ACTBuilder

ASM Talk

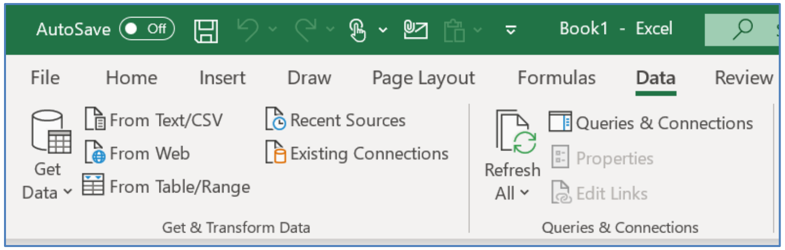
INTRODUCTION OF POWER QUERY IN EXCEL (20.04.2020)

Agenda

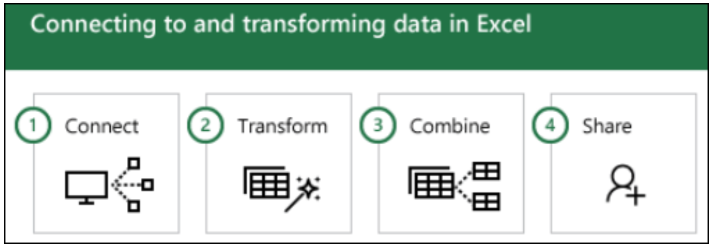
- What are Power Query & M Language?
- Connect to Data Files
- Data Manipulation I – Built in Functionalities
- Data Manipulation II – Self-defined Calculations

Case Study used in this ASM Talk: Claim experience study on medical products, i.e. loss ratio & incidence rate.

What is Power Query?



"Get & Transform Data" section under Excel's Data tab



Connecting to and transforming data in Excel

1 Connect

2 Transform

3 Combine

4 Share

Image Source: https://support.office.com/en-us/article/power-query-overview-and-learning-ed614c81-4b00-4291-bd3a-55d80767f81d

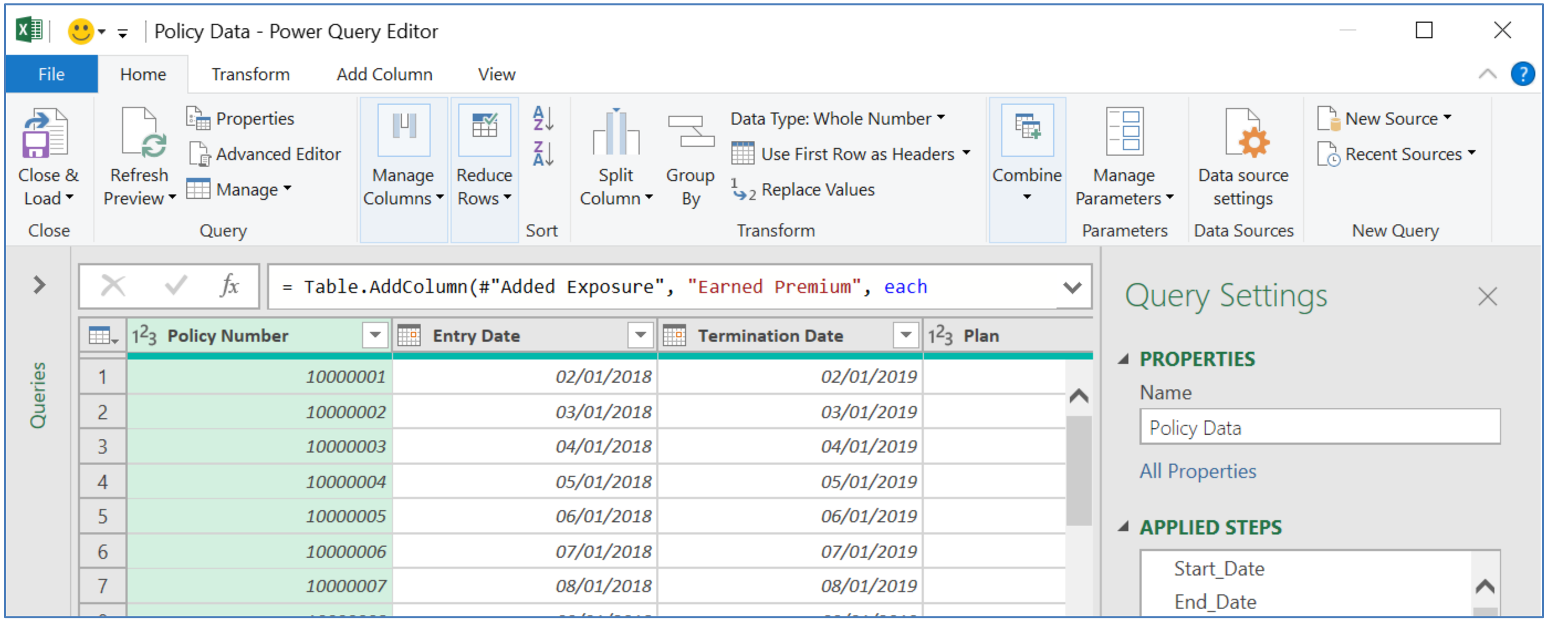
- Microsoft Power Query is a business intelligence tool available in Excel that allow users to connect data from various sources and clean, transform and reshape data in order to meet our data analysis needs.

