# Should the Power Query tools change our approach to spreadsheet risk?

An initial assessment

Simon Hurst

# Contents

# 1 Introduction

## 1.1 The importance of Power Query

Because it radically changes the way in which an extensive range of spreadsheet operations can be performed, including those not directly connected to data acquisition, Power Query can how many spreadsheets are designed and constructed. This has the potential to affect all aspects of spreadsheet risk, including the likelihood of errors occurring; the methods available to avoid errors and the techniques available to identify and correct errors.

It is important to understand that Power Query is being used as far more than a replacement for the legacy Excel data acquisition tools. It would be more accurate to see Power Query as an alternative to the use of macros and Visual Basic for Applications (VBA) code for the automation of spreadsheet processes. As a result, it is important to consider whether Power Query has been used, or should have been used, when reviewing many types of spreadsheet and not just those that are obviously based on the analysis of external data sources.

## 1.2 Scope

This paper will consider ways in which using Power Query to replace 'legacy' cell-based Excel can change the risks associated with Excel calculations and processes; looking both at reducing some of the risks inherent in cell-based Excel and also the new types of risk that Power Query introduces. It will go on to consider how Power Query changes the ways in which spreadsheets need to be reviewed to identify errors and potential errors before considering whether Power Query has a role to play in the review of cell-based models. Finally, it will consider ways in which Power Query itself can be made less risky and easier to review. For the rest of this paper, cell-based Excel, without the use of the Power Query tools, will be referred to using the term: ‘legacy Excel’. The Power Query tools have been included within the Excel Get & Transform Ribbon tab (latterly the Get & Transform Data Ribbon tab) since the introduction of Excel 2016. These tools will be referred to using the term Power Query.

This is not an extensively researched paper and much of the evidence is personal and anecdotal. Each of the areas covered would warrant a detailed paper in its own right.

At this stage, the intention is to further the debate on whether the use of the Power Query tools changes the ways in which we seek to reduce spreadsheet risk and whether the use of Power Query should be promoted as a potentially more efficient and robust way to deal with a range of Excel processes, or whether Power Query increases spreadsheet risk without realising any substantial benefits.

The examples provided are far from comprehensive and definitive and should instead be viewed as an initial guide to some of the possibilities available.

In this paper, we will concentrate on the use of the Power Query interface to create steps rather than exploring the full capabilities of M code as a programming language in its own right.

## 1.3 Not Business Intelligence

This paper does not seek to consider Power Query when used primarily as the data acquisition and shaping element of a business intelligence solution, for which methodologies already exist, but instead looks at the use of the various Power Query tools to replace or augment standard spreadsheet processes.

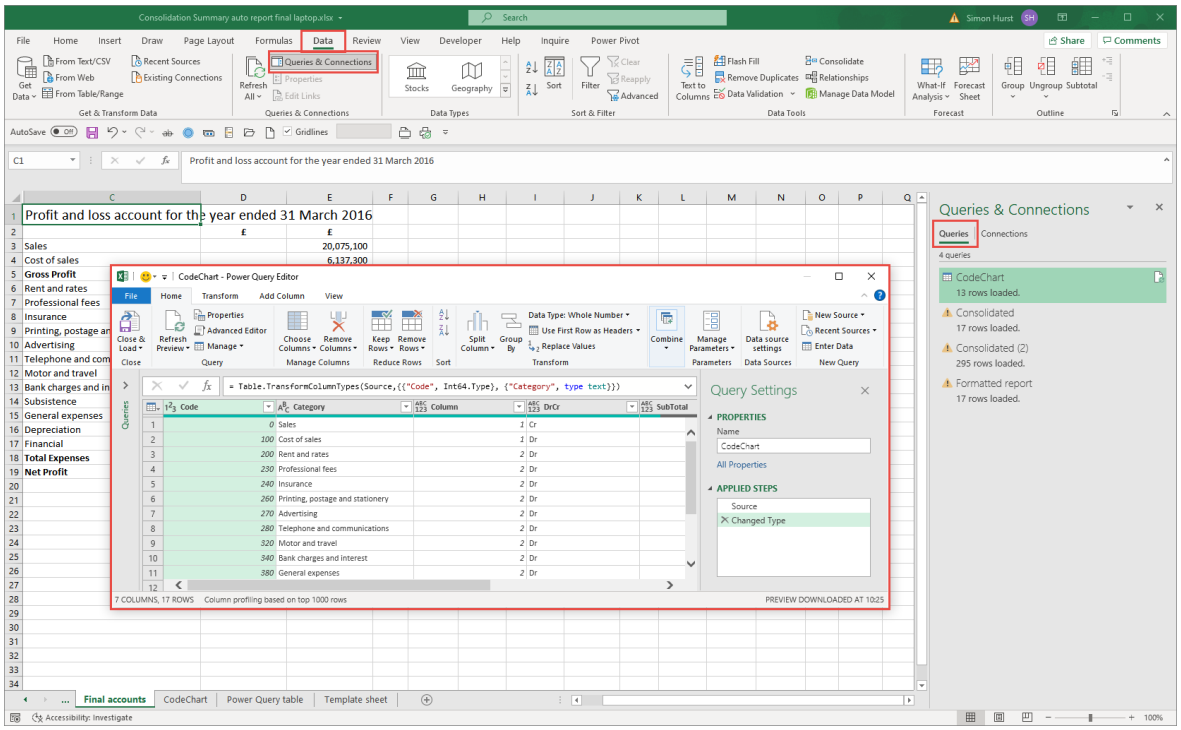
# 2 Power Query: a brief overview

Power Query was introduced in Excel 2010 as an optional add-in. In Excel 2016 it changed from being an optional add-in to become a built-in part of the Data Ribbon tab as the Get & Transform group of tools. In 2017 an update to Excel relegated the tools in the former Get External Data ribbon group to legacy status and merged the Get External Data group with the Get & Transform group to create the new Get & Transform Data group. This had the effect of making Power Query the default method of linking Excel to any external data source.

Power Query works in a very different way to legacy Excel. Rather than using the Excel grid and formulae, it uses commands, usually entered via the user interface, to create a series of steps which process the source data into an output table of data. Each step usually uses the output of the previous step as its starting point. Refreshing the query processes all of the steps in order. Power Query works with tables of data, rather than individual cells, and steps generally involve processes that affect entire columns or entire tables.

The Power Query interface converts user choices and operations into lines of 'M language' code that can be viewed through the Advanced Editor view. Code can be changed and created directly in the Advanced Editor or, line by line, using the Formula Bar. Not all code operations are available to be edited in the Formula Bar. Many Power Query processes, including complex ones, can be created directly from the user interface with no need to enter or change the M code, or even to view it.

The existence of queries within a spreadsheet is not immediately obvious. A Queries & Connections pane can be opened from the Queries & Connections group of the Data Ribbon tab. The Queries section of this pane shows the Queries in the active spreadsheet together with an indication as to error conditions. Queries can be edited and refreshed by right-clicking on an individual query in this pane. Queries are viewed and edited in a separate Power Query Editor window:




An important benefit when working with, and reviewing, Power Query operations is the ability to view the data in the state that it was in at the end of any step in the process. The APPLIED STEPS box lists all the steps in order of processing and clicking on a step displays the state of the table of data, after the application of that step, in the main window.

