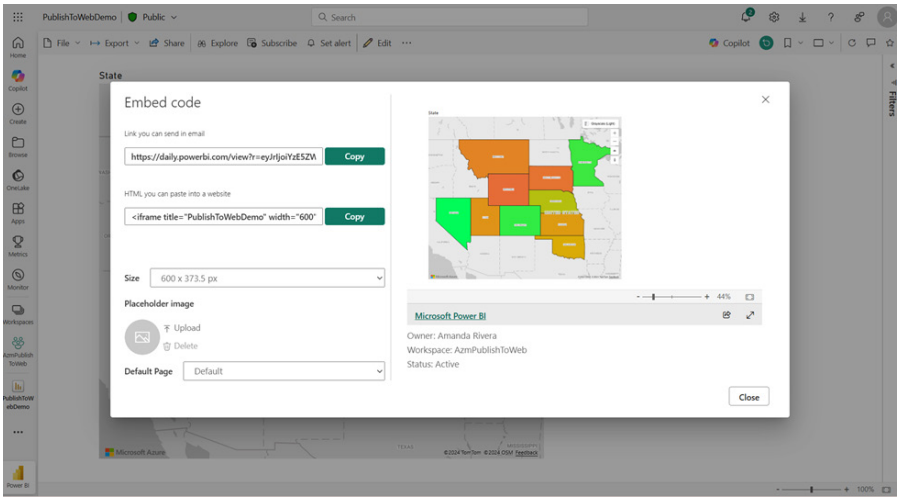
Fiscal Quarter	Total Sales
FY2018 Q1	$6,005,207.3244
FY2018 Q2	$5,923,348.1943
FY2018 Q3	$5,965,011.1177
FY2018 Q4	$5,967,324.5299
FY2019 Q1	$10,192,098.1337
FY2019 Q2	$8,392,458.0141
FY2019 Q3	$6,769,487.9037
FY2019 Q4	$8,716,064.4486
FY2020 Q1	$13,864,317.3072
FY2020 Q2	$13,545,239.8416
FY2020 Q3	$11,735,065.655
FY2020 Q4	$12,733,651.7328
Total	**$109,809,274.203**

As mentioned, you can also format the visual calculations themselves, as done here with the same visual calculation that was made visible to show the result:

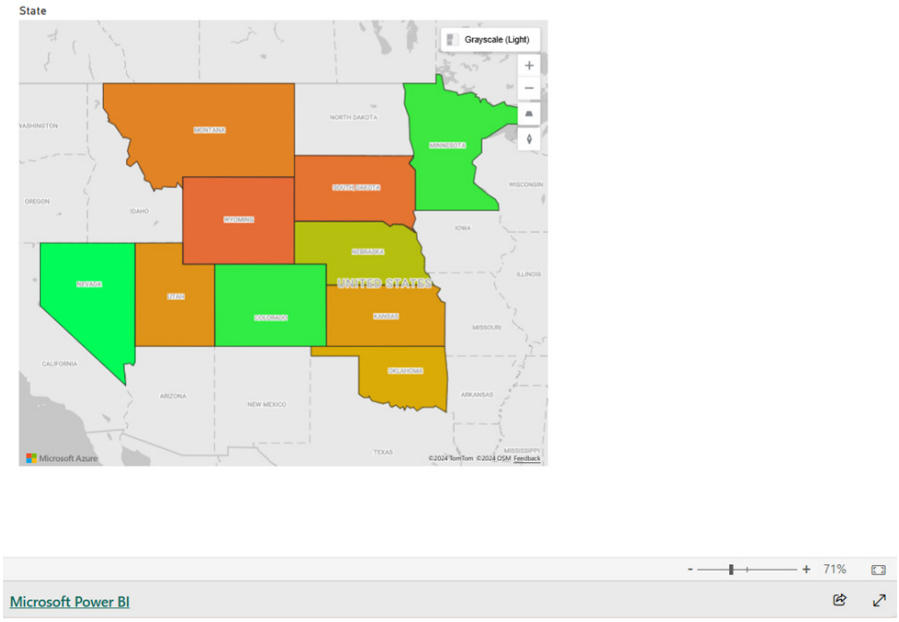


Publish to Web support for the Azure Maps visual (coming soon)

Microsoft has also announced that Azure Maps support for Publish to Web reports is rolling out. With this update, you don't need to take any extra steps to take advantage of your Azure Map visuals in Publish to Web reports. Just get your embedded code as you normally would using the Publish to Web (public) option under the Embed Report section of the File menu.



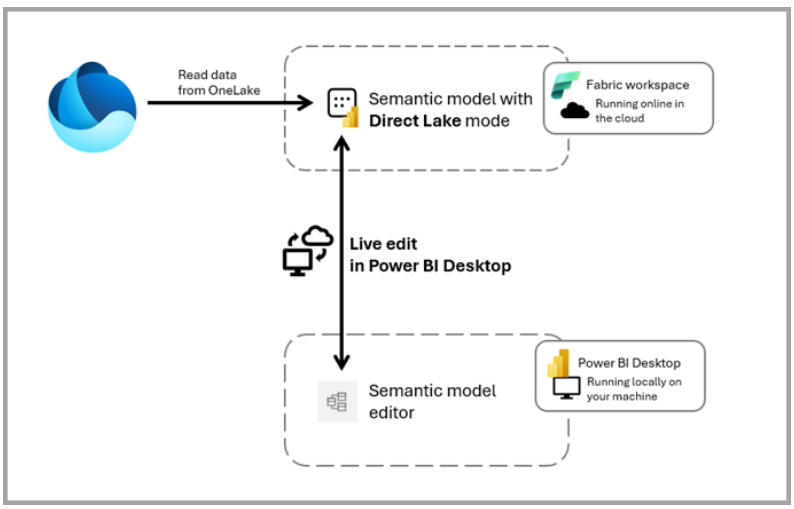
From there, you can use the embedded report as you normally would, now with your maps showing.



This update is in the process of rolling out, so you should see the change within the next couple of weeks.

Live edit of semantic models in Direct Lake mode with Power BI Desktop – updates (Preview)

Power BI has significantly enhanced the modelling performance for live editing of Direct Lake semantic models using Power BI Desktop. Upgrading to the latest version will result in at least a 50% improvement in each modelling change.



Improvements in Snowflake connector (Preview)

Microsoft has noted users' trials, tribulations and feedback on the new Snowflake connector released in Preview in January. Now, this connector has been enhanced with performance improvements and bug fixes to provide a smoother experience.

Use the official ODBC driver to connect to Vertica database (Preview)

This month, Microsoft has introduced a new option for getting data from the Vertica database using the official Vertica ODBC driver, currently available in Preview. This feature allows you to take advantage of the native client tool provided by the data source.

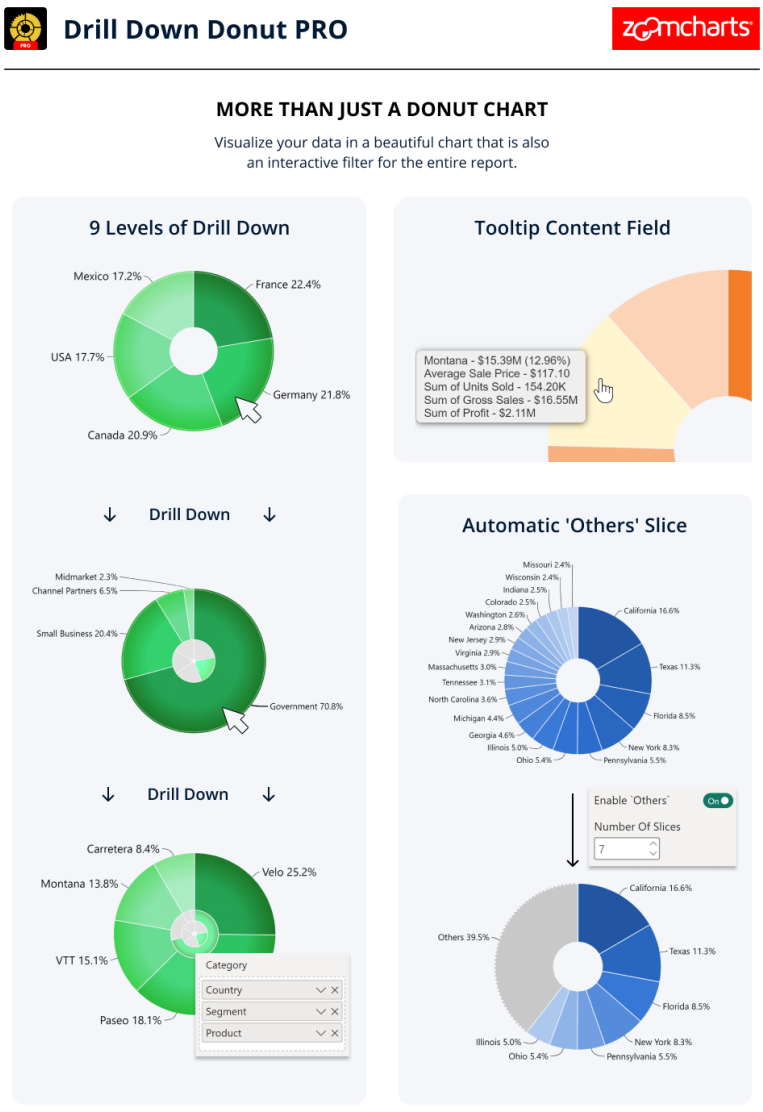
Drill Down Donut PRO by ZoomCharts

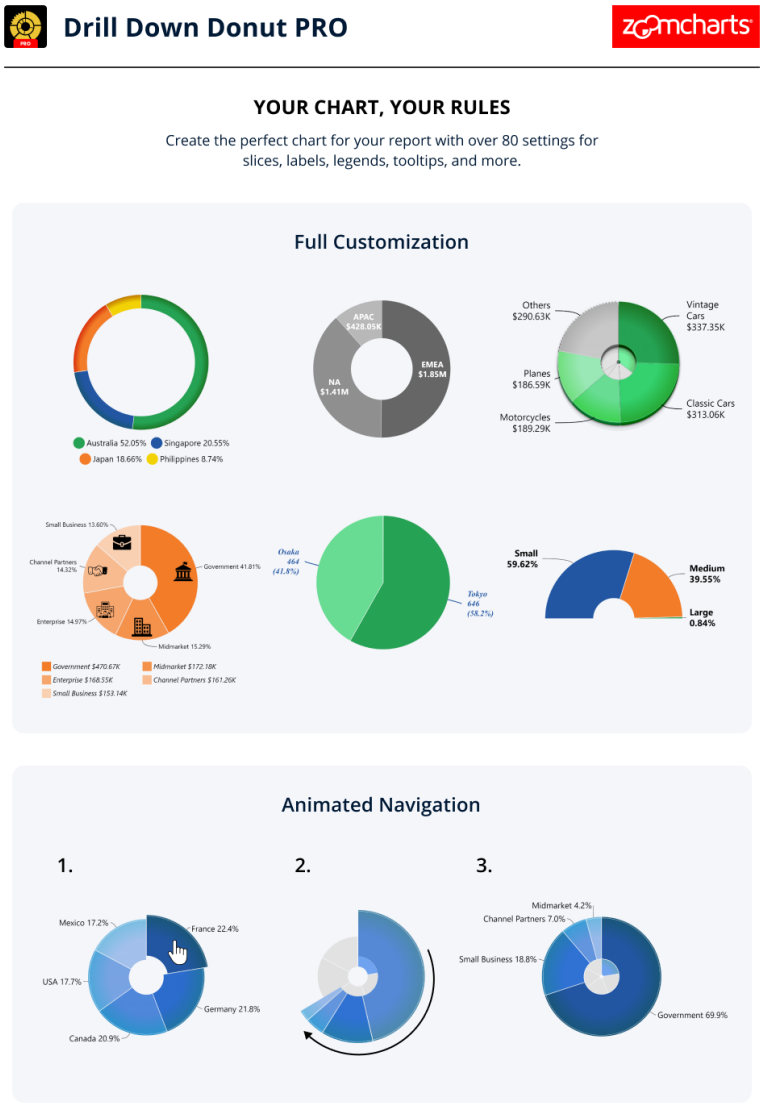
Donut PRO makes your reports more interactive and user-friendly. You may simply click directly on a slice to drill down a level and reveal more details. Donut PRO will also cross-filter with other visuals, allowing users to quickly filter data and instantly uncover valuable insights that empower business decisions.

Main features:

- **interactivity:** Donut PRO provides both mouse and touch input experiences: simply click / tap to drill down, hold to select, triple click / tap to invert selection
- **animated navigation:** each user interaction has smooth animation feedback that aids with navigation and shows exactly what changed in the chart
- **multi-level drill down:** create up to nine [9] levels of hierarchy and drill down with just a click
- **automatic 'others' slice:** declutter the chart by grouping smaller slices into one larger slice. Simply click to reveal its contents
- **cross-chart filtering:** enhance data storytelling by dynamically filtering data across multiple visuals
- **full customisation**: finetune your chart with more than 80 settings, including Donut appearance, slice colours, detail labels, legend and ToolTip customisation.

It is available on AppSource now.





Heatmap by Powerviz

The Powerviz Heatmap visualises data density and value distribution using colour gradients, transforming complex datasets into an intuitive matrix format. It is ideal for identifying patterns, correlations and outliers, aiding data-driven decision-making.

Key features include:

- **shapes:** choose from Default, Fixed or Diverging built-in shapes
- **data colours:** offers 30+ palettes, colour-blind mode and custom colour options
- **data labels:** select from different styles available or add custom labels with formatting
- **grid lines:** improve data readability with visual grid lines
- **reference lines:** highlight key points using **x**-axis and **y**-axis lines
- **totals:** display row and column totals with advanced bar customisation
- **null values:** customise *null* values (Blank / NA / Null / “0”) and their styling
- **Small Multiples:** split visuals into smaller ones using selected fields
- **conditional formatting:** spot outliers with Value / Percentage / Ranking based rules
- **ranking:** filter Top / Bottom **N** rows or columns
- **other features** included Axis Settings, Grid View, Sorting, Show Condition and more.

Business use cases include Sales Analysis, Performance Evaluation and Risk Assessment.

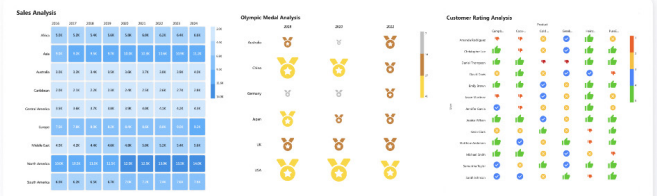
You can try Heatmap visual for free from App Source.

Heatmap PBI Certified

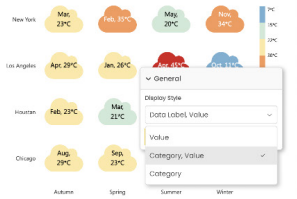
Powerviz
www.powerviz.ai

INTRODUCING HEATMAP BY POWERVIZ

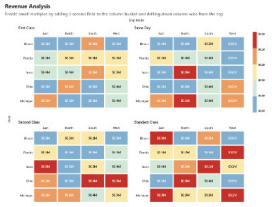
Most advanced Power BI Heatmap with Shapes, Custom Labels, Grid Lines, Conditional Formatting, & Small Multiples.



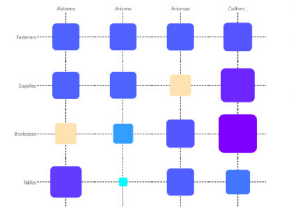
Shapes



Labels



Small Multiples



Grid Lines

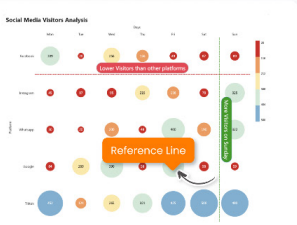
Conditional Formatting

Heatmap PBI Certified

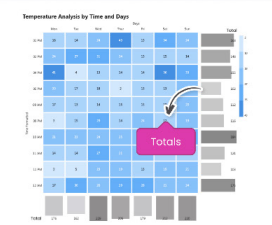
Powerviz
www.powerviz.ai

INTRODUCING HEATMAP BY POWERVIZ

Reference Lines, Totals, Data Colors, Null value customization, & More.

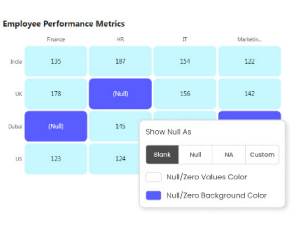


Reference Lines



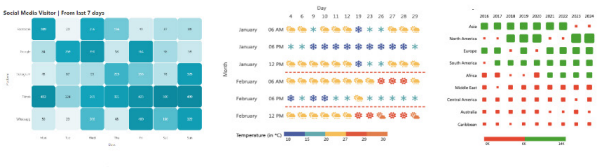
Totals

Data Colors



Null Values Customization

Heatmap Examples



Social Media Visitor | From last 7 days

Climate Change in Delhi

Sales Analysis

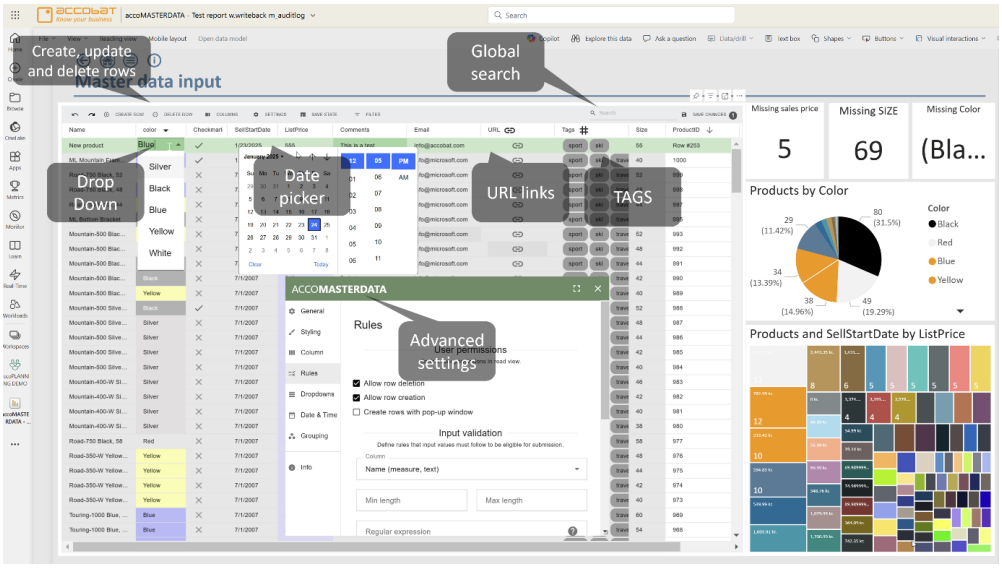
Maximise data efficiency with accoMASTERDATA writenback for Power BI

You can now unlock the power of masterdata management with accoMASTERDATA for Power BI. You may streamline your data processes and enhance your writeback capabilities for smarter business decisions.

Key features include:

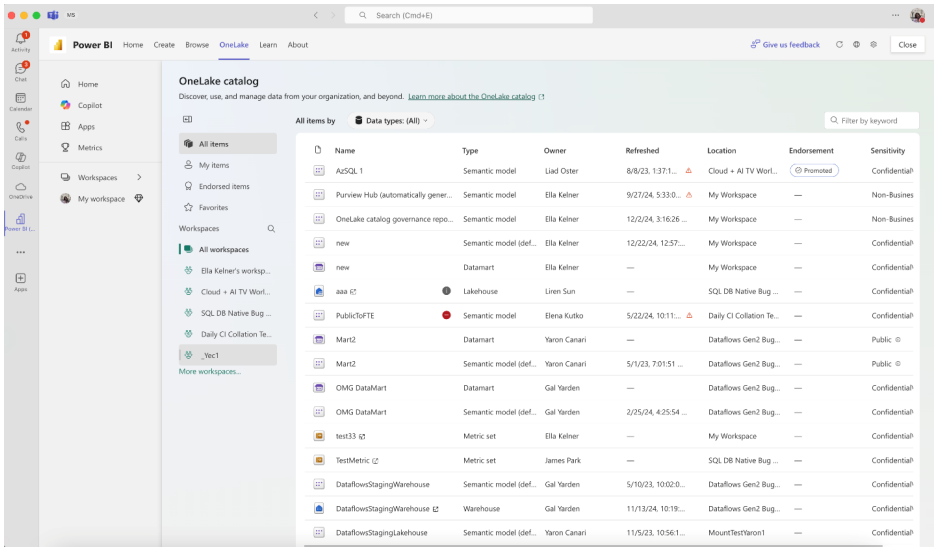
- **effortless writeback:** easily configure writeback to SQL and FABRIC databases. Track changes in a dedicated log for full transparency
- **full control:** create, update and delete rows in Masterdata tables to keep your data dynamic and accurate
- **custom validation:** use regular expressions for custom data validation and ensure integrity
- **dropdown list validation:** restrict input to predefined options for consistent data entry
- **conditional formatting and rules:** ensure quality data with powerful validation and formatting tools
- **enhanced copy-paste:** copy and paste from Excel directly into Power BI whilst maintaining validation rules
- **customisable themes:** create visually stunning reports with flexible theme options
- **tags and column grouping:** organise data with multi-selected tags and improve navigation with collapsible column sections.

