Data Action Lab

Sysabee DAVHILL Group IDLEWYLD

Enseignement univ.: 50+ cours; **ateliers:** 40+; **projets:** 60+; **expérience combinée:** 35+ années.
Co-entreprise pré-qualifiée à la liste des fournisseurs I.A. du GdC – EN578-180001/A (1ière bande).

Nouveau catalogue de formation approfondie disponible sur la toile au **data-action-lab.com**

Combined experience: 50+ university courses, 100+ corporate workshops, 60+ projects, 35+ years.
Joint venture qualified for GoC A.I. Source List – EN578-180001/A (Band 1).

New advanced training catalogue available at **data-action-lab.com**

Data Action Lab

What We Do

- Training and long courses
- Workshops and short courses
- Knowledgebase curation
- Data labs

Training Paths

- Data Novice
- Data Engineer
- Data Practitioner
- Data Scientist
- Data Manager
- Data Champion

Training Learning Interests

- Visualization and Dashboards
- Introduction to Data Science
- Advanced Data Science
- Machine Learning Toolbox
- Spotlight on Classification
- Spotlight on Clustering
- Text Analysis
- Special Topics in AI and DS
- Hands on Data Analysis
- Data Strategy and Governance

Expertise

Business intelligence

Data visualization design

Data analytics and data science

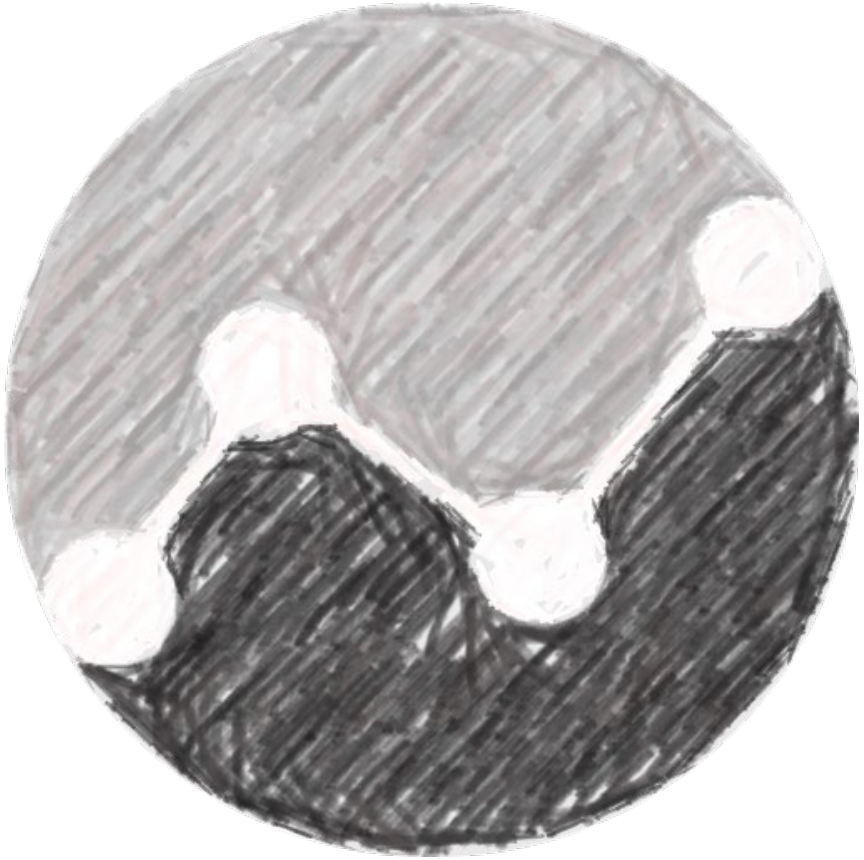
Data engineering

Advanced statistics and machine learning

Artificial and augmented intelligence

Process and systems modeling

Software implementation and integration



Data Action Lab

Vision

Provide a space for data consumers, producers, practitioners, scientists and champions to make a place for themselves in the digital world.

Provide paths for education and enrichment for all these groups.

Keep pace with developments in the digital arena and keep Data Action Lab participants moving and aligned with these relevant developments.

Provide just-in-time learning opportunities for Data Action Lab members, focusing on their specific challenges and skillsets.

PBI-2: POWER BI – BEYOND THE BASICS

DATA ACTION LAB – POWER BI SERIES

COURSE OVERVIEW

Course #: PBI-2 **Duration:** 1.5 day

Course Title: Power BI – Beyond the BAsics

Description:

1. Introduction
   - Next steps in using Power BI?
2. Importing Data
   - Pulling data from Folders, PDFs and Web Scraping (and reviewing a few others)
3. Data Wrangling
   - Merging tables, Pivoting Data, Parameters

COURSE OVERVIEW

Course #: PBI-2 **Duration:** 1.5 day
Course Title: Power BI – Beyond the Basics

Description:

4. Data Modeling
   - Cross Filter Direction, Cardinality, Data Model Types
5. Best Practice Alerts!
   - Declaring Variables, Boolean Logic
6. Doing more with DAX
   - Calculated Tables, Dealing with Nested IF, Ranking, Filtering, GoC Fiscal Years, Advanced Formatting, What if Parameters

INTRODUCTION

PBI-2: POWER BI – BEYOND THE BASICS

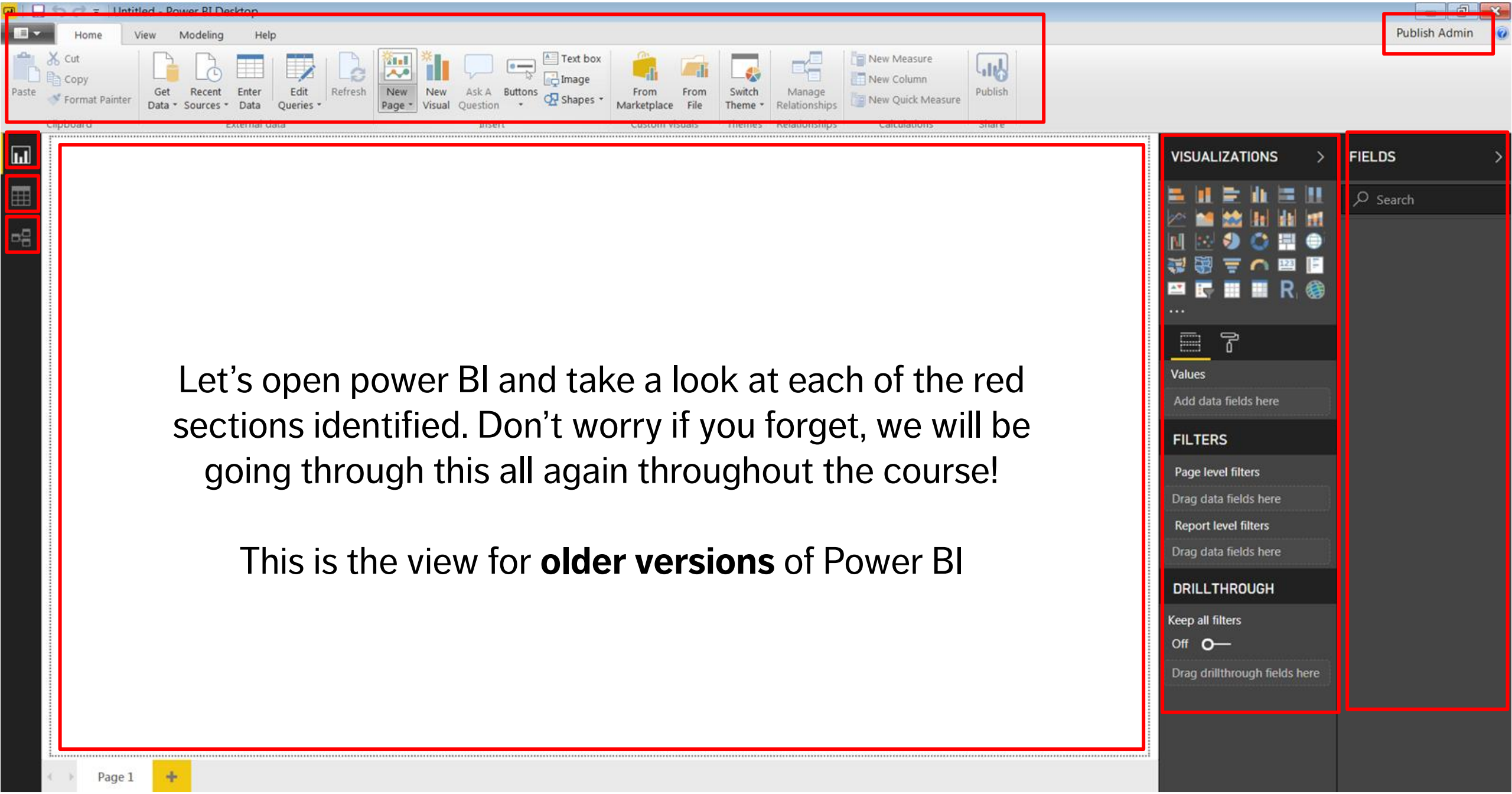
INTRODUCTION

In the “Getting to know Power BI Course” we identified and worked through basic PBI functionality that complimented your GoC work environment and requirements.

We now need to dig a little deeper into PBI functionality to be able to perform some more complex, real world functions and to streamline how we use PBI on a day to day basis.

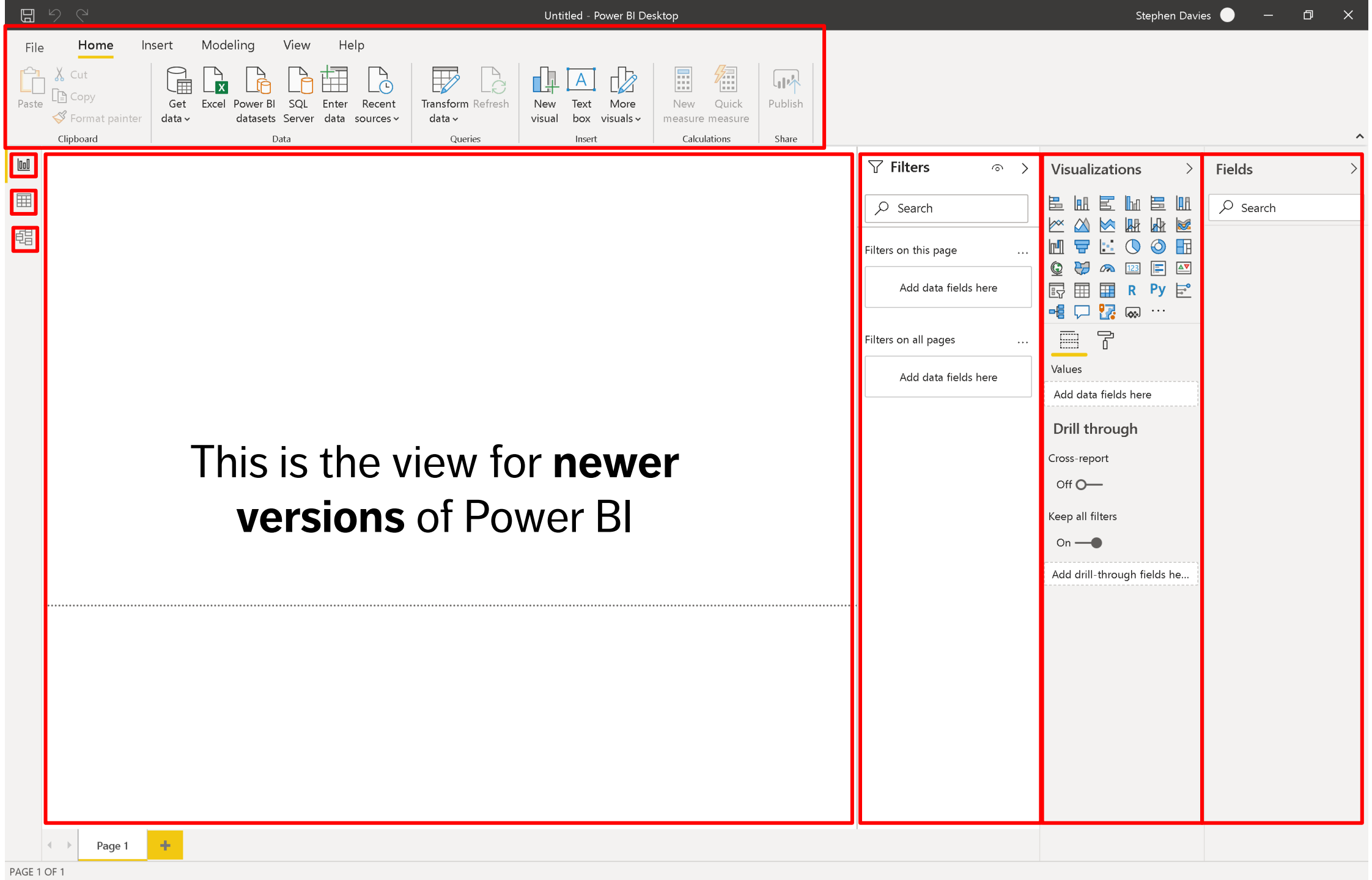
In summary we are going to

1. Be more flexible in the data we import
2. Discover more efficient ways of manipulating that data
3. Gain a deeper understanding of PBI analysis and visualization



Let's open power BI and take a look at each of the red sections identified. Don't worry if you forget, we will be going through this all again throughout the course!

This is the view for **older versions** of Power BI


Untitled - Power BI Desktop
Stephen Davies
File
Home
Insert
Modeling
View
Help
Paste
Cut
Copy
Format painter
Clipboard
Get data
Excel
Power BI datasets
SQL Server
Enter data
Recent sources
Data
Transform data
Refresh
Queries
New visual
Text box
More visuals
Insert
New measure
Quick measure
Calculations
Publish
Share
This is the view for newer versions of Power BI
Filters
Search
Filters on this page
Add data fields here
Filters on all pages
Add data fields here
Visualizations
Values
Add data fields here
Drill through
Cross-report
Off
Keep all filters
On
Add drill-through fields he...
Fields
Search
Page 1
PAGE 1 OF 1

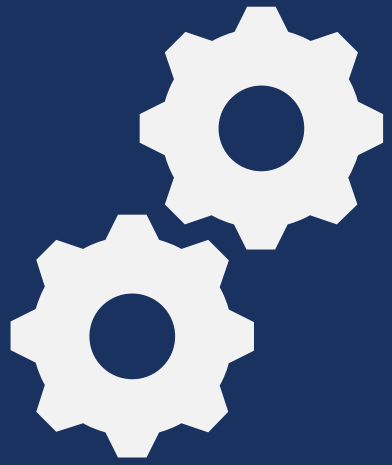
IMPORTING DATA

PBI-2: POWER BI – BEYOND THE BASICS

REMINDER!
WHY WE CLEAN DATA

BINGO!!!!

random'missing' values	outliers	values'outside'of' expected'range'4 numeric	factors' incorrectly/iconsiste ntly'coded	date/time'values'in' multiple'formats
impossible'numeric' values	leading'or'trailing' white'space	badly'formatted' date/time'values	non4random'missing' values	logical' inconsistencies' across'fields
characters'in' numeric'field	values'outside'of' expected'range'4 date/time	DCB!	inconsistent'or'no' distinction'between' null,'0,not'available,' not' applicable,missing	possible'factors' missing
multiple'symbols' used'for'missing' values	???	fields'incorrectly' separated'in'row	blank'fields	logical'iconsistencies' within'field
entire'blank'rows	character'encoding' issues	duplicate'value'in' unique'field	non4factor'values'in' factor	numeric'values'in' character'field



SETTING UP OUR PBIX

SETTING UP OUR PBIX

The next couple of slides are repeats of importing the Accounting and Projects data from the “Getting to know Power BI” course.