- Power Query is available as add-ins for Excel 2010 & 2013 (need to download and install). Starting from Excel 2016, it was renamed to "Get & Transform" and appears in the Data Tab – no longer need to perform additional installation.

- How does Power Query works?

  - **Connect** - Connect to data files, as Tables.
  - **Transform** – Shape the Tables in accordance to your needs, without altering the original source data.
  - **Combine** – Create consolidated tables from multiple data sources, which is commonly known as "table joining". Furthermore, you can insert new columns to additional calculations.
  - **Share** – Save your queries and load the results to Excel as Tables. If your queries are setup to allow for "reuse", just change the run parameters and refresh the tables available in the worksheets.

What is Query Editor?



```
= Table.AddColumn(#"Added Exposure", "Earned Premium", each
```

	Policy Number	Entry Date	Termination Date	Plan
1	10000001	02/01/2018	02/01/2019	
2	10000002	03/01/2018	03/01/2019	
3	10000003	04/01/2018	04/01/2019	
4	10000004	05/01/2018	05/01/2019	
5	10000005	06/01/2018	06/01/2019	
6	10000006	07/01/2018	07/01/2019	
7	10000007	08/01/2018	08/01/2019	

Query Settings

PROPERTIES

Name: Policy Data

All Properties

APPLIED STEPS

- Start_Date
- End_Date

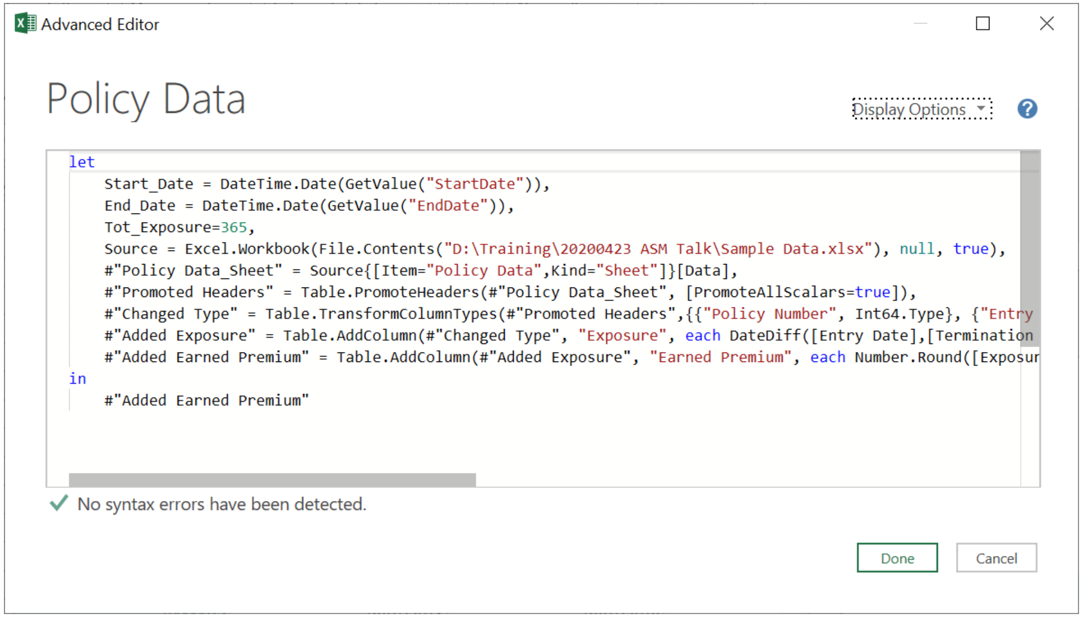
- To transform & combine data files, we need to amend or create new queries using Query Editor. Under Applied Steps section, we can see the step-by-step data manipulation done on a query – which some of the steps are automatically created by Power Query!

