theta

2021

Power BI for Intermediates

A step-by-step training guide

Using Best Practice Methodologies

contact

theta.co.nz enquiries@theta.co.nz 0800-484-382

Authors: Soheil Bakhshi (Microsoft MVP) and Bruce Anderson

Contents

1. Introduction

In this how-to-guide, we build upon learnings from our first guide: Power BI for Beginners. If you're new to Power BI, we recommend you start with this first guide if you haven't done so already.

We'll continue to use sample data from the fictitious Adventure Works bicycle company.

The guide is split into two parts: A and B. In the resources, you can access the .pbix file (ADW-Part1-Theta) that shows completed steps after part A. This might be useful to reference your own dataset against if you need to, or if you want to jump to part B.

In part A of the guide, you'll use Power Query to connect, prepare and model data from multiple Excel files.

In part B of the guide, you'll use Power BI Desktop to:

- Visualise the data to get a clearer picture about the business based on the facts hidden in the data.
- Use Quick Measures to extend measures with time intelligence to provide meaningful insights.
- Get custom visuals.

For terminology, refer to the Glossary in the appendix.

denotes an extra tip or trick you might want to know.

shows where there are instructions to follow.

Note: If you see a popup window that says, **"There are pending changes in your queries that have been applied. Do you want to apply them"**, select yes.

Getting Started

In this section, we discuss Power BI Desktop, sample data and different phases in development.

1.1 Power BI Desktop

Full instructions for downloading Power BI Desktop are available in our first guide. Once installed on your local computer, you will be able to connect to different sources, transform, and visualise your data.

Note: Power BI occasionally updates its user interface, so screenshots in this guide may vary slightly to what you see on your screen.

1.2 Sample Data

You will have received the sample files alongside this guide. Although the files are similar to the ones used in the Beginners Guide, there are some key differences. You must use these new files for this Intermediate Guide.

- Find the sample files.
- Save the file to your local computer and unzip the content into a folder that can be accessed by Power BI Desktop.
- Browse to the saved data files: **DimCustomer** and **AdventureWorks2017.** You will also have **ADW-Part1-Theta.**

1.3 Power BI – Phases in Development

Power BI is designed to be user friendly. Once you connect to a data source, you can shape and transform the data (remove columns, change data types, and so on), do data modelling (create relations) and visually present that data.
This guide steps you through the following phases:

- Connect to data sources.
- Shape the data.
- Model the data.
- Report on the data.

Now, let's get started with the Power BI Desktop tool.

Part A

You'll need these files:

1. DimCustomer (.csv)
2. AdventureWorks 2017 (.xlsx)

Looking for further Power BI training for your team?
Check out our **training options**.

For other data and insights related training, try our
Data Accelerate workshops.

2. Connecting to Data Sources

2.1 Data Sources

Power BI can already connect to over 110 different data sources and connection types, with more being added. As well as connecting to Text/CSV files - like we covered in the Beginners Guide - some are more complex.

Commonly used Power BI data sources include:

- **File** (Excel, Text/CSV, XML, JSON, PDF).
- **Database** (SQL Server, Oracle, IBM DB2, MySQL, PostgreSQL, Snowflake, etc).
- **Power Platform** (Power BI datasets, Power BI Dataflows, Common Data Services)
- **Azure** (SQL Database, Synapse Server, Analysis Services, Blob Storage, Data Lake, Cosmos DB, etc).
- **Online Services** (SharePoint, Dynamics 365, Dynamics Business Central, Azure DevOps, Salesforce, Google Analytics, Facebook, GitHub, etc).
- **Other** (Web, OData, ODBC, Hadoop, Spark, R script, Python script, etc).

The available data sources can be accessed via the **Get data** button in Power BI

2.2 Connection Modes

Some data sources allow you to choose the Data Connectivity mode (i.e. connecting directly to data). There are four options available:

2.2.1 Import

Data is imported into the Power BI dataset and cached in memory. When you submit report and dashboard queries to the dataset, it returns results from the imported data.

You must refresh the dataset to get any changes that have occurred in the underlying data source. This has many advantages, including increased performance and the ability to work offline.

We used this mode in the Beginners Guide when importing from CSV files. We'll use the import mode in this guide.

2.2.2 DirectQuery

Data is not imported into the Power BI dataset from the data source. Report and dashboard queries submitted to the dataset will result in new data being returned from the data source. In this mode, refreshing the dataset is not necessary. Not all data sources offer a DirectQuery option; for example, CSV files are only available for import, so we will not discuss this mode in this guide.

Data sources have limitations when using DirectQuery. Details of these can be found on the Microsoft website here.

2.2.3 Mixed Mode

Mixed mode refers to occasions where data is imported into Power BI, while some others are in DirectQuery mode. This mixed mode leans towards more advanced data modelling in Power BI, so won't be covered in this guide.

2.2.4 Connect Live

Connect Live is a specific type of connection which only supports SQL Server Analysis Services (SSAS) databases, either Multidimensional or Tabular models, and Power BI Datasets. In this mode, the data model is held by an instance of SQL Server Analysis Services or in a Power BI Dataset in Power BI Service. Currently, when we're connecting live to an instance of SSAS or a Power BI Dataset, Power BI turns into a data visualisation tool only. Therefore, data transformation and data modelling are not currently available in this mode. We can create report level measures using DAX when connected live to a SSAS Tabular instance or a Power BI Dataset. Connect Live won't be covered in this guide.

2.3 Importing Data

Let's get started with importing the data:

- Open Power BI Desktop as shown in Figure 1 below.

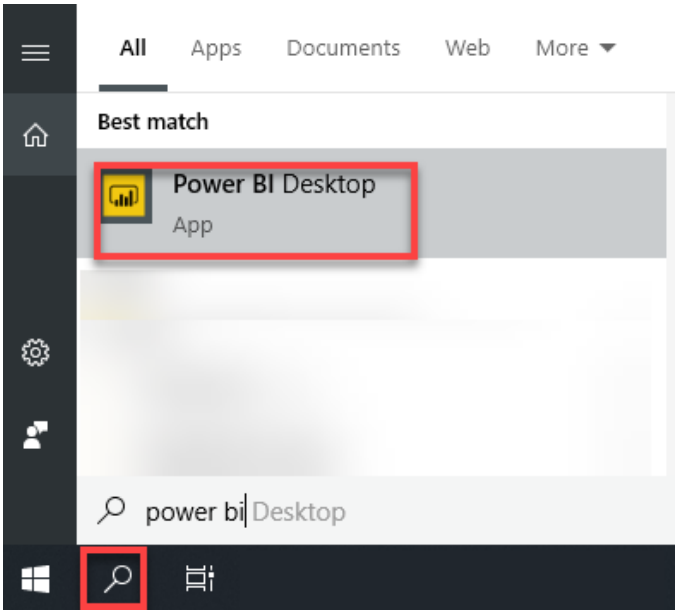


Figure 1: Opening Power BI Desktop from Windows Search

- When you launch Power BI Desktop, a welcome splash screen is displayed
- To connect to the sample data for this exercise, select **Get data**

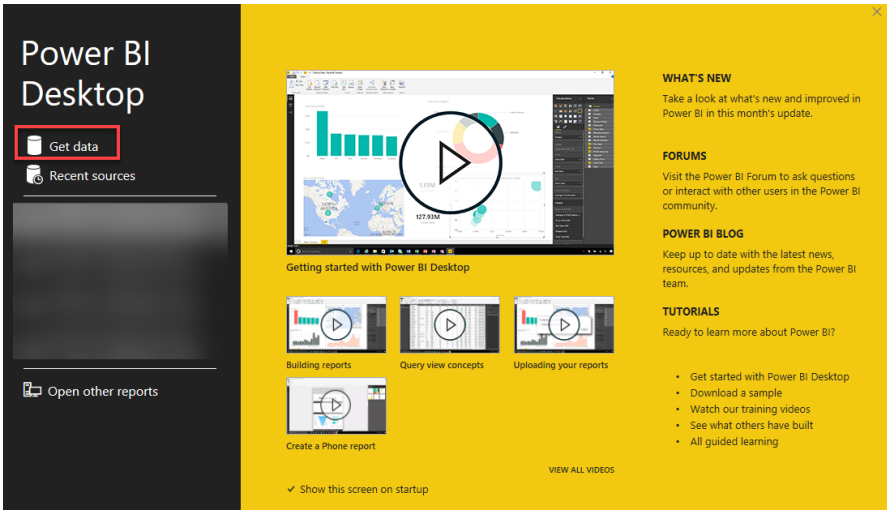


Figure 2: Get data from Power BI Desktop splash screen

- Alternatively, click the **Get data** button from the **Home** tab on the ribbon bar

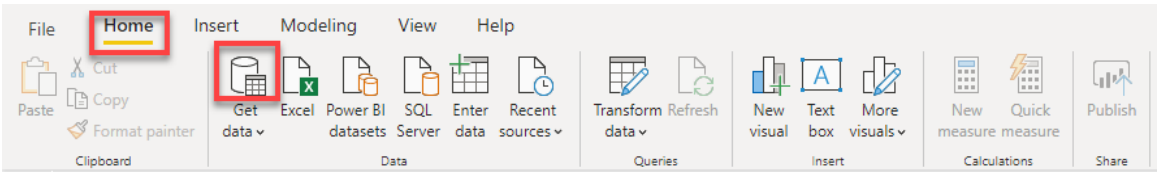


Figure 3: Get data from the Home tab in Power BI Desktop

Selecting the down arrow on the Get data button shows the most common data sources menu. Select More… to open the Get Data dialog.

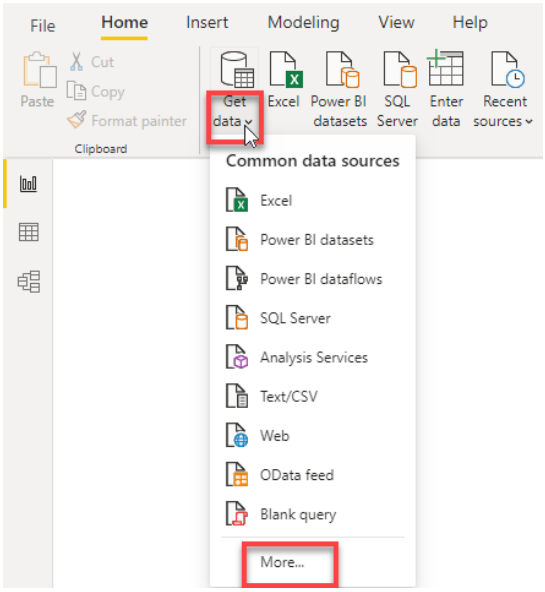


Figure 4: Most common data sources shows by clicking the down arrow on the Get data button

- Select **Text/CSV** from the list and click **Connect.**

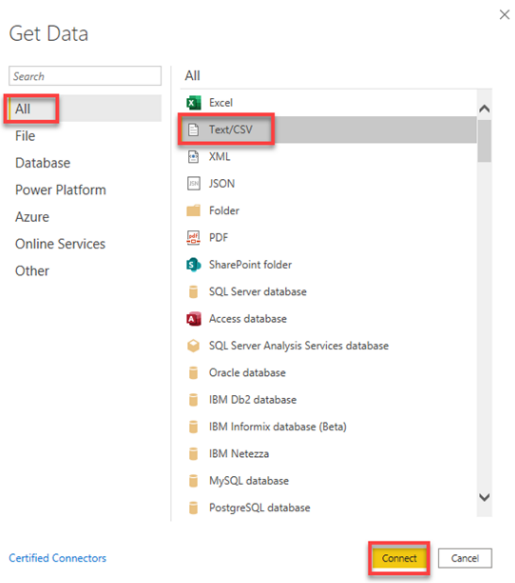


Figure 5: Getting sample CSV data

- Browse to the unzipped data files, select the first file **DimCustomer.csv**, and click **Open.**

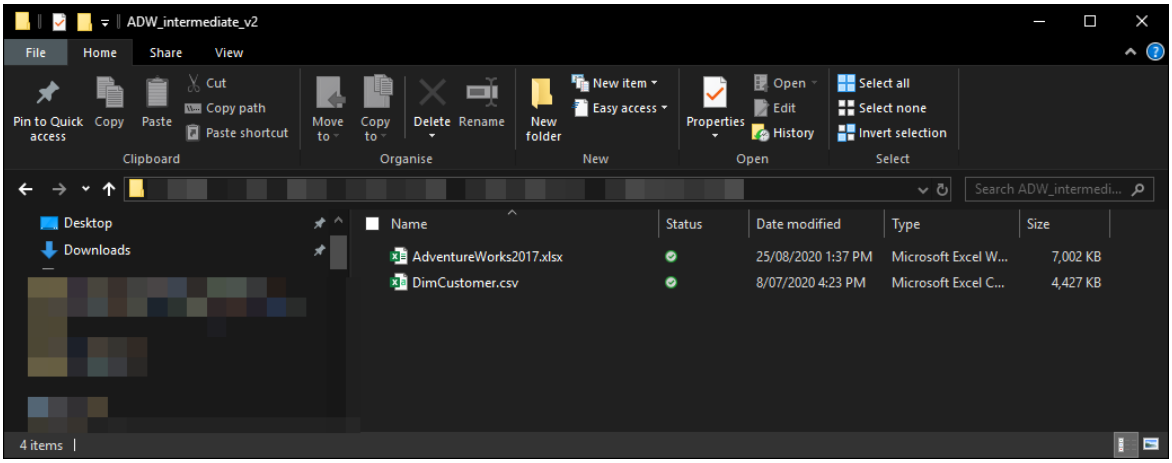
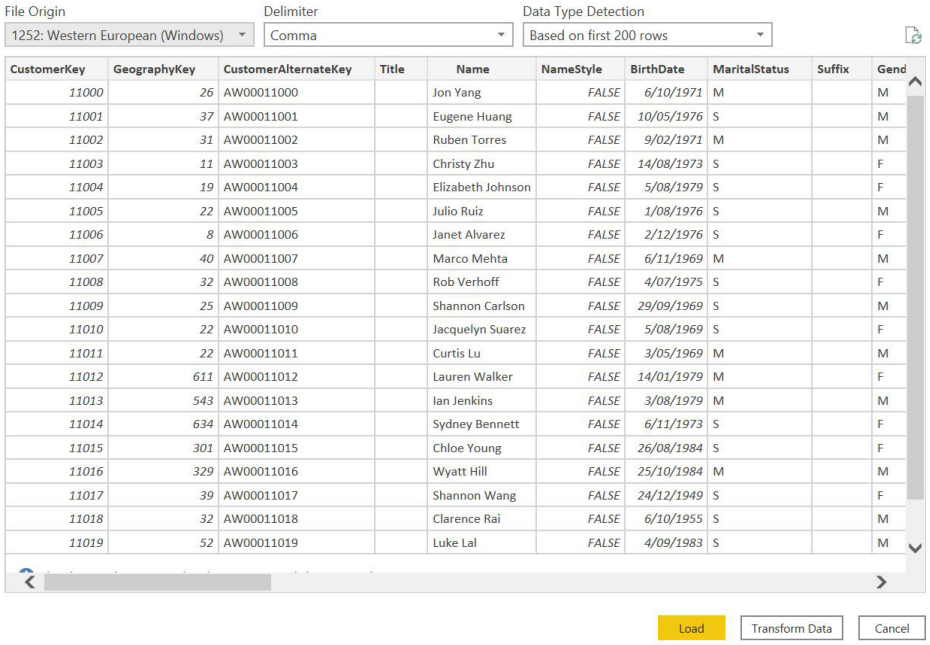


Figure 6: List of the sample files to be loaded into Power BI

- When you click **Open**, the window shown in Figure 7 below appears, displaying sample data from the selected file.



CustomerKey	GeographyKey	CustomerAlternateKey	Title	Name	NameStyle	BirthDate	MaritalStatus	Suffix	Gend
11000	26	AW00011000		Jon Yang	FALSE	6/10/1971	M		M
11001	37	AW00011001		Eugene Huang	FALSE	10/05/1976	S		M
11002	31	AW00011002		Ruben Torres	FALSE	9/02/1971	M		M
11003	11	AW00011003		Christy Zhu	FALSE	14/08/1973	S		F
11004	19	AW00011004		Elizabeth Johnson	FALSE	5/08/1979	S		F
11005	22	AW00011005		Julio Ruiz	FALSE	1/08/1976	S		M
11006	8	AW00011006		Janet Alvarez	FALSE	2/12/1976	S		F
11007	40	AW00011007		Marco Mehta	FALSE	6/11/1969	M		M
11008	32	AW00011008		Rob Verhoff	FALSE	4/07/1975	S		F
11009	25	AW00011009		Shannon Carlson	FALSE	29/09/1969	S		M
11010	22	AW00011010		Jacquelyn Suarez	FALSE	5/08/1969	S		F
11011	22	AW00011011		Curtis Lu	FALSE	3/05/1969	M		M
11012	611	AW00011012		Lauren Walker	FALSE	14/01/1979	M		F
11013	543	AW00011013		Ian Jenkins	FALSE	3/08/1979	M		M
11014	634	AW00011014		Sydney Bennett	FALSE	6/11/1973	S		F
11015	301	AW00011015		Chloe Young	FALSE	26/08/1984	S		F
11016	329	AW00011016		Wyatt Hill	FALSE	25/10/1984	M		M
11017	39	AW00011017		Shannon Wang	FALSE	24/12/1949	S		F
11018	32	AW00011018		Clarence Rai	FALSE	6/10/1955	S		M
11019	52	AW00011019		Luke Lal	FALSE	4/09/1983	S		M

Figure 7: Loading the data to be used

- We have two options, **Load** or **Transform Data**. We want to transform our data, so we click the **Transform Data** button.

Clicking Load will import the tables exactly as is. This is how we imported the CSV files in the Beginners Guide. It is still possible to transform the data later by clicking the Transform Data button from the Home tab in Power BI Desktop.

- Clicking **Transform Data** opens a separate window - the **Power Query Editor**: a powerful data profiling and data preparation tool.

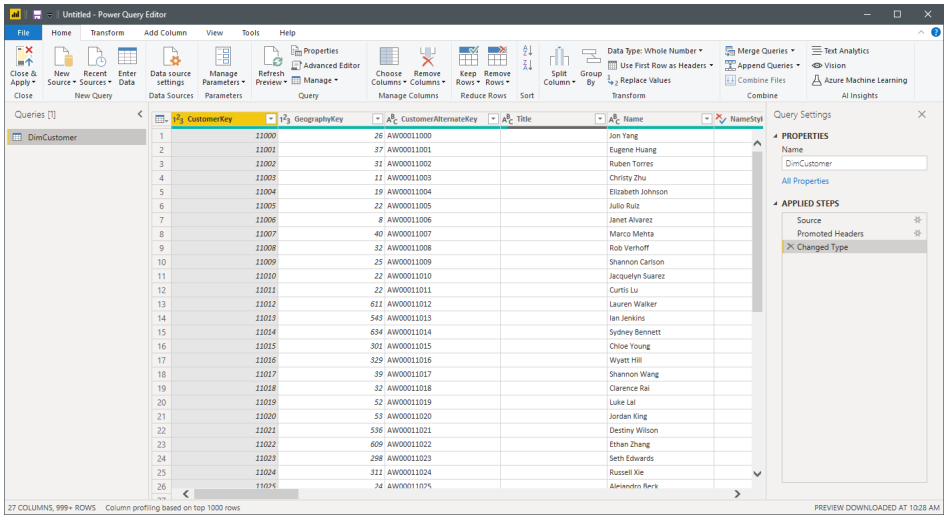


Figure 8: Power Query Editor

- Before we continue transforming data, we need to get the remaining data. We can get additional data directly in the Power Query Editor by clicking **New Source.**

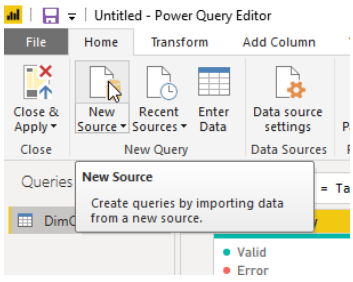


Figure 9: Adding new data sources

- Select **Excel** under All or File and click **Connect**.

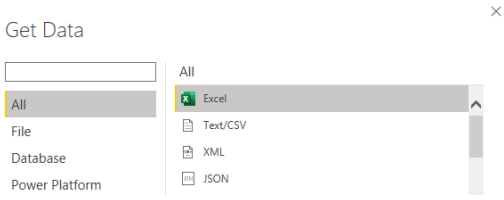


Figure 10: Getting data from Excel

- To import the data, browse to the unzipped data and select **AdventureWorks2017 .xlsx** then click **Open**.
- A dialog window opens listing the data sources within the Excel file. Click the checkboxes next to each source and click **OK**.