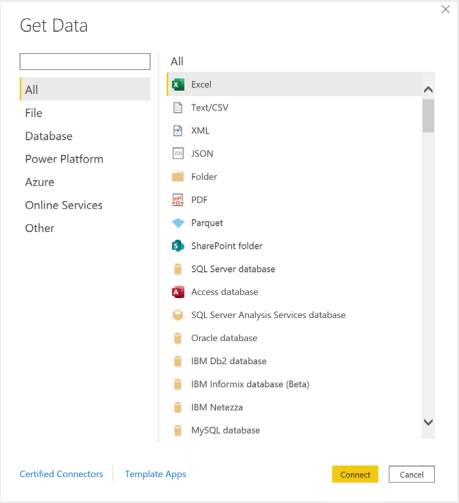
We will be using this data and adding to it significantly throughout this course.

If you have the file from the previous course it is better to start a new one from scratch.

SETTING UP OUR PBIX

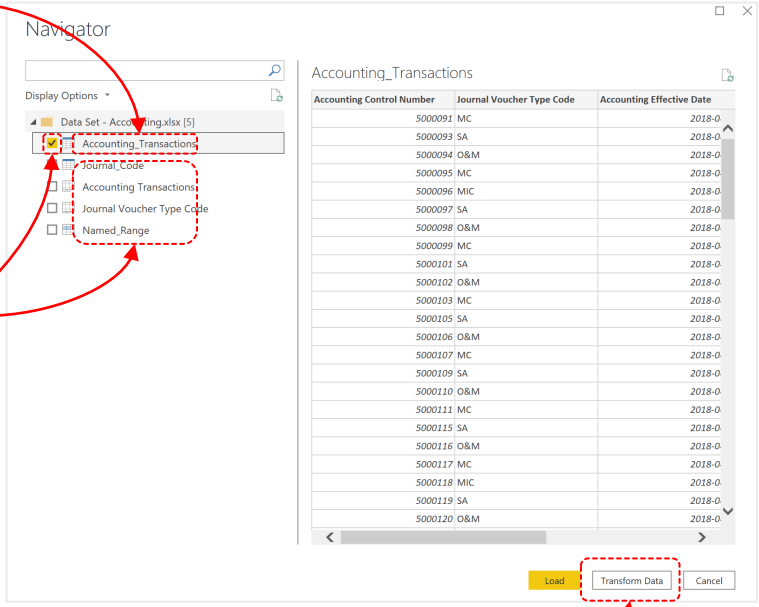
Let's get some data into Power Query (follow the instructor):

1. Open Power BI.
2. Close the yellow "Hello" screen (we will come back to this later).
3. In the top "Home" tab you will see a "Get data" button. Click on that and the instructor will walk through several different options you have.
4. Once we have explored a few options select "Excel" .
5. Click on the file you saved called "Data Set - Accounting.xlsx" and click open.
6. PAUSE while the instructor tells you about all the different ways you can import the data (next slide) – *ignore this line for now, it was for the earlier course*



SETTING UP OUR PBIX

7. Click on the first TABLE named “Accounting_Transactions” but don’t select the check mark (yet).
8. You will see a summary of the data when you do.
9. Click on the other options to get a summary of that data.
10. Go back to “Acounting_Transactions” and select the checkbox.
11. **DON’T CLICK ON LOAD!!!!!**
12. Instead let's tweak the data a bit, so we will edit the transformation by clicking on “Transform Data”.



SETTING UP OUR PBIX

Follow the instructor to do the following

1. Change the name of the table: Double click (or right click) on the table name and edit to remove the underscore.
2. Change the name of the “Accounting Effective Date” column to “Effective Date” and “Journal Voucher Item Amount” to “Item Amount” (note the new step in the query step box).
3. The instructor will show you where Power BI keeps the “M” language version of the query (this will be re-visited in a more advanced course).
4. Click on “Close and Apply” and the Instructor will take you through Power BI again but this time with data imported. Remember to save your pbix!

SETTING UP OUR PBIX

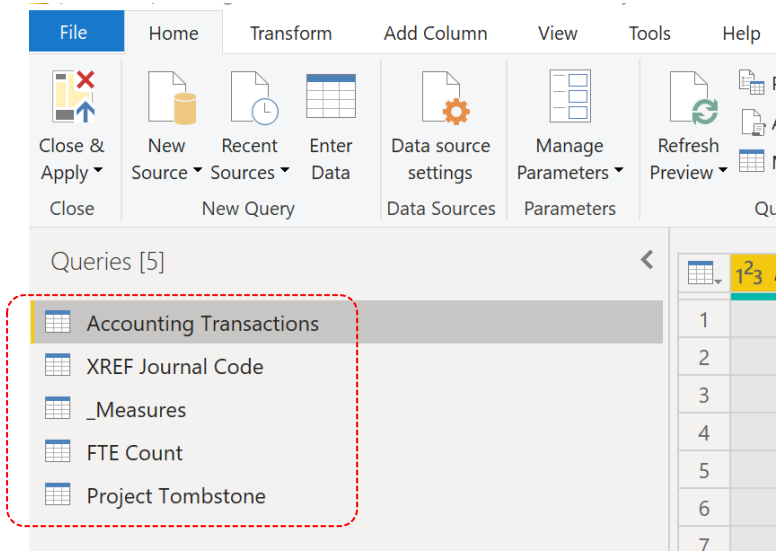
Go back to the same spreadsheet and add in the “Journal Voucher Type Code” tab. Remember to click on “Transform Data” to double check that all of the data looks good. Change the Table Name to “XREF Journal Code”.

- XREF stands for “Cross Reference”
- Cross reference tables are tables that explain data and are not the primary data values we typically evaluate.
- By starting all your Cross Reference tables with XREF we can group them together in Power BI, this will help to keep you sane!

SETTING UP OUR PBIX

The Instructor will guide you through the following steps:

1. Import BOTH tabs (or tables) from the “Data Set - Project.xlsx” (Project_Tombstone & FTE_Count).
2. Change the name of both the tables to remove the underscore (Optional).
3. Review all the columns in both tables to make sure that all the formatting is correct.



SETTING UP OUR PBIX

We have now got the data in the same format as the “Getting to know Power BI” course.

We are not going to do any data modeling just yet as we are going to import more data from different types of sources.

If you hit “Close and Apply” then go to your data model and simply delete all of the connections that Power BI automatically created – we will update those later

IMPORTING FROM FOLDERS

IMPORTING FROM FOLDERS

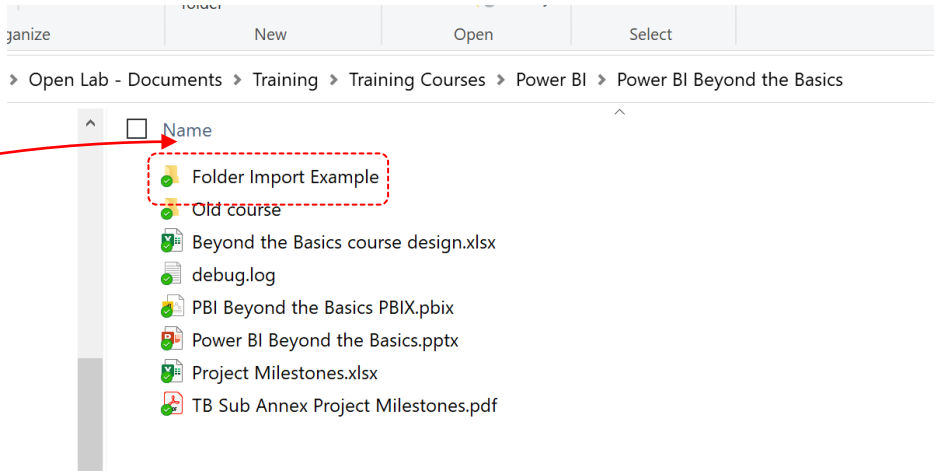
A really useful method of importing data is to upload an entire folder. NOTE we need to be VERY careful how we do this though. In general:

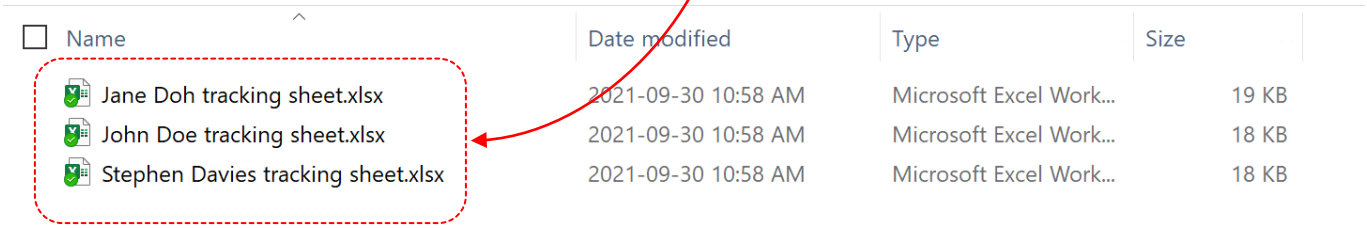
- Only use folders with one type of file (e.g. Excel). We can get around this in Power Query but having a single type makes life easier
- Make sure the format of the files (e.g. Excel Column names) is IDENTICAL or the import will fail

IMPORTING FROM FOLDERS

Follow these steps to setup a folder for you to try importing from:

1. Create a folder on your computer and call it "Folder Import Example"
2. From the course website copy the three Excel "Tracking Sheets" into the folder



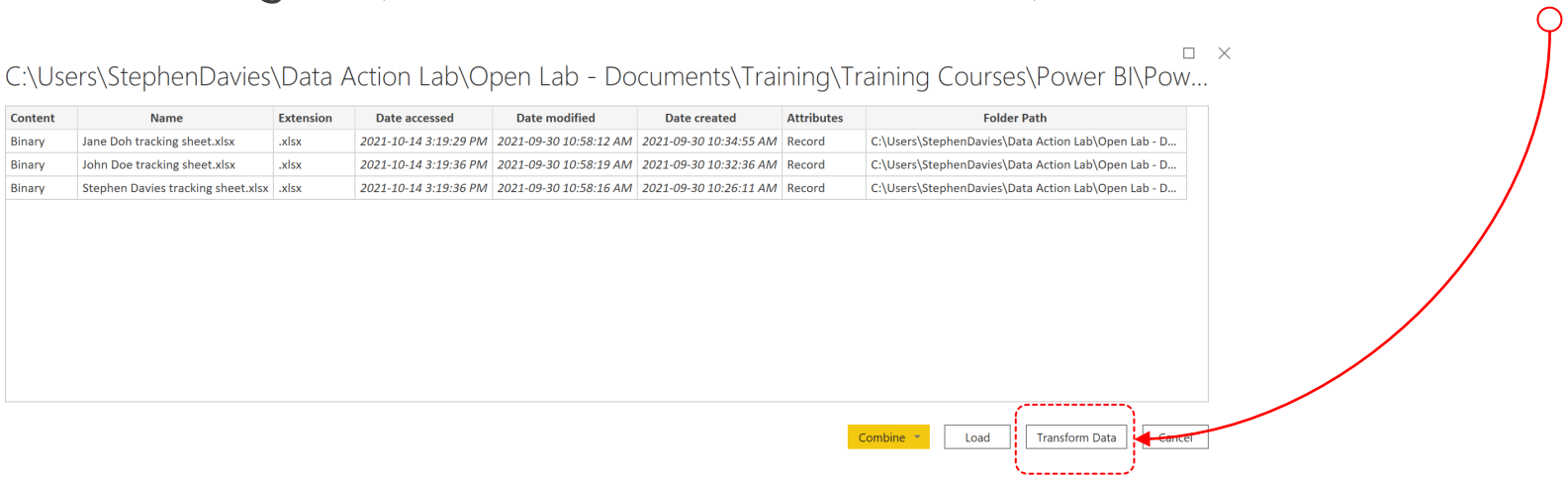


Name	Date modified	Type	Size
Jane Doh tracking sheet.xlsx	2021-09-30 10:58 AM	Microsoft Excel Work...	19 KB
John Doe tracking sheet.xlsx	2021-09-30 10:58 AM	Microsoft Excel Work...	18 KB
Stephen Davies tracking sheet.xlsx	2021-09-30 10:58 AM	Microsoft Excel Work...	18 KB

IMPORTING FROM FOLDERS

To import the data follow the instructor through the following steps:

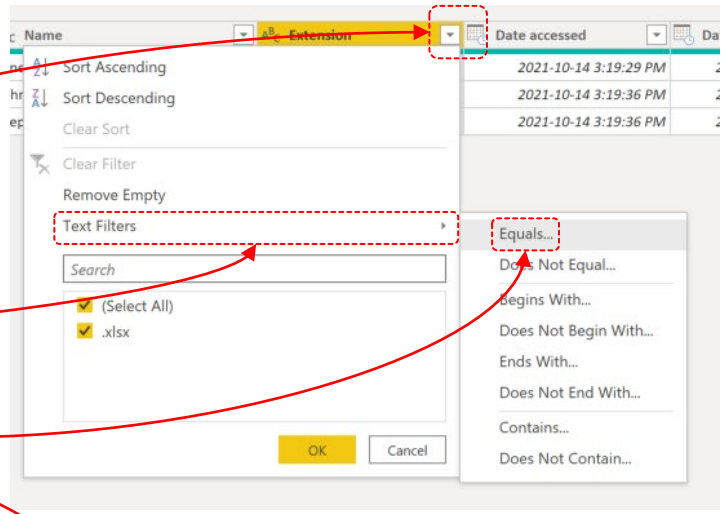
1. Click on “Get Data” and then “Folder”
2. Power BI will ask for a “Folder Path”, click “Browse” and navigate to the folder you created in the previous slide, then select it and hit “Ok”
3. You will then see the following box, do NOT hit Combine or Load, instead hit “Transform Data”

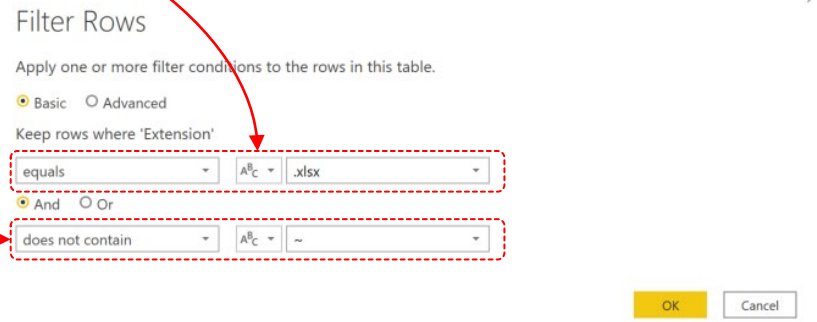


Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder Path
Binary	Jane Doh tracking sheet.xlsx	.xlsx	2021-10-14 3:19:29 PM	2021-09-30 10:58:12 AM	2021-09-30 10:34:55 AM	Record	C:\Users\StephenDavies\Data Action Lab\Open Lab - D...
Binary	John Doe tracking sheet.xlsx	.xlsx	2021-10-14 3:19:36 PM	2021-09-30 10:58:19 AM	2021-09-30 10:32:36 AM	Record	C:\Users\StephenDavies\Data Action Lab\Open Lab - D...
Binary	Stephen Davies tracking sheet.xlsx	.xlsx	2021-10-14 3:19:36 PM	2021-09-30 10:58:16 AM	2021-09-30 10:26:11 AM	Record	C:\Users\StephenDavies\Data Action Lab\Open Lab - D...