- There are two ways to transform & combine data files:

  - Use build-in functions under Home, Transform & Add Column tabs. Power Query will enter relevant codes in the query (similar to how Record Macro works in Excel VBA – we may not remember how to write VBA codes for paste value, right?)

  - Use Advanced Editor under View tab to manually required codes.

What is M Language?



Policy Data

```
let
    Start_Date = DateTime.Date(GetValue("StartDate")),
    End_Date = DateTime.Date(GetValue("EndDate")),
    Tot_Exposure=365,
    Source = Excel.Workbook(File.Contents("D:\Training\20200423 ASM Talk\Sample Data.xlsx"), null, true),
    #"Policy Data_Sheet" = Source{[Item="Policy Data",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(#"Policy Data_Sheet", [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Policy Number", Int64.Type}, {"Entry
    #"Added Exposure" = Table.AddColumn(#"Changed Type", "Exposure", each DateDiff([Entry Date],[Termination
    #"Added Earned Premium" = Table.AddColumn(#"Added Exposure", "Earned Premium", each Number.Round([Exposur
in
    #"Added Earned Premium"
```

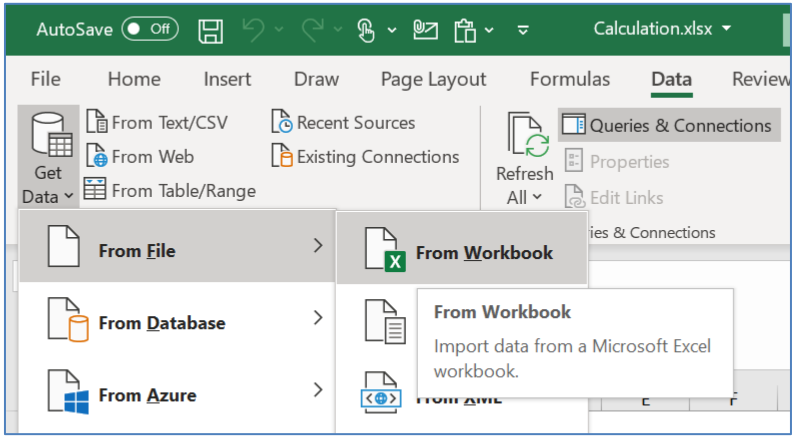
✓ No syntax errors have been detected.

- You can find out codes used by a query from the formula bar or Advanced Editor. Normally, it is insufficient for actuarial users to solely rely on the built-in functions in the Power Query.

- Programming language used in Power Query is normally known as “M Language”. It is important to take note of the following characteristics:

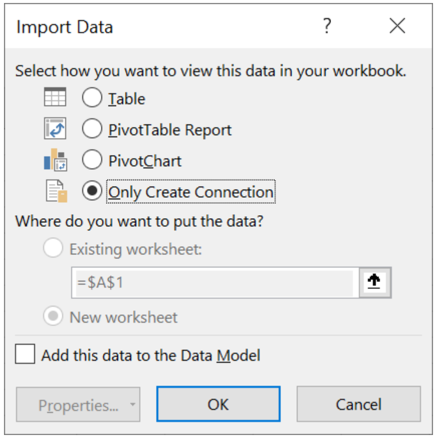
  - Need to included “**let…in…**” pairing in a query that involves tables.
  - Codes, variables & built-in functions are **case-sensitive**.
  - Each line of codes (except the last line of codes before “in”) need to be ended by a “**,**” before entering a new line.

How to Connect to Data?



From Workbook

Import data from a Microsoft Excel workbook.



Import Data

Select how you want to view this data in your workbook.

○ Table

○ PivotTable Report

○ PivotChart

● Only Create Connection

Where do you want to put the data?

○ Existing worksheet:

=A1

○ New worksheet

□ Add this data to the Data Model

- To connect to a standalone data file, go to Data tab > Under Get & Transform Data section, click Get Data > Click From File > Select the option as per your data file type > select your data file & click Open.