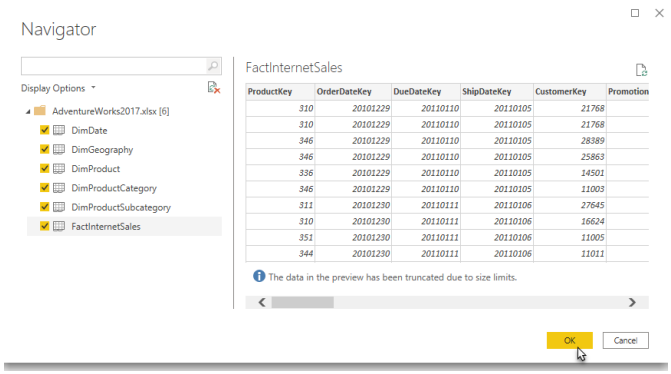


Figure 11: Selecting Excel work sheets

We're now connected to the data sources. It's time to profile and transform our data before creating our data model.

3. Data Preparation with Power Query

Let's take a moment to explore the Power Query interface.

The Application Ribbson contains all options and settings. Complete the steps:

1. Click the **View** tab from the ribbon. Make sure the following are ticked:
2. **Formula Bar**
3. **Column quality**
4. **Column distribution**

Note: If **show white space** is ticked, you can leave as is.

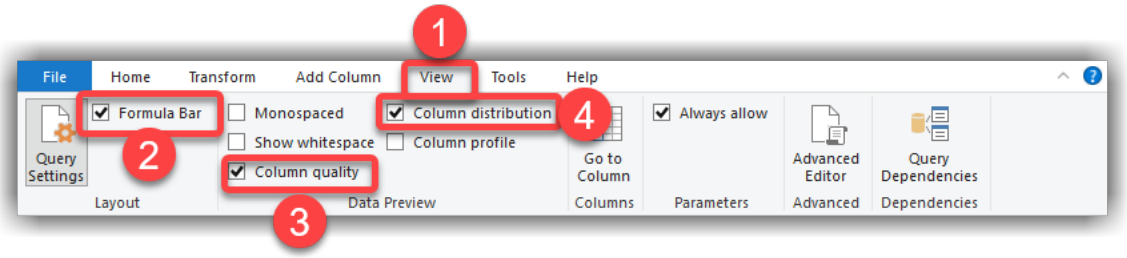


Figure 12: The Power BI Desktop application interface

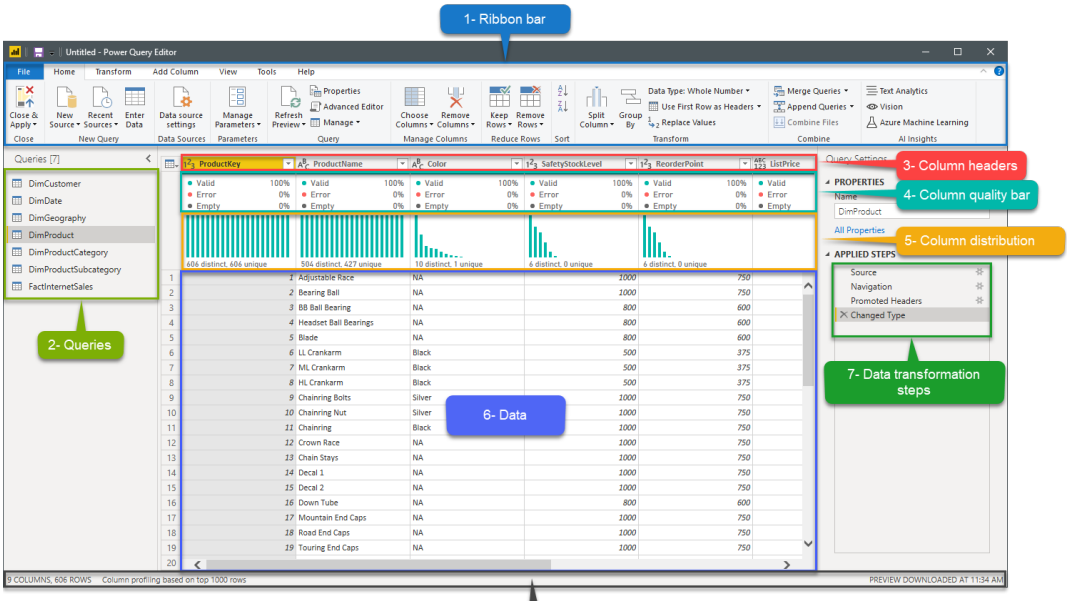


Figure 13: Enabling some layout and view items in Power Query Editor

As Figure 13 illustrates, the application ribbons contain all options and settings, transformations, and other settings configurations.

1. The left pane shows different **Queries**. These can be the tables you selected for import or custom functions, query parameters, or queries with constant values that you create in Power Query.
2. The **Column header** shows the column name and data type (Number, Date, Text, True/False etc).
3. The **Column quality bar** shows details on the number of valid, empty and error records in the data.
4. The **Column distribution bar** provides counts of distinct and unique values.
5. The **Data** displays the view of the data based on the data transformation step that has been selected.
6. The **Data Transformation Steps** shows a list of data transformations that have been applied to the data.
7. The **Status bar** shows the number of columns the selected query from the Queries pane have, number of rows and data preview refresh date and time.

3.1 Splitting Columns

Power Query allows us to split a column into one or more other columns. There are various options available under the **Split Column** dropdown.

We will be using By Delimiter to split the Name into First and Last Name.

- Go to the Queries pane on the left and select **DimCustomer**.
- Select the **Name** column.
- From the ribbon bar, go to the **Transform** tab, click **Split Column** and select **By Delimiter.**

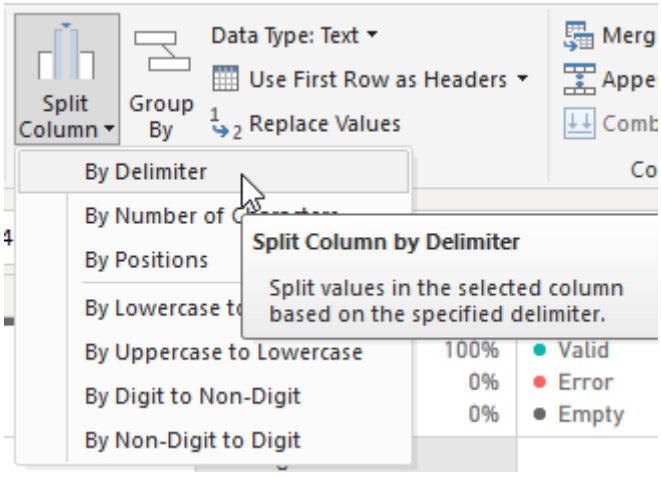


Figure 14: Split column by delimiter

- In the dialog box, ensure **Select or enter delimiter** is set to **Space**.
- Select **Each occurrence of the delimiter** and click **OK**.

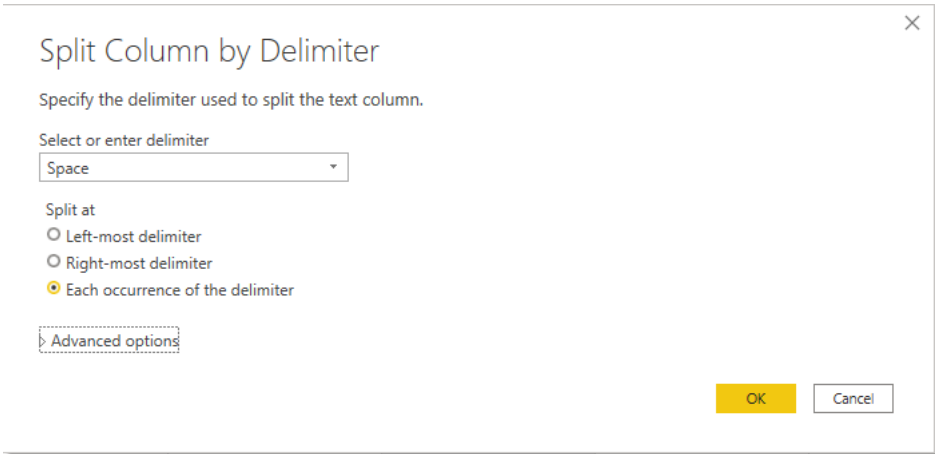


Figure 15: Selecting the delimiter

- The Name column is split into **Name.1** that has the First Name and **Name.2** that has the Last Name. The splitting column by delimiter action above creates a new applied step as shown in Figure 16 below:

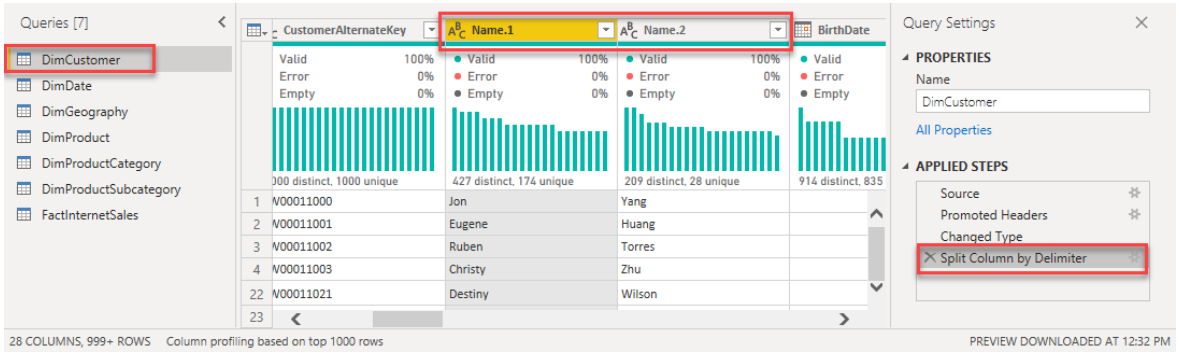


Figure 16: Column "Name" split to two columns

3.2 Renaming Columns

We will rename the two columns created by the splitting of the Name column. There are a couple of ways to do this: either double click the Header name or right-click the header and select rename.

- Double click the **Name.1** Header and rename it to **FirstName**.
- Right click the **Name.2** Header and rename it to **LastName**.

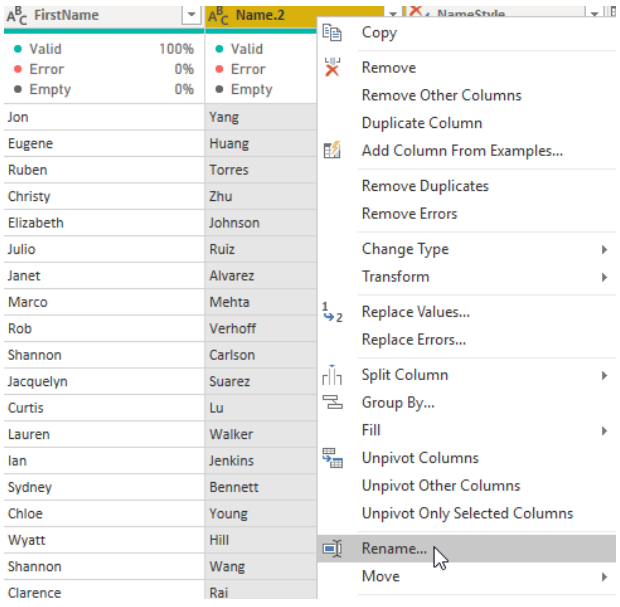


Figure 17: Renaming Columns

3.3 Adding Columns

Adding Columns is a common task and there are many ways to do this. There is a dedicated tab in the **Ribbon bar** for adding columns. In the next few steps, we use a **Conditional Column** and a **Column from Example** to add new columns.

In the steps below, we will add a new conditional column - **TotalChildrenNumber**.

1. Go to the Queries pane, select the **DimCustomer** query.
2. Go to the **Ribbon bar**, select **Add Column**, select **Conditional Column.**
3. In the dialog box, change column name from **Custom** to **TotalChildrenNumber**.
4. Modify the If statement to read: If **TotalChildren equals one** Then **1**.
5. Click **Add Clause**.
6. Add the additional 'if' clauses for the remaining values (the values in this dataset goes up to five. Remember that **one** is the only value that needs to be in all lowercase!). *
7. Enter 0 for **Else,** this will assign 0 to any remaining records including those with the value 'None'.
8. Click **OK** and our new column will be added.

*Important: Note that Power Query is CAsE sEnSiTive. In this dataset, you'll need to write 'one' in **all** lowercase. For Two, Three, Four and Five, the first letter will be uppercase.

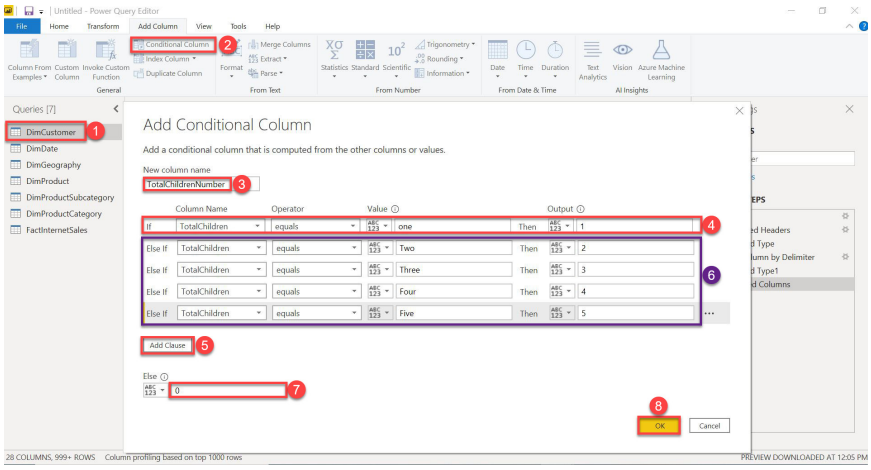


Figure 18: Adding a conditional column

Remember to save your work as you go.

In the next steps, we'll add an Annual Income column:

1. From the Queries pane, select the **DimCustomer** query.
2. Go to the **Add Column** tab, select the **Column from Example** dropdown
3. Select **From Selection.** This adds a new column called **Column1** at the end of the columns list.
4. Select the **YearlyIncome** column.

5. In the first row in **Column1**, type in the same number as is shown on the first row of **YearlyIncome**. (Unless the table has been sorted differently, you should type in 90000, note you don't need to type USD just the numbers.).
6. Press **Enter** and all the rows will be updated. You may need to enter more rows for the rest of the column to populate.
7. **Rename Column1** to **AnnualIncome** (double click the column title to rename, note that in some versions of PowerBI your column maybe be renamed "**YearlyIncome-Copy**").
8. Click **OK**

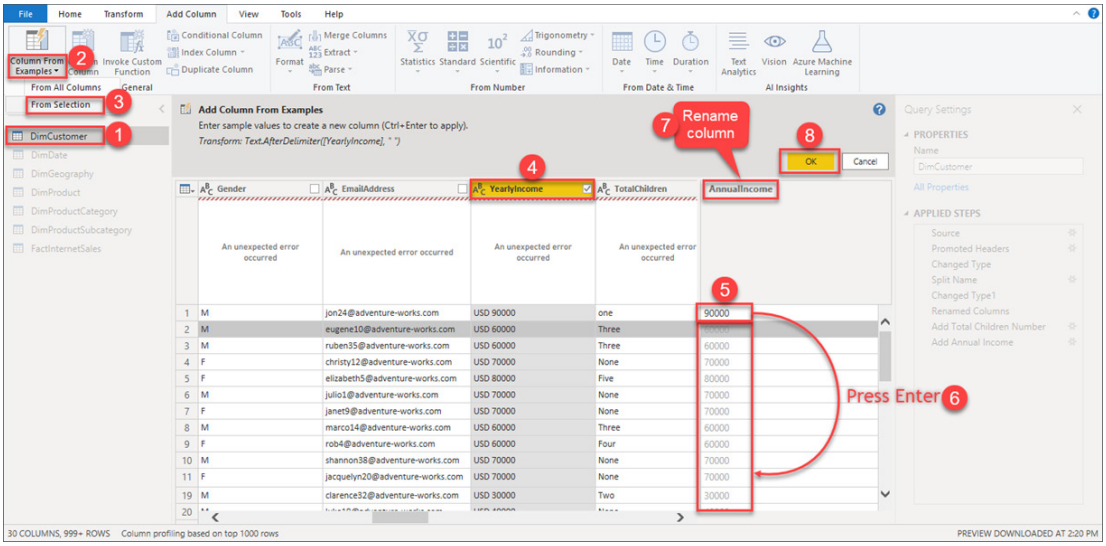


Figure 19: Adding column from example

3.4 Removing Columns

We can remove columns that we no longer need e.g. duplicate columns. This also applies to columns that are used to add new columns such as the **TotalChildren** and **YearlyIncome**.

You can remove a column that has previously been used to create a new column. It does not impact on the data in the new column.

- Select the **DimCustomer** query from the Queries panel.
- Hold Ctrl and click the column headers for **YearlyIncome**, **TotalChildren**, **SpanishEducation, FrenchEducation, SpanishOccupation and FrenchOccupation.**
- Right click in any of the selected column headers and click **Remove Columns.**

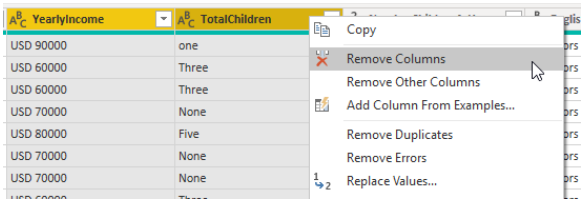


Figure 20: Removing columns in Power Query Editor

3.5 Changing Data Types

Each column in Power BI has a data type. When connecting to a data source, Power Query loads 1,000 rows of data as sample data and automatically detects the data types. While in many cases the detected data types are correct, in some cases we may face data type conversion errors when the detected data type is incorrect. Therefore, it is advised to always review the data types before continuing with other transformation steps. For instance, storing a customer number as a number where it should probably be stored as a text field.

The most common data types and their column icon in Power BI are:

- Text e.g. Bicycle
- True/False
- Whole Numbers e.g. 148
- Decimal Numbers e.g. 3.5432
- Fixed Decimal Number e.g. 5.45
- Percentage e.g. 46.8%
- Date e.g. 15/02/2020
- Time e.g. 12:35:00 PM
- Date/Time e.g. 15/02/2020 10:25:00 AM

In the steps below, you'll make a correction in the automatically detected data types:

- Select the **DimCustomer** query from the Queries pane.
- Go to the **TotalChildrenNumber** column and click the data type icon.
- Select **Whole Number** from the context menu.
- The data type icon changes to the Whole Number icon.
- Click the data type icon next to the **AnnualIncome** column.
- Select **Fixed Decimal Number.** This is similar to currency.
- The data type icon changes to the **Fixed Decimal Number** icon.
- Click the data type icon next to the **HouseOwnerFlag** column.
- Select **True/False.**
- The data type icon changes to the **True/False** icon. The 0 values have been set to False and 1 has been changed to True.

Figure 21 shows changing the data type of the **TotalChildrenNumber.**

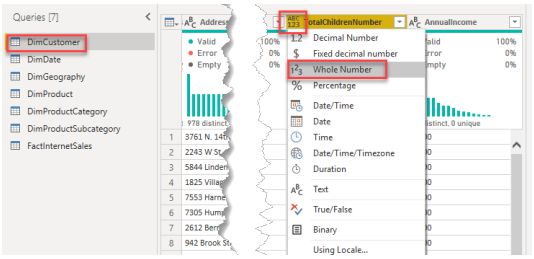


Figure 21: Changing column data types

3.6 Transformation Steps

All transformations are applied as a series of steps shown in the **Query Settings** pane as shown in the Figure 22 below.

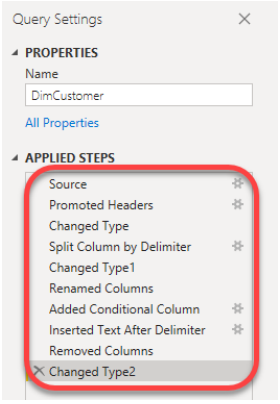


Figure 22: Transformation steps

It's possible to see the resulting data of each step by clicking on a desired step from the **Query Settings** pane as shown in Figure 23 below. This makes it easy to visually inspect the data after applying each transformation step.

