Monkey Tools

Build better, faster

Installation & Feature Guide

Last Updated: 2020-05-05

Table of Contents

1 Introduction & Philosophy

The philosophy of Monkey Tools is simple: Build better, faster.

Our aim is to provide you with a good set of tools that help you, as a business intelligence author, to:

- Build models more rapidly
- Follow recommended practices
- Document your work
- Audit files that you receive

1.1 Our "Do No Harm" Philosophy

We have a "Do No Harm" philosophy with regards to your workbook. What does that mean?

It means that we never add something to your workbook unless you ask us to. All our analysis is done in-memory. We never add hidden flags or data to your workbook, and only modify or inject items into your workbook when specifically asked to do so.

1.2 No Residual Hooks

One of the features that Monkey Tools provides is the ability to inject queries into your workbook to make your life easier. But what happens when you send that file to someone else? It will still work, even without Monkey Tools installed on the other user's PC. Our software is developed to make the creator's life easier, not lock the entire organization into user licenses.

1.3 Community Version

As a Microsoft MVP, Ken values community highly. For this reason, we will always provide a free version of Monkey Tools. While it won't allow you to do everything the Pro version will, it should still add some useful tools to your toolbox.

1.4 As Current as You Like

We're excited about this product, and we have big plans to add many more features. Sometimes those take a lot of planning. Sometimes they come quickly as we code a fix or feature for things that irritate us. And sometimes, we quietly fix a bug. As we do this, we want to push those features or fixes as quickly as possible, which is why we developed the release schedule we use.

By default, Monkey Tools will check for an update every two weeks from the day you install it. But you have granular control over this, allowing you to check for updates more frequently, or more slowly, or even on-demand. We hope that you'll enjoy this feature, as we know that some people are eager for updates, while others would prefer to avoid update notifications.

(Speaking of which, if you ever find a bug, please email us a support@excelguru.ca so that we can fix it!)

We hope you enjoy Monkey Tools and that it becomes a valuable part of your analysis toolkit!

2 Installation

This chapter will help you install the Monkey Tools software tool.

2.1 Supported Software Versions

Before you get started, it's a good idea to check that your version of Excel is supported, as we cannot provide support for versions of Excel that are not on our list.

Excelguru's Monkey Tools add-in is supported for both 32- and 64-bit versions of:

- Excel 2016
- Excel 2019
- Microsoft Office 365 (any version installed via an Office 365 subscription)

To check your version information, open Excel and go to File → Account. Your version information is shown on the top right side of the screen:

Figure 1 - Determining your Office product

The Monkey Tools Excel add-in can also connect to any downloaded version of Power BI Desktop, although you will need to set your default launcher in the Options section (see page 88).

2.2 Installing the Monkey Tools Excel Add-in

When you purchased Monkey Tools, you should have received an email with two important pieces of information:

1. A license key
2. A link to the Monkey Tools installer

To install the software:

1. Close Excel
2. Click the link above and choose "Run" if prompted

You will then be asked if you would like to install the Monkey Tools application[1]:

[1] If you receive a message that you cannot install programs from the internet, please see Installation Errors on page 24 for steps to solve the issue.

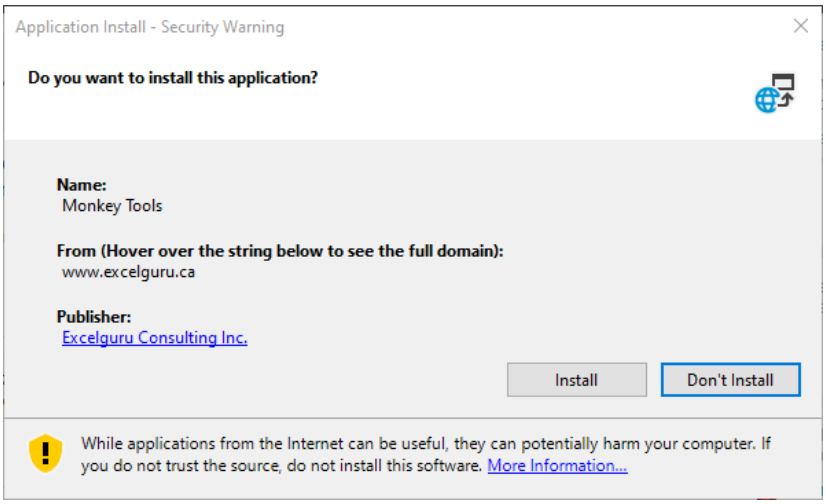


Figure 2 - Installing Monkey Tools

3. Click Install

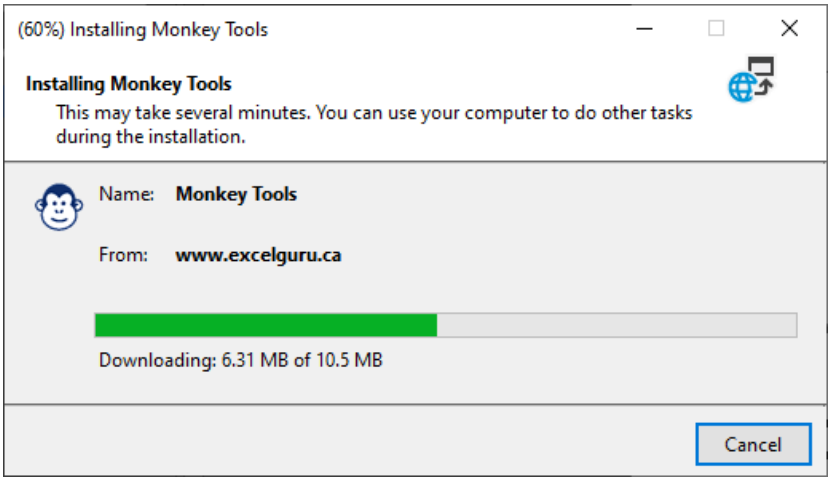


Figure 3 - The Monkey Tools installer in action

Monkey Tools will then begin the process of installing. Once advised that the product registration has succeeded, click OK:

Figure 4 - Monkey Tools has successfully installed!

You are now ready to launch Excel, where you should see the new Monkey Tools ribbon in place:

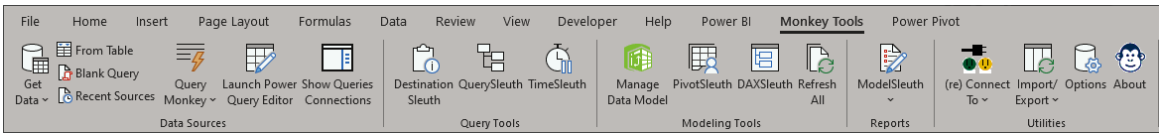


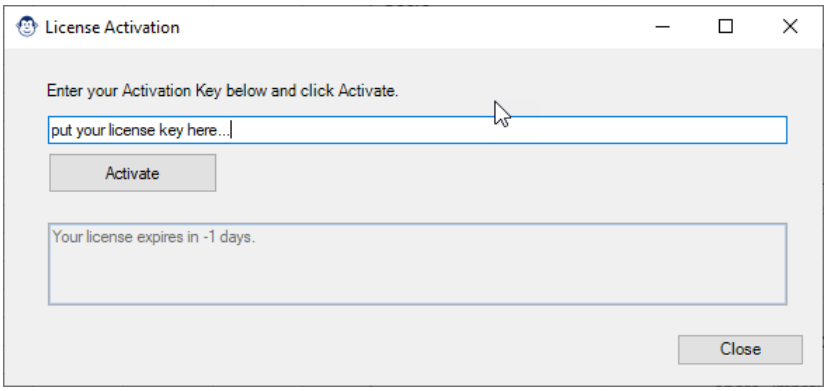
Figure 5 - The Monkey Tools ribbon

2.3 Activating your Pro (or Trial) License

After installing Monkey Tools, don't forget to activate your license. To do this:

- Open Excel
- Go to the Monkey Tools ribbon → Options → License Activation

You will be taken to this form:



Copy the license key from the email you received, paste it in the form and click Activate in order to activate the pro (or trial) features. You can then close the form and begin using Monkey Tools!

3 Ribbon and Feature Overview

Once installed, you will see a new Monkey Tools tab on the Excel ribbon, which looks like this:

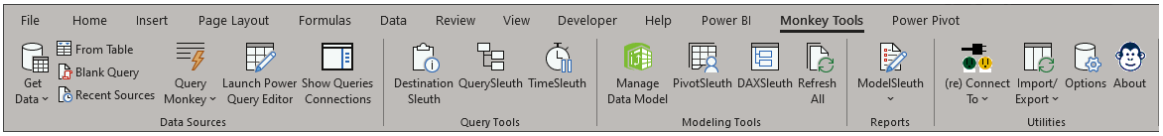


Figure 6 - The Monkey Tools ribbon

3.1 Data Sources Group

This Data Sources group holds the commands in order to generate new Power Queries for sourcing data:

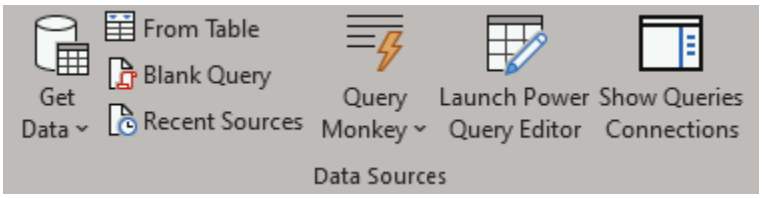


Figure 7 - Monkey Tools Data Sources ribbon group

The purpose of each button is explained below:

3.1.1 Get Data

The Get Data button is a duplicate of the Get Data button found on Excel's Data tab[2]. It is placed here in order to avoid tab flipping when you are creating new queries.

3.1.2 From Table, Blank Query, & Recent Sources

Like the Get Data button, these are duplications of the same buttons from the Data tab's Get Data menu. As these are commonly used buttons for BI pros, they were added here to save clicks.

3.1.3 QueryMonkey

This button contains a menu that helps create useful queries for your work to save you time. Each is detailed in Chapter 4 - QueryMonkey.

3.1.4 Launch Power Query Editor

The Launch Power Query Editor button is the standard button to launch the Power Query editor but placed on the top level of this tab for ease of access (rather than buried in a sub-menu).

3.1.5 Show Queries & Connections

This button will open (or close) the Queries & Connections pane on the right side of the Excel window.

3.2 Query Tools Group

Monkey Tool's Query Tools group contains features intended to help the Power Query developer manage, understand, and benchmark their queries:

[2] This button is version appropriate, i.e. in Excel 2016 it is named "New Query".

Figure 8 - The Query Tools group

3.2.1 DestinationSleuth

The DestinationSleuth button displays a form that shows load destinations for all queries in much more detail than the simplified "x rows Loaded" vs "Connection Only" you see in the Queries & Connections pane. Full details of this form can be found in Chapter 5 - DestinationSleuth.

3.2.2 QuerySleuth

- Monkey Tools QuerySleuth is a robust form that provides you with a large amount of information regarding your queries. It displays the query precedent/dependency tree, indents and colours for M code, and allows query editing right from Excel. Learn more in Chapter - to change the destination of the four queries required in the data model.

Why? Simple. Its all about speed. Consider a scenario where you have a complicated query chain that creates 10 tables and each has 10,000 rows of data, but where only 5 need to be loaded to the data model. Which will be faster?

- Wait for all 10 to load to the data model, change 5 of them to connection only and wait for them to unload, or
- Load all 10 to connection only, then use DestinationSleuth to change the 5 you need to load to the data model?

Its obviously the latter, as you don't have to wait for the data to unload when removing unneeded tables from the data model. In fact, we believe in this so strongly, that we set our load destinations for all queries to load to Connection Only by default. That way, if we ever make a mistake, it happens very quickly, and we can repoint the tables to the correct destination with DestinationSleuth.

If you'd like to do the same, go to:

- Data tab → Get Data → Query Options → Data Load
- Choose "Specifiy custom default load settings"
- Clear the Load to Worksheet and Load to Data Model checkboxes
- Click OK

3.2.3 Understanding What Happens to Connections Upon Change

It is important to realize that there are two different types of connections inside an Excel workbook; those that populate the data model, and those that don't. The reason this is important to understand is that it can have an impact on what happens to data tables when a load destination is changed.

A point in case is when you change a query from "Load to Table" to become "Load to Table and Data Model", the original Table connection must be deleted and recreated as the internal connection string is

different. Naturally, this will cause issues for any formulas or macros that are pointed at the historical table.

The following matrix describes what will happen with worksheet tables during a change.

Change From	Change To	Original table deleted?
Table	Connection Only	Yes
Table	Data Model (only)	Yes
Table	Table & Data Model	Yes, then recreated
Table & Data Model	Connection Only	Yes
Table & Data Model	Data Model (only)	Yes
Table & Data Model	Table (only)	Yes, then recreated
Data Model	Connection Only	Yes
Data Model	Table (only)	Yes
Data Model	Table & Data Model	No, just added to connection

Figure 35 - Understanding Impacts of Destination Changes on Tables

This behaviour is completely consistent with changing load destinations manually but is something that you should be aware of.

QuerySleuth.

3.2.4 TimeSleuth

This feature allows you to time the length of time it takes for a full or partial refresh of your Power Queries. Developed to allow performance testing and benchmarking of query load times, you can see which query is slowing down your work the most. You can also compare queries when loading with privacy enabled or disabled. A full description of how to use this feature is contained in Chapter 7 - TimeSleuth.

3.3 Modeling Tools

The features in the Modeling Tools group are built to make your life easier when building and managing data models:

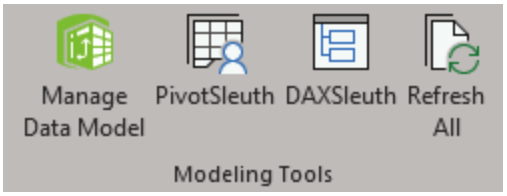


Figure 9- The Modeling Tools group

3.3.1 Manage Data Model

This is the standard Manage Data Model button that comes with Excel, placed here for ease of access. Clicking it will take you in to the Power Pivot editor as you are used to.

3.3.2 PivotSleuth

PivotSleuth will launch a form that allows you to discover all the fields used to drive a PivotTable, including filters and timelines. Not only does it display them with their actual field names from the Data model, it also highlights fields that are contributing to the “Relationships May Be Needed” error, or fields that may do so in future

3.3.3 DAXSleuth

Similar to QuerySleuth, the DAXSleuth displays a full dependency tree of measures and calculated columns used in the Data model. DAX indenting and colourfication are present, as well as the ability to quickly duplicate or modify measures. In addition, for Excel based models, each PivotTable or PivotChart that contains the selected measure is listed in a separate dependency tree. All the features of DAXSleuth are detailed in Chapter 9 - DAXSleuth.

3.3.4 Refresh All

As data refreshes are just part of the development cycle, we’ve added the standard “Refresh All” button to our ribbon to save you switching away to the Data tab. A simple click will reload your Power Query and Power Pivot chains as normal.

3.4 ModelSleuth

Monkey Tools comes with several built-in Analysis Reports in order to help you understand your model more quickly. In addition, we also include the ability to pull directly from SQL’s Dynamic Management Views (DMV’s) if you want to do your own model analysis:

Figure 10 - The Reports group

3.5 Utilities

The Monkey Tools Utilities ribbon contains several important features for working with the tool:

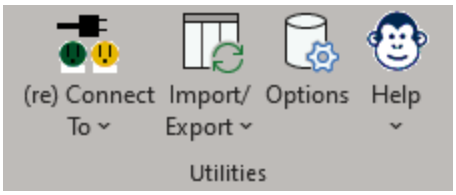


Figure 11 - The Utilities group

3.5.1 (re) Connect To…

This menu contains the ability to connect to Excel workbooks, Power BI models, and Monkey Tools Backup files in order to analyze the model components. These connections are essential to being able to drive all the other features of the Monkey Tools product.

Full details on the connections available, as well as how to use them, are contained in Chapter 11 - (re) Connect To…

3.5.2 Import/Export

This menu holds the launchers allowing you to import models into Excel or export them in a variety of formats for later retrieval/analysis. Full details can be found in Chapter 12 - Import/Export Queries.

3.5.3 Options

The main purpose of the Options form is to allow you to configure global options and activate your trial or pro license. It also allows resetting forms to their default states, should you want to revert to the installation defaults. All features of this form are discussed in Chapter 13 - Options.

3.5.4 Help

This menu allows you to quickly see your version information, as well as download the most up to date version of this document.

4 QueryMonkey

The QueryMonkey is our helpful assistant who will automatically inject defined queries into your workbook, saving you the hassle of creating them from scratch yourself. QueryMonkey currently provides access to the following queries:

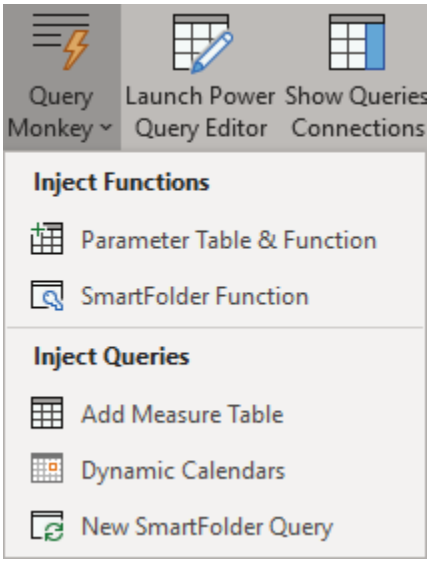


Figure 12 - Queries available for injection

4.1 Parameter Table & Function

The purpose of this button is to quickly inject the key components into your workbook to allow you to feed your query parameters that are maintained in the Excel worksheet. Upon selecting this button, two things are added to the workbook:

1. A new “Query Options” worksheet
2. The “fnGetParameter” function

4.1.1 The Query Options worksheet

This worksheet is injected with the table required to feed the fnGetParameter function. It comes pre-loaded with a default variable of “File Path” and the required Excel formula to generate it.

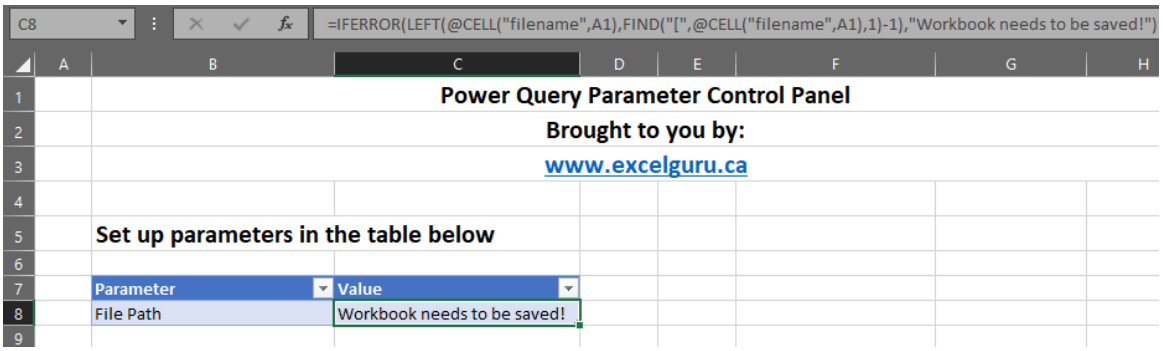


Figure 13 - The Parameters table required for the fnGetParameter function

To add new parameters to your table, simply:

- Type in the new parameter name in column B
- Type the value for the parameter in column C

Please note that all new parameters must be added in the row *immediately below* the last row of the table, as the table expands automatically to include them. You may use hard coded values, text, or formulas in the Value column of the table, making this solution fairly dynamic.

4.1.2 The fnGetParameter Query

The second part that gets injected into your file via the Parameter Table & Function feature is the fnGetParameter function:

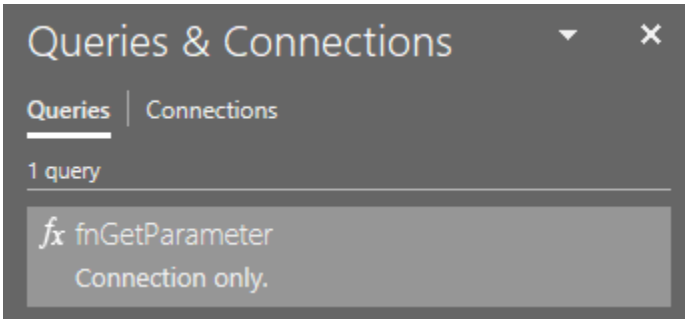


Figure 14 - The fnGetParameter function is now contained in the workbook

This function allows you to pull parameters from the Excel table, and use them in your other queries by typing the following in the Power Query editor's formula bar:

```
=fnGetParameter("<the parameter name>")
```

Just replace <the parameter name> with the exact name that you used in the first column of the parameter table.

In order to reduce your chances of falling afoul of Power Query's Formula Firewall, we highly recommend that you never nest this function call directly into another query step. Instead, edit your existing query in the Advanced Editor and – immediately after the let line – create a variable to contain the result of the fnGetParamter function. Then, refer to that variable with the item you wish to make dynamic:

```
let
      filepath = fnGetParameter("File Path"),
      Source = Folder.Files(filepath)
in
      Source
```

Don't forget to place a comma at the end of the line you use to create your variable!

4.1.3 Working with the File Path Parameter

There are two potential issues that you will experience when working with the File Path parameter specifically:

1. The cell returns “Workbook needs to be saved!”. This message indicates that you have injected the queries into a new file, and it doesn’t yet have a save location. Save the workbook, select the cell, press F2 and then Enter, and it will update to the new file path.
2. The cell returns a URL instead of a local file path. This issue occurs for files saved in a OneDrive sync folder when the user has the Office 365 version of Excel installed. Unfortunately, Microsoft has not provided us a way to determine the local file path, so the only way to ensure you get the local file path is to edit your OneDrive sync settings and uncheck the box shown below:

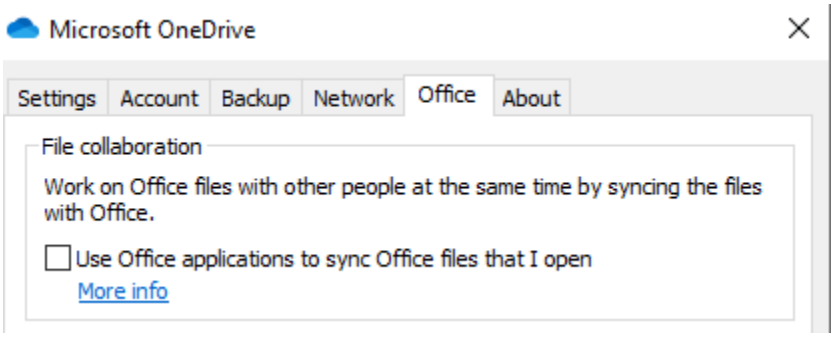


Figure 15 - Turning off Office sync in OneDrive

For more about this query, read the Excelguru blog post on the subject. The query is virtually identical to what has been shared there, with the exception that it reads from a table called “XLG_Parameters” so as not to conflict with any existing tables in the workbook using the name “Parameters”.

4.2 SmartFolder Function

This feature inserts the SmartFolder function into the workbook, allowing you to solve a couple of issues. The ultimate goal is to provide a list of files in the target folder, with a little futureproofing done in the form of forcing all file extensions to lower case. (This ensures that filtering for an .xlsx file will never miss .XLSX files.) In addition, it also removes the target folder from the Folder name column, making it very easy to see what sub folders (if any) still exist in the list. On its own these things seem relatively minor, but they are just a convenience for the function’s larger goals.

The true purpose of the SmartFolder function:

The real benefit of this function is that it can accept a local drive path to a folder on your system, or the full path to a subfolder on an Office 365 SharePoint site. This makes connecting to a SharePoint folder *much* easier, as it saves you having to work out the root of the site, connect to it, and then filter the Folder name column to drill down to the files you actually want. The function does all of that for you *and* can switch easily should your file path “smart switch” from local to SharePoint based on Microsoft’s OneDrive policy implementation. Even better, it does this all without violating the Formula Firewall!

4.2.1 Implementing the SmartFolder Function

Adding this function to your workbook is very easy: Just click the button and it will be injected automatically:

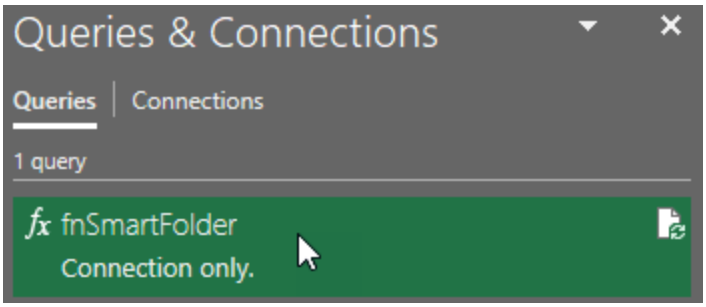


Figure 16 - The fnSmartFolder function has been added to the workbook

4.2.2 Using the SmartFolder Function

To invoke this function. you have (at least) two different options:

4.2.2.1 Option 1:

Double-click the function in the Queries & Connections pane, and type in the file path you wish to read from. You'll be prompted to enter the folder path as shown in Figure 17:

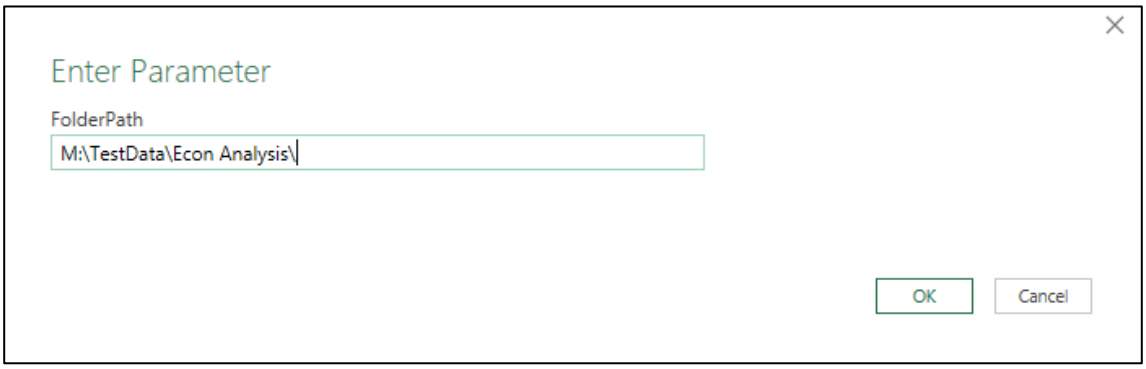


Figure 17 - Invoking the fnSmartFolder function

Upon clicking OK, you'll be taken into the Power Query editor and find yourself in a new query, ready to do what you want with the files:

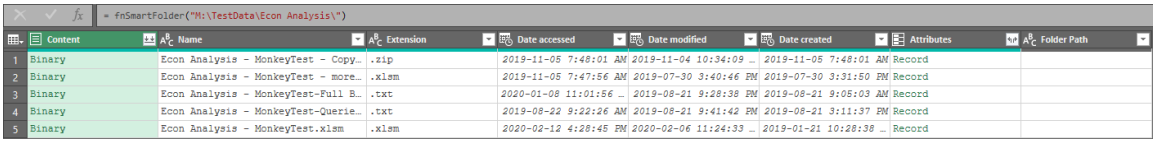


Figure 18 - The new "Invoked Function fnSmartFolder" query

Remember that the path you provide can be a local file path, an Office 365 based SharePoint path, or even the path to a OneDrive for business folder!

4.2.2.2 Option 2:

Insert a Parameter Table/Query as shown earlier in this section. Then create a query that passes the file path into the function by using the following steps:

- If you don't already have it, insert a Parameter Table/Query
- Insert the SmartFolder query

- Click the Blank Query button on the Monkey Tools ribbon
- Type in the following formula in the formula bar and press Enter:

```
=fnSmartFolder( fnGetParameter("File Path") )
```

At this point you've got the list of all the files from the folder in which you've stored the workbook and can filer to subfolders from that location. And the best part? Whether the CELL() function enumerates the local path or SharePoint-hosted path, it will still work!

4.3 Add Measure Table

Many modellers like to store their measures on a single "Measures" table in the data model. If you're one of them, you may appreciate the ability to quickly add a disconnected table into your data model for this purpose. Thanks to Monkey Tools, this is quite easy!

Upon selecting this option, you will get prompted to enter or confirm the measure table name:

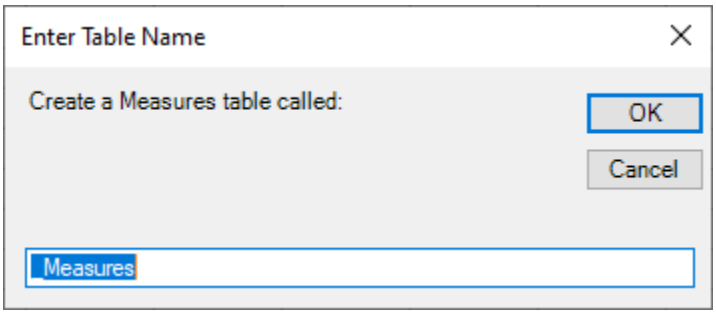


Figure 19 - Enter the name for your Measures table

You may notice that the table name provided by default is "_Measures" not "Measures". This is because "Measures" is a reserved word and cannot be used as a table name in the data model.

At this point, Monkey Tools will check if you already have a table with the name you provided. It is worth noting that – if you do – clicking OK will only replace the Power Query associated with the table. In other words, your existing measures will not be deleted!

Unfortunately, in order to create a proper measure table for your model, there is one other thing left to do, and it's something that we are unable to automate for you as Microsoft hasn't provided us a way to do so. The good news is that Monkey Tools tells you the exact steps needed:

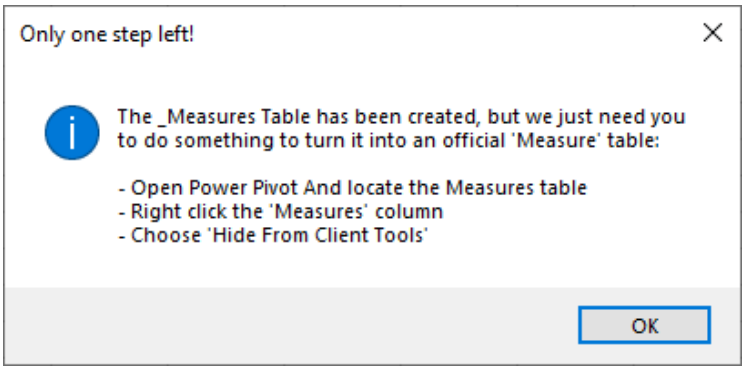


Figure 20 - Steps required to mark your table as a "Measures" table

While ignoring these steps will not cause your DAX formulas to miscalculate, it will cause the yellow "Relationships between tables may be needed" message to show on every PivotTable or PivotChart where you use measures stored on this table.

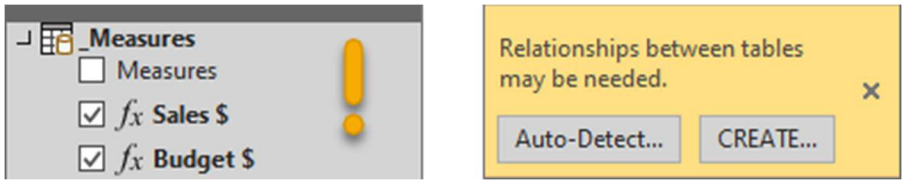


Figure 21 - This "Measure" table is acting as a Dimension table!