Integrating accoMASTERDATA with Power BI will assist your data management, improve reporting and streamline your operations (apparently).



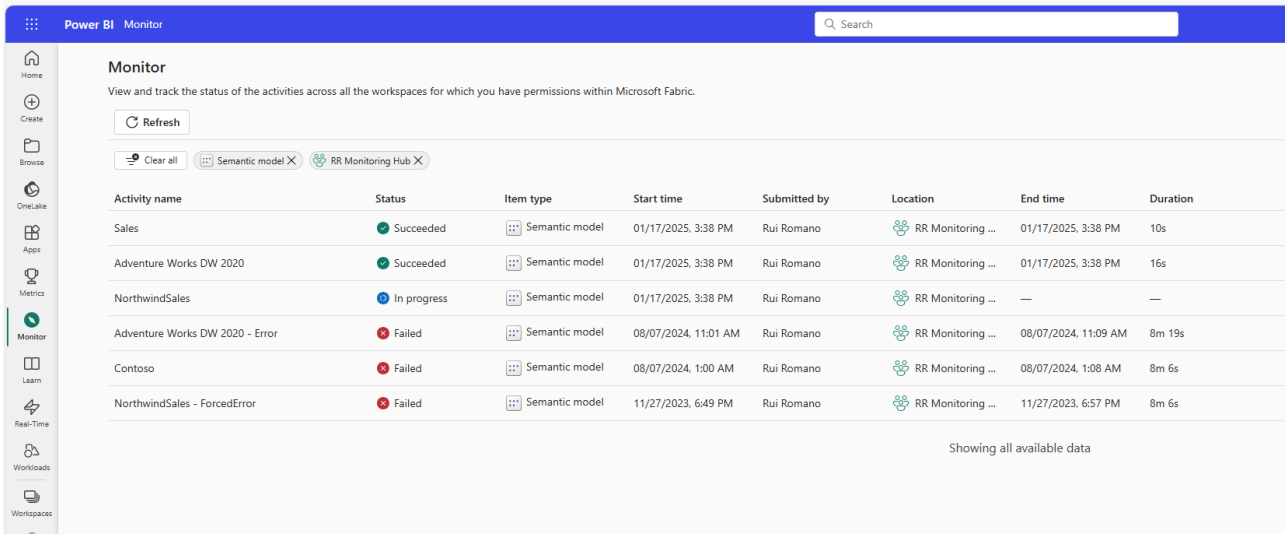
The OneLake catalog is now available in Microsoft Teams

The OneLake catalog is now integrated into the Power BI app experience in Microsoft Teams, enabling data discovery and exploration within the Microsoft Office ecosystem. This integration empowers business users and professionals to interact with their data – explore it, take actions and more – all without leaving Teams.

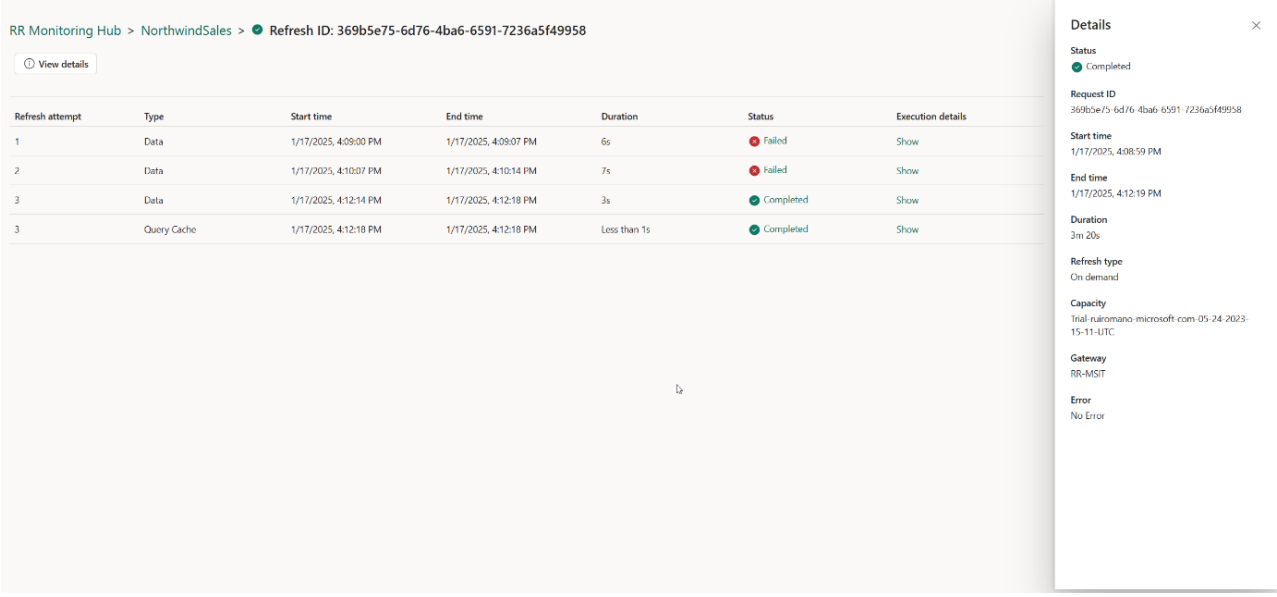


Monitoring Hub enhancement for semantic models

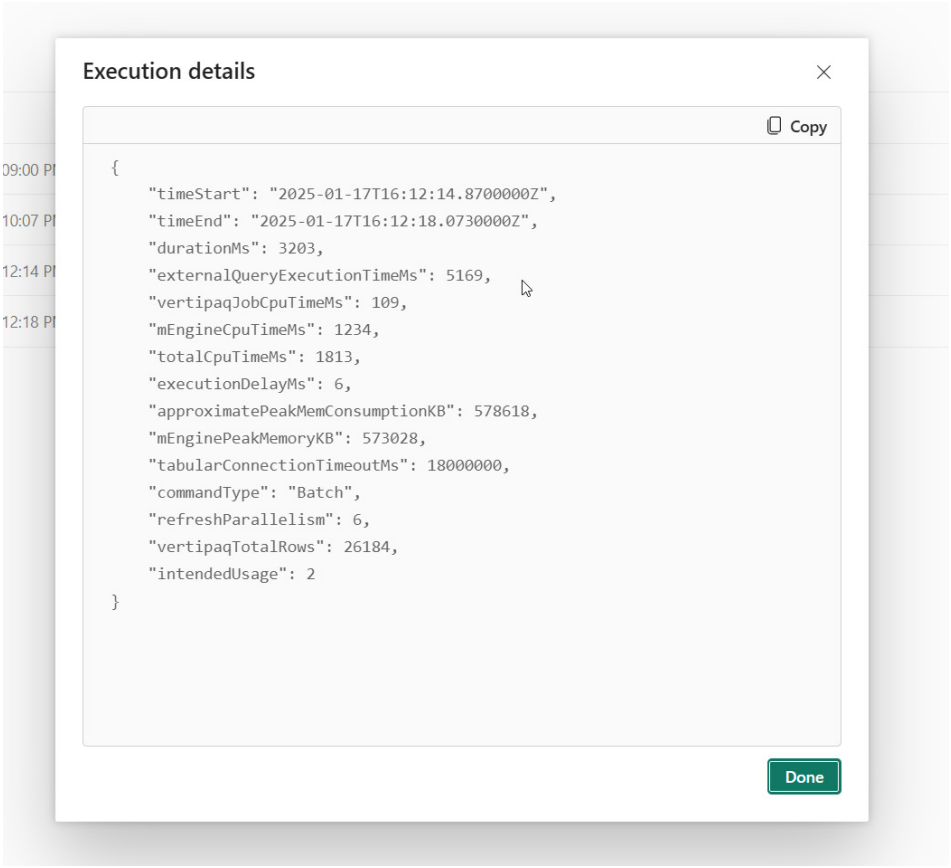
In Fabric Monitoring Hub, you can centrally monitor Microsoft Fabric activities. It displays refresh activities for all semantic models, each showing one line with the status of the last refresh.



Previously, clicking on an activity name in the Monitoring Hub directed you to the semantic model detail page. This month, Microsoft has introduced the 'Semantic model refresh detail' page. This new page shows comprehensive details of a selected refresh activity, including capacity, gateway, start and end times, error details and multiple refresh attempts.



For each refresh attempt, you can view the execution metrics by clicking on the 'Show' link in the 'Execution details' column. This information can assist with troubleshooting or optimising the semantic model refresh. Previously, this data was accessible through Log Analytics or Fabric Workspace Monitoring.

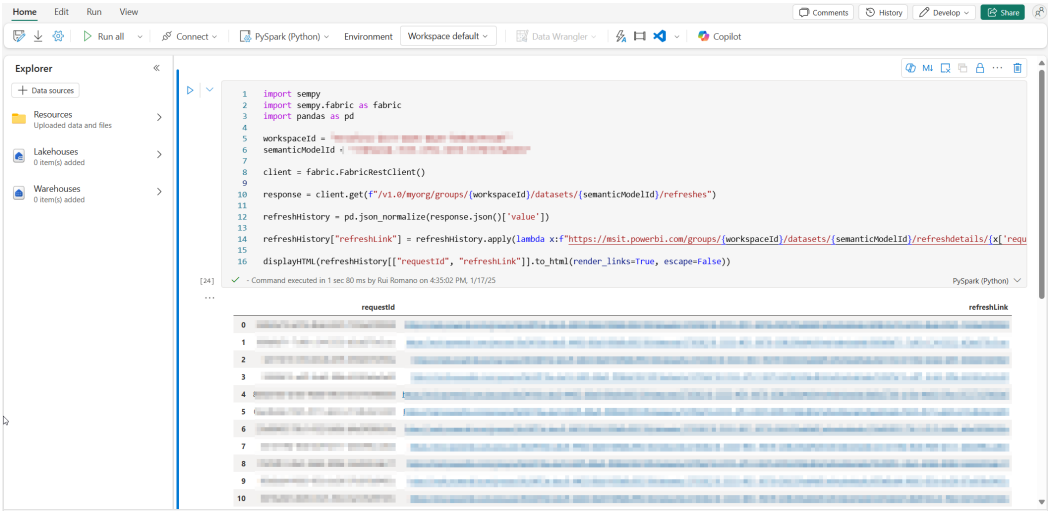


It's also possible to link refresh details from external applications. The semantic model refresh detail page can be accessed from other locations by constructing a URL with the workspace, semantic model and refresh ID:

https://app.powerbi.com/groups/{workspaceId}/datasets/{semanticModelId}/refreshdetails/{refreshId}

For instance, this Fabric Notebook uses semantic link sempy and Power BI API Get Refresh History to create a refresh detail URL for each run of a semantic model:

```
import sempy

import sempy.fabric as fabric

import pandas as pd

workspaceId = "[Your Workspace Id]"

semanticModelId = "[Your semantic model Id]"

client = fabric.FabricRestClient()

response = client.get(f"/v1.0/myorg/groups/{workspaceId}/datasets/{semanticModelId}/refreshes")

refreshHistory = pd.json_normalize(response.json()['value'])

refreshHistory["refreshLink"] = refreshHistory.apply(lambda x:f"https://msit.powerbi.com/groups/
{workspaceId}/datasets/{semanticModelId}/refreshdetails/{x['requestId']}", axis=1)

displayHTML(refreshHistory[["requestId", "refreshLink"]].to_html(render_links=True, escape=False))
```



More next month.

New Features for Excel

With the recent announcement of Copilot now becoming available to Microsoft 365 Personal and Family subscribers (for most international markets), attention nor returns to other functionalities and features.

The full list is as follows:

Excel for Windows, Mac and the web

- Text Analysis in Excel

Excel for Windows and Mac

- Focus Cell
- **TRANSLATE** and **DETECTLANGUAGE** functions

Excel for Windows

- Dark Mode in Excel (Insiders)
- Improvements to five [5] text functions (Insiders)
- New Forms integration for Windows.

Let's get started.

Text Analysis in Excel

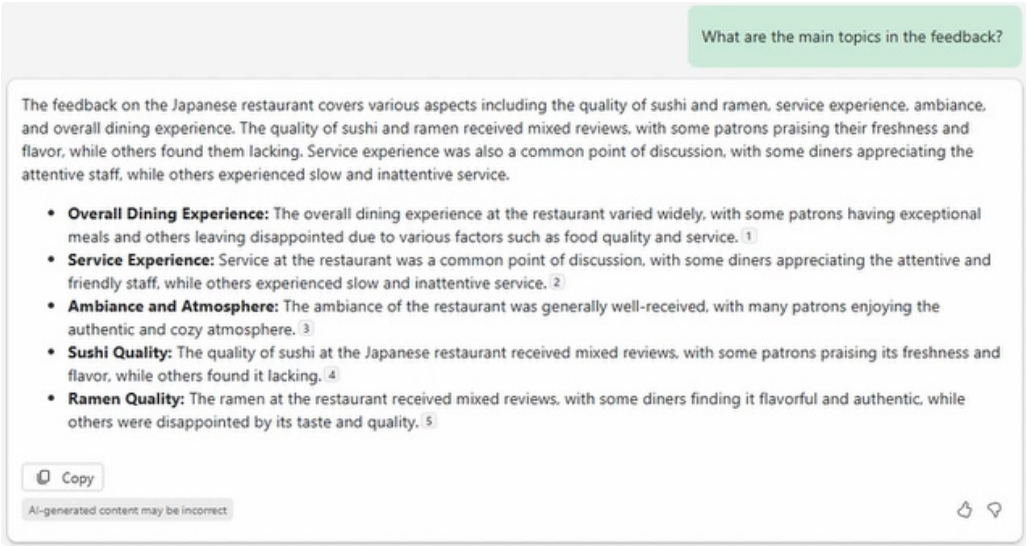
Text can be as important as numbers. There are important insights to be found through surveys, reviews and feedback surveys, amongst other written documents and actions. For those working with such data, it can be time-consuming and difficult to uncover trends and insights.

In Excel for Windows, Mac and the web, Microsoft wants to make Excel a one-stop shop for analysing text data, whether you want quick and easy insights or deep analysis. Currently, there are three [3] new features that you can use to analyse your text data.

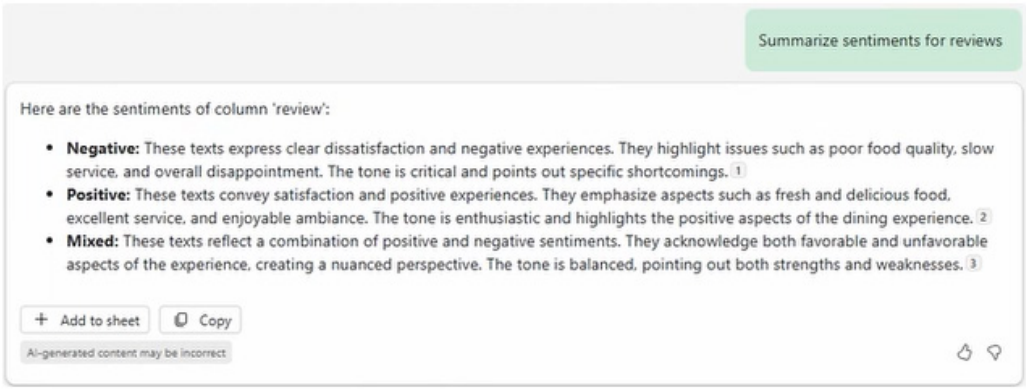
1. COPILOT IN EXCEL

Copilot can help you analyse a column of text data: you may do all of the following simply by typing a prompt.

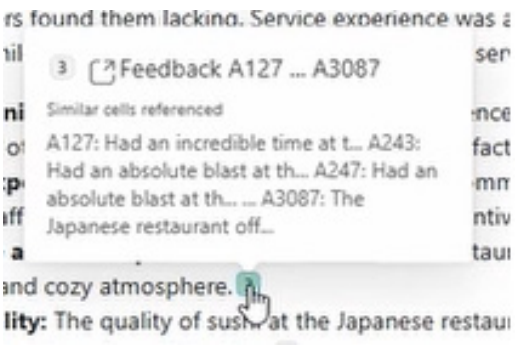
For example, Copilot can scan your text data and return a summary of the text and a list of themes. You can ask for a specific length, tone or focus area in your prompt.



Coming soon, you will be able to run sentiment analysis, whereby Copilot can identify attitudes and views in your text data, summarising insights for each sentiment.

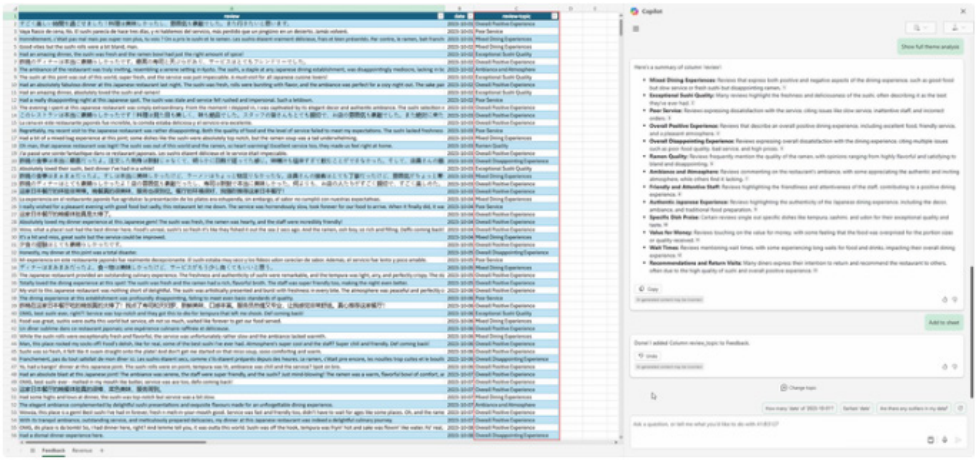


Another feature appearing soon is Copilot's ability to undertake text analysis results to include sources from your data so you can validate results and don't have to spend extra time searching for examples.



It's going to be harder for it to hallucinate!

Furthermore, once you have a topic list or sentiment analysis, Copilot can insert a column to your dataset with a label for each of your text items.

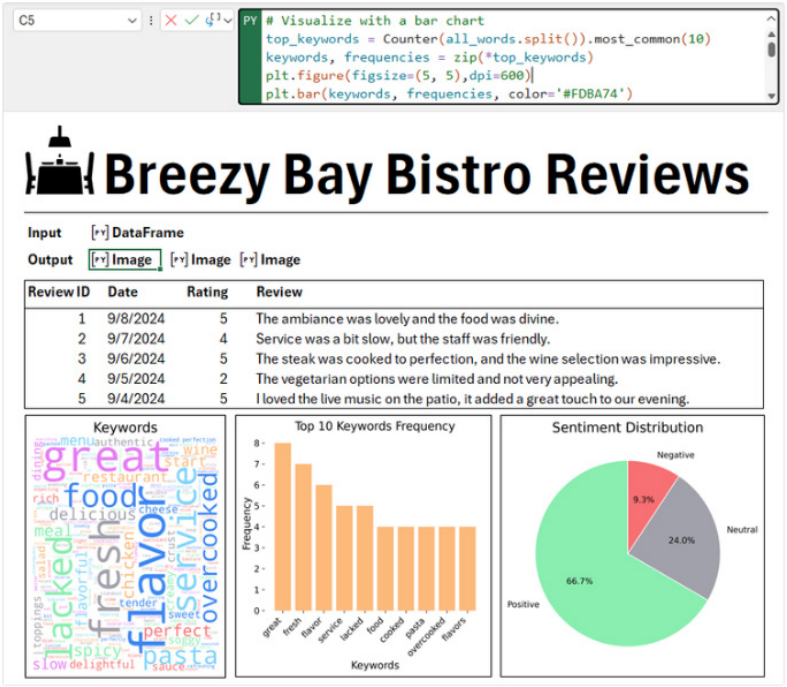


Then, you can ask for charts or PivotTables to help you find deeper insights about top issues or trends.