IMPORTING FROM FOLDERS

4. We now want to make sure that we only import Excel files and that temporary files (files that are created when a file is opened) are not imported
5. Select the drop down button on Extension, then select "Text Filters" and finally "Equals"
6. In the first line enter ".xlsx" (without the quotations)
7. In the second row select "Does Not Contain" and type in a "~" character (without the quotations) – all temporary files start with this character.
8. Hit "OK". This now means that ONLY non-temporary .xlsx files will be imported

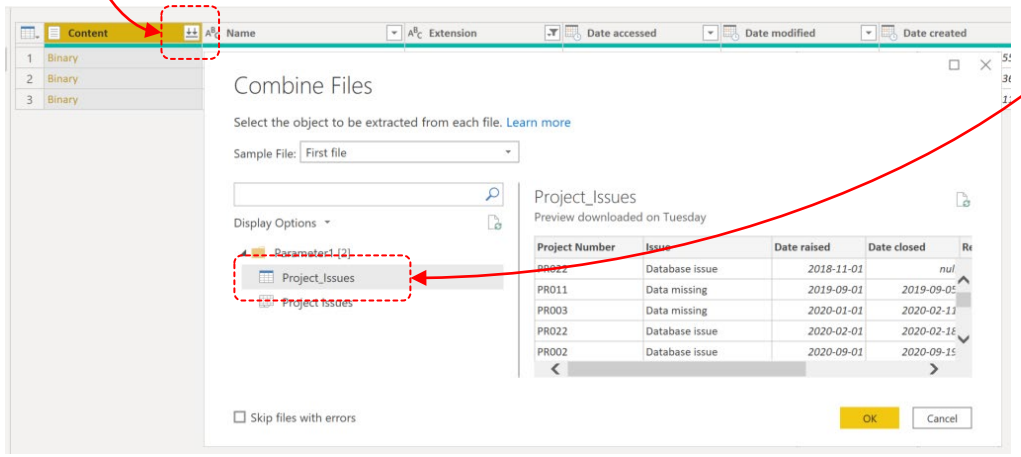


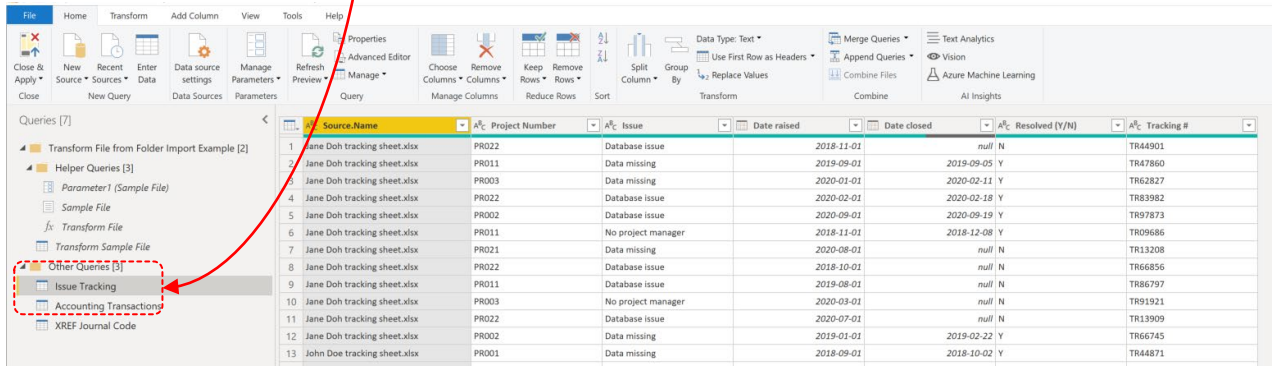


IMPORTING FROM FOLDERS

9. We now need to expand out each of the files into a table. To do this select the two down arrow button in the “Content” column.

10. You will then see a dialog that requires you to select a table for or tab (it is looking at the first file in the folder). Select the “Project_Issues” Table then hit “OK”, you will see the table with a descriptor first column – rename the table “Issue Tracking”





IMPORTING FROM PDF

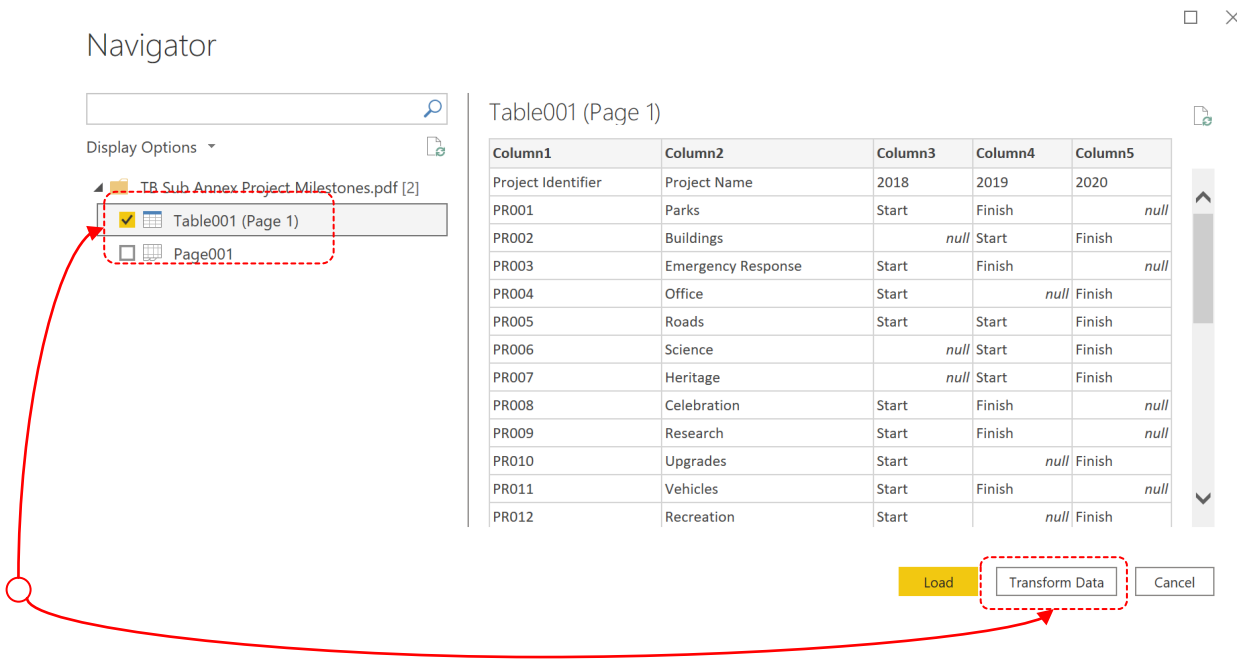
IMPORTING FROM PDF

Sometimes there may be table in a PDF that would be useful to import into a Power BI file. For example a TBSub annex, or a reference table.

The next few steps are highly dependent on how the PDF was setup in the first place but typically extracting data from PDF files is reasonably reliable.

IMPORTING FROM PDF

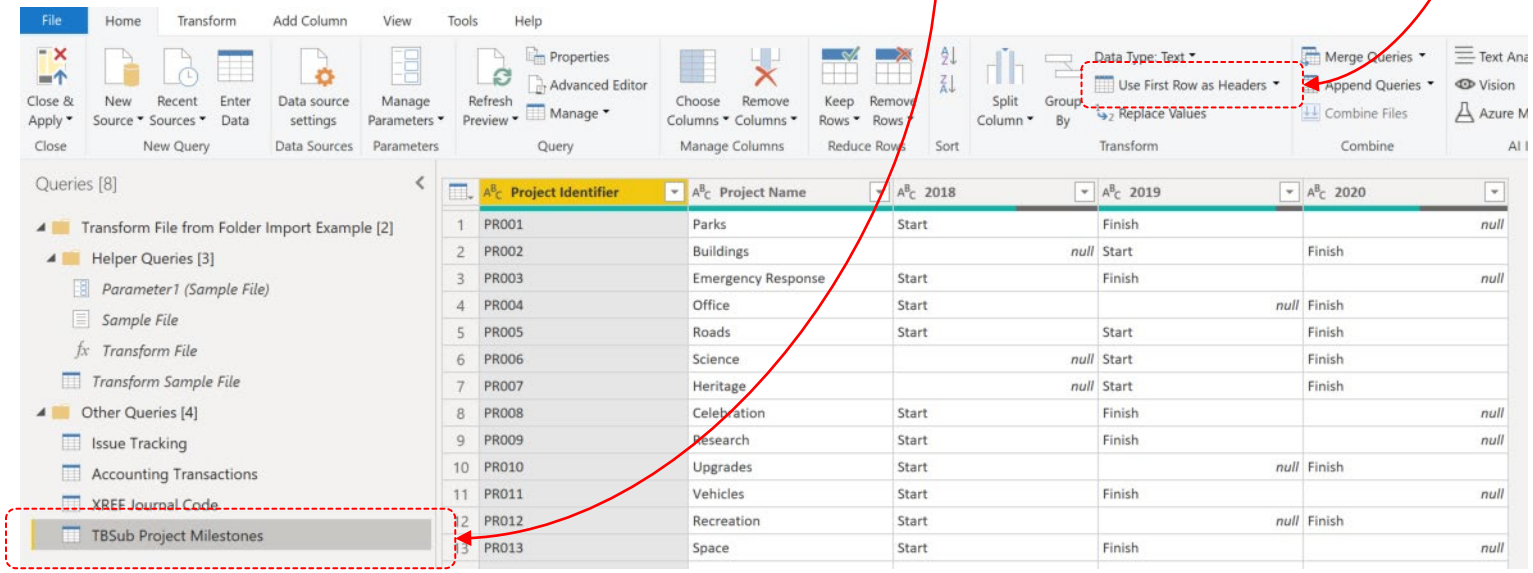
1. Make sure you have the file “TB Sub Annex Project Milestones.pdf” saved onto your computer.
2. Click on “Get Data”, then select “PDF”. Power BI will then prompt you to select the PDF you just saved
3. You will then be presented with load option (in a similar way to importing Excel data), select the Table option and hit “Transform Data”



Column1	Column2	Column3	Column4	Column5
Project Identifier	Project Name	2018	2019	2020
PR001	Parks	Start	Finish	null
PR002	Buildings	null	Start	Finish
PR003	Emergency Response	Start	Finish	null
PR004	Office	Start	null	Finish
PR005	Roads	Start	Start	Finish
PR006	Science	null	Start	Finish
PR007	Heritage	null	Start	Finish
PR008	Celebration	Start	Finish	null
PR009	Research	Start	Finish	null
PR010	Upgrades	Start	null	Finish
PR011	Vehicles	Start	Finish	null
PR012	Recreation	Start	null	Finish

IMPORTING FROM PDF

4. The imported table does not recognize the first row as columns so select "Use First Row as Headers"
5. Finally rename the query "TBSub Project Milestones"



IMPORTING FROM THE WEB

IMPORTING FROM WEBSITES (WEBSCRAPING)

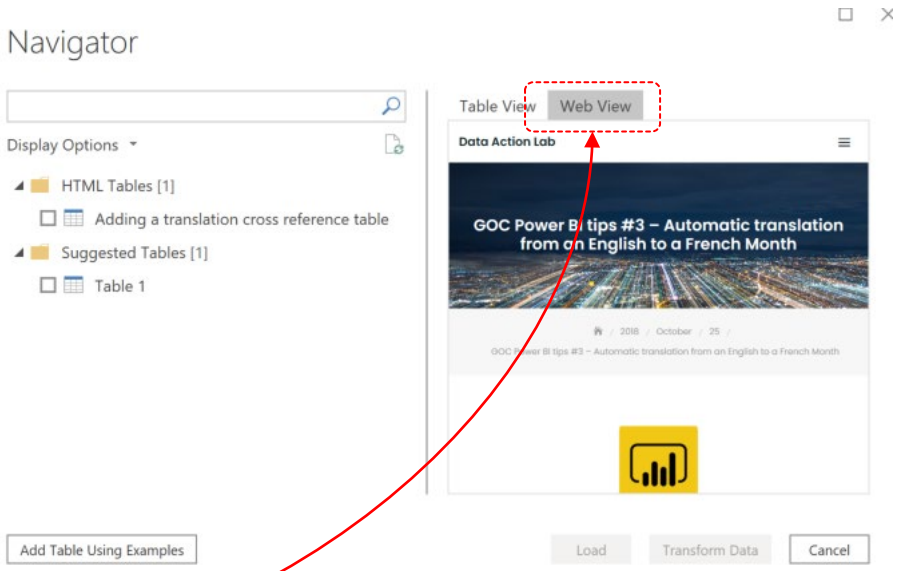
Sometimes there are useful data sources posted on webpages (intranet, reference website etc).

We can link to these in Power BI, but we do need to be careful that the webpage does not change URL or get moved

IMPORTING FROM WEBSITES (WEBSCRAPING)

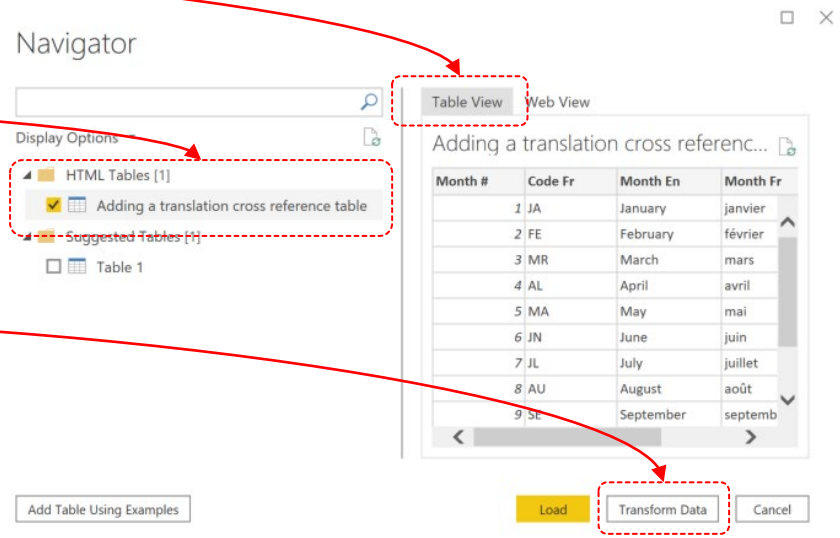
To import from a webpage follow the instructor through the following steps:

1. Make sure you have an internet connection and you can navigate to the following URL >> https://www.data-action-lab.com/2018/10/25/goc-power-bi-tips-3-automatic-translation-of-a-french-month
2. Click on “Get data” and then click on “Web”
3. Paste the URL above into the dialog box and click on “OK”
4. Power BI will give the option to “Web View” to make sure you have selected the correct URL



IMPORTING FROM WEBSITES (WEBSCRAPING)

5. Select “Table View”
6. Under “HTML Tables” select “Adding a translation cross reference table”
7. Finally let’s hit “Transform Data” to check what we have imported and change the table name to “XREF Date Translation”



ADDING IN THE FINAL TABLES

ADDING IN THE FINAL TABLES

We now need to add in a XREF table to link all of the Project Identifier columns together. We will do this in the same way we did it in the “Getting to know Power BI” course by appending tables together, this time we have more tables to append.

The instructor will walk you quickly through the steps required to do this, but you can also refer back to the previous course.