To future-proof your model and avoid this irritating message in future, you simply need to:

- Go to Manage Data Model (on the Monkey Tools or Power Pivot tabs)
- Select the newly created measures table
- Right click the "Measures" column and choose "Hide from Client Tools"

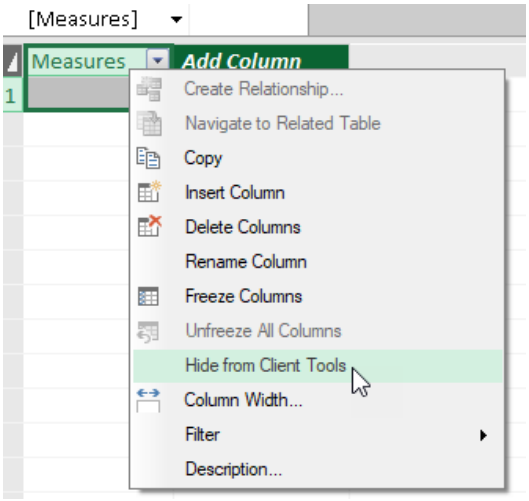


Figure 22 - Hiding Power Pivot columns

The effect of this change is to hide the unaggregated column, letting the data model know that this is a valid Fact table. It will then suppress error messages when you use fields from this table in the values area of any PivotTable.

It should be noted that this behaviour is not specific to Measure tables, but rather Fact tables. If you have any existing PivotTable which are displaying the "Relationships between tables may be needed" error, see Chapter 8 - PivotSleuth, as this feature is specifically designed to help you debug design and relationship errors in your Power Pivot based PivotTables.

4.4 Inject Calendar Generator

Query Monkey's Inject Calendar feature helps you build a calendar table that will automatically expand to cover the entire fiscal year for which you have data:

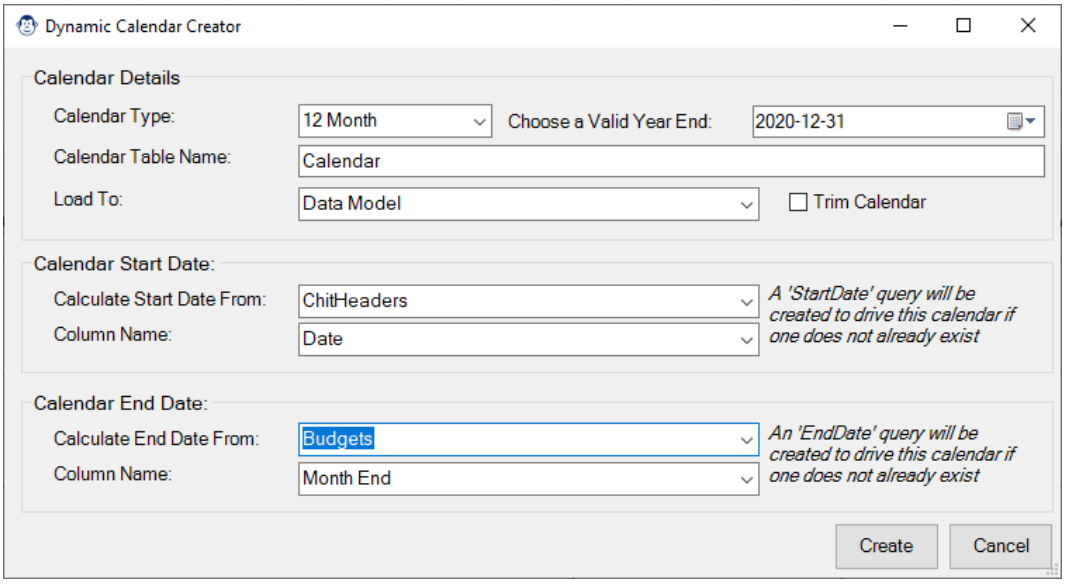


Figure 23 - Injecting a Dynamic Calendar

The calendar supports several common formats, custom year ends, and will even let you trim your dates (rather than return the entire fiscal year), as chosen in the form. Upon clicking Create, it will create three queries for you:

- StartDate (or StartDate364)
- EndDate (or EndDate 364)
- Calendar

The first two will always load as Connection Only queries, with the final query's destination set based on the Load To selection you make in the form. You'll then have a single column calendar table which you can edit to add extra columns you require, as shown in Figure 24, below.

If you intend to convert your Excel file to a Power BI file, be aware that these dynamic calendar patterns will not support DirectQuery.

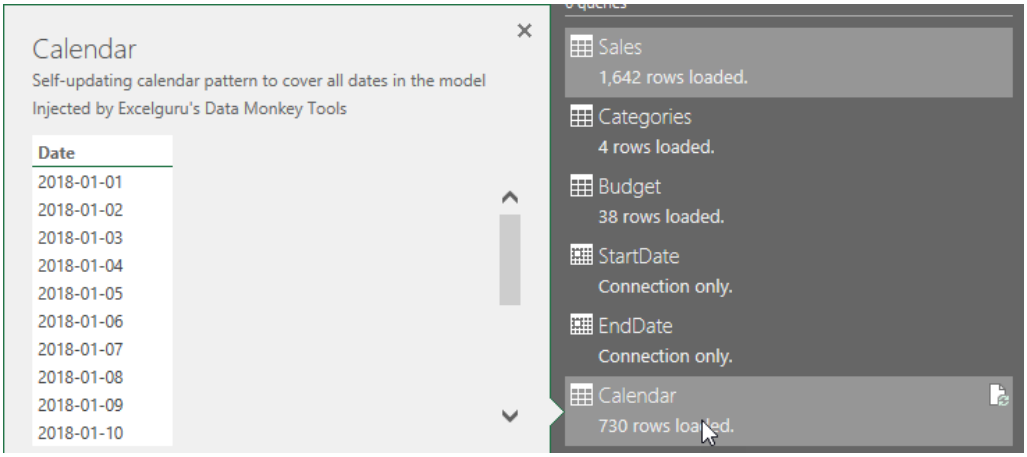


Figure 24 - A single column Calendar table

4.4.1 Calendar Types

The Calendar generator provides templates for:

- 12-Month Calendar tables (with a December 31 or non-standard year end)
- 364-day Calendar tables (including 4-4-5, 4-5-4 or 5-4-4 ISO, or 13-month calendars)

Since the 364-day calendars require PeriodID columns in order to drive their time intelligence patterns, selecting one of these calendars will pop out an additional option in order to add those for you:

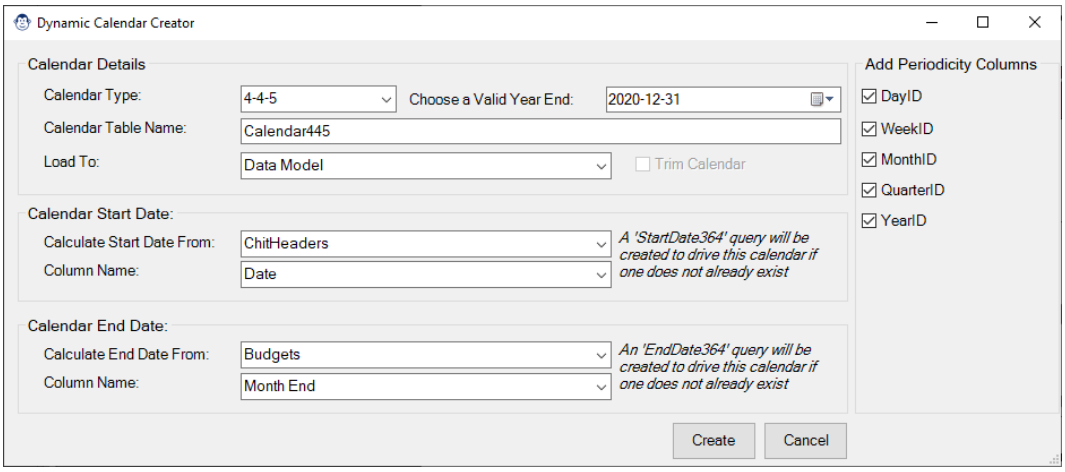


Figure 25 - A 4-4-5 Calendar offering PeriodID columns

4.4.2 Choosing a Year End Date

When creating your calendar, it is important to choose a valid year end date. It's important to realize that the year end doesn't have to be included in your data, it just needs to be any valid fiscal year end date in your corporate history. This is intended to make it easier for you, as you can select the current year end, even if you are reporting on data from four years prior.

4.4.3 Calendar Table Name

QueryMonkey will automatically provide you a descriptive name for your Calendar table, but this name is editable should you prefer something different. The implication here is that you can actually create and link multiple calendar tables to your model to slice and dice your dates in different ways!

4.4.4 Load To

This drop down allows you to configure the load destination for your table. By default, it will load to the data model, but you can change that to Connection Only, Worksheet, or Worksheet & Data Model should you need to do so.

4.4.5 Choosing Start and End Dates

In these two sections, you need to select the columns that will always contain the earliest and latest dates that will ever be found in your model. Tables like a Sales table are often good sources for a Start Date, as it's highly likely that there will be transactions for the year you are analyzing. Likewise, Budgets can also be a great option for an End Date, as budgets are typically done before Sales occur. Having said this, if your organization ever has budget cycles that don't get uploaded until part way through a year due to a lag in approvals, Budgets would NOT be a good candidate!

The QueryMonkey can pull its starting and ending dates from any valid date column. You'll find the "Calculate Start Date From" box pre-populated with all the query names in the model. The Column Name box, however, shows one of three things:

1. A list of all fields formatted as dates,
2. A message that reads "No date columns in table!", or
3. A blank cell

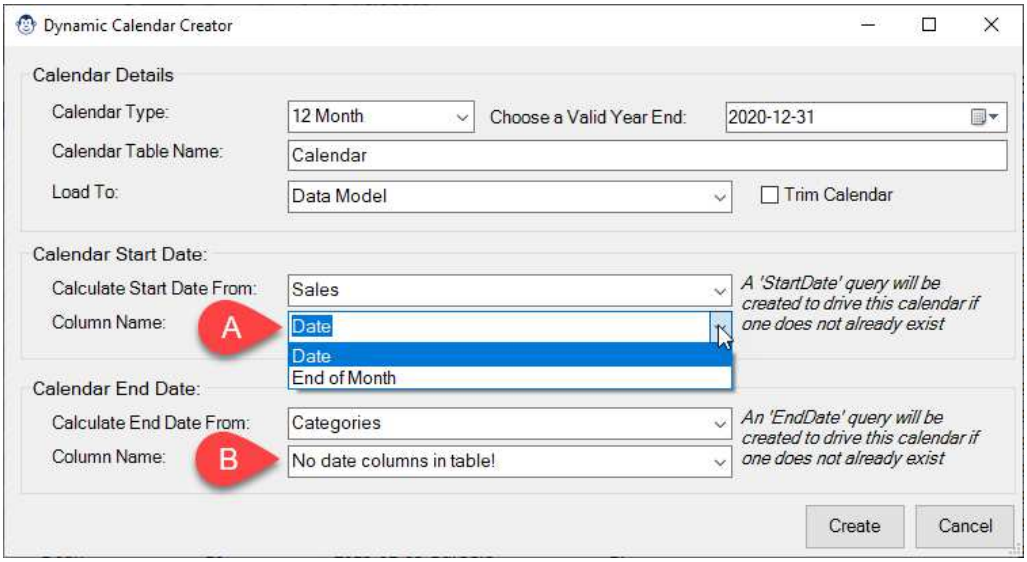


Figure 26 - Sales provides two choices for dates, but Categories shows none

The first two are self-explanatory. A blank will occur if you select a staging query (one loaded as Connection Only), as we cannot guarantee that we know which columns are in the data. You can type the column name manually and it will be used, just make sure you case it and spell it correctly to avoid an error when the queries are created.

4.4.6 Loading Your Calendar Table

When you click okay to load the calendar table, you'll see a couple of things happen:

- The Queries & Connections pane will pop out to display the progress
- The Calendar Creator form will show a red message at the bottom letting you know things are happening:

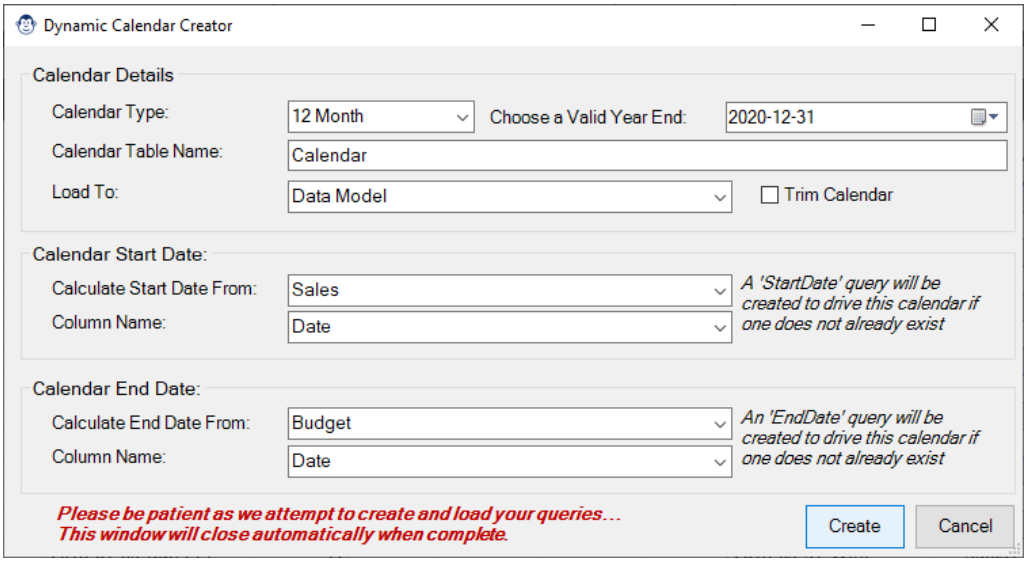


Figure 27 - Please wait while the calendar table loads

Once complete, the form will close automatically, and you'll see your new queries in the Queries & Connections pane. For 12-month calendars, it will have a single column of dates, and for 364-day calendars, it will also contain any PeriodID columns that you selected when building the calendar.

At this point you can add as many other date columns as you need by editing the Calendar query and following the next two steps for every date column you wish to add:

- Select the Date Column
- Go to Add Column → Date → <the date format you wish to add>

Once finished, go to Home → Close & Load to commit the changes.

4.5 New SmartFolder Query

This QueryMonkey command makes it super easy to create a new SmartFolder solution for an existing workbook. Simply click the button, and it will insert all the required components, including the following:

- A parameters table with the path to the current workbook
- The fnGetParameter function
- The fnSmartFolder function
- A new query that leverages fnSmartFolder to list all files in the given path

Should some of these pieces exist, QueryMonkey will just re-use them. And if they aren't there, they'll be created for you!

Upon launching this button, QueryMonkey will leverage/create the parameter table and two required functions, then prompt you to name your new query:

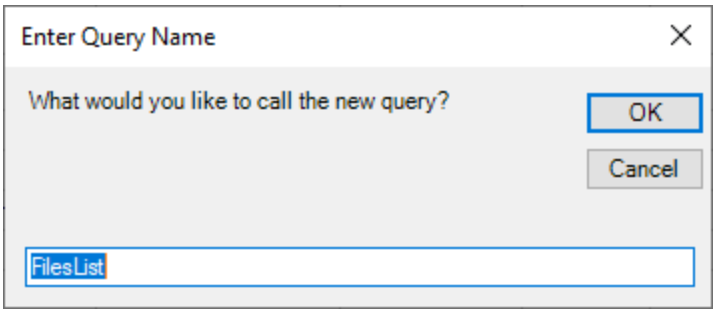


Figure 28 - What would you like to call your new query?

Once you click okay, it will build that query, linking it to your parameter table. You'll then find a new "Connection Only" query in the workbook with this name, which lists all of the files in the folder, as shown in Figure 29. From there, it's a simple matter of right clicking this query, choosing Reference, and combining the files as you need.

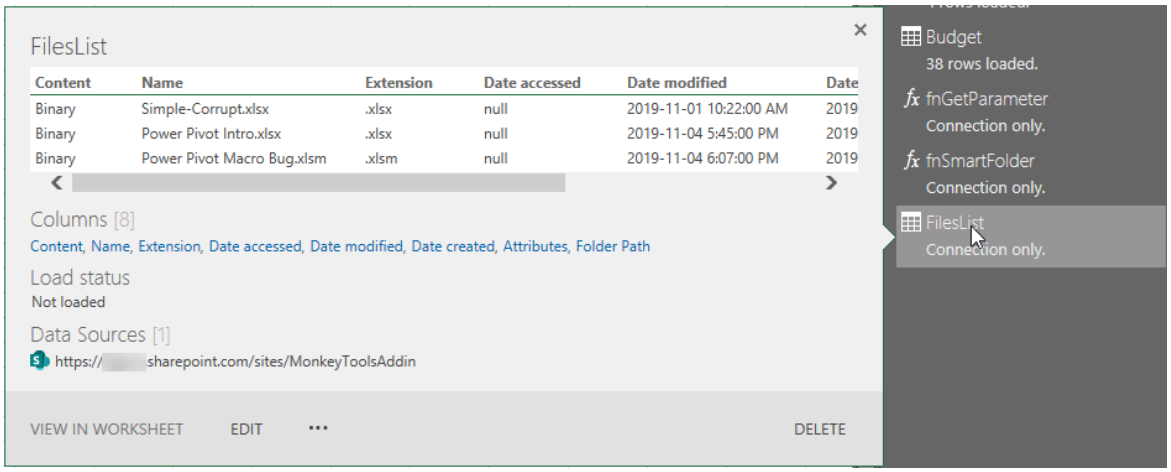


Figure 29 - Three queries were added to the workbook and show the files in the SharePoint hosted folder

4.5.1 If the Query Shows an Error

If the workbook has never had an external data source, it may not have a privacy level set for the workbook itself. This will manifest in the query showing an exclamation icon and an error prompting you to set privacy levels. This needs to be set once and will stick with the workbook thereafter. To do this:

- Go to Get Data → Data Source Settings → Select Current Workbook → Edit
- Set the privacy level to your preference
- Click OK → Close

At this point, you should be able to right click and refresh the new query without issue.

An image of the Privacy dialog is shown in Figure 30, below:

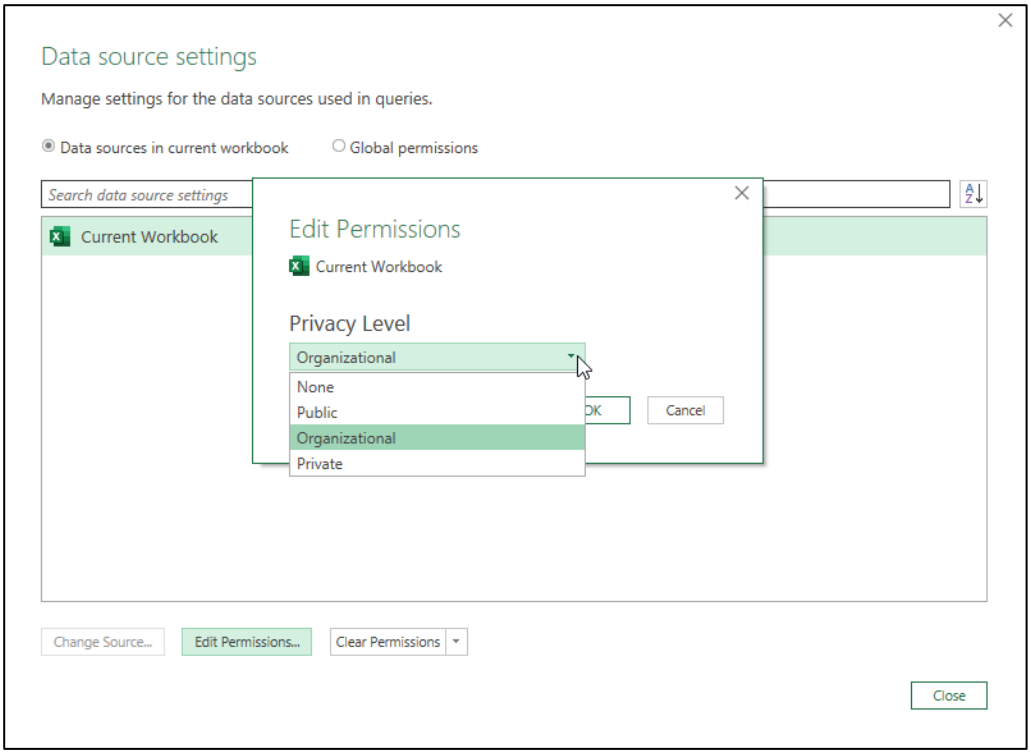


Figure 30 - Setting the Workbook's privacy levels

5 DestinationSleuth

The DestinationSleuth is intended to solve two main issues with regards to working with Power Query in Excel:

1. Help identify *where* a query has been loaded to
2. Change multiple query load destinations at one.

5.1 Identifying Query Load Destinations

While the Queries & Connection pane tells you the general state of a Power Query, that state is restricted to the fact that it loaded as a Connection Only, or to a destination with 8 rows loaded. That's great, but how do you know which destination it went to?

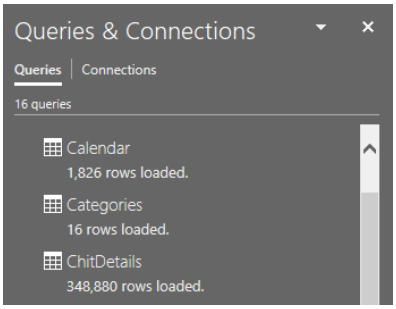


Figure 31 - These queries are loaded... but to where?

DestinationSleuth dialog provides more information that the standard Queries & Connections pane available in Excel by listing and colour-coding the ultimate load destination of each query:

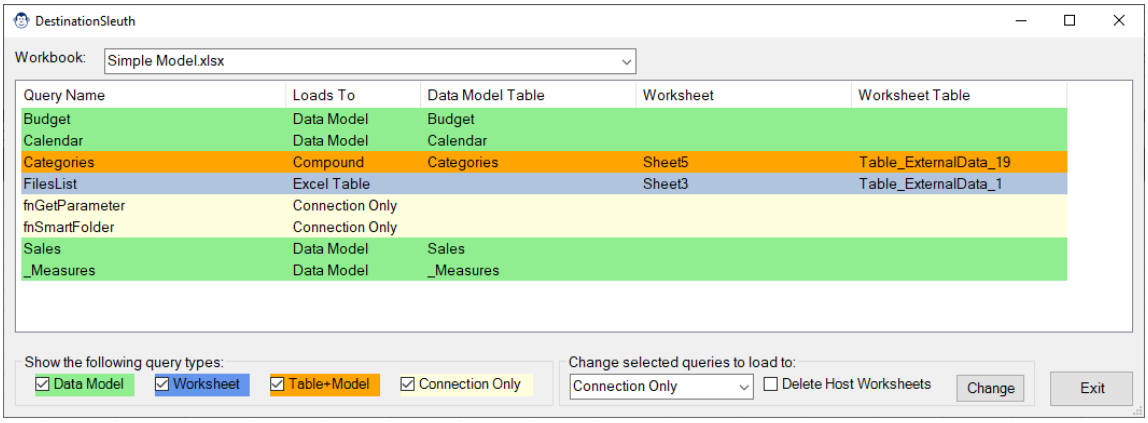


Figure 32 - The DestinationSleuth

The colours used represent:

- Green: Query is loaded to the Data model (only)
- Blue: Query is loaded to the worksheet (only)
- Orange: Query is loaded to the Data model and a worksheet
- Yellow: Query is a connection only

Another useful feature of the DestinationSleuth is that you can filter to include or exclude certain load destinations. This can be helpful if you'd like to see, for example, only queries loaded to worksheet tables.

5.2 Changing Load Destinations

Changing load destinations of queries in Excel is relatively straight forward once you know how; simply right click it in the Queries and Connections pane (not inside Power Query) and choose Load To… and you'll be able to reconfigure it as needed.

But what if you want to change the load destination for multiple queries at once? As Excel has no ability to do this natively, you must change each load destination individually. (Right click each query in the Queries & Connections pane, go to Load To, configure the load destination and… wait.) And that is where DestinationSleuth can help.

5.2.1 Toggling Load Destinations

Changing the load destinations for one or more queries involves the following steps:

1. Select what type of Load Destination you want your queries to embody
2. Select the queries you want to change
3. Click Change

In the image shown here, the user is requesting to change four queries – currently loading to Connection Only – to load to the Data Model:

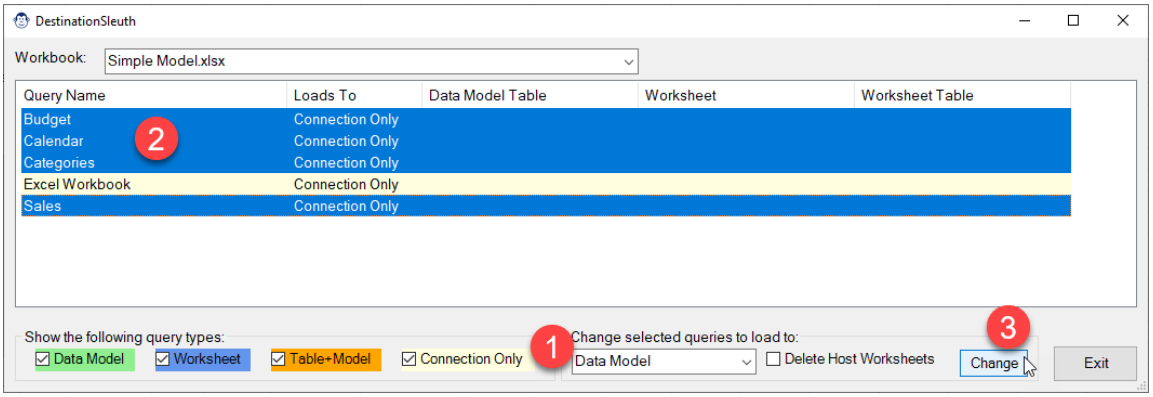


Figure 33 - Changing Load Destinations for Multiple Queries

Upon clicking Change, the Queries & Connections pane will open, and you can watch as each of the connections gets loaded to the appropriate destination. Once complete, the DestinationSleuth will refresh with the updated load destinations, as shown in the image below.

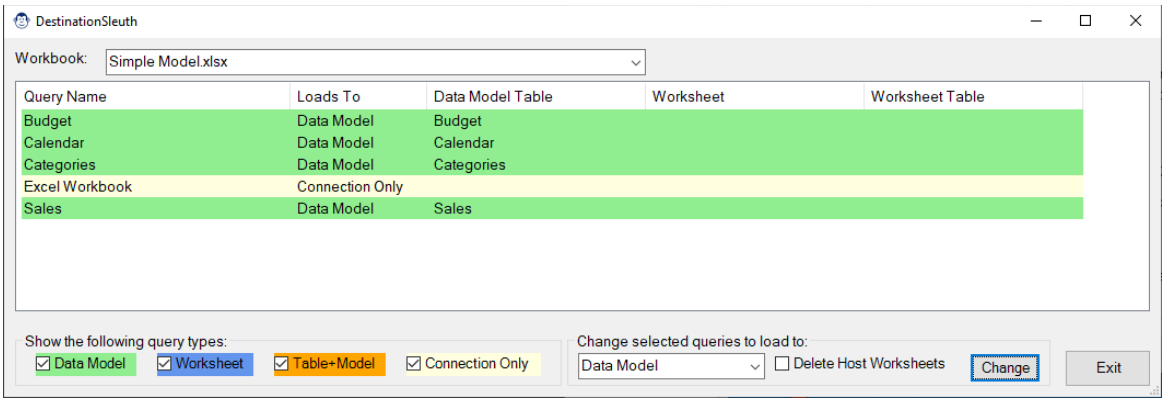


Figure 34 - The DestinationSleuth reporting the modified load destinations

5.2.2 Load Destination Options

When changing Load Destination options, there are a few features that you should be aware of:

5.2.2.1 Selecting Multiple Queries

Like in Windows, selecting multiple queries can be done by holding down the SHIFT or CTRL buttons as you select them. Queries can also be individually unselected by holding down CTRL and clicking on them.

5.2.2.2 Available Load Destinations

DestinationSleuth will allow you to change any query type to one of four possibilities:

- Connection only (removing all data model or worksheet connections)
- Data model (removing any table connections
- Table on a New Worksheet (removing any data model connections)
- Table and Data Model (adding the required connections that are missing)

At this time, DestinationSleuth does not support loading tables to individually named worksheets.

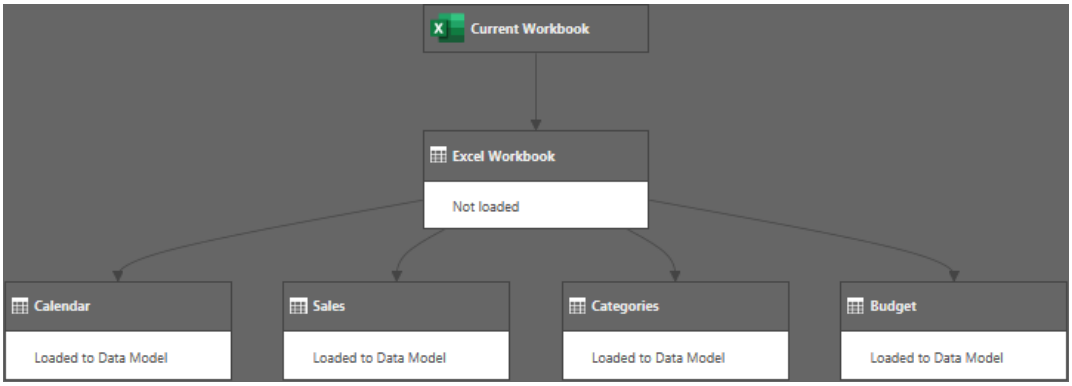
5.2.2.3 Delete Host Worksheets

If you've ever accidentally created a table, changed it to Connection Only, then manually deleted the worksheet it created, you'll love this option. Not only will it remove the table when you change the load destination, it will also delete the worksheet the table was hosted on. This can be very useful if you accidentally landed multiple queries to individual worksheets in error.

This setting is off by default, as Monkey Tools makes no check to see if there is anything else on the worksheet. You should only check this box if you are certain that the table is the only thing on the hosting worksheet listed in the Worksheet column.

5.2.3 Practical Use Case

Assume that you have connected to a table in an Excel file and referenced the table four times in the Power Query editor to create the required Fact and Dimension tables for your model. So far so good, but now you have a challenge. What you want is to load four of these table to the data model, and one to a connection as shown below, but you created all of these queries in one session inside the Power Query editor:



The problem, of course, is that you get to choose one load destination for all these tables. Which do you choose?

- Load all five queries to the data model, then change the destination of the Excel workbook query to remove it from the Data Model?
- Load all five queries as Connection Only, then go back and add each of the four required tables to the data model individually?

While it is frustrating that you can't choose multiple load destinations inside the Power Query editor, it is even more frustrating that there is no way from the Queries & Connections pane to change the load destinations of multiple tables at once.

So how to deal with this scenario? Here's what we do:

- Load them all to connection only, and
- Launch DestinationSleuth to change the destination of the four queries required in the data model.

Why? Simple. Its all about speed. Consider a scenario where you have a complicated query chain that creates 10 tables and each has 10,000 rows of data, but where only 5 need to be loaded to the data model. Which will be faster?

- Wait for all 10 to load to the data model, change 5 of them to connection only and wait for them to unload, or
- Load all 10 to connection only, then use DestinationSleuth to change the 5 you need to load to the data model?