Text analysis is currently available for datasets with up to about 3,000 rows. It may not sound that high, but it's a start, with support for larger datasets coming soon.

2. PYTHON IN EXCEL

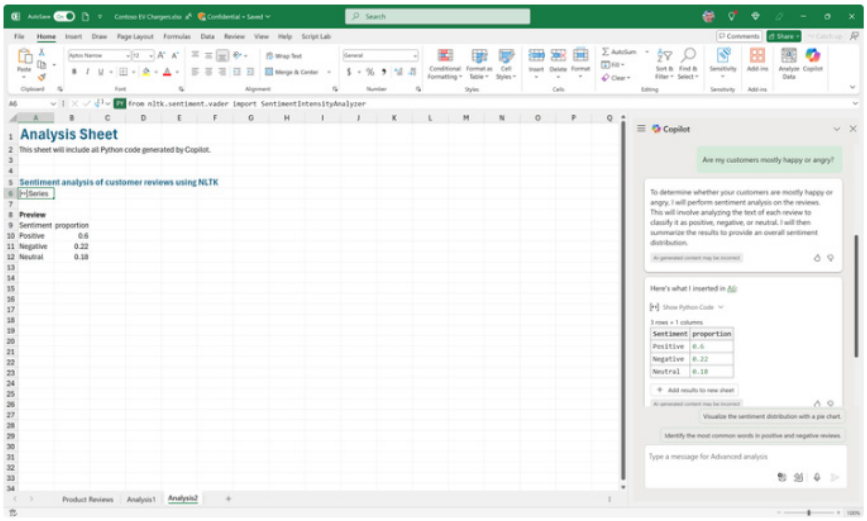
You may perform customised text analysis using Python in Excel. With access to the NLTK library, the possibilities for manipulation and analysis have increased.