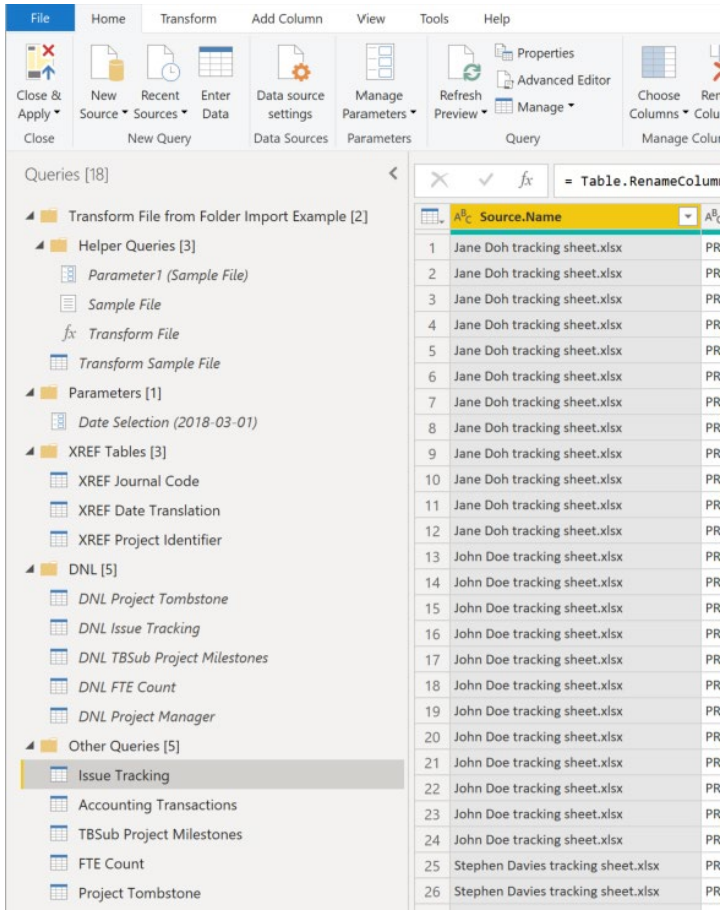
ADDING IN THE FINAL TABLES

1. In the “Issue Tracking” query, change the column “Project Number” to “Project Identifier”, remember we need identical column names to do a Table Append
2. Duplicate all of the data tables and remove all of the columns except the “Project Identifier” columns. Rename the duplicated “Accounting Transactions (2)” table “XREF Project Identifier” and all of the other duplicated queries “DNL [query name]” (e.g. “DNL Issue Tracking”)
3. Use the “Append Queries” function (exactly the same as the previous course) to append ALL of the DNL queries into the “XREF Project Identifier” query
4. In the “XREG Project Identifier” query select “Remove Rows” and then select “Duplicates” then repeat for “Blank Rows”
5. Finally select each “DNL” query, right click then un-check “Enable Load”

TIDYING THINGS UP

TIDYING THINGS UP

- To keep things organized in Power Query we can make groups.
- These are pretty much the same as folders
- To create a new group go to the Queries section, right click and select “New group”
- You can now drag any items you want to these groups



DATA WRANGLING

PBI-2: POWER BI – BEYOND THE BASICS

MERGING TABLES

MERGING TABLES

We may encounter a situation where we want to merge two (or more) tables together.

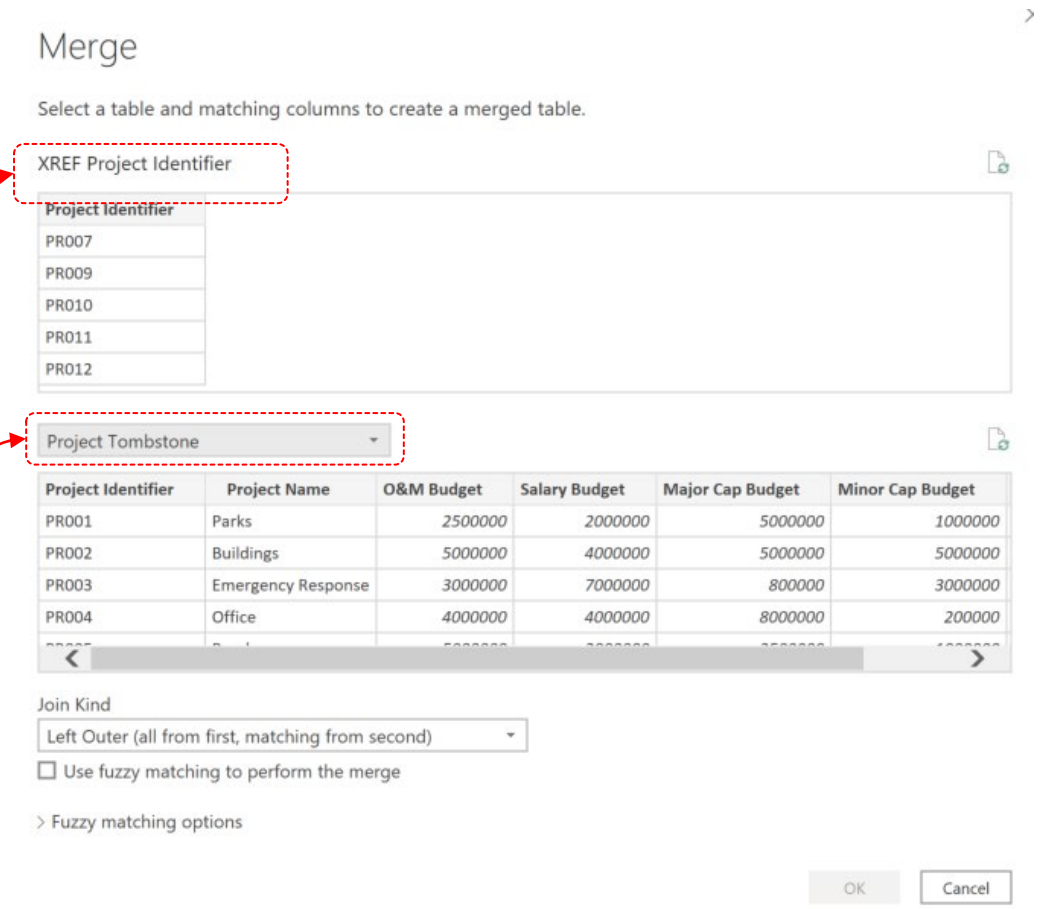
There are many reasons to do this but the example we are going to work through is that we have an existing table with one column of data we want to add to our “XREF Project Identifier” and a new table where we want to do the same.

We will first merge the column “Project Name” from the “Project Tombstone” table into the “XREF Project Identifier”

MERGING TABLES

Follow the instructor through the following steps:

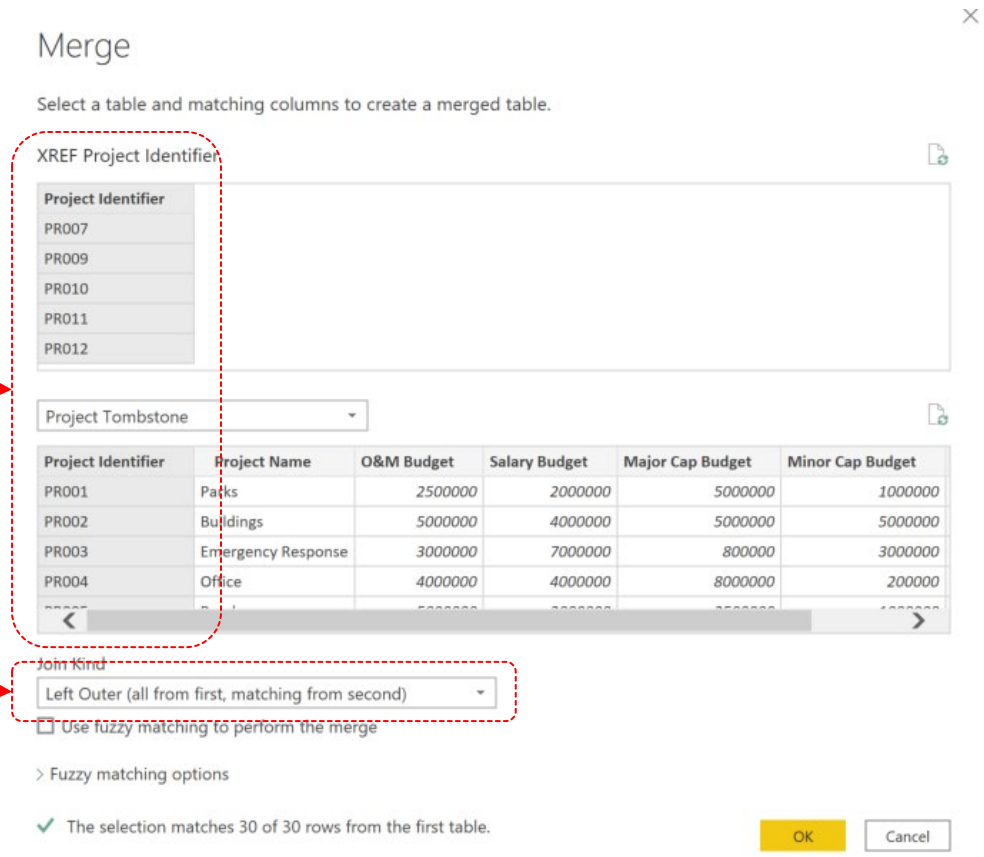
1. Click on “Transform Data”
2. Select the “XREF Project Identifier” query
3. Select “Merge Queries” (NOT “Merge as New”)
4. You will then see a dialog box, the top of which is already populated with the data from the “XREF Project Identifier” query
5. In the lower dropdown select the “Project Tombstone” query



MERGING TABLES

Follow the instructor through the following steps:

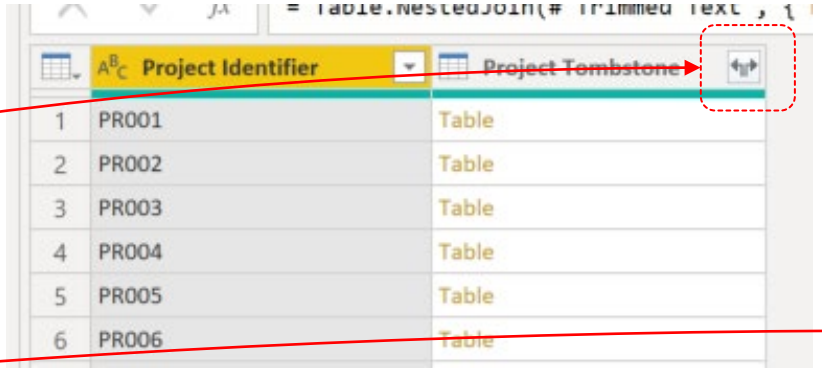
6. We next need to select which columns we are going to match. Click on the “Project Identifier” column in each of the tables
7. We then need to select the Join Kind. We are going to leave this for now as “Left Outer” but we will review all of the options once we have complete this operation
8. Click on “OK”

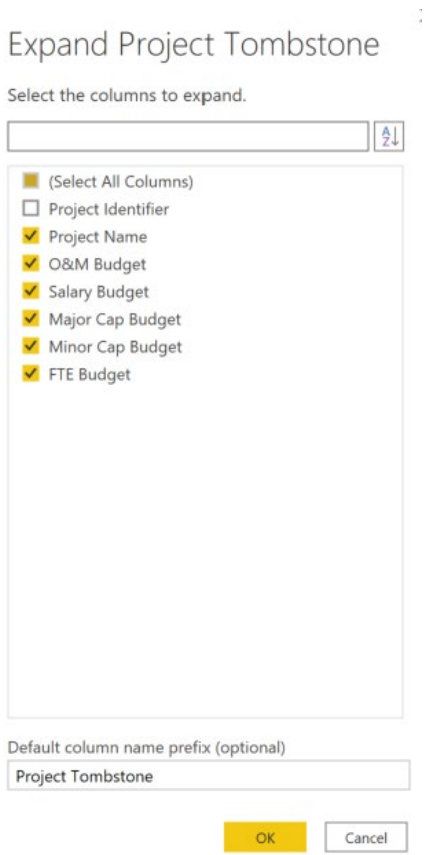


MERGING TABLES

Follow the instructor through the following steps:

7. We want to now expand the table, click on the double outward pointing arrow and select everything except "Project Identifier" (we already have that)
8. Hit "OK"
9. Rename "Project Tombstone" to "DNL Project Tombstone 1" and exclude from load (we no longer need it)





MERGING TABLES

EXERCISE

- Import the new Excel spreadsheet "List of Project Managers.xlsx" and merge it with "XREF Project Identifier" ONLY expanding the "Project Manager" name column
- You will notice that this file has more project numbers on there BUT we don't want the ones that begin with "EXT". The Join Kind we selected "Left Outer" took all the values from the first query "XREF Project Identifier" and only extracted the data with the ones that matched from the second query "List of Project Managers"
- Once you are done the merge, rename the "List of Project Managers" query "DNL List of Project Managers" and uncheck the "Enable Load" option as we did before.

	Project Identifier	Project Tombstone.Project Name	DNL Project Manager.Project Manager
1	PR001	Parks	S Davies
2	PR002	Buildings	P Boily
3	PR003	Emergency Response	J Schellinck
4	PR004	Office	P Davies
5	PR005	Roads	K Warren
6	PR006	Science	P Stewart
7	PR007	Heritage	E Greve
8	PR008	Celebration	P Boily
9	PR009	Research	J Schellinck
10	PR010	Upgrades	P Davies
11	PR011	Vehicles	K Warren
12	PR012	Recreation	J Schellinck
13	PR013	Space	P Davies
14	PR014	Learning and Development	K Warren
15	PR015	Technology	P Stewart
16	PR016	Retirement	E Greve
17	PR017	Innovation	P Boily
18	PR018	Computing	J Schellinck

MERGING TABLES

There are different options for the Join Kind, they are:

- Left Outer: all values from the first query and the values that match those from the second. This is the default behaviour
- Right Outer: all values from the second query, matching from the first
- Full Outer: all rows from both queries
- Inner: only the rows that match
- Left Anti: the rows only from the first
- Right Anti: the rows only from the second

PIVOTING / UNPIVOTING DATA

PIVOTING / UNPIVOTING DATA

Sometimes we get data that requires Pivoting or Unpivoting.

- Pivoting is transforming rows into columns
- Unpivoting is transforming columns into rows

Unpivot by course

Name	Course	Score
Steve	Physics	95
Steve	English	50
Emily	Physics	95
Emily	English	90

Name	Physics	English
Steve	95	50
Emily	95	90

Pivot by course

PIVOTING / UNPIVOTING DATA

Sometimes we get data that requires Pivoting or Unpivoting.