# 3 Fewer formulae, fewer errors?

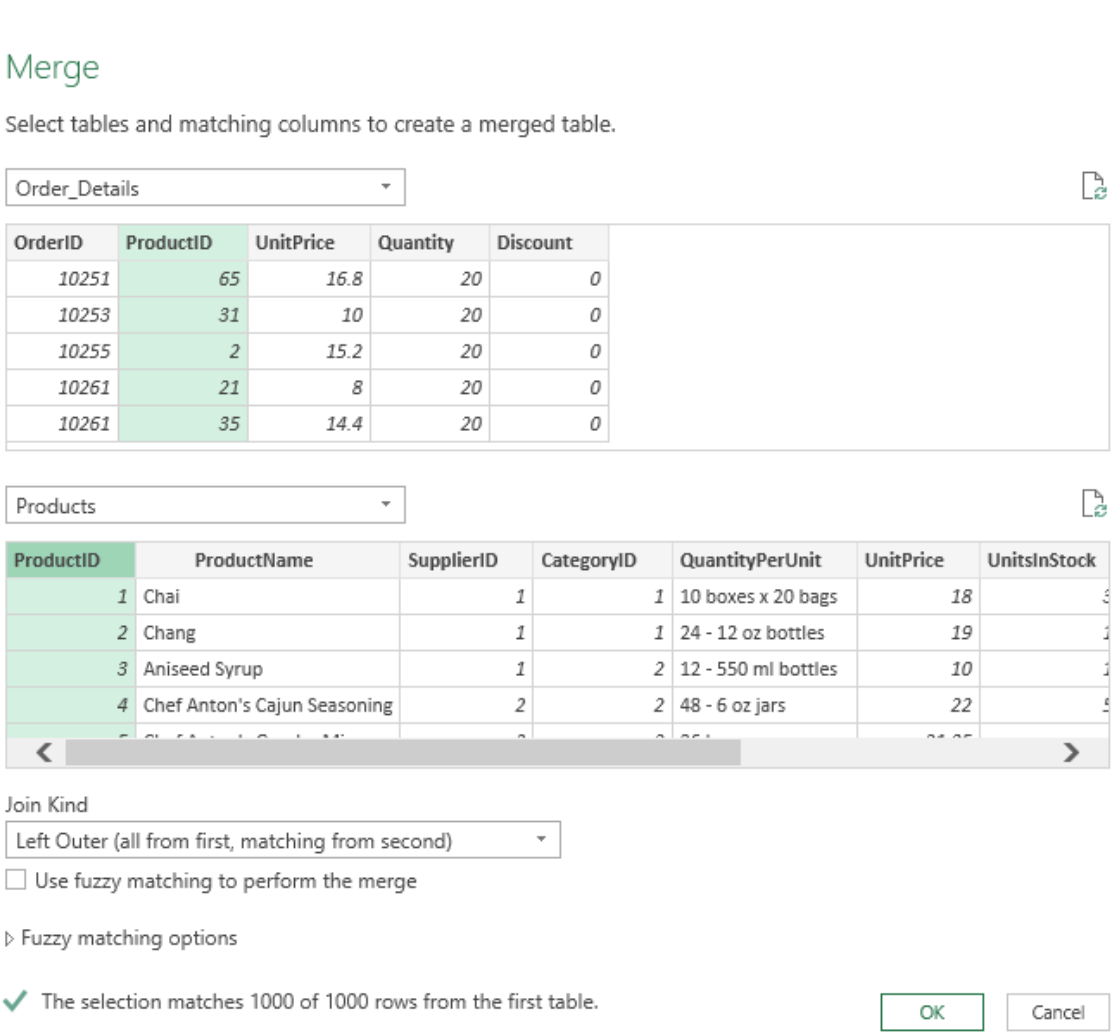
## 3.1 Cells versus steps

Ray Panko and others have published significant research into error rates in spreadsheets [Panko 2015]. Panko's analysis provides evidence for an error rate of 1% to 5% per cell and shows how error compounding quickly translates into a very high likelihood of error, as the number of cells that contain formulae increases.

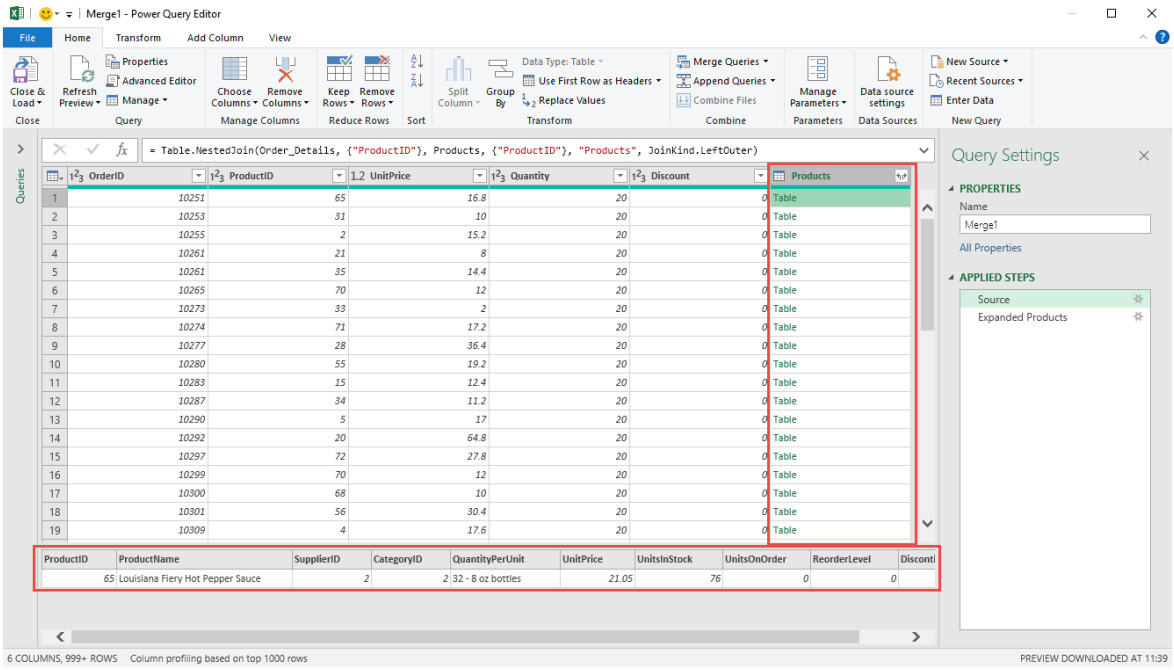
Because Power Query works with tables and columns, rather than individual cells, the number of formulae entered, or created by copying a root formula, can be dramatically reduced, and even eliminated altogether.