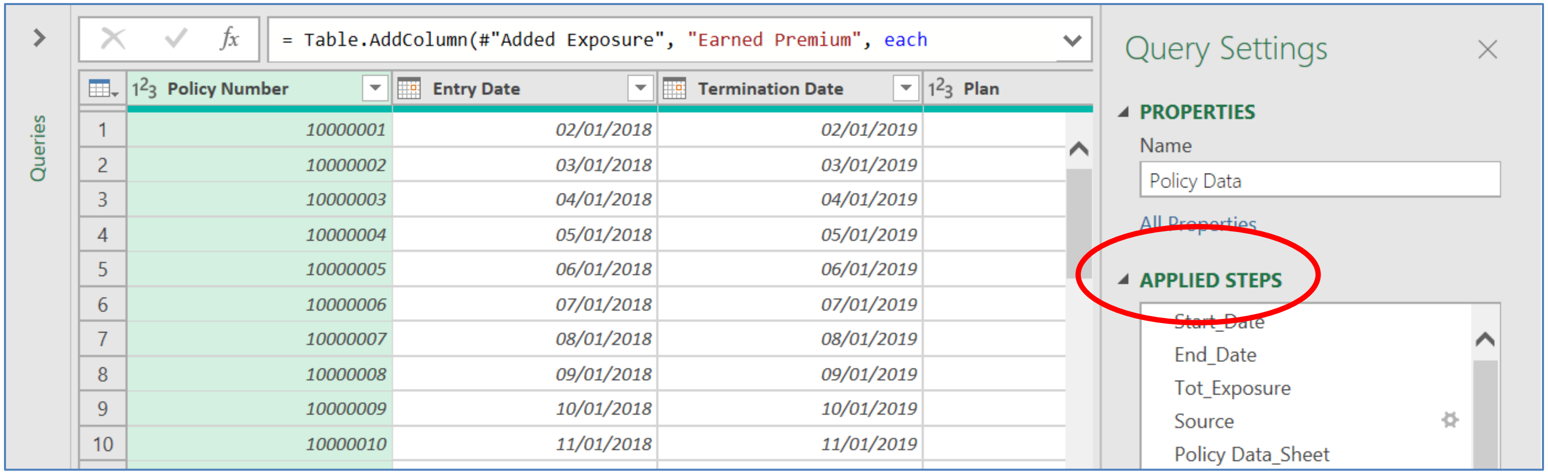
- If you connect to an Excel file, you need to select the worksheet which store the required data. Select “Load To” (the drop-down button at the bottom of the dialog box) to display Import Data dialog.

- Normally, we only connect to the required data files instead of loading the entire data set to a worksheet (which occupy a lot of computer resources). Select Only Create Connection > Click OK.

Suggestions for Microsoft Access users:

- If you are an Access user, you will find Power Query has similarities to Access. As an Access file has a 2GB file size limit, it is always recommended to always connect to data files instead of import the entire data set into Access database – furthermore, Access database requires users to do compact & clean up manually.

How Connected Data Look Like?



```
= Table.AddColumn(#"Added Exposure", "Earned Premium", each
```

	Policy Number	Entry Date	Termination Date	Plan
1	10000001	02/01/2018	02/01/2019	
2	10000002	03/01/2018	03/01/2019	
3	10000003	04/01/2018	04/01/2019	
4	10000004	05/01/2018	05/01/2019	
5	10000005	06/01/2018	06/01/2019	
6	10000006	07/01/2018	07/01/2019	
7	10000007	08/01/2018	08/01/2019	
8	10000008	09/01/2018	09/01/2019	
9	10000009	10/01/2018	10/01/2019	
10	10000010	11/01/2018	11/01/2019	