Its obviously the latter, as you don't have to wait for the data to unload when removing unneeded tables from the data model. In fact, we believe in this so strongly, that we set our load destinations for all queries to load to Connection Only by default. That way, if we ever make a mistake, it happens very quickly, and we can repoint the tables to the correct destination with DestinationSleuth.

If you'd like to do the same, go to:

- Data tab → Get Data → Query Options → Data Load
- Choose "Specifiy custom default load settings"
- Clear the Load to Worksheet and Load to Data Model checkboxes
- Click OK

5.2.4 Understanding What Happens to Connections Upon Change

It is important to realize that there are two different types of connections inside an Excel workbook; those that populate the data model, and those that don't. The reason this is important to understand is that it can have an impact on what happens to data tables when a load destination is changed.

A point in case is when you change a query from "Load to Table" to become "Load to Table and Data Model", the original Table connection must be deleted and recreated as the internal connection string is different. Naturally, this will cause issues for any formulas or macros that are pointed at the historical table.

The following matrix describes what will happen with worksheet tables during a change.

Change From	Change To	Original table deleted?
Table	Connection Only	Yes
Table	Data Model (only)	Yes
Table	Table & Data Model	Yes, then recreated
Table & Data Model	Connection Only	Yes
Table & Data Model	Data Model (only)	Yes
Table & Data Model	Table (only)	Yes, then recreated
Data Model	Connection Only	Yes
Data Model	Table (only)	Yes
Data Model	Table & Data Model	No, just added to connection

Figure 35 - Understanding Impacts of Destination Changes on Tables

This behaviour is completely consistent with changing load destinations manually but is something that you should be aware of.

6 QuerySleuth

The QuerySleuth form was designed to address several shortcomings in Power Query's "View Dependencies" feature (not the least of which that it was several clicks away from the Excel canvas). In our opinion, that feature was unusable with large volumes of queries, so we needed to design a better view for tracing query relationships in our models.

6.1 View of the Query Relationship Window

The QuerySleuth is packed with information, as you can see here:

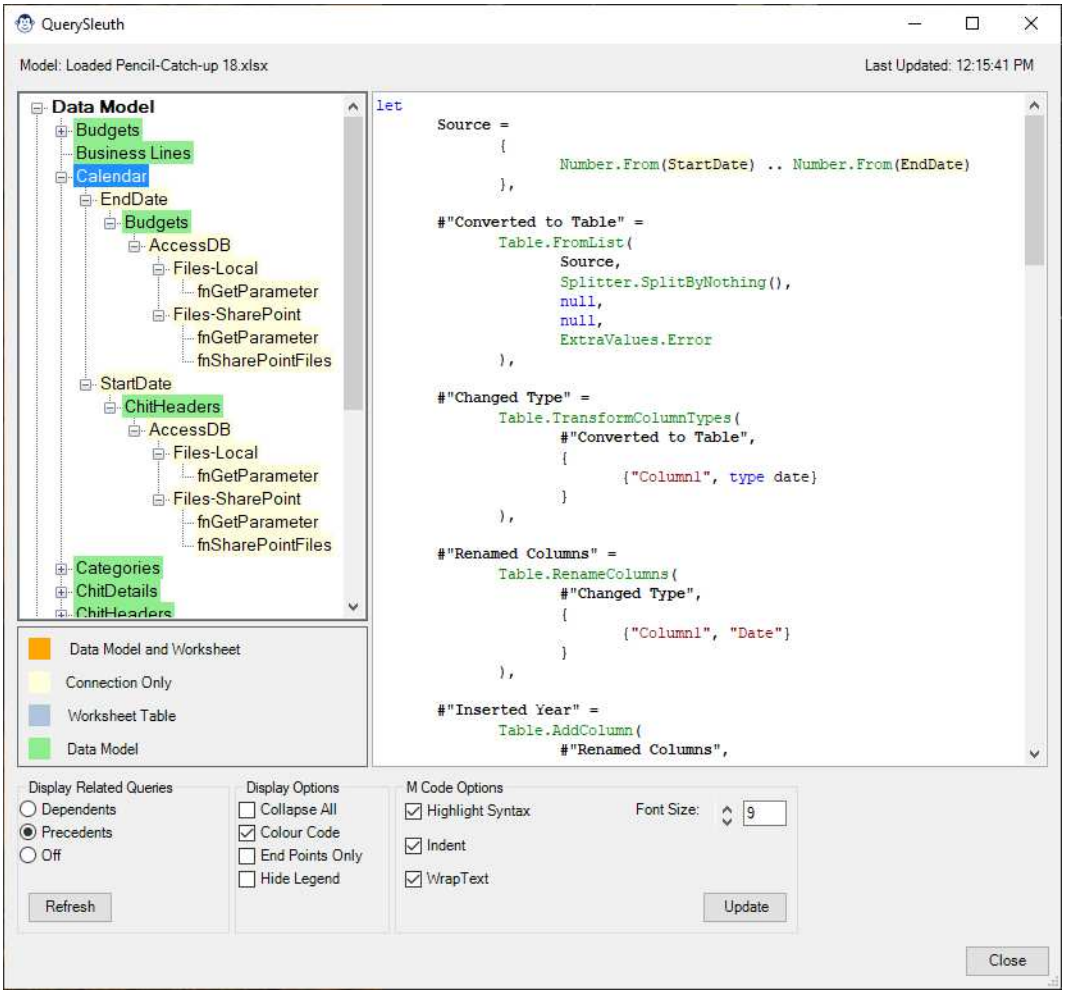


Figure 36 - The QuerySleuth

Some of the key things to be aware of in this window:

- **Model:** This lists the model name, allowing you to see which file QuerySleuth is currently analyzing.
- **Last Updated:** This lists the time the QuerySleuth last queried Excel or Power BI to get the model details.

- **The Query Navigator:** Shown at left, this "tree view" lists all queries with their dependencies or precedents as determined by the selection in "Display Related Queries" at the bottom of the form. (The default is to show Precedents.)
- **The M Code Window:** This is the large area on the right side of the screen. It is fully editable and shows you a large portion of information about your query. This area is discussed in more detail in the section on Controlling the Query Navigator.
- **The Options Area:** At the bottom of the form are a variety of configurable options to help you get the most out of the form, each of which is discussed below.

6.2 Controlling the Query Navigator

The purpose of the Query Navigator is to show how individual queries are related to each other. There are three options for this area of the form, each of which are controlled via the radio buttons in the "Display Related Queries" section of the Options area (at the bottom left of the form). These options work as follows:

6.2.1 Precedents

Choosing this setting means that Monkey Tools will scan the model and show all precedents that feed the selected query. This view is very useful for figuring out the entire chain of queries that fed your data model or Excel table. Also, this can be a useful view to trace back and figure out where you eliminated a key piece of data.

6.2.2 Dependents

The Dependents view shows all queries that depend on the selected query. This can be useful for figuring out what queries you may affect if you change a given query.

6.2.3 Off

Setting the option to "Off" will still list each query in the Query Navigator but will not show +/- buttons for dependent or precedent queries at all.

6.2.4 The Refresh Button

As QuerySleuth is non-model (unlike the Power Query window, Monkey Tools will allow you to interact with Excel), it means that you could potentially change queries in Excel while the QuerySleuth is open. To bring QuerySleuth back in sync with your file, you may want to refresh the Power Query model, which can be done by clicking the "Refresh" button.

6.2.5 Hidden Features

One thing that isn't readily apparent from looking at the form is that you can expand an entire Query Navigator node's precedents or dependents by double clicking on that node. It's just a little feature that makes it super easy to drill into the query you want to examine!

6.3 Understanding the M Code Window

The M code window is the heart of understanding what your query does, and we've added loads of features to make this easier to understand. In addition, you should know that the code here is fully editable and can even be written back to Excel by clicking the "Update" button at the bottom of the form.

6.3.1 M Code Indentation

By default, we indent your M code (although you can turn this off in the options below). When you read your code, you'll see that the step name is indented by one tab, and the formula begins indented on the next row. Our general rule of thumb is to indent anything between pairs of (), [], or {}, except in certain cases as defined by our indentation algorithm.

6.3.2 Syntax Highlighting

While our syntax highlighting doesn't match Power Query's editor exactly, it's pretty close, and will hopefully help you understand your code a little bit better. One thing we do that the Power Query editor doesn't, is that we highlight precedent query names so that you can see not only that they are a precedent, but also what type they are (as defined by the colour coding options for the Query Navigator). Have a look at the query shown here:

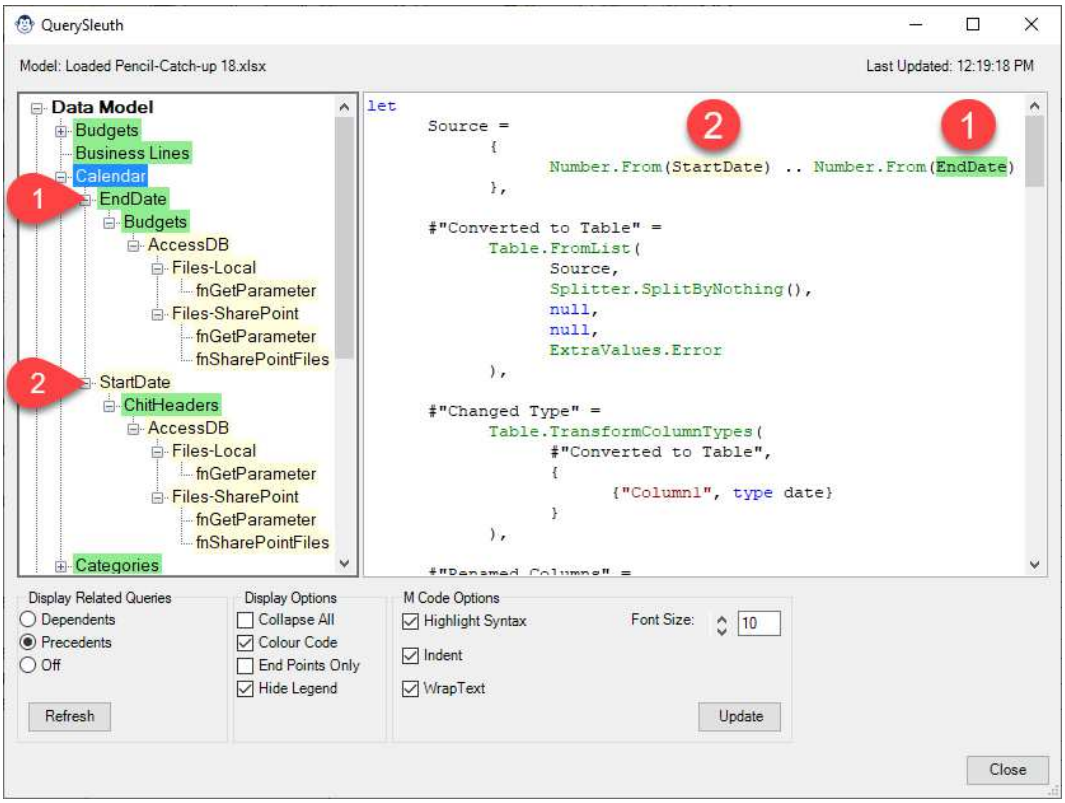


Figure 37 - Understanding M Code Highlighting

The selected query in Figure 37 is Calendar. In the Query Navigator, we can see that this query has two precedents: EndDate and StartDate. Given that EndDate is shown in green, we know that this query loads to the data model, while StartDate is yellow, and is only loaded as a connection.

But now move to the M Code window. Notice how the StartDate query (shown by the 2) has a yellow background, matching the view from the Query Navigator? In addition, EndDate (shown by the 1), has a green background. Even without the numbers on the M code window, you'd have to agree that this makes it very easy to scan the query and recognize anything that is coming from another query.

These options can, of course, be toggled via the M Code Options listed at the bottom of the form.

6.4 Understanding the QuerySleuth Options

6.4.1 Display Options

The Display Options control how the Query Navigator displays the queries. It is also worth noting that Monkey Tools saves your most recent selection as a default, so it preserves your preferred view every time you open the form.

6.4.1.1 Collapse All

Unchecking the Collapse All checkbox will collapse all queries back to the main heading levels as shown here:

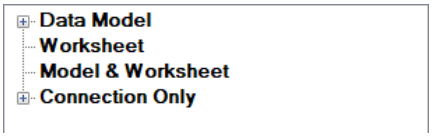


Figure 38 - Collapse All loads the form with all nodes collapsed

The nodes can still be expanded by clicking on the + button.

6.4.1.2 Colour Code

By default, the queries are all colour coded as follows:

- Yellow: Query is loaded as a Connection Only
- Green: Query is loaded to the data model
- Blue: Query is loaded to an Excel table
- Orange: Query is loaded to both the data model and an Excel table

If you find this view too noisy, you can mute the colours by unchecking the “Colour Code” checkbox.

6.4.1.3 End Points Only

This option suppresses any queries which have no precedents or dependents (as determined based on the “Display Related Queries” choice). A sample of the difference can be seen in Figure 39:

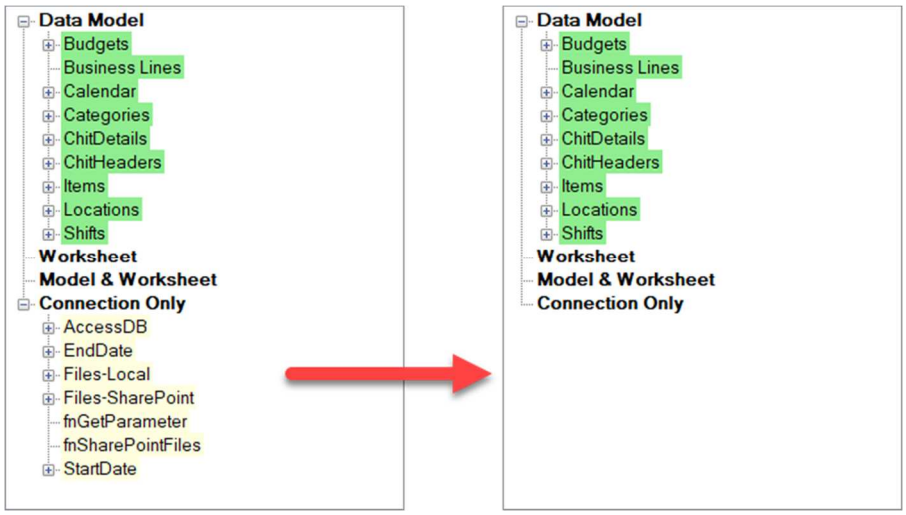


Figure 39 - View on right shown with End Points Only

The key takeaway from this view is that all the Connection Only queries are used to feed other queries in the model. If they weren't, they still show up in the Connection Only section. This can be a very useful setting for eliminating noise, as well as tracing Connection Only queries that are not used at all (and can therefore be eliminated from the model).

6.4.1.4 Hide Legend

While the legend is useful for understanding the query load destinations, it does take up a lot of space. Once you understand what the colours mean, it's helpful to suppress, which you can do by unchecking the box. We also save this selection as your default, so you won't have to un-check the box every time you load QuerySleuth.

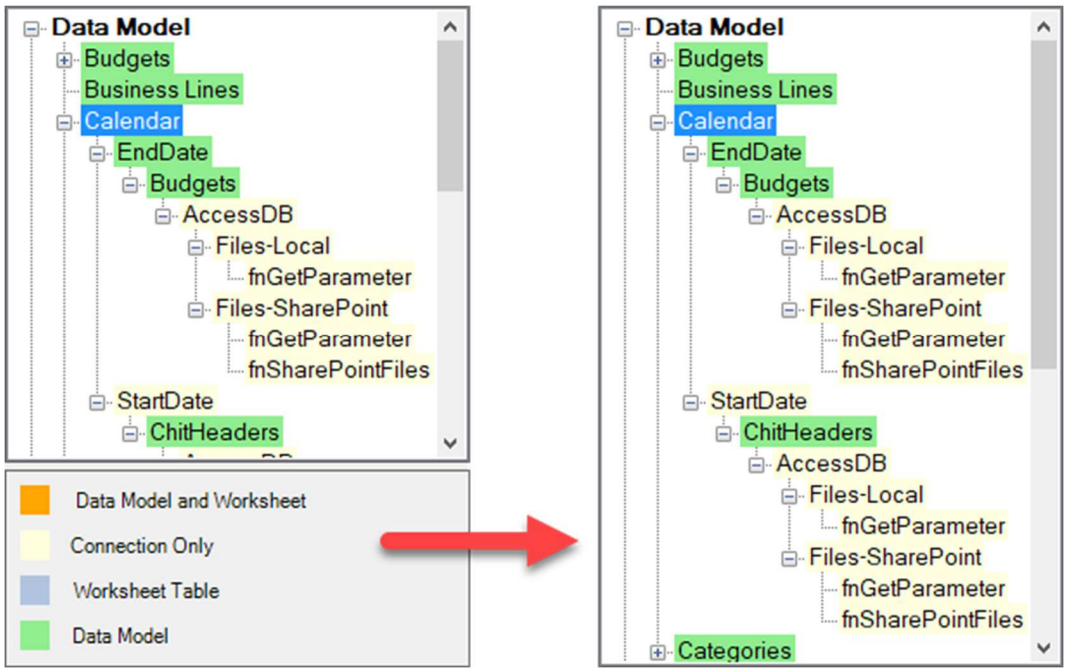


Figure 40 - The effects of hiding the QuerySleuth legend

6.4.2 M Code Options

The M Code Options area allows you to toggle several features that you may or may not want active in the M code window at any given time:

6.4.2.1 Highlight Syntax

If the M code window is "too colourful" for you, you can turn this feature off by unchecking this box.

6.4.2.2 Indent the M Code

We recognize that M indentation isn't for everyone. While it can make your queries much easier to read, it can lead to a lot of scrolling. Should you want to disable the indentation at any given time, just uncheck the Indent box. (Remember, you can always turn it back on later!)

6.4.2.3 Wrap Text

The Wrap Text feature is on by default but doesn't really have much of an impact unless you turn the M indenting feature off. Figure 41 below displays the query in an un-indented form, but with line wrapping still active. Figure 42 shows what happens when you also uncheck the Wrap Text checkbox. And if you

want to send it back to the classic days of Power Query editing… uncheck the Highlight Syntax box as well!

```
let
    Source = {Number.From(StartDate)..Number.From(EndDate)},
    #"Converted to Table" = Table.FromList(Source,
Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    #"Changed Type" = Table.TransformColumnTypes(#"Converted to
Table",{{"Column1", type date}}),
    #"Renamed Columns" = Table.RenameColumns(#"Changed
Type",{{"Column1", "Date"}}),
    #"Inserted Year" = Table.AddColumn(#"Renamed Columns",
"Year", each Date.Year([Date]), Int64.Type),
    #"Inserted Quarter" = Table.AddColumn(#"Inserted Year",
"Quarter", each Date.QuarterOfYear([Date]), Int64.Type),
    #"Inserted Month" = Table.AddColumn(#"Inserted Quarter",
"Month", each Date.Month([Date]), Int64.Type),
    #"Inserted Month Name" = Table.AddColumn(#"Inserted Month",
"Month Name", each Date.MonthName([Date]), type text),
    #"Extracted First Characters" =
```

Figure 41 - Un-indented query WITH line wrapping

```
let
    Source = {Number.From(StartDate)..Number.From(EndDate)},
    #"Converted to Table" = Table.FromList(Source, Splitter.SplitB
    #"Changed Type" = Table.TransformColumnTypes(#"Converted to Ta
    #"Renamed Columns" = Table.RenameColumns(#"Changed Type",{{"Co
    #"Inserted Year" = Table.AddColumn(#"Renamed Columns", "Year",
    #"Inserted Quarter" = Table.AddColumn(#"Inserted Year", "Quart
    #"Inserted Month" = Table.AddColumn(#"Inserted Quarter", "Mont
    #"Inserted Month Name" = Table.AddColumn(#"Inserted Month", "M
    #"Extracted First Characters" = Table.TransformColumns(#"Inser
    #"Inserted End of Month" = Table.AddColumn(#"Extracted First C
    #"Inserted Day" = Table.AddColumn(#"Inserted End of Month", "I
    #"Inserted Day Name" = Table.AddColumn(#"Inserted Day", "Day N
    #"Inserted Day of Week" = Table.AddColumn(#"Inserted Day Name"
    #"Extracted First Characters1" = Table.TransformColumns(#"Inse
    #"Added Custom" = Table.AddColumn(#"Extracted First Characters
```

Figure 42 - Un-indented query WITHOUT line wrapping

6.4.2.4 Font Size

The Font Size controls allow you to scale the font size in the M code window. You can manually enter your desired font size in the box or use the arrow keys to scroll it up and down to find the size you prefer.

6.4.2.5 Update Button

The last option on this form is the Update button, which allows you to write the current query's state back to the Excel file. It is important to be aware that the display on in the Monkey Tools M code window has nothing to do with how the query is contained in Excel's query formula. To prove this, look at one of your queries in the QuerySleuth, then edit the query, go to the Advanced Editor, and compare the results. It's a good bet that they'll look different.

If you'd like to update them to be the same, or if you made any tweaks to the query in order to get it right, you can click the Update button to write the updated query back into the Excel file. It's as simple as that!

7 TimeSleuth

One of the realities of building complex models is that we build models which take a long time to refresh. While timing with a stopwatch is easy, working with complicated query chains makes things harder, as your stopwatch can only time the global refresh.

Have you ever wanted to:

- Time all queries in a refresh to see which of your child queries is impacting your load time the most?
- Create two queries and compare them to see which loads faster?

That is the purpose of this tool; it to allow you to test the global or individual refresh times of queries in the workbook.

7.1 Using the TimeSleuth

Immediately upon launching this tool, the form will pre-populate with the active workbook, and list all its queries which load to an output destination[3]. Should you wish to analyze a different (open) workbook, simply select it from the drop-down list. At this point, you will then be presented with a list of all the queries in the workbook and will be able to time their refresh performance.

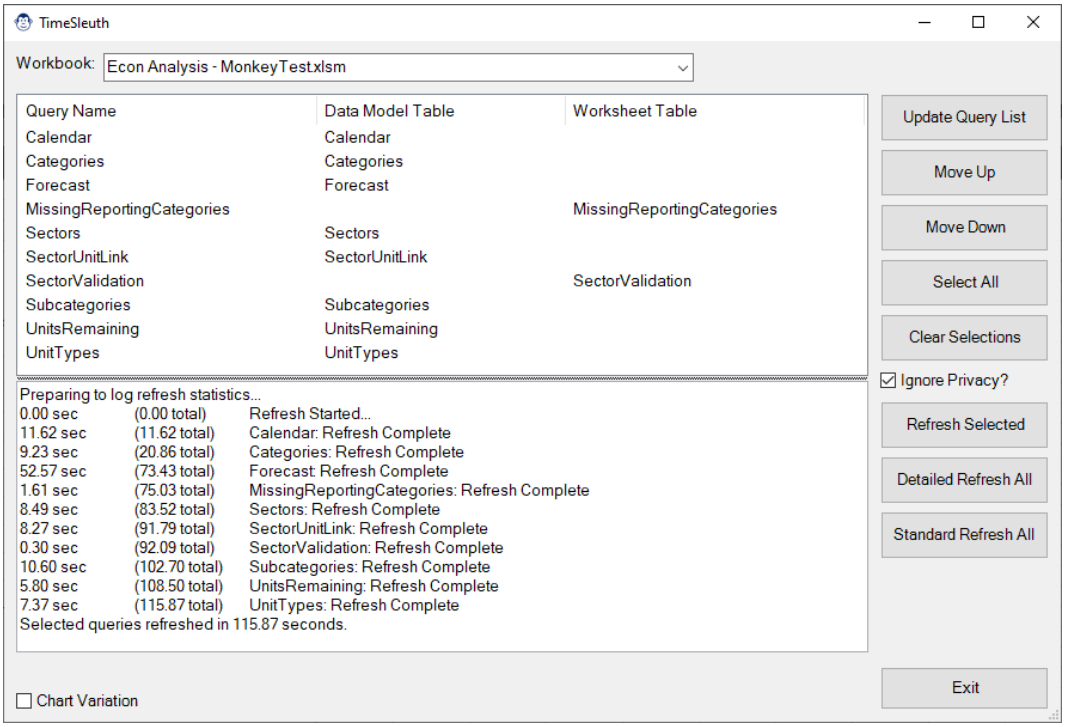


Figure 43 - TimeSleuth running a "Detailed Refresh All"

[3] Queries that load as Connection Only are excluded, as we cannot test their refresh times. To test a Connection Only query, change its load destination to Table or Data Model before running this tool.

7.1.1 Quick Overview of the TimeSleuth

There are three main ways to use the TimeSleuth in Monkey Tools. They are:

1. **Standard Refresh All:** This button replicates the Refresh All feature of Excel, allowing you to time a normal refresh
2. **Detailed Refresh All:** This button refreshes each loadable query individually, recording the individual and cumulative times of the refresh. This feature is useful for working out which queries could benefit from performance tuning.
3. **Refresh Selected:** This button allows you to refresh selected queries only, allowing you to ignore queries that aren't relevant to your current goals.

Working with each of these queries is described in more details below.

7.1.2 The Queries & Connections Pane

During the operation, you will notice that we open the Queries & Connections pane. The reason for this is that it can take a long time to refresh the queries, and watching the Queries & Connections pane is the only true indicator you have that something is happening.

7.1.3 Understanding Power Query Refresh Times

Before you dive into using this tool, it is important to be aware that the overhead of Power Query's mashup engine can fluctuate. Because the Power Query engine is "single threaded", any related query chain can only be refreshed on a single processor core. Due to this fact, the engine is highly susceptible to other programs running on the same machine. In other words, if your OneDrive sync client kicks off a huge sync on the same core that your refresh is running on, it could significantly impact the load time for that run.

Given the above, this feature is of limited use for performance tuning queries with low refresh times. It is better suited to timing long running queries to see which is causing the bottleneck in your overall chain. The main focus is to get the overall magnitude of a query refresh, not try to shave out a millisecond at run time, as each run will be susceptible to other running processes.

To get the most accurate read of a query refresh time, we do recommend trying to eliminate "processor noise" during testing. To do this, shut down all unneeded applications and focus just on the timing task at hand.

7.2 Toggling Privacy

Another feature that can have a significant impact on query refresh times is Power Query's Privacy setting. We allow toggling this setting via the checkbox on our form so that you can test the effect of query load times with this setting turned on or turned off.

> *Pro Tip: Based on our "do no harm" philosophy, we re-set your workbook to its original privacy state upon closing the form, preventing you from accidently changing it permanently while testing. If you want to change the privacy, you'll need to do it manually by toggling the setting shown below in Figure 44.*

The "Ignore Privacy?" checkbox on the right side of the form will inherit the setting that is defined for the workbook under the following area:

Get Data → Query Options → Current Workbook → Privacy

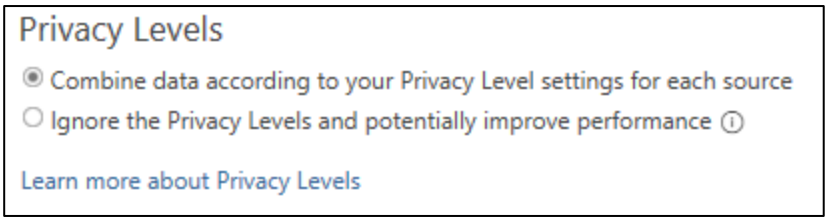


Figure 44 - Privacy Options for the Current Workbook

If this option is greyed out, it means that your workbook is locked to a specific setting. In order to toggle this checkbox, your global privacy settings must be set as shown in Figure 45, which you can access via:

Get Data → Query Options → Global → Privacy

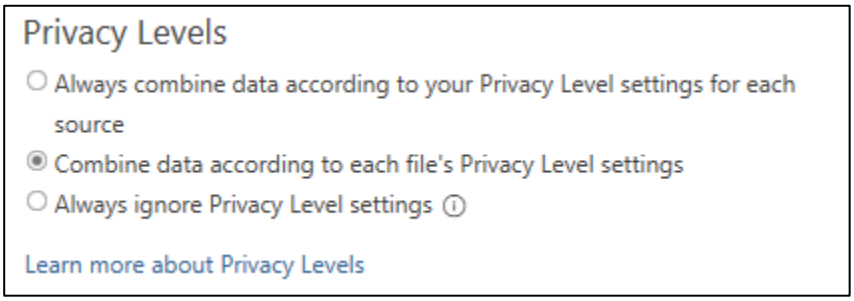


Figure 45 - Configuring your Global Privacy settings

During the process of testing refresh times, it is highly likely that you'll find the performance to be significantly faster with Privacy disabled. It's very tempting to just disable Privacy on a global basis, but this is not recommended.

In order to protect you, Power Query tries to validate the privacy levels of each data sources to be combined, which is why you'll occasionally get prompted to define Privacy Levels (public, private or organizational) for your data. The intent of this feature is to prevent you from combining public data sources with private or organizational data, thereby preventing you from accidentally leaking information.

Here's an example…

In Canada, every person has a Social Insurance Number (or SIN) that we provide to an employer. This is our unique identifier that is associated with our earnings and reported to the Canada Revenue Agency. It's obviously a pretty important and sensitive piece of information. One of the things that many people don't realize, however, is that there is an algorithm that can be run on any SIN number to ensure that it is valid.

Assume you have an Excel spreadsheet of employee SIN numbers, and you try to send it to a Web API to validate the number. If you've left your global Privacy Levels as shown in Figure 45, you would get prompted to define the Privacy Level of the Excel spreadsheet (Organizational) and the web service (Public). At this point you would be told that you can't combine Organizational and Public data. Why is

this a big deal? Because you shouldn't be sending your employee's private SIN numbers outside the corporate firewall unencrypted!

The challenge, however, is that even if all your data resides inside the corporate firewall, privacy still gets checked for your data sources, and this has a significant speed impact. For this reason, we recommend that you only ever toggle Privacy on a workbook by workbook level, and only set it to Ignore when you know that all data sources reside within the corporate firewall.

So, go ahead and test it! But only turn it off when it is safe to do so!

7.3 Working with the Standard Refresh All button

As mentioned earlier, this button emulates and times the regular Refresh All behaviour triggered via Excel's Data → Refresh All button. To use it, simply load the form and click "Standard Refresh All":

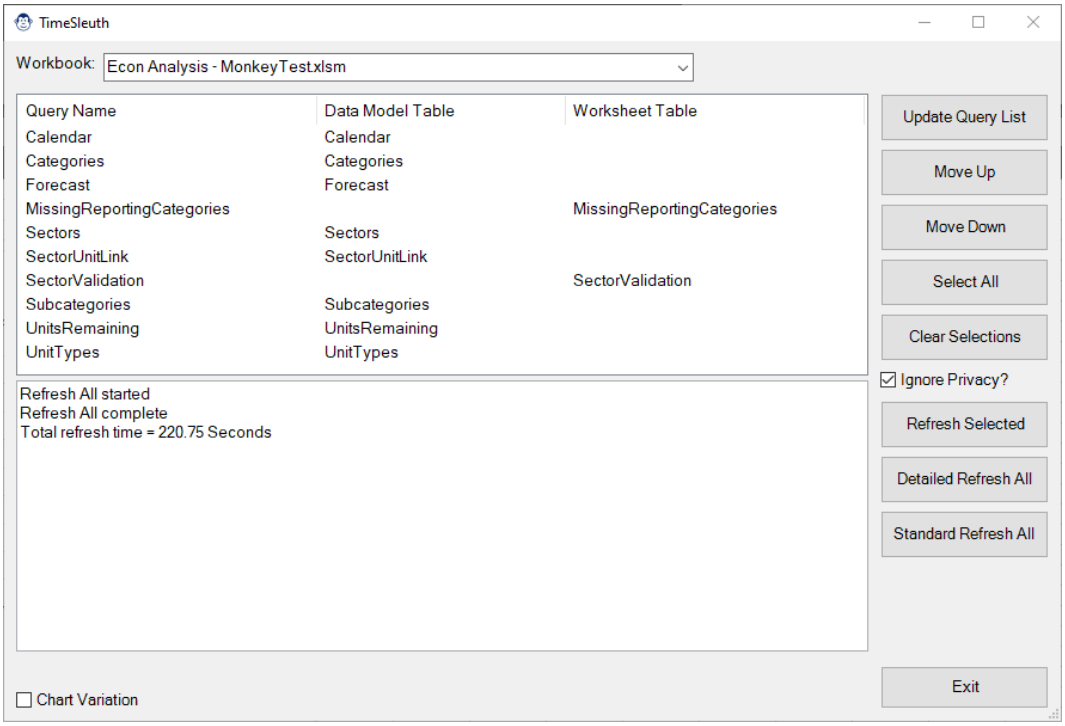


Figure 46 - TimeSleuth running a "Standard Refresh All"

As you can see, the feedback is relatively minor. It simply times the global load time of the queries.