To take a simple practical example, if it were necessary to add a column of product names to a list of 1,000 sales order lines by using a lookup formula and a table of product IDs and names, this would require 1,000 cells containing the lookup formula. Clearly, this is unlikely to involve the separate entry of all 1,000 cells, but it does create 1,000 cell formulae, each of which is potentially susceptible to being edited or overwritten to create an error and, as Panko points out: "copying is not error-free, especially when the formula changes in “copied” cells in complex ways".

It is true that using Power Query we would first have to use the Get & Transform, From Table/Range operation on each of our two tables to make them accessible as queries but, having done so, we just need two interface operations to create a separate output table that includes a product name column. First, we use Get Data, Combine Queries, Merge to join our two tables using the common ProductID column:




This creates a new query containing our Order Details query with an additional column containing our Products table:



Our new Products column includes an 'Expand/Aggregate' icon in the column header that allows the choice of the column or columns to expand from the full list of columns in the merged table. In our case, we would just expand our ProductName column.

Behind the scenes, our two operations have resulted in the following lines of code:

Combine Queries, Merge:

= Table.NestedJoin(Order_Details, {"ProductID"}, Products, {"ProductID"}, "Products", JoinKind.LeftOuter)

Expand ProductName column:

= Table.ExpandTableColumn(Source, "Products", {"ProductName"}, {"ProductName"})

Of course, we can make mistakes using the interface, but rather than 1,000 formula cells to check, we have just two operational steps. Additionally, Power Query separates our source data, our calculations, and our output data, rather than combining all three within the same table as our formula approach does.

A very simplistic assessment of the two methods, based purely on error rate compounding, would suggest advantages in the use of Power Query; advantages that would increase significantly as the number of cells, and the number of operations, involved increased. There are, however, several other important considerations to take into account.

## 3.2 Visibility

It might appear that cell-based Excel has a clear advantage in terms of the visibility of the process. Because everything happens in cells on the face of the spreadsheet, it is easy to see, review and assess all of our calculations. In contrast, without knowledge of how Power Query works, we might see no evidence of its operation – just an output table with no visible formulae or calculations.

On the other hand, although this issue of visibility is often used as a supposed advantage for legacy Excel over alternative approaches, such as PivotTables or Power Query, Excel functions have their own event horizon, sucking in argument values and radiating a result with no visible workings. A Power Query can be explored with a simple right-click, Edit and can then often be more transparent than an Excel function, allowing the results of each step of a process to be separately reviewed and understood.

## 3.3 Different methods create different errors

One of the most common issues that can occur in a lookup operation is the failure to find an expected match. The cell-based approach using a lookup or match function will generate a #N/A! error in each cell where the formula cannot find a match. By default, Power Query will return a blank rather than an error, but this depends on the join type selected when our two tables are merged. Choosing an inner join will just omit the rows where no match is found altogether. Using a left or right anti join can just include the rows from one table where there is no match in the other table.

We should also consider the opposite problem – the existence of duplicate entries in the lookup table. Our lookup formula will just find the first match (reading from top or bottom, depending on which lookup function is used and the arguments entered). Our default Power Query join will return additional rows for each duplication.

## 3.4 State of development and risk

The situation is complicated by the current state of development of the tools in question. Unlike many of the Excel functions and formula constructions that users have grown used to working with over a long period, Power Query is just a few years old and continues to evolve quite rapidly. Changes are frequent and many of the possible techniques are yet to be subjected to extensive

exploration and testing in practice. On the other hand, Excel itself has also undergone some key changes recently with the introduction of important new functions such as XLOOKUP(), XMATCH(), LET() and LAMBDA() and the significant change to the calculation engine required to implement Dynamic Arrays, together with the associated new set of functions.

## 3.5 Skill sets

An additional risk factor for the use of Power Query is the knowledge and skills required to use it properly. Many Excel users have built up substantial knowledge of the use of formulae and functions within Excel through years of practical use, shared experiences, and internet searches. However, basic knowledge of how to work with relational databases is far less common. As seen with our simple lookup comparison, some knowledge of key database techniques, such as working with different join types, can be vital to avoid errors and to realise the full potential of tools such as Power Query. In addition, more advanced uses of Power Query can involve the need to think in more dimensions than for a simple grid of rows and columns. In Power Query, the equivalent of a single Excel cell can contain a list, or an entire table with multiple rows and columns. This can entail a radically different approach to addressing spreadsheet problems compared to the restrictions imposed by flat rows and columns.

Although much can be accomplished with Power Query without these additional skills, using unfamiliar tools without appropriate experience and training is likely to increase the risk of error.

# 4 Adapting review procedures for Power Query

## 4.1 Introduction

This section is intended to cover some of the most significant changes to review procedures that might be needed when Power Query is used, rather than covering the entire subject.

## 4.2 Refresh

For those used to Excel's default automatic recalculation, the need to use a manual or periodic refresh operation creates the possibility that output will not reflect recent changes.

As with PivotTables, changes to the source data do not trigger a Power Query refresh. It is possible to set queries to refresh at time intervals or whenever the file is opened, but this does not ensure that a Power Query will immediately show the result of changes to the source data. Power Query has additional issues to consider compared to PivotTables. A Power Query output table can be the source of another Power Query. Data, Refresh All, only refreshes the first level of queries meaning that PivotTables based on Power Query output tables, and queries based on other Power Query output tables, need a further refresh to force an update.

An additional consideration is the ability for a user to make changes directly in a Power Query output table. These will flow through to calculations that reference the changed cell or cells but, when the output table is refreshed, any such changes will be overwritten. To further complicate the situation, it is possible to add columns to a Power Query output table. Such columns will be retained when the table is refreshed but, if they are used to add data, changes in the source data, particularly a change to the sort order, would mean that the data will no longer be associated with the intended rows.