Query Settings

PROPERTIES

Name: Policy Data

All Properties

APPLIED STEPS

- Start_Date
- End_Date
- Tot_Exposure
- Source
- Policy Data_Sheet

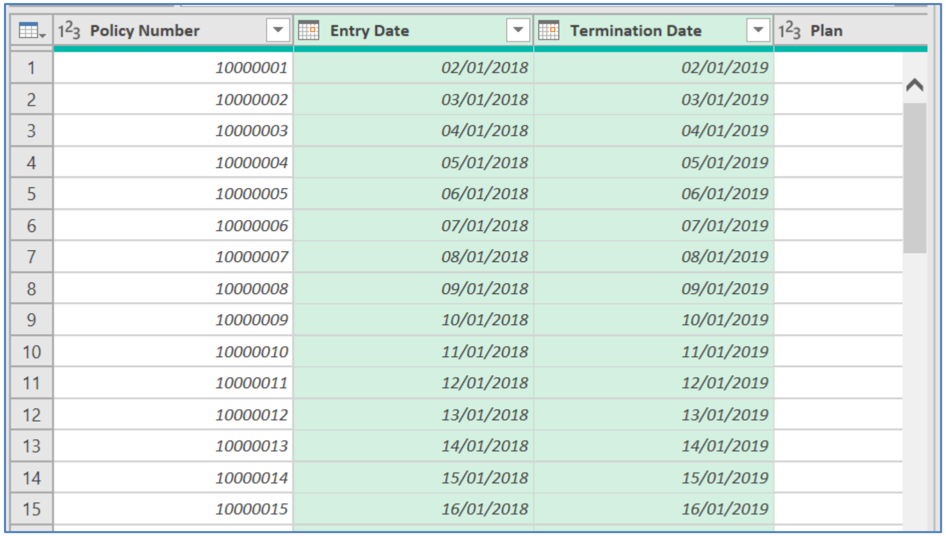
- To view connected data file, go to Data tab > Under Queries & Connections section, click Queries & Connections to display the respective window. Select required query > Right click to select Edit. Excel will display Query Editor.

- In the Query Editor, you can view the values of the connected data file. However, you are unable to alter any values like you normally do it Excel.

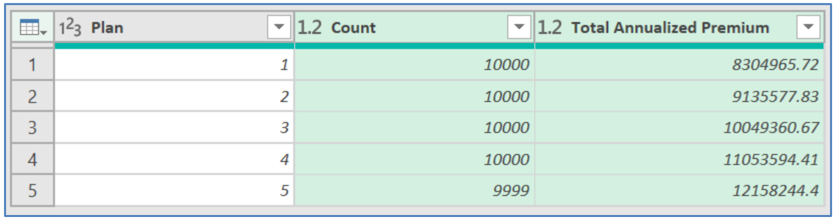
- Under Query Settings window > Applied Steps section, you can click every step to view actions applied on the table, after doing all precedent steps.

- It is important to note that Power Query manages steps by fields / columns, instead of individual "cell" – In Excel, we need to enter formulas into every cell; whereas for Power Query, we are defining formulas by columns.

Data Manipulation I: Built in Functionalities



	1²3 Policy Number	Entry Date	Termination Date	1²3 Plan
1	10000001	02/01/2018	02/01/2019	
2	10000002	03/01/2018	03/01/2019	
3	10000003	04/01/2018	04/01/2019	
4	10000004	05/01/2018	05/01/2019	
5	10000005	06/01/2018	06/01/2019	
6	10000006	07/01/2018	07/01/2019	
7	10000007	08/01/2018	08/01/2019	
8	10000008	09/01/2018	09/01/2019	
9	10000009	10/01/2018	10/01/2019	
10	10000010	11/01/2018	11/01/2019	
11	10000011	12/01/2018	12/01/2019	
12	10000012	13/01/2018	13/01/2019	
13	10000013	14/01/2018	14/01/2019	
14	10000014	15/01/2018	15/01/2019	
15	10000015	16/01/2018	16/01/2019	



	1²3 Plan	1.2 Count	1.2 Total Annualized Premium
1	1	10000	8304965.72
2	2	10000	9135577.83
3	3	10000	10049360.67
4	4	10000	11053594.41
5	5	9999	12158244.4

- Power Query has many user-friendly built-in functionalities that allow users to perform data manipulations without needing to understand M Language.

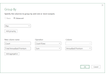
- Examples of built-in functionalities include filtering, grouping (such as to count rows and sum selected fields).

How do we summarize a table?

- Open required query > Go to Home tab > Under Transform section, click Group by.