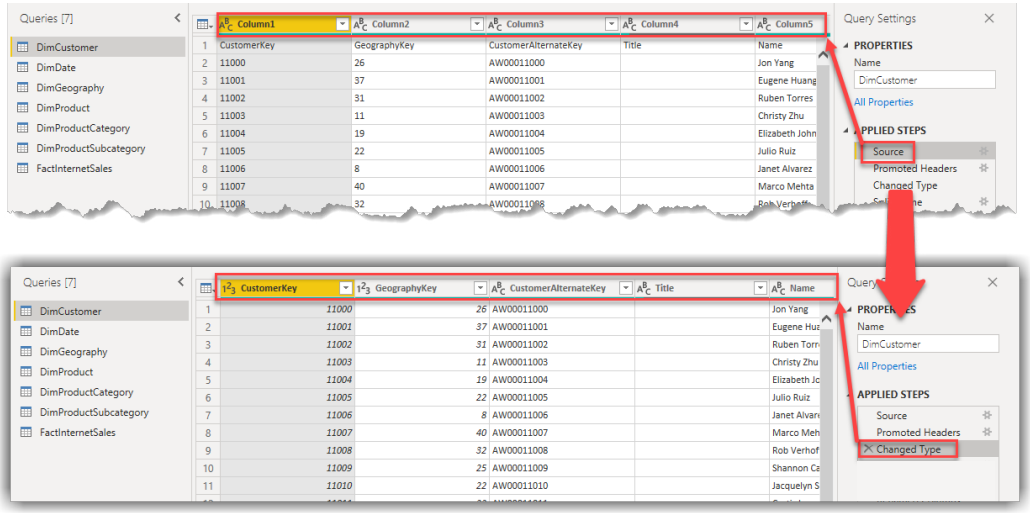


Figure 23: Illustrating the resulting changes of each step in the data

Transformations can be added, edited, removed, renamed, or reordered by right clicking a desired step from the **Applied Steps** selector in the **Queries Settings** as shown in Figure 24.

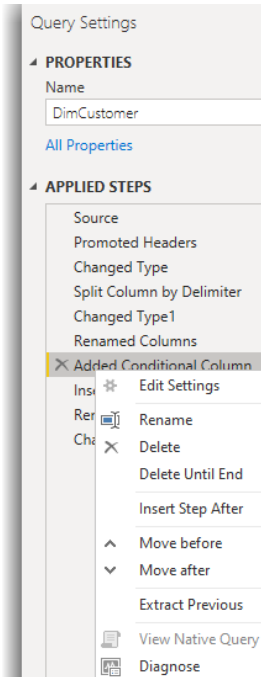


Figure 24: Renaming, deleting, inserting new step and moving steps

Depending on the data source type, the following transformation steps may already exist for each query:

1. **Source**
   The very first step of the query is normally connected to the data source. In this tutorial, the sources will be a Text/CSV source and the Excel Source containing the other tables.

2. **Navigation**
   If the data source we are connecting to has different objects containing data, such as an Excel workbook that can contain many worksheets and/or tables, then by selecting each object from the source system we are navigating through that object to get the containing data. For each Query, this will be the corresponding worksheet or table name in the Excel File, while Text/CSV files do not have a **Navigation** step as each file contains only one set of data.

3.6.1 Renaming an applied step

The existing steps can be renamed within the Power Query Editor. This shows what the step did and makes future modification easier. It's highly recommended to rename the steps to something more meaningful.

- Select the **DimCustomer** query in the Queries panel.
- Right click the **Split Column by Delimiter** step.
- Click **Rename** from the context menu and change to **Split Name.**
- Right click the **Added Conditional Column** step.
- Click **Rename** and change to **Add Total Children Number.**

- Right click the **Inserted Text After Delimeter** step.
- Click **Rename.**
- Type **Add Annual Income.**

Figure 25 shows the renamed steps (and the rest of steps we left as is). We've left those steps for you to rename.

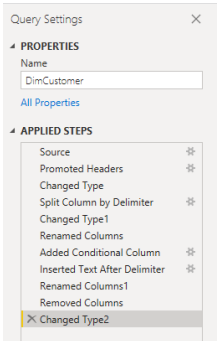


Figure 25: Renaming steps

3.6.2 Viewing transformation step changes

Select the **DimCustomer** query in the Queries pane. The **Applied Steps** pane shows all steps in the order that they are applied:

1. Click **Source** to see what the original data looked like. You should see the single **Name** column.
2. Click **Split Name** to see the changes made by this step. The Name.1 and Name.2 columns should be visible.
3. Click **Renamed Columns** in the Applied Steps panel and you should see the columns renamed to **FirstName** and **LastName**.

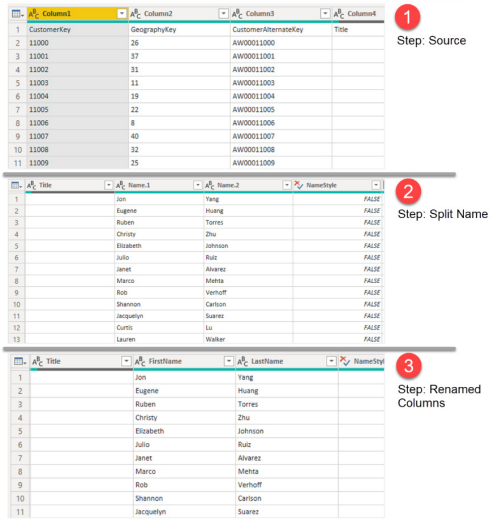


Figure 26: Viewing how transformation steps change the data

3.6.3 Adding a step between an applied step

The steps added earlier to remove columns and change types resulted in new steps being added to the end of the applied steps. The results of the last step contain the data that will be imported into the data model and available for visualisation. You can add a step in between existing steps. Let's add a step that renames **TotalChildrenNumber** to **TotalChildren,** but we'll add it after the existing **TotalChildren** column has been removed.

1. Select the **DimCustomer** query in the Queries pane.
2. Go to the **Applied Steps** pane and click **Removed Columns**.
3. Right click **TotalChildrenNumber**
4. Select **Rename**
5. Type **TotalChildren** and press enter
6. A popup will ask if you are sure you want to insert the step. Click **Insert.**

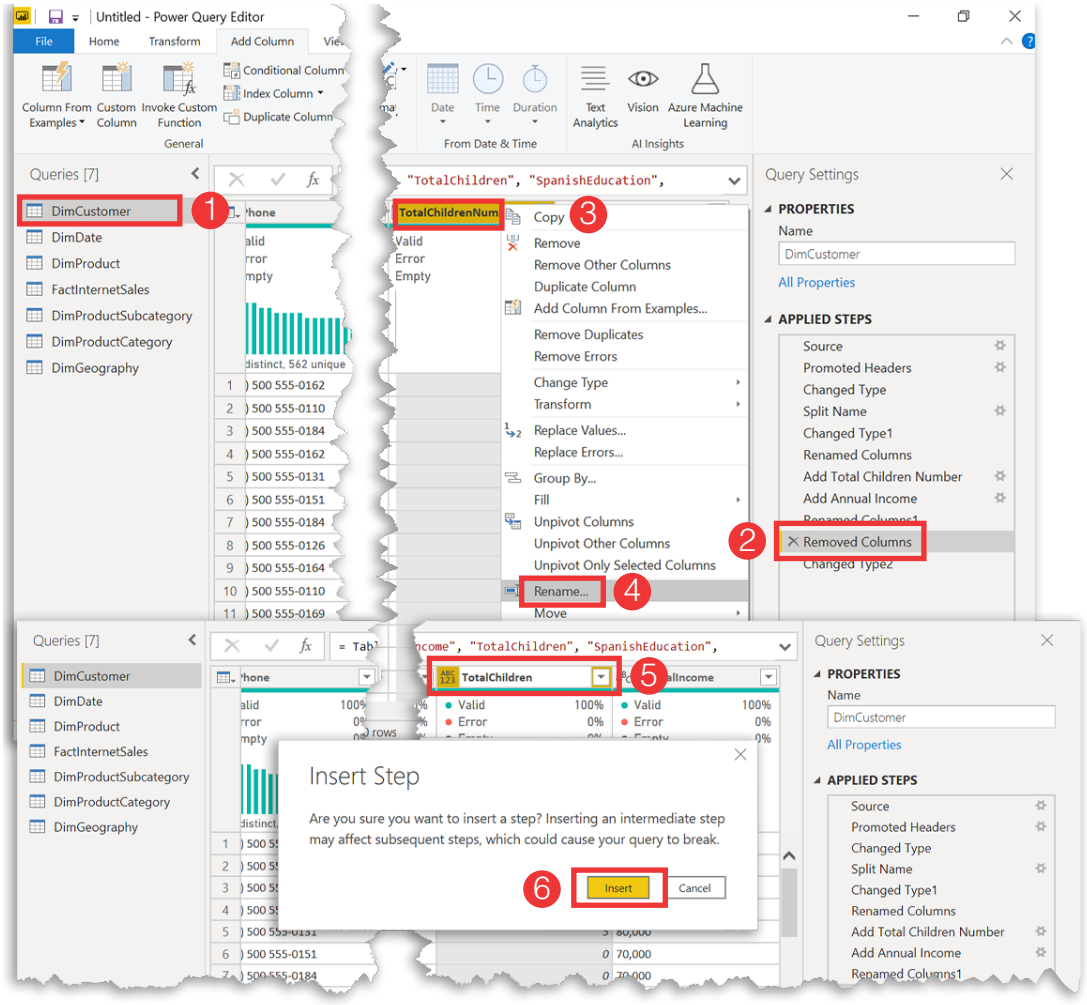


Figure 27: Inserting a step between existing steps in the Power Query Editor

You should see a step called **Renamed Columns2** is added between Removed Columns and Changed Type2. Don't worry if you see an error, we are about to fix that!

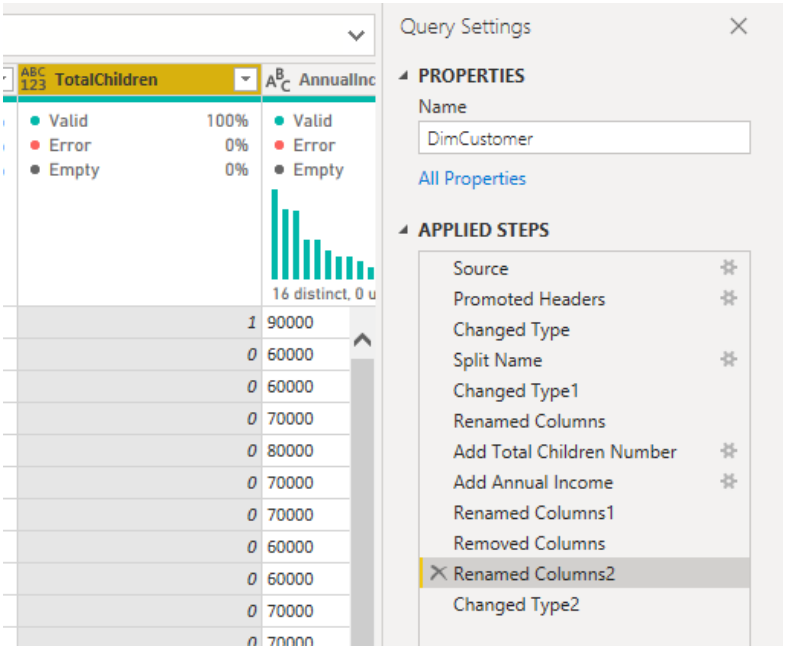


Figure 28: A new step is added between existing steps

3.6.4 Reordering applied steps

You can change the order of the steps by dragging and dropping a step to another location.

Be careful when reordering steps as it may have a negative impact on future steps that depend on the moved step.

- Select the **DimCustomer** query in the Queries panel.
- Click and drag **Add Annual Income** and drop it above **Add Total Children Number.** This step will now take place before the **Add Total Children Number** step.

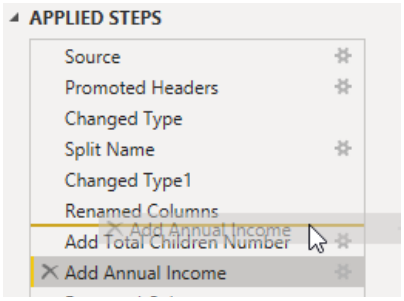


Figure 29: Moving existing steps up or down

3.6.5 Editing an existing step

All existing steps can be modified. If there's a **settings** icon to the right of the name, you can edit the step from the UI. Otherwise, you can modify the existing steps from the code - either by opening the "Advanced Editor" or from the "Formula bar" (if enabled). Before we look at editing existing steps, let's see how we open the "Advanced Editor" and how we can enable the "Formula bar".

- You can access the "Advanced Editor" from various places. You can right click a query then click "Advanced Editor". You can alternatively click the "Advanced Editor" button from the "Home" tab.

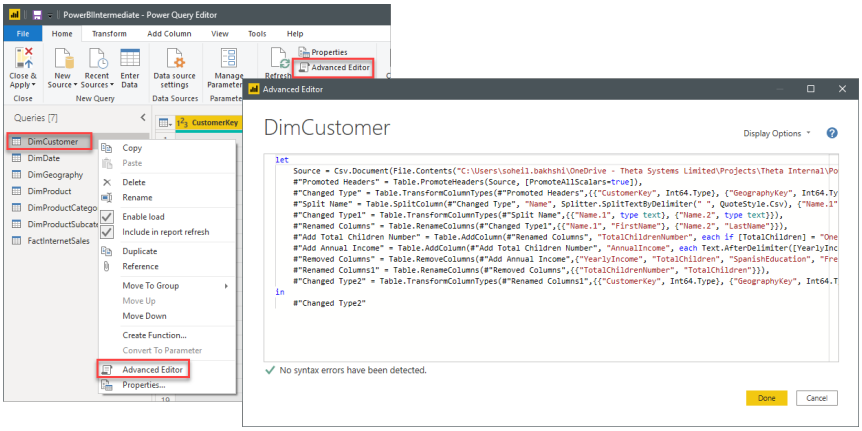


Figure 30: Opening "Advanced Editor"

- Enabling **Formula bar** is easy. It's beneficial as it can help you to understand and learn how Power Query works. Clicking each step allows you to see and modify the Power Query expressions.

So, for those columns that cannot be edited, using the **Formula bar** is handy. To enable the **Formula bar**, click the corresponding tick box from the **View** tab as shown in the Figure 31:

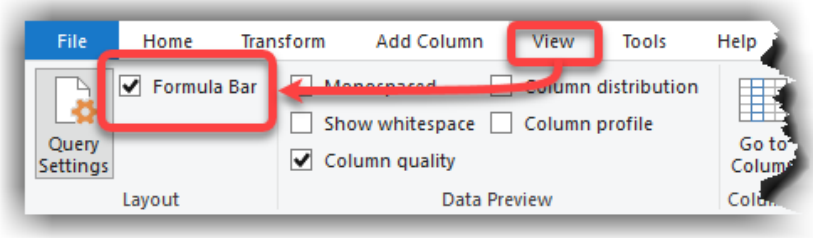


Figure 31: Enabling Formula bar from the Power Query Editor

Note: In section 3.6.4, we renamed the TotalChildrenNumber column to TotalChildren. This breaks the Changed Type2 step. You'll see an error message ⚠ on the Queries Panel to the left. To fix this error, we have to:

- Click the Changed Type2 step.
- From the formula bar, replace the words **TotalChildrenNumber** with **TotalChildren**.
- Click the Commit button as shown in Figure 32, in some versions of PowerBI this may happen automatically.

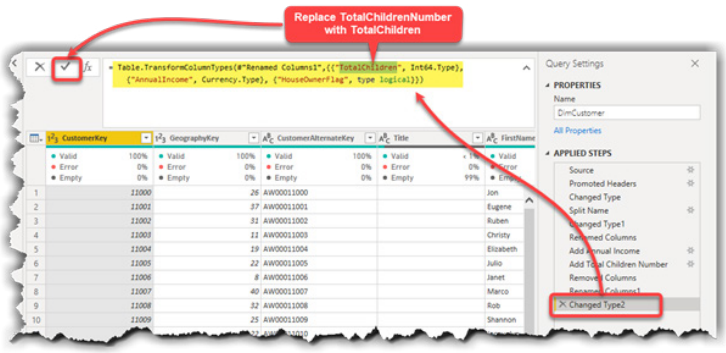


Figure 32: Edit steps from the UI (when applicable)

3.6.6 Deleting an applied step

All steps can be deleted except the source steps. You can either delete just the selected step or all steps from that one until the end.

Be careful when deleting steps. It may have a negative impact on future steps that depend on the deleted step. Power BI will give you a warning when you attempt to delete any steps.

In this instance, you don't need to delete any steps.

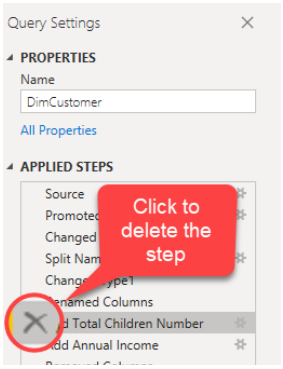


Figure 33: Deleting the steps

In this chapter, you learnt some data preparation steps like splitting columns, renaming columns, adding new columns, changing data types and so on. Once you've completed the required transformation steps, you can load the data into the data model. In the next chapter, we look at the basics of data modelling.

4. Modelling the Data

Data modelling is one of the most important aspects of data analysis, regardless of the tools we use, and Power BI is not an exception. After we import data into the data model, we need to create and manage relationships, creating analytical calculations and implement the business logics available for data visualisation. Here are the basics of data modelling.

4.1 Relationships in Data Modelling

Relationships between tables are necessary to accurately calculate the results and visualise the correct information in the report. When we create a relationship between two tables, we are creating a linkage between the data stored in those tables in one of the following ways. Let's imagine we have two tables, Table X and Table Y:

- A row of data from Table X is related to one and only one row of Table Y. This type of relationship is called a **One-to-one** relationship. Like when we stored the personal details of a customer in a table called **Customer** and we stored the customers' address in a table called **Address** and a customer has only one address.
- A row of data from Table X is related to many rows from Table Y. This type of relationship is called a **One-to-many** relationship. In our previous example, a customer can have many sales orders, so if we stored the sales orders in a **Sales** table, then each customer from the **Customer** table relates to many rows of sales data from the Sales table.
- A row or data from Table X is related to many rows of data from Table Y AND a row of data from Table Y is related to many rows of data from Table X. This type of relationship is called a **Many-to-many** relationship.

Now that you know about the different types of relationships in Power BI, let's see how to identify the columns contributing to a relationship.

4.1.1 Identifying Key Columns in Power Query

To be able to create a relationship, we firstly need to identify the key columns in both related tables. There are two types of key columns, Primary Key and Foreign Key.

4.1.1.1 Primary Key

In data modelling, we refer to tables as **Entities**. A table consist of columns holding the data. Each column describes an **Attribute** of an **Entity.**

The columns (**attributes**) hold the data that describe **Records** of columns (**attributes**).

Each **record** of data in a table (**entity**) is a row.

For instance, in a **Product** table (**entity**), the columns (**attributes**) of the table contain the product data. Each row (**record**) of data describes a single product.

In data modelling, we normally have a column or a combination of columns who can describe a unique row, hence we do not need to mention all columns to describe a single unique row (**record**) of data. The column or columns that describe a unique row are so called **Primary Key** columns. The primary keys consist of combination of columns are so called **Composite Keys**.

In Power BI, Composite Keys are NOT supported, therefore we must have one primary key column only. It is NOT mandatory for all tables to have a primary key column. At this stage you may ask “well, how can we identify the primary keys then?”. The answer is that it depends. In many cases, the tables in the source system already contain a column that has a **Key** or **ID** in the column name, especially if the source system is a relational database like SQL Server, Oracle etc. But there are still many other cases that you do not have that option, therefore, you may need to find the primary key columns yourself. Let’s have a look at an example and find the primary key of the Customer table from the sample file provided with this guide.

Follow the steps below:

- If you previously closed the Power Query Editor and you are back to the Power BI Desktop window, go to the **Home** tab and select the **Transform data** button to open the **Power Query Editor**. Otherwise jump to the next step.
- Go to the **View** tab and make sure that **Column Distribution** is ticked. This will show a new box on top of each column below the column title. This box is called **Column Distribution Box**.
- Look at the **Column Distribution Box** to find a column that has the most unique and distinct values. Hover-over each column’s distribution box and look at the **Distinct %** and **Unique %** values in Figure 34 below:

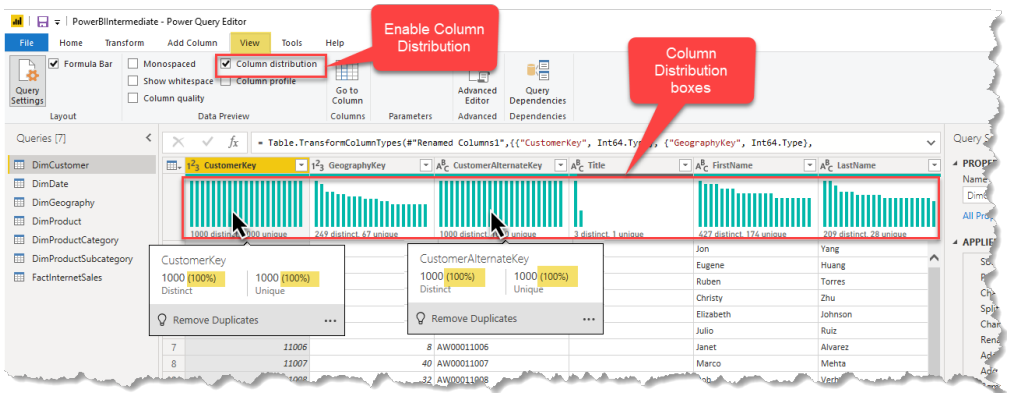


Figure 34: Identifying Primary Key using Column Distribution feature in Power Query

As you see in Figure 34, there are currently two columns that can be primary keys for the **DimCustomer** table. In cases like this, we need to have a good level of understanding of the business to be able to pick one of those columns as the primary key column. In this case, we already know that the **CustomerKey** column is indeed the primary key of the **DimCustomer** table. Therefore, we will use the **CustomerKey** column when we want to create a relationship between this table and other tables in the data model.