7.4 Detailed Refresh All

The Detailed Refresh All button provides much more information than the Standard Refresh All and can be launched by simply clicking the "Detailed Refresh All" button.

The big difference between the Standard Refresh All and the Detailed Refresh All is that the tool refreshes every query individually (in the order they are shown in the top table). During the refresh, times are captured for the individual query, as well as the cumulative time, as shown in Figure 47:

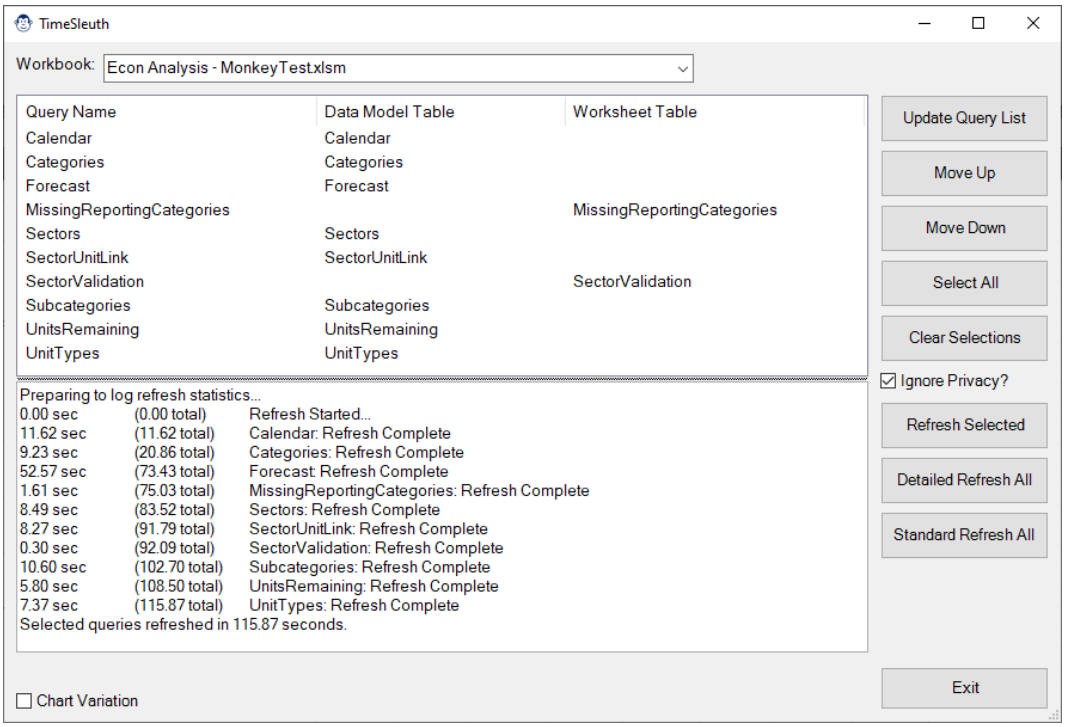


Figure 47 - QuerySleuth running a "Detailed Refresh All"

Queries are always refreshed in order from the first item in the table at the top of the form to the last. Should you wish to change the refresh order of the queries when timing, you can easily do this by:

- Selecting the Query in the top table
- Selecting Move Up or Move Down as required
- Running the Detailed Refresh All

With regards to performance timings, you may notice that a Detailed Refresh All is always slower than a Standard Refresh All. The reason for this is that the Standard Refresh All capability takes advantage of child node caching, loading the data once and re-using it for other queries later[4]. With the Detailed Refresh All feature, we are forcing each query to load individually to get an idea of the timing, so we do not take advantage of child node caching.

7.5 Refresh Selected

The real benefit of being able to time queries in Power Query is when you can make multiple versions of the query and see if one performs better than another. This is the main purpose of the Refresh Selected button.

7.5.1 Timing a Single Query

To time a single query, just select it in the query listing at the top of the form, then click "Refresh Selected".

[4] Excel 2019 and later only.

7.5.2 Timing Multiple Queries

To select multiple queries, you have a few options:

- Click the first query you want, hold down SHIFT, and click the last query you want. This will allow you to select a contiguous block of queries to be tested.
- Click the first query you want, hold down CTRL, and select other queries. This will allow you to select multiple queries that are not adjacent to each other.
- Click the Select All button to select all queries.
- Click Clear Selections to clear the selections and start over.

Once you have the appropriate queries selected, press "Refresh Selected" to kick off the query timing process.

7.6 Advanced Testing and Comparing Results

Due to the volatility of the Power Query engine, getting a true benchmark of query performance should be accomplished by running multiple tests and comparing results. For this reason, you'll find an option at the bottom of the form to "Chart Variation":

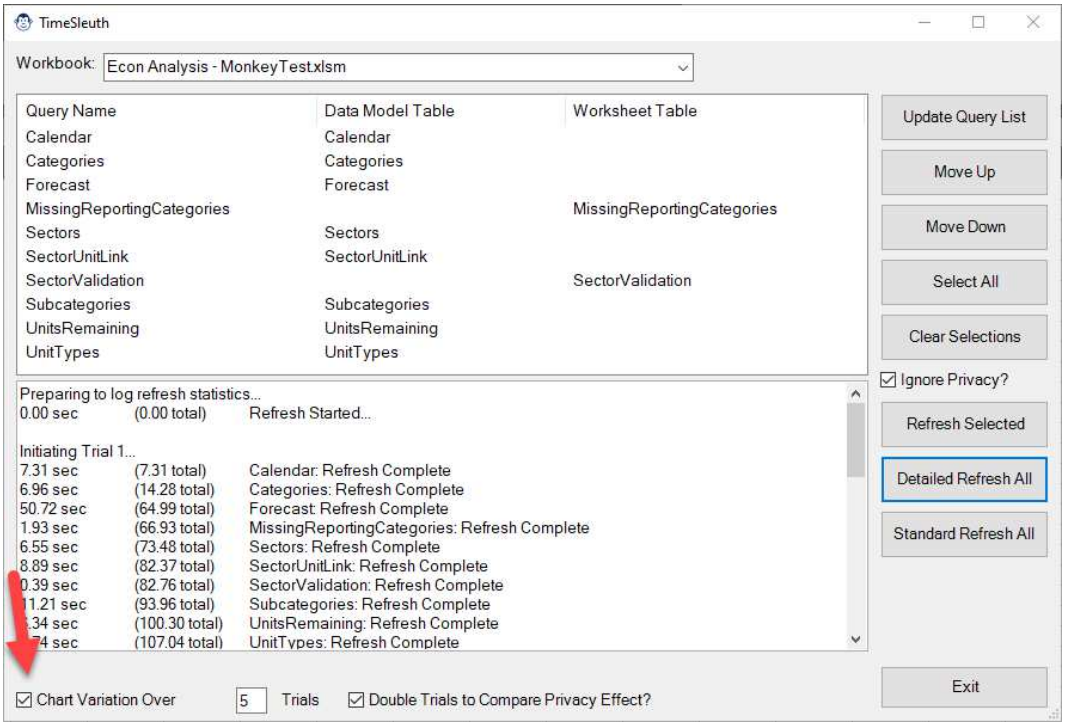


Figure 48 - Charting variation of multiple query refreshes

Once you have checked this box, you'll have the option to declare how many trials you'd like to chart, and even the option to double the run in order to create a comparison of refreshing with privacy on or off. Just be aware that depending on your query performance this can take a LONG time to complete, so you need to be patient!

7.6.1 Interpreting Variation Results

Once the timing runs have completed, the TimeSleuth form will close automatically, and present you with box and whisker chart. A sample chart of a model (which took over x minutes to refresh) is shown here:

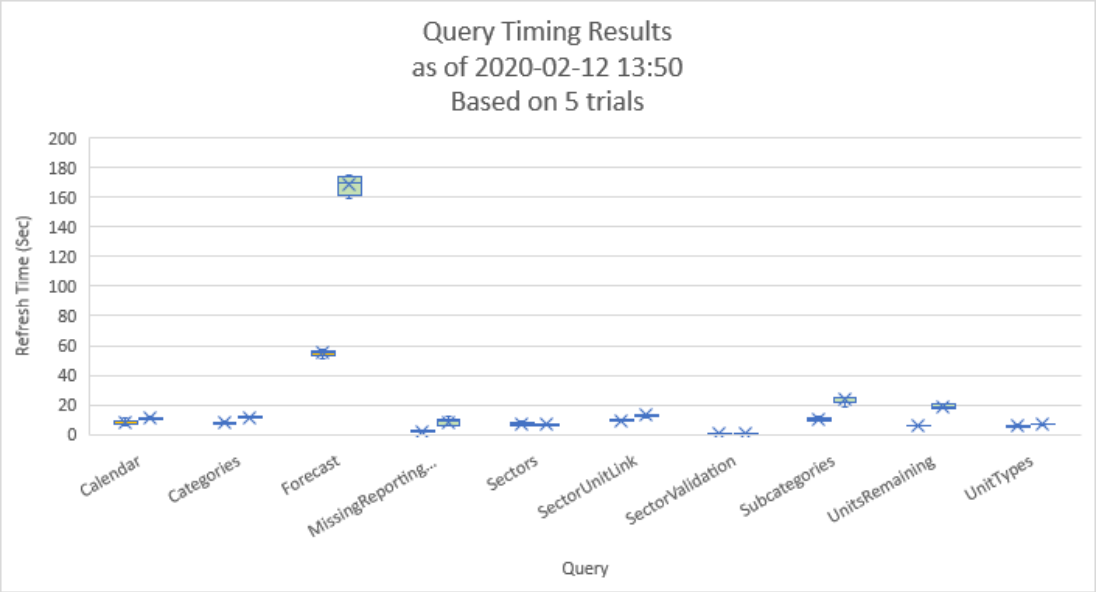


Figure 49 - Query Timing Results with Privacy comparison on

But what does that mean, and how can we slice it down?

The Box & Whisker chart provides a ton of information, but it can basically be summarized like this:

- The "x" is the mean (average) of the results
- The line in the middle is the median of the results (50% of values are greater or lower)
- The box contains all values less than the top 25% of time and greater than the bottom 25%
- The whiskers show the maximum and minimum values, excluding outliers
- Dots are used to represent outliers that fall way outside the range

So, what does that mean to you?

The smaller the box, the less variation in timing. The higher it is on the chart, the longer it takes to run. In other words, most of the queries seem to be quite fast, except for that Forecast query, which takes around 170 seconds to run.

For a great article on understanding how to read a Box & Whisker chart, see Nathan Yau's article on the subject at flowingdata.com.

7.6.2 Adding a Legend to the Chart

This is a bit of a nuisance, but unfortunately the object model we use to automate the chart creation has a bit of a hole in it and doesn't respect the command to add a legend to the chart. If you'd like to add a legend to show that the green values represent timings with Privacy=Off and the orange values represent timings with Privacy=On, here is what you need to do:

- Select the Chart
- Go to the Chart Design tab → Add Chart Element → Legend → Bottom

The result is the way we WISH we could present the chart to you by default!

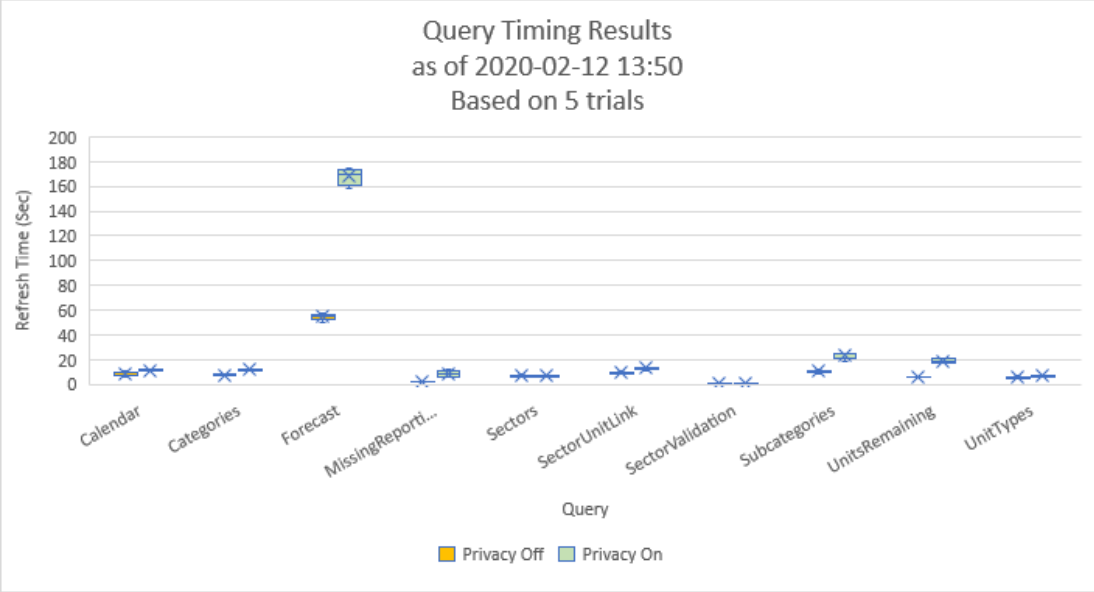


Figure 50- Query Timing Results with Privacy comparison on... and a legend!

7.6.3 Hiding one of the Chart Series

There is a lot of data on the chart shown in Figure 49, but you can reduce it to hide one of the series:

- To hide the "Privacy Off" series: Right click Column B's header → Hide
- To hide the "Privacy On" series: Right click Column C's header → Hide

Hiding Column C would result in the chart shown in Figure 51:

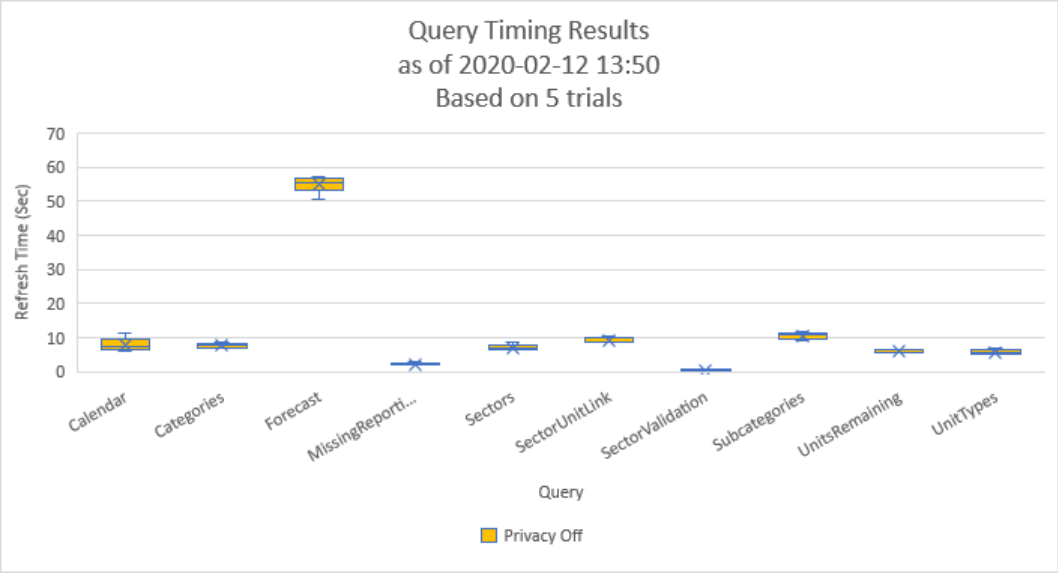


Figure 51 - Query Timing Results for only the Privacy Off (Ignored) results

To bring the series back, use the following steps:

- Select columns A:D
- Right click the column A header → Unhide

This should restore the chart to show both series of data.

Hiding Column B (the Privacy=Off series), would then result in the chart shown in Figure 52:

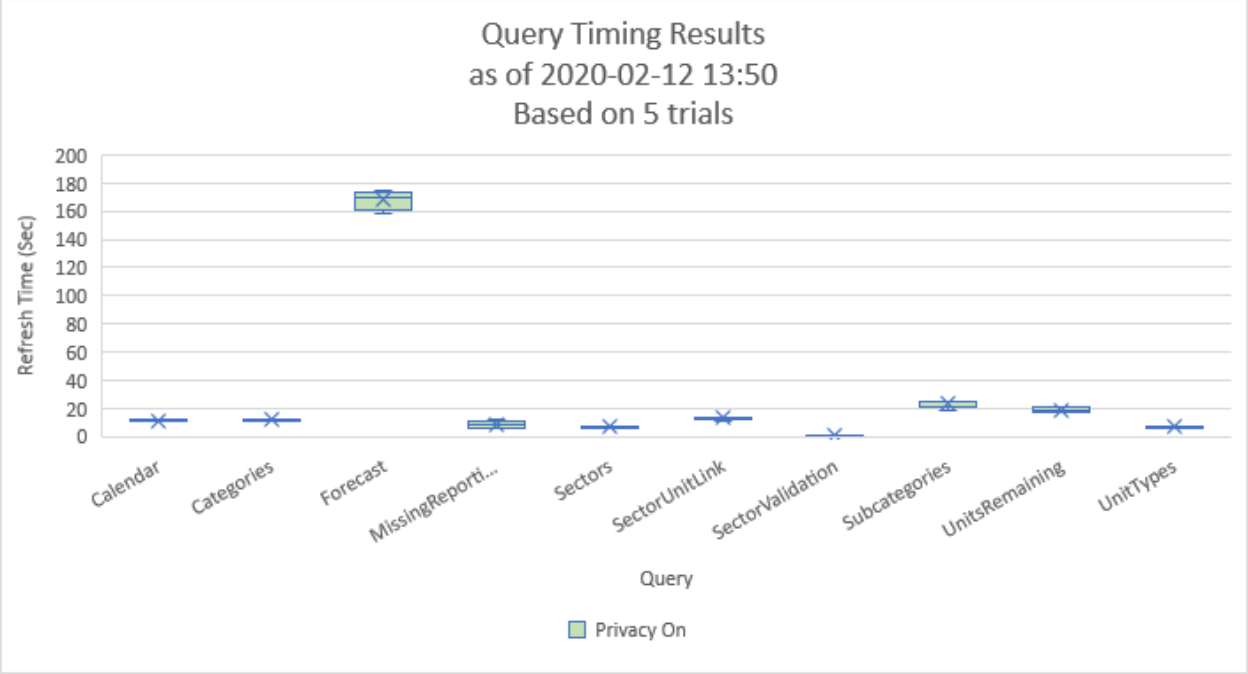


Figure 52 - Query Timing Results for only the Privacy On results

The key takeaways from these two charts are:

- Privacy seems to have a have a fairly significant effect.
- The Forecast query is the biggest issue in this model's refresh

7.6.4 Examining Specific Queries

While the global trend is interesting, what if you now want to drill in to compare the results of only the Forecast query? Let's take a look:

- Select columns A:D
- Right click the column A header → Unhide
- Click Forecast in the Slicer

The result should look as follows:

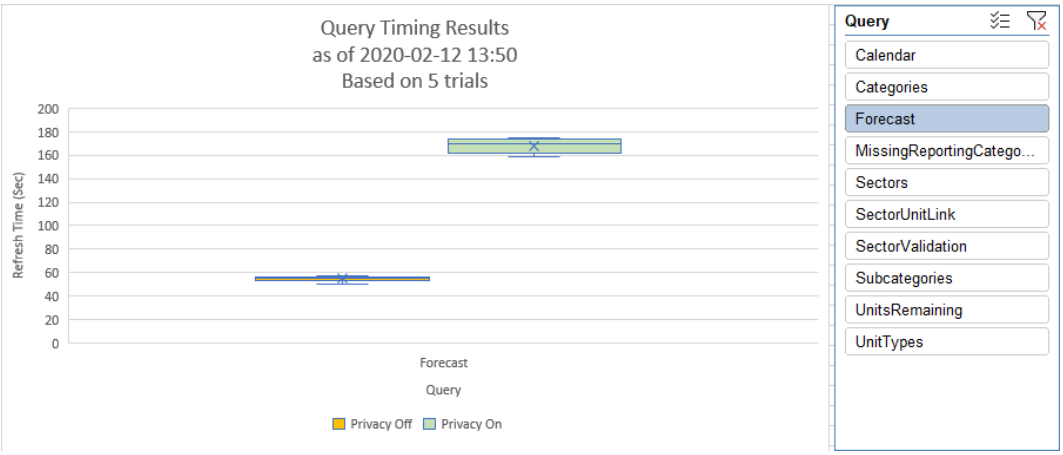


Figure 53 - Examining the results of the Forecast Query

The effect of Privacy now becomes very apparent. With Privacy Levels set to Ignore (off), the query consistently refreshes in the 58 second range. With Privacy on, it varies more, but takes between 160-175 seconds!

Other queries also appear to have similar behaviour:

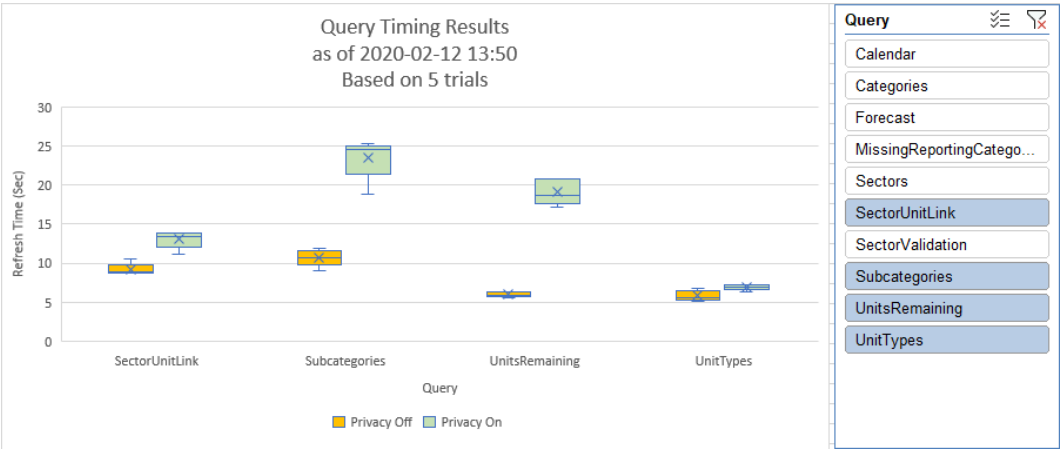


Figure 54 - Examining the results of Privacy Levels on various queries

While the differences may not always be massive, one truth becomes evident here: Privacy Levels definitely have an impact on your refresh times.

7.7 Other Features

The only other button that has not been discussed in this chapter is the “Update Query List” button. As this form is non-modal, it means that you can make changes to the workbook’s query chain while this form is open. Should you do so, you’ll need to click this button to refresh the list of the Power Queries in the form.

8 PivotSleuth

If you've worked with PivotTables for any amount of time, you know that there can be some challenges when trying to understand why a PivotTable is not delivering the expected results. Some of these challenges include:

- The original Pivot field has been renamed, making it difficult to identify the original column or measure name, as well as which table it may be hosted on.
- Identifying Slicers and Timelines attached to the PivotTable requires jumping to a different interface.
- Locating the source of the dreaded "Relationships May Be Needed" message is difficult without a sound understanding of the data model.
- Seeing problematic combinations of Fact and Dimension fields is not possible.

PivotSleuth is intended to be a one-stop shop for understanding everything that impacts your PivotTable. It is designed to help you quickly identify issues impacting a PivotTable and provide some context sensitive help around your model semantics.

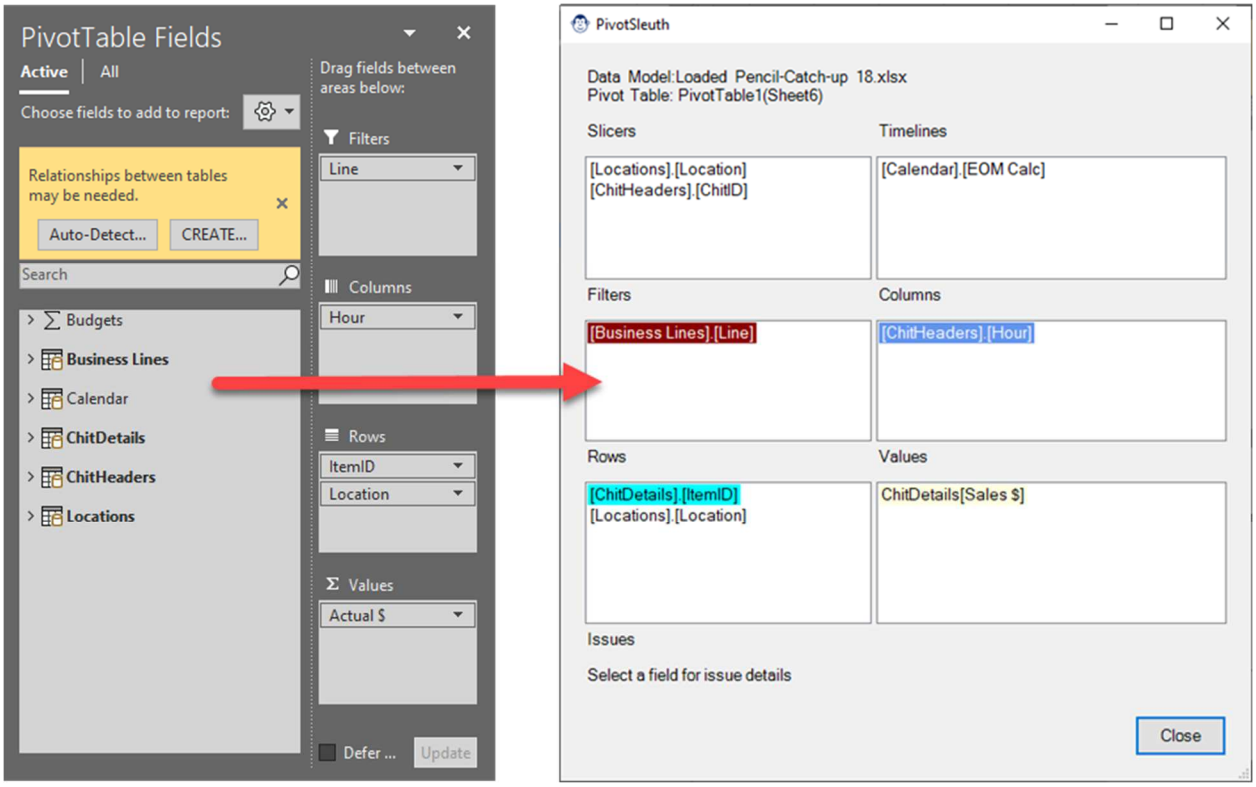


Figure 55 - This PivotTable appears to have some problems...

General features that you should be aware of when working with the PivotSleuth include:

- To use the PivotSleuth, you must select a cell inside the PivotTable before launching.
- Full filter context applicable to the PivotTable is provided, including a list of all slicers and timelines connected to the PivotTable, whether they are on the same worksheet or not.

- Selecting a slicer or timeline in PivotSleuth will active its parent worksheet and select the object (unless the sheet is hidden).
- All field names are presented in the fully qualified context related to the actual name from the data model.
- Hovering the mouse over any field will provide extra information related to that field.

8.1 Context Sensitive Help

PivotSleuth provides context sensitive help that differs between the various areas of the form, so let's explore each of the relevant areas:

8.1.1 Values Area

The important things to be aware of in the Values area is that PivotSleuth always displays the column name in its fully qualified data model syntax of TableName[ColumnName]. To see what the column is called in the PivotTable field list, you simply need to mouse over the name you're interested in.

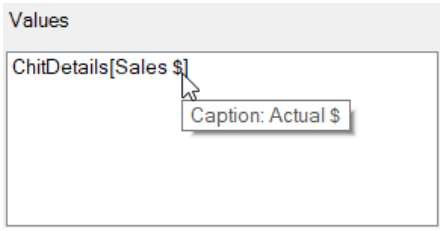


Figure 56 - The 'Actual $' showing on the PivotTable is really 'Sales $'

Since renaming a field on the PivotTable also renames it in the field list, it has always been challenging to figure out the original measure or column name being used. Now it's easy!

8.1.2 Slicers and Timelines

You could certainly acquire the information provided by the Slicer and Timeline boxes by selecting your PivotTable then going to PivotTable Analyze → Filter Connections. It provides the slicer captions and data model field names, which is great. But it's still a click away and littered with all the other slicers and timelines that are NOT connected to your PivotTable.

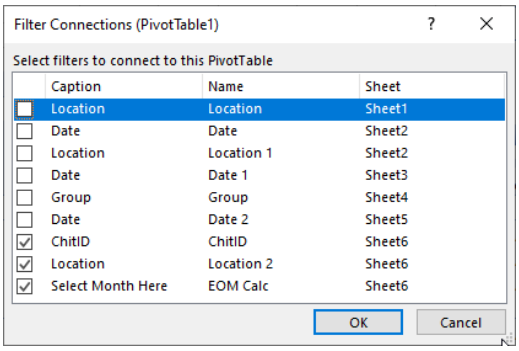


Figure 57 - The first six items in this list are not relevant to this PivotTable...

PivotSleuth lists only the objects already bound to your PivotTable and displays the fully qualified data model context first. It does allow you to see the Slicer and Timeline captions by hovering over the item in question:

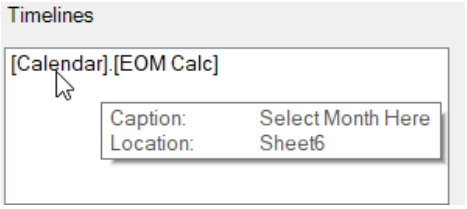


Figure 58 - The EOM Calc field was renamed to 'Select Month Here' on the Timeline

Don't forget: should you wish to select the Slicer or Timeline for any reason, you just need to click it to be taken right to it.

8.1.3 Coloured Messages

Each of the different colours used in the PivotSleuth carries a meaning, as well as a message on the root cause or potential issue you may end up facing over the long term. Let's look at each in turn.

8.1.3.1 Red Fields

A red field indicates that the there is no relationship between this field and at least one of the fields used in the values area of the Pivot. These fields are the root cause of the dreaded "Relationships May Be Needed" message that shows in the PivotTable fields list.