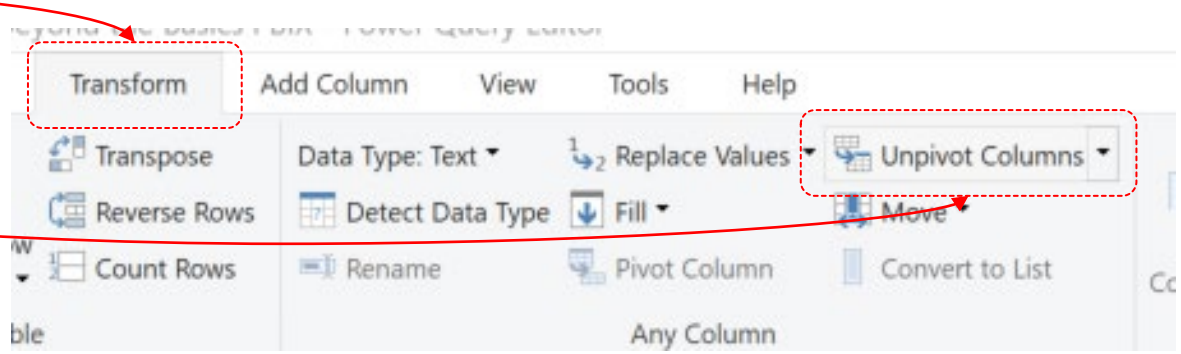
- In our data we have one table that we want to unpivot – “TBSub Project Milestones”
- Class question – why do we want to unpivot this table?
- Follow the instructor through the slide to see how to unpivot these columns

PIVOTING / UNPIVOTING DATA

Follow the instructor:

1. Select the table "TBSub Project Milestones"
2. Click on the "2018" column, hold down "shift" and then click on the "2020" column
3. Click on the "Transform" tab and then click on "Unpivot Columns"

	Project Identifier	Project Name	2018	2019	2020
1	PR001	Parks	Start	Finish	null
2	PR002	Buildings	null	Start	Finish
3	PR003	Emergency Response	Start	Finish	null
4	PR004	Office	Start	null	Finish
5	PR005	Roads	Start	Start	Finish
6	PR006	Science	null	Start	Finish
7	PR007	Heritage	null	Start	Finish
8	PR008	Celebration	Start	Finish	null
9	PR009	Research	Start	Finish	null
10	PR010	Upgrades	Start	null	Finish
11	PR011	Vehicles	Start	Finish	null
12	PR012	Recreation	Start	null	Finish



PIVOTING / UNPIVOTING DATA

Follow the instructor:

4. You will then see the unpivoted data
5. Rename the final two columns “Year” and “Milestone” respectively

	Project Identifier	Project Name	Year	Milestone
1	PR001	Parks	2018	Start
2	PR001	Parks	2019	Finish
3	PR002	Buildings	2019	Start
4	PR002	Buildings	2020	Finish
5	PR003	Emergency Response	2018	Start
6	PR003	Emergency Response	2019	Finish
7	PR004	Office	2018	Start
8	PR004	Office	2020	Finish
9	PR005	Roads	2018	Start
10	PR005	Roads	2019	Start
11	PR005	Roads	2020	Finish
12	PR006	Science	2019	Start
13	PR006	Science	2020	Finish
14	PR007	Heritage	2019	Start

PARAMETERS

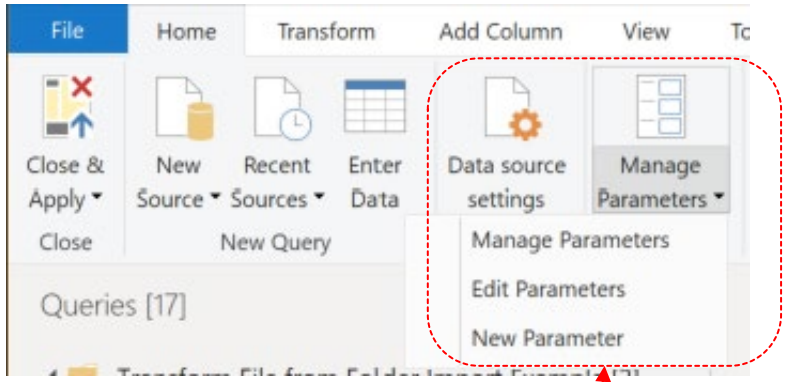
PARAMETERS

- Parameters are essentially global variables that, when changed, updates the raw data set
- Essentially a parameter serves to easily store and manage a value that can be reused
- Parameters can be changed in Power Query but more importantly can also be changed in Power BI
- Parameters can be driven from three inputs
  - Any value: The end user can type in anything that they want
  - List of values: A user defined list with a default value that the end user can pick from
  - Query: A list of inputs derived from an existing query (must be defined as a “list”, more on that later)
  - Finally, a “current value” must be entered manually to create the parameter
- Follow the instructor in the following slide to create our first parameter

PARAMETERS

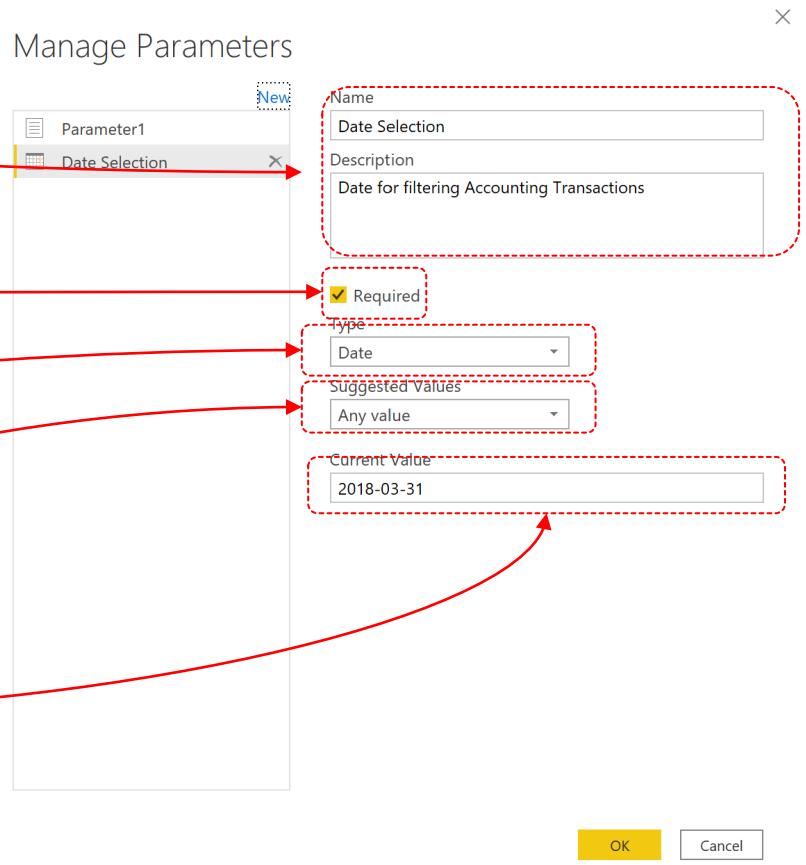
- We want to be able to filter the “Effective Date” column in the “Accounting Transactions” table
- Rather than manually entering a value in the filter we are going to use a parameter instead
- Here are the steps for creating the parameter:

1. In the “Home” tab, click on “Manage Parameters”, then “New Parameter”



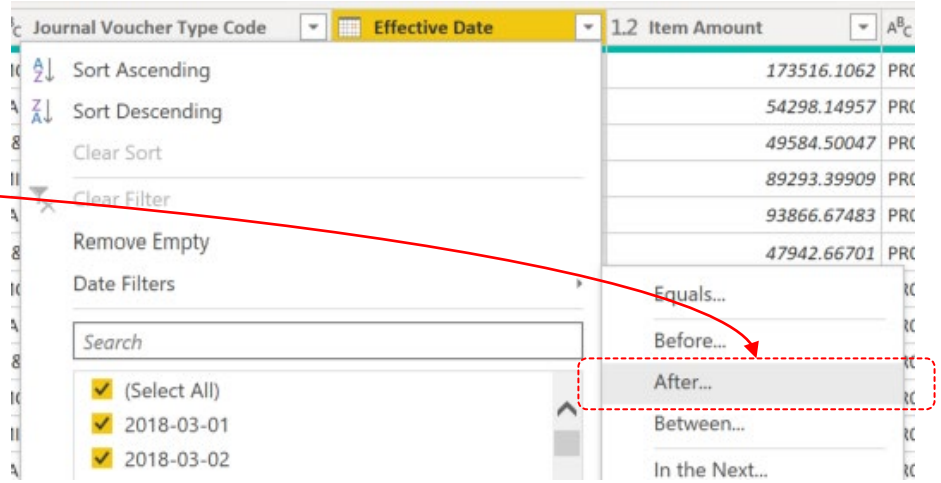
PARAMETERS

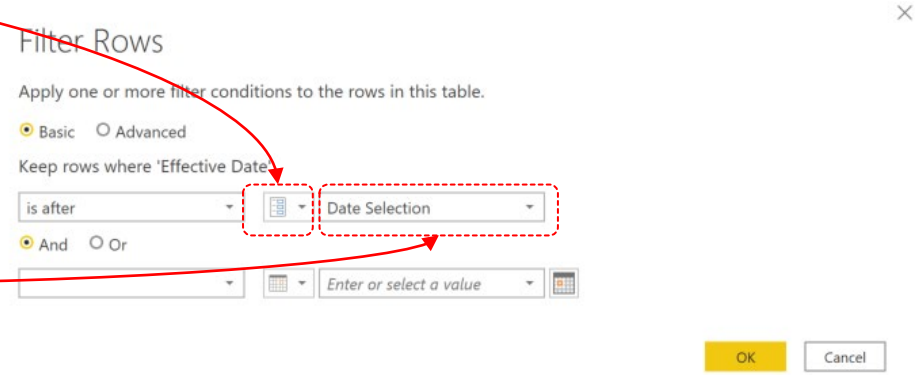
2. Rename the parameter “Date Selection” and add in a description
3. Make sure the “Required” check box is selected
4. Select “Type” as “Date” (it is really important to select a type for every parameter you create)
5. In “Suggested Values” select “Any value”
6. For the “Current Value” type in “2018-03-31”
7. Hit “OK” and move the parameter to a new group called “Parameters”



PARAMETERS

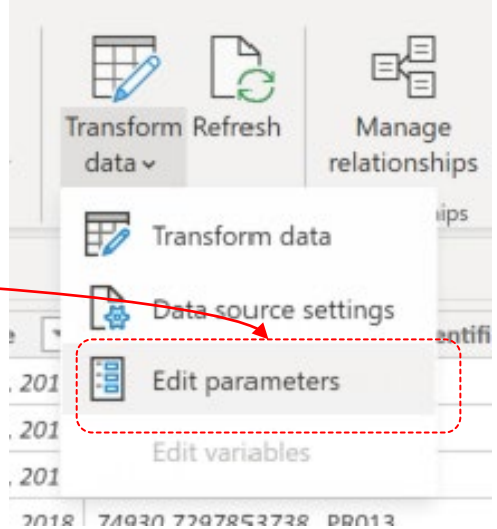
8. Select the “Accounting Transactions” table and the “Effective Date” column
9. Click on the filter arrow, choose “Date Filters” then “After”
10. In the new dialog box, click the middle dropdown arrow and select “Parameter”
11. As this is our first Parameter “Date Selection” will automatically be selected, with more you will be prompted to pick one
12. Hit “OK” and the table will be filtered.
13. Hit “Close and Apply”

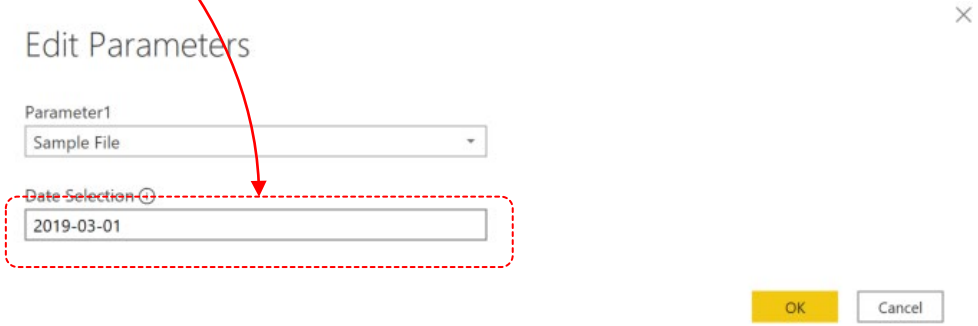




PARAMETERS

14. To see how the end user can use Parameters in Power BI select the “Accounting Transactions” Table
15. Click on the “Transform data” button in the “Home” menu and then select “Edit parameters”
16. Now change the “2018” to “2019” (or whatever you want)
17. Hit “OK”, then “Apply changes” when prompted
18. Power Query now updates and pushes the new filtered dataset into Power BI





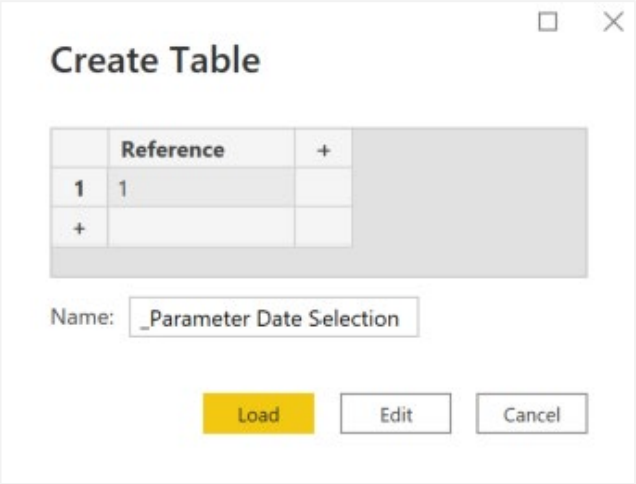
PARAMETERS

- Parameters are extremely useful and help to keep things in order
- In future course we will show you how to build from lists and from other queries
- If you are going to learn to do this on your own remember one thing, to use a query in a parameter it must contain a single column and be converted into a LIST (the instructor can show you how to do this if there is time)

PARAMETERS

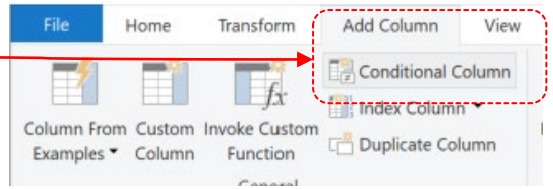
One more thing that is REALLY useful is to return the parameter value into a table that we can then use in Power BI (e.g. on a card as part of a dynamic heading)

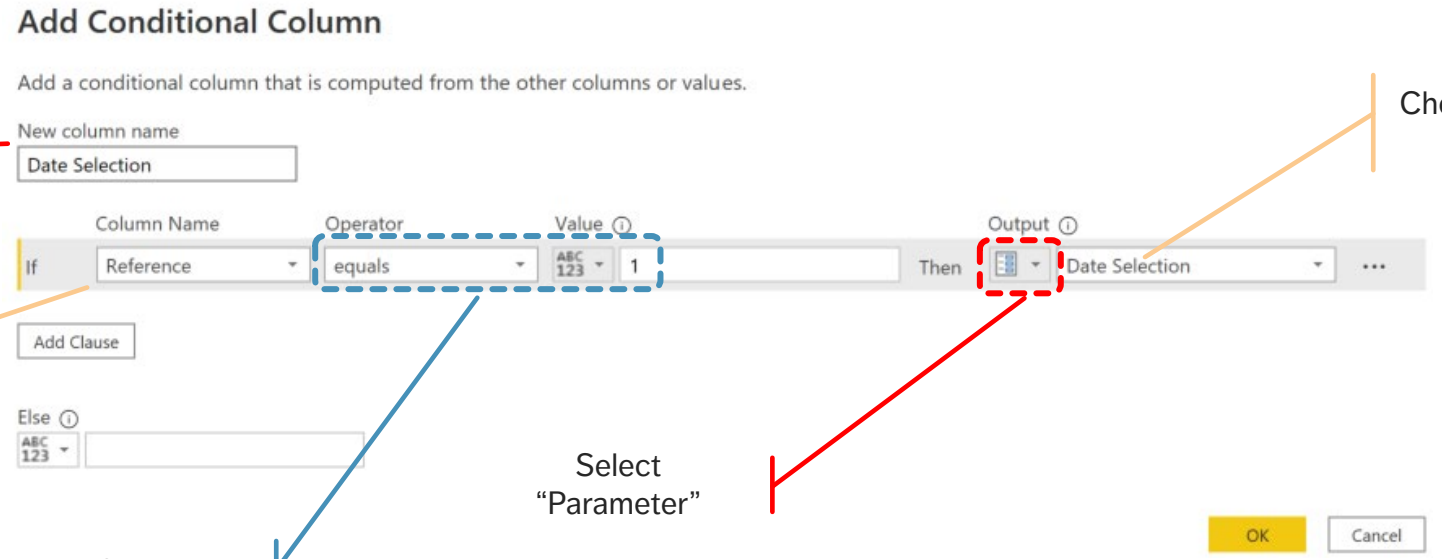
1. Click on “Enter data” button, create a new table with a single column called “Reference” and a single value of “1”. Name the table “_Parameter Date Selection” and hit “Load”
2. Click on “Transform Data” to open up Power Query
3. Select the “_Parameter Date Selection” table