You can create a word cloud from your text data to visualise patterns in word frequency:

```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt

# Combine all reviews into a single string
text = ' '.join(Reviews_df['Reviews'])

# Define stopwords to exclude common articles and words
stopwords = set(STOPWORDS)

# Generate the word cloud
wordcloud = WordCloud(stopwords=stopwords, background_color='white', width=800, height=400).generate(text)

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

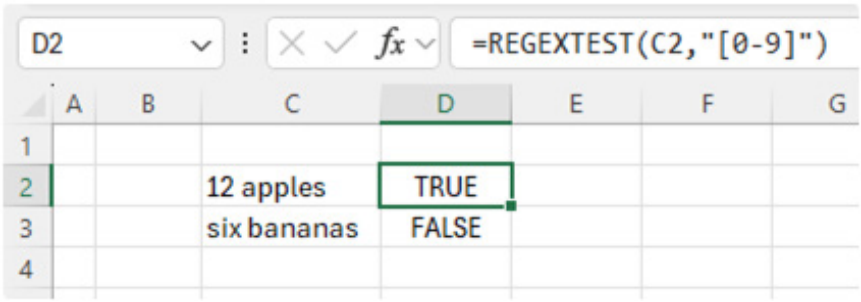
All of this is made easy using Copilot in Excel with Python. Just ask Copilot to analyse your text with Python and Copilot will do the rest.



You may also undertake sentiment analysis using Copilot in Excel with Python. However, this is currently available in only a limited capacity, starting with Excel for Windows in the United States.

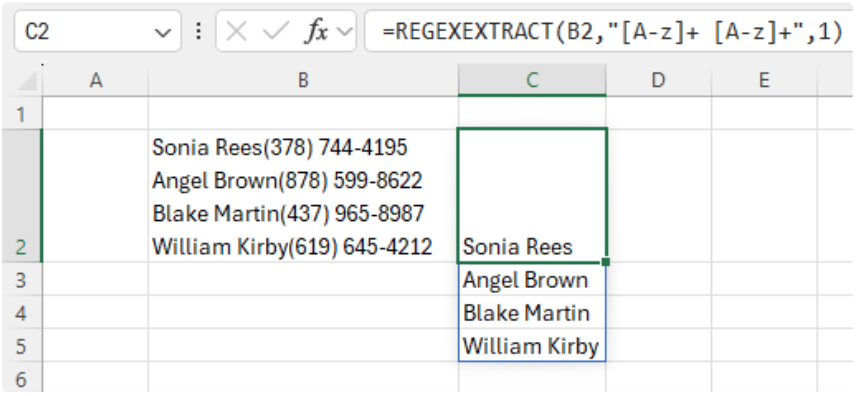
3. NEW TEXT FUNCTIONS

Also, Microsoft has added new functions that allow you to manipulate text directly. Regular expression ("regex") functions unlock powerful pattern matching capabilities. **REGEXTEST** identifies whether text matches a particular pattern.

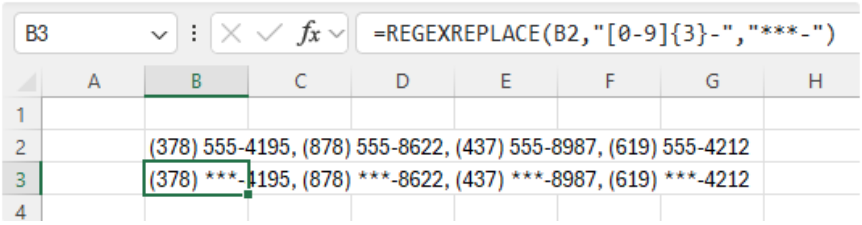


The above checks whether the strings in column **C** contain numerical digits, using the regex pattern **"[0-9]"**.

You may also use the **REGEXEXTRACT** function to extract strings from your text according to a pattern. In this example, we can extract names from text using the pattern **"[A-z]+ [A-z]+"**, which matches two [2] groups of alphabet characters separated by a space:



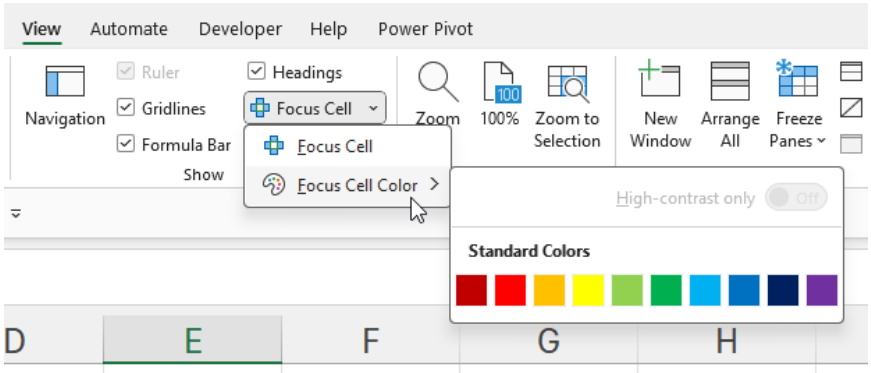
Finally, **REGEXREPLACE** can replace text that matches a pattern with a string of your choice. Here, we replace the first three [3] digits of each phone number with ***, using the pattern **"[0-9]{3}-"**, which matches against three [3] numerical digits followed by **"-"**:



If you have text data in multiple languages, you can now translate text directly in Excel. With the **TRANSLATE** function *(see below also)*, you can use Microsoft Translation Services on your data without leaving the Excel grid. Here is a translation from Japanese to English:

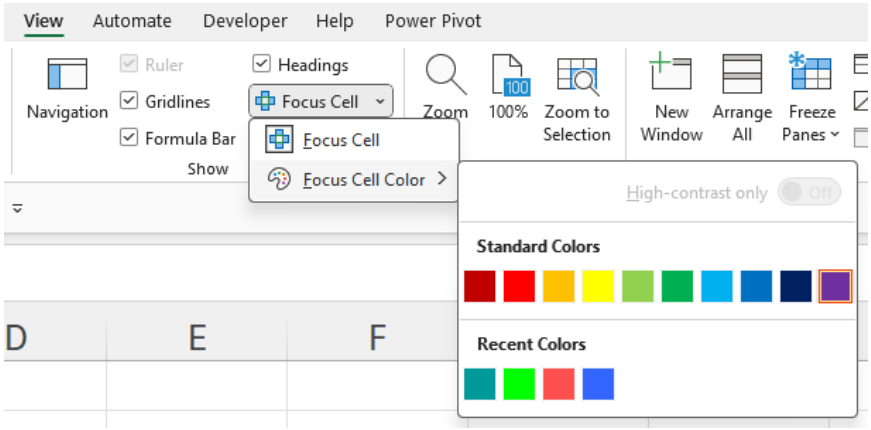
Focus Cell

Focus Cell is now Generally Available for Excel for Windows and Mac.



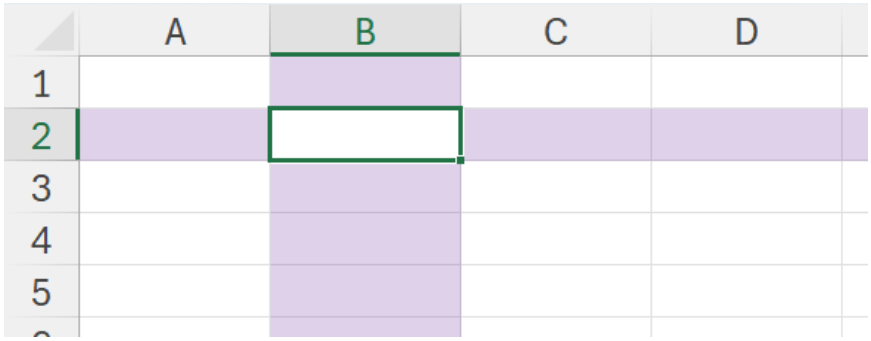
Found on the View tab in the Ribbon, 'Focus Cell' is new to Excel for Windows and has now been "officially announced" (even though we detailed it two months ago!). It provides a small drop-down menu:

- **Focus Cell:** this toggles the feature on or off
- **Focus Cell Color:** this allows you to select from a wide array of, er, 10 colours to use for highlighting. Actually, that's not true as 'Recent Colors' *(sic)* will show other colours that you may employ:



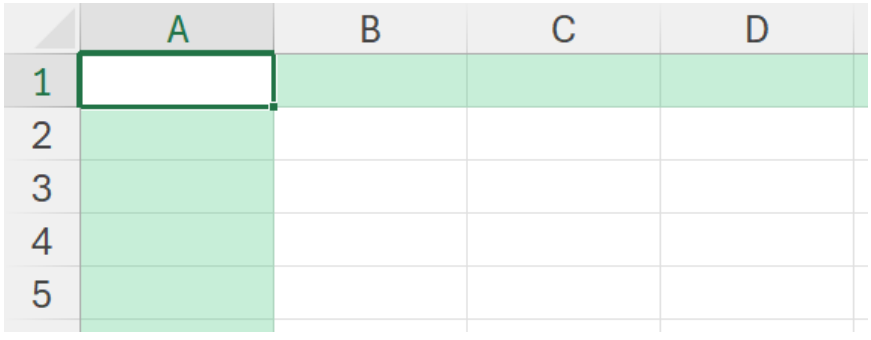
At this stage, we are not sure what the 'High-contrast only' toggle switch does as it appears to be permanently disabled.

So what does it do? It appears to enhance accessibility for the visually impaired as it highlights the active row and column of the cell selected, *viz.*



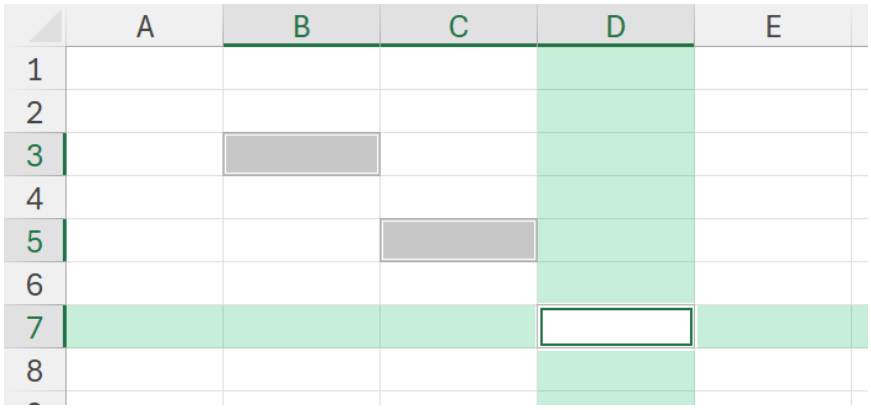
Here, cell **B2** imitates Whoopi Goldberg's 1985 hit 'The Color Purple', although clearly the colour displayed is not quite the same as the colours depicted in the 'Focus Cell Color' dialog.

Clearly, this feature helps users to zoom in on the cell selected. Other colours may be used, *e.g.*

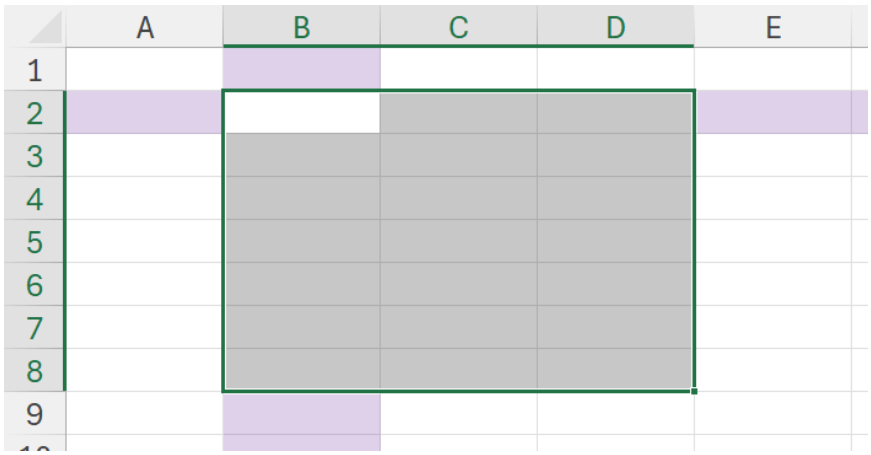


Here, cell **A1** sees green. Envious, methinks.

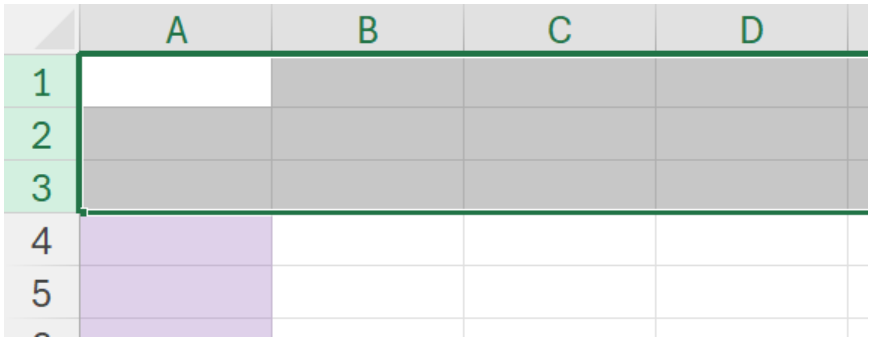
If multiple cells are selected, the active cell remains highlighted:

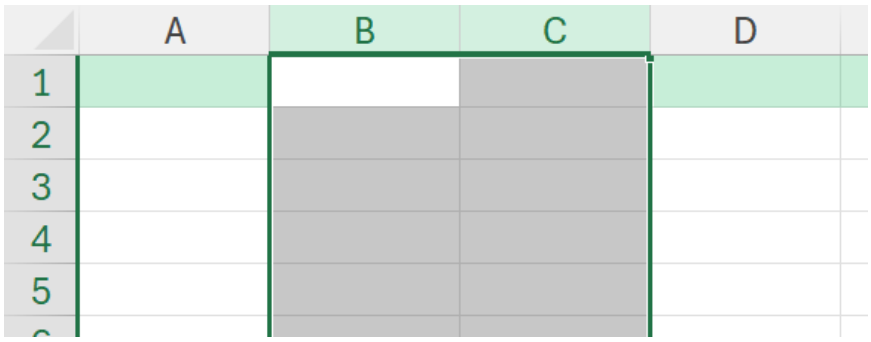


or



It does seem to act a little unexpectedly though if entire rows or column are chosen:





TRANSLATE and DETECTLANGUAGE functions

TRANSLATE and **DETECTLANGUAGE** have now been made available in Excel for Windows and Mac, but they remain what is known as "Preview" functions. This means that their signature – and consequently, their results – may change significantly before they become Generally Available.

Microsoft is introducing these two new functions to simplify and automate translations directly within your spreadsheet:

- **TRANSLATE():** a function that translates a text from one language to another
- **DETECTLANGUAGE():** a function that detects the language of the specified text.

TRANSLATE

TRANSLATE takes text you provide and translates it from one language to another using Microsoft Translation Services. Currently, there are 133 languages supported – including two variations of Klingon!!

At the time of writing, these are:

1. Afrikaans
2. Albanian
3. Amharic
4. Arabic
5. Armenian
6. Assamese
7. Azerbaijani (Latin)
8. Bangla
9. Bashkir
10. Basque
11. Bhojpuri
12. Bodo
13. Bosnian (Latin)
14. Bulgarian
15. Cantonese (Traditional)
16. Catalan
17. Chinese (Literary)
18. Chinese Simplified
19. Chinese Traditional
20. chiShona
21. Croatian
22. Czech
23. Danish
24. Dari
25. Divehi
26. Dogri
27. Dutch
28. English
29. Estonian
30. Faroese
31. Fijian
32. Filipino
33. Finnish
34. French
35. French (Canada)
36. Galician
37. Georgian
38. German
39. Greek
40. Gujarati
41. Haitian Creole
42. Hausa
43. Hebrew
44. Hindi

45. Hmong Daw (Latin)
46. Hungarian
47. Icelandic
48. Igbo
49. Indonesian
50. Inuinnaqtun
51. Inuktitut
52. Inuktitut (Latin)
53. Irish
54. Italian
55. Japanese
56. Kannada
57. Kashmiri
58. Kazakh
59. Khmer
60. Kinyarwanda
61. Klingon
62. Klingon (plqaD)
63. Konkani
64. Koreanko
65. Kurdish (Central)
66. Kurdish (Northern)
67. Kyrgyz (Cyrillic)
68. Lao
69. Latvian
70. Lithuanian
71. Lingala
72. Lower Sorbian
73. Luganda
74. Macedonian
75. Maithili
76. Malagasy
77. Malay (Latin)
78. Malayalam
79. Maltese
80. Maori
81. Marathi
82. Mongolian (Cyrillic)
83. Mongolian (Traditional)
84. Myanmar
85. Nepali
86. Norwegian
87. Nyanja
88. Odia
89. Pashto
90. Persian
91. Polish
92. Portuguese (Brazil)
93. Portuguese (Portugal)
94. Punjabi
95. Queretaro Otomi
96. Romanian
97. Rundi
98. Russian
99. Samoan (Latin)
100. Serbian (Cyrillic)
101. Serbian (Latin)
102. Sesotho
103. Sesotho sa Leboa
104. Setswana
105. Sindhi
106. Sinhala
107. Slovak
108. Slovenian
109. Somali (Arabic)
110. Spanish es
111. Swahili (Latin)
112. Swedish
113. Tahitian
114. Tamil
115. Tatar (Latin)
116. Telugu
117. Thai
118. Tibetan
119. Tigrinya
120. Tongan
121. Turkish
122. Turkmen (Latin)
123. Ukrainian
124. Upper Sorbian
125. Urdu
126. Uyghur (Arabic)
127. Uzbek (Latin)
128. Vietnamese
129. Welsh
130. Xhosa
131. Yoruba
132. Yucatec Maya
133. Zulu.

As mentioned above, the **TRANSLATE** function allows you to translate text from one language to another in Microsoft Excel by using Microsoft Translation Services. The full signature is:

TRANSLATE(text, [source_language], [target_language])

This function has the following arguments:

- **text:** the **text** to translate. This value should either be enclosed in quotation marks or be a reference to a cell containing the appropriate **text**
- **source_language (optional):** the language code of the source language (*e.g.* "en" for English or "es" for Spanish). If not specified, the language will be automatically detected based upon the **text** provided. Auto-detection is supported for most languages. It is recommended to specify the language if known, especially for shorter texts
- **target_language (optional):** the language code of the target language (*e.g.* "en" for English or "es" for Spanish). If not specified, the system language will be used as the target language.

The supported languages and their respective language codes are as follows:

Language	Language code
Afrikaans	af
Albanian	sq
Amharic	am
Arabic	ar
Armenian	hy
Assamese	as
Azerbaijani (Latin)	az
Bangla	bn
Bashkir	ba
Basque	eu
Bhojpuri	bho
Bodo	brx
Bosnian (Latin)	bs
Bulgarian	bg
Cantonese (Traditional)	yue
Catalan	ca
Chinese (Literary)	lzh
Chinese Simplified	zh-Hans
Chinese Traditional	zh-Hant
chiShona	sn
Croatian	hr
Czech	cs
Danish	da
Dari	prs
Divehi	dv
Dogri	doi
Dutch	nl
English	en
Estonian	et
Faroese	fo
Fijian	fj
Filipino	fil
Finnish	fi
French	fr
French (Canada)	fr-ca
Galician	gl
Georgian	ka

Language	Language code
German	de
Greek	el
Gujarati	gu
Haitian Creole	ht
Hausa	ha
Hebrew	he
Hindi	hi
Hmong Daw (Latin)	mww
Hungarian	hu
Icelandic	is
Igbo	ig
Indonesian	id
Inuinnaqtun	ikt
Inuktitut	iu
Inuktitut (Latin)	iu-Latn
Irish	ga
Italian	it
Japanese	ja
Kannada	kn
Kashmiri	ks
Kazakh	kk
Khmer	km
Kinyarwanda	rw
Klingon	tlh-Latn
Klingon (plqaD)	tlh-Piqd
Konkani	gom
Korean	ko
Kurdish (Central)	ku
Kurdish (Northern)	kmr
Kyrgyz (Cyrillic)	ky
Lao	lo
Latvian	lv
Lithuanian	lt
Lingala	ln
Lower Sorbian	dsb
Luganda	lug
Macedonian	mk
Maithili	mai

Language	Language code
Malagasy	mg
Malay (Latin)	ms
Malayalam	ml
Maltese	mt
Maori	mi
Marathi	mr
Mongolian (Cyrillic)	mn-Cyrl
Mongolian (Traditional)	mn-Mong
Myanmar	my
Nepali	ne
Norwegian	nb
Nyanja	nya
Odia	or
Pashto	ps
Persian	fa
Polish	pl
Portuguese (Brazil)	pt
Portuguese (Portugal)	pt-pt
Punjabi	pa
Queretaro Otomi	otq
Romanian	ro
Rundi	run
Russian	ru
Samoan (Latin)	sm
Serbian (Cyrillic)	sr-Cyrl
Serbian (Latin)	sr-Latn
Sesotho	st
Sesotho sa Leboa	nso
Setswana	tn

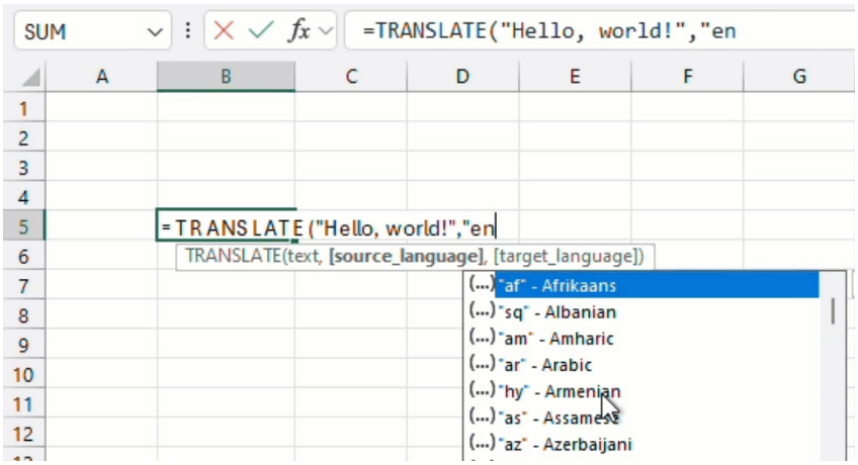
Language	Language code
Sindhi	sd
Sinhala	si
Slovak	sk
Slovenian	sl
Somali (Arabic)	so
Spanish	es
Swahili (Latin)	sw
Swedish	sv
Tahitian	ty
Tamil	ta
Tatar (Latin)	tt
Telugu	te
Thai	th
Tibetan	bo
Tigrinya	ti
Tongan	to
Turkish	tr
Turkmen (Latin)	tk
Ukrainian	uk
Upper Sorbian	hsb
Urdu	ur
Uyghur (Arabic)	ug
Uzbek (Latin)	uz
Vietnamese	vi
Welsh	cy
Xhosa	xh
Yoruba	yo
Yucatec Maya	yua
Zulu	zu

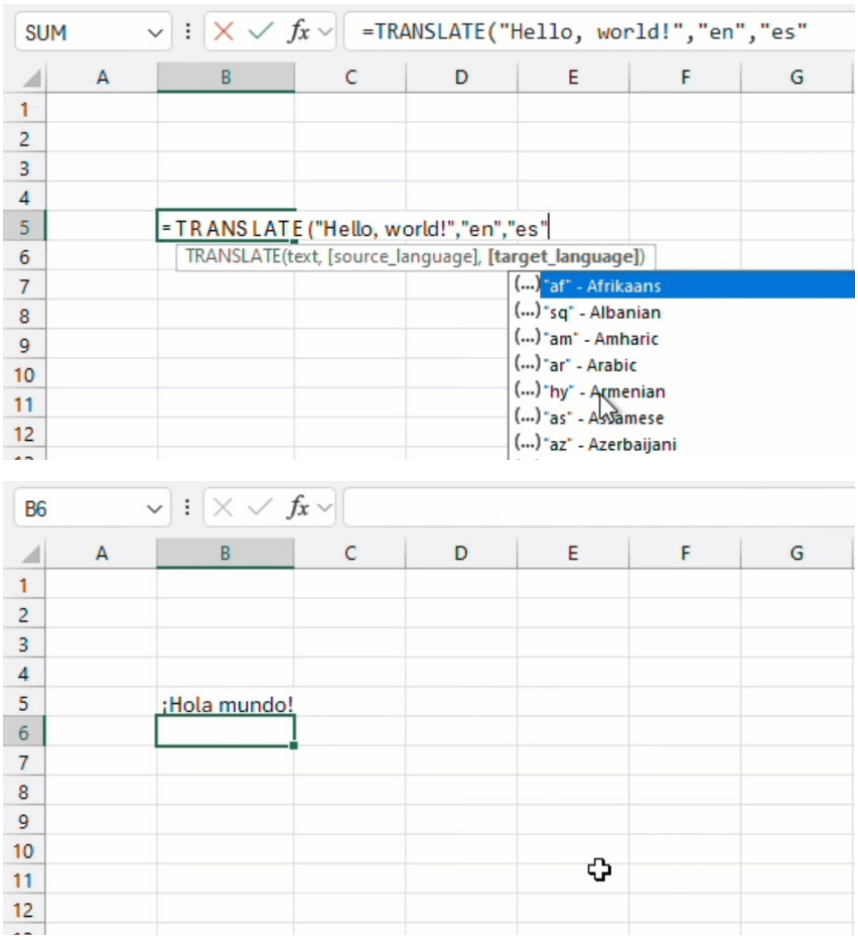
Suppose you have the following text in cell **A1**: "Hello, World!" and you want to translate it to Spanish. You can use the **TRANSLATE** function as follows:

=TRANSLATE(A1, "en", "es")

In this example, the source language is English (en) and the target language is Spanish (es). The translated text, "Hola mundo!" will be displayed in the cell where you entered the formula.

Alternatively, you may just type the text in, *viz.*





Common errors include the following:

- **Text Too Long:** you have too many characters in a cell. Reduce your cell size and try again
- **Error in Value:** you have a non-text value in your cell. The function only accepts a text argument
- **Invalid Language:** you have entered an invalid language code or one not presently supported (see above)
- **Request Throttled:** you have exceeded your daily quota of the translation function (now that is interesting, but we are not quite sure what that means at the time of writing).

DETECTLANGUAGE

DETECTLANGAUGE detects the language of text you provide using the Microsoft Translation Services and returns the language code. The full signature is:

DETECTLANGUAGE(text)

The function has the following arguments:

- **text:** the **text** or reference to cells containing **text** to evaluate.

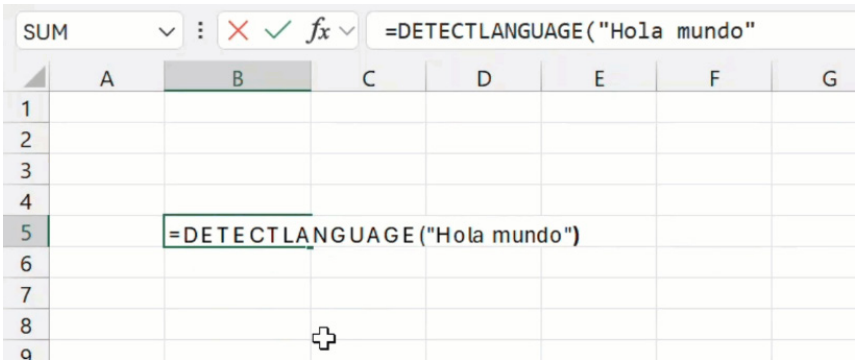
The supported languages and their respective language codes are as above.

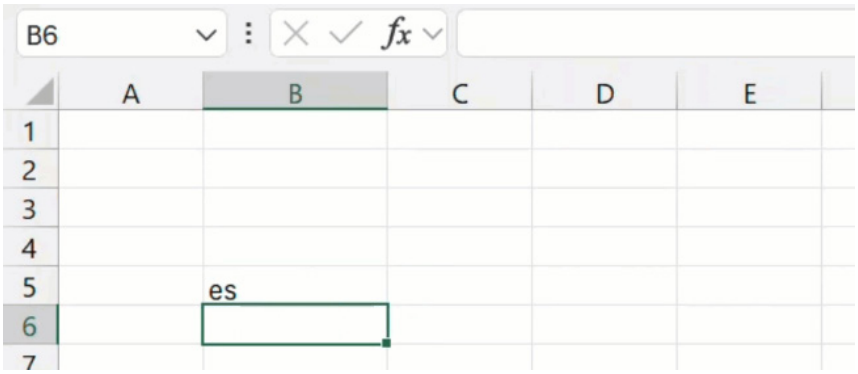
Suppose you have the following text in cell **A1**: "Hola mundo!" and you want to find out what the language of the text is. You can use the **DETECTLANGUAGE** function as follows:

=DETECTLANGUAGE(A1)

This will return the detected language for the text in cell **A1**. The language code "es" for Spanish will be displayed in the cell where you entered the formula.

Alternatively, you may just type the text in, *viz.*

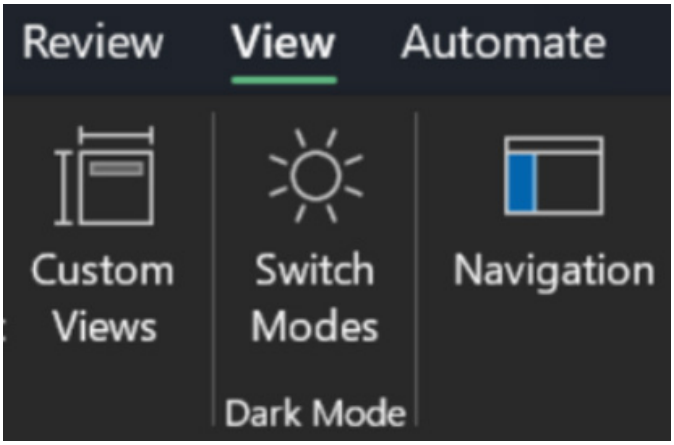




Common errors include the following:

- **Text Too Long:** you have too many characters in a cell. Reduce your cell size and try again
- **Error in Value:** you have a non-text value in your cell. The function only accepts a text argument
- **Invalid Language:** you have entered an invalid language code or one not presently supported (see above)
- **Request Throttled:** you have exceeded your daily quota of the translation function.

Dark Mode in Excel (Insiders)

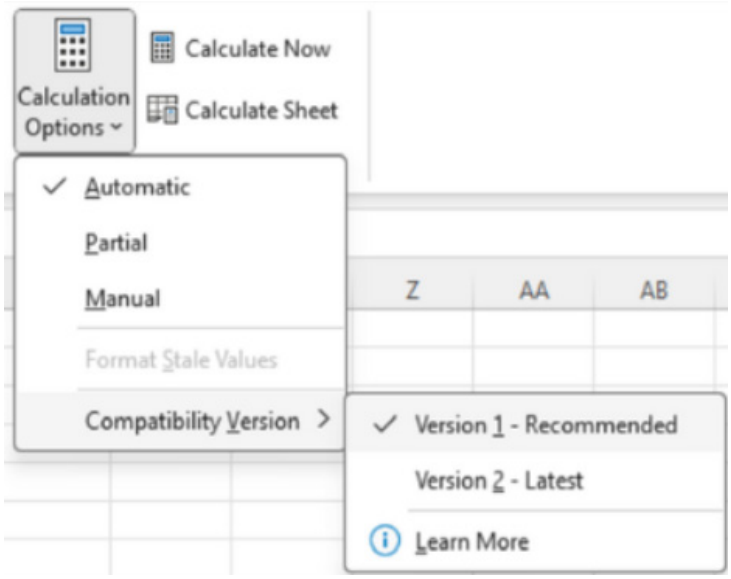


You can now enable Dark Mode which darkens your entire sheet including the cells, by using the 'Switch Modes' button on the View tab. To make sure you are using a supported Office theme, select **File -> Account -> Office Theme** and confirm that either Black or 'Use system setting' is selected. Do note that if you select the 'Use system settings' option, you will also need to make sure that your Windows theme is set to Black.

To turn on Dark Mode, select View, then 'Switch Modes'. To turn Dark Mode off, simply select **View -> Switch Modes** again.

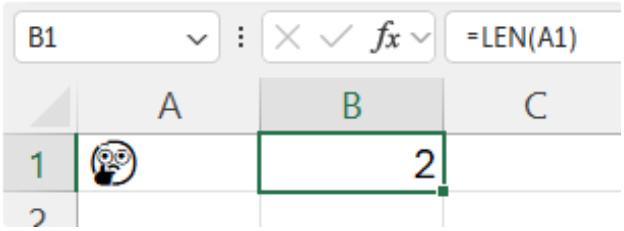
This feature is rolling out to Excel for Windows Beta Channel users running Version 2502 (Build 18508.20000) or later.

Improvements to five [5] text functions (Insiders)



Microsoft is updating the **LEN**, **MID**, **SEARCH**, **FIND** and **REPLACE** functions to support Unicode surrogates in Excel for Windows (Insiders). This has been a major pain point for international users and those who use emojis, because these five functions have been double-counting certain characters. This was inconsistent with more modern text functions, which did not have this issue (including **LEFT**, **RIGHT**, **TEXTBEFORE**, **TEXTAFTER** and **TEXTSPLIT**).

For example, you can see the existing behaviour where **LEN(🤔)** = 2. That's a bit confusing! 🤔

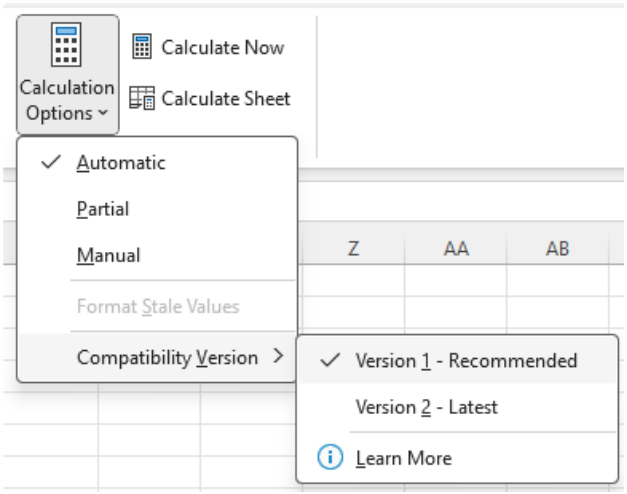


With this update, the **LEN**, **MID**, **SEARCH**, **FIND** and **REPLACE** functions now count each character once, no matter what that character is. To ensure that calculations in your existing workbooks don't change, these improvements are being rolled out as what is known as **Compatibility Version 2**.

INTRODUCING COMPATIBILITY VERSIONS

To release improvements to functions while ensuring that workbooks continue to calculate as expected, Microsoft has introduced a new feature called **Compatibility Versions**. These are set per workbook. Version 1 reflects historical calculation behaviour whilst Version 2 contains the improvements to text functions described above.

To change a workbook's Compatibility Version, select **Formulas -> Calculation Options -> Compatibility Version**:



COMPATIBILITY VERSIONS FOR EXISTING FILES

Existing spreadsheets will be set to Version 1, so their calculations will remain consistent. Once a spreadsheet's version has been set, it will not change unless you change it in through the menu described above.

COMPATIBILITY VERSIONS FOR NEW FILES

As of today, new workbooks will be set to Version 1, but this will change to Version 2 after a transition period while it rolls out to all Microsoft 365 subscription users (January 2026 for Current Channel).

The version that is currently being set on new workbooks is called the **Recommended Version**. It should be considered the best option for most users and scenarios, because it will have the most improvements, yet be accessible by all Microsoft 365 subscription users.

If you collaborate with others who are using non-Microsoft 365 Excel versions that predate this functionality (Excel 2024 and earlier), you can set the compatibility level to Version 1 to ensure consistency. Workbooks set to Version 1 will continue to calculate as they always have.

If a user opens a spreadsheet set to a version they do not have access to that contains functions that have changed, a warning will be shown indicating that calculation results may differ. Otherwise, the file will behave normally.

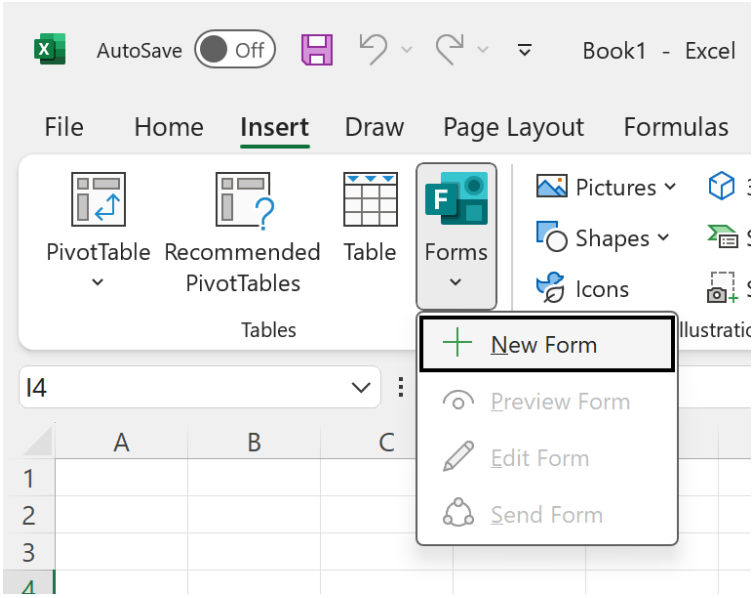
Versions are cumulative, so include all changes from prior versions. For example, a future Version 3 will include the text function changes introduced by Version 2.

This is the current state of play:

Version	Functions with changes	Description of changes	Excel versions that have access
Version 1 (Recommended)	N/A (Compatible with legacy Excel)		All versions
Version 2	LEN, MID, FIND, SEARCH and REPLACE functions	Support for Unicode surrogate pairs	Excel for M365

New Forms integration for Windows

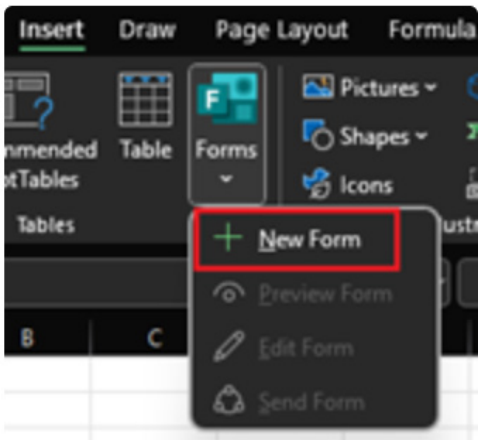
With the latest update, you can now create a new Microsoft Form directly within an Excel workbook for Windows, linking the form to the workbook. This feature, previously available in Excel for the web, is now available in the Windows app, with Mac support coming soon.



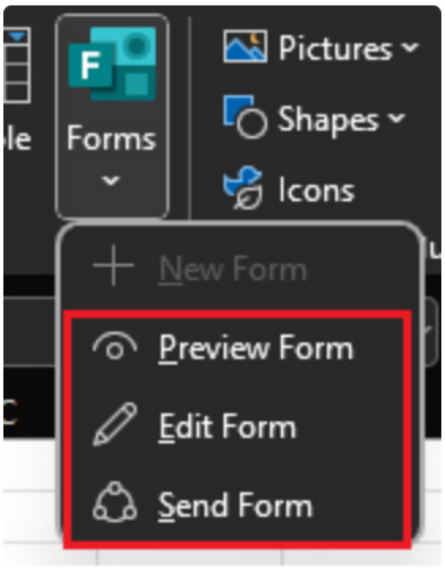
This new integration is particularly beneficial for users who frequently collect data from other people. By enabling the creation of a form directly from an Excel workbook, you can streamline the data collection process, making it more efficient and less prone to errors. Whether you are gathering survey responses, feedback or any other form of data, this feature ensures that all information is centrally located and easily accessible.

To access it:

- On the Insert tab, select Forms -> New Form:



- A new browser tab will open, presenting you with a blank form to customise according to your needs. Simultaneously, a linked Table will appear in your workbook, ready to synchronise with the form. As you edit the form or receive new submissions, the linked Table in your workbook will automatically update, ensuring that your data is always current and accurate
- After creating your form, you can access several other capabilities from the Forms menu:

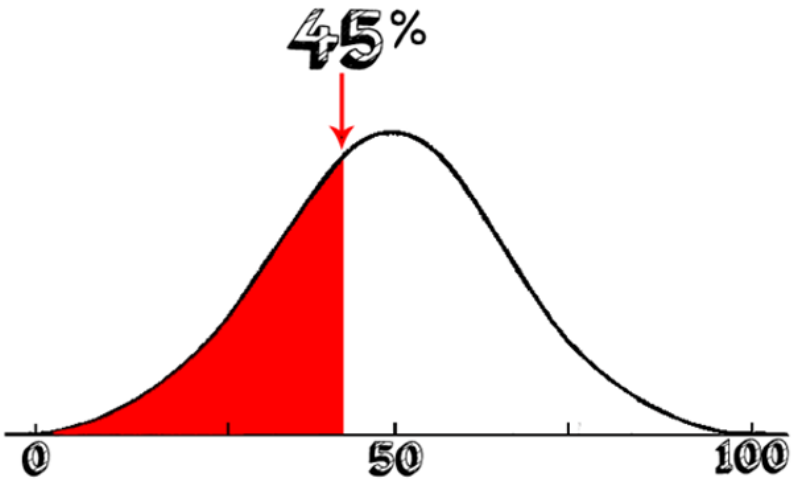


  - **Preview Form:** this option will open the form in preview mode in a new browser tab, allowing you to see how the form will look to respondents
  - **Edit Form:** this option will open the form in edit mode in a new browser tab, so you can make changes to the form as needed
  - **Send Form:** this option will open the form in a new browser tab and show the dialog letting you send the form out to respondents and begin collecting responses.

The new Microsoft Forms integration is available to all Current Channel users running Version 2410 (Build 16.0.18227.20000) or later. Mac users can also look forward to this integration in an upcoming update.

Until next month.

New Features for Excel



The **n**th percentile of a dataset is the value that cuts off the first **n**% of the data values when all of the values are sorted from least to greatest. For example, the 90th percentile of a dataset is the value that segregates the bottom 90% of the data values from the top 10% of data values.

There are three [3] distinct functions you can use to calculate percentiles in Excel:

1. **PERCENTILE.EXC:** this function returns the **k**th percentile of a dataset, excluding the values zero [0] and one [1]
2. **PERCENTILE.INC:** this function returns the **k**th percentile of a dataset, including the values zero [0] and one [1]
3. **PERCENTILE:** this function returns the **k**th percentile of a dataset as well. It will return the same value as the **PERCENTILE.INC** function.

In almost all cases, it is probably best to use the **PERCENTILE.INC** function because this function includes the values zero [0] and one [1] when calculating the percentiles and is the function that effectively "replaces" **PERCENTILE**.

No matter which of the three functions you use to calculate percentiles, the difference between the values calculated by **PERCENTILE.EXC** and **PERCENTILE.INC** will be similar in most cases. In some instances, it's even possible that the two functions will return precisely the same values depending upon the numbers analysed.

The **PERCENTILE** function is a statistical function that calculates the value at a given percentile in a dataset. It considers the range between zero [0] and one[1] including the extreme values. This function is particularly useful when analysing data distributions and understanding the relative position of a specific value within a dataset. For example, you can decide to examine candidates who score above the 90th percentile.

The **PERCENTILE** function has the following syntax:

=PERCENTILE(array, k)

It contains two [2] arguments:

- **array** which is required and represents the array of values or range of data that defines relative standing
- **k** which is also required is a value $0 \le k \le 1$ (0..1 inclusive), which represents the percentile (or **k**th percentage).

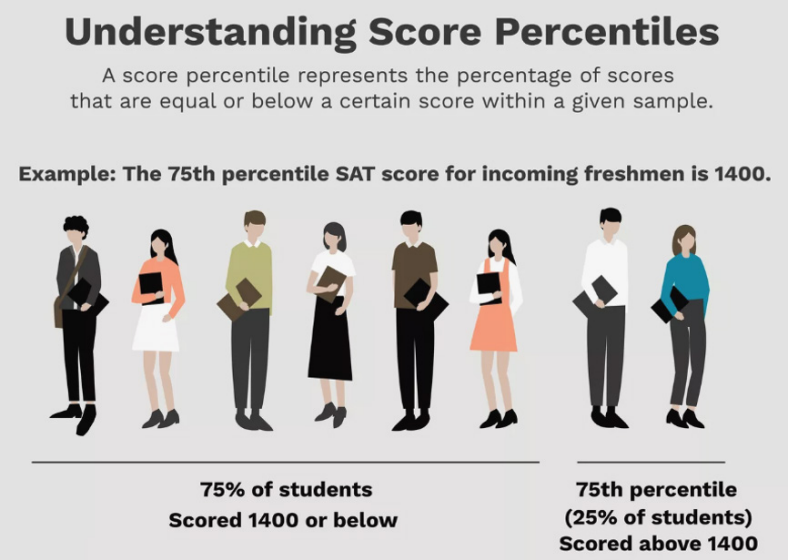
It should be noted that:

- this function has been replaced with one or more new functions that may provide improved accuracy and whose names better reflect their usage. Although this function is still available for backward compatibility, you should consider using the new functions from now on, because this function may not be available in future versions of Excel
- if **array** is empty, **PERCENTILE** returns the *#NUM!* error value
- if **k** is nonnumeric, **PERCENTILE** returns the *#VALUE!* error value
- if **k** is < 0 or if **k** > 1, **PERCENTILE** returns the *#NUM!* error value
- if **k** is not a multiple of **1/(n + 1)** (where **n** is the number of values in the **array**), **PERCENTILE** interpolates to determine the value at the **k**th percentile
- **PERCENTILE** will interpolate when the value for the specified percentile lies between two [2] values in the array. If it cannot interpolate for the percentile, **k** specified, Excel will return *#NUM!* error
- the **array** argument ignores text, logic, date and time values.

Please see our examples below:

	A	B	C
1	Data		
2	1		
3	2		
4	3		
5	4		
6	5		
7	6		
8	7		
9	8		
10	9		
11			
12			
13	Formula	Description	Result
14	=PERCENTILE(A2:A10, 0.25)	With 1 and 9 as the extreme values, there are eight values to relate in the range. 25% represents the second value in the range in order, which is 3	3.00