When you see a red field, you can hover the mouse over it and you'll receive contextual information to tell you exactly what is going on, as shown in Figure 59:

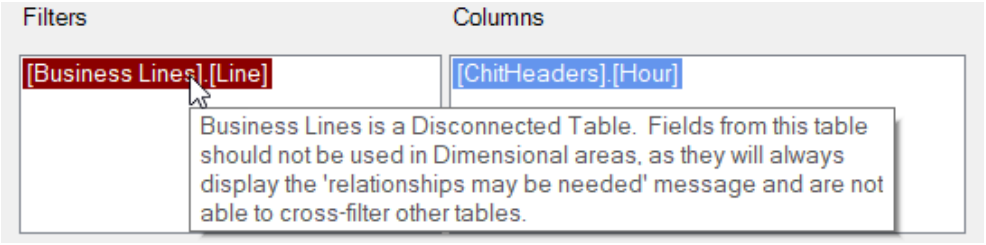


Figure 59 - A disconnected table cannot filter the items in the PivotTable Values area

In this case we are looking at a disconnected table, as indicated in the hover message. Since disconnected tables have no relationships in the model, they cannot propagate a filter to the other tables, and will always display a red background colour as shown above.

8.1.3.2 Cyan Fields

Cyan fields indicate that you have used a field from a Fact table in a Dimensional field:

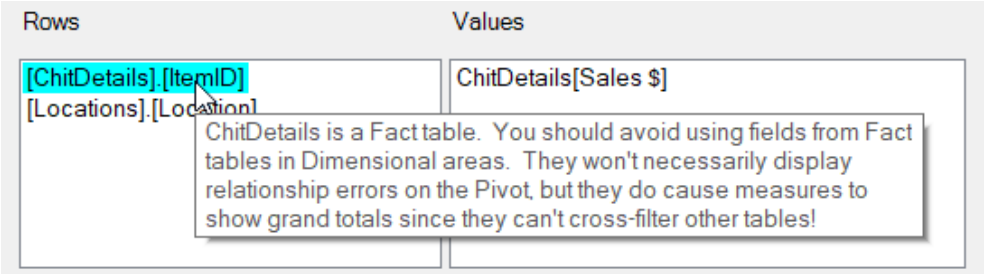


Figure 60 - A Fact table field has been used in a Dimensional area

While not recommended, in a table with a single Fact table this *can* work. But as soon as you add a field from another Fact table, this will lead to cross filtering errors, as you'll see later.

8.1.3.3 Blue Fields

The darker blue highlights fields which are tagged as Foreign Keys. This is also not recommended, as the preferred field to use is the relationship's Primary Key because it allows filters to propagate correctly through the model.

While hovering over each field shows the tooltip text, these tips do disappear after a few seconds. Rather than keep attempting to hover again, you can also click the field to display the message at the bottom of the form:

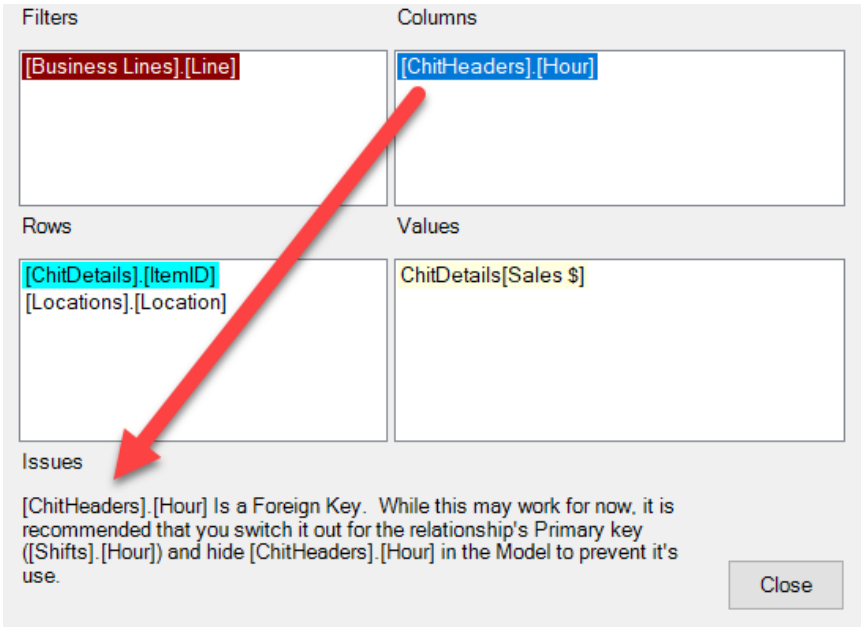


Figure 61 - A Foreign Key is being used instead of the relationship's Primary Key...

8.1.3.4 Yellow Fields

Like red fields, yellow fields indicate that there is a cross-filtering problem, but you will only see yellow fields in the Values area. They tell you exactly which fields cannot filter the specific measure:

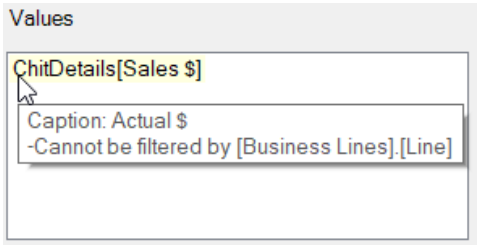


Figure 62 - The Business Lines field cannot filter the values for the Sales $ measure

8.1.3.5 Italicized Measures

Something else you may see is certain measures listed in Italics, where others are not, as you can see below in Figure 63. Italics indicate Implicit measures, where regular fonts indicate Explicit measures. Even renamed, Implicit measures can now be found easily.

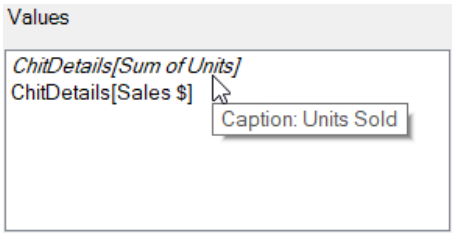


Figure 63 - We have a mixture of Implicit and Explicit measures

8.2 Why Are All Fields Shown in Red?

One thing you'll run into at some point is where all the highlighting in the form is red, instead of the various colours you saw earlier. The only different between this PivotTable and the ones displayed earlier in this chapter is that a measure has been added from a different Fact table. What the form is now telling you is that all the potential problems identified earlier are actively causing filtering challenges as they cannot cross filter the new Fact:

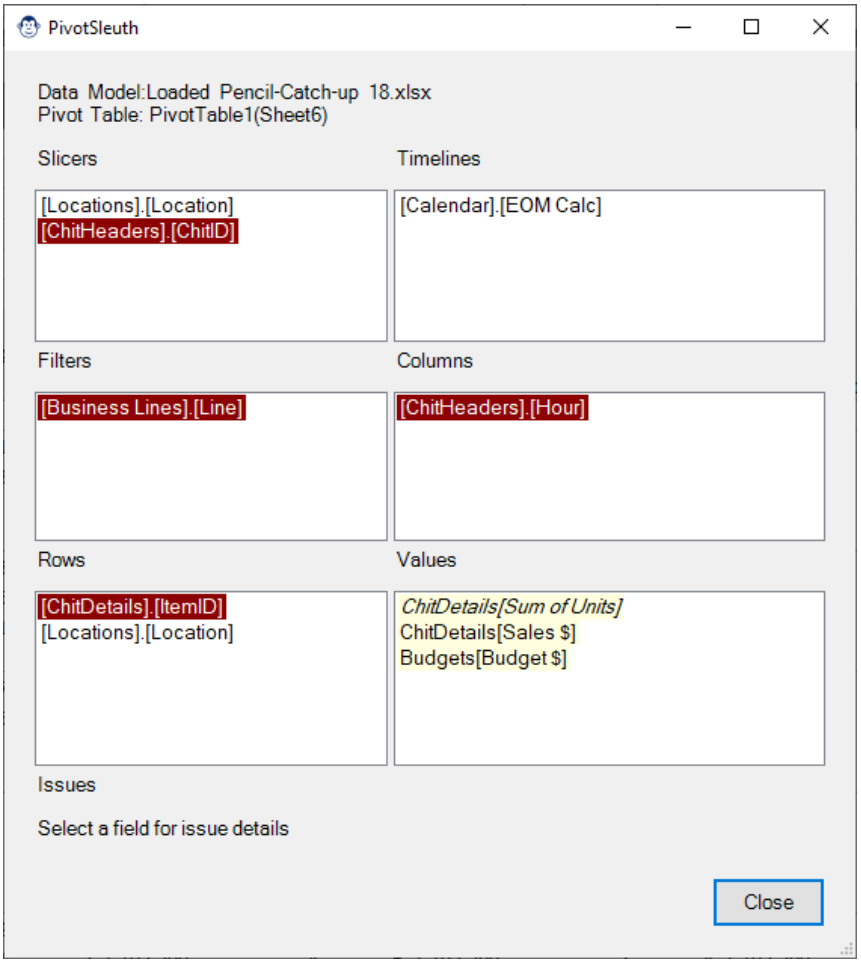


Figure 64 - A new Fact has been added and things got nasty!

Let's look at each of the messages, starting from top left working across to bottom right, to see what we can learn.

This field has no relationship path to and cannot filter Budgets[Budget $]
Caption: ChitID
Location: Sheet6

Figure 65 - Tooltip text for the ChitHeader[ChitID] Slicer field

As you can see above, the ChitID field is plainly problematic, as it can't filter Budgets[Budget $].

The Business Lines field was previously coloured red and remains so. By hovering over it, we can see why:

Business Lines is a Disconnected Table. Fields from this table should not be used in Dimensional areas, as they will always display the 'relationships may be needed' message and are not able to cross-filter other tables.

Figure 66 - We still have a disconnected table on our PivotTable

Originally shown in blue, that Foreign Key of ChitHeaders[Hour] is now causing a problem:

This field has no relationship path to and cannot filter Budgets[Budget $] and is thefore showing grand totals for that field. Additionally, [ChitHeaders].[Hour] Is a Foreign Key. While this may work for now, it is recommended that you switch it out for the relationship's Primary key ([Shifts].[Hour]) and hide ([ChitHeaders].[Hour]) in the Model to prevent it's use.

Figure 67 - We have a cross filtering problem in addition to using inappropriate fields

It's important to realize that you've been presented with two problems here. As you can see below in Figure 68, the lack of a relationship between the ChitHeaders and Budgets table is the cause of the cross-filtering issue. Replacing ChitHeaders[Hour] with Shifts[Hour] will not fix that problem. On the other hand, if ChitHeaders[Date] was in use on the Pivot, replacing it with Calendar[Date] *would* fix the issue as there is a relationship path between Calendar and Budgets.

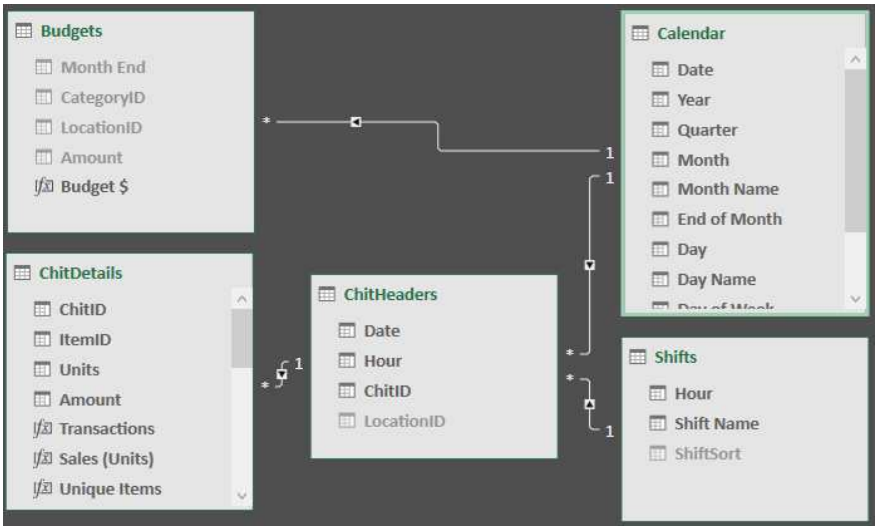


Figure 68 - An excerpt of the data model in question

Next is the ChitDetails[ItemID] in the Rows area. Once again, it displays two issues, as you can see below in Figure 69. As before, the main issue is that there is no relationship path between the ChitDetails table and the Budgets table, as filters cannot propagate against the arrow direction. In addition, this Dimensional field still holds a value from a Fact table:

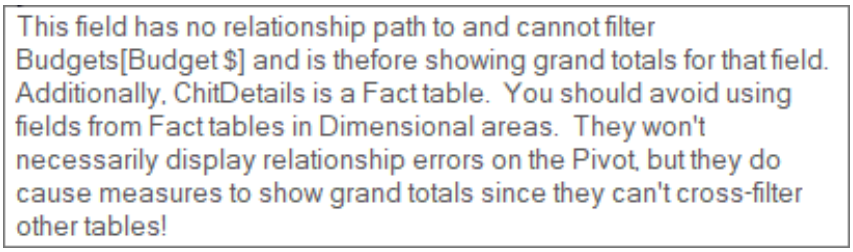


Figure 69 - The ChitDetails[ChitID] field displays two issues

Finally, the Values area also shows all fields in yellow. Looking at each of these tooltip messages (or selecting the fields), we can see how each is interacting with the rest of the field configurations:

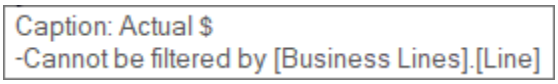


Figure 70 - One of the fields on the PivotTable cannot filter this measure

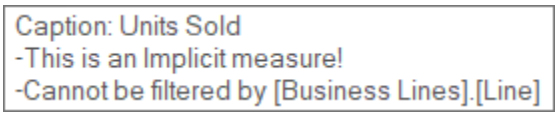


Figure 71 - Not only is this an Implicit measure, but one of the PivotTable fields cannot filter it

Figure 72 - There are multiple fields on the PivotTable that cannot filter Budget $!

8.3 Excel Doesn't Show Errors, so Why Does PivotSleuth?

Have a look at the following PivotTable and PivotSleuth configurations:

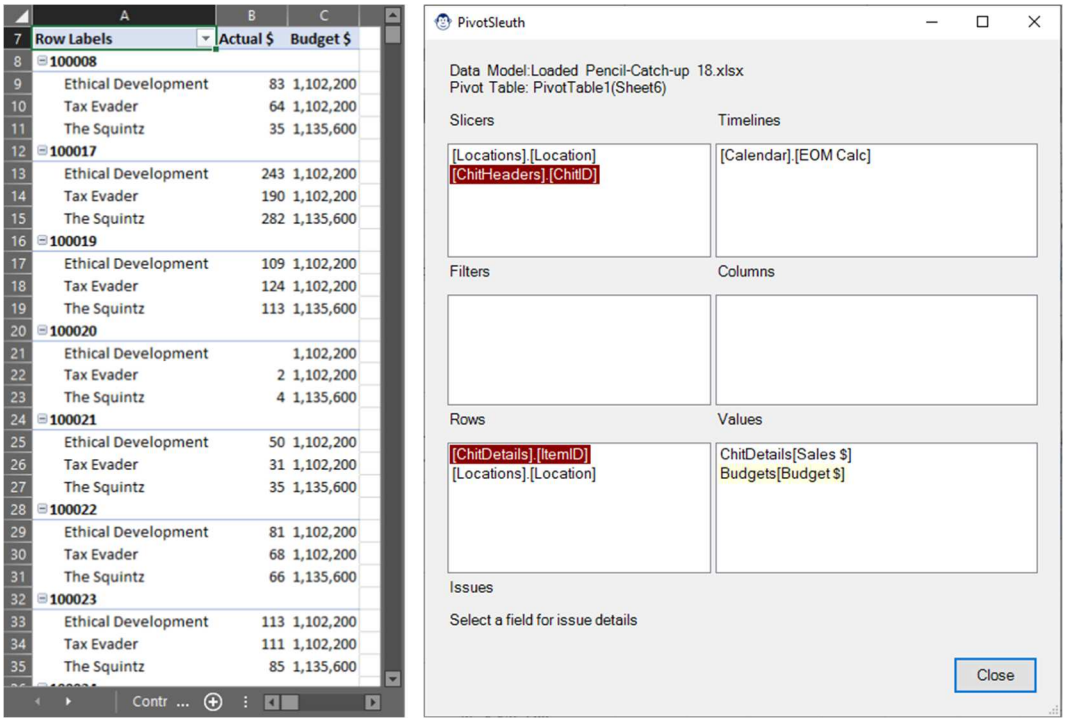


Figure 73 - A PivotTable and its PivotSleuth configuration

The important thing to know about this combination is that the PivotTable field list does not show the "Relationships May Be Needed" error. So why are we getting red fields in the PivotSleuth?

Take a quick look at the budget by Product and Location. Notice how they are the same for every product. Take a look at the tooltip text on the ChitDetails[ItemID] field:

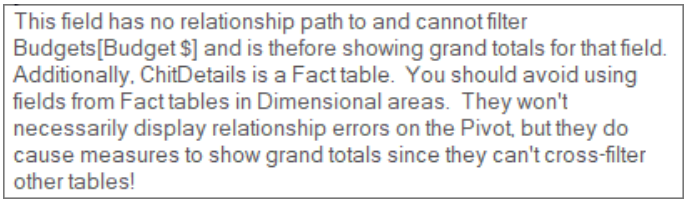


Figure 74 - The ChitDetails[ChitID] field displays two issues

And the Budget $ measure in the Values area:

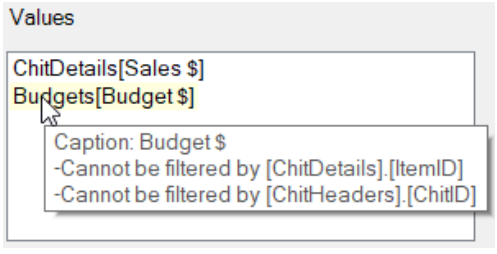


Figure 75 - There are actually two fields leading to cross filtering problems!

Now you know why you have grand totals on every row, but also where the issue is occurring and even how to fix it!

8.4 Why is PivotSleuth reporting errors for my Measures table?

There is a school of thought that says that Measures should be stored on a separate table. So why, when you do this, does Monkey Tools report that your PivotTable is mis-configured? The short answer is – if you set up your model correctly – it won't.

8.4.1 Diagnosing (and fixing) an improperly configured "Measure" table

Have a look at the following (perfectly valid) model:

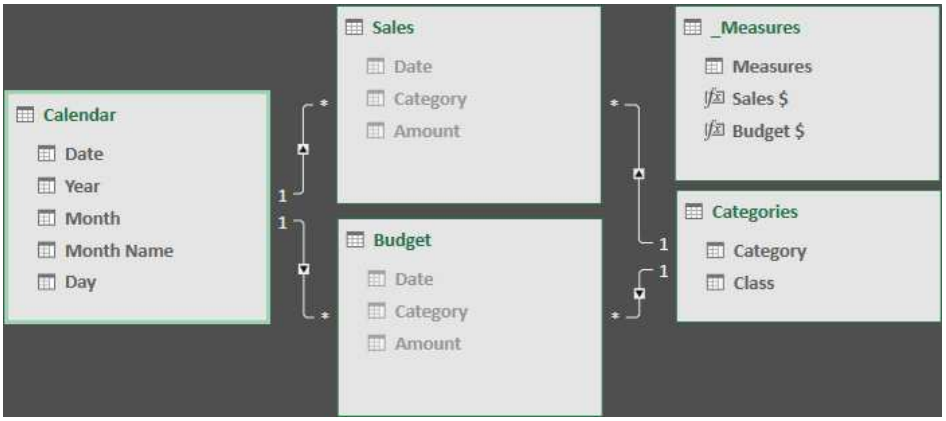


Figure 76 - A perfectly valid data model with a disconnected _Measures table

And yet, whenever you build a PivotTable using these measures, you get prompted with the dreaded "Relationships between tables may be needed" message. So, what does PivotSleuth say about it?

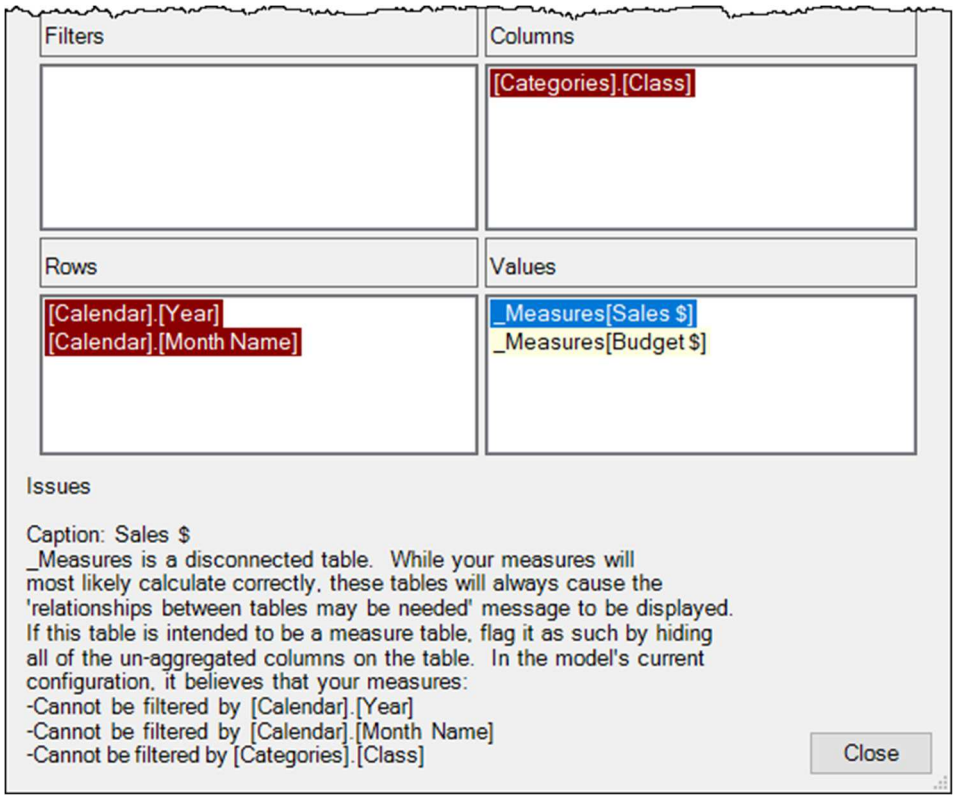


Figure 77 - PivotSleuth has a lot to say about mis-configured Measure tables!

The issue with the configuration shown above can be seen if we take a close look at the PivotTable field list:

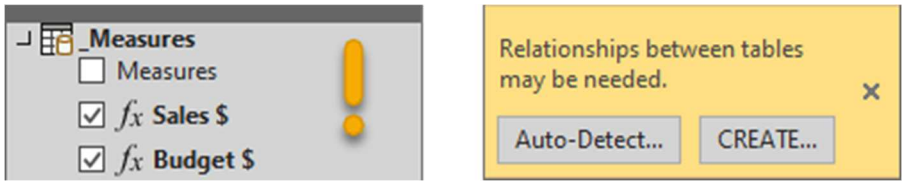


Figure 78 - This "Measure" table is acting as a Dimension table!

All the issues and red fields shown in PivotSleuth are caused by that visible “Measures” column. And the fix is super easy: Hide the un-aggregated “Measures” column on the _Measures table!

To do this:

- Go to Manage Data Model (on the Monkey Tools or Power Pivot tabs)
- Select the new measures table
- Right click the “Measures” column and choose “Hide from Client Tools”

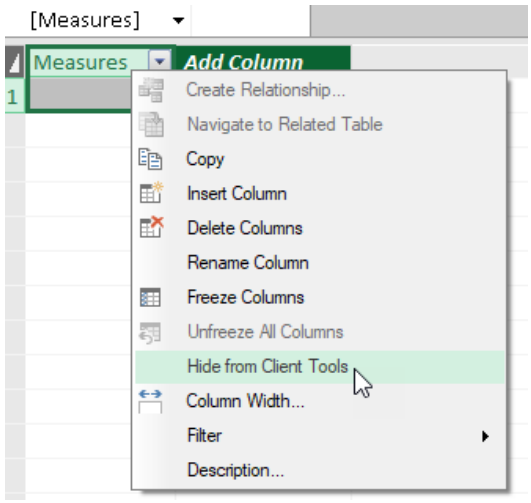


Figure 79 - Hiding Power Pivot columns

As soon as you make this change, you’ll notice that the table gets a new icon in the PivotTable field list, and you’ll never see that irritating yellow message again (at least not because of measures from this table!)

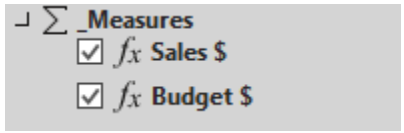


Figure 80 – An official Measures table

8.4.2 Should Measures be stored on a separate table?

The practice of storing measures on another table as a recommended practice was born out of Power Pivot instability, back when things crashed a ton. Sometimes the fix would require removing the table from the data model and re-creating it, at which point you would lose any measures or relationships

built on those tables. It was frustrating and annoying, and led people to keeping their measures into a separate table to protect themselves from having to do that work. The challenge, however, was that it caused a "Relationships May be Needed" message every single time you used a measure. And there was no way (at the time) to make that go away.

Since 2016, Microsoft focused on fixing bugs related to Power Pivot, with many of them making their way back into the Excel 2016 product, even if they were fixed after 2019 was released. While they are certainly not all gone, it is unusual to see issues that force the need for tables to be deleted and rebuilt now. To us, this reason for separating your DAX has basically become a non-issue.

Some people also argue that this gives you a central place to go to get your measures. We would argue that the list can become overwhelming when all your measures are in one folder without any categorization. (Unlike Power BI, we have no ability in Excel to group measures into folders.)

We far prefer to put our measures on the appropriate tables, then hide all the unaggregated columns on the table. This offers three benefits:

1. It groups the measures by table, making them easier to find, as they act like folders for the measures. (Yes, there is a search function, but we do not feel that should be your first stop!)
2. It means the "Relationships May be Needed" warning only shows up when a measure cannot be cross filtered by a natural relationship in the model.
3. It changes the icon of the table to the Σ icon, which is synonymous with measures.

Ultimately which method is best? The answer is solely one of personal preference.

9 DAXSleuth

Monkey Tools' DAXSleuth is loaded with features to make it easier for you to understand your DAX measures. You can see a full dependency tree for measures and calculated columns, located where measures are being used in your workbook. You are also able to view and edit your measures.

9.1 View of the DAXSleuth Window

A view of the DAXSleuth being used on a real model is shown here:

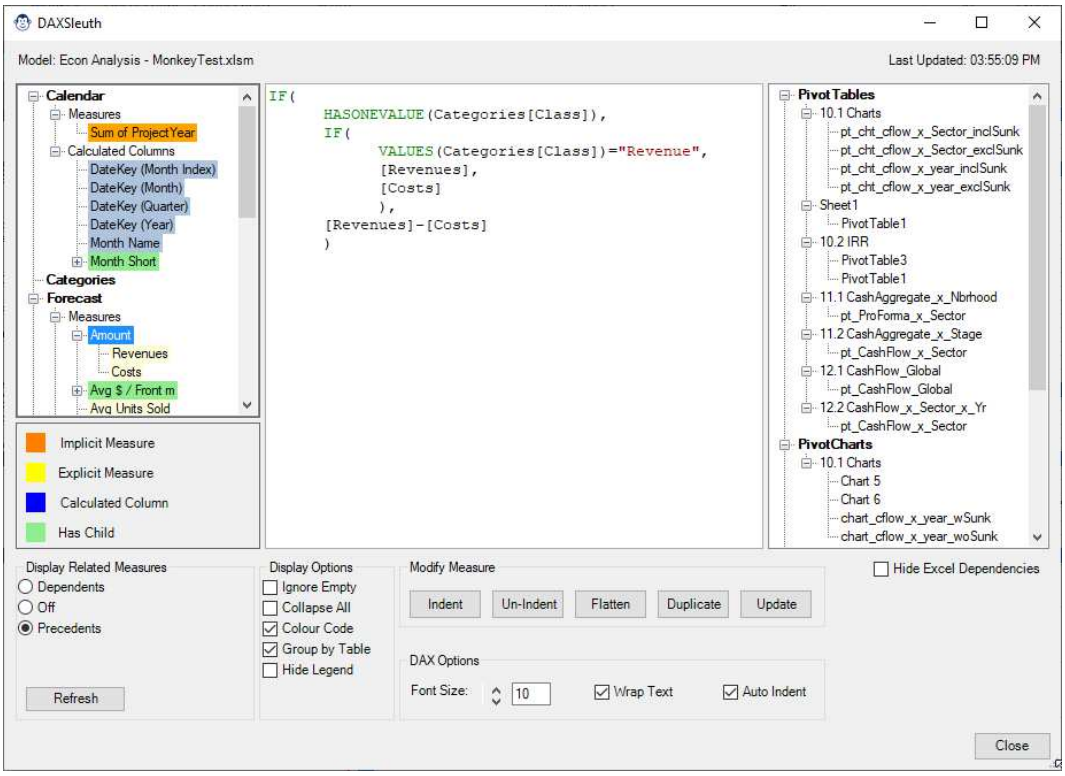


Figure 81 - The DAXSleuth in action

Some of the key things to be aware of in this window:

- **Model:** This lists the model name, allowing you to see which file DAXSleuth is currently analyzing.
- **Last Updated:** This lists the time the DAXSleuth last queried Excel or Power BI to get the model details.
- **The Measure Navigator:** Shown at left, the navigator lists all measures and calculated columns with their dependencies or precedents as determined by the selection in "Display Related Measures" at the bottom of the form. (The default is to show Precedents.)
- **The DAX Expression Window:** This is the large area on the middle of the form. It is fully editable and shows you the DAX expression used to calculate the selected measure or calculated column.
- **The Excel Dependencies window:** Shown in the right side of the form, this area displays all the Excel objects that use the currently selected measure or calculated column.

- **The Options Area:** At the bottom of the form are a variety of configurable options to help you get the most out of the form, each of which is discussed below.

9.2 Controlling the Measure Navigator

The purpose of the Measure Navigator is to show how individual measures are related to each other. There are three options for this area of the form, each of which are controlled via the radio buttons in the "Display Related Measures" section of the Options area (at the bottom left of the form). These options work as follows:

9.2.1 Precedents

Choosing this setting means that Monkey Tools will scan the model and show all precedents that feed the selected measure or calculated column. This view is very useful for figuring out the measure chain and how deeply nested your measure chain actually is.

9.2.2 Dependents

The Dependents view shows all measures that depend on the selected measure. This can be useful for figuring out what measures you may affect if you change a given measure or calculated column.

9.2.3 Off

Setting the option to "Off" will still list each measure and calculated column in the Measure Navigator but will not show +/- buttons for dependent or precedent measure or calculated columns.

9.2.4 The Refresh Button

This section of the form also contains a Refresh button. As DAXSleuth is non-model (i.e. Monkey Tools will allow you to interact with Excel), it means that you could potentially change measures in Excel while the DAXSleuth is open. To bring DAXSleuth back in sync with your file, you may want to refresh the Power Measure model, which can be done by clicking this button.

9.2.5 Hidden Features

One thing that isn't readily apparent from looking at the form is that you can expand an entire Navigator node's precedents or dependents by double clicking on that node. It's just a little feature that makes it super easy to drill into the measure you want to examine!

9.3 Working with the DAX Expression Window

By default, we indent all DAX measures in the DAX Expression Window, whether they are indented in the actual model itself or not. (Who doesn't want indented DAX, right?) The window is completely editable, so you can tweak your formulae at any time, and you'll find a row of buttons below it that you could find useful. Let's look at each of those now:

9.3.1 Indent

If you are not using the Auto Indent feature, or if you make modifications to a measure and wish to re-ident it, simply click the Indent button in order to do so. While this doesn't do a true syntax check to ensure your measure is valid, it does very much help if you just need to check that your parenthesis "balance" before committing it.