Important Notes:

- The method mentioned above is just to identify the Primary Key column of the tables.
- The **Column Distribution** information is based on top 1,000 rows of sample of data as shown in the status bar in the Power Query Editor window in Figure 35. If we have more than 1,000 rows of data in the underlying table, which in our example we do, the **Column Distribution** information is not very accurate.
- To make the **Column Distribution** information more accurate, we have to change the column profiling data to be calculated based on the entire dataset as shown in Figure 35. Select this by clicking on the footer.

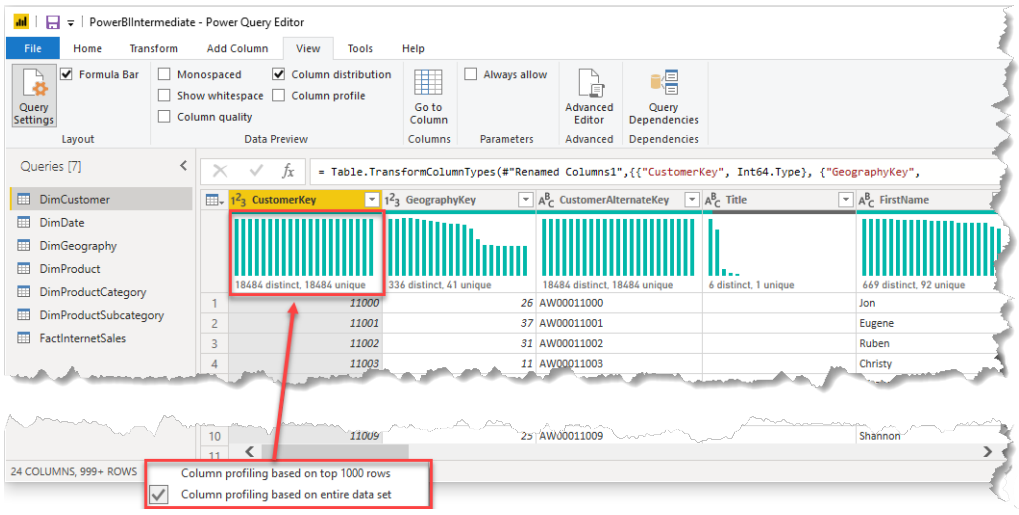


Figure 35: Changing column profiling sampling data

4.1.1.2 Foreign Key

So far you learned what **Primary Keys** are. When we have a **Primary Key** of a table in another table, then that column is called **Foreign Key**. It is NOT necessary that the foreign keys contain unique values.

You've now finished using Power Query. You need to save the data that you have worked on.

- Save what you have done in Power Query. Go to **file** and **save as** 'ADW-Part1'
- **Apply all changes**
- Exit out of Power BI

When we resume in part 2, you will need to use the new file that you have saved 'ADW-Part1'.

If you're jumping straight into part 2 of the guide, or you were unable to complete part 1, you should use the supplied file 'ADW-Part1-Theta'. This should match the one that you've just created (providing all the steps in this guide were completed successfully).

Introduction - Part B

In Part B of the guide, we jump back into Power BI Desktop.

Some notes before we get started:

- You'll see that we've highlighted expressions in grey boxes. You can copy/paste these, but we'd recommend retrieving them from GitHub as it preserves the correct formatting. Access the expressions in GitHub here.
- If you're copy/pasting expressions, you must ensure that the code copies straight speech marks " as opposed to curly speech marks ". Curly speech marks in the code will not work!
- Part B explains more concepts around Power BI. To make it easier to follow, you can jump to the yellow highlighted sections; these show the instructional steps to take.
- Part B uses the dataset from where you finished in part A. If you need to access the completed dataset from Part A (e.g. if you're starting from Part B or you had issues completing part A), use the supplied file ADW-Part1-Theta.

Part B

You'll need:

1. Saved dataset from part A (ADW-Part 1)
   OR Theta's supplied .pbix file (ADW-Part1-Theta)
2. Access to the GitHub repository

How are you getting on? If you need a bit more 'hands-on' help, try our **Power BI team training**.

For other data and insights related training, take a look at our **Data Accelerate workshops**.

5. Understanding relationships in Power BI

We always create relationships between the **Primary Key** of a table and its corresponding **Foreign Key** in another table. Let's look at our imaginary tables - **Product** and **Sales** - to get a better understanding of the relationships.

Product	
ProductKey	Product Name
1	Laptop
2	Mobile
3	Mouse

Sales			
ProductKey	Sales Amount $	Sales Date	CustomerKey
1	2,000	7/01/2020	1
1	1,500	8/01/2020	2
1	980	9/01/2020	2
2	550	7/03/2020	2
3	50	7/02/2020	3
3	65	6/01/2020	2
2	890	12/03/2020	1
1	3,500	22/02/2020	1

Figure 36: The structure of Product and Sales tables

In Figure 36 above, the **ProductKey** column is the **Primary Key** of the **Product** table and a Foreign Key in the **Sales** table.

We create a relationship between **Product** and **Sales** using the **ProductKey** column in both tables. As you can see in Figure 36, every single value of **ProductKey** from the **Product** table has many corresponding values in the **Sales** table; the relationship between the two tables is a **One-to-many** relationship. Creating a relationship between the two tables will look like Figure 2: the (1) resembles the **One** side of the relationship and the (*) resembles the **Many** side of that relationship.

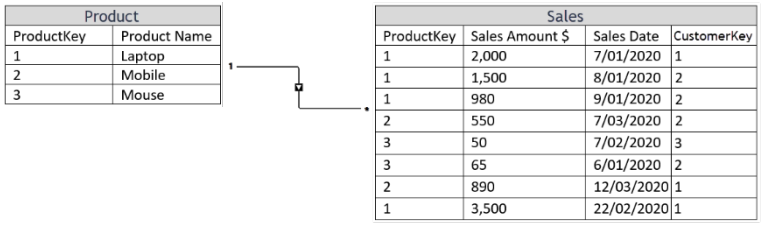


Figure 37: One-to-many relationship between Product and Sales tables

Let's add another set of tables to our existing set of imaginary tables, **Customer** and **Customer Address**:

Customer	
CustomerKey	Name
1	John Doe
2	Jane Doe
3	Joe Simpson

Customer Address	
CustomerKey	Address
1	#1 Street 1
2	#2 Street 1
3	#1 Street 3

Figure 38: The structure of Customer and Customer Address tables

As Figure 39 illustrates, each customer from the Customer table has one and **only one** corresponding address in the Customer Address table. Therefore, the relationship between the two tables is a One-to-one relationship.

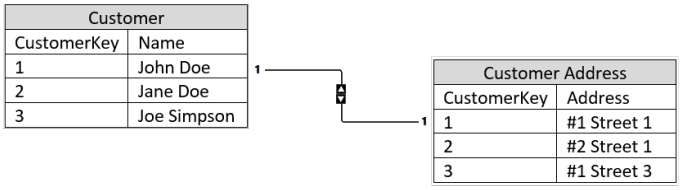


Figure 39: One-to-one relationship between Customer and Customer Address tables

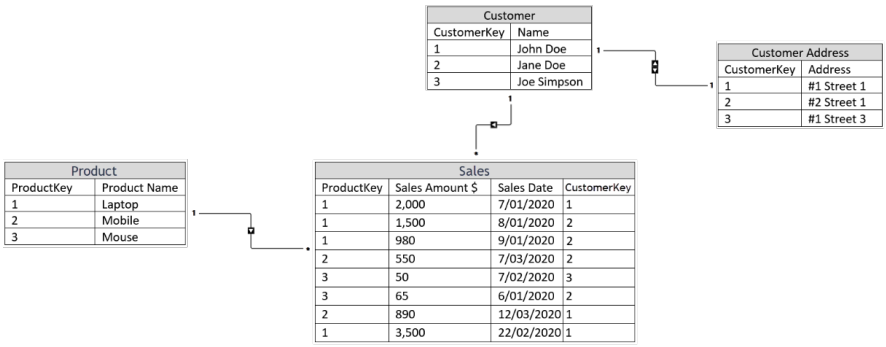


Figure 40: Creating one-to-many relationship between Sales and Customer

5.1 Creating Relationships in Power BI

Power BI is not only a reporting tool. It's an analytical tool that you can create data models with. The data model in Power BI includes tables and their relationships. There's a specific tab in Power BI called the Model View - placed in the left pane of the Power BI Desktop as shown in Figure 41.

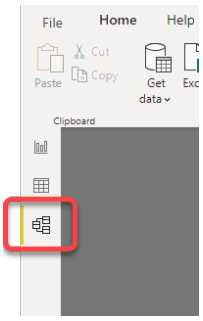


Figure 41: The Model view tab in the laft pane in Power BI Desktop

So far, you've learnt about Primary Keys, Foreign Keys and relationships. Power BI can also create the relationships between tables. However, it sometimes detects incorrect relationships; we must review the automatically generated relationships from the Model view. Follow these steps:

- Go to **Power BI Desktop.**
- Go to **file, open report** and select the file that you saved at the end of part 1 of the guide. 'ADW-Part1.pbix' (or access our supplied file ADW-Part1-Theta).
- Go to the **Model** view. Some relationships have been automatically created between a few tables. Note that your data model may be arranged differently.

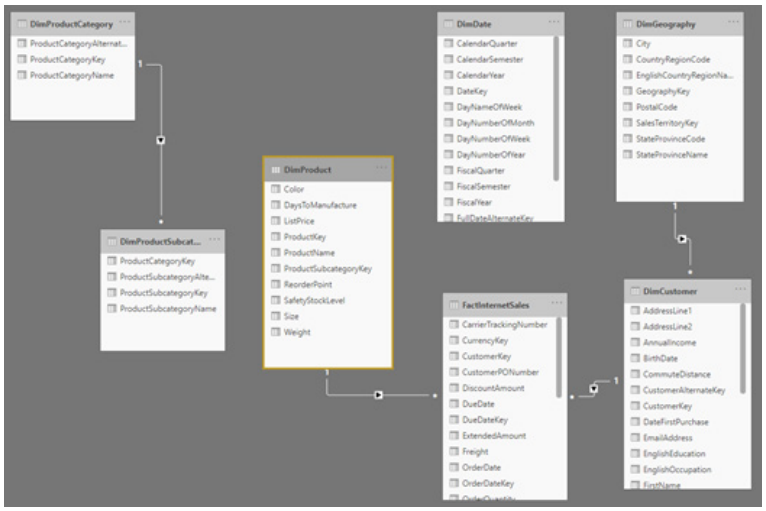


Figure 42: Power BI automatically detected some relationships

The arrows on the relationships indicate which direction filtering will occur. In Figure 43, if we put a filter on a value of a column from the DimCustomer, the FactInternetSales will only show records related to the selected value.

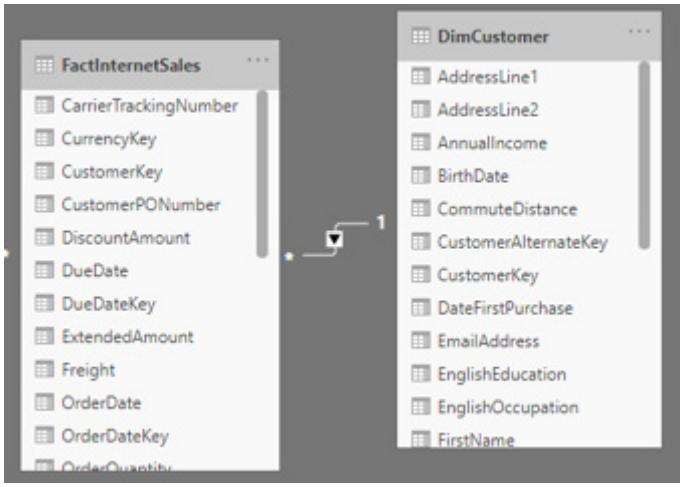


Figure 43: The direction of filtering from DimCustomer to FactInternetSales

Figure 42 showed that not all relationships have been automatically detected by Power BI; we'll need to create the rest of them manually. Follow the steps:

1. Click the **Manage Relationships** button either from:

A. The Modelling tab from ribbon bar

B. Or the Model view tab from the left pane

C. Or the Data view tab from the left pane under the Table tools tab from the ribbon bar (Figure 44).

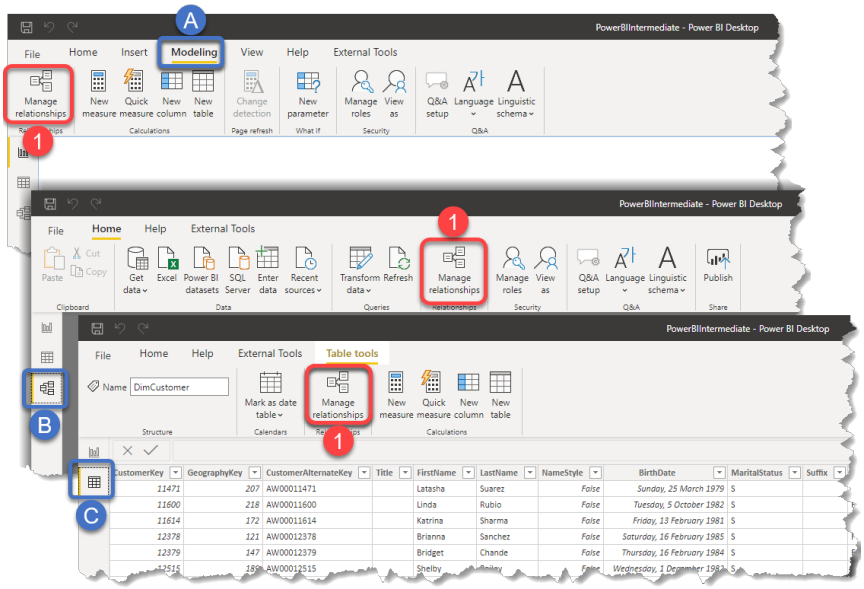


Figure 44: Accessing the Manage Relationships button in Power BI Desktop

2. Select the **New…** button from the **Manage relationships** window.
3. Select **DimProduct** from the first dropdown box.
4. Highlight the column **ProductSubCategoryKey.**
5. Select **DimProductSubCategory** table from the second dropdown box.
6. Highlight the column **ProductSubCategoryKey.**
7. Ensure the cardinality is set to **Many to One (*:1).**
8. Leave the Cross filter direction to **Single.**
9. Make sure the **Make this relationship active** is ticked.
10. Click **OK,** then close the window.

Figure 45 shows these steps.

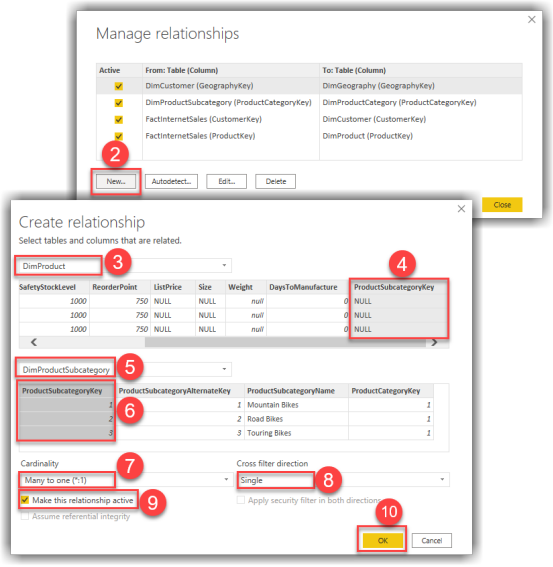


Figure 45: Adding a new relationship

We'll create the final relationship in an alternative way. Follow this step:

1. Drag the **OrderDateKey** from the **FactInternetSales** table and drop it over the **DateKey Column** of the **DimDate** table.

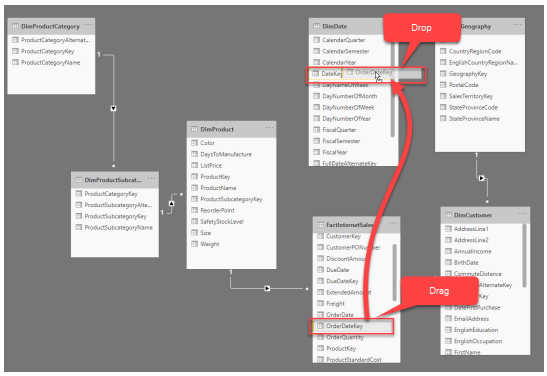


Figure 46: Creating relationship from the Model view by dragging and dropping a key column

After creating all relationships, your model should look like Figure 47.

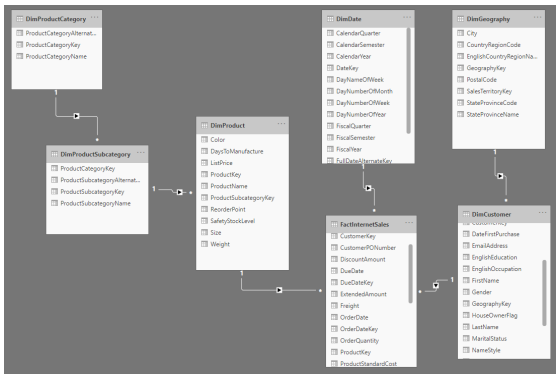


Figure 47: The Model view after creating relationships

5.2 Creating Calculated Columns and Measures with DAX

A data model consists of Tables and their Relationships. There are also other elements included in the data modelling:

1. Calculated Tables
2. Calculated Columns
3. Measures

All these can be created programmatically using DAX. In the next few sections, we look at them in more detail.

5.2.1 Calculated Tables

On some occasions, you need to add new tables based on data you've already loaded into the model. These tables can be created using DAX.

You can also use table constructor in DAX to create a calculated table. Table constructor isn't a function; it's a set of characters that allow you to create a table in DAX. For example, the following DAX expression creates a table with one column (Figure 13).

```
Calculated Table 1 =
  {"A","B","C"}
```

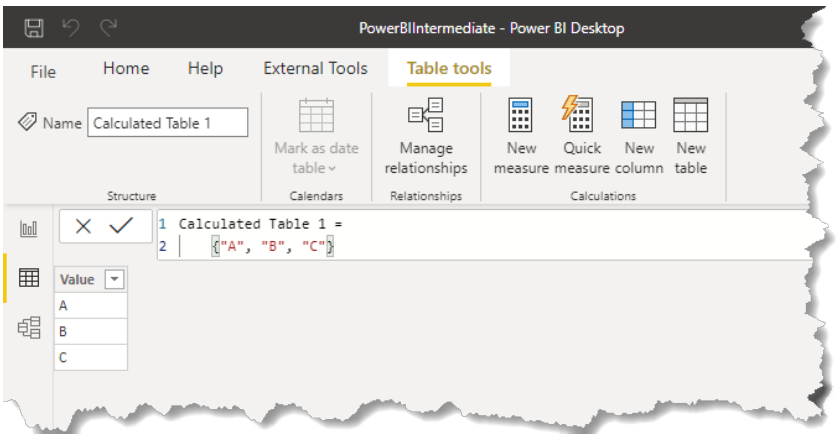


Figure 48: Creating a calculated table with one column using table constructor

You can use curly brackets {} to construct a table. This is a simple form of using table constructor in DAX. Here's how it works:

- Start of table construct - open curly bracket {
- "A", "B" and "C" are the values of a single column
- End of table construct - close curly bracket }

You can use any scalar DAX expressions in the values. The example below creates another calculated table using the table constructor and the DATE() function:

```
Calculated Table 2 =
  {DATE(2020, 7, 22), DATE(2020, 7, 23), DATE(2020, 8, 2)}
```

Figure 49 shows the results of running the above DAX expression (you don't need to complete this as a step).