15	=PERCENTILE(A2:A10, 0)	With 1 and 9 as the extreme values, the zeroth percentile is the first value when ordered	1.00
16	=PERCENTILE(A2:A10, 0.01)	Here, interpolation is used to determine the value reported	1.08
17	=PERCENTILE(A2:A10, 10)	Since the percentile is greater than one [1], Excel returns the *#NUM!* error	#NUM!

The A to Z of Excel Functions: PERCENTILE.EXC



The **PERCENTILE.EXC** function is a statistical function that calculates the value at a given percentile in a dataset. It considers the range between zero [0] and one [1] excluding the extreme values. This function is particularly useful when analysing data distributions and understanding the relative position of a specific value within a dataset.

The **PERCENTILE.EXC** function has the following syntax:

=PERCENTILE.EXC(array, k)

It contains two [2] arguments:

- **array** which is required and represents the array of values or range of data that defines relative standing
- **k** which is also required is a value 0 < **k** < 1 (0..1 exclusive), which represents the percentile (or **k**th percentage).

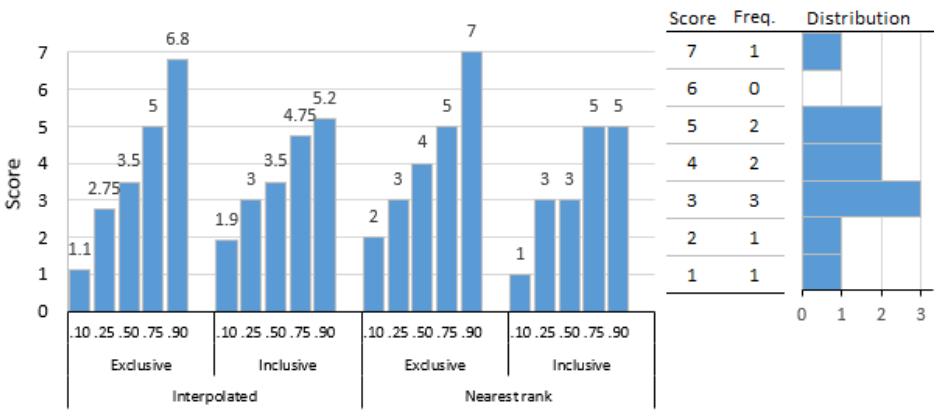
It should be noted that:

- if **array** is empty, **PERCENTILE.EXC** returns the *#NUM!* error value
- if **k** is nonnumeric, **PERCENTILE.EXC** returns the *#VALUE!* error value
- if **k** is ≤ 0 or if **k** ≥ 1, **PERCENTILE.EXC** returns the *#NUM!* error value
- if **k** is not a multiple of **1/(n + 1)** (where **n** is the number of values in the **array**), **PERCENTILE.EXC** interpolates to determine the value at the **k**th percentile
- **PERCENTILE.EXC** will interpolate when the value for the specified percentile lies between two [2] values in the array. If it cannot interpolate for the percentile, **k** specified, Excel will return *#NUM!* error
- the **array** argument ignores text, logic, date and time values.

Please see our examples below:

	A	B	C
1	Data		
2	1		
3	2		
4	3		
5	4		
6	5		
7	6		
8	7		
9	8		
10	9		
11			
12			
13	Formula	Description	Result
14	=PERCENTILE.EXC(A2:A10, 0.25)	Interpolates when the value for the specified percentile lies between two values in the array	2.5
15	=PERCENTILE.EXC(A2:A10, 0)	Since it cannot interpolate for the specified percentile, Excel returns the *#NUM!* error	#NUM!
16	=PERCENTILE.EXC(A2:A10, 0.01)	Since it cannot interpolate for the specified percentile, Excel returns the *#NUM!* error	#NUM!
17	=PERCENTILE.EXC(A2:A10, 10)	Since the percentile is greater than one [1], Excel returns the *#NUM!* error	#NUM!

The A to Z of Excel Functions: PERCENTILE.INC



The **PERCENTILE.INC** function is a statistical function that calculates the value at a given percentile in a dataset. It considers the range between zero [0] and one [1] including the extreme values. This function is particularly useful when analysing data distributions and understanding the relative position of a specific value within a dataset. For example, you can decide to examine candidates who score above the 90th percentile.

The **PERCENTILE.INC** function has the following syntax:

=PERCENTILE.INC(array, k)

It contains two [2] arguments:

- **array** which is required and represents the array of values or range of data that defines relative standing
- **k** which is also required is a value 0 ≤ **k** ≤ 1 (0..1 inclusive), which represents the percentile (or **k**th percentage).

It should be noted that:

- if **array** is empty, **PERCENTILE.INC** returns the *#NUM!* error value
- if **k** is nonnumeric, **PERCENTILE.INC** returns the *#VALUE!* error value
- if **k** is < 0 or if **k** > 1, **PERCENTILE.INC** returns the *#NUM!* error value
- if **k** is not a multiple of **1/(n + 1)** (where **n** is the number of values in the **array**), **PERCENTILE.INC** interpolates to determine the value at the **k**th percentile
- **PERCENTILE.INC** will interpolate when the value for the specified percentile lies between two [2] values in the array. If it cannot interpolate for the percentile, **k** specified, Excel will return *#NUM!* error
- the **array** argument ignores text, logic, date and time values.

Please see our final examples below:

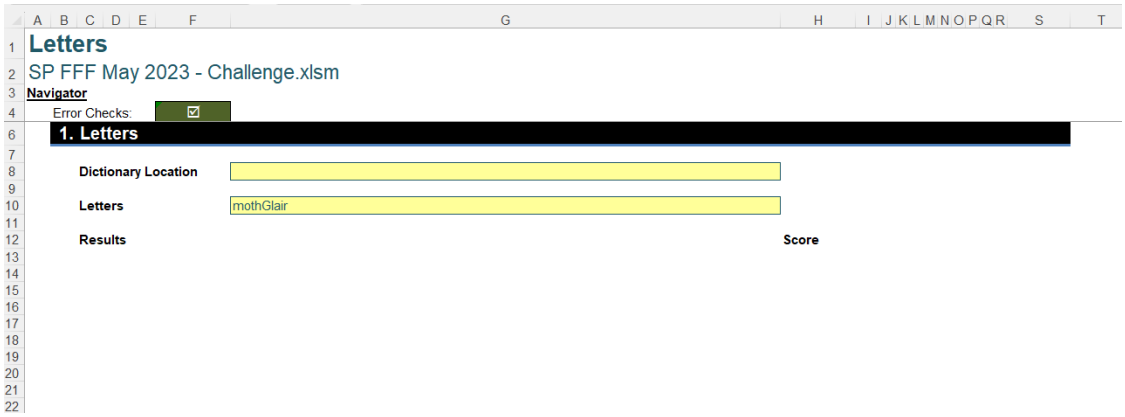
	A	B	C
1	Data		
2	1		
3	2		
4	3		
5	4		
6	5		
7	6		
8	7		
9	8		
10	9		
11			
12			
13	Formula	Description	Result
14	=PERCENTILE.INC(A2:A10, 0.25)	With 1 and 9 as the extreme values, there are eight values to relate in the range. 25% represents the second value in the range in order, which is 3	3.00
15	=PERCENTILE.INC(A2:A10, 0)	With 1 and 9 as the extreme values, the zeroth percentile is the first value when ordered	1.00
16	=PERCENTILE.INC(A2:A10, 0.01)	Here, interpolation is used to determine the value reported	1.08
17	=PERCENTILE.INC(A2:A10, 10)	Since the percentile is greater than one [1], Excel returns the *#NUM!* error	#NUM!

More Excel Functions next month.

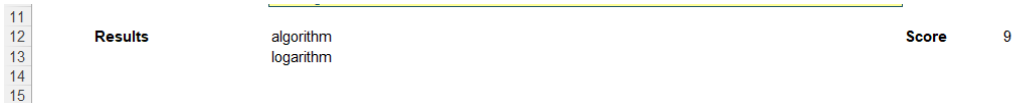
Beat the Boredom Suggested Solution

We dedicate this month's challenge to a certain member of the team who has a penchant for letters and numbers games…

This month, we had a hybrid Power Query and Excel challenge.



The challenge was to accept up to nine [9] input letters and find the longest word(s) that can be created from them. Only the longest word(s) should be returned in 'Results' and the 'Score' will be the length of the longest word. We have input some letters as an example. For this selection of letters, the 'Results' (for an English dictionary) would be:



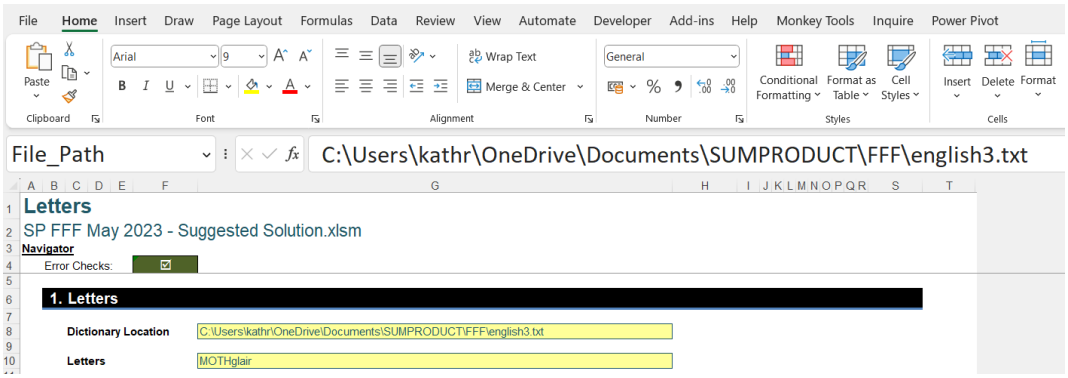
As always, there were some conditions:

- this is a Power Query challenge and Excel challenge – no VBA was required
- you may use a free dictionary source of your choice
- the solution should not be case-sensitive
- if no words can be found, the solution should indicate this.

Suggested Solution

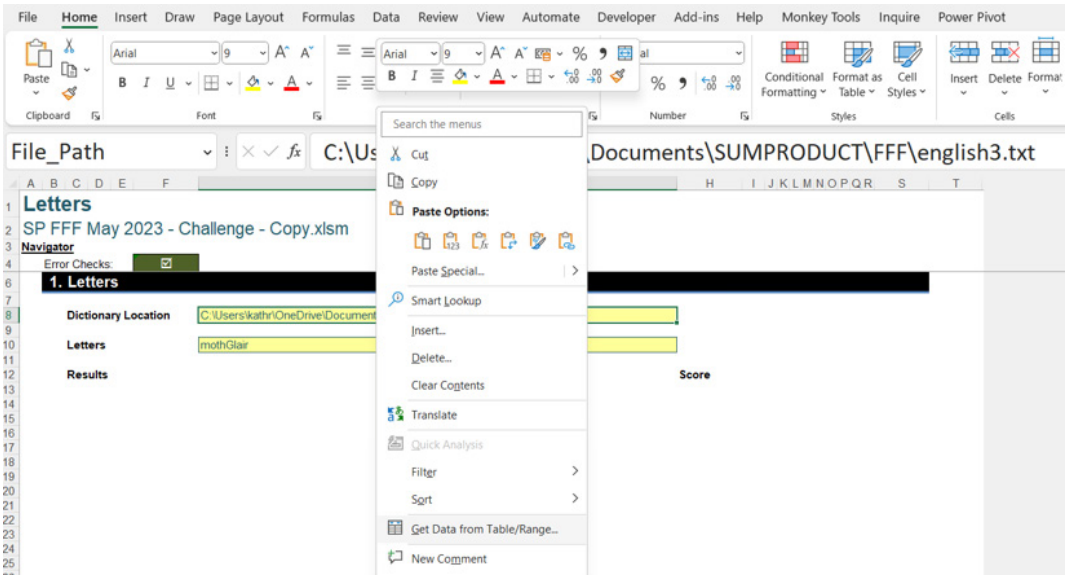
Sorry, this is quite a long, comprehensive solution! It may be divided into Power Query and Excel sections. We will use Power Query to prepare the dictionary and Excel to compare the input letters to the dictionary contents.

We have selected a free dictionary source which we have in a text file. We enter the location of the text file into cell **G10** on the Letters sheet. Note that this is also identified by a Named Range **File_Path**.

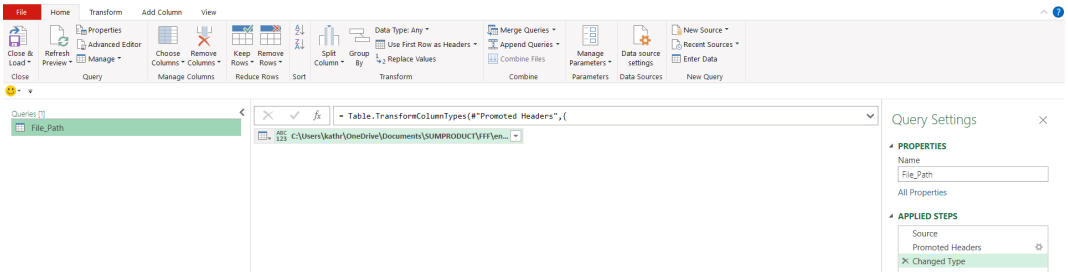


Using Power Query to Create the Dictionary Matrix

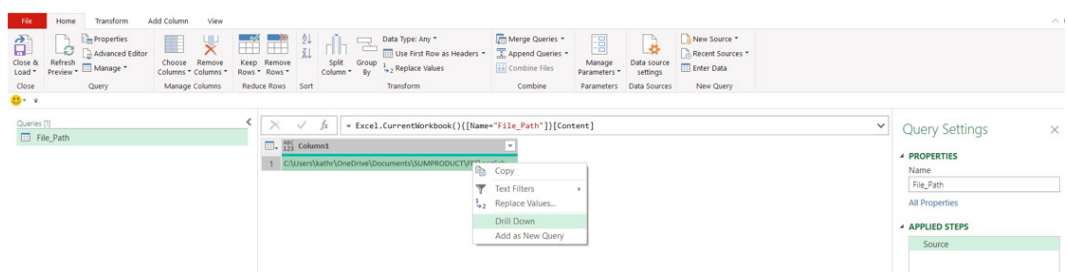
We start by extracting the **File_Path** information. We can click inside the cell, and right-click to choose 'Get Data from Table/Range':



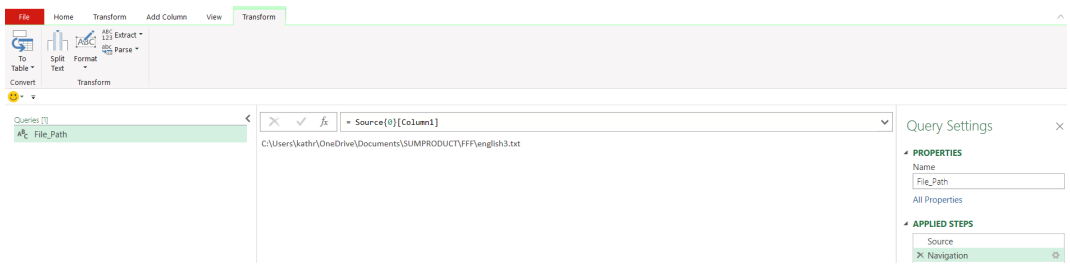
This takes us to the Power Query editor:



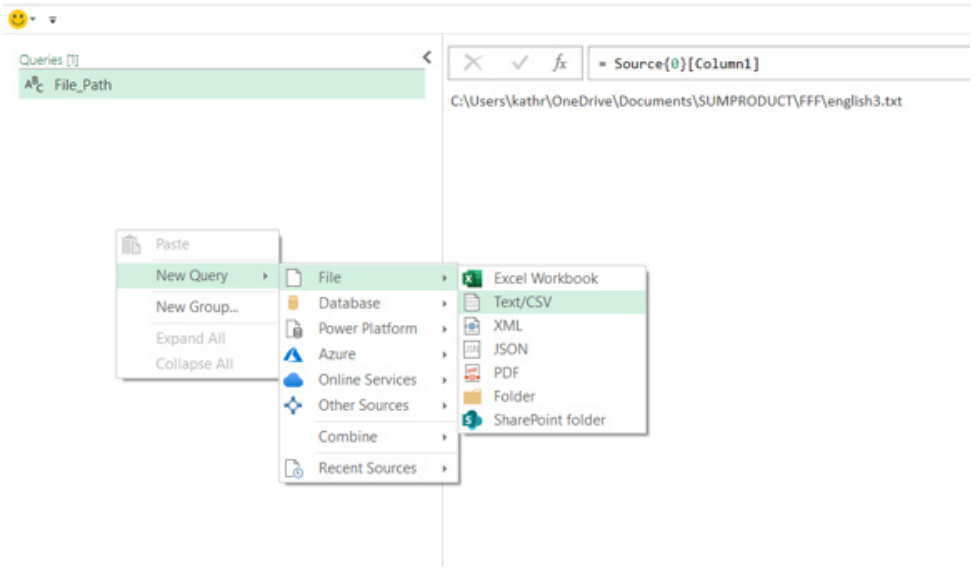
Power Query has generated a 'Promoted Headers' and 'Changed Type' step: we don't need these, so we can delete them. We only need the value, which we can 'Drill Down' into:



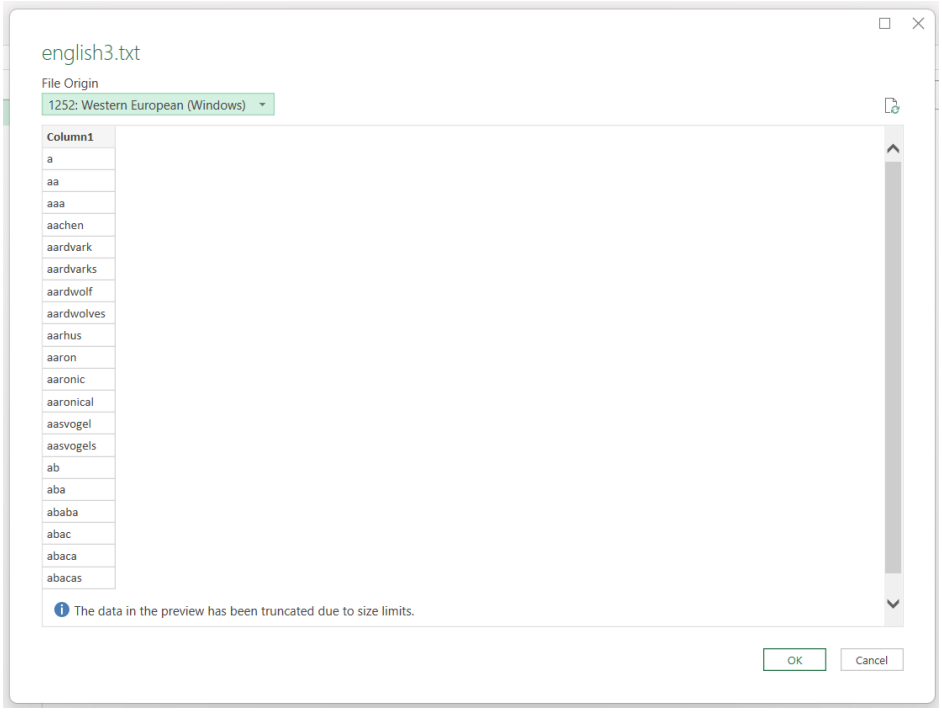
We now have our **File_Path** parameter:



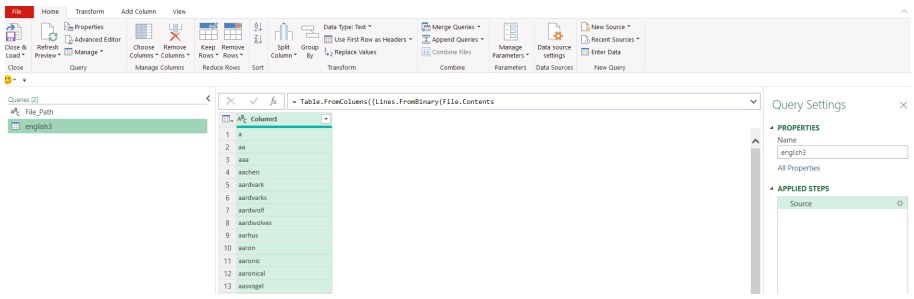
Now we need to extract the dictionary data. We can create a new query by right-clicking in the blank space in the Queries pane:



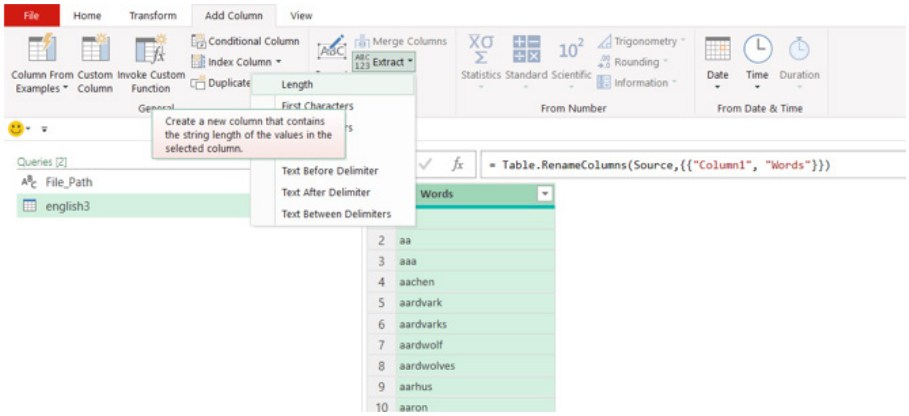
We select 'Text/CSV' from the File dropdown from the 'New Query' dropdown, navigate to the location of the dictionary file and choose to Import. The dialog that appears will look something like the following image:



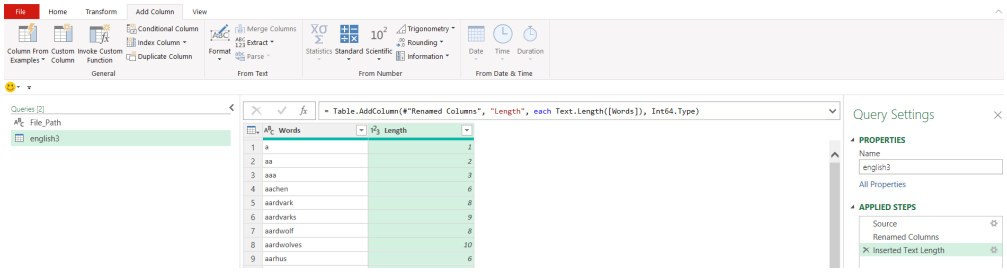
Since we are already in the Power Query editor, clicking OK will allow us to transform the data:



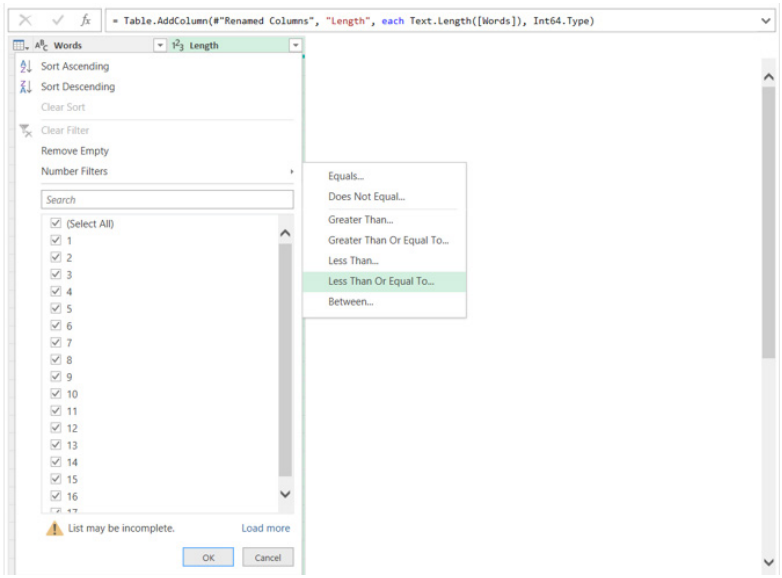
If your dictionary has extra columns, then choose the column containing the words and right-click to 'Remove Other Columns'. We rename **Column1** to **Words**. We need to filter out words that have more than nine [9] letters. On the 'Add Column' tab, we can choose to Extract Length from the 'From Text' section:



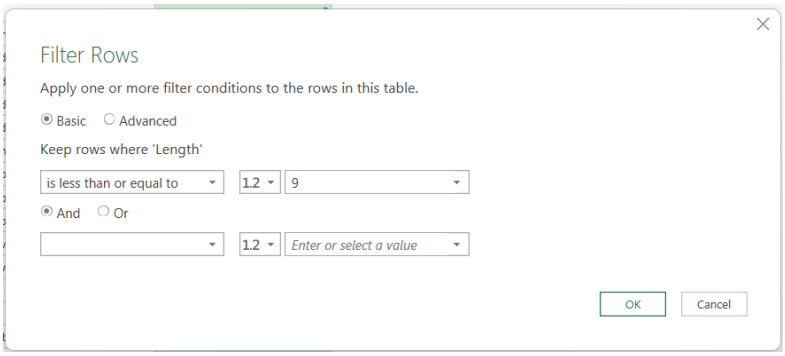
This gives us column **Length**:



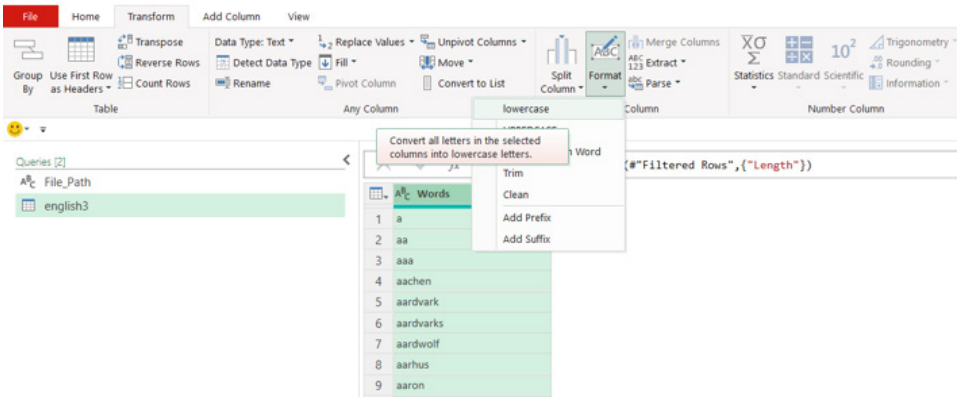
On the filter dialog, we have the option to choose values 'Less Than Or Equal To…':



Choosing this option triggers a dialog, where we 'Keep rows where 'Length'' 'is less than or equal to' 9.



Now we have reduced the data, we can remove the **Length** column. We access the Format dropdown on the Transform tab and transform the data in **Words** to lowercase:



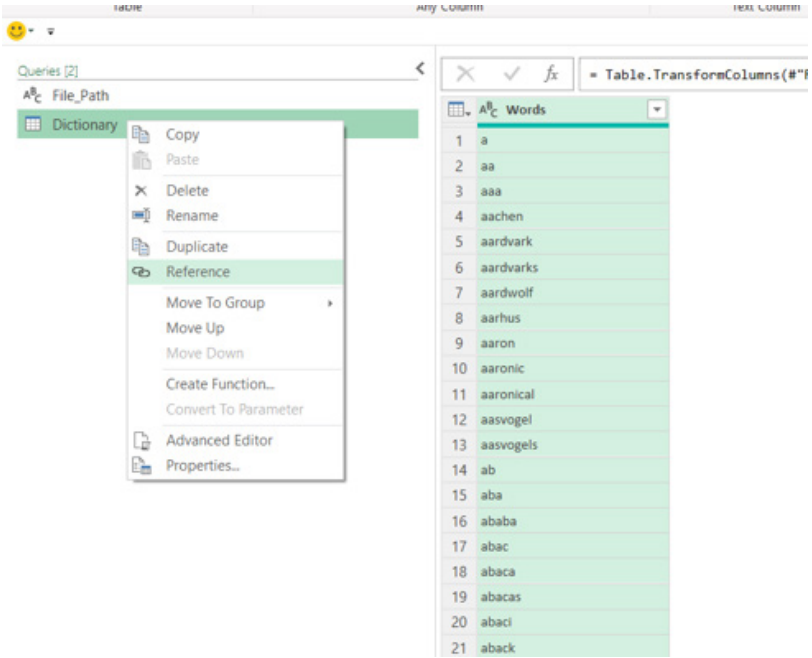
Now we have completed the transformations, we need to ensure that the source of this query will change if the user changes the location of the dictionary file. The 'Source' step currently looks something like this:

= Table.FromColumns({Lines.FromBinary(File.Contents("C:\yourfilename"), null, null, 1252)})

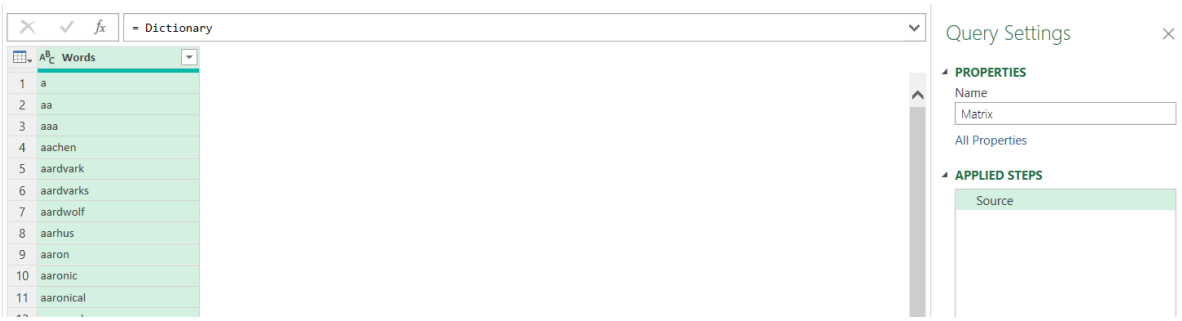
We need to change this to use the **File_Path** parameter:

= Table.FromColumns({Lines.FromBinary(File.Contents(File_Path), null, null, 1252)})

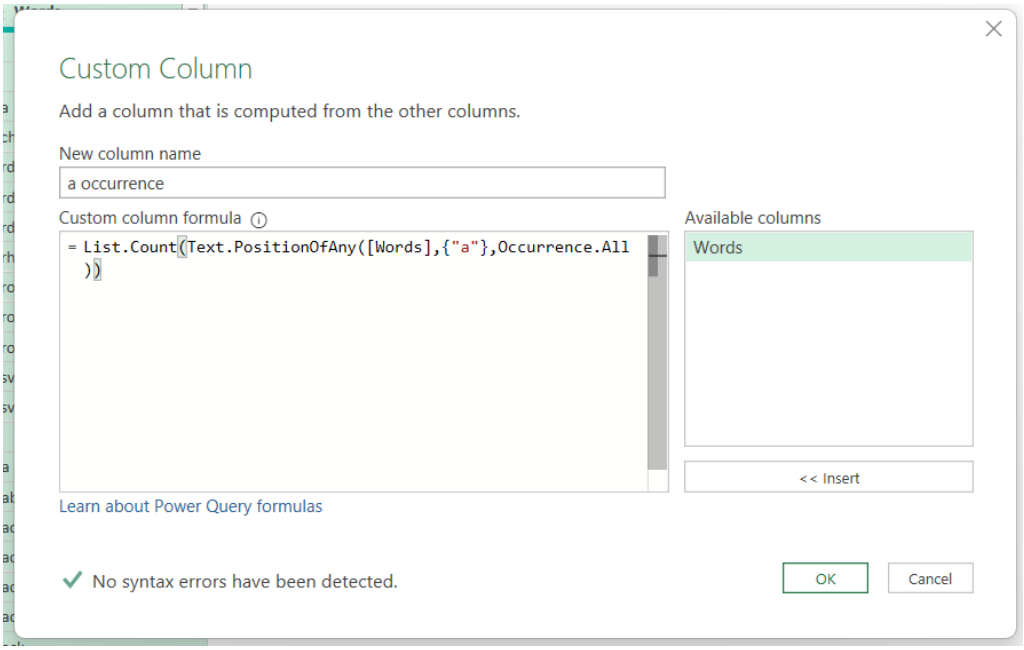
Now the query is complete, we rename it to **Dictionary** and take a Reference copy:



The query we create will use **Dictionary** as its source. We rename it **Matrix**:



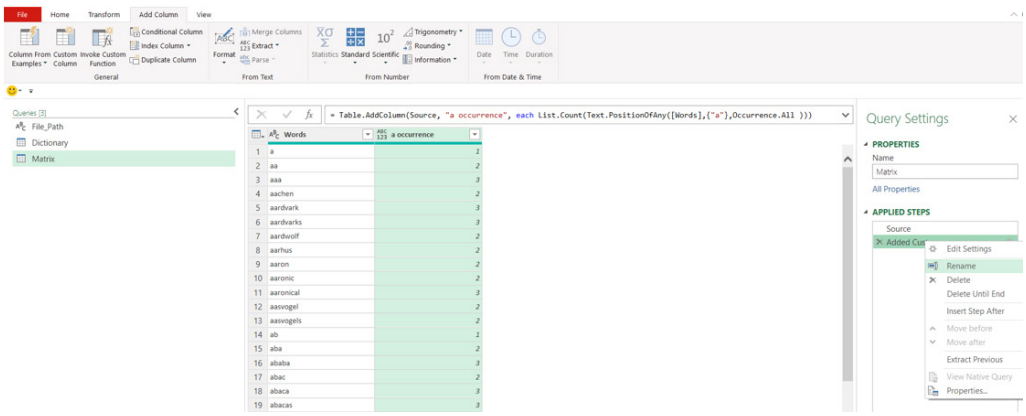
As the name suggests, we are going to create a matrix. We want to know the occurrence of each letter in each word. I will show the long version of this, and then a more efficient version. For the long version, we need to create a 'Custom Column' from the 'Add Column' tab for each letter:



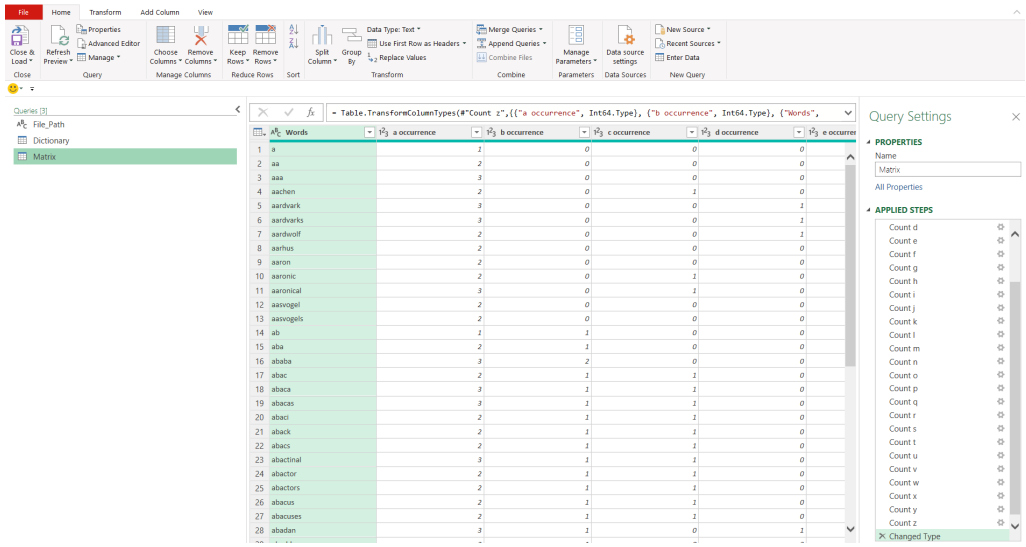
The **M** code used is:

= List.Count(Text.PositionOfAny([Words],{"a"},Occurrence.All))

This creates a list of the positions of each occurrence of 'a' in the word and then counts the values in the list. We can create similar code for the other columns. We can then give each step a more sensible name as we go:



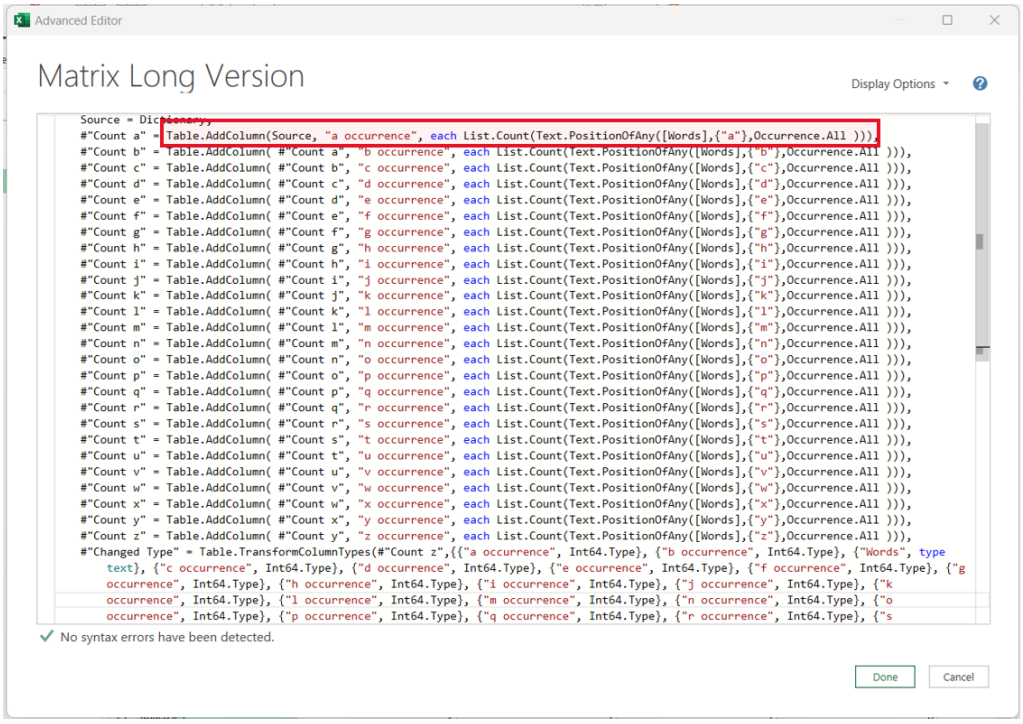
This is the rather long result, including a 'Changed Type' step at the end to change the data types of all the new columns to a 'Whole Number':



We rename this query **Matrix Long Version**, and take another reference copy of **Dictionary**, which we will call **Matrix**. This time we're going to create all the columns at once, and we can do this by creating a record for each row. The function we will use is **Table.FromRecords**:

Table.FromRecords(records as list**, optional columns** as any, **optional missingField** as nullable number**)** as table

If we consider the **M** code in the 'Advanced Editor' (accessible from the Home tab) for **Matrix Long Version**, we can see that to add each column, Power Query uses the function **Table.AddColumn**:



Instead of the code we are using here:

Table.AddColumn(Source, "a occurrence", each List.Count(Text.PositionOfAny([Words],{"a"},Occurrence.All))

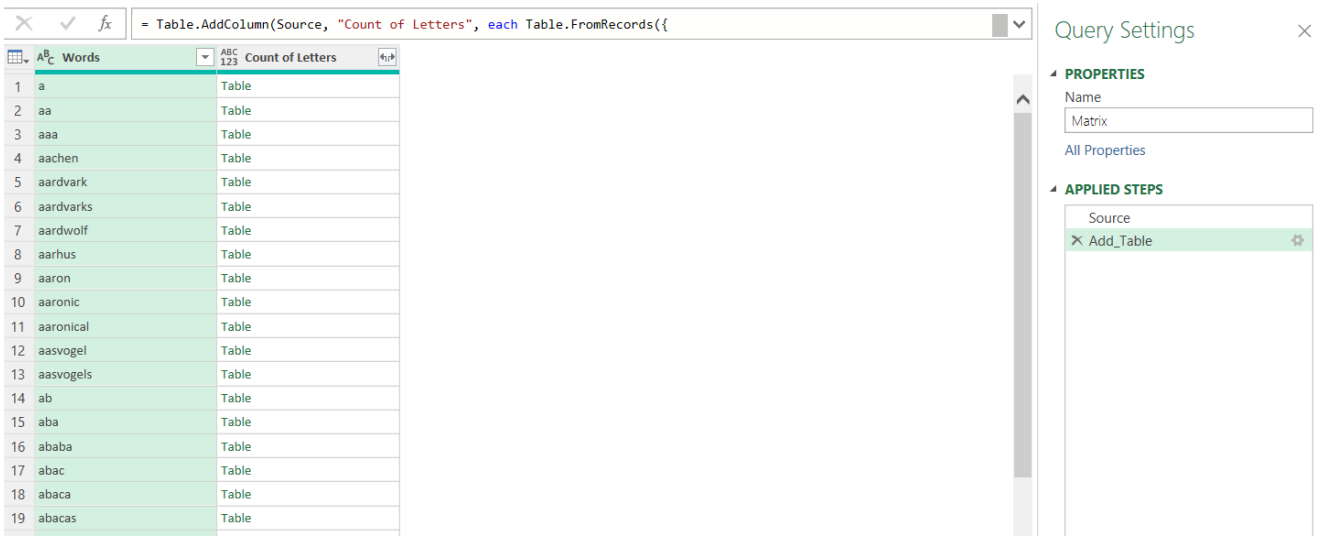
In the new query, we want to create a record for the code after '**each**'. The record will contain *all* of the columns:

```
Table.FromRecords({
  [a occurrence = List.Count(Text.PositionOfAny([Words],{"a"},Occurrence.All )),
   b occurrence = List.Count(Text.PositionOfAny([Words],{"b"},Occurrence.All )),
   c occurrence = List.Count(Text.PositionOfAny([Words],{"c"},Occurrence.All )),
   …
   z occurrence = List.Count(Text.PositionOfAny([Words],{"z"},Occurrence.All ))]
                                                                              })
```

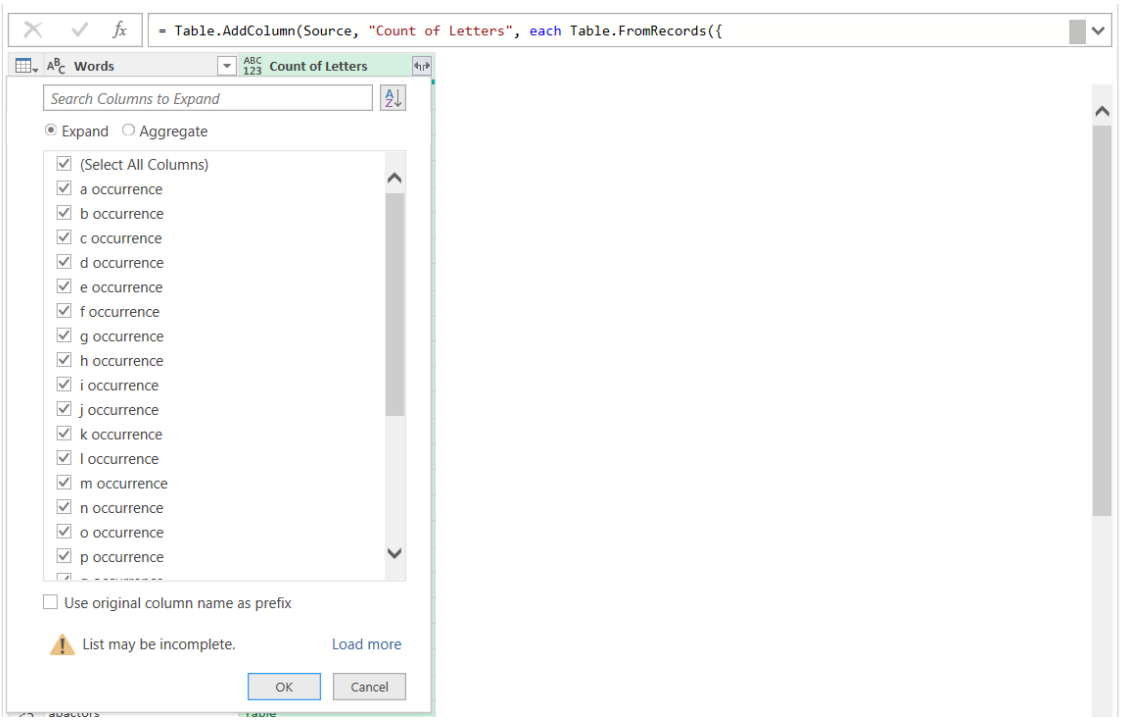
If we combine these we get:

```
let
   Source = Dictionary,
   Add_Table = Table.AddColumn(Source, "Count of Letters", each Table.FromRecords({
   [a occurrence = List.Count(Text.PositionOfAny([Words],{"a"},Occurrence.All )),
    b occurrence = List.Count(Text.PositionOfAny([Words],{"b"},Occurrence.All )),
    c occurrence = List.Count(Text.PositionOfAny([Words],{"c"},Occurrence.All )),
    d occurrence = List.Count(Text.PositionOfAny([Words],{"d"},Occurrence.All )),
    e occurrence = List.Count(Text.PositionOfAny([Words],{"e"},Occurrence.All )),
    f occurrence = List.Count(Text.PositionOfAny([Words],{"f"},Occurrence.All )),
    g occurrence = List.Count(Text.PositionOfAny([Words],{"g"},Occurrence.All )),
    h occurrence = List.Count(Text.PositionOfAny([Words],{"h"},Occurrence.All )),
    i occurrence = List.Count(Text.PositionOfAny([Words],{"i"},Occurrence.All )),
    j occurrence = List.Count(Text.PositionOfAny([Words],{"j"},Occurrence.All )),
    k occurrence = List.Count(Text.PositionOfAny([Words],{"k"},Occurrence.All )),
    l occurrence = List.Count(Text.PositionOfAny([Words],{"l"},Occurrence.All )),
    m occurrence = List.Count(Text.PositionOfAny([Words],{"m"},Occurrence.All )),
    n occurrence = List.Count(Text.PositionOfAny([Words],{"n"},Occurrence.All )),
    o occurrence = List.Count(Text.PositionOfAny([Words],{"o"},Occurrence.All )),
    p occurrence = List.Count(Text.PositionOfAny([Words],{"p"},Occurrence.All )),
    q occurrence = List.Count(Text.PositionOfAny([Words],{"q"},Occurrence.All )),
    r occurrence = List.Count(Text.PositionOfAny([Words],{"r"},Occurrence.All )),
    s occurrence = List.Count(Text.PositionOfAny([Words],{"s"},Occurrence.All )),
    t occurrence = List.Count(Text.PositionOfAny([Words],{"t"},Occurrence.All )),
    u occurrence = List.Count(Text.PositionOfAny([Words],{"u"},Occurrence.All )),
    v occurrence = List.Count(Text.PositionOfAny([Words],{"v"},Occurrence.All )),
    w occurrence = List.Count(Text.PositionOfAny([Words],{"w"},Occurrence.All )),
    x occurrence = List.Count(Text.PositionOfAny([Words],{"x"},Occurrence.All )),
    y occurrence = List.Count(Text.PositionOfAny([Words],{"y"},Occurrence.All )),
    z occurrence = List.Count(Text.PositionOfAny([Words],{"z"},Occurrence.All ))]}))
in Add_Table
```

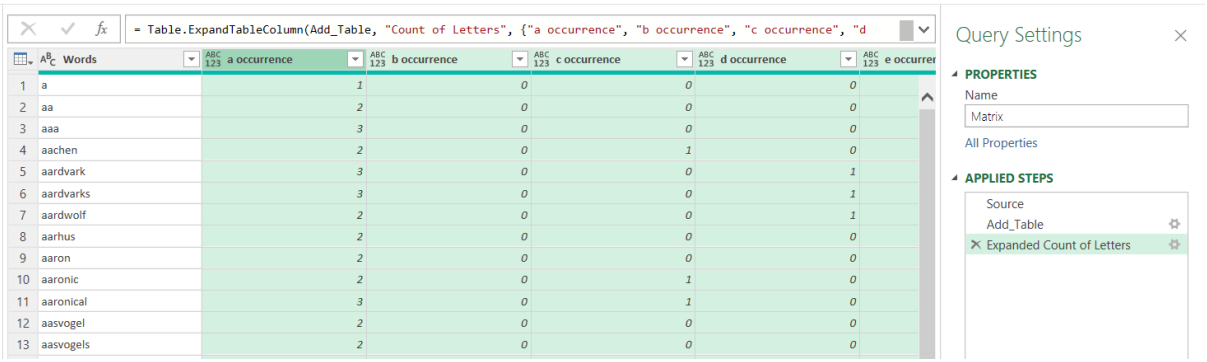
If we put this in the Advanced Editor for **Matrix**, we get a column of Tables:



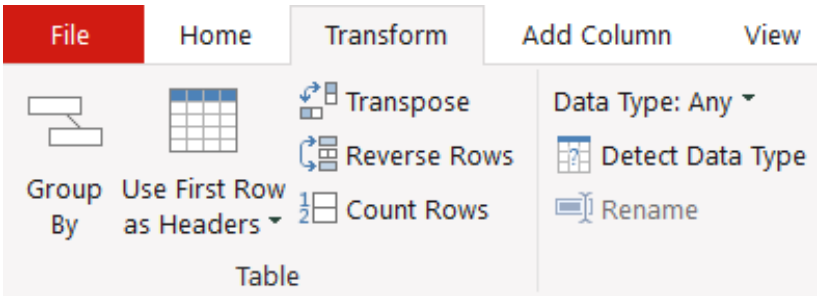
We may now expand the records by using the icon next to the heading of **Count of Letters**:



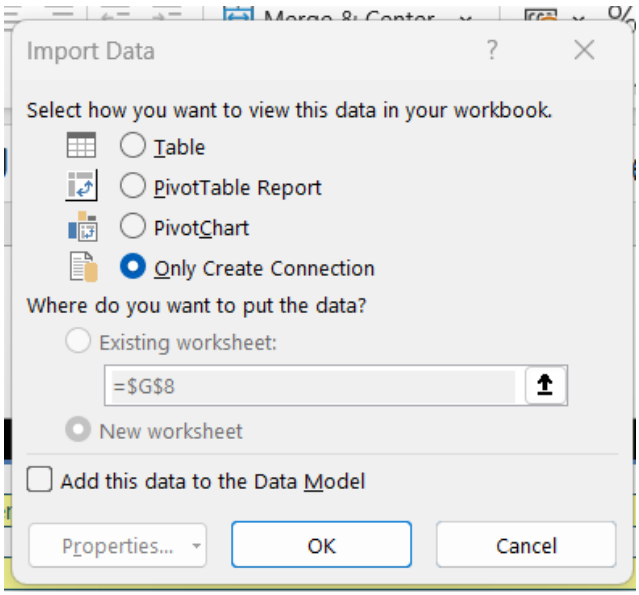
We choose not to 'Use original column name as prefix' and click 'OK':



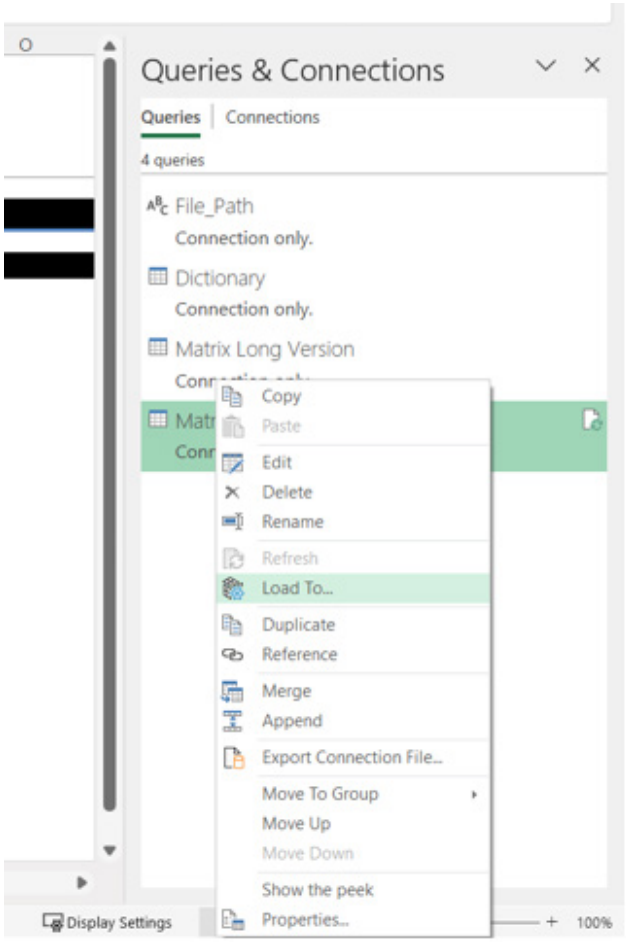
All the columns have been created in one step. We choose to set the data type for all the columns to 'Whole Number', which we may undertake from the Transform Tab either by choosing the type, or using the 'Detect Data Type' option:



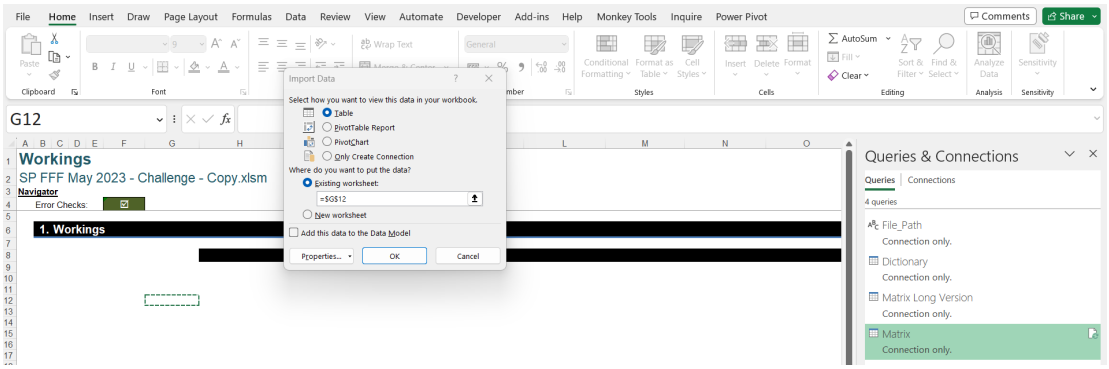
Finally, let's choose to 'Close & Load To', in order to specify which queries we wish to load and where we want to load them to. Initially, we choose to 'Only Create Connection':



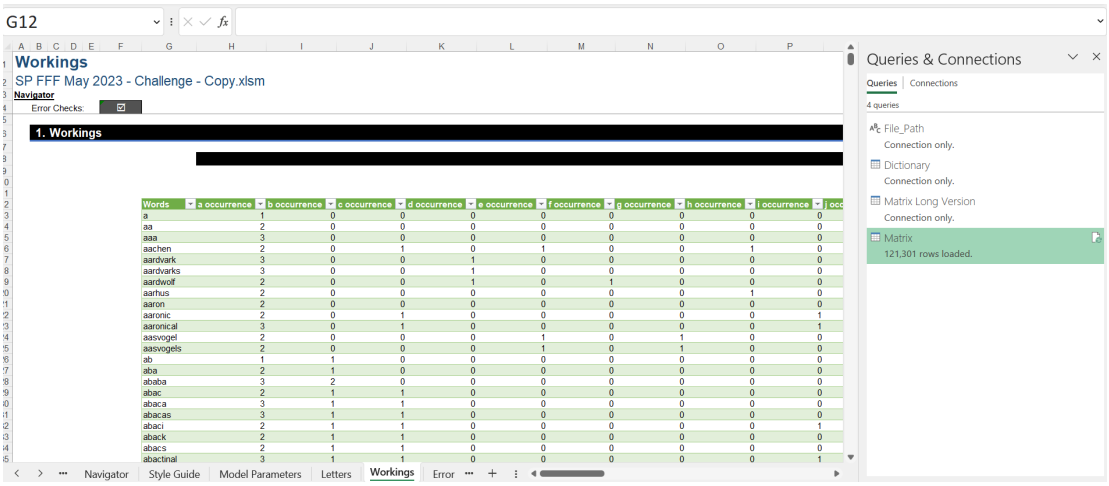
We may then change the setting for the **Matrix** query by selecting it, and right-clicking to access 'Load To…'



This time, we choose to load to a Table on an 'Existing worksheet' and choose cell **G12** on sheet Workings:



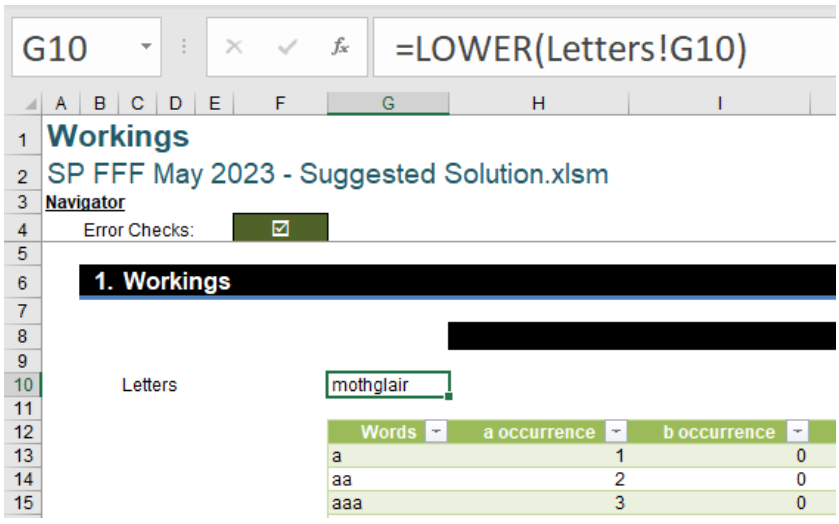
The dictionary data is loaded.



Now we need to compare the input letters with the dictionary. This will be done with Excel formulae.

Using Excel to Calculate the Matching Words

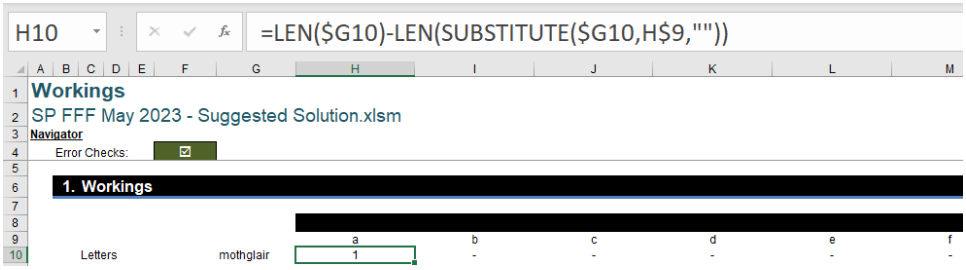
Once the resultant Table has been loaded to Excel, the first step is to bring in the input set of letters. We make use of the **LOWER** function to convert any uppercase letters to lowercase:



We're going to need to count the number of occurrences of each letter to compare to the matrix generated in Power Query. Creating headings in **row 9** for each of the letters (in our case across 26 columns) will allow us to make use of the following formula:

=LEN($G10)-LEN(SUBSTITUTE($G10,H$9,""))

This will take the input text string, substitute any letters equal to the column heading with blanks (effectively removing them) and then calculate the length of the resulting text string. The length of this altered text string is subtracted from the length of the input text string, returning the number of occurrences of the letter.

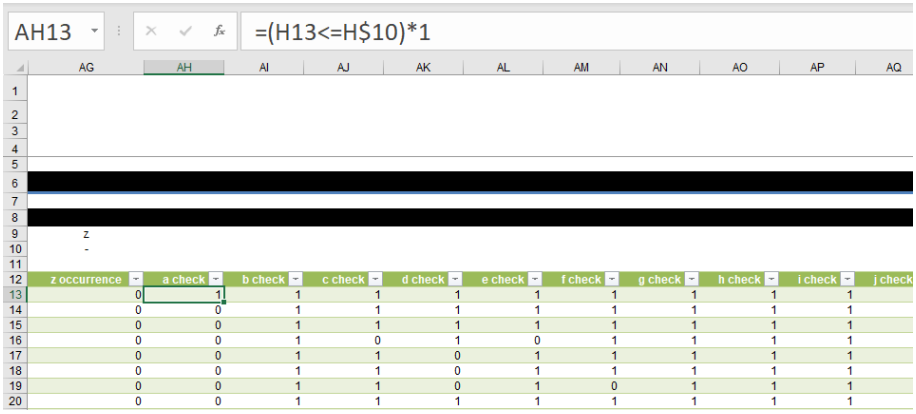


We now need to check if each word in our list can be created using only the letters from the input text string. To do this we check if the number of occurrences of each letter in the word is less than or equal to the number of occurrences of the letter in our input text string. We use the following formula starting in cell **AH13** as our "a check":

=(H13<=H$10)*1

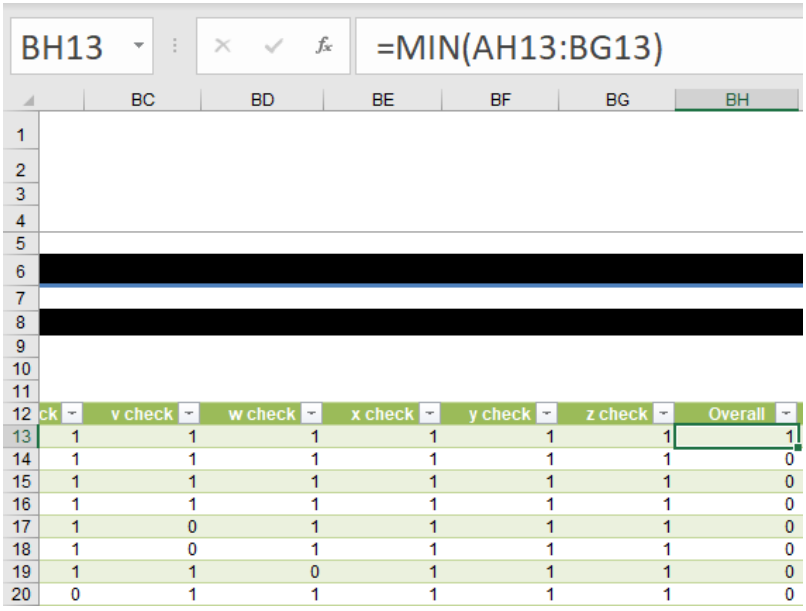
We multiple by one [1] to convert values of TRUE or FALSE to one or zero [0] respectively.

This check must be performed for each letter in each word:



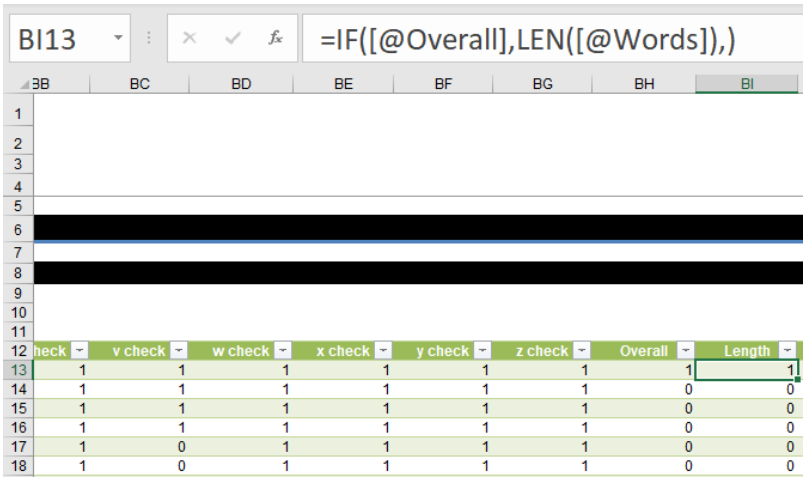
z occurrence	a check	b check	c check	d check	e check	f check	g check	h check	i check	j check
0	1	1	1	1	1	1	1	1	1	
0	0	1	1	1	1	1	1	1	1	
0	0	1	1	1	1	1	1	1	1	
0	0	1	0	1	0	1	1	1	1	
0	0	1	1	0	1	1	1	1	1	
0	0	1	1	0	1	1	1	1	1	
0	0	1	1	0	1	0	1	1	1	
0	0	1	1	1	1	1	1	1	1	

We will need all of these checks to return true for a word to be valid, so the next step is to build an overall check. A simple way to perform an **AND** check here is to make use of the **MIN** function across this range of checks from "a check" to "z check":



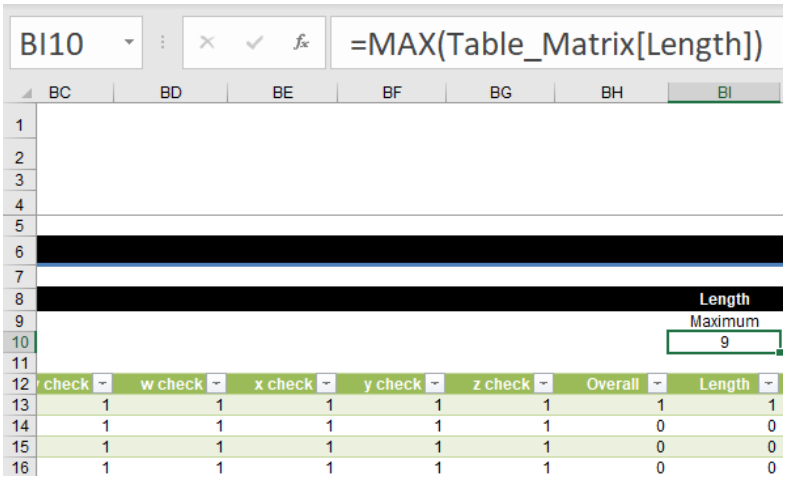
ck	v check	w check	x check	y check	z check	Overall
1	1	1	1	1	1	1
1	1	1	1	1	1	0
1	1	1	1	1	1	0
1	1	1	1	1	1	0
1	0	1	1	1	1	0
1	0	1	1	1	1	0
1	1	0	1	1	1	0
0	1	1	1	1	1	0

Now that we have a check that will return a one [1] for valid words and zero [0] for invalid words, let's calculate the length of each valid word. We can once again make use of the **LEN** function, performing it on each word only when the overall check is equal to one [1]:



heck	v check	w check	x check	y check	z check	Overall	Length
1	1	1	1	1	1	1	1
1	1	1	1	1	1	0	0
1	1	1	1	1	1	0	0
1	1	1	1	1	1	0	0
1	0	1	1	1	1	0	0
1	0	1	1	1	1	0	0

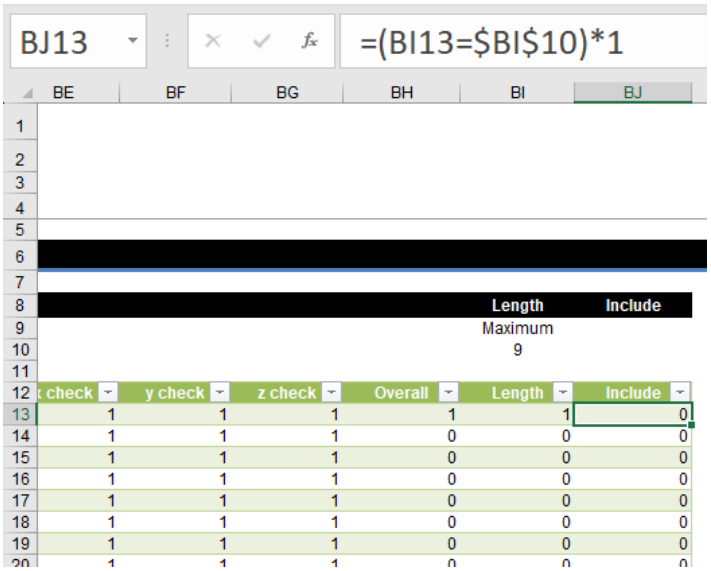
We will also need to know the length of the longest applicable word. This can be calculated by taking the maximum value of the column we have just generated through use of the **MAX** function:



check	w check	x check	y check	z check	Overall	Length
1	1	1	1	1	1	1
1	1	1	1	1	0	0
1	1	1	1	1	0	0
1	1	1	1	1	0	0

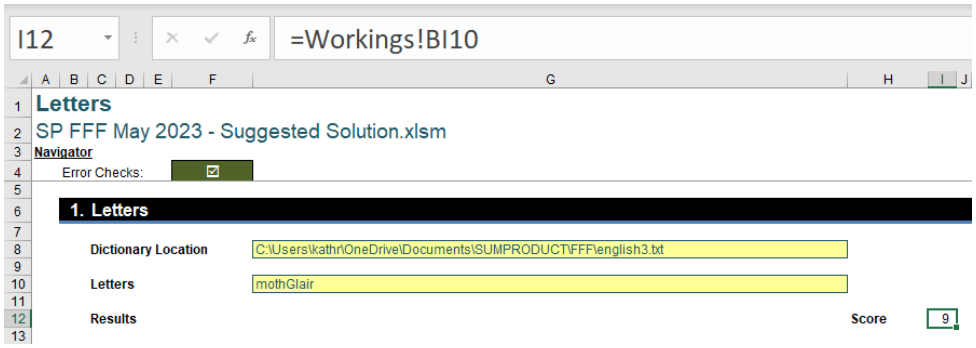
This value will be the 'Score', as well as being used in our final check. We need to check if the length of the word is equal to the maximum, as we only want to display the longest word(s) that can be made.

To do this, we check if the length calculated in the **Length** column of our table is equal to the maximum length calculated in cell **BI10**. As before, this is multiplied by one [1] to return a one [1] in place of TRUE and a zero [0] in place of FALSE.



check	y check	z check	Overall	Length	Include
1	1	1	1	1	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0

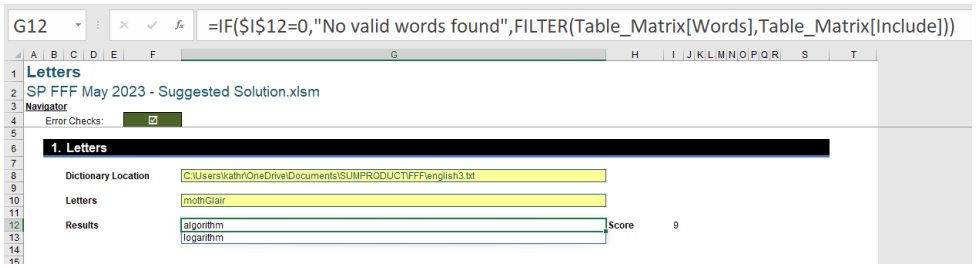
Finally, we can return to the **Letters** sheet and produce our results. We'll start by bringing the score into cell **I10**:



Now all that's left is to bring in the words. We've chosen to make use of one of Excel's newer functions, **FILTER**:

=IF($I$12=0, "No valid words found", FILTER(Table_Matrix[Words], Table Matrix[Include]))

First, this formula will check if the 'Score' is equal to zero [0]. If the 'Score' is equal to zero [0], then "No valid words found" will be returned. Otherwise, the **Words** column will be filtered based on the **Include** column, returning only words where this is equal to one [1]:



Of course, further improvements (such as adding data validation to cell **G10**) can be performed, but there you have it! This the real secret as to how Liam got his hands on the coveted teapot! *(Yeah, right… - Liam]*

Did you have a better solution? Let us know, we'd be glad to hear if there are better ways to work around this problem!

Upcoming SumProduct Training Courses

Location	Course	Course Date	Local TIme	UTC	Duration
Sydney Australia	Excel Tips and Tricks	10 March 2025	09:00 - 17:00 AEDT	09 March 2025 22:00 UTC - 10 March 2025 06:00 UTC	1 Day
Virtual (Australia)	ChatGPT Part 1	24 March 2025	09:00 - 12:30 AEDT	23 Mar 2025 22:00 UTC - 24 Mar 2025 01:30 UTC	1 Day
Sydney Australia	Financial Modelling	24 March 2025 - 25 March 2025	09:00 - 17:00 AEDT	23 March 2025 22:00 UTC - 25 March 2025 06:00 UTC	2 Days
Virtual (Australia)	ChatGPT Part 2	7 April 2025	09:00 - 12:30 AEDT	6 Apr 2025 22:00 UTC - 7 Apr 2025 01:30 UTC	1 Day
Sydney Australia	Power Pivot, Power Query and Power BI	10 April 2025 - 11 April 2025	09:00 - 17:00 AEDT	9 April 2025 22:00 UTC - 11 April 2025 06:00 UTC	2 Days

Key Strokes

Each newsletter, we'd like to introduce you to useful keystrokes you may or may not be aware of. This time, we thought we would regain **Alt**er the alphabet, but be warned – it's a bit of a blast from the past:

Keystroke	What it does
CTRL + SHIFT + A	Insert arguments in formula
CTRL + SHIFT + B	Address book (only when emailing not as an attachment)
CTRL + SHIFT + F	Font face (Format Cells)
CTRL + SHIFT + L	Toggle AutoFilter
CTRL + SHIFT + O	Select comments
CTRL + SHIFT + P	Font size
CTRL + SHIFT + T	Toggle Total Row
CTRL + SHIFT + U	Expand / collapse Formula bar

There are *c.*550 keyboard shortcuts in Excel. For a comprehensive list, please download our Excel file at http://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