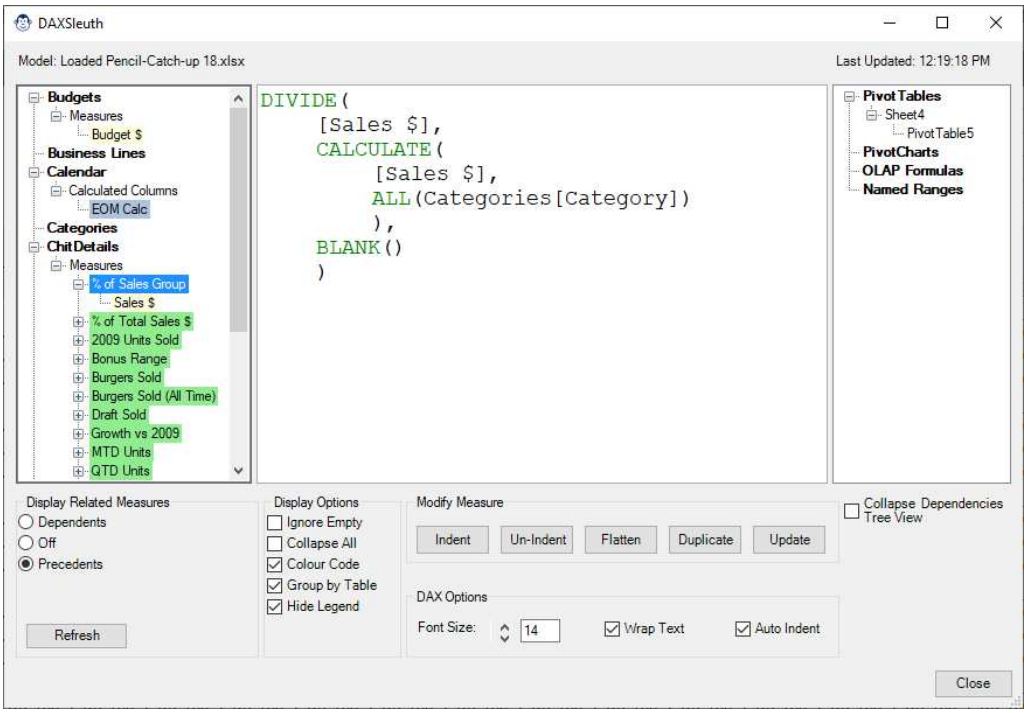


Figure 82 - DAXSleuth showing an indented measure

9.3.2 Un-Indent

This button will strip all indentation from the DAX Expression Window, leaving you with an un-indented measure or calculated column.

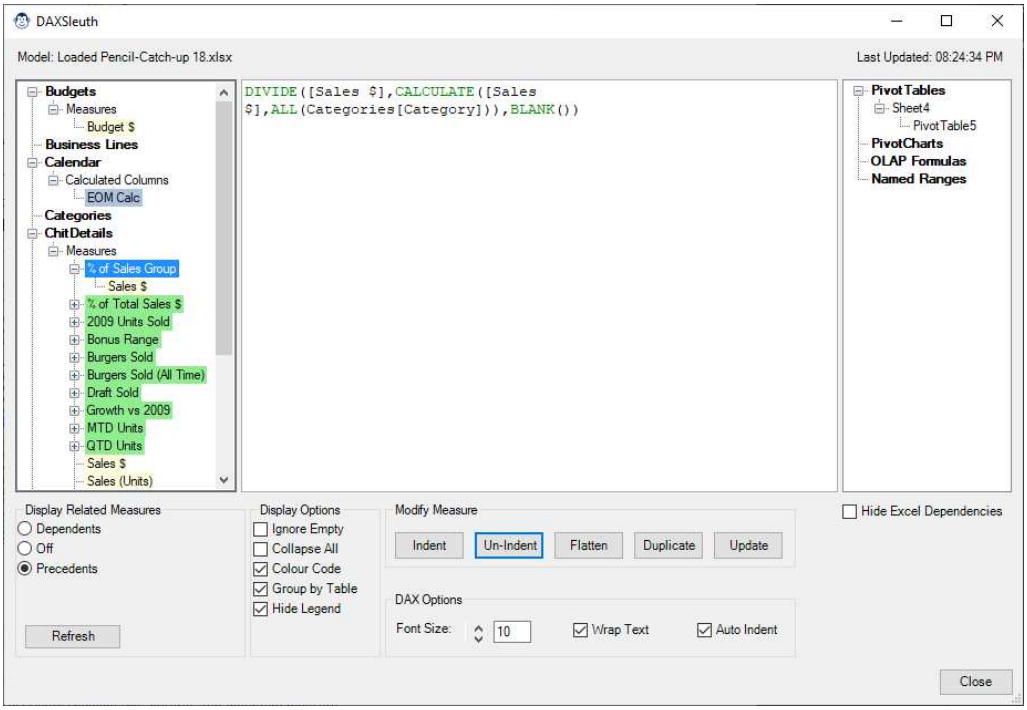


Figure 83 - DAXSleuth showing the "natural" (un-indented) measure

9.3.3 Flatten

The purpose of the Flatten button is to replace any nested DAX expression (measure or calculated column) in the current formula with the contents of the child expression. When flattening measures, you'll also notice that each child measure's signature gets wrapped in a CALCULATE() statement, in order to ensure proper context transition:

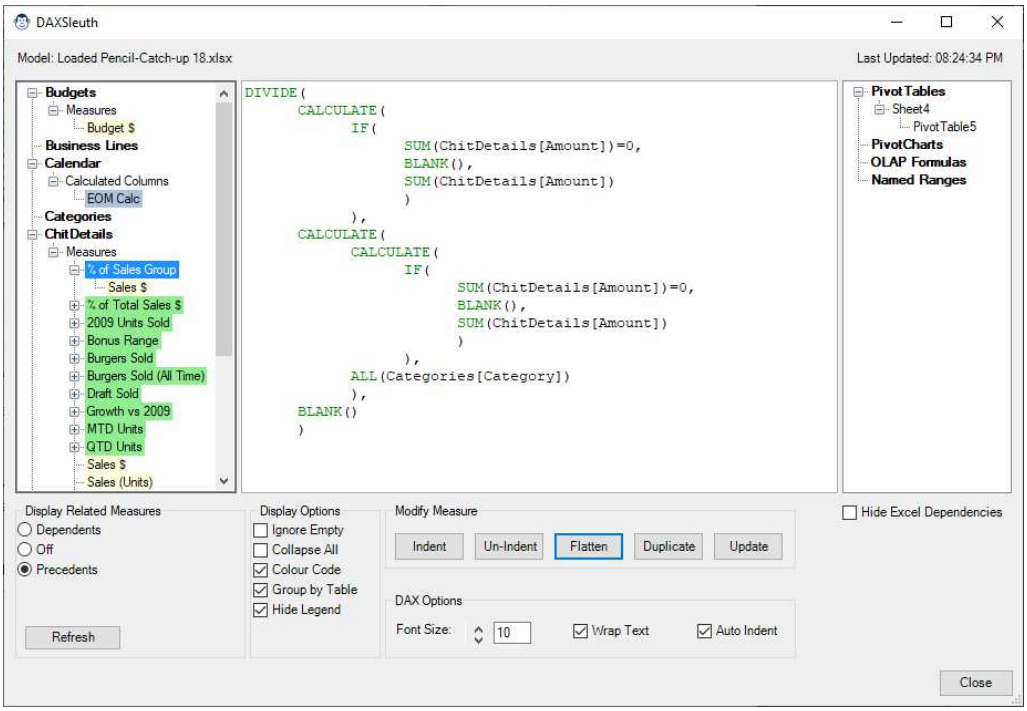


Figure 84 - DAXSleuth showing a "flattened" measure

9.3.4 Duplicate

One feature that is missing in Excel's Power Pivot management window is the ability to easily duplicate a measure. Yes, you can edit one, copy the formula, close it and create a new measure. And yes, you've got the formula, which is the guts of the measure, but what about the formatting? You still must tell it which table to store the measure on as well.

Monkey Tools' Duplicate feature assumes that you really want a duplicate of everything, including the host table, descriptions, DAX expression, and default formatting. When you click the Duplicate button, the currently selected measure will prompt you for a new name, as shown here:

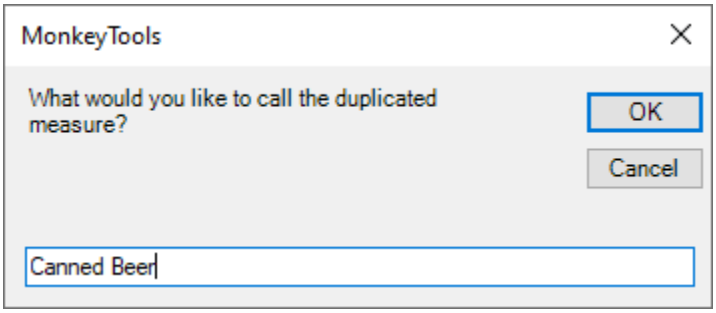


Figure 85 - What would you like to call the new measure?

Upon clicking OK, the measure will be created and added to the data model under this new name. The only steps left are to edit the DAX signature, and then click the Update button to commit the changes as shown below:

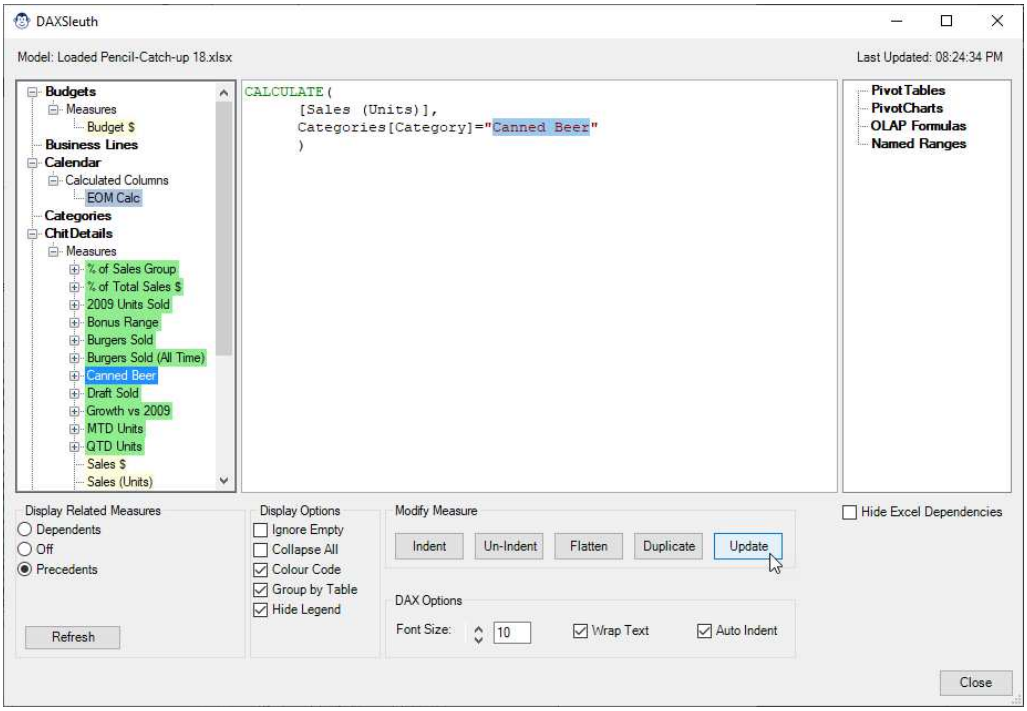


Figure 86 - Modifying and updating the duplicated measure

9.3.5 Update

As you might expect, this button writes the visible DAX expression to the model for the selected measure. It's useful for committing the indented DAX to the formula, or for committing to any tweaks you've made to the DAX expression.

NOTE: Due to a shortcoming in the Excel Object Model, we have no ability to create or modify Calculated Columns via code.

9.4 Working with the Excel Dependencies Pane

The Excel Dependencies Pane shows you all the objects in Excel which rely on the selected measure or calculated column. Just select the item, and it will even take you right to it so that you can see exactly what the usage context is!

Should you find that this pane is in the way, just check the box below it to Hide Excel Dependencies.

NOTE: This feature is obviously not available for Power BI Desktop models, as they don't have any Excel-related content.

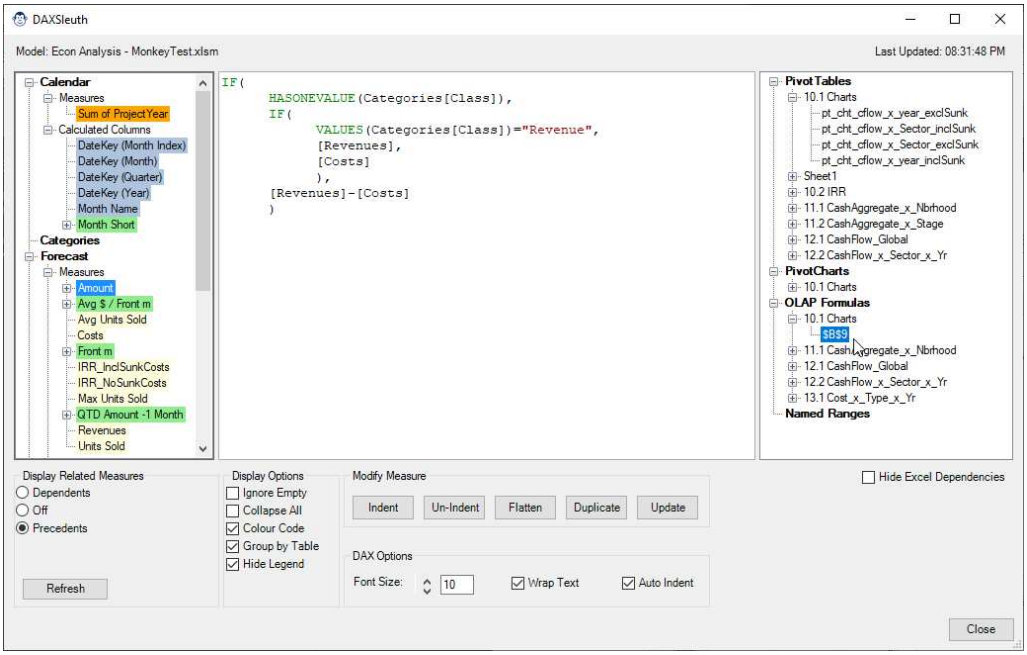


Figure 87 - Identifying where measures have been used in your workbook

9.5 Understanding the DAXSleuth Options

9.5.1 Display Options

The Display options control how the Measure Navigator displays the measure view. It is also worth noting that Monkey Tools saves your most recent selection as a default, so it preserves your preferred view every time you open the form.

9.5.1.1 Ignore Empty

Selecting the Ignore Empty option will suppress tables that do not contain any measures or calculated columns. This view allows you to focus on the dependency tree without irrelevant lines taking up extra space.

9.5.1.2 Collapse All

Unchecking the Collapse All checkbox will collapse all queries back to the main heading levels as shown here:

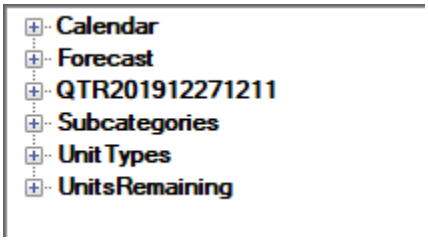


Figure 88 - Leveraging the Collapse All checkbox to collapse all tables

The Measure Navigator nodes can still be expanded by clicking on the + button.

9.5.1.3 Colour Code

By default, the measures and calculated columns are all colour coded as follows:

- Yellow: Explicit measures with no children (precedents or dependents)
- Green: Explicit measures with children (precedents or dependents as selected)
- Blue: Calculated Columns
- Orange: Implicit measures

If you find this view too noisy, you can mute the colours by unchecking the “Colour Code” checkbox.

9.5.1.4 Group by Table

By default, the Measure Navigator groups measures by table first, then by measure and calculated columns within each table. Un-checking this box will switch the order of the grouping so that measures are grouped first, separated by table within, and then calculated columns are grouped and separated by their tables:

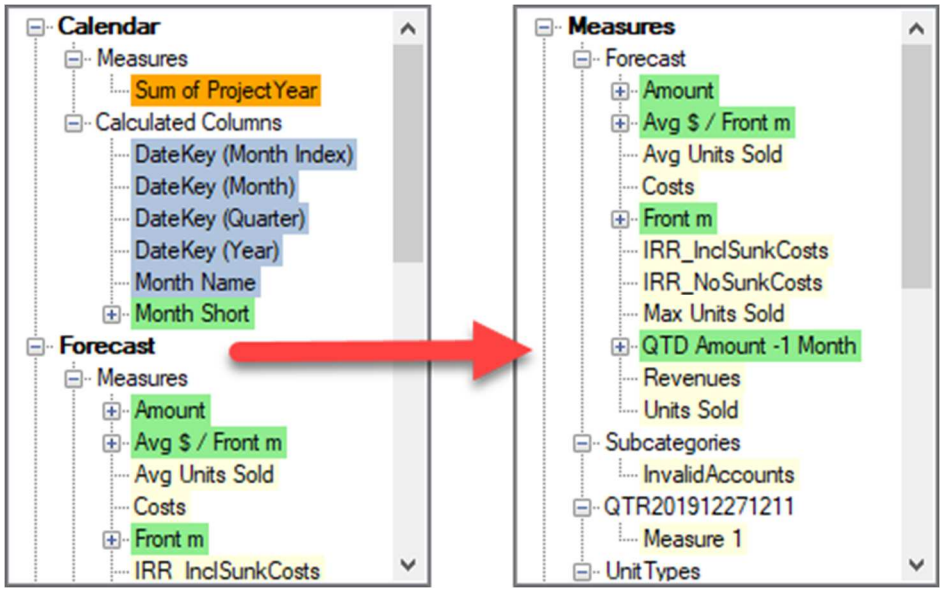


Figure 89 - Group by Table (left) vs Grouping by Expression Type (right)

9.5.1.5 Hide Legend

While the legend is useful to understanding what types of elements you are reviewing, it does take up a lot of space. Once you understand what the colours mean, it’s helpful to suppress the legend, which you can do by unchecking the box. We also save this selection as your default, so you won’t have to un-check the box every time you load DAXSleuth.

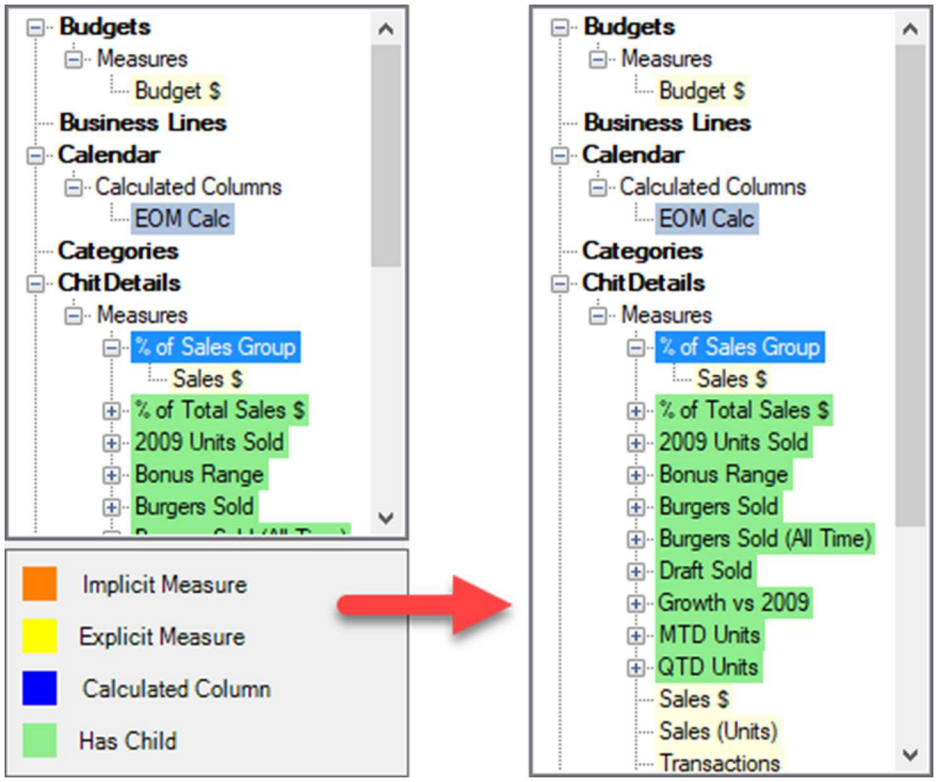


Figure 90 - DAXSleuth with legend showing (left) vs using the "Hide Legend" setting

9.5.2 DAX Options

9.5.2.1 Font Size

The Font Size controls allow you to scale the font size in the DAX Expression Window. You can manually enter your desired font size in the box or use the arrow keys to scroll it up and down to find the size you prefer.

9.5.2.2 Wrap Text

The Wrap Text feature is on by default and will wrap text onto the next line if the line width is too large to fit in the DAX Expression Window. (To be fair, this rarely happens when indentation features are active.) If this checkbox is not selected, and your line is too long to be shown in the DAX expression editor, a horizontal scroll bar will automatically be added to the bottom to allow you to see the rest of the formula.

9.5.2.3 Auto Indentation

We expect that you'll want to see your DAX expressions indented, which is why this setting is on by default. The key thing to recognize is that based on our "do no harm" philosophy, we are only indenting your DAX in our form, and not in the model (unless you click the Update button).

Should you wish to see the "natural" measure (in the format that is truly stored in the data model), then just uncheck this box.

10 ModelSleuth

The ModelSleuth provides custom analysis reports designed to help you understand your model, as well as the ability to pull SQL's Dynamic Management Views from the data model to build your own custom reports. Its options are shown here:

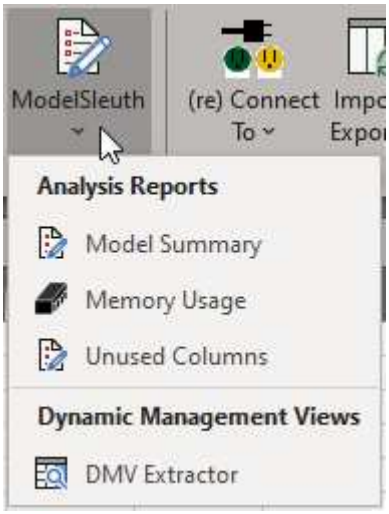


Figure 91 - The Analysis Reports menu

10.1 Model Summary

This report is broken into six main sections that are critical to understanding your model. They consist of:

- Summary Stats
- Relationship Summary
- Table Stats
- Column Stats
- Measure Stats
- Query Stats

Each is explained more fully in its own section below.

10.1.1 Summary Stats

This section of the report is geared towards a quick overview of the model and collects those stats in three related columns: model, queries, and model data sources.

Summary Stats					
Model		Queries		Model Data Sources	
Model Tables:	9	Loaded to Data Model	8	Power Query	8
Model Relationships:	8	Loaded to Connection	36	Direct-External Sources	0
Model Measures:	18	Loaded to Worksheet	2	Direct-Linked Tables	1
Model Memory Used:	1,369.06 KB	Total Queries	46		

Figure 92 – Model Summary Report: Summary Stats table

10.1.1.1 Model

This column displays the number of tables, relationships, measures, and overall memory consumed by the model.

10.1.1.2 Queries

The Queries column gives a quick overview of how many queries are used in the model, and their respective load destinations. It is important to understand that the Total Queries stat may appear to be understated on occasion. This occurs when queries are set to load to both the data model and an Excel worksheet, as the same query will then show up in both categories.

10.1.1.3 Model Data Sources

In the Model Data Sources section, you'll find a breakdown of how your data sources are connected to the data model.

In order to create the most flexible model, it is recommended to always load data to the data model via Power Query, as this allows you to easily modify and update the data later. The first line of this section shows how many queries follow this recommended load practice.

The second line provides a count of the tables that are pulling data from "legacy" connectors (i.e. the connectors inside Power Pivot). The reason this is a challenge is that repointing a data source to a different type involves rebuilding the table if it is based on one of these legacy direct connections. This involves dropping the original table, adding the new one, and then rebuilding all the relationships and measures that were stored on the table. Power Query avoids this as you can just build the new query, compare that the data looks the same as the legacy query, and then repoint your model to pull from the newer version. New data source? Unlike the classic method, with Power Query it's no problem!

Finally, we also display the number of tables that are linked from Excel worksheets directly into the data model. These are challenging to identify visually, so it's important to expose them to you in a fashion that makes sense.

10.1.2 Relationship Summary

Relationships are at the heart of every model, so it's important to be able to identify what they are. This table summarize each model relationship, listing the Primary Key → Foreign Key relationship, identifies if the relationship is active or inactive, and lists the relationship cardinality as well:

Relationship Summary						
Primary Key Table	Primary Key Column	Linked To	Foreign Key Table	Foreign Key Column	Status	Cardinality
Sectors	SectorID	----->	SectorUnitLink	SectorID	Active	Many to One
UnitTypes	UnitTypeID	----->	SectorUnitLink	UnitTypeID	Active	Many to One
Categories	CategoryID	----->	Subcategories	CategoryID	Active	Many to One
SectorUnitLink	Lnk_SUKey	----->	UnitsRemaining	Lnk_SUKey	Active	Many to One
Calendar	DateKey	----->	UnitsRemaining	Date	Active	Many to One
SectorUnitLink	Lnk_SUKey	----->	Forecast	Lnk_SUKey	Active	Many to One
Calendar	DateKey	----->	Forecast	Date	Active	Many to One
Subcategories	Lnk_AcctClass	----->	Forecast	Lnk_AcctClass	Active	Many to One

Figure 93 - Model Summary Report: Relationship Summary

While Excel doesn't currently support relationships other than Many to One, Power BI does. Therefore, One to One or Many to Many relationships will be correctly identified and listed here.

10.1.3 Table Summary Stats

This table shows some key qualitative and quantitative information about your data tables including the table type, observations related to these tables, the number of rows and column, and finally the source of the data table itself:

Table Summary Stats					
Table Name	Table Type	Observations	Rows	Columns	Data Source
Calendar	Dimension		7,671	9	Power Query
Categories	Dimension	Snowflaked	12	3	Power Query
Forecast	Fact		7,945	8	Power Query
QTR201912271211	Fact	Disconnected	30	3	Worksheet Table
SectorUnitLink	Dimension		200	8	Power Query
Sectors	Dimension	Snowflaked	25	7	Power Query
Subcategories	Dimension		108	5	Power Query
UnitTypes	Dimension	Snowflaked	8	5	Power Query
UnitsRemaining	Fact		474	3	Power Query

Figure 94 - Model Summary Report: Table Summary Stats

Why are these items of particular importance to you? Let's look at each:

10.1.3.1 Table Type

Dimensional models contain two types of tables: Fact tables and Dimension tables.

Fact tables are tables that hold data that will be aggregated or summarized, in order to report a value. Examples of this might be the Sum of Units Sold, Sum of Sales $, or Count of Customers. In a well-designed model, your measures will live on Fact tables.

Dimension tables are the tables we use to slice our Facts. They are typically identified with the "by" keyword when you are describing your Fact. In other words, Sales $ ***by*** Year, Customer Count ***by*** Country, or Units Sold ***by*** Month.

10.1.3.2 Observations

The Observations column reports on specific items that Monkey Tools has identified about your model structure including if the table is:

- **Snowflaked:** This indicates that you have a table which contains only one relationship, and this table sits on the "many" side of that relationship. In virtually all cases this table can be merged back to the table on the "one" side of the relationship, then removed from the model.
- **Disconnected:** This indicates that you have a table which contains no relationships at all. While these tables can be very useful to expert modelers, they tend to be problematic for those who don't have a solid command of DAX relationship functions, as they frequently cause issues by allowing the user to build PivotTables or visuals which cannot filter each other. This often manifests in the following PivotTable dialog:

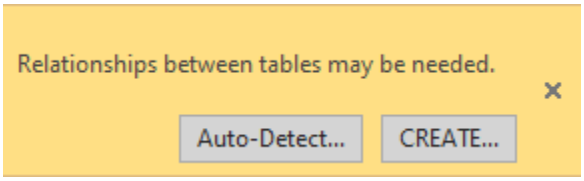


Figure 95 – This error could be caused by adding a field from a disconnected table to an existing PivotTable

10.1.3.3 Rows & Columns

The row and column count of a table is important to be aware of when you are building models that need to be performant. As the Vertipaq engine compresses data within the data model based on unique values in a column, it's important to have a good awareness of this statistic if you need to improve performance.

In order to build efficient models, you want to target building Fact tables that are narrow (as few columns as possible) and long, rather that short and wide. Dimensions, on the other hand can often become short (few rows) and wide (many columns) without massive performance impacts.

10.1.3.4 Data Source

If you recall the initial Model Summary Stats section, the final column reported how the data sources are loaded into the data model. This column shows you the mechanism used to load the given table. This can be extremely useful if you are trying to hunt down which table came from an Excel table, or which table was created based on a legacy direct connection to the data source, rather than coming through Power Query.

10.1.4 Table Column Stats

The Table Column Stats section of the Model Summary report provides a detailed breakdown of your tables by column. This shows you even more information that can be used to optimize your data model:

Table Column Stats						
Table Name	Column Name	Data Type	Unique Values	Hidden	Column Type	DAX Expression
Calendar						
	DateKey	Date	7,671	FALSE	Data Column	
	DateKey (Month Index)	Whole Number	12	TRUE	Calc Column	=MONTH([DateKey])
	DateKey (Month)	Text	12	FALSE	Calc Column	=FORMAT([DateKey], "MMM")
	DateKey (Quarter)	Text	4	FALSE	Calc Column	=CONCATENATE("Qtr", INT((MONTH([DateKey]) + 2) / 3))
	DateKey (Year)	Text	21	FALSE	Calc Column	=FORMAT([DateKey], "yyyy")
	Month Name	Text	12	FALSE	Calc Column	=FORMAT('Calendar'[DateKey],"mmmm")
	Month Short	Text	12	FALSE	Calc Column	=LEFT('Calendar'[Month Name],3)
	ProjectYear	Whole Number	21	FALSE	Data Column	
	Year	Text	21	FALSE	Data Column	
Categories						
	Category	Text	12	FALSE	Data Column	
	CategoryID	Whole Number	12	TRUE	Data Column	
	Class	Text	2	FALSE	Data Column	
Forecast						
	Avg Per Unit	Decimal Number	1,708	FALSE	Data Column	

Figure 96 - Model Summary Report: Table Column Stats

10.1.4.1 Data Type

A column's data type can have a massive impact on the efficiency of the data model, as certain column types require less memory to store values. Some key things to consider:

- If decimal precision is not necessary to you, store your values as Whole Number. Make this change in Power Query, and it will set the values as Whole Number when loaded to the data model.
- If decimal precision *is* important to you, challenge yourself on how much precision you need. If you'll never need more than 4 digits of precision, set the values to Currency in Excel (or Fixed Decimal Number in Power BI) instead of Decimal. Even though the data model uses a Decimal Number format to store all of these data types, if you have only one value that runs out to 12 decimals, for example, the data model has to extend its memory to keep the precision consistent for all values. For this reason, enforcing a smaller number of decimals can help reduce the memory footprint needed to store your data.