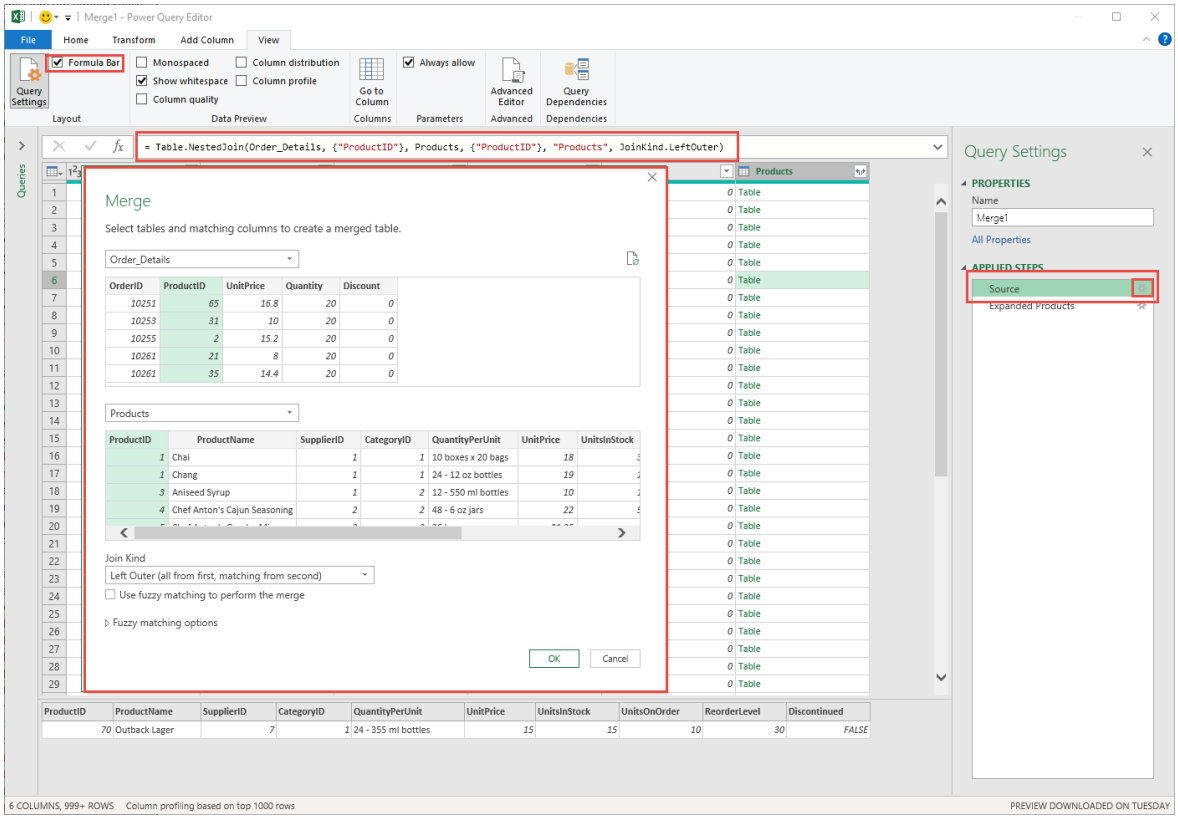
Accordingly, it is important to ensure that users are aware of the need to refresh and to try and build in cross checks that highlight differences between source data and output.

## 4.3 Review processes not individual cells

Rather than reviewing the contents of individual cells, and the calculations that they contain, where Power Query is involved, cell contents are divorced from the calculations that created them. Accordingly, rather than using formula auditing tools to trace precedent and dependent cells, it will be necessary to trace all Power Query outputs and review the steps involved in their creation. This could be just a case of following through the steps of a single query, or following through a series of precedent queries. The Power Query editor, View Ribbon tab includes a Query Dependencies command that displays a schematic layout of query dependencies.

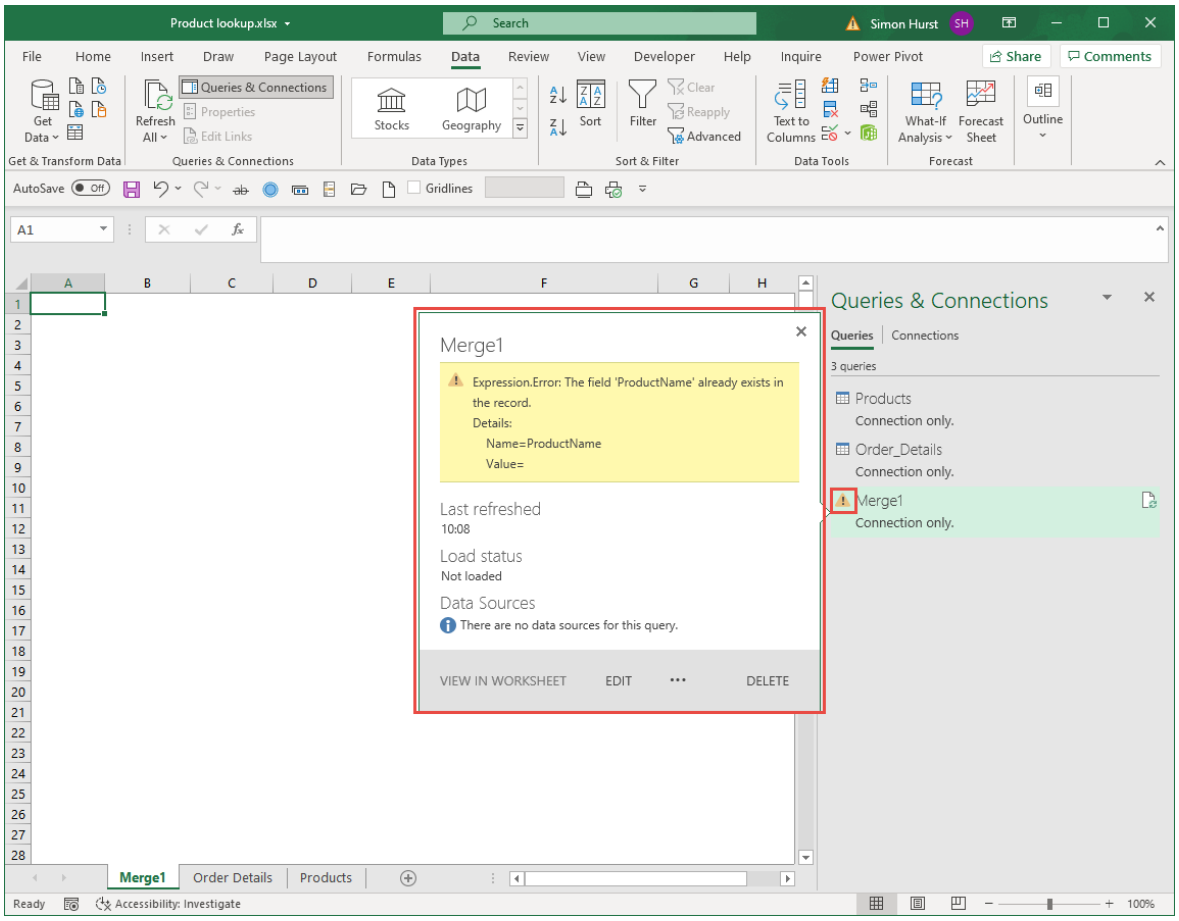
## 4.4 Power Query review tools

Power Query works with tables and columns. Steps change or create columns. One of the most useful aspects of Power Query is the ability to display the entire dataset as it was after the selected step. For many types of step, it is also possible to display the original dialog used to create the step. It is also possible to display a Formula Bar to view the underlying code of each step:

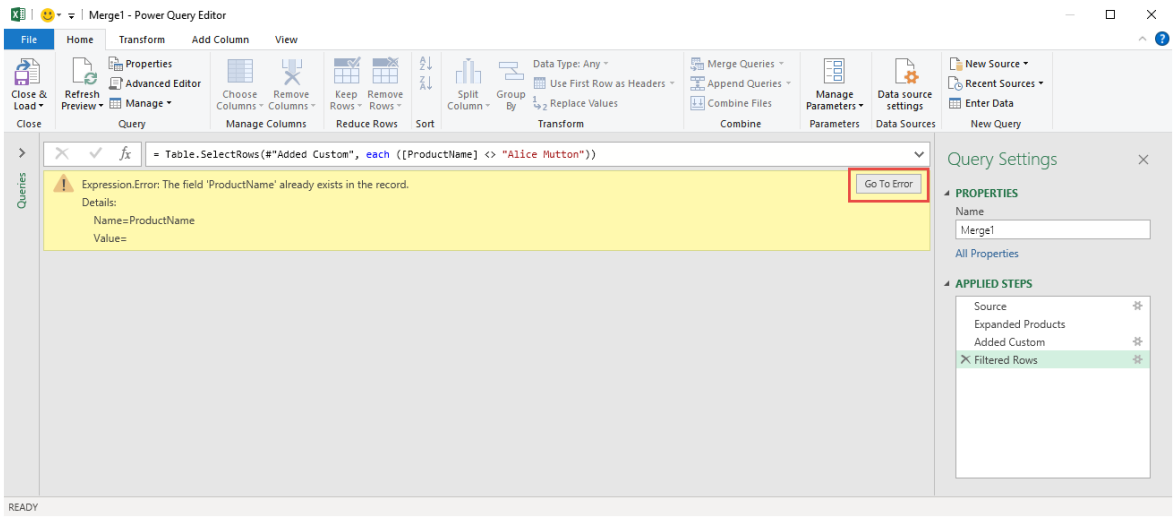



As can be seen in the above screenshot, a 'cell' in a query can contain an entire table, potentially containing multiple rows and columns. The contents of each table can be displayed beneath the main query window by clicking in the white space in the chosen 'cell'. This can be of considerable help when reviewing steps that create tables, such as a Merge operation.

Where a query cannot be completed because of an error, a warning will be displayed in the Queries & Connections pane. Hovering over the query description in the pane will display details of the error:



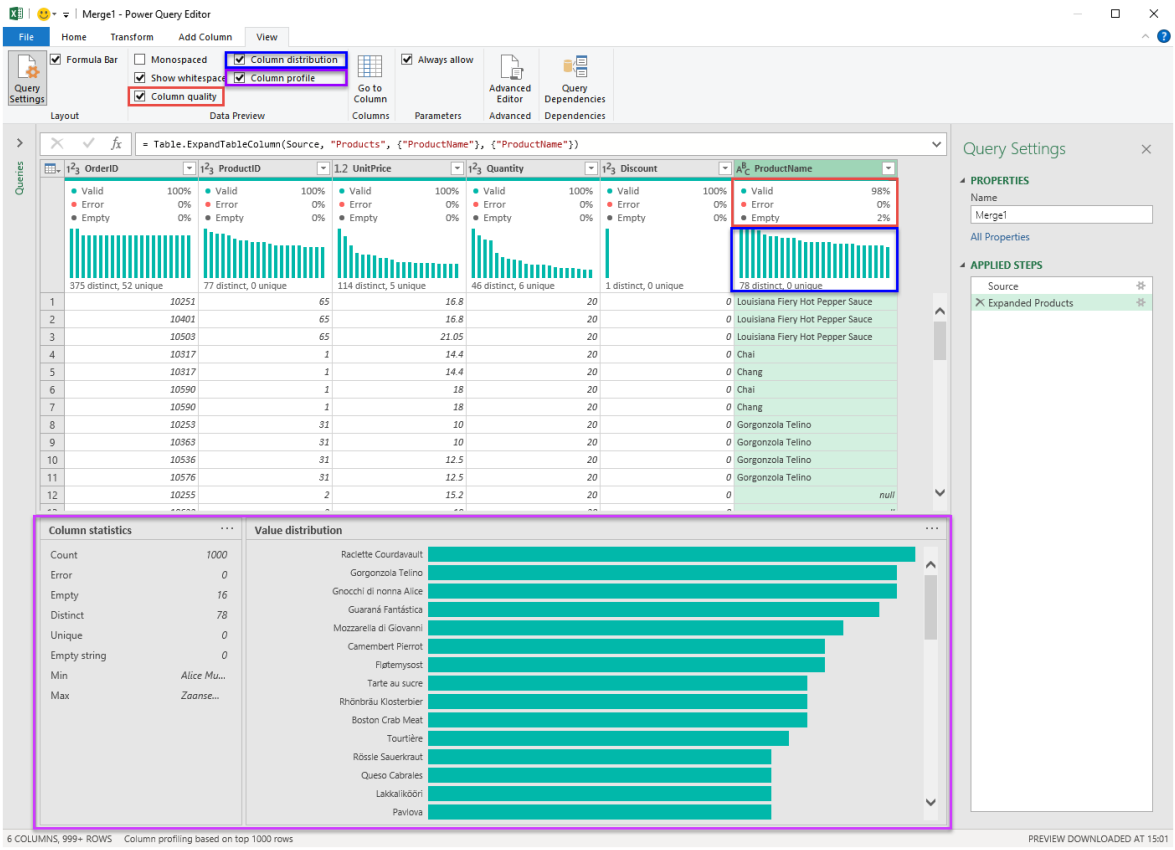

In the query editor, steps after a step that generates an error will show an error message that includes a 'Go to Error' button to go directly to the step that is causing the error:




## 4.5 Data profiling

Although the ability to see the results of any step is useful, the editor can only fit a limited number of rows in the window and scrolling through thousands of rows can be time consuming. The View Ribbon tab includes a Data preview group that provides a visible indication of the contents of an entire column. Here, we have turned on the following options:

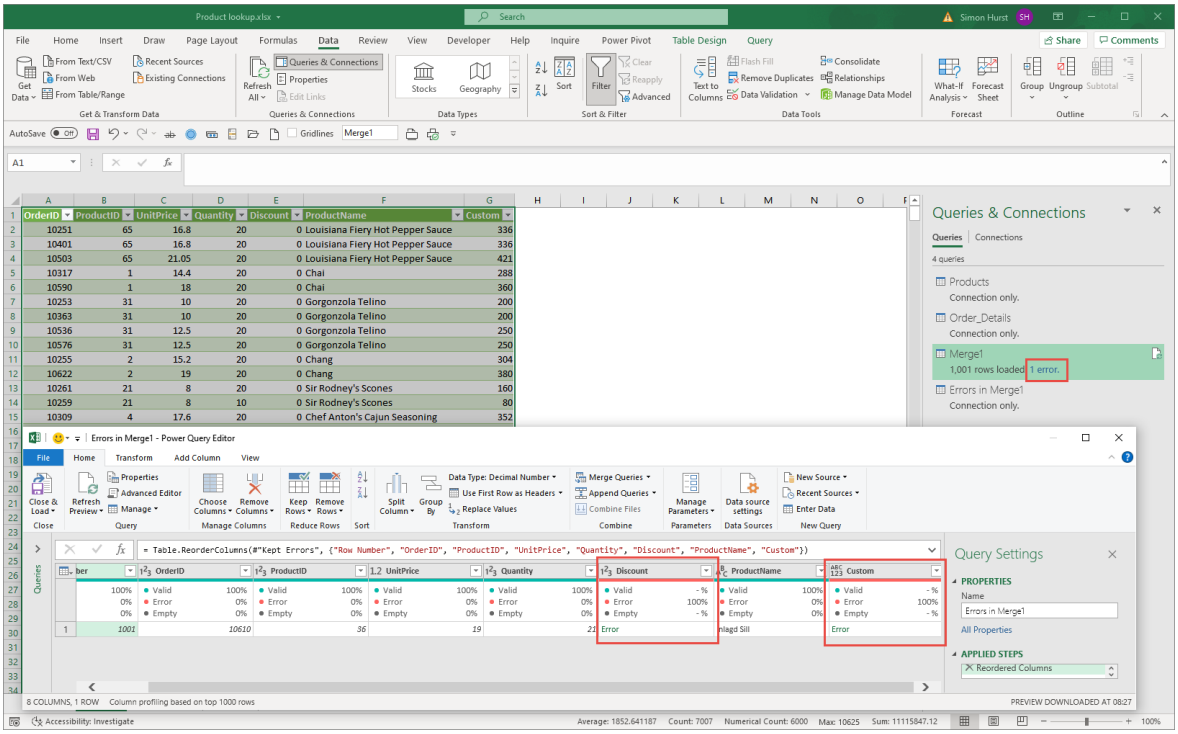
- Column quality
- Column distribution
- Column profile