PARAMETERS

4. Click on the “Add Column” menu and then “Conditional Column”
5. Once the dialog box comes up fill it in as per below then hit “OK”





Rename column

Choose the appropriate parameter

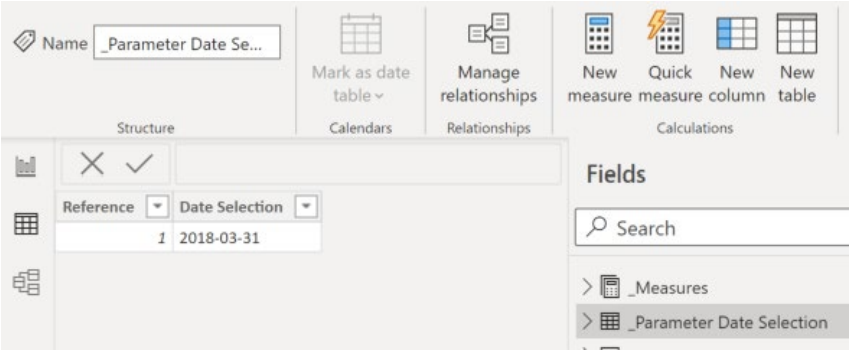
Use “Reference” column for test

Select “Parameter”

Test is where “Reference” = 1

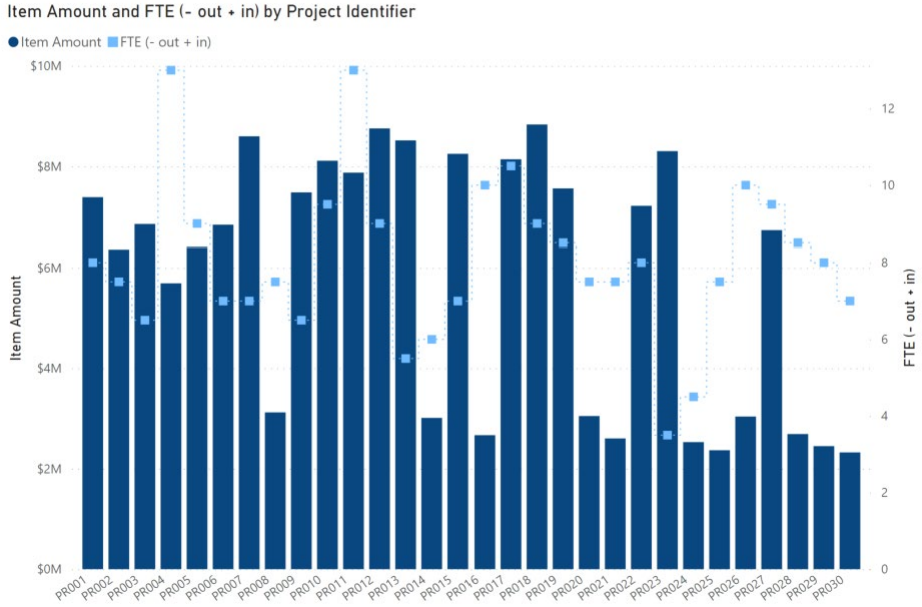
PARAMETERS

6. Now we have a new table with a single value that we can use wherever we need to!



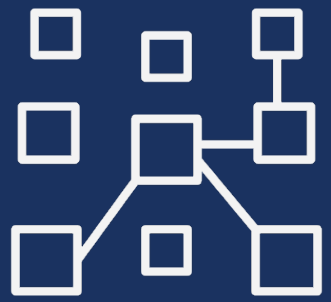
Data displayed is AFTER

31-Mar-2018



DATA MODELING

PBI-2: POWER BI – BEYOND THE BASICS



DATA MODELING

DATA MODELING

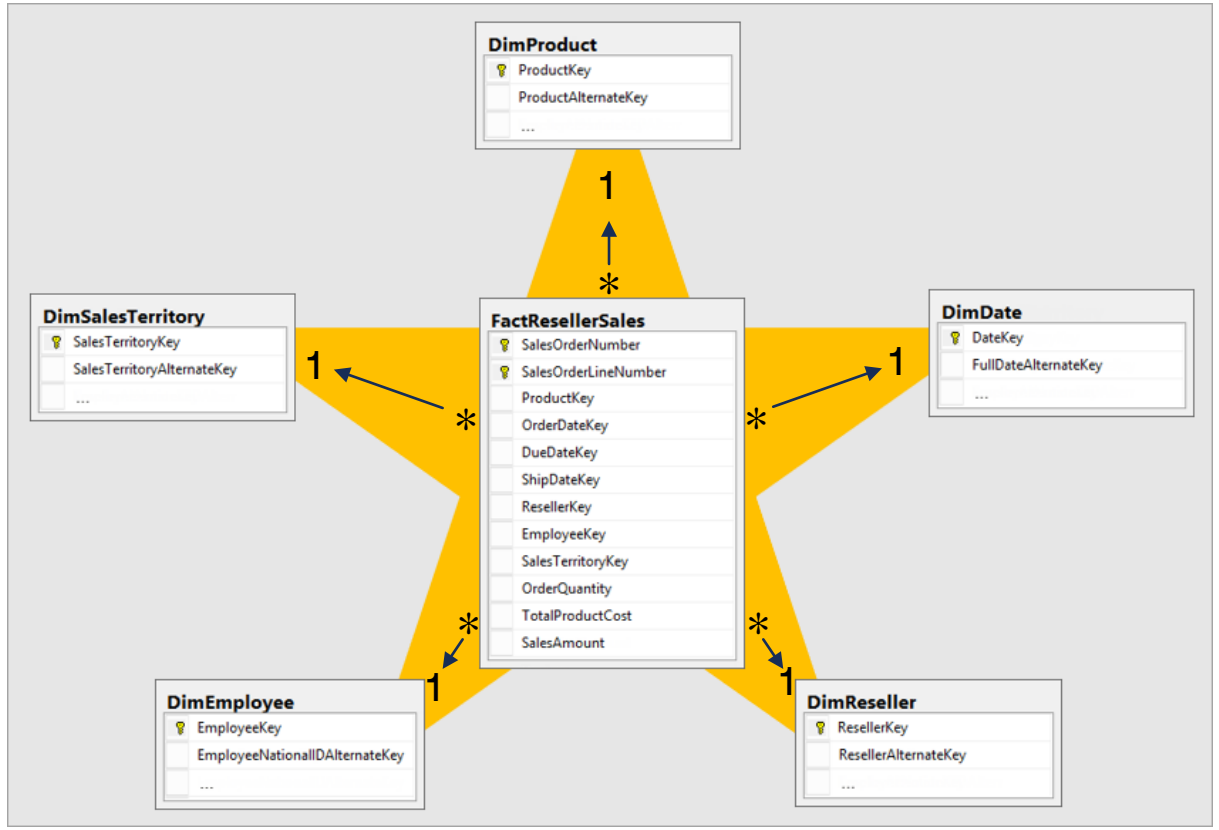
We are now back in Power BI, the next thing we want to do it to understand data modeling in a little more detail

Data models have (typically) two types of tables:

1. Fact tables: Tables that we can use to perform operations like “Sum”
2. Dimension tables: Tables that describe the data (we are calling these XREF)

In an ideal world we try and build these as “Star Schemas”

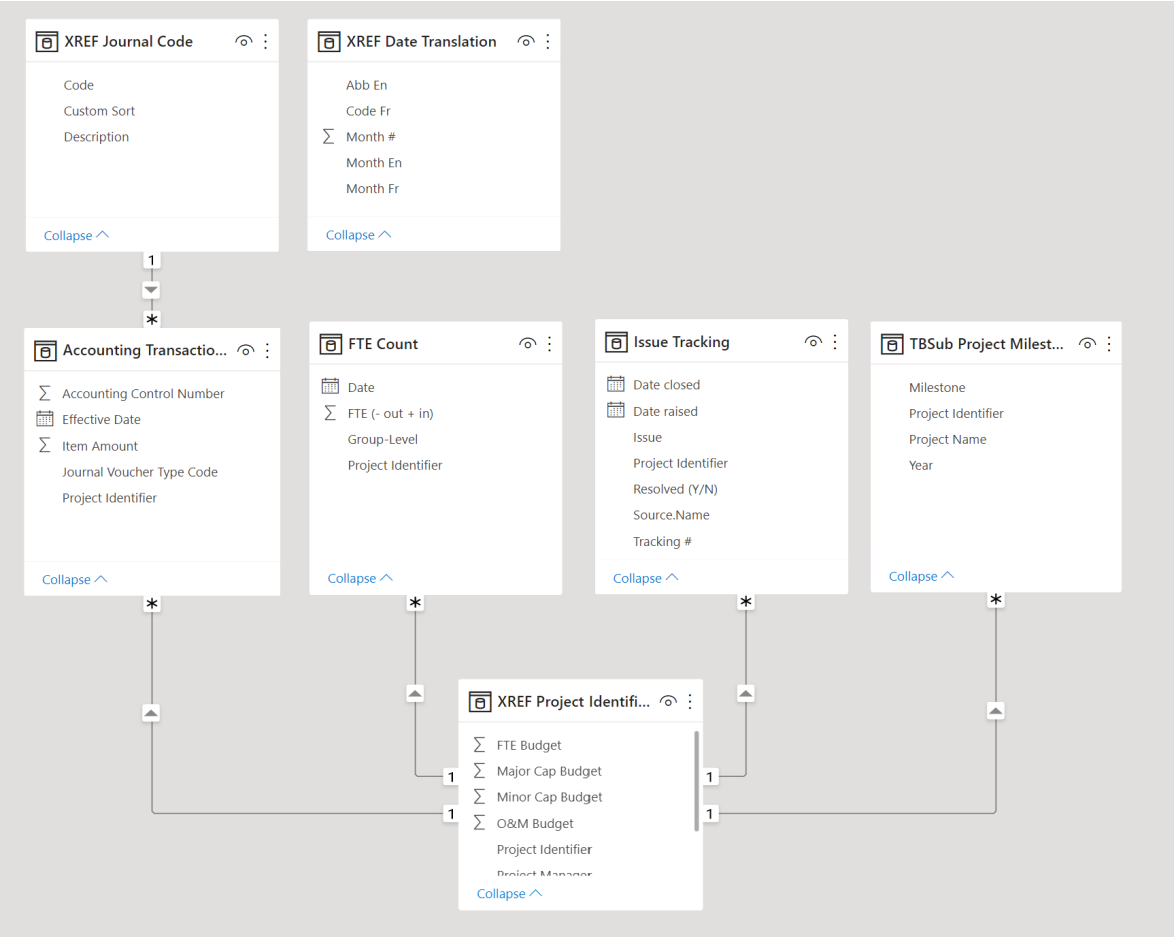
DATA MODELING



https://docs.microsoft.com/en-us/power-bi/guidance/star-schema

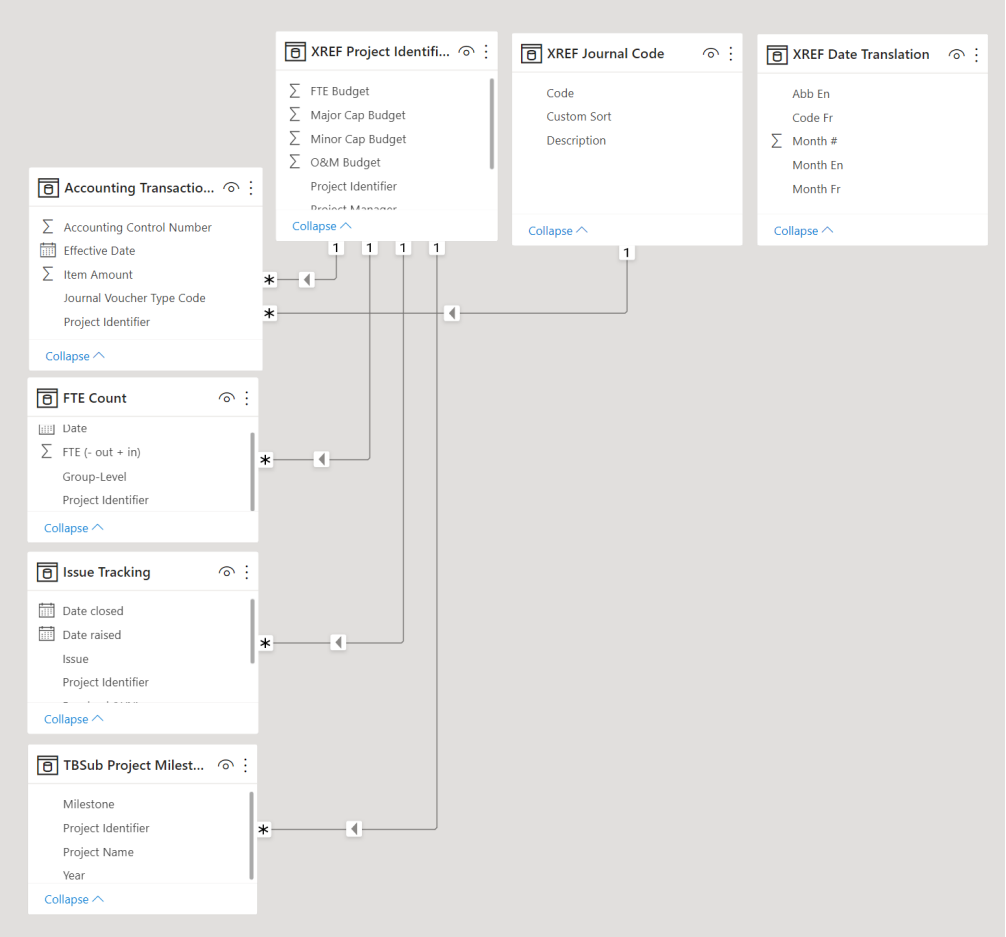
DATA MODELING

- Our data isn't quite as straightforward as this, in fact we have 4 fact tables and only 3 dimension tables
- We can still set it up as a Star Schema – in fact we have multiple fact tables that can share dimension tables
- For now we are leaving the "XREF Date Translation" table (we will connect that later)



DATA MODELING

- Sometimes laying tables out can get a bit messy
- One approach (if you don't have TOO many tables) is to put your dimension tables (XREF) across the top and your fact tables down the left hand side
- This can make it easier to keep track of which tables are connected when things get a bit more complicated (but it is personal preference)