- Split DateTimes into separate Date and Time columns. Combined with the fact that it will create fewer unique values (explained next), the DateTime format carries the maximum number of decimals if times are added.
- If it looks like a number, consider formatting it as a number. This is the exact opposite of how we used to look at data in Excel in the past. Our rule used to be, "if you'll never perform math on it, format it as text." Unfortunately in the data model, text is the least efficient way to store data from a memory standpoint.

> *Pro Tip: Be aware that the data type chosen can affect not only compression, but also your user interface experience. Setting a field like Year as a whole number affects what happens when you simply click the checkbox next to a field when building a PivotTable or Visual. If the format is numeric, it will place it in the values area, assuming it needs to be summed. If it's text, it will place it on a dimensional field, assuming you want to split your facts by the selected field.*

10.1.4.2 Unique Value

Unique values in a column is *the most important* fact to be aware of in terms of column compression, and therefore to overall model size. Simply put, fewer unique values lead to better compression. The better the compression, the faster the reaction in filtering and the more stability you get in an Excel model. If you want to build highly performant models, you should challenge the assumptions around how you currently store your data.

As an example, assume you have a billion unique values ranging from $0.12 through $98,244,142.96, formatted as Currency (Fixed Decimal). That's whole lot of unique values. To increase model compression, you could split the existing column into four separate columns as follows:

- Millions: A maximum of 99 unique values from 0 to 98
- Thousands: A maximum of 1,000 unique values from 0 to 999
- Hundreds: A maximum of 1,000 unique values from 0 to 999
- Decimals: A maximum of 100 unique values from 0 to 99

Each of these could now also be formatted as a whole number data type as well, further reducing the memory pressure on the model.

So how do you summarize it later? Simple, you use a measure:

```
=SUM(
      SUM(Sales[Millions]) * 1,000,000
      + SUM(Sales[Thousands]) * 1,000
      + SUM(Sales[Hundreds])
      + DIVIDE(SUM(Sales[Decimals]), 100, BLANK()
      )
```

The reason this could work better for you has to do with the speed differences of retrieving column contents from RAM (typically processed in speeds of MHz per second) vs calculating measures with CPU (typically measured in speeds of GHz per second).

10.1.4.3 Hidden

Hiding columns can and should be done for four reasons:

- Preventing Cross Filtering Errors
- Flagging Fact Tables
- Security
- Cosmetics

10.1.4.3.1 Preventing Cross Filtering Errors:

As a rule, all Foreign Keys (the column on the Many side of the relationship) should be hidden. This will prevent a user (including you) from accidentally placing a Foreign Key on a table, and instead force them to use the Primary Key from the related table. This can cause both the dreaded "Relationships May Be Needed" message in Excel, or manifest in a Pivot Table or visual reporting the grand total on every row.

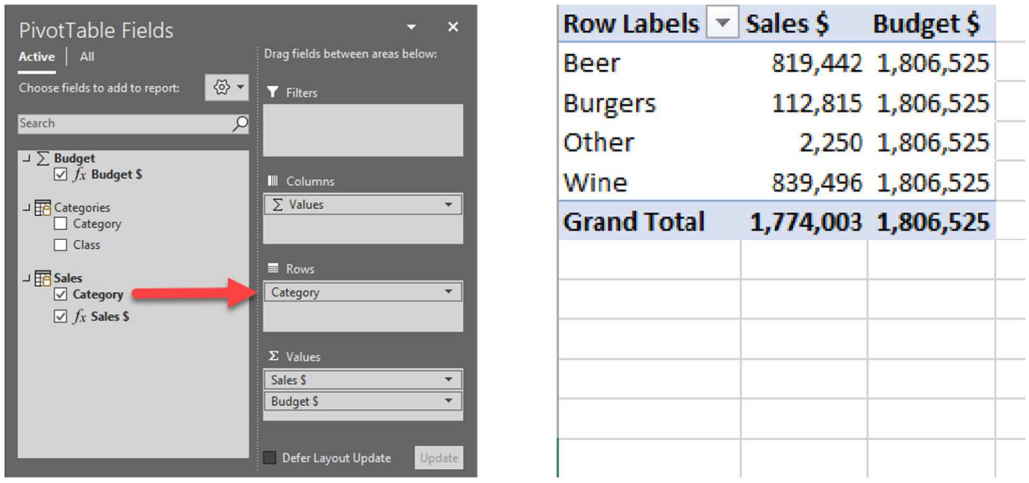


Figure 97 - Category was pulled from a Fact table and can't cross-filter Budgets

Our recommended rule is to play it safe. When you create a relationship, immediately hide the Foreign Key and prevent your users from ever getting into this situation.

10.1.4.3.2 Flagging Fact Tables

When you hide all un-aggregated columns on a table and leave only measures, it changes the way the data model shows the table in the user interface, recognizing it as a Fact table. In Excel, the table will be prefaced with the Σ icon, the same symbol that appears next to the Values area of a PivotTable field list. In Power BI, a calculator symbol is used. In addition to the symbol, you'll also see that all Fact tables move to the top of the field list, with all dimension tables appearing below.

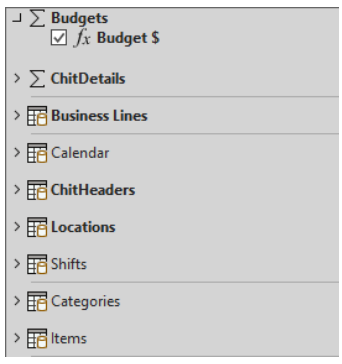


Figure 98 - Budgets and ChitDetails showing as Fact tables in Excel

10.1.4.3.3 Security

The next reason to hide columns, and potentially some measures, is for security. For any models published to Power BI (even those from Excel), visible columns contain two potential security issues:

1) They can be exposed via Power BI's Q&A feature, even if they are not on your reports. Hide them and this will no longer be the case.
2) Raw data columns placed in the values area of a Power BI visual create an "Implicit Measure" (such as Sum of Sales $). Implicit measures expose the "See Records" feature in a Power BI visual, which allows displaying of all the table rows that were summarized to generate the data point, including all columns. Hide these columns to prevent accidentally leaking sensitive information!

For more on this subject, see Can Users See my Raw Data in Power BI?

10.1.4.3.4 Cosmetics

The final reason to hide columns in your model is simply one of cosmetics and ease of use. If a user is never going to place the field on a Pivot Table or Visual, then why leave it in the field list at all? Hide it to avoid overwhelming them with irrelevant information. (And if it's not used in the model for any other reason, consider removing it all together!)

10.1.4.4 Column Type

The Column Type portion of the Table Column Stats section simply lists if the column is a Data Column, or if it is a Calculated Column. Data Columns (also known as "natural" columns) are contained in the original data loaded to the table, while Calculated Columns are created via DAX expressions. The key thing to be aware of here is that Calculated Columns take both processor power to calculate as well as memory to store, so are not extremely efficient. They should be avoided wherever possible.

10.1.4.5 DAX Expression

The DAX Expression column contains the DAX formula used to generate any calculated columns and is expressed in the natural (un-indented) format that is present in the model.

> *Pro Tip: Even though it doesn't look like it, this report is a real Excel table. This means that you can select it, go to Table Tools, and turn on the Filter drop downs or alternate row banding should you wish to do so. It also means that if you have SQLBI's Power Pivot Analyzer installed, you can select the DAX Expression column and use their DAX Formatter on it to indent and beautify the code.*

10.1.5 Measure Stats

The Measure Stats table answers key questions you should ask with regards to your measures, including:

- Host Table: What table is the measure actually stored on?
- Table Type: Is the table a Fact or Dimension table?
- Measure Type: Is the measure Implicit or Explicit?
- Used: Is the measure actually used anywhere in the model?
- DAX Expression: What is the formula used to calculate the measure value?

The big question, of course, is why you might care.

Measure Stats						
Measure Name	Host Table	Table Type	Measure Type	Used		DAX Expression
Amount	Forecast	Fact	Explicit	Yes	----->	=IF(HASONEVALUE(Categories[Class]), IF(VALUES(Categories[Class])="Revenue", [Revenues], [Costs]), [Revenues]-[Costs])
Avg $ / Front m	Forecast	Fact	Explicit	Yes	----->	=DIVIDE([Amount],[Front m])
Avg Units Sold	Forecast	Fact	Explicit	Yes	----->	=CALCULATE(AVERAGE(Forecast[Units]), Categories[Category]="Unit Sales")
Costs	Forecast	Fact	Explicit	Yes	----->	=CALCULATE(SUM(Forecast[Gross Amount])*-1, Categories[Class]="Costs")
Front m	Forecast	Fact	Explicit	Yes	----->	=CALCULATE([Units Sold], ALL(Categories[Class]))*AVERAGE(UnitTypes[Frontage (m)])

Figure 99 - Model Summary Report: Measure Stats

10.1.5.1 Host Table

This column identifies which table hosts the measure. If you prefer editing your DAX formulae inside the Power Pivot window, this column tells you exactly which table you need to go to in order to do so.

10.1.5.2 Table Type

Generally, it's a good idea to store all of your measures on Fact tables, rather than Dimension tables. This helps keep them collected in one place, as well as occasionally helps avoid the "Relationships May Be Needed" error. This column allows you to quickly scan your model and see what type of table you've stored your measure on so that you can take corrective action if necessary.

10.1.5.3 Measure Type

The difference between an Implicit and Explicit measure is as follows:

Implicit measures are very easy to create, as you simply need to drag any un-aggregated column into the values area of a PivotTable, PivotChart, or Power BI visual. They tend to be limited in scope, only allowing certain aggregations, and don't carry any default formatting with them. In addition, they expose Power BI visuals to the "See Records" feature, which can potentially leak data accidentally.

Explicit measures require writing a DAX expression in order to generate the exact results you need. While they are harder to create – it does require learning the DAX language – they carry many benefits, including much more robust calculation ability, default formats, and hiding the "See Records" feature.

Our recommendation is to use Explicit measures over Implicit measures in all cases, and the Measure Type column allows you to quickly identify where they are.

Pro Tip: Unfortunately, due to the inability to delete Implicit measures via code, we cannot do this for you. And even worse, Implicit measures are hidden in the model by default, whether used or not. To remove an Implicit measure from your model completely, you must:

- *Open the Power Pivot window*
- *Select the table the Implicit measure is stored on*
- *Go to Advanced → Show Implicit measures*
- *Select the Implicit measure and press the DEL key*

10.1.5.4 Used

For Excel-based models, Monkey Tools scans all data model objects that could potentially use your columns, including relationships, calculated columns, measures, and sorting hierarchies, as well as all Excel slicers, timelines, Pivot Tables, Pivot Charts, OLAP formulas, and named ranges. If it comes back and tells you that the measure is not in use, you may want to simply delete it.

10.1.5.5 DAX Expression

The DAX Expression column contains the DAX formula used to generate any calculated columns and is expressed in the natural (un-indented) format that is present in the model.

> *Pro Tip: Even though it doesn't look like it, this report is a real Excel table. This means that you can select it, go to Table Tools and turn on the Filter drop downs or alternate row banding should you wish to do so. It also means that if you have SQLBI's Power Pivot Analyzer installed, you can select the DAX Expression column and use their DAX Formatter on it to indent and beautify the code.*

10.1.6 Query Stats

The final section of the Model Summary Report is the Query Stats section, which lists key information about every query in the solution, including:

- Query Name
- "Load To" Behaviour
- Dependent Queries
- Precedent Type
- Precedent Source

Query Stats					
Query Name	Load To	Dependent Queries	Precedent Type		Precedent Source
Calendar	Data Model	Level2 Staging-UnitsRemaining	Query	----->	Level1 Staging-Calendar
Categories	Data Model		Query	----->	Level1 Staging-RevenueCategories
			Query	----->	Level2 Staging-CostCategories
fnGetParameter	Connection Only	Level1 Staging-Calendar	Current Workbook	----->	Parameters
			Query	----->	Level1 Staging-Parameters
fnRevenueAllocation	Connection Only	Level1 Staging-SaleTiming			
Forecast	Data Model		Query	----->	Level4 Staging-Forecast
Level1 Staging-Calendar	Connection Only	Calendar	Query	----->	fnGetParameter
Level1 Staging-Declining	Connection Only	Level2 Staging-CostCategories	Current Workbook	----->	rngCosts_Declining
		Level2 Staging-CostSubCategories			
		Level3 Staging-DecliningCosts			
		Level4 Staging-Subcategories			
Level1 Staging-ErosionA	Connection Only	Level2 Staging-ErosionAdj	Current Workbook	----->	rngSedimentErosion_Adj
Level1 Staging-ErosionTi	Connection Only	Level4 Staging-ErosionAccount	Current Workbook	----->	rngSedimentErosion_Title
Level1 Staging-OngoingC	Connection Only	Level2 Staging-tSubCategories	Current Workbook	----->	rngCosts_Ongoing

Figure 100 - Model Summary Report: Query Stats

10.1.6.1 Reading the Query Stats Table

While many of the columns are self-explanatory, it helps to recognize how the Dependent Queries, Precedent Type, and Precedent Source columns work.

10.1.6.1.1 Dependent Queries

As each Power Query can be referenced by multiple others, this provides a list of all Power Queries which rely on the query listed in the first column, with each dependent query listed on its own line.

10.1.6.1.2 Precedent Type

The Precedent Type tells us about the query (or queries) that feed into the query listed in the first column. The most important thing to be aware of here is that any query that is not listed as "Query" or

<blank> attaches to an actual data source. (See more on how to leverage this in the Working with the Query Stats Table section below.)

10.1.6.1.3 Precedent Source

The Precedent Source column provides you details about the actual precedent source that feeds the query referenced by the first column. In the case that the precedent is a Power Query, you'll see the precedent query's name. But in the case that the precedent query connects to a data source, you'll see the details of the data source (folder path, file name, variable, URL, etc.).

10.1.6.2 Working with the Query Stats Table

One handy thing you can do with this table is quickly filter to just the queries that attach to data sources. To do this:

- Select any cell in the table and go to the Table Design tab
- On the Table Style Options group, check the Filter Button checkbox
- Click the Filter icon at the top of the Precedent Type column
- Uncheck Connection Only and <blank>

You'll now be looking at a table that shows only the Power Queries that make connections to data sources:

Query Stats					
Query Name	Load To	Dependent Queries	Precedent Type		Precedent Source
fnGetParameter	Connection Only	Level1 Staging-Calendar	Current Workbook	----->	Parameters
Level1 Staging-Declining	Connection Only	Level2 Staging-CostCategories	Current Workbook	----->	rngCosts_Declining
Level1 Staging-ErosionA	Connection Only	Level2 Staging-ErosionAdj	Current Workbook	----->	rngSedimentErosion_Adj
Level1 Staging-ErosionTi	Connection Only	Level4 Staging-ErosionAccount	Current Workbook	----->	rngSedimentErosion_Title
Level1 Staging-OngoingC	Connection Only	Level2 Staging-CostSubCategories	Current Workbook	----->	rngCosts_Ongoing
Level1 Staging-Paramete	Connection Only	fnGetParameter	Current Workbook	----->	Parameters
Level1 Staging-Pricing	Connection Only	Level3 Staging-Revenues	Current Workbook	----->	rngPricing
Level1 Staging-Proportio	Connection Only	Level4 Staging-LandAdjustments	Current Workbook	----->	rngLandUse
Level1 Staging-Reporting	Connection Only	Subcategories	Current Workbook	----->	ReportingCategories
Level1 Staging-SaleTimir	Connection Only	Level2 Staging-CostTiming	Current Workbook	----->	rngUnits_SalesDates
Level1 Staging-SectorCo	Connection Only	Level2 Staging-CostCategories	Current Workbook	----->	rngCosts
Level1 Staging-Sectors	Connection Only	Level2 Staging-ErosionAdj	Current Workbook	----->	tblSectors
Level1 Staging-SunkCos	Connection Only	Level2 Staging-CostSubCategories	Current Workbook	----->	rngSunkCosts
Level1 Staging-UnitFront	Connection Only	Level2 Staging-CostsPerUnit	Current Workbook	----->	rngLandUse
Level1 Staging-UnitTypes	Connection Only	Level2 Staging-SectorUnitLink	Current Workbook	----->	UnitTypes

Figure 101 - Listing all data source connections made by Power Query

10.2 Model Memory Usage

The ModelSleuth also contains a pre-built Model Memory Usage report, allowing you to discover what is eating up your model memory:

Memory (KB)				Status	Encoding	
				In Use	Recoverable	
Source Table	Column	Data Type	Unique Values			Total Memory (KB)
Forecast				488.25	19.45	507.71
Calendar				432.92	74.67	507.60
SectorUnitLink				23.45	66.80	90.25
Sectors				52.28	33.28	85.56
Subcategories				60.59	20.22	80.81
Categories				35.04		35.04
UnitsRemaining				24.82		24.82
UnitTypes				18.67		18.67
QTR201912271211					18.62	18.62
Total Memory (KB)				1,136.02	233.05	1,369.06

Figure 102 - Model Memory Usage Report

Running the report is easy: simply select ModelSleuth → Model Memory Usage report. Monkey Tools will analyze your model, and then present you with a drillable PivotTable that you can explore to learn more about it.

10.2.1 Working with the Model Memory Report

As you'll see, this report contains a ton of syntactical information that can help you eliminate unneeded components from Excel models, as well as tweak your model to make it more memory efficient (in either Excel or Power BI). To expose this information, just click the + icon beside a table to expand its contents:

Memory (KB)				Status	Encoding	
				⊞In Use	⊞Recoverable	Total Memory (KB)
Source Table	Column	Data Type	Unique Values			
⊞Forecast				488.25	19.45	507.71
⊟Calendar	⊟DateKey	⊟Date	7,671	388.85		388.85
Calendar	⊟Month Name	⊟Text	12	20.60		20.60
Calendar	⊟Month Short	⊟Text	12		20.52	20.52
Calendar	⊟DateKey (Month)	⊟Text	12		20.52	20.52
Calendar	⊟Year	⊟Text	21	16.96		16.96
Calendar	⊟DateKey (Year)	⊟Text	21		16.96	16.96
Calendar	⊟DateKey (Quarter)	⊟Text	4		16.66	16.66
Calendar	⊟DateKey (Month Index)	⊟Whole Number	12	5.11		5.11
Calendar	⊟ProjectYear	⊟Whole Number	21	1.40		1.40
Calendar Total				432.92	74.67	507.60
⊞SectorUnitLink				23.45	66.80	90.25
⊞Sectors				52.28	33.28	85.56
⊞Subcategories				60.59	20.22	80.81
⊞Categories				35.04		35.04
⊞UnitsRemaining				24.82		24.82
⊞UnitTypes				18.67		18.67
⊞QTR201912271211					18.62	18.62
Total Memory (KB)				1,136.02	233.05	1,369.06

Figure 103 - Looking at memory details for the Calendar table

While your eyes are probably immediately draw to the data bars and values in the table, don't forget to pay attention to the Data Type and Unique Values counts for each column. It's through this information that you can truly start to understand how the memory and compression algorithms work.

> *Pro Tip: It is a good idea to either refresh the model prior to running this tool or run it twice to get an accurate picture of memory usage. The model seems to compress further upon close, so this routine tests the "active" footprint.*

10.2.1.1 Data Compression and Memory Size

For example, notice that the Month Name field contains 12 unique Text values, and consumes 20.52KB in the model. Now look at DateKey (Month Index), which contains 12 unique Whole Numbers, and only consumes 5.11KB.

More interesting is that ProjectYear, with 21 unique Whole Numbers, takes even less space to store, using only 1.4KB. This exposes the fact that it's not *just* unique values that contribute to column compression (although that is the biggest factor). In truth, the order of the columns can also have an impact.

> *Pro Tip: If you want to get into the detailed metrics, you can expand the value columns, which will break the report down further to display the breakdown of Hash encoded and Value encoded memory. The simplified rule is that Value encoded memory is more efficient than Hash encoded*

memory. Therefore, it's preferable to push as much to that encoding as possible, by leveraging numeric-based columns over text-based columns.

Data and memory compression is a complicated topic. If you'd like to learn more about the inner workings, why it is important, and how to optimize your model to the highest degree, consider reading these articles:

- Checklist for Memory Optimizations in PowerPivot and Tabular Models
- Memory Considerations about PowerPivot for Excel
- Creating a Memory Efficient Data Model Using Excel and Power Pivot
- How Does Power Pivot Store and Compress Data?
- What is Eating Up My Memory in Power Pivot?

Caution: The articles linked above are highly technical, and not for the faint of heart!

10.2.2 The "In Use" vs "Recoverable" Breakdown

For Excel-based models, Monkey Tools scans all data model objects that could potentially use your columns, including relationships, calculated columns, measures, and sorting hierarchies, as well as all Excel slicers, timelines, Pivot Tables, Pivot Charts, OLAP formulas, and named ranges. This information is then used to generate an "In Use" or "Recoverable" column. When added to the PivotTable, this feature allows you to see the exact memory cost of carrying unused columns in your data model.

Unfortunately, this feature is currently only available for Excel models, as Monkey Tools doesn't (yet) scan Power BI files to locate used, or unused, components.

10.2.3 Why am I seeing extra Date Tables?

It is not uncommon to connect to a Power BI model and find that Monkey Tools reports a bunch of extra calendar tables that you just don't see in your Power BI file. This is not an error; Monkey Tools is reporting something that is suppressed from the user interface.

This issue occurs when a Power BI file has the "Auto Date/Time" setting for Time Intelligence set to on. What happens at this point is that Power BI will automatically create a full calendar table for every individual date column in your model. (At least, it will unless you have designated a calendar table.)

This setting is so impactful, that we once saw a user's Power BI model where 70% of the file's storage space was dedicated to these extra tables!

We highly recommend that you create proper calendar tables in all the models you build and turn off the Auto Date/Time settings. To do this in Power BI:

- Go to File → Options and Settings → Options

For the model you are currently working on:

- Go to Current File → Data Load → Time Intelligence → uncheck the "Auto Date/Time" box

To supress this for all newly created models in future:

- Go to Global → Data Load → Time Intelligence → Uncheck "Auto Date/Time for New Files"

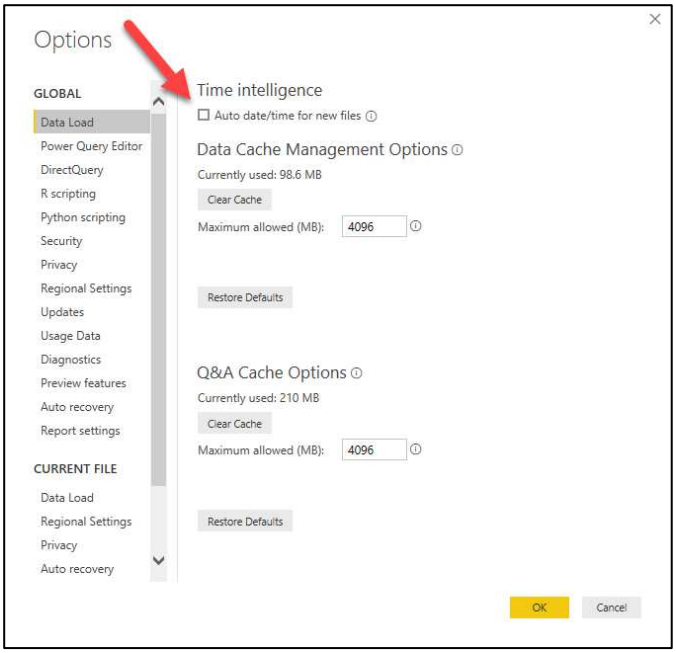


Figure 104 - Preferred settings for Power BI's "Auto Date/Time" Intelligence

10.3 Unused Model Columns

ModelSleuth's Unused Model Columns feature will generate a report that tells you what Power Pivot columns are/aren't being used in your model.

Upon running the report, ModelSleuth will scan your Excel workbook, reading all data model objects that could potentially use your data model columns. This includes relationships, calculated columns, measures, and sorting hierarchies, as well as all Excel slicers, timelines, Pivot Tables, Pivot Charts, OLAP formulas, and named ranges. It then returns a quick message letting you know how many columns are being carried in your model without adding value:

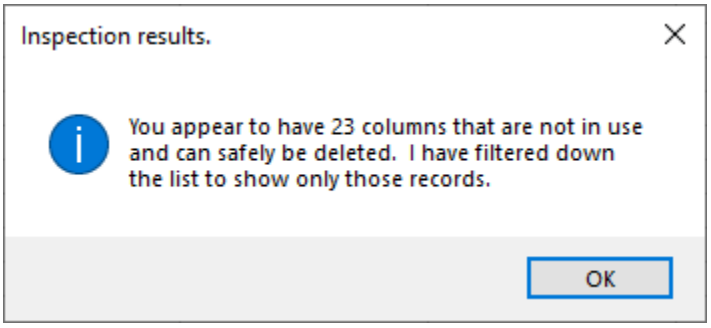
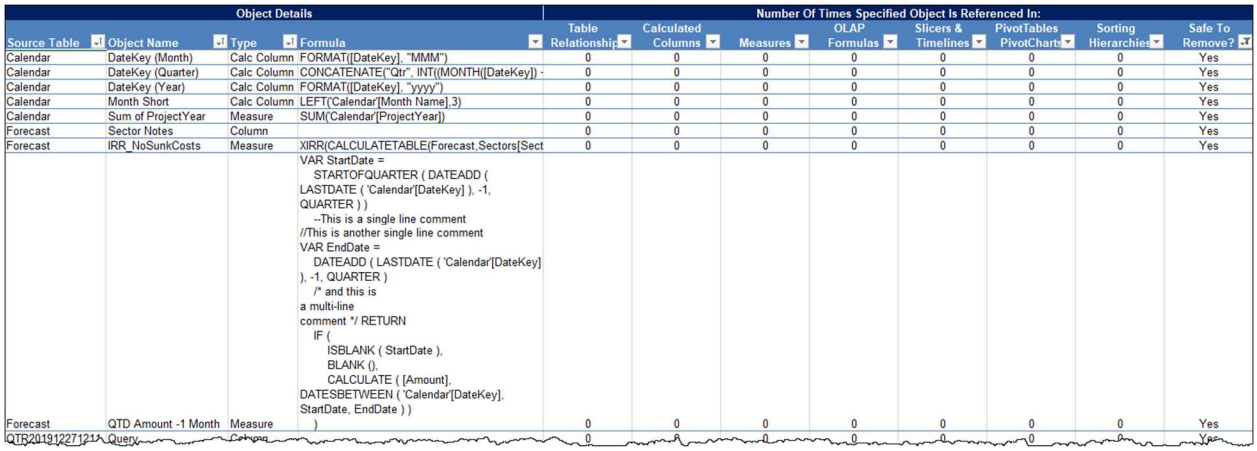


Figure 105 - It appears that we have columns just taking up space in our model for no reason...

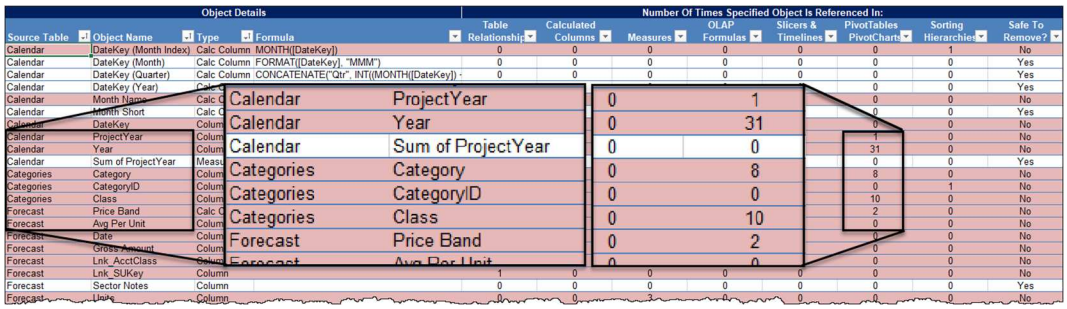
After accepting the dialog, you'll be shown a report that is filtered to only the unused items, as shown below in Figure 106:



Object Details				Number Of Times Specified Object Is Referenced In:							
Source Table	Object Name	Type	Formula	Table Relationship	Calculated Columns	Measures	OLAP Formulas	Slicers & Timelines	PivotTables PivotCharts	Sorting Hierarchies	Safe To Remove?
Calendar	DateKey (Month)	Calc Column	FORMAT([DateKey], "MMM")	0	0	0	0	0	0	0	Yes
Calendar	DateKey (Quarter)	Calc Column	CONCATENATE("Qtr", INT((MONTH([DateKey]) -	0	0	0	0	0	0	0	Yes
Calendar	DateKey (Year)	Calc Column	FORMAT([DateKey], "yyyy")	0	0	0	0	0	0	0	Yes
Calendar	Month Short	Calc Column	LEFT('Calendar'[Month Name],3)	0	0	0	0	0	0	0	Yes
Calendar	Sum of ProjectYear	Measure	SUM('Calendar'[ProjectYear])	0	0	0	0	0	0	0	Yes
Forecast	Sector Notes	Column		0	0	0	0	0	0	0	Yes
Forecast	IRR_NoSunkCosts	Measure	XIRR(CALCULATETABLE(Forecast,Sectors[Sect	0	0	0	0	0	0	0	Yes
Forecast	QTD Amount -1 Month	Measure	VAR StartDate = STARTOFQUARTER (DATEADD (LASTDATE ('Calendar'[DateKey]), -1, QUARTER)) --This is a single line comment //This is another single line comment VAR EndDate = DATEADD (LASTDATE ('Calendar'[DateKey]), -1, QUARTER) /* and this is a multi-line comment */ RETURN IF (ISBLANK (StartDate), BLANK (), CALCULATE ([Amount], DATESBETWEEN ('Calendar'[DateKey], StartDate, EndDate)))	0	0	0	0	0	0	0	Yes
QTR20191227121	Query	Column		0	0	0	0	0	0	0	Yes

Figure 106 - A list of unused model columns

If you'd like to learn about how often any given column is used, simply click the filter icon in the top right corner of the "Safe to Remove" column and check the "No" option. You'll then be able to see how often each column has been referenced in other objects:



Object Details				Number Of Times Specified Object Is Referenced In:							
Source Table	Object Name	Type	Formula	Table Relationship	Calculated Columns	Measures	OLAP Formulas	Slicers & Timelines	PivotTables PivotCharts	Sorting Hierarchies	Safe To Remove?
Calendar	DateKey (Month Index)	Calc Column	MONTH([DateKey])	0	0	0	0	0	0	1	No
Calendar	DateKey (Month)	Calc Column	FORMAT([DateKey], "MMM")	0	0	0	0	0	0	0	Yes
Calendar	DateKey (Quarter)	Calc Column	CONCATENATE("Qtr", INT((MONTH([DateKey]) -	0	0	0	0	0	0	0	Yes
Calendar	DateKey (Year)	Calc C							0	0	Yes
Calendar	Month Name	Calc C							0	0	No
Calendar	Month Short	Calc C							0	0	Yes
Calendar	DateKey	Colum							0	0	No
Calendar	ProjectYear	Colum							1	0	No
Calendar	Year	Colum							31	0	No
Calendar	Sum of ProjectYear	Measu							0	0	Yes
Categories	Category	Colum							8	0	No
Categories	CategoryID	Colum							0	1	No
Categories	Class	Colum							10	0	No
Forecast	Price Band	Calc C							2	0	No
Forecast	Avg Per Unit	Colum							0	0	No
Forecast	Date	Colum							0	0	No
Forecast	Gross Amount	Colum							0	0	No
Forecast	Lnk_AcctClass	Colum							0	0	No
Forecast	Lnk_SUKey	Column		1	0	0	0	0	0	0	No
Forecast	Sector Notes	Column		0	0	0	0	0	0	0	Yes
Forecast	Units	Column		0	0	3	0	0	0	0	No

Calendar	ProjectYear	0	1
Calendar	Year	0	31
Calendar	Sum of ProjectYear	0	0
Categories	Category	0	8
Categories	CategoryID	0	0
Categories	Class	0	10
Forecast	Price Band	0	2
Forecast	Avg Per Unit	0	0

Figure 107 - Examining the usage frequency of a given column

10.4 DMV Extractor

As useful as ModelSleuth's built-in reports are, we also recognize that some users may want to dig further into the items driving their model. To this end we've included the DMV Extractor feature, in order to make it easy to pull SQL-based Dynamic Management Views (DMV's) from the data model and return them to an Excel worksheet.