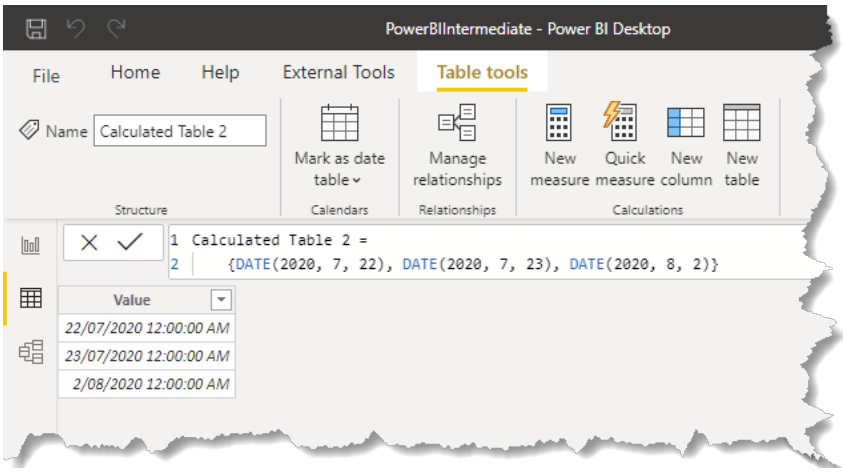


Figure 49: Creating calculated table with DAX table constructor using DATE() function

The following DAX expression is another structure of table constructor: when you define rows of data records using parentheses () where the values are separated with comma. As you see in Figure 50, you can use constant values or scalar expressions to define the rows.

```
Calculated Table 3 =
{
    ("A", 1.5, DATE(2017, 1, 1), CURRENCY(199.99) ),
    ("B", 2.5, DATE(2017, 1, 2), CURRENCY(249.99) ),
    ("C", 3.5, DATE(2017, 1, 3), CURRENCY(299.99) )
  }
```

Figure 50 shows the results of running the above expression in Power BI:

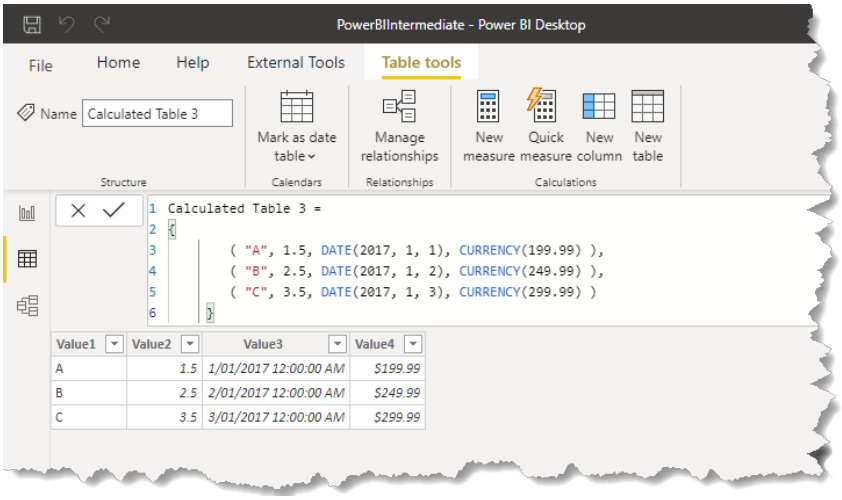


Figure 50: Creating a calculated table with DAX constructors using constant values or scalar expressions

As above, when defining the rows in the table constructor, the default column names are specified as Value1, Value2, Value3,…

We previously stated that we can also use any DAX functions that result in a table value. There are many use cases for that. For example, if you want to create a calculated table to show:

- The customers' full name - which is the concatenation of their first name and last name.
- Their total sales.
- Where the country region is Australia.

To create a calculated table, click the New table button from the Modelling tab from the ribbon (**You need to have the data view tab highlighted in the leftpane**). Then use the following DAX expression in the expression box. These steps shown in Figure 52:

```
Australian Customers Sales =
SUMMARIZECOLUMNS (
   DimCustomer[CustomerKey]
   , FILTER(
      VALUES(DimGeography[EnglishCountryRegionName])
         , DimGeography[EnglishCountryRegionName] = "Australia")
   , "Customer Full Name", CONCATENATEX(DimCustomer, DimCustomer[FirstName] & " " &
DimCustomer[LastName])
   , "Sales Amount", SUM(FactInternetSales[SalesAmount])
)
```

Figure 52 shows the results of running the preceding expression.

In Figure 51, the table has only 3,591 rows while the original DimCustomer table has more than 18,000 rows.

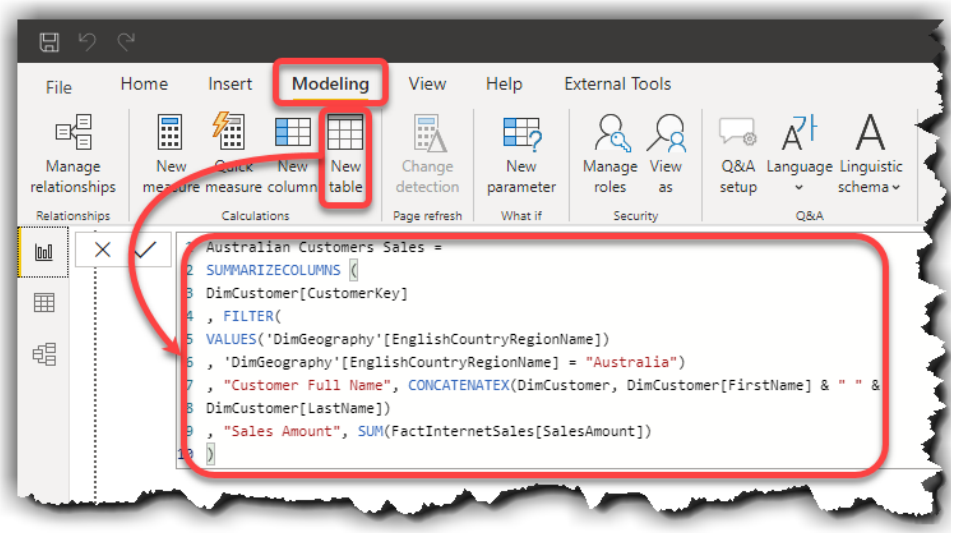


Figure 51: Creating a calculated table to show sales for Australian customers

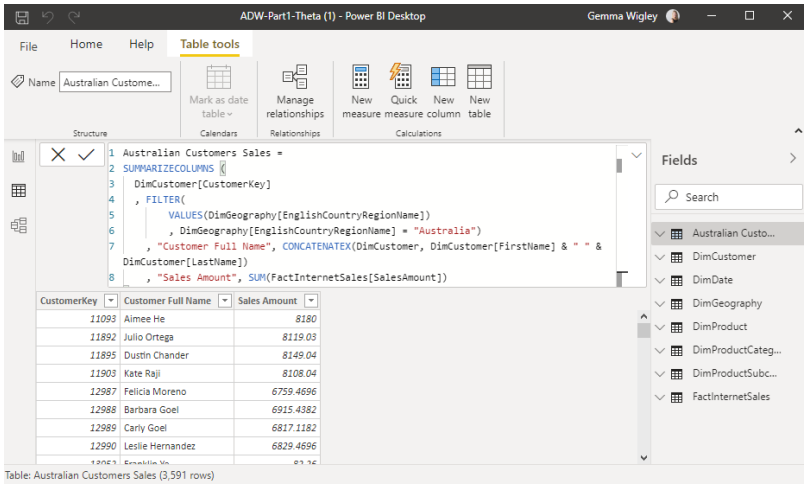


Figure 52: The results of running the preceding DAX expression

5.2.2 Calculated Columns

Calculated columns are the new columns created in the data model using DAX. There are many scenarios when you want to create a calculated column; the general rule of thumb is that you only create a new calculated column if:

- There's a complex scenario and you want to create calculated columns to use them in other calculations like measures.
- You need to create a new calculated column to be used in a slicer or as a filter on a report element.

Other than that, you should avoid creating new calculated columns as there's risk of performance degradation.

You can create a calculated column either from the Report view or the Data view (you don't need to complete this as a step):

A. Right click on a table from the Fields pane and select New Column.
B. Select a table from the Fields pane: select New Column from the Table Tools tab.
C. Select a table from the Fields pane: select New Column from the Modelling tab.

Figure 53 shows how to create a calculated column from the Report view.

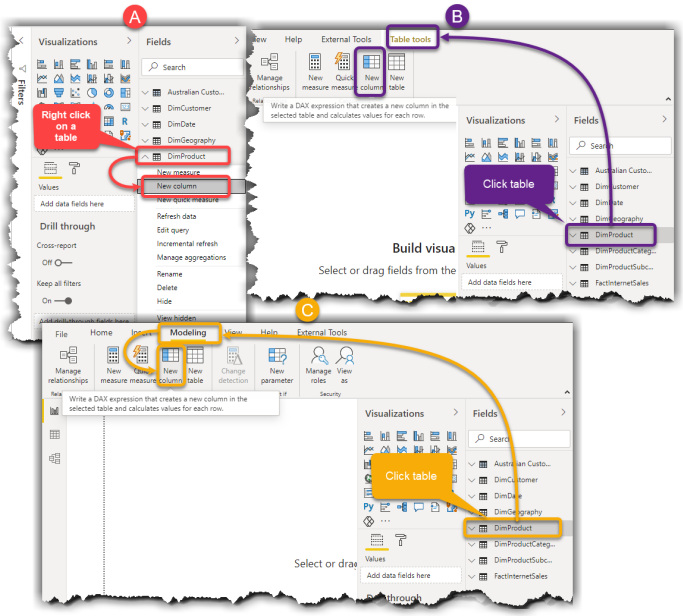


Figure 53: Create a Calculated Column

Let's have a look at Calculated Columns in action. Follow the steps below:

1. Click the **Data view** tab from the left pane.
2. Go to the **Fields** column on the right, select **DimCustomer.**
3. Right click, go to **New column**. Type in the expression below.
4. Press enter.

```
Full Name = CONCATENATE(DimCustomer[FirstName] &" ", DimCustomer[LastName])
```

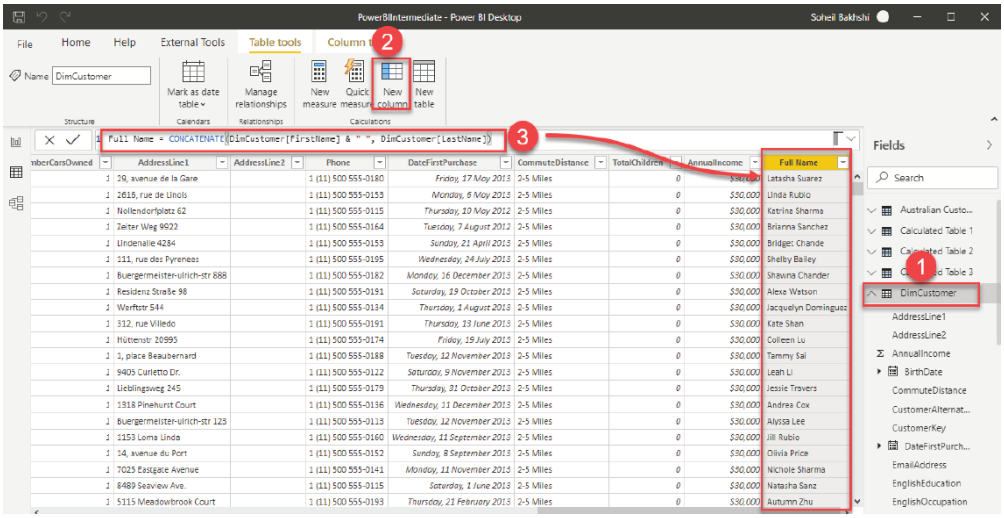


Figure 54: Creating Full Name calculated column in the DimCustomer table

Note that there's a space between the "and" in the DAX expression!

5.2.3 Measures

Measures are normally analytical calculations: summations, calculating averages, minimum, maximum, counts and so on. You can use the measures for visuals in Power BI.

The result of the measures change depending on how we interact with them in different visuals. For instance, you create a measure to calculate Sales Amount. If you put the Sales Amount on a Card visual, it shows total sales amount over the whole data. If you use the same measure in a Column Chart with Product Category on the Axis, the measure always calculates the correct results for each category (Figure 55).

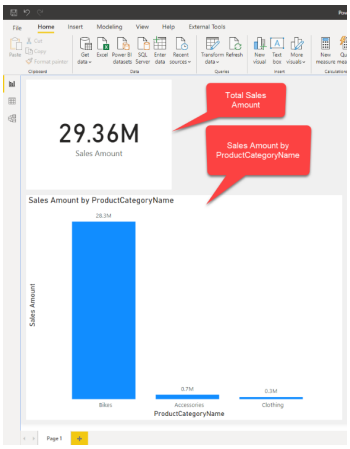


Figure 55: The result changes depending on visual interactions

You can create measures in various ways, just like the way you create calculated columns - either from the Report view or from the Data view (you don't need to follow these as steps).

A. Right click on a table from the **Fields** pane and select **New measure.**

B. Select a table from the **Fields** pane and select **New measure** from the **Table Tools** tab.

C. Select a table from the **Fields** pane and select **New measure** from the **Modelling** tab.

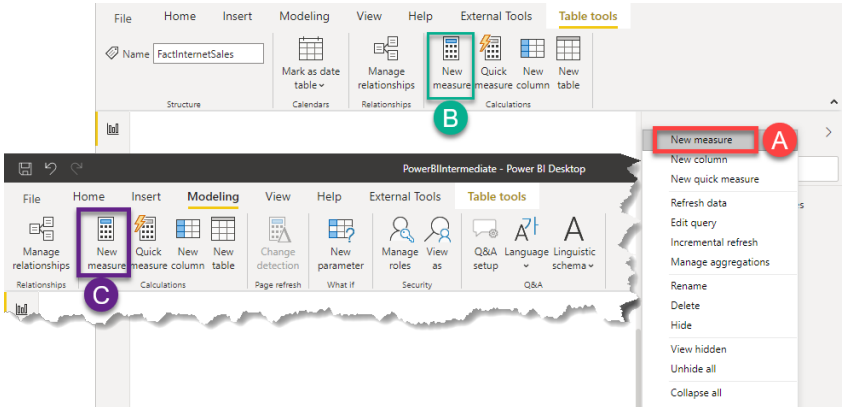


Figure 56: Creating a new measure in Power BI Desktop

Follow these steps below:

1. Click the **Report** view on the left pane.
2. Right click the **FactInternetSales** table in the **Fields** pane.
3. Click **New Measure.**
4. Use the DAX expression below. This creates a **Sales Amount** measure in the **FactInternetSales** table.

```
Sales Amount = SUM(FactInternetSales[SalesAmount])
```

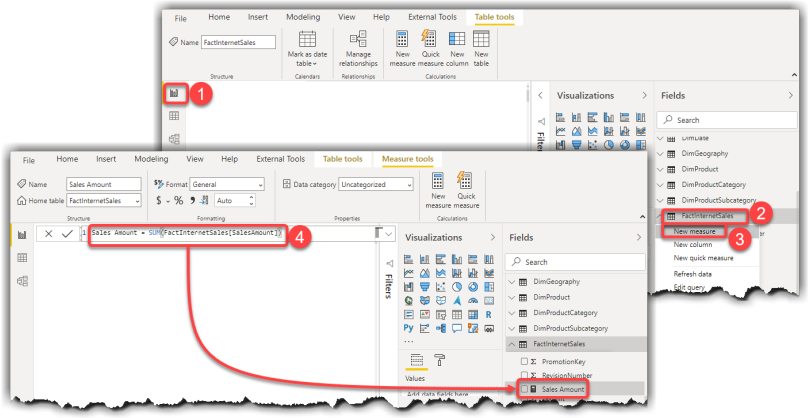


Figure 57: Creating Sales Amount measure in the FactInternetSales

5. Click the **Card visual** from the **Visualizations** pane on the reporting canvas.

6. Tick the **Sales Amount** Sales Amount measure to display its value on the Card visual.

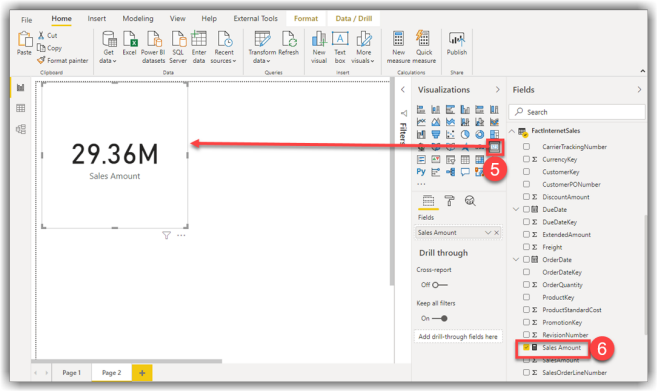


Figure 58: Showing Sales Amount on a Card visual

7. Click on a white space outside of the Card visual to release the focus from the Card visual.

8. Go to **FactInternetSales** and tick **Sales Amount** to add the measure to the visual.

9. Click the **Clustered Column Chart** visual from the **Visualizations** pane to add on the report canvas.

10. Go to the **Fields** pane on the right.

11. Go to DimProductCategory and tick the ProductCategoryName column to add it to the visual Axis.

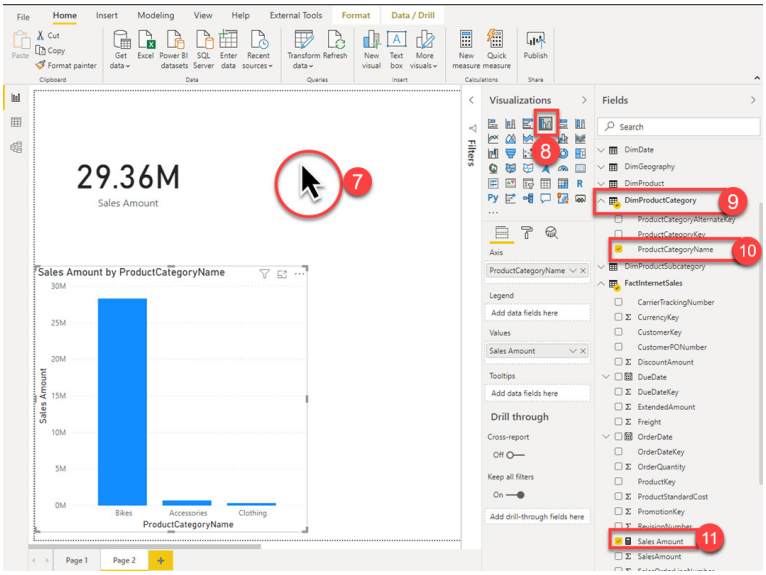


Figure 59: Sales Amount by ProductCategoryName in a Clustered Column Chart

Sometimes, creating analytical calculations is not as easy as a simple summation.

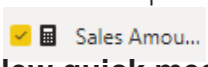
There are often other measures you have to create to satisfy the business requirements; some requirements are complex. Power BI makes lots of these calculations super easy via Quick Measures - which you're about to discover more about in the next section.

5.2.4 Quick Measures

You can use Power BI to create complex reporting logics to analyse and visualise data.

Quick Measures are pre-defined calculations provided by Microsoft and can be created on the fly. Let's have a look at some use cases.

Follow the steps to create a new measure on top of the existing Sales Amount measure:

1. Go to the **Fields** pane, go to **FactInternetSales,** right click the **Sales Amount** measure 

2. Click **New quick measure.**
3. Go to the **Calculation** dropdown list, go to **Totals**, select **Running total.**
4. The **Sales Amount** is already selected in the **Base value.**
5. Go to the DimDate table, drag and drop the **Date column** to the **Field** section.
6. Click **OK.**

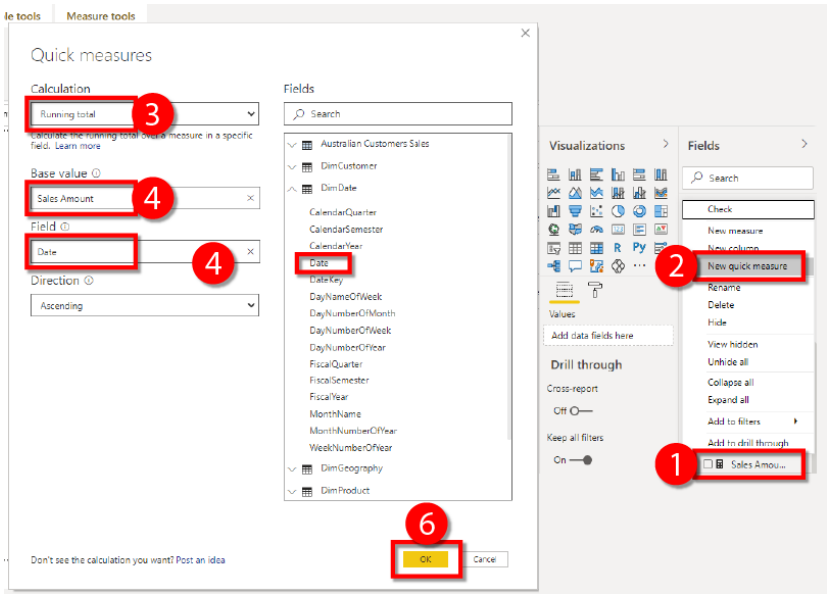


Figure 60: Creating Sales Amount running total in Date measure using Quick Measures feature

You'll have a new measure named 'Sales Amount running total in Date.'