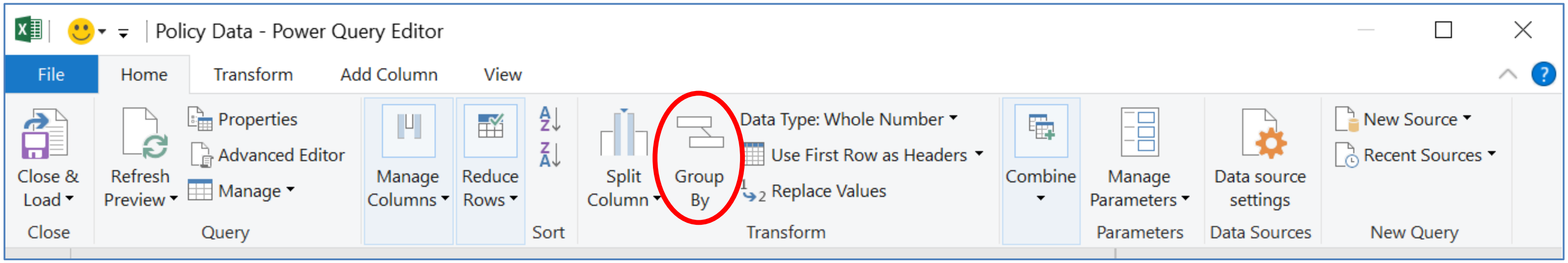
- In Group By dialog > Select Advanced > Select the field which the data should be summarized by. Click Add Grouping if there are more than 1 field > Define how the aggregation to be done. Click Add Aggregation if you need to summarize more than 1 field > Click OK.

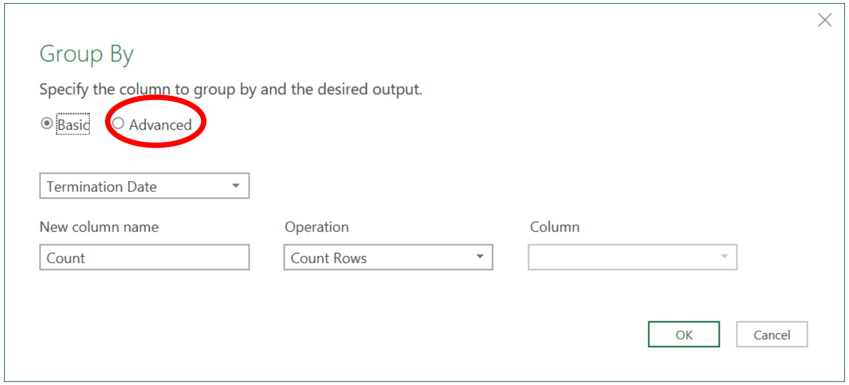
- To undo the grouping step, go to Applied Steps > Select the grouping step > Click × next to the selected step.



Data Manipulation I: Built in Functionalities (Group By)

STEP 1





Group By

Specify the column to group by and the desired output.

● Basic ○ Advanced

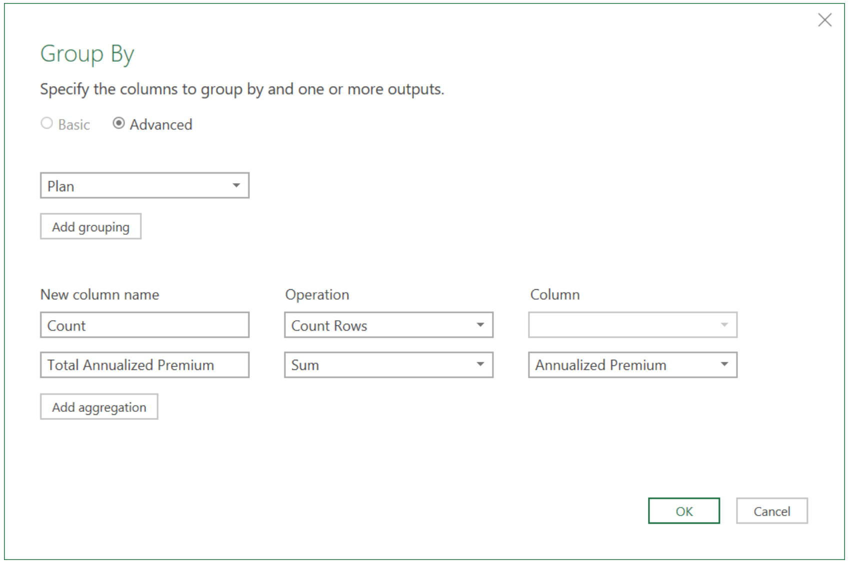
Termination Date

New column name	Operation	Column
Count	Count Rows	

OK Cancel

STEP 2

STEP 3



Group By

Specify the columns to group by and one or more outputs.