10.4.1 Basic Usage of the DMV Extractor

To extract any built-in DMV, you simply need to:

- Select the DMV Collection
- Choose the DMV you wish to load
- Click Create

At this point Monkey Tools will load the details of the DMV to a new worksheet in the Excel file.

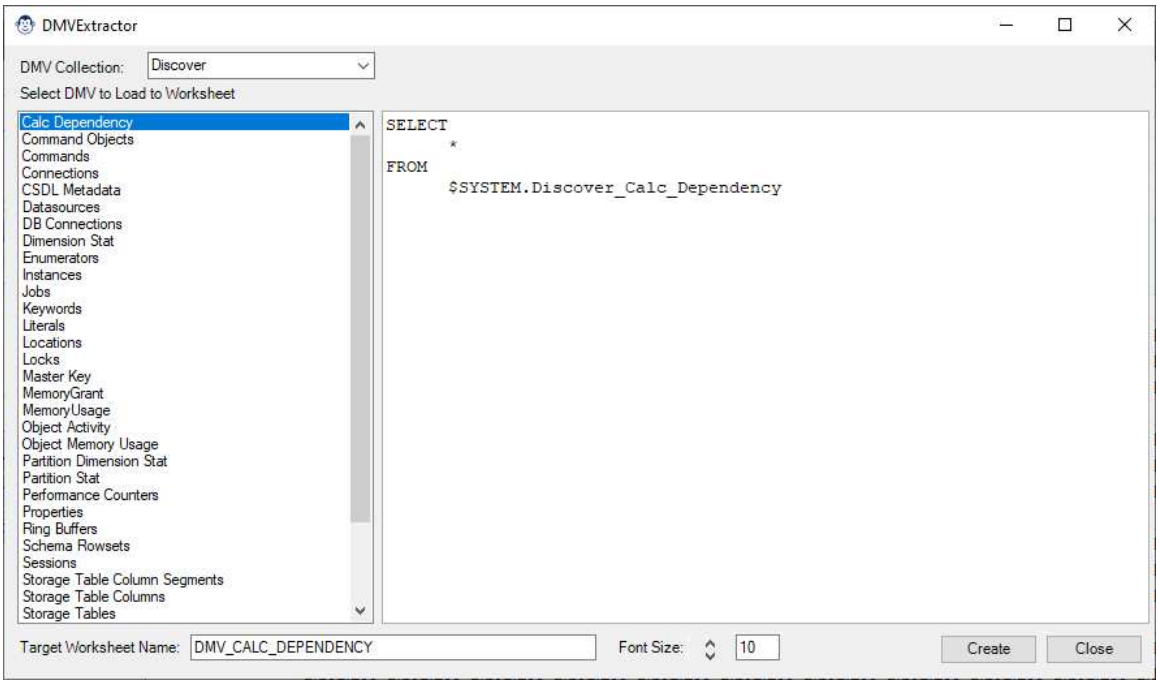


Figure 108 - Creating the DMV_CALC_DEPENDENCY DMV

It should also be noted that not all DMV's can be evaluated for any given model. Should you choose one that isn't supported, you will get the following error message:

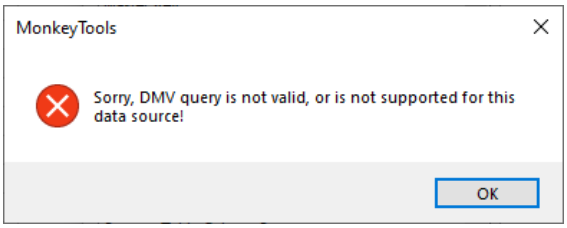


Figure 109 – Shoot, that DMV doesn't seem to work for this model...

10.4.2 Advanced Usage of the DMV Extractor

If you are comfortable modifying the SQL code in order to write your own version of the DMV, this form will allow you to do it. Keep in mind, however, that there is no syntax validation on your SQL. In other words, if you get it wrong, you're going to see a message that looks like this:

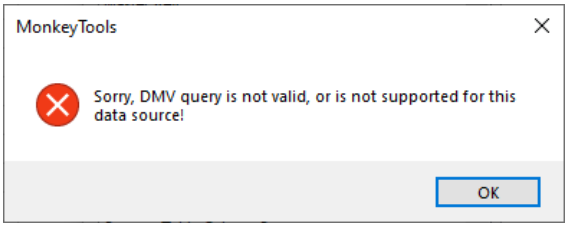


Figure 110 - Uh oh... looks like the query you provided didn't work!

For this reason, when creating custom DMV's, remember to pay careful attention to the following points:

- Make sure you have a space at the end of each line, as characters like hard returns and tabs are trimmed before sending the query to the model
- Use 'single quote' characters around text
- Field names with spaces need to be enclosed in [square braces]
- Not all SQL commands seem to work properly against the DMV's (which can be quite frustrating)

The good news is that we do not close the DMV Extractor after you press the Create button, allowing you to clear the message, adjust your code, and try running it again.

An example that returns a list of measures in the model could look like this:

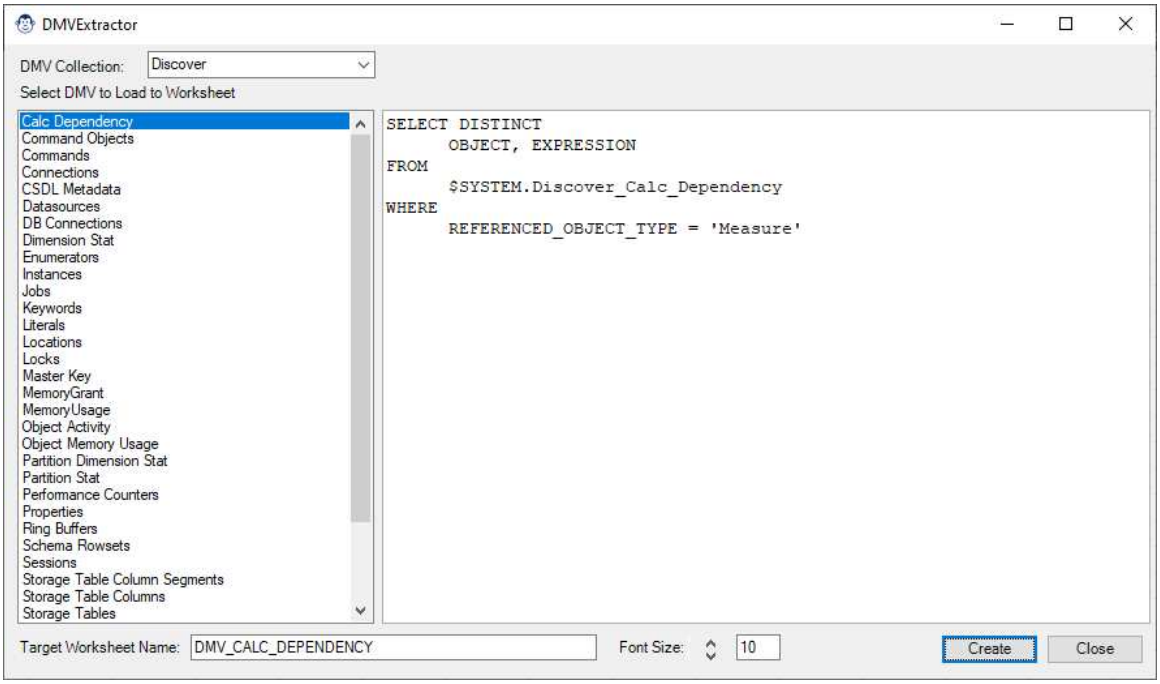


Figure 111 - Creating a custom DMV query

Yielding something like the following:

OBJECT	EXPRESSION
Variance	[Sales $]-[Budget $]
% of Total Sales $	DIVIDE([Sales $],CALCULATE([Sales $],ALL(Categories)),BLANK())
% of Sales Group	DIVIDE([Sales $],CALCULATE([Sales $],ALL(Categories[Category])),BLANK())
MTD Units	TOTALMTD([Sales (Units)],'Calendar'[Date])
Bonus Range	ROUND([Budget $]*1.05,2)
Growth vs 2009	[Sales (Units)]-[2009 Units Sold]
2009 Units Sold	CALCULATE([Sales (Units)],YEAR('Calendar'[Date])=2009)
QTD Units	TOTALQTD([Sales (Units)],'Calendar'[Date])
YTD Units	TOTALYTD([Sales (Units)],'Calendar'[Date])
YTD Units - Mar 31	TOTALYTD([Sales (Units)],'Calendar'[Date],'Calendar'[Date],"Mar 31")
Burgers Sold (All Time)	CALCULATE([Burgers Sold],ALL('Calendar'))
Burgers Sold	CALCULATE([Sales (Units)],Categories[Category]="Burgers")
Draft Sold	CALCULATE([Sales (Units)],Categories[Category]="Draft Beer")

Figure 112 - Results of a custom DMV query

10.4.3 Other features of the DMV Extractor

If you choose to load a DMV to a worksheet that already exists, it will be replaced. For this reason, when you select a new DMV, we set the worksheet name (at the bottom of the form) to *DMV_<the DMV name>*. Should you wish to preserve the DMV results you have, just change the name in the Target Worksheet Name box.

The Font Size spin button will allow you to increase the font size used to display the SQL query.

To indent your code in the main section of the form, press CTRL + TAB. Just don't forget to add a space at the end of the previous line or you could trigger an error, since both hard returns and tab characters are stripped before sending the query to the DMV engine.

11 (re) Connect To…

The (re) Connect To… menu is found on the Utilities group of the Monkey Tools ribbon:

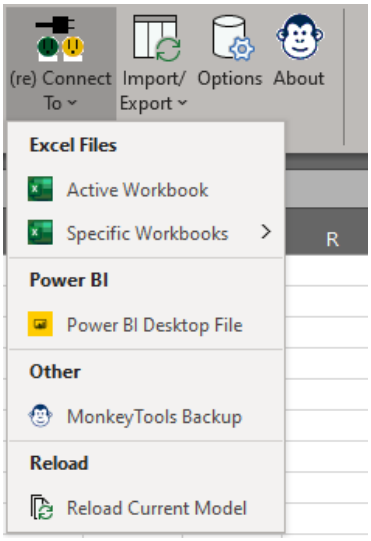


Figure 113 - Monkey Tools Connection options

There are five options in this menu:

11.1 Active Workbook

This feature allows you to connect (or refresh the connection to) the Power Query and Power Pivot model in the current workbook. While most forms will connect to the active workbook by default upon initial launch, this feature allows you to switch back to the active workbook if you have chosen a specific model to connect to via the other options on this menu.

11.2 Specific Workbook

This menu will allow you to connect Monkey Tools to any other workbook that is currently open in Excel. The benefit of this feature is that you can have one workbook active but can analyze a different workbook if needed for any reason.

11.3 Power BI Desktop File

When launching this option, you will be asked to browse to locate the .pbix file you wish to open. Once you have selected to it, Monkey Tools will launch Power BI Desktop, and then perform its analysis of the .pbix file.

Please note that Monkey Tools needs to wait for the Power BI file to launch before it can connect to the model inside the file. Naturally, this can take some time. Unfortunately, Power BI tends to launch in front of the Excel window, obscuring the message that pops up once Excel has received the information it needs:

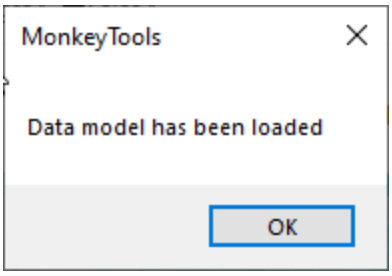


Figure 114 - Data model has been loaded and is ready for analysis!

Our recommendation is to let Power BI launch until it opens the report canvas. At that point – even if the visuals have not all loaded – switch back to Excel where you should see the message shown above. Click OK and you're ready to start exploring the Power BI-based model.

It is also worth knowing that not all features of Monkey Tools currently work on Power BI files (although they are on our roadmap). A comparison matrix of features is shown below:

Group	Excel	Power BI	Missing Features for Power BI
Inject Queries	Yes	No	
-Parameter Table	Yes	No	
-SmartFolder	Yes	No	
-Dynamic Calendars	Yes	No	
-New SmartFolder Query	Yes	No	
Query Load Destinations	Yes	No	
QuerySleuth	Yes	Yes *	Update (to write query to model)
Time Model Refresh	Yes	No	
PivotSleuth	Yes	No	
DAXSleuth	Yes	Yes *	Duplicate, Update
Analysis Reports			
-Full Model Analysis	Yes	Yes	
-Memory Usage	Yes	Yes *	Does not declare unused memory
-Unused Columns	Yes	No	
-DMV Extractor	Yes	Yes	
Import/Export			
-Import to Excel	N/A	Coming	
-Export	Yes	Yes *	Must connect to model first

11.4 Monkey Tools Backup

Monkey Tools contains an Export function in the Import/Export Queries section. The Monkey Tools Backup connector will allow you to import those exported model components so that you can run an analysis on the backup file.

11.5 Reload Current Model

If you ever find yourself needing to reconnect to your active model to refresh the data, this is the feature that will allow you to do so. (Its primary focus is to re-load a Power BI file should you make changes to it.)

12 Import/Export Queries

This section is all about exporting your model components to file for later analysis or retrieval:

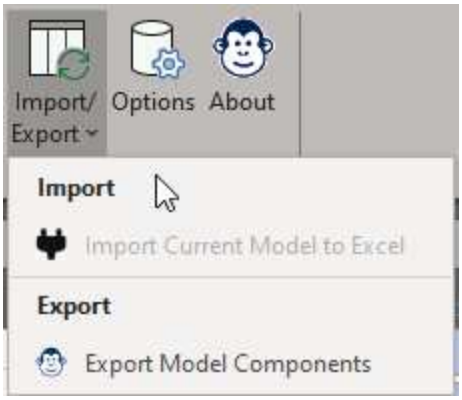


Figure 115 - The Import/Export menu

12.1 Import Current Model to Excel

This feature is currently under development but is intended to re-create the full Power Query and Power Pivot model in the selected workbook. Stay tuned!

12.2 Export Model Components

This section allows you to export model components in a variety of ways:

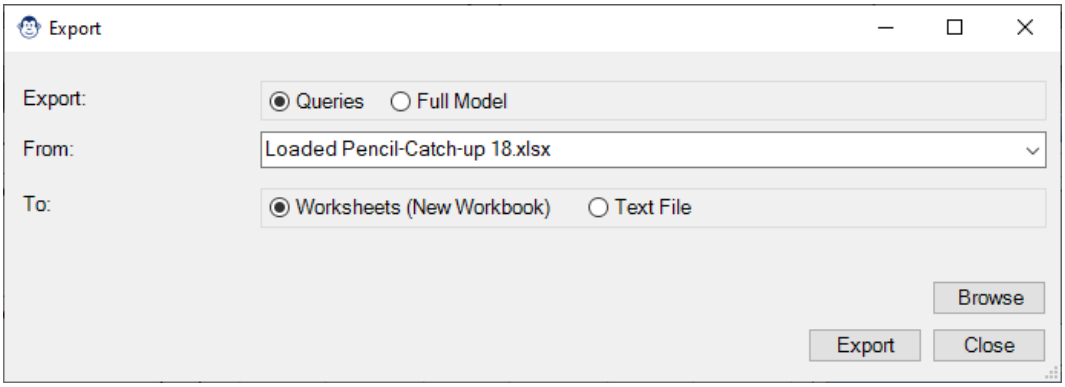


Figure 116 - Monkey Tools Import/Export dialog

12.2.1 Exporting Queries

When exporting Queries, you have two options:

- Export all Queries to worksheet in a new workbook
- Export all Queries to a text file

12.2.1.1 Export Queries to Worksheets

This feature copies the details of each query to a worksheet in a new workbook and provides a summary sheet of the number of queries. Its functionality is intended to allow capturing the state of queries into a static file, allowing users to compare them using third party differencing tools.

12.2.1.2 Export Queries to Text File

This function allows exporting the M code for Power Queries into a text file. These files can then be re-imported for analysis using the Connect To → Monkey Tools Backup feature.

12.2.2 Exporting Full Model

This feature allows exporting the key components of a model into a text file for later retrieval via the Connect To → Monkey Tools Backup feature:

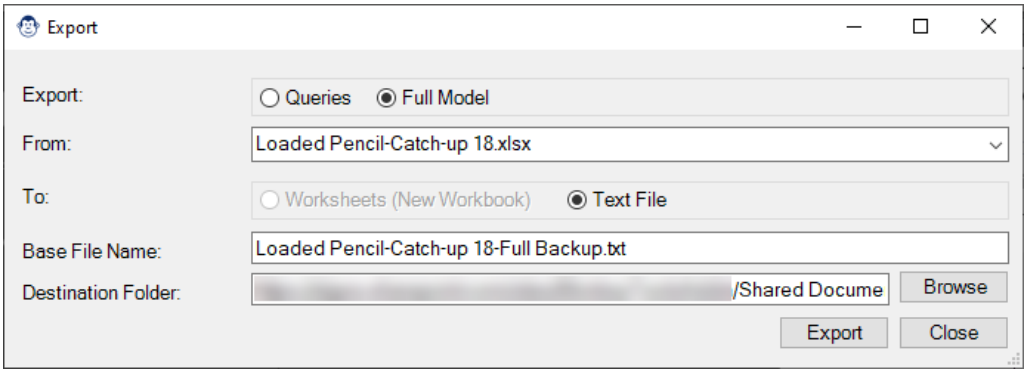


Figure 117 - Exporting the Full Model details to a text file

13 Options

The Options button lets you set and reset global options for Monkey Tools:

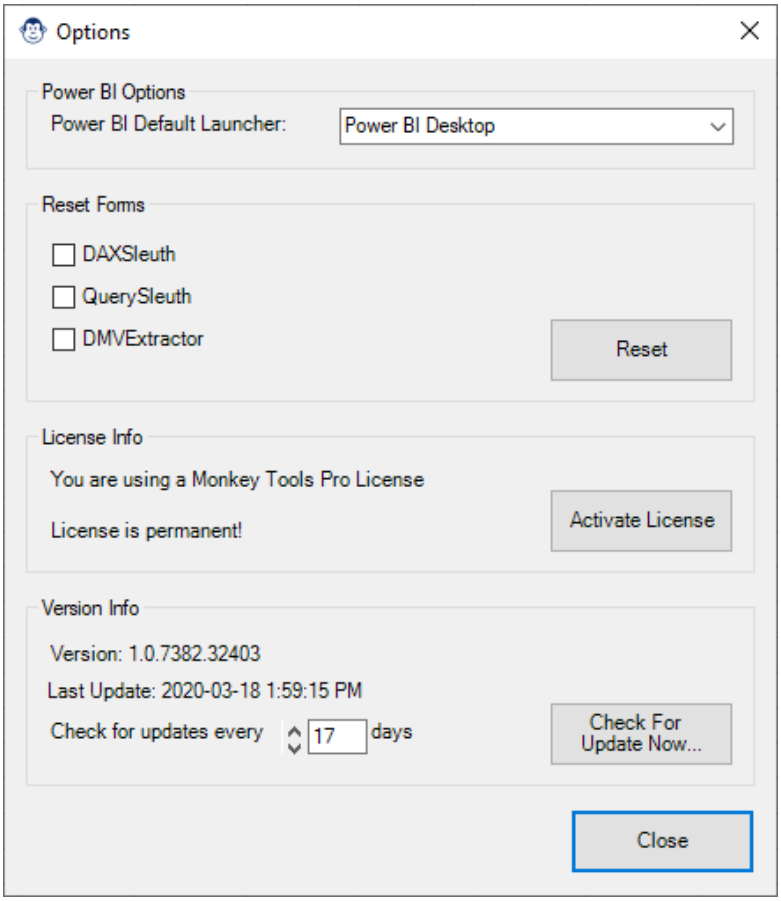


Figure 118 - The Monkey Tools Options form

13.1 Configuring the Power BI Default Launcher Setting

As of Monkey Tools version 1.0.7416.41359, this feature is no longer needed, as Monkey Tools will scan all open Power BI folders (no matter the default launcher) in order to connect. This option will be removed in a future update.

13.2 License Info

This section tells you what type of license you're using, as well as provides you the expiration date for you license. In addition, this is where you can activate your trial/pro license, which is done by clicking Activate License button:

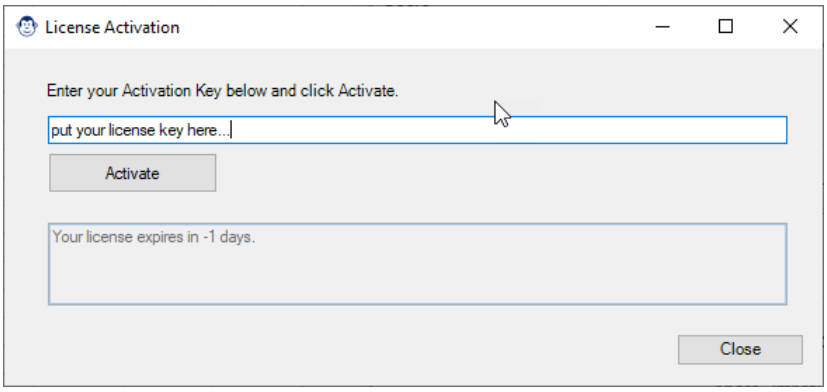


Figure 119 - Entering a Monkey Tools license key

Copy the license key from the email you received, paste it in the form, and click Activate in order to activate the pro (or trial) features.

13.3 Reset Forms

These checkboxes allow you to reset each of the available options to their default states, resetting things like form and font sizes, as well as various configuration options. Check the box(es) next to the form(s) you wish to reset, and then click the Reset button.

13.4 Version Info

This section allows you to see various information related to Monkey Tools, including:

- The Version and build information
- The date and time of the last update

It also allows you to change how often Monkey Tools will check for a new version and allows you to check for an update immediately by pressing the “Check Now…” button.

14 Help

The Help menu contains two important buttons:

14.1 About

The About button is just one of those screens that every software has. The main purpose is to display our amazing company logo, declare that we really do own the copyright to the software, and also provide you with a very quick way to access the current version and build information.

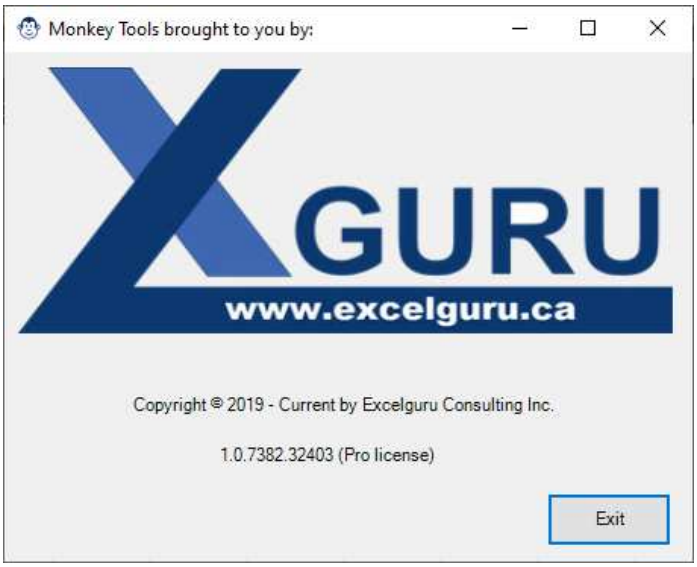


Figure 120 - Monkey Tools About screen

14.2 User Guide

Clicking this button will allow you to view and/or download the most up-to-date version of this document.

15 Troubleshooting & Support

15.1 Installation Errors

It is possible that you'll receive an error attempting to install Monkey Tools from the web URL, advising that your administrator has blocked the installation of our software:

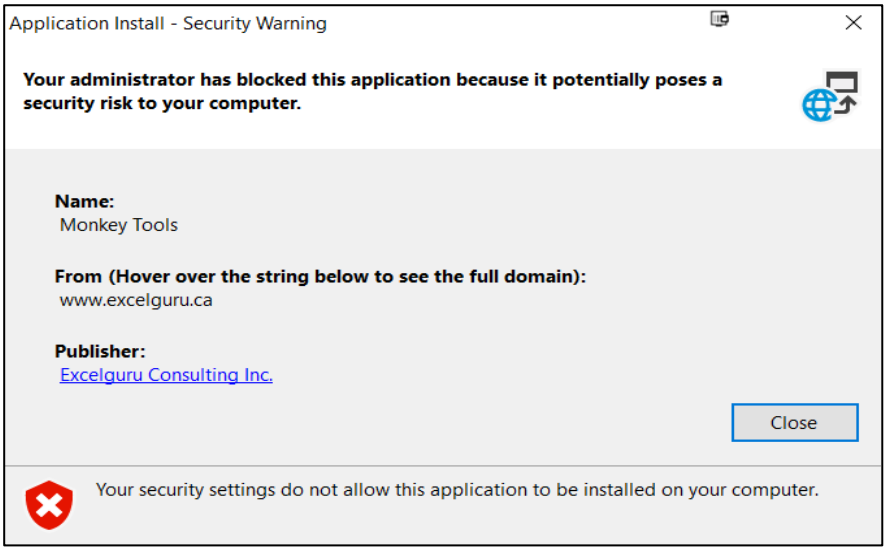


Figure 121 - Your administrator has blocked installation of Monkey Tools

Should this happen, it may be because you have a policy in place that blocks installing software from the internet. Before taking the steps below, please ensure that this dialog still lists Excelguru Consulting Inc as the publisher. Optionally, you can click that hyperlink and install the digital certificate (although it will not resolve the issue on its own).

The good news is that it is possible to remove this block (temporarily or permanently), provided that you have rights to your registry. The bad news is that it takes a quick registry change:

- Press the ⊞ key on your keyboard
- Type RegEdit → press Enter → confirm that you'd like to open the Registry Editor
- Browse to:

  `HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\.NETFramework\Security\TrustManager\PromptingLevel\Internet`

- Change the key to Enabled instead of Disabled
- Close the Registry Editor

Monkey Tools should now install from the web link without issue.

15.2 Disabling Monkey Tools Temporarily

You may need to temporarily disable Monkey Tools. If you wish to do so without completely uninstalling the product (to take screen shots or test if Monkey Tools is conflicting with other add-ins), you can do so via the steps below:

- Go to File → Options
- Select Addins → COM Add-ins → Go

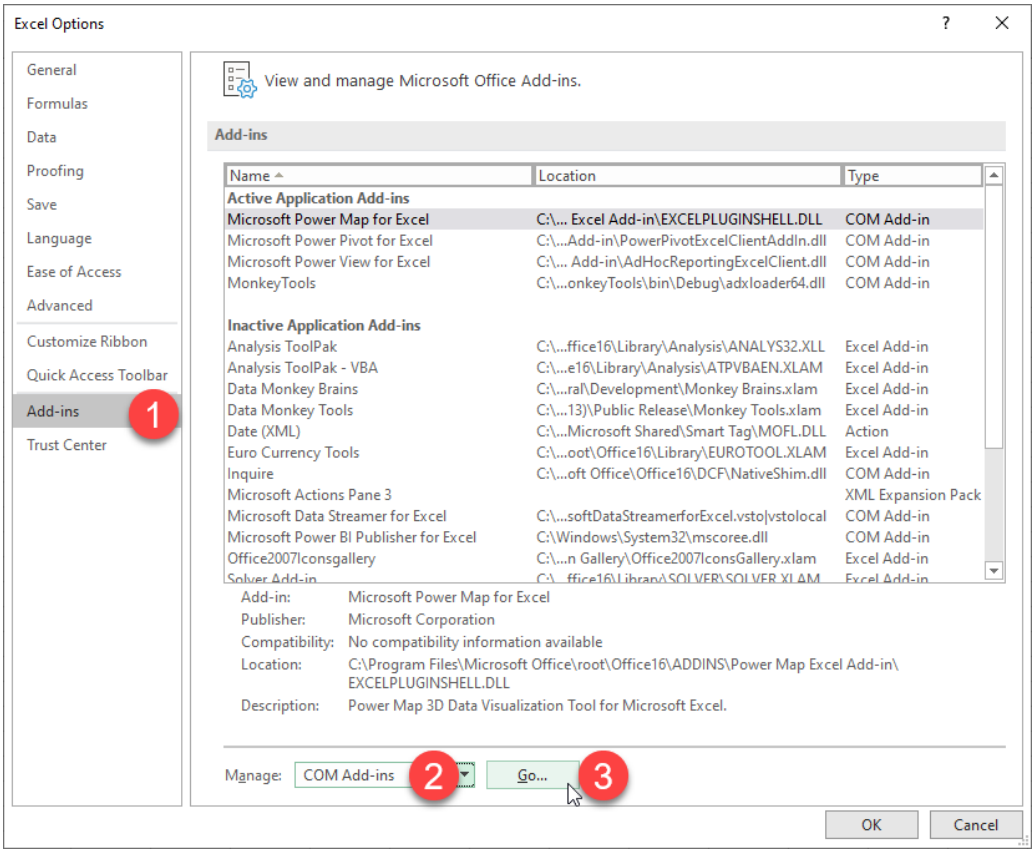


Figure 122 - Managing Excel COM Add-ins

You will be launched into the following dialog, where you can disable Monkey Tools by unchecking the box, then clicking OK.

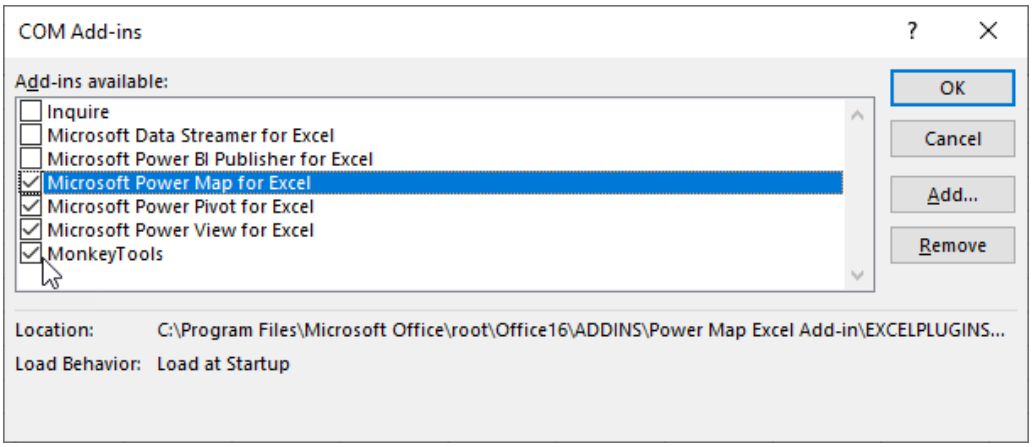


Figure 123 - Enabling and Disabling Excel COM Add-ins

To re-enable the add-in, simply follow the same steps, but re-check the box next to Monkey Tools.