7. Select the **Line Chart** visual from the **Visualizations** pane.
8. Go to the **Fields** Pane, select **FactInternetSales**, tick the **Sales Amount running total in Date** measure to put it in the chart's Values.
9. Go to **DimDate**, tick the **Date** column.
10. Right click on the **Date** from the **Axis** field.
11. Click **Date** to show the Date column as normal date (not a date hierarchy).

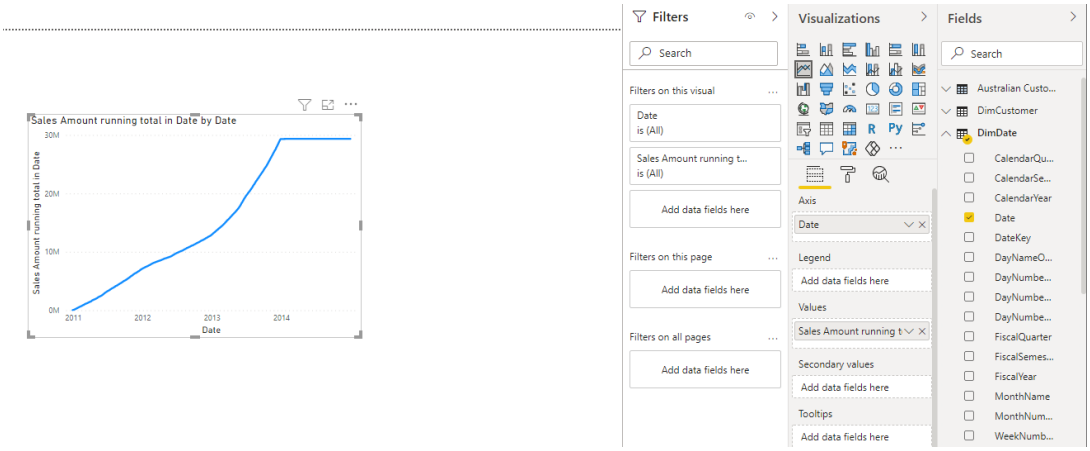


Figure 61: Visualising Sales Amount running total in Date

There's a long list of available quick measures; lots of them useful. You can also learn how to write DAX using quick measures. For instance, you can click the Sales Amount running total in Date measure and look at the formula bar to learn how to write a running total over date as Figure 62 shows:

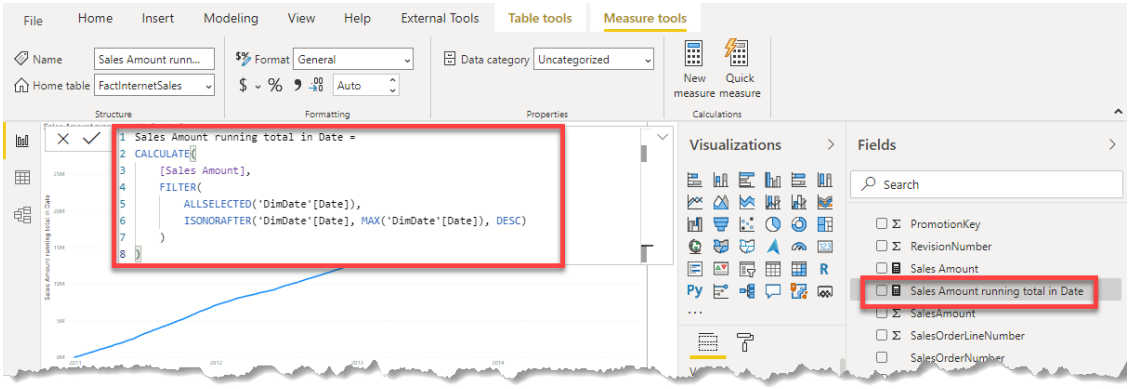


Figure 62: Click on a quick measure to learn how the corresponding DAX is written

5.2.5 Time Intelligence

Time intelligence provides measures that simplify complex time-based reporting.

For example, time-based calculations for Year-to-Date (YTD), Month-to-Date (MTD), and Previous Period comparisons. In this section, you'll learn:

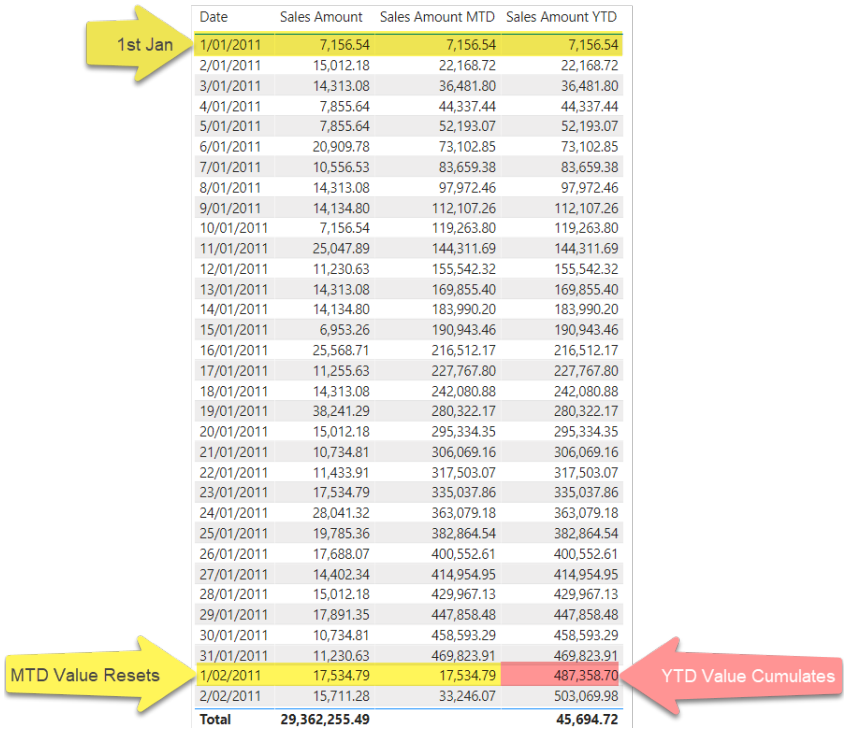
- What time intelligence means in the data level.
- Which time intelligence functions are available in DAX .
- How to overcome complex time intelligence challenges using Quick Measures.

If you're unfamiliar with time intelligence, it's about time-based calculations.

For instance, when you calculate a Year-to-Date (YTD) metric, you're summing the values of that metric on a daily basis, starting from 1st Jan of each year up until the provided date. When you calculate a Month-to-Date (MTD) metric, you're summing the values of that metric for each day, starting from the 1st day of the month up until the provided date.

Let's have a look at an example.

Figure 63 shows Sales Amount values side-by-side to Sales Amount MTD and Sales Amount YTD :



Date	Sales Amount	Sales Amount MTD	Sales Amount YTD
1/01/2011	7,156.54	7,156.54	7,156.54
2/01/2011	15,012.18	22,168.72	22,168.72
3/01/2011	14,313.08	36,481.80	36,481.80
4/01/2011	7,855.64	44,337.44	44,337.44
5/01/2011	7,855.64	52,193.07	52,193.07
6/01/2011	20,909.78	73,102.85	73,102.85
7/01/2011	10,556.53	83,659.38	83,659.38
8/01/2011	14,313.08	97,972.46	97,972.46
9/01/2011	14,134.80	112,107.26	112,107.26
10/01/2011	7,156.54	119,263.80	119,263.80
11/01/2011	25,047.89	144,311.69	144,311.69
12/01/2011	11,230.63	155,542.32	155,542.32
13/01/2011	14,313.08	169,855.40	169,855.40
14/01/2011	14,134.80	183,990.20	183,990.20
15/01/2011	6,953.26	190,943.46	190,943.46
16/01/2011	25,568.71	216,512.17	216,512.17
17/01/2011	11,255.63	227,767.80	227,767.80
18/01/2011	14,313.08	242,080.88	242,080.88
19/01/2011	38,241.29	280,322.17	280,322.17
20/01/2011	15,012.18	295,334.35	295,334.35
21/01/2011	10,734.81	306,069.16	306,069.16
22/01/2011	11,433.91	317,503.07	317,503.07
23/01/2011	17,534.79	335,037.86	335,037.86
24/01/2011	28,041.32	363,079.18	363,079.18
25/01/2011	19,785.36	382,864.54	382,864.54
26/01/2011	17,688.07	400,552.61	400,552.61
27/01/2011	14,402.34	414,954.95	414,954.95
28/01/2011	15,012.18	429,967.13	429,967.13
29/01/2011	17,891.35	447,858.48	447,858.48
30/01/2011	10,734.81	458,593.29	458,593.29
31/01/2011	11,230.63	469,823.91	469,823.91
1/02/2011	17,534.79	17,534.79	487,358.70
2/02/2011	15,711.28	33,246.07	503,069.98
Total	**29,362,255.49**		**45,694.72**

Figure 63: Sales Amount MTD and Sales Amount YTD

Look at the first and last 3 rows of Figure 64 to see how the numbers are calculated.

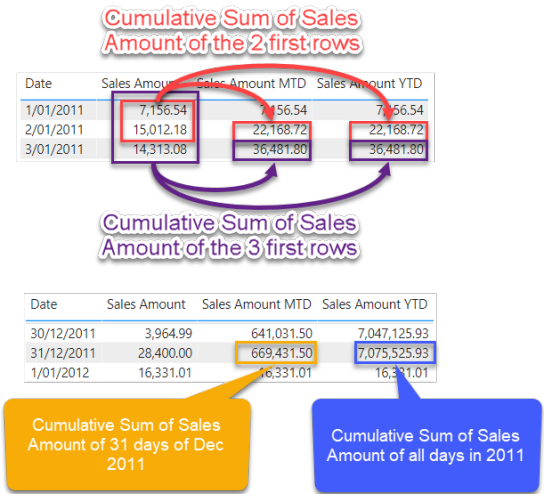


Figure 64: The math behind Month to Date and Year to Date calculations

You might think that the underlying calculation is complex.

The good news is that there are specific functions in DAX that take care of the complexities around the logics. This is shown below.

```
Sales Amount MTD =
  TOTALMTD([Sales Amount], 'DimDate'[Date])
```

```
Sales Amount YTD =
  TOTALYTD([Sales Amount], 'DimDate'[Date])
```

We used **TOTALMTD()** or **TOTALYTD()** DAX functions to create the above measures.

We used the (already created) **Sales Amount** measure in those functions along with the Date column from the DimDate table. The DimDate is a Date table holding date related data. The Date table is the basis for all time intelligence functions available in Power BI, therefore we need to have a Date table in our model.

What if our data source doesn't have a Date table? We have two options:

A. Use the **Auto DateTime** feature in Power BI Desktop
B. Generate a Date table

Let's look at the Date table in more detail and see why we need one in our data model.

5.2.5.1 Why a Date Table is needed in Power BI

The Date table is the basis for time intelligence functions to work correctly, but there's another reason why we need to have a Date table in a Power BI model.

The Date table provides the detail for a date. If you go to the **Data view** (make sure you have selected **FactInternetSales** on the right pane outlined in red in Figure 65), you'll see 3 date columns: OrderDate, ShipDate and DueDate columns. While those columns provide date values and we can use them to analyse our data, what if we need to analyse by other date elements? e.g. Sales Amount by month to see our bestselling month.

You can achieve this by creating a calculated column to show Month values.

What if you want to analyse the Sales Amount by financial year, do you create a calculated column? Business queries can grow over time; it's unwise to create new columns whenever the business asks new questions. Look at Figure 65. There are 60,399 rows of data.

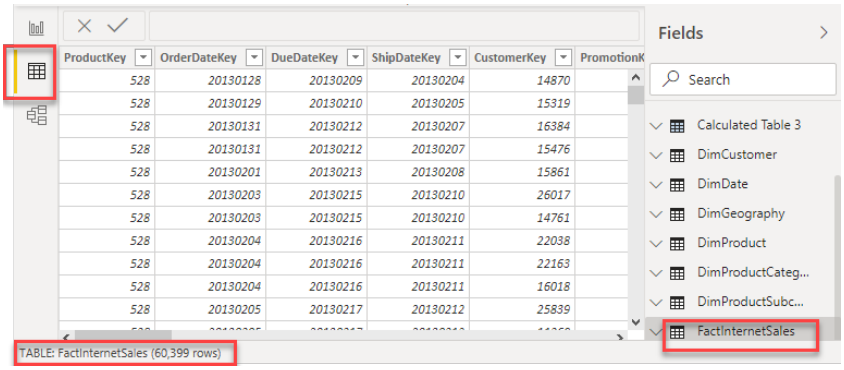


Figure 65: Number of rows of the FactInternetSales table in the Data view

Creating a new column to only add the Month values increases the size of the FactInternetSales table, i.e. it unnecessarily increases the size of the report file and leads to poor performance in larger reports.

To overcome this challenge, create a separate Date table by adding as many columns as you need, then create relationships to other tables to analyse their data over date elements. Figure 66 shows the data in DimDate from the sample file supplied with this guide. DimDate only has 3,652 rows, but it provides rich and detailed date elements for data analysis.

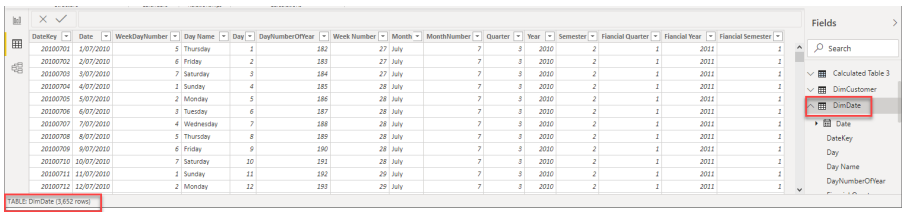


Figure 66: DimDate of the sample file

5.2.5.2 Auto Date/Time Feature in Power BI

If you don't have a Date table in your data source, don't worry. You can enable the Auto date/time feature in Power BI Desktop. Power BI automatically takes care of the Date table. To enable this feature, follow the steps:

1. In Power BI Desktop click the **File** menu
2. Click **Options and settings**
3. Click **Options**

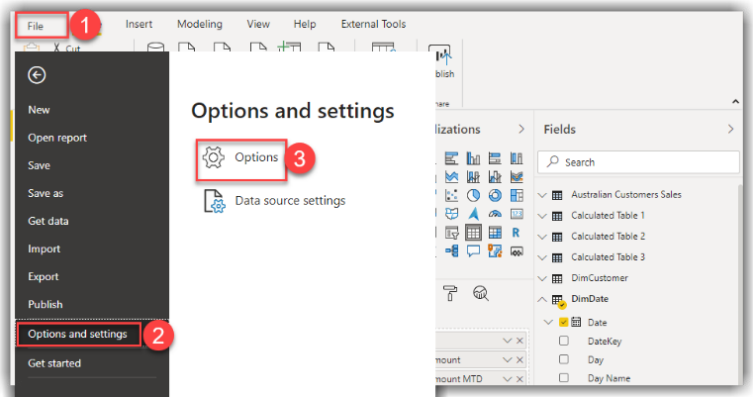


Figure 67: Opening Power BI Desktop Options

4. Under the **GLOBAL** section, go to the **Data Load** tab. Under **Time intelligence -** tick **Auto date/time for new files** (This may be already ticked).

This enables the feature globally across all Power BI files you create in future.

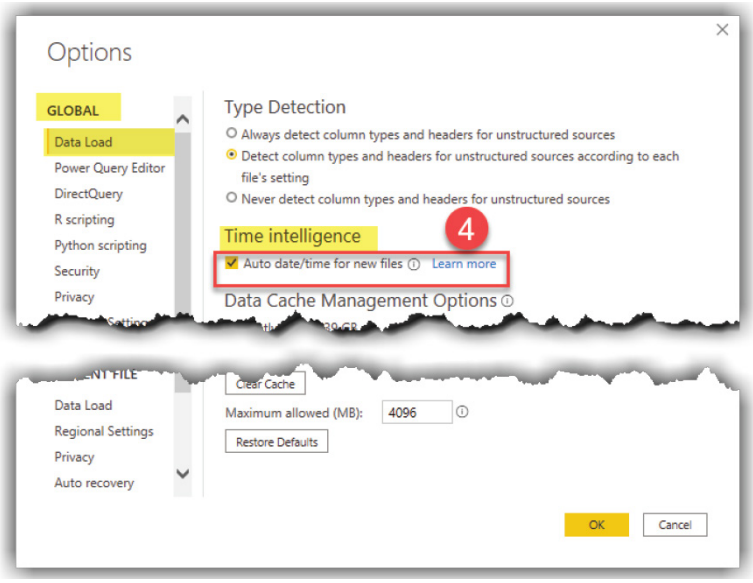


Figure 68: Enabling Auto date/time globally

5. Under the **CURRENT FILE**, tick **Auto date/time.**
6. Click **OK.**

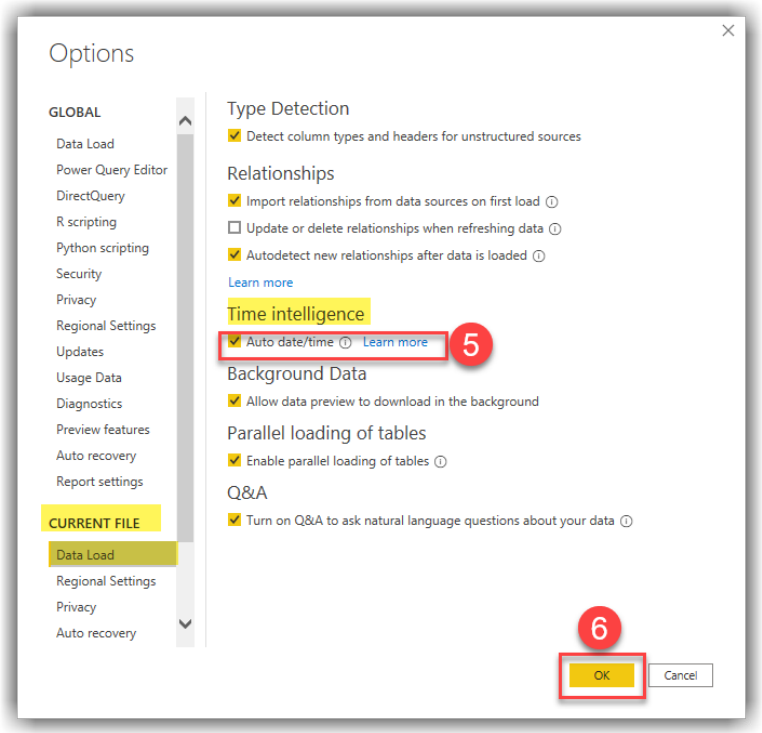


Figure 69: Enabling Auto date/time for the current file

After enabling the Auto date/time feature, you'll see that all Date or DateTime type columns have changed to Date Hierarchy (Figure 70).

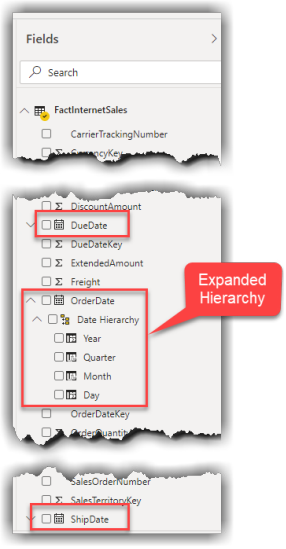


Figure 70: Enabling Auto date/time automatically creates date hierarchies for date columns

Now that you've enabled the Auto date/time feature, you can use Quick Measures to build time intelligence measures - without having (or creating) a Date table in the model.

It's advised to create a Date table if your source system doesn't have one, rather than relying on the Auto date/time feature. While the Auto date/time feature can be very helpful, it comes with some side effects. If you enable the Auto date/time feature, Power BI Desktop creates private Date tables in the background. Indeed, Power BI creates a Date table per column with either Date or DateTime data type regardless of whether we need to analyse them or not.

Not only is it excessive to have a Date table per column, but it can cause serious performance and storage issues. To disable this feature, just reverse back the settings we made in this section.

5.2.5.3 Generating a Date Table in Power BI Desktop

There are two ways to create a Date table in Power BI Desktop:

1. Create a Date table in Power Query (M) language.
2. Create a Date table using DAX.

Note: Our sample file already has a Date table (DimDate), so we don't need to generate another Date table. However, we include two extra Date tables just for the purpose of this guide - helping you solving real-world challenges.