○ Basic ● Advanced

Plan

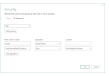
Add grouping

New column name	Operation	Column
Count	Count Rows	
Total Annualized Premium	Sum	Annualized Premium

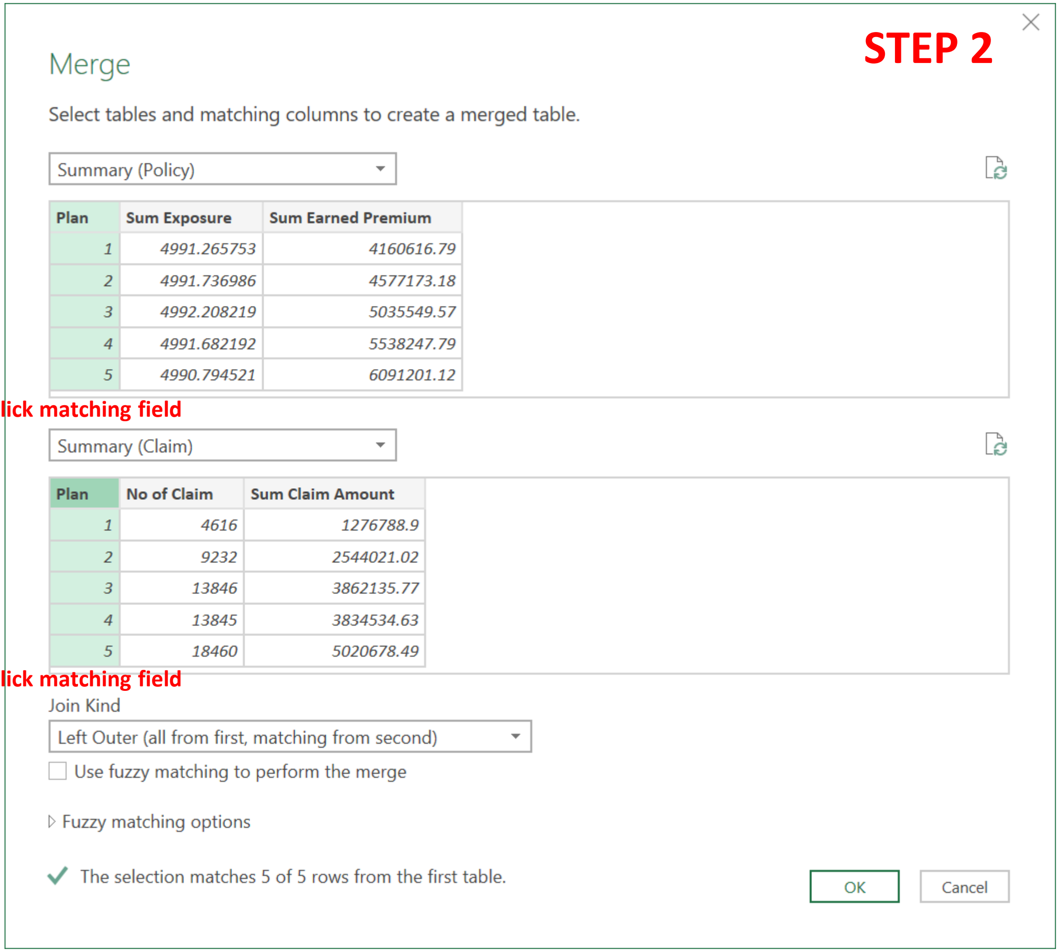
Add aggregation

OK Cancel

ACTBuilder



Data Manipulation I:
Built in Functionalities (Merge)



STEP 2

Merge

Select tables and matching columns to create a merged table.

Summary (Policy) ▾

Plan	Sum Exposure	Sum Earned Premium
1	4991.265753	4160616.79
2	4991.736986	4577173.18
3	4992.208219	5035549.57
4	4991.682192	5538247.79
5	4990.794521	6091201.12

Click matching field

Summary (Claim) ▾

Plan	No of Claim	Sum Claim Amount
1	4616	1276788.9
2	9232	2544021.02
3	13846	3862135.77
4	13845	3834534.63
5	18460	5020678.49