Note that the default is to base this column information only on the top 1,000 rows. Clicking on the column profile status in the Status Bar allows this to be changed to the entire data set.

As well as the optional displays, the editor also displays a simple line in each column heading to indicate the presence of errors or empty entries in a column. This indicator is also limited to the first 1,000 rows unless it is specifically changed to refer to the entire data set.

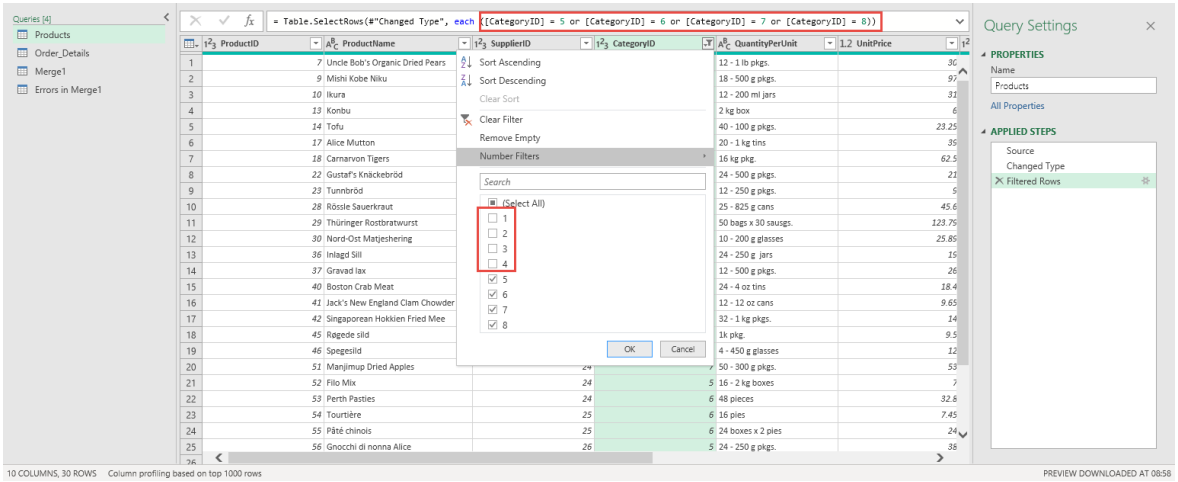
Where a query output table includes one or more errors, this will be indicated by an error link in the Queries & Connections pane. Clicking on the link will create and open a query that lists the rows including errors, together with an added index column:



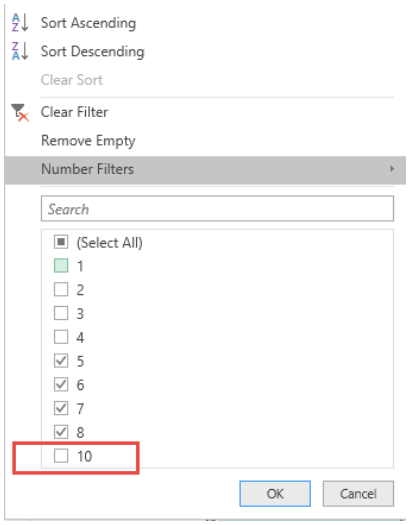
It is important to note that 'Connection only' queries do not display the number of errors in this way.

## 4.6 Interface to formulae conversion dangers

The way in which most users employ Power Query involves allowing Power Query to generate the underlying code from the interface options the user chooses. This does create the possibility that lines of code will be generated that don't work exactly as the user intended. For example, using a column heading to deselect some specific items will not necessarily create a filter that excludes just those items. Depending on the balance between the number of excluded/included items, Power Query can choose to select the other items rather than deselect the specifically chosen ones. This will mean that, if another item is added to the dataset, it will not be included in the output. In this example we want to exclude items 1,2,3 and 4 and have deselected those items in the filter list. Power Query translates this into filtering to display items 5,6,7 and 8:



As mentioned, this difference is significant if a subsequent update includes an additional item as it will not be automatically included:




There are many Power Query operations that can be subject to Power Query taking decisions which might be more efficient at the time but will not allow for future changes in the way that the user might expect. There is one feature that does this explicitly. 'Column From Examples' allows the user to enter examples of the end result required, leaving Power Query to decide which formula best achieves the required result. However, a formula that creates the correct result for the limited number of items originally visible might not cope with other existing items or with items added subsequently.

These issues emphasise how important it is for users and reviewers to make sure that the formula bar is visible and to check for all hard-coded values and the underlying logic.

## 4.7 Reference and duplicate queries

One of the most important techniques available to troubleshoot query problems and review Power Query processes is the ability to take any query and use it as the starting point for another query (Reference) or to duplicate an existing query and then delete, change, or add to the steps.

Referencing an existing query could be used to group and subtotal the query contents to enable the creation of totals for use in cross-checks.

Duplicating a query could allow check totals to be extracted at any stage for use in cross checks. For example, where a query filters results, Duplicate could be used to extract check totals before any filtering.

These approaches could be combined to create a full reconciliation, starting with totals before filtering and then using Reference to create a 'balancing' query that includes the unfiltered items enabling the reconciliation of the query to the full, unfiltered data set. Such a reconciliation would provide significant assurance that no data has been lost during the processing.

## 4.8 Text case

One very specific difference between Power Query and legacy Excel is the importance of text case. Power Query is case sensitive. Different capitalisation applied to a column heading for example will cause the column not to be recognised. In addition, when entering formulae using a dialog or directly in the formula bar, function case needs to be respected. This step retrieves the CodeChart table from the current workbook:

```
= Excel.CurrentWorkbook(){[Name="CodeChart"]}[Content]
```

Changing the capitalisation in any way would result in an error.