5.2.5.3.1 Generating Date Table in Power Query

Open the Power Query Editor and go through the steps below to generate a Date table using Power Query (M) language. Follow these steps:

1. Open Power Query Editor (go to **Transform Data**).
2. Click **New Source** (or Get Data) dropdown button.
3. Click **Blank Query.**

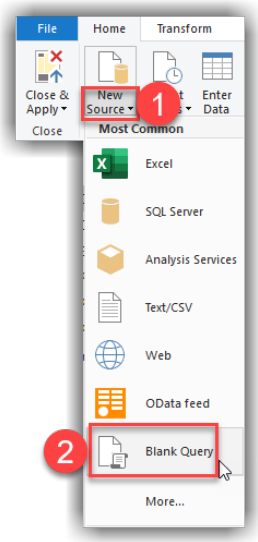


Figure 71: Opening a Blank Query in Power Query Editor

4. Click the **Advanced Editor** button from the **Home** tab.
5. Copy the scripts below and paste in the Advanced Editor window.
6. Click **Done.**

```
let
  Source = List.Dates(#date(2010, 1, 1), Duration.Days(Duration.From(#date(2014, 12, 31) -
#date(2010, 1, 1))), #duration(1, 0, 0, 0) ),
  #"Converted to Table" = Table.FromList(Source, Splitter.SplitByNothing(), null, null, ExtraValues.
Error),
  #"Renamed Columns" = Table.RenameColumns(#"Converted to Table",{{"Column1", "Date"}}),
  #"Added Custom" = Table.AddColumn(#"Renamed Columns", "DateKey", each Text.
Combine({Date.ToText([Date], "yyyy"), Date.ToText([Date], "MM"), Date.ToText([Date], "dd")})),
  #"Changed Type" = Table.TransformColumnTypes(#"Added Custom",{{"Date", type date},
{"DateKey", Int64.Type}}),
  #"Year Column Added" = Table.AddColumn(#"Changed Type", "Year", each Date.Year([Date])),
  #"Quarter Column Added" = Table.AddColumn(#"Year Column Added", "Quarter", each "Qtr
"&Text.From(Date.QuarterOfYear([Date]))),
  #"MonthOrder Column Added" = Table.AddColumn(#"Quarter Column Added", "MonthOrder",
each Date.ToText([Date], "MM")),
  #"Short Month Column Added" = Table.AddColumn(#"MonthOrder Column Added", "Month
Short", each Date.ToText([Date], "MMM")),
  #"Month Column Added" = Table.AddColumn(#"Short Month Column Added", "Month", each
Date.MonthName([Date])),
  #"Changed Columns Type" = Table.TransformColumnTypes(#"Month Column Added",{{"Year",
Int64.Type}, {"MonthOrder", Int64.Type}})
in
  #"Changed Columns Type"
```

The script generates dates between 2010 to 2014 (our sample data only covers that date range).

This is most probably different in your real-world scenarios; therefore, you need to change the code to satisfy your needs. To cover a different date range, you need to change those numbers in the code.

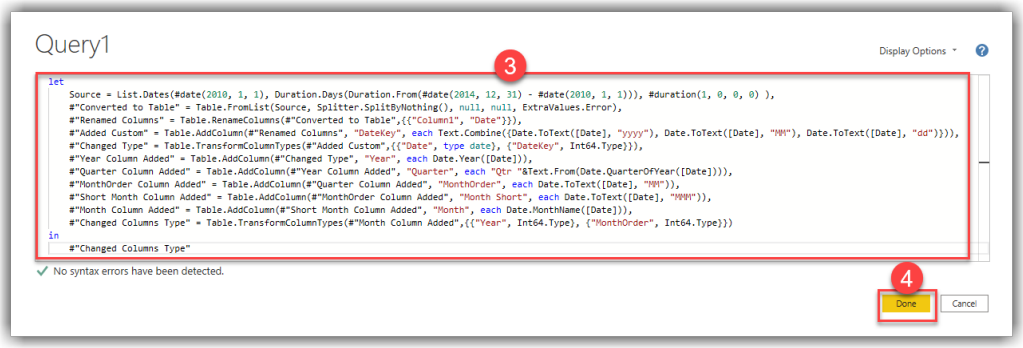


Figure 72: Generating Date table in Power Query

6. Rename the Query1 query to Date with M
7. Close and Apply to exit Power Query

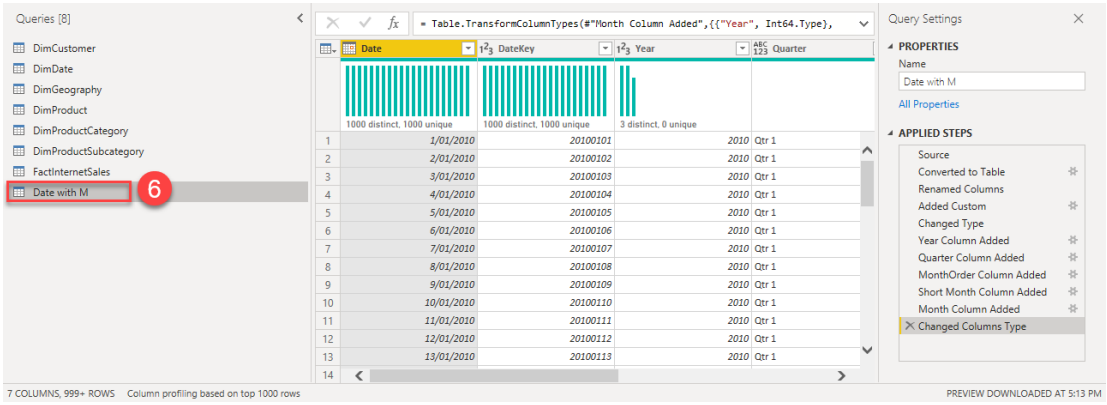


Figure 73: Date table in Power Query

5.2.5.3.1 Generating Date Table using DAX

The second method to create a Date table is using DAX by creating a new calculated table (as per 6.4.1).

The DAX code generates the same Date table as the one we created with Power Query:

```
Date with DAX =
  ADDCOLUMNS(CALENDAR(DATE(2007,1,1), DATE(2020,12,31))
        , "DateKey", VALUE(FORMAT([Date], "YYYYMMDD"))
        , "Month", FORMAT([Date], "MMMM")
        , "Month Short", FORMAT([Date], "MMM")
        , "MonthOrder", FORMAT([Date], "MM")
        , "Quarter", CONCATENATE("Qtr ", QUARTER([Date]))
        , "Year", YEAR([Date])
    )
```

The result of the DAX formula is shown in Figure 74.

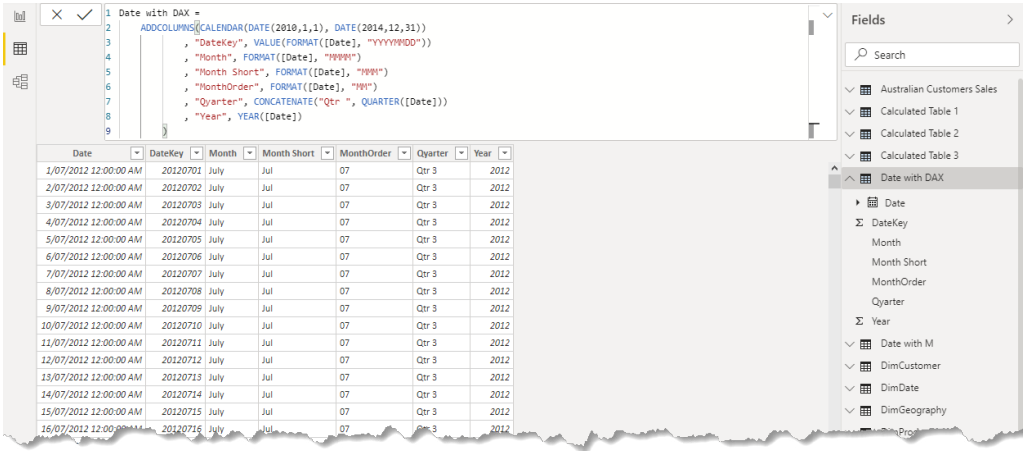


Figure 74: Generating Date table using DAX

5.2.5.4 Marking Date Table as Date

So far you've learnt: what a Date table is, why you need one in the Power BI model, how to generate one if the Date table doesn't exist in the source system. You 've also learnt how to configure and use the Auto date/time feature in Power BI Desktop.

But having a Date table in your data model doesn't mean that the time intelligence functions will work correctly. Your Date table must meet requirements below:

- Your Date table MUST have one column with Date data type
- The Date column MUST contain unique values
- The Date column MUST have continuous date values - without any gaps in the dates
- The Date column values MUST start from 1st Jan of a starting year; going upto the 31st Dec of an ending year
- Mark the Date table as Date

The first 4 requirements are the prerequisites for the 5th requirement.

Here's how to 'Mark as Date'.

1. Right click the **Date table** you would like to mark as Date.
2. Hover-over **Mark as date table.**
3. Click **Mark as date table.**
4. Select a **Date column** from the dropdown list.
5. Click **OK.**

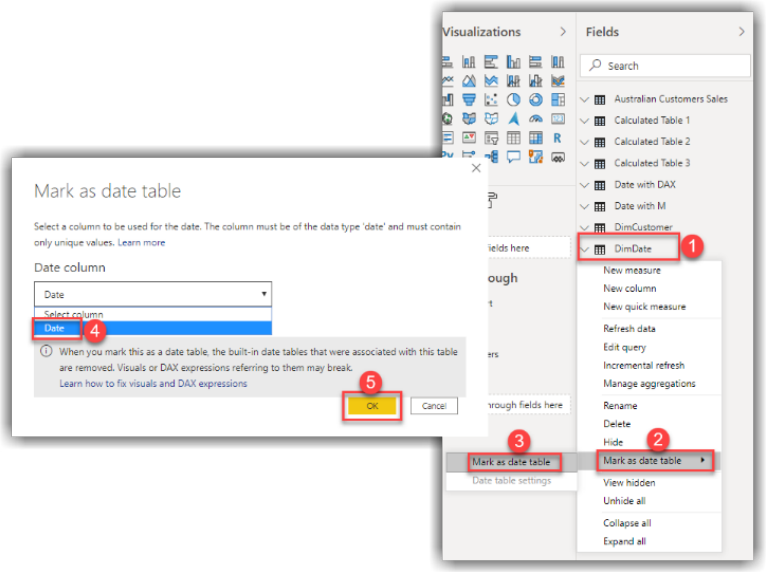


Figure 75: Marking Date table as Date

5.2.5.5 Adding Time Intelligence Quick Measures

Quick Measures are pre-defined calculations.

In the following steps, you'll create Year-to-Date measures over the OrderQuantity column; from the FactInternetSales table using Quick Measures. Follow these steps:

1. When in Power BI Desktop, expand **FactInternetSales.**
2. Right click the **OrderQuantity** column.
3. Select **New quick measure** from the context menu.
4. Select **Year-to-date total** from the **Calculation** dropdown.
5. Make sure Base value is **Sum of OrderQuantity.**
6. Expand the **DimDate table.**
7. Drag **Date** to the **Date box.**
8. Click **OK.**

These steps are shown in Figure 76.

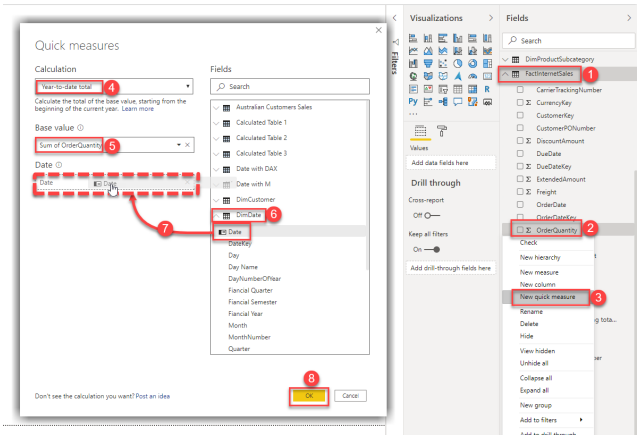


Figure 76: Creating a new time intelligence with Quick Measures feature

Power BI Desktop adds your new measure to the Fields pane and displays the generated DAX code (Figure 77).

The DAX code is fully functional and requires no further changing. It's strongly advised that only qualified BI technicians edit or create DAX code for business-critical systems.

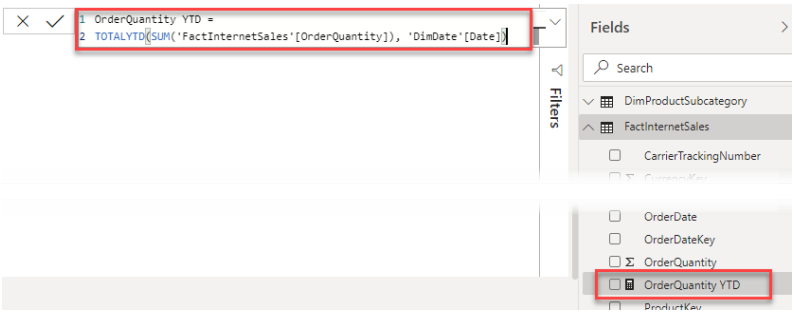


Figure 77: The new measure created by Quick Measures feature

6. Reporting on the Data – Creating Visualisations

In the previous chapter we learnt:

- How to prepare data in Power Query
- Load resulting data into the data model
- Creating calculated tables, calculated columns and measures
- Creating and configuring the Date table.

Now, it's time to visualise the data in Power BI Desktop.

6.1 Building Basic Visualisations

Let's start with a graph to support a simple business scenario.

SCENARIO: Follow these steps:

Our Sales Manager wants to track the value of website bike sales over the last few years.

1. From the **Report** view, select the **Stacked Column Chart** visual from the **Visualizations** pane (This will add a grey "place holder" graphic to the Report Canvas).

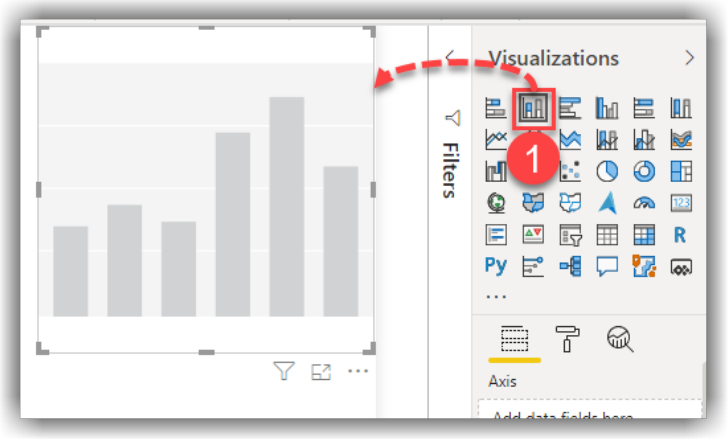


Figure 78: Adding a Stacked Column Chart to the report canvas

2. Drag the **Sales Amount** Sales Amount measure, drop it on the visual place holder (this creates one column).

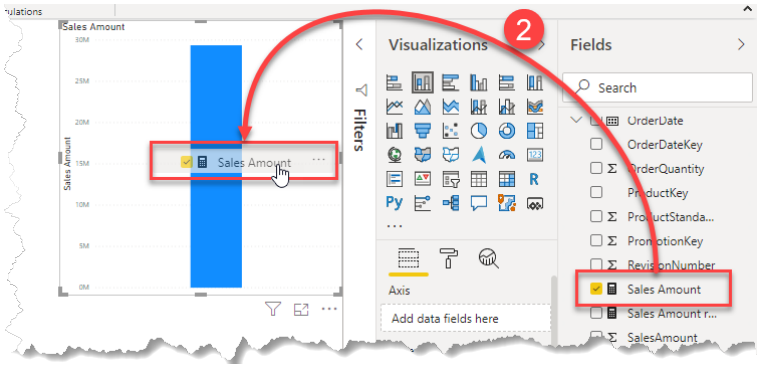


Figure 79: Drag and drop the Sales Amount measure to the visual

3. Drag the **CalanderYear** column from the **DimDate** table and drop it on the visual (this adds the Year to the visual's Axis).

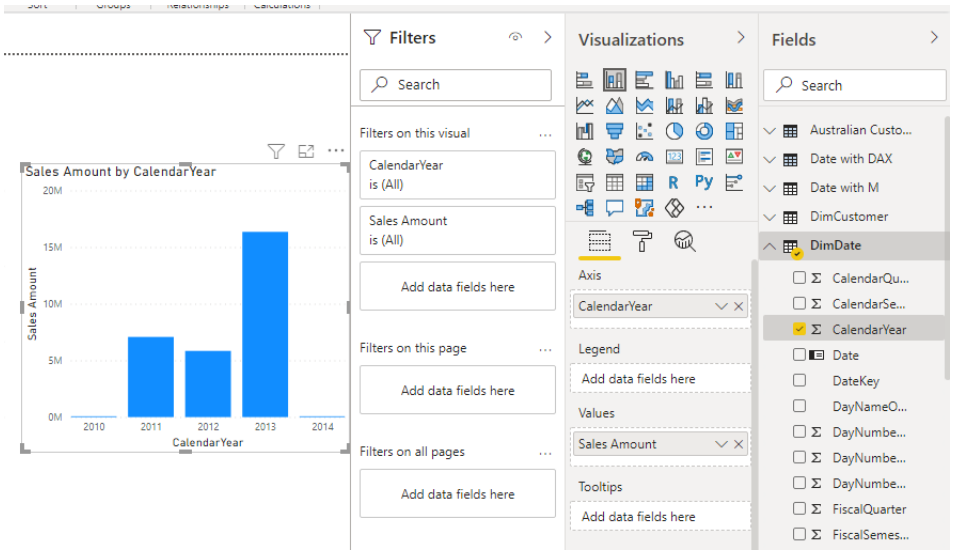


Figure 80: Drag and drop Year to the visual

4. Click the chart, go to the the Visualizations pane, click the **format** icon from the visualizations pane.
5. Disable Y Axis.
6. Enable **Data Labels** to show values on each bar. This makes the visual more readable.

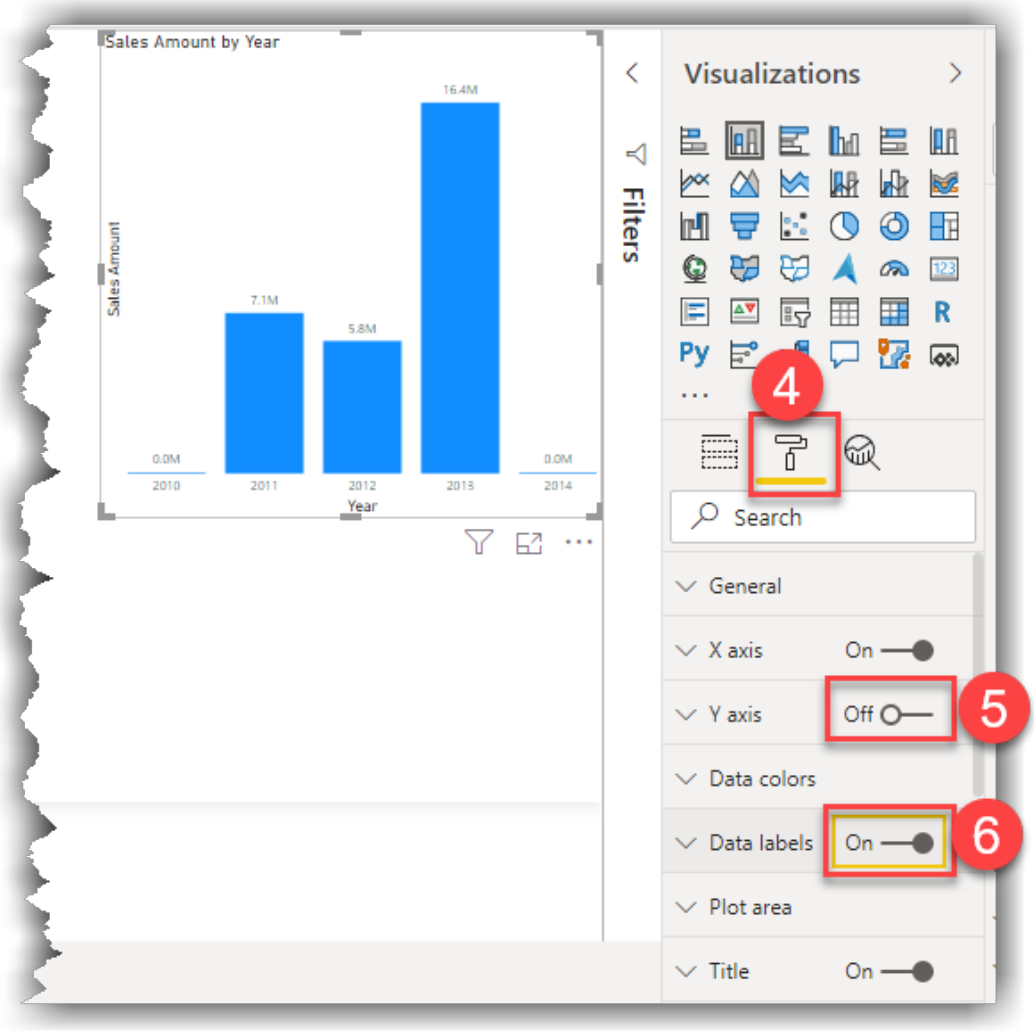


Figure 81: Disabling Y Axis and enabling Data Labels of the Stacked Column Chart

Next, our marketing manager would like to know the occupation of customers who order the most stock to target with a new campaign.