CARDINALITY AND CROSS FILTER DIRECTION

CARDINALITY

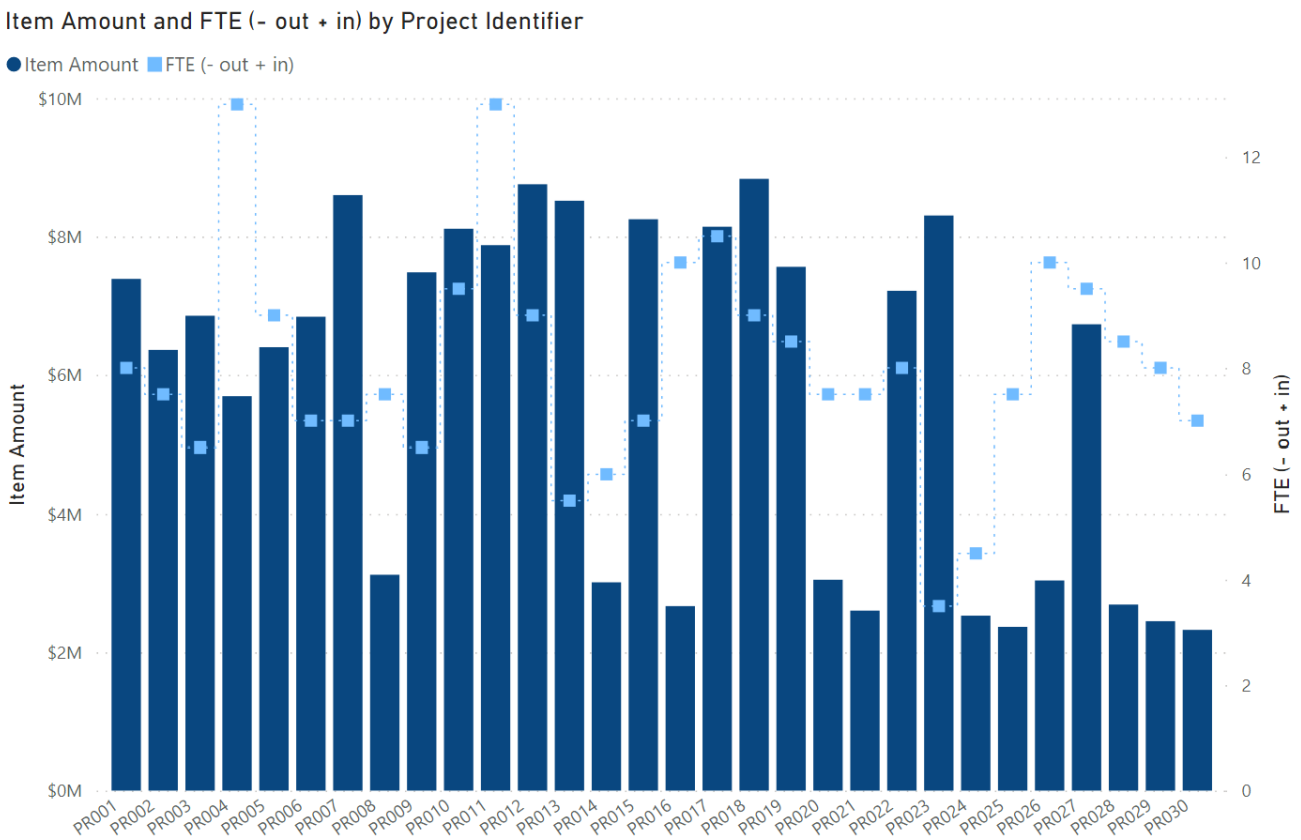
As you can see the Star Schema works if our dimension tables have a one-to-many relationship with our fact tables

We can also have one-to-one relationship BUT in general if there is only one of each data key (e.g. Project Identifier) in a table is it likely a dimension table

As per the “Getting to know Power BI” course we are still staying clear of many-to-many relationship. There are circumstances where we can make them work but for now if Power BI is identifying a relationship as many to many you are likely trying to link two fact tables directly together, rather than to link them through a shared dimension table

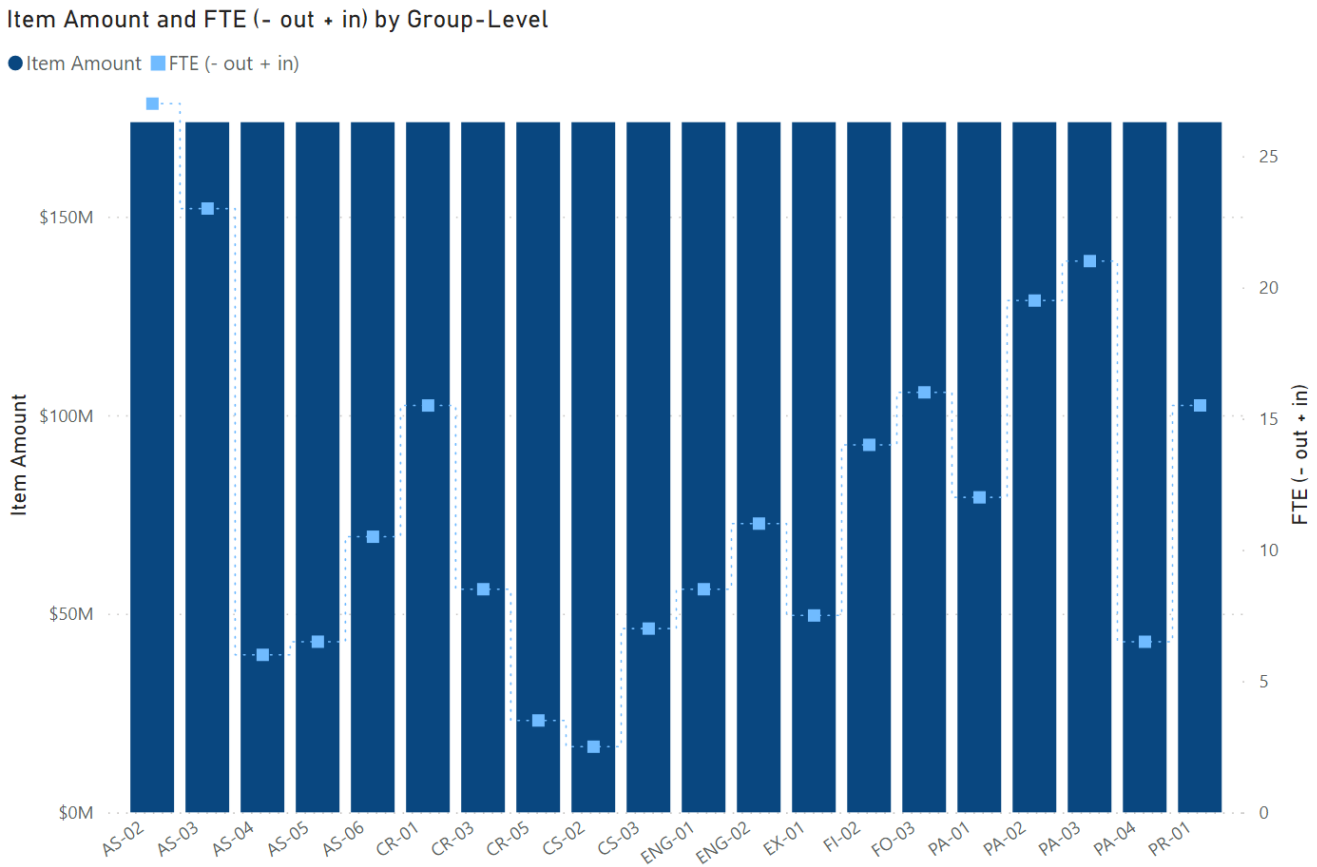
CROSS FILTER DIRECTION

- Going back to the “Getting to know Power BI” course we built a line and staked column chart that compared “Item Amount” and “FTE (-out +in)” for each Project Identifier
- In this case we had a common axis, let’s try doing the same with a column from one of our fact tables and see what happens



CROSS FILTER DIRECTION

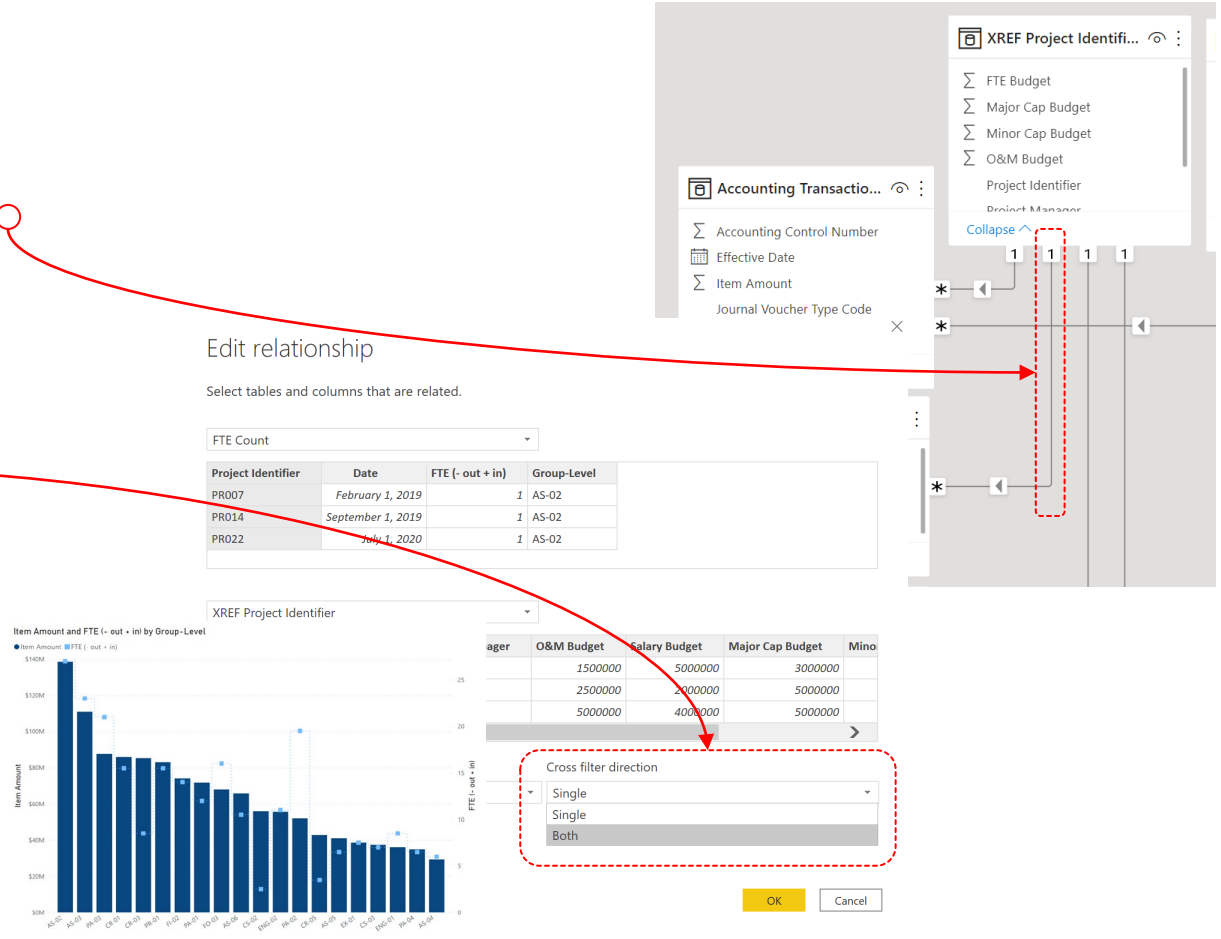
- Let's build a chart where instead of "Project Identifier" on the axis we have "Group-Level"
- We still want the same values – "Item Amount" for the columns and "FTE (-out +in) for the line
- Odd – for some reason we have a single value (the total amount) for all of our columns.
- Can anybody guess why this is happening?



CROSS FILTER DIRECTION

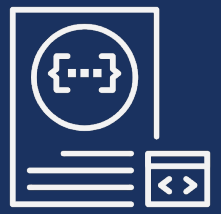
To fix this let's do the following

1. Go to the data model and double click on the connection between the "FTE Count" table and "XREF Project Identifier"
2. In the dialog change the "Cross Filter Direction" from "Single" to "Both"
3. This now allows the chart to use the axis values to filter the data in table "Accounting Transactions"



BEST PRACTICE ALERTS!

PBI-2: POWER BI – BEYOND THE BASICS



DECLARING VARIABLES
BOOLEAN LOGIC
NESTED IF

DECLARING VARIABLES

In a Power BI formula we can declare variables, that is to assign a value or function to a label that we can reuse. For example lets build a measure where we assign 2 variables, “BeginningDate” to be the first date and “EndDate” to be the last date in the “Effective Date” column in the “Accounting Transactions” table.

```
Effective Date Difference =

    VAR BeginningDate = min('Accounting Transactions'[Effective Date])
    VAR EndDate = max('Accounting Transactions'[Effective Date])
    RETURN

    DATEDIFF(BeginningDate,
             EndDate,
             DAY)
```

BOOLEAN LOGIC

Another issue with Power BI is when we use AND and OR functions we are restricted to using 2 arguments (that is a little lie, we can nest them to get more but that is a pain!). For example:

```
Total Capital = CALCULATE(sum('Accounting Transactions'[Item Amount]),
                          OR('Accounting Transactions'[Journal Voucher Type Code]="MIC",
                             'Accounting Transactions'[Journal Voucher Type Code]="MC"))
```

Instead of the OR we can use "||" (and if we were using AND we would use "&&")

```
Total Capital Alt = CALCULATE(sum('Accounting Transactions'[Item Amount]),
                              'Accounting Transactions'[Journal Voucher Type Code]="MIC" ||
                              'Accounting Transactions'[Journal Voucher Type Code]="MC")
```

https://docs.microsoft.com/en-us/dax/dax-operator-reference

DEALING WITH NESTED IF

As many of you may have experienced when using Excel, when using a number of nested IF statements things can get pretty messy. For example in our “XREF Journal Code” table we could add in descriptions using a set of nested “IF” statements:

```
Description using IF = if('XREF Journal Code'[Code]="O&M", "Operations and Maintenance",
                            if('XREF Journal Code'[Code]="MC", "Major Capital",
                                if('XREF Journal Code'[Code]="MIC", "Minor Capital",
                                    "Salary")))
```

In the above example it isn’t too bad but things can get messy pretty quickly!

DEALING WITH NESTED IF

Alternatively we can use the DAX “SWITCH” function, which just tends to be simpler to use and cleaner to layout:

```
Description using SWITCH = SWITCH(TRUE,
                                  'XREF Journal Code'[Code]="O&M", "Operations and Maintenance",
                                  'XREF Journal Code'[Code]="MC", "Major Capital",
                                  'XREF Journal Code'[Code]="MIC", "Minor Capital",
                                  'XREF Journal Code'[Code]="SA", "Salary")
```

DOING MORE WITH DAX

PBI-2: POWER BI – BEYOND THE BASICS

CALCULATED TABLES

CALCULATED TABLES

In the “Getting to Know Power BI” course we worked with both Measures and Calculated Columns. We also reviewed Calculated Tables but did not work through any examples.

To refresh our memory, some DAX functions return a matrix result (i.e., multiple not single value), in these cases we need to save those results in a table (a single cell is not enough space!!). We also may want to create XREF tables automatically in DAX rather than Power Query (for convenience or another reason).

The first Calculated Table we are going to create is a list of dates that starts with the earliest date from ALL of our date columns (across all tables) and ends with the latest date from ALL of our date columns (across all tables).

We will also see how important using VAR becomes!! Ok so before the next slide click on “New Table” then either type on paste in the following code:

CALCULATED TABLES

```
XREF Date =
        VAR TempEarly =
                {
                MIN('Accounting Transactions'[Effective Date]),
                MIN('FTE Count'[Date]),
                MIN('Issue Tracking'[Date closed]),
                MIN('Issue Tracking'[Date raised])
                }

        VAR EarliestDate = MINX(TempEarly, [Value])

        VAR TempLate =
                {
                MAX('Accounting Transactions'[Effective Date]),
                MAX('FTE Count'[Date]),
                MAX('Issue Tracking'[Date closed]),
                MAX('Issue Tracking'[Date raised])
                }
        VAR LatestDate = MAXX(TempLate, [Value])
        RETURN
CALENDAR(EarliestDate, LatestDate)
```

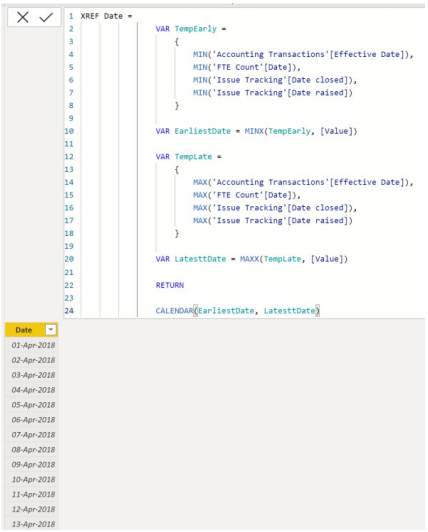
This first Variable creates a temporary table (using the { }) and each row is defined as the MINIMUM date from each of our target columns

This second Variable uses MINX (a function that looks at each row in the above table) and returns the Minimum of all of the Minimum values!