= Excel.Currentworkbook(){[Name="CodeChart"]}[Content]

This is not only significant in the creation of formulae. By default, if you are matching two columns in a Merge, text case will be respected so Sales will not match sales or SALES. Although a recent 'fuzzy matching' feature allows you to Ignore Case when performing a match, in many situations it might be preferable to convert both sides of the match to completely UPPER or lower case before the merge step.

# 5 Reducing Power Query risk

## 5.1 Introduction

For obvious reasons, many of the methods for reducing risk when using Power Query are related to the risk factors discussed in the previous section. Once again, the intention is just to provide an initial assessment of some of the more significant areas rather than producing a comprehensive guide to the subject.

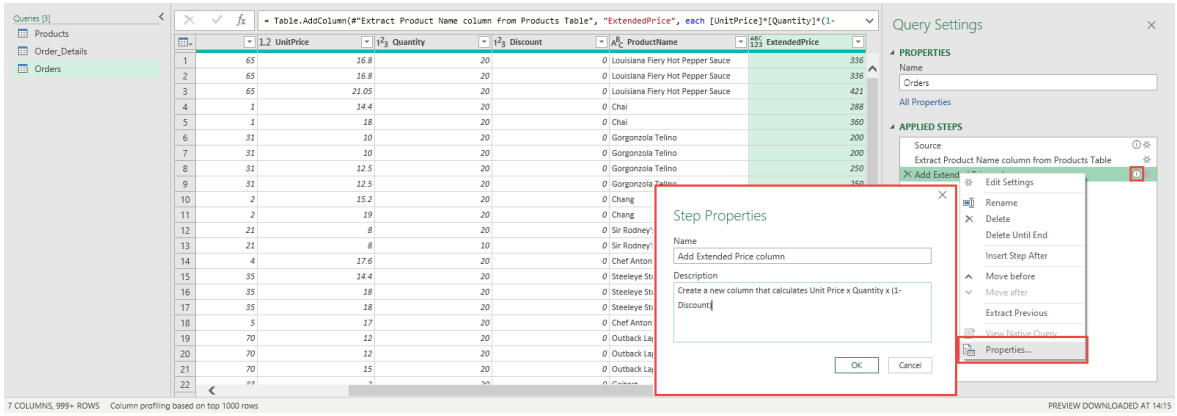
## 5.2 Modular queries

Rather than creating a single query that acquires and processes a table from raw data to result, breaking the process into a series of logical sections makes it easier to isolate and track down issues and also allows the creation of cross checks at different stages of the process.

It is also clearly more efficient, and easier to review, a single set of steps than to have to review the same set of steps repeated in multiple queries.

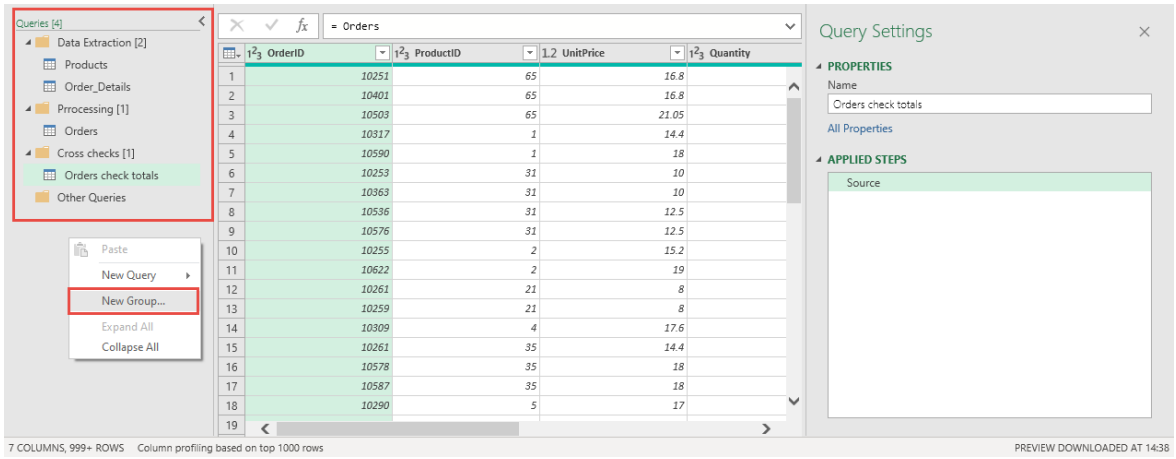
## 5.3 Documentation

As well as the general spreadsheet documentation that is equally as applicable to a spreadsheet that uses Power Query as it is to legacy Excel, Power Query has specific documentation capabilities. Although each step is given a default name based on the interface feature used to create it, it makes a series of steps much easier to understand if the step is renamed to something more informative as soon as it is set up. In addition to the name of the step, it is possible to enter a fuller description by right-clicking on a step and selecting the Properties option:




When a Description has been added to a step, an 'information' icon appears towards the right-hand side of the step in the APPLIED STEPS pane and hovering over the step will display the description.

Queries themselves can be given descriptive names and can also be allocated to groups that are created by right-clicking in the Queries pane:



These groups are also shown in the Queries & Connections pane in the main Excel window and, in both settings, groups can be extended and condensed to show or hide their constituent queries.

## 5.4 Avoiding hard-coded references

One of the most common causes of an error in a query that stops it from completing is a reference to a column name that doesn't exist. This occurs frequently in Power Query because most steps reference at least one table column and many query operations create references to all columns. Should the heading of a column in the source data change, existing references to the original column name will result in an error. Sometimes this is unavoidable without full control over the source data, but it also occurs when columns that are eventually discarded are referenced.

Some of the most likely causes of this issue are the automatic Changed Type step that Power Query defaults to applying when it comes across new columns and Remove Columns and Move Columns steps.

When acquiring a set of 'unstructured' data, Power Query automatically analyses the contents of each column and, in a Changed Type step, sets the Data Type for each column. This means that each column in the data set is referenced by name. There is a good argument for always deleting this automatic step or, perhaps even better, changing the default option in File, Options & Settings to turn on "Never detect column types and headers for unstructured sources". You can then specifically set the data types required for each column when you have removed any columns from the original data source that are not needed. Setting data types is important, as matches and mathematical operations can depend on data types and cause errors when these are not set correctly.

This leads on to the different methods for removing columns. The most obvious way to remove columns that you don't need is to select them and the use Remove Columns. However, this has the significant disadvantage of creating a hard-coded reference to all the columns that you aren't going to use:

= Table.RemoveColumns(#"Promoted Headers",{" Postcode Data Status ", " Lines < 2Mbps(Y/N) ", " Average Speed/Mbps ", " Median Speed/Mbps ", " NGA Available(Y/N) ", "Number of Connections"})

The alternative is to select the columns that you do want to keep and to use Remove Other Columns. This uses Table.SelectColumns to keep the remaining columns:

= Table.SelectColumns(#"Promoted Headers",{"Postcode(No Spaces) ", " Maximum Speed/Kbps"})

Because these columns will be used anyway, should the source heading change it is something that has to be dealt with anyway, in contrast to a change in the heading of a column that is destined to be discarded.