We'll follow a slightly different approach to build this visualisation:

7. Drag the **OrderQuantity** column from the **FactInternetSales** and drop it onto a blank space of the report canvas. This creates a new Visualisation using the default Stacked Column Chart.

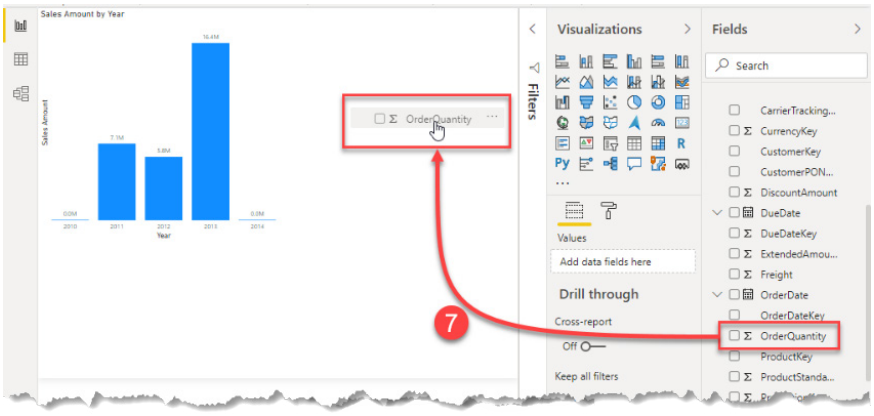


Figure 82: Creating new visual on the report canvas by dragging and dropping a column to a blank space

8. Drag the **EnglishOccupation** column from the **DimCustomer** and drop it on the newly created chart.

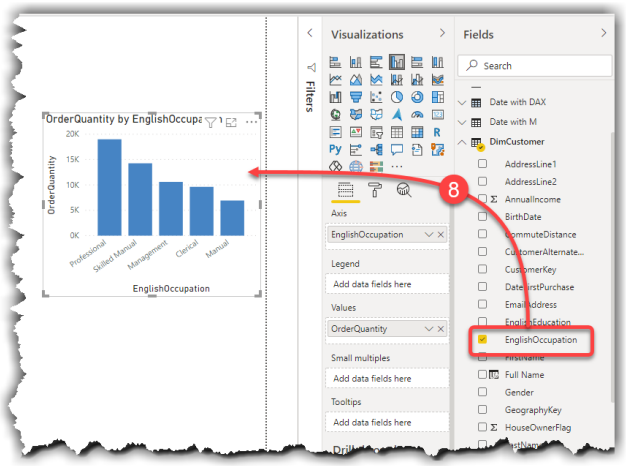


Figure 83: Continue adding columns to visuals by drag and drop

9. Make sure the column chart is still on focus, then click on **Pie Chart** visual from the Visualizations pane to turn the column chart to a Pie chart.

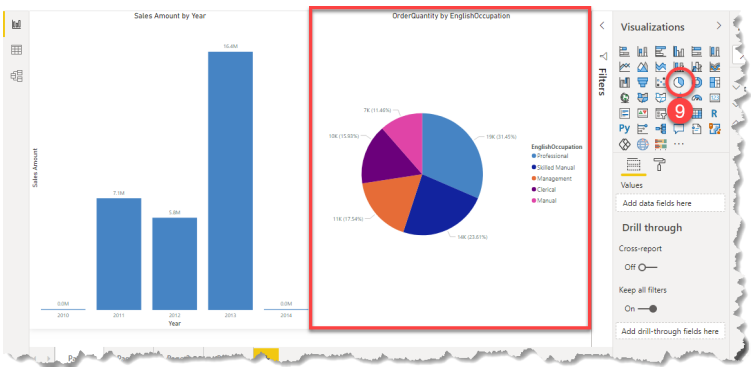


Figure 84: Converting Column chart to Pie chart

10. Move and resize the visuals to nicely fit the page.

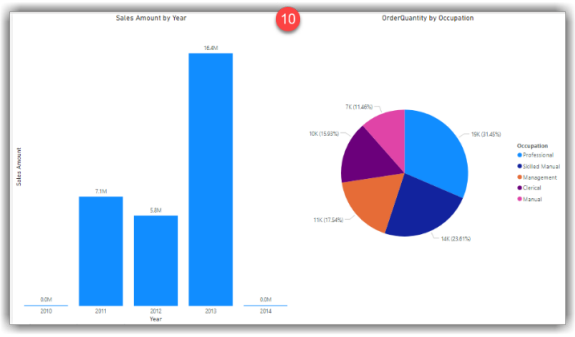


Figure 85: Adjusted visuals to fit the page

Now that you've learnt various ways of creating visuals, let's have a look at visualisation interactivity.

6.2 Visualisation Interactivity

With multiple visualisations visible on a page, Power BI Desktop allows you to cross-highlight and cross-filter the data between visualisations. You would do this to further drill down into your data.

Let's see how the visuals interact with each other. Click on the Professional slice on the Pie chart to see sales distributed across all years for this occupation. Follow these steps:

1. In the **Report Design** view, click on the **Professional** slice of the Pie Chart.
2. The portion of InternetSales related to Professionals is highlighted on the Stacked Column Chart and Pie Chart - this will cross highlight the other charts on the report page.

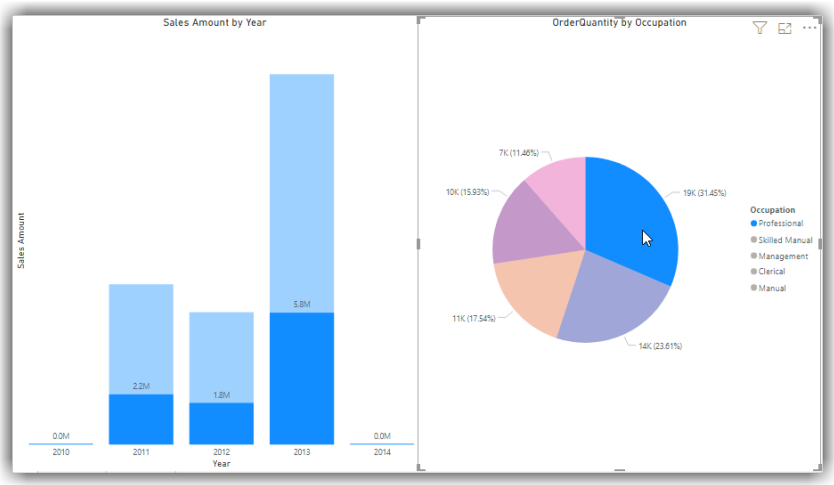


Figure 86: Cross-highlighting in visual interactions in Power BI Desktop

3. You can edit the cross-highlighting behavior to cross-filtering:
    A. Select the **Pie chart** from the report canvas
    B. Go to the **Format** tab, click the **Edit interactions**
    C. Change the interactivity of the Column chart to **Filter.**
    D. Select the **Column chart**
    E. Change the interactivity of the Pie chart to **Filter** (Figure 87)

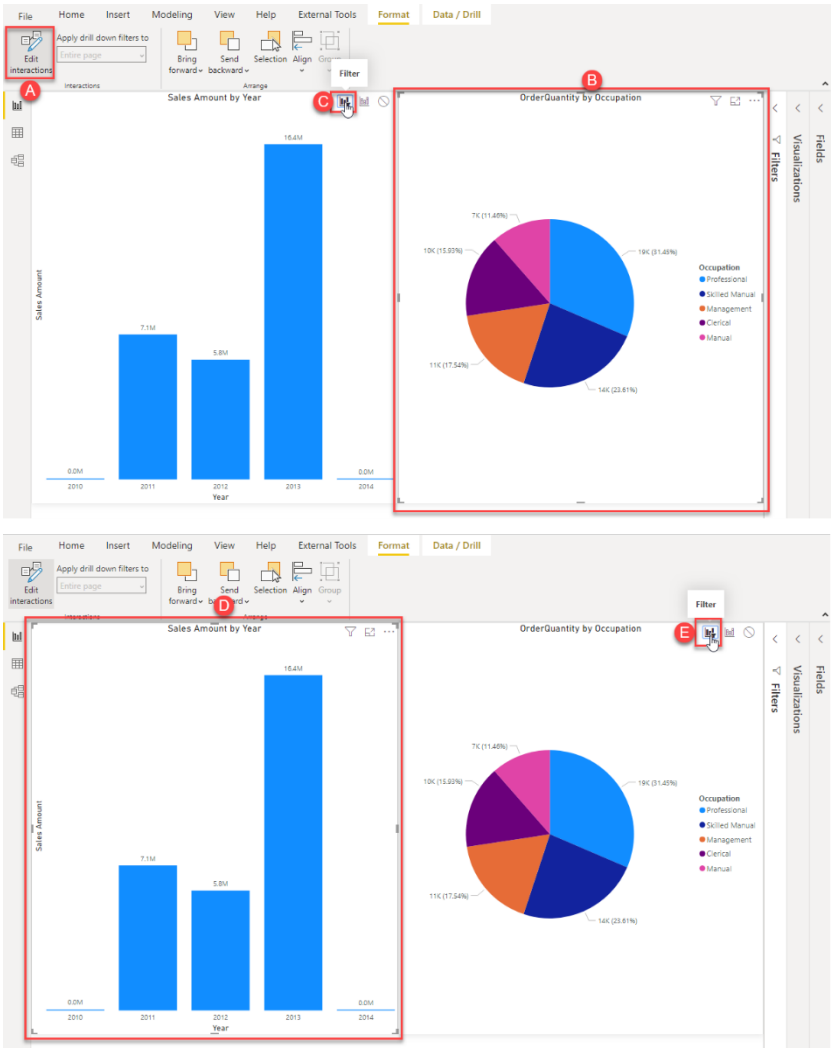


Figure 87: Editing visuals interactions

4. Click the **Edit Interaction** button again to set the changes.

You can change the default interaction behavior from cross-highlighting to cross-filtering by following the steps below:

1. **File**
- **Options and settings**
- **Options**
2. In the Options window, click **Report settings** tab (under the Current File).
3. Tick the Change default visual interaction from cross highlighting to **cross filtering.**
4. Click **OK.**

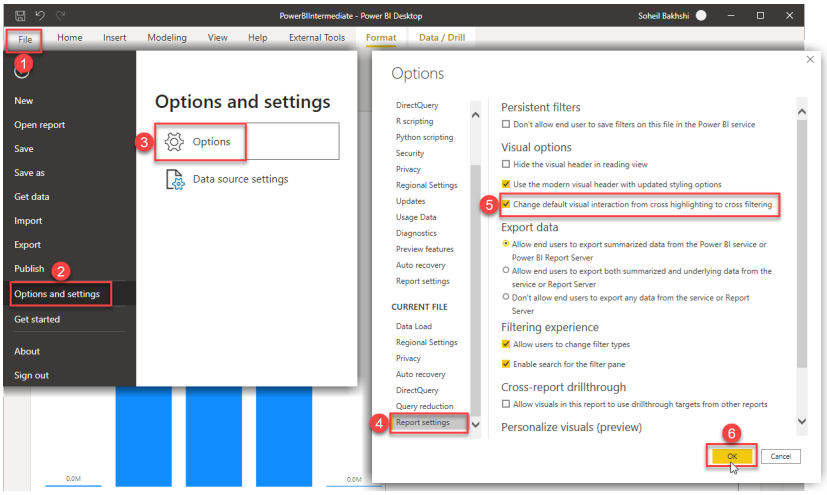


Figure 88: Changing the default visual interaction setting

7. Custom Visuals in Power BI

Power BI has a rich selection of visuals that can be used in your reports. You can also get Custom Visuals from AppSource.

7.1 Getting Custom Visuals from AppSource

1. Click the ... button on the Visualizations pane.
2. Click **Get more visuals.**
3. Search to find the custom visual you are after.
4. Click **Add.**

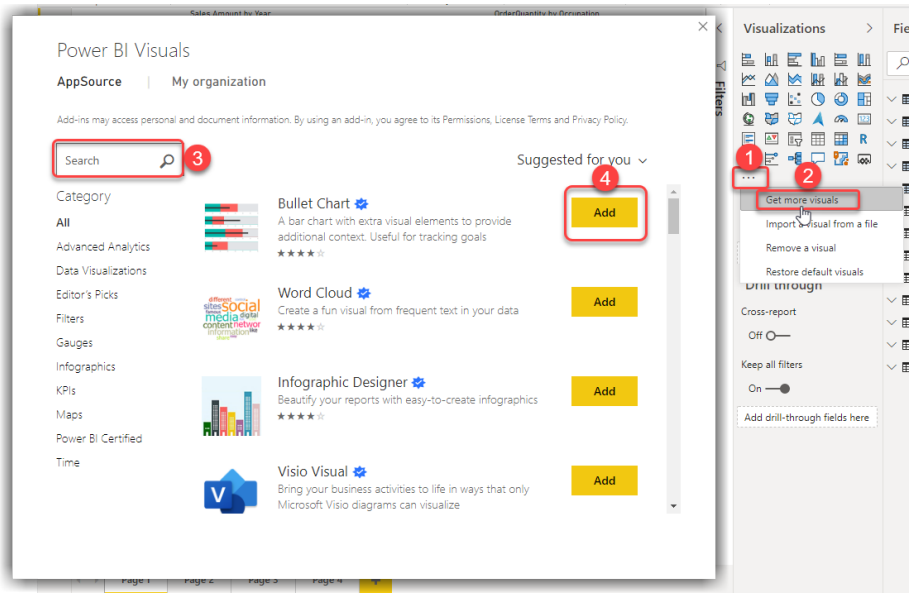


Figure 89: Getting custom visuals from AppSource

Custom Visuals are created by third parties, so they are NOT available within the Power BI Desktop by default. Just because they're on AppSource, it doesn't mean they're all safe to use. We'd recommend only using those which are Certified by Microsoft: they are fully tested and recognised as a safe visual. The custom visuals that are certified come with a Certified badge.

The custom visual appears in a separate section beneath the default visuals in the Visualizations pane (Figure 90).

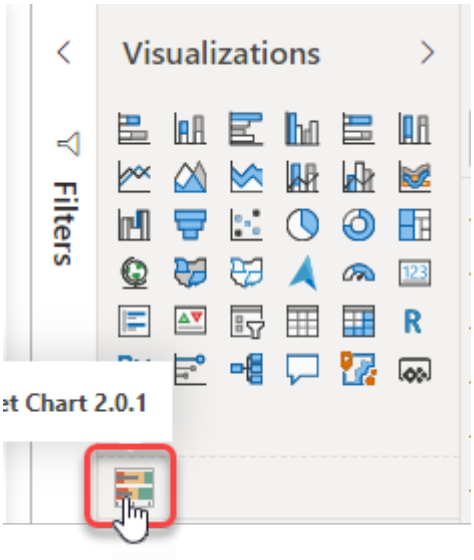


Figure 90: Custom Visuals show in a separate space underneath the default visuals in the Visualizations pane

7.2 Removing a Custom Visual

To remove a custom visual, follow the steps below:

1. Click **...** from the **Visualizations** pane.
2. Click **Remove a visual.**
3. Select the custom visual you want to remove. You can also select multiple custom visuals - Ctrl + select the custom visuals you want to remove.
4. Click the **Remove** button.

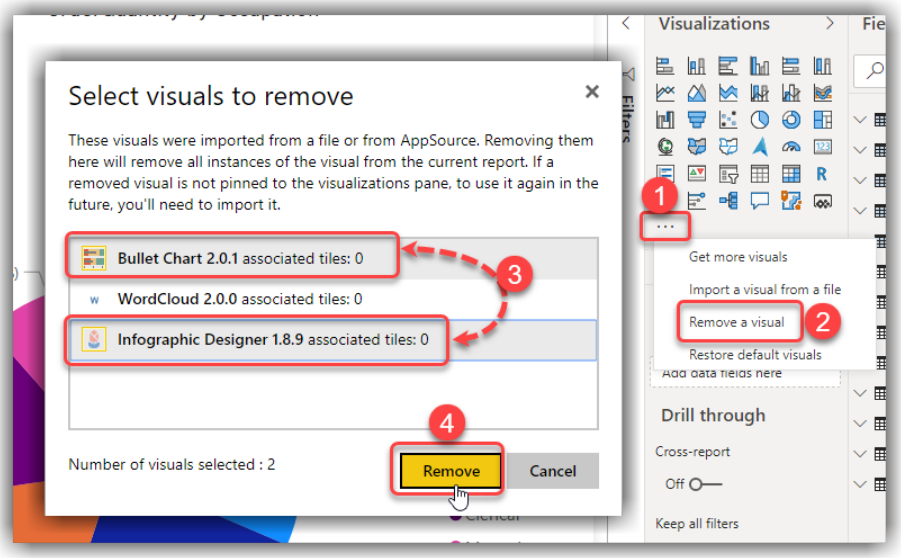


Figure 91: Removing Custom Visuals from Power BI Desktop

8. Publishing to Power BI Service

After you've finished your data visualisations in Power BI Desktop, it's time to Publish the report to Power BI Service.

If you have permission to publish reports to the service, it's easy. If you don't have access/permission, you won't be able to publish and share your reports in the cloud. You can still use your reports in Power BI Desktop or share it with others by sending them the .pbix file.

Follow the steps below to publish your report to Power BI Service **(Figure 92)**

1. Click the **Publish** button from the ribbon bar.
2. Type in you Power BI Service credentials.
3. Click **Sign in.**
4. Select a workspace you want to publish your report to.
5. Click **Select.**
6. After your report is successfully published, you can click the report link on the Publishing to Power BI window.
7. Click **Got it.**

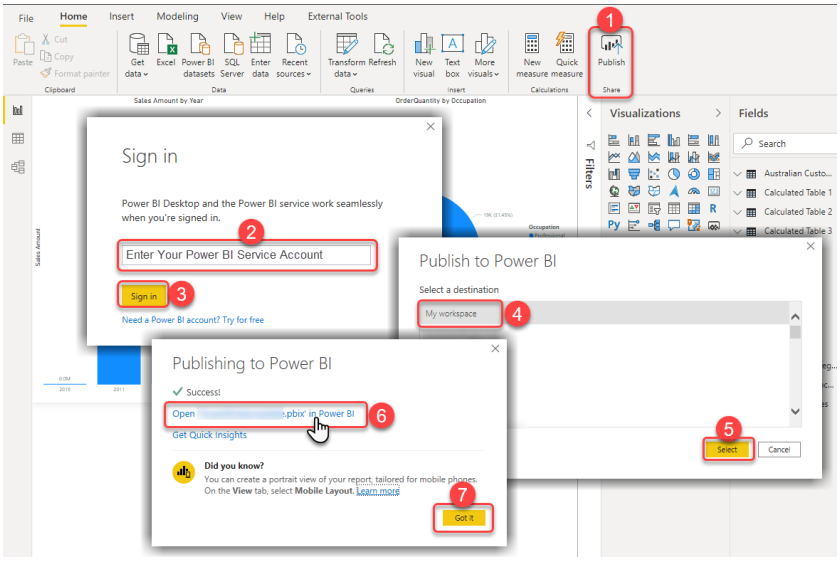


Figure 92: Publishing report from Power BI Desktop to Power BI Service

9. Conclusion

In this guide, you've learnt how to import data from multiple Excel files and join related data to build up a meaningful data model that you can use to visualise and interact with.

By using the Quick Measures feature, you've been able to extend measures with time intelligence; enhancing analysis without writing any code and to provide meaningful insights.

You also know how to get custom visuals and how to remove those that are not necessary anymore.

We hope you enjoyed the guide.

If you need some formal training, you can find out more about our training courses or email us enquiries@theta.co.nz.

10. Appendix

Table of Figures

11. Glossary

Data transformation – converting data from one format/structure to another format/structure

Visualise – graphical representation of data

Shaping/shape – also known as wrangling or manipulation. The process of transforming and mapping data from one "raw" data form into another format so that it is more appropriate/valuable for analytical purposes

Power Query Editor – a dedicated window that facilitates and displays your data connections and transformations you apply

Report Canvas – a blank canvas area where visualisations are placed

One-to-Many – a type of cardinality that refers to the relationship between two entities A and B in which an element of A may be linked to many elements of B, but a member of B is linked to only one element of A. For instance, think of A as product, and B as sales

Auto detected – when loading data from different sources, Power BI Desktop will automatically detect or create relationships between two or more tables

Quick measures - lets you quickly create new measures (calculations) without coding based on measures and numerical columns in your table

Dimension – descriptive data that adds context to transactional (fact) data e.g. year/month/date, location, customer and product. They are in the filters/columns/rows field.

Fact – transactional data that is generally aggregated in report visuals, e.g. sum of amount. They are in the values field.

enquiries@theta.co.nz

theta.co.nz

0800 484 382