Click matching field

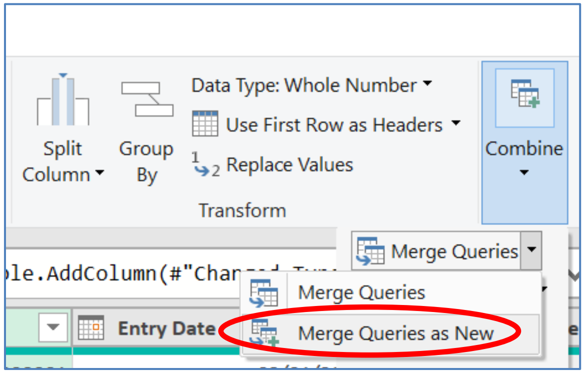
Join Kind

Left Outer (all from first, matching from second) ▾

□ Use fuzzy matching to perform the merge

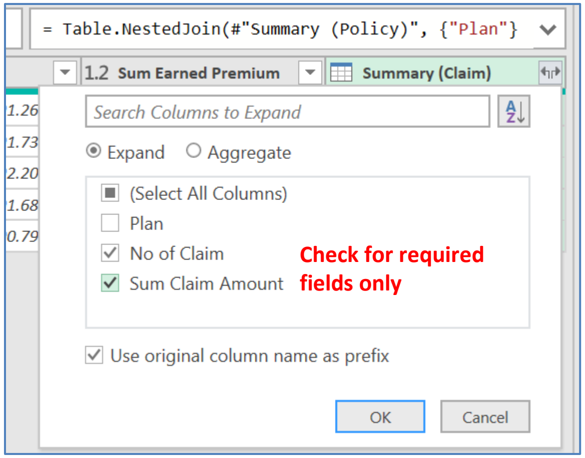
▷ Fuzzy matching options

✓ The selection matches 5 of 5 rows from the first table.



```
le.AddColumn(#"Char
```

STEP 1



```
= Table.NestedJoin(#"Summary (Policy)", {"Plan"}
```

● Expand ○ Aggregate

■ (Select All Columns)

□ Plan

☑ No of Claim

☑ Sum Claim Amount

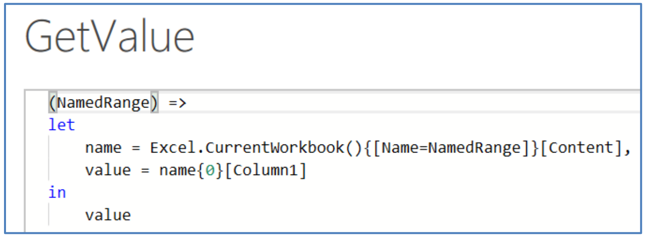
Check for required fields only

☑ Use original column name as prefix

STEP 3

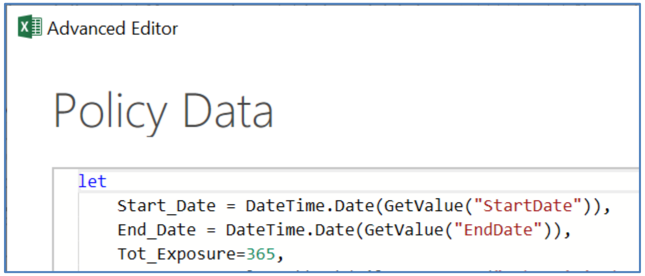
ACTBuilder

Data Manipulation II:
Self-defined Calculations



GetValue

```
(NamedRange) =>
let
    name = Excel.CurrentWorkbook(){[Name=NamedRange]}[Content],
    value = name{0}[Column1]
in
    value
```



Advanced Editor

Policy Data

```
let
    Start_Date = DateTime.Date(GetValue("StartDate")),
    End_Date = DateTime.Date(GetValue("EndDate")),
    Tot_Exposure=365,
```

- Similar to other programming languages, we can define variables in the codes that can be used in the subsequent steps.
- We can assign a constant to a variable or link it to a user-defined function.
- It is important that data types in Power Query may be different from Excel. For example, difference between two dates are considered as *Duration* Type.

How to make the queries "reusable"?

- Instead of hardcoding the run parameters (e.g. calculation dates) in coding, we can create a user-defined function to read values from a worksheet of Excel file. (Please note that user-defined functions do not need to have "let…in…" pairings.
- To do so, we need to assign names to the cells that store run parameters in advance.
- If the run parameters are date values in the worksheet, they are read as *DateTime* format by Power Query. To retrieve the date value, we need to use *DateTime.Date* function.

Disclaimer

- The contents of this document are only for the reference of participants attending ASM Mini Sharing on 09.04.2019. Unauthorized circulation or duplication of any part of this document to any unauthorized third party is prohibited.
- The copyrights of this document are owned by ACTBuilder Actuarial Solutions Sdn. Bhd.
- ACTBuilder Actuarial Solutions Sdn. Bhd is not responsible for unauthorized use of any part of the contents of this document.

Contact Info:
ACTBuilder Actuarial Solutions Sdn. Bhd. (1108112-P) | Lim Chee Beng | CheeBeng.Lim@actbuilder.com.my