A less obvious cause of the same problem is moving columns. You might only move a single column, but the Table.ReorderColumns step that is created references all columns:

= Table.ReorderColumns(#"Promoted Headers",{"Postcode(No Spaces) ", " Postcode Data Status ", " Lines < 2Mbps(Y/N) ", " Average Speed/Mbps ", " Median Speed/Mbps ", " NGA Available(Y/N) ", " Maximum Speed/Kbps", "Number of Connections"})

A good general approach is to keep wanted columns and filter out unwanted rows as early as possible to both reduce the scope for naming errors and to improve efficiency.

## 5.5 Check all filter operations

Power Query makes its own choices about how to apply filters. Whether you select items you want, or deselect items that you don’t want, using the interface, Power Query chooses how to set the filter based on the balance of selected/unselected options. If you choose to deselect 3 items from a list of 8, Power Query will choose to use <> for those 3 items:

= Table.SelectRows(#"Changed Type", each ([CategoryID] <> 1 and [CategoryID] <> 2 and [CategoryID] <> 3))

However, if you deselect 5 items from the same list of 8 Power Query will instead use = for the remaining 3 items:

= Table.SelectRows(#"Changed Type", each ([CategoryID] = 6 or [CategoryID] = 7 or [CategoryID] = 8))

The difference is crucial when it comes to what happens when further items are added to the data set. If Power Query has chosen to use <> with the original omitted items your query will include new items, but if it has chosen to use = with the original remaining items then your new items will be excluded.

Accordingly, it is vital to check the formula bar to see just how Power Query has interpreted the interface instructions, or to use the detailed filter dialog to set the filter as you want it in the first place.

## 5.6 Text case

Because Power Query is case sensitive, where text might be used in a match or a conditional formula or filter for example, the query should include steps to allow for the data being entered with different text case. A Transform operation is available to convert a text column to UPPER, lower or Sentence Case.

## 5.7 Pre-checks

A set of queries can be created to pre-check data before the actual processing stages. For example: listing rows with unexpected data types; missing values in lookup tables and unexpected duplicate values.

## 5.8 Test data

It is very easy to change the data source in individual queries or globally. This makes it possible to create specific test data sets that you can easily switch to in order to test the entirety of any process.

# 6 Power Query as review tool for legacy Excel

## 6.1 Introduction

In addition to considerations of risk in Power Query based solutions, it is worth considering whether the Power Query tools might also have a role to play in reviewing and troubleshooting legacy Excel models. For example, Power Query could be used to carry out parallel processes to create check totals, generate exception reports and check the structure of a report.

## 6.2 Parallel process

If a legacy Excel model performs calculations on tables of data, or even poorly structured blocks of values, Power Query could be used to extract the values involved and process them to generate equivalent values for cross-checking. A particular example would be lookup functions. Because Power Query uses database joins between tables, the generation of the same result as the use of lookup formulae would give a reasonable level of assurance that the lookups are working as intended. Similarly, Power Query can be used to group and summarise data to produce the same totals as a PivotTable.

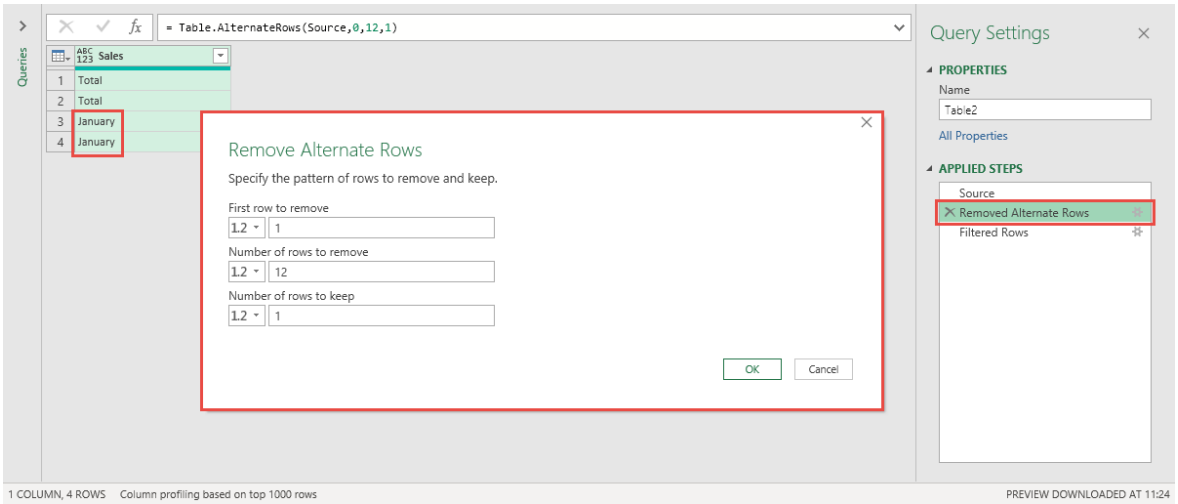
## 6.3 Exception reports

Once again, the ability of Power Query to use database joins, and additionally to implement different join types between tables, makes it possible to create tables of unmatched items between tables. So, where an Excel model relies on a lookup between tables, Power Query could generate an exception report listing any unexpected missing matches, or list duplicate entries in a lookup table that should only have unique values in the lookup column.

Power Query could also be used to report on any data type discrepancies. For example, non-numeric items in a column that should only include numbers, invalid dates in a date column or missing values.

## 6.4 Structure

Where a range or report is anticipated to follow a particular structure – perhaps having a total row every 13th row, Power Query could be used to confirm this structure and report on exceptions. Here, Remove Alternate Rows is used to just keep each 13th row. The result could then be filtered to show any rows that don't contain 'Total' which would indicate an issue with the consistency of the structure:



# 7 Conclusion

Power Query certainly doesn't eliminate all spreadsheet risk. In many cases it replaces one type of risk with another. However, Power Query does have the capability to replace individual cell formulae with steps that affect all the cells in a column or even an entire table. Properly used, this could substantially reduce the number of individual points of failure that need to be assessed and tested.

As with legacy Excel, overall design and the application of particular techniques can reduce the inherent risks involved in the use of Power Query.

Power Query could have a role to play in the review of legacy Excel spreadsheets given its ability to carry out calculations in different ways and to highlight exceptions.

Although spreadsheet risk is generally considered in terms of errors that result in financial or reputational damage, it is possible that another more insidious type of spreadsheet risk results in even greater damage – the inefficient use of spreadsheets. Because the use of spreadsheets is so ubiquitous, by using spreadsheets for inappropriate applications or choosing sub-optimal methods and techniques, vast amounts of time and effort are being wasted. This waste of resources is not limited to the time spent in creating the spreadsheet solution, but also the time spent in reviewing, troubleshooting, and correcting bad spreadsheets.

Failing to understand the potential of using Power Query for overall automation, rather than just for data acquisition, risks the continued use of inefficient manual methods where a practical alternative exists.