This third Variable creates a temporary table (using the { }) and each row is defined as the MAXIMUM date from each of our target columns

This fourth Variable uses MAXX (a function that looks at each row in the above table) and returns the MAXIMUM of all of the MAXIMUM values!

Finally the “Calendar” function returns a table of dates from the Earliest to the Latest date we defined above



CALCULATED TABLES

Another use for calculated tables is to summarize other tables. E.g. we can create a table of "TBSub Project Milestones" by "Year" and "Milestone". We might do this if our data tables are large and we only want to create charts from the summarized data.

SUMMARIZECOLUMNS(<groupBy_columnName> [, < groupBy_columnName >]..., [<filterTable>]...[, <name>, <expression>]...)

1. Click on "New Table"
2. Use the following DAX to add in the summary data:

```
TBSub Milestone (SUMMARIZECOLUMNS) = SUMMARIZECOLUMNS('TBSub Project Milestones'[Year],
                                     'TBSub Project Milestones'[Milestone],
                                     "Count of Milestones",
                                      count('TBSub Project Milestones'[Project Identifier]))
```

RANKING

RANKING

Another useful function in Power BI is to be able to rank items. We can do this when we sort tables and chats (lowest to highest for example) but sometimes we might want to rank some items so we can do further analysis. In Power BI we use the RANKX function to do this.

In our next example we are going to do the following:

- Create a new summary table of our FTE data summing the #FTE's per project
- We are then going to add in two new columns to rank them in order from most to fewest FTE using slightly different parameters

```
RANKX(<table>, <expression>[, <value>[, <order>[, <ties>]]])
```

RANKING

Follow the instructor through the following steps:

1. Create a new table and type or paste in the following code:

```
FTE Summary = SUMMARIZECOLUMNS('FTE Count'[Project Identifier],
                                "Total FTE",
                                sum('FTE Count'[FTE (- out + in)]))
```

2. This will give us a summary table with two columns – “Project Identifier” and a second column “Total FTE” which gives us the total FTE for each project

RANKING

3. Our first ranking column is created as follows:

```
Project FTE Rank (Skip) = RANKX('FTE Summary’,
                                'FTE Summary'[Total FTE],
                                ,
                                ASC,
                                Skip)
```

4. You will notice that in the ranking numbers are ”Skipped” when there is a tie:

Total FTE	Project Identifier	Project FTE Rank (Skip)
3.5	PR023	1
4.5	PR024	2
5.5	PR013	3
6	PR014	4
6.5	PR009	5
6.5	PR003	5
7	PR007	7
7	PR006	7
7	PR015	7
7	PR030	7
7.5	PR008	11
7.5	PR002	11
7.5	PR021	11
7.5	PR020	11
7.5	PR025	11
8	PR022	16
8	PR001	16
8	PR029	16

RANKING

3. Our second ranking column is created as follows:

```
Project FTE Rank (Dense) = RANKX('FTE Summary’,
                                 'FTE Summary'[Total FTE],
                                 ,
                                 ASC,
                                 Dense)
```

4. You will notice that in the ranking numbers are continuous when there is a tie:

Total FTE	Project Identifier	Project FTE Rank (Skip)	Project FTE Rank (Dense)
3.5	PR023	1	1
4.5	PR024	2	2
5.5	PR013	3	3
6	PR014	4	4
6.5	PR009	5	5
6.5	PR003	5	5
7	PR007	7	6
7	PR006	7	6
7	PR015	7	6
7	PR030	7	6
7.5	PR008	11	7
7.5	PR002	11	7
7.5	PR021	11	7
7.5	PR020	11	7
7.5	PR025	11	7
8	PR022	16	8
8	PR001	16	8
8	PR029	16	8

FILTERING

FILTERING

Filtering data is a huge topic in Power BI, we are going to scratch the surface by looking at the following DAX functions:

- REMOVEFILTERS[1]: REMOVEFILTERS ([<table> | <column>[, <column>[, <column>[,...]]]]) - Returns all the rows in a table or all the values in a column, ignoring any filters which might have been applied.
- ALLEXCEPT: ALLEXCEPT(<table>,<column>[,<column>[,...]]) – Returns all the rows in a table or all the values in a column, ignoring all context filters applied but taking into account the specified columns filter
- FILTER: FILTER(<table>,<filter>) – Returns a filtered *table*

[1]REMOVEFILTERS replaced ALL but you can pretty much still use either one at this level.

FILTERING

The scenario is that we want to calculate the % of the “Item Amount” for each project compared to the total “Item Amount”

To do this we need to divide the sum of the item amount for each project by the sum of all the item amounts

Let’s create a measure and put it in a table and see what happens

```
% of total No ALL = divide(
                        sum('Accounting Transactions'[Item Amount]),
                        sum('Accounting Transactions'[Item Amount]),
                        0)
```

Project Identifier	% of total No ALL
PR001	100%
PR002	100%
PR003	100%
PR004	100%
PR005	100%
PR006	100%
PR007	100%
PR008	100%
PR009	100%
PR010	100%
PR011	100%

Hmmm – not really what we wanted (obviously!)

FILTERING

Obviously, the table is filtering both the numerator and denominator the same, which is why we are getting 100%.

What we need to do here is to stop the filtering on the denominator so we can actually work out the true %. To do this we can use the REMOVEFILTERS function

```
% of total ALL = DIVIDE(sum('Accounting Transactions'[Item Amount]),
                CALCULATE(sum('Accounting Transactions'[Item Amount]),
                REMOVEFILTERS('Accounting Transactions')),
                0)
```

Project Identifier	% of total No ALL	% of total ALL
PR001	100%	4%
PR002	100%	4%
PR003	100%	4%
PR004	100%	3%
PR005	100%	4%
PR006	100%	4%
PR007	100%	5%
PR008	100%	2%
PR009	100%	4%
PR010	100%	5%
PR011	100%	5%

But hang on, what happens if we add a slicer (add in Year from “Effective Date”

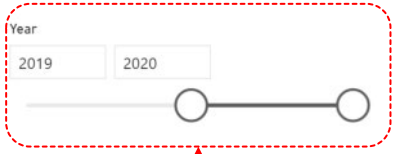
FILTERING

What happens when we move the slicer? The ROW context (Project Identifier) is still applied but the data set is sill reduced by not including 2019

This might be ok for what we want but we may be in a scenario where we don't want the slicer filter to be applied as well

In this case we can use ALLEXCEPT which returns all the rows in a table or all the values in a column, ***ignoring all context filters (i.e. the slicer)*** applied but taking into account the specified columns filter (i.e. the rows)

Project Identifier	% of total No REMOVEFILTERS	% of total REMOVEFILTERS
PR001	100%	3%
PR002	100%	3%
PR003	100%	3%
PR004	100%	2%
PR005	100%	3%
PR006	100%	3%
PR007	100%	4%
PR008	100%	1%
PR009	100%	3%
PR010	100%	3%
PR011	100%	3%
PR012	100%	4%
PR013	100%	4%
PR014	100%	1%
PR015	100%	3%
PR016	100%	1%
PR017	100%	3%
PR018	100%	3%
PR019	100%	3%
PR020	100%	1%
PR021	100%	1%
PR022	100%	3%
PR023	100%	3%
PR024	100%	1%
PR025	100%	1%
PR026	100%	1%
PR027	100%	3%
PR028	100%	1%
PR029	100%	1%
PR030	100%	1%
Total	**100%**	**69%**

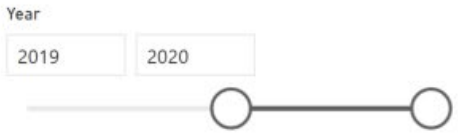


FILTERING

So, let's create a new measure using the following DAX and see what happens:

```
% of total ALLEXCEPT = DIVIDE(sum('Accounting Transactions'[Item Amount]),
                CALCULATE(sum('Accounting Transactions'[Item Amount]),
                ALLEXCEPT('Accounting Transactions','Accounting Transactions'[Effective Date].[Year])),
                0)
```

Project Identifier	% of total No REMOVEFILTERS	% of total REMOVEFILTERS	% of total ALLEXCEPT
PR001	100%	3%	4%
PR002	100%	3%	4%
PR003	100%	3%	4%
PR004	100%	2%	3%
PR005	100%	3%	4%
PR006	100%	3%	4%
PR007	100%	4%	5%
PR008	100%	1%	2%



FILTERING

Finally (and remember this is scratching the surface) we might want to make a new table that is a filtered subset of an existing table. We did cover one approach off in the calculated tables section, but this is another (potentially easier) approach if we want ALL of the columns from the original table.

For example, we might want to summarise the “Issue Tracking” table to only show where the issue is “No project manager”. To do this click on “New Table” and use the following DAX:

```
Issue Tracking (FILTER) = FILTER('Issue Tracking’,
                                 'Issue Tracking'[Issue]="No project manager")
```

Source.Name	Project Identifier	Issue	Date raised	Date closed	Resolved (Y/N)	Tracking #
Jane Doh tracking sheet.xlsx	PR011	No project manager	2018-11-01 12:00:00 AM	2018-12-08 12:00:00 AM	Y	TR09686
Jane Doh tracking sheet.xlsx	PR003	No project manager	2020-03-01 12:00:00 AM		N	TR91921
John Doe tracking sheet.xlsx	PR021	No project manager	2020-07-01 12:00:00 AM	2020-07-22 12:00:00 AM	Y	TR13868
John Doe tracking sheet.xlsx	PR025	No project manager	2019-01-01 12:00:00 AM	2019-01-07 12:00:00 AM	Y	TR66556
Stephen Davies tracking sheet.xlsx	PR004	No project manager	2019-09-01 12:00:00 AM	2019-10-07 12:00:00 AM	Y	TR44420
Stephen Davies tracking sheet.xlsx	PR021	No project manager	2020-07-01 12:00:00 AM	2020-07-22 12:00:00 AM	Y	TR13485
Stephen Davies tracking sheet.xlsx	PR024	No project manager	2019-03-23 12:00:00 AM		N	TR91527

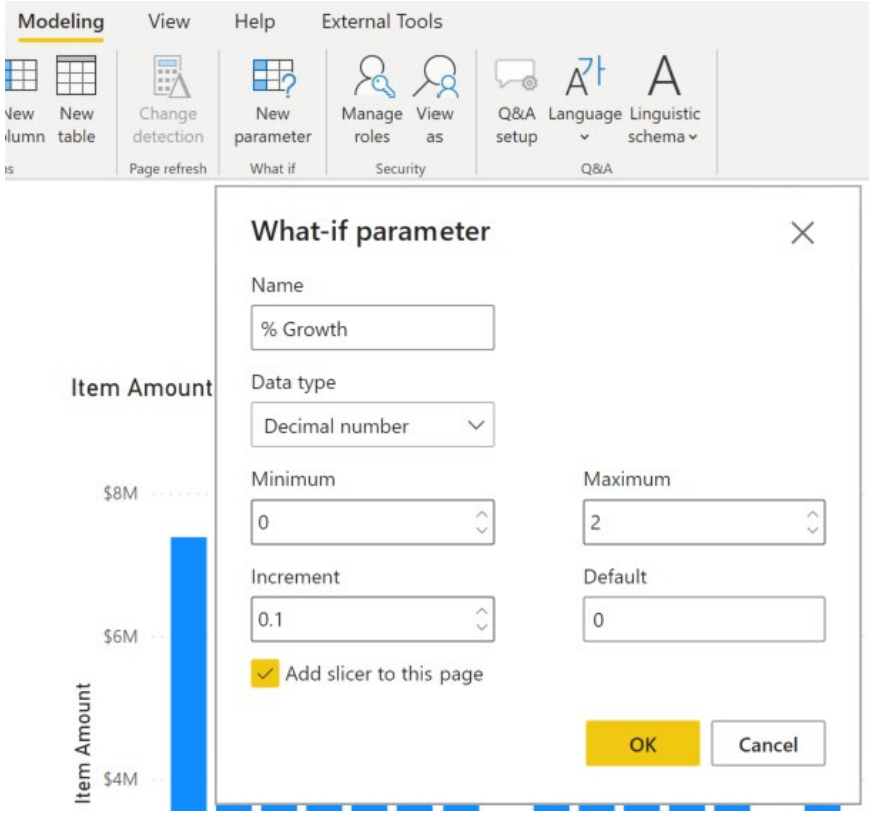
WHAT-IF PARAMETERS

WHAT IF PARAMETERS

There are situations where we might want to explore different options, or perform “What if” scenarios. Power BI has us covered – under the “Modeling” tab click on the “New parameter” in the “What if” section of the menu.

Make sure you fill in the information correctly – Name (% Growth), Data type (Decimal number), Minimum (0), Maximum (2), Increment (0.1), Default (0), Add slicer to this page (checked)

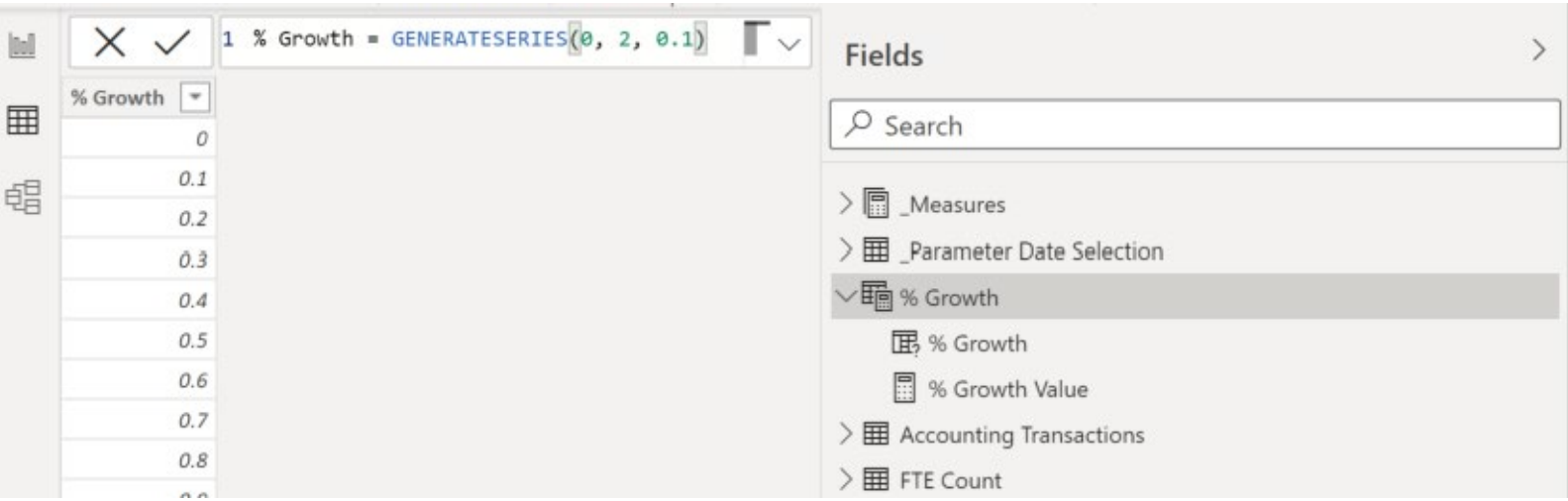
NOTE this isn’t really the same as the previous parameters (it’s a bit misleading), we are simply creating a user defined variable that we can use in DAX calculations (it uses the SELECTEDVALUE and GENERATESERIES DAX functions).



WHAT IF PARAMETERS

The GENERATESERIES function creates a table based on the following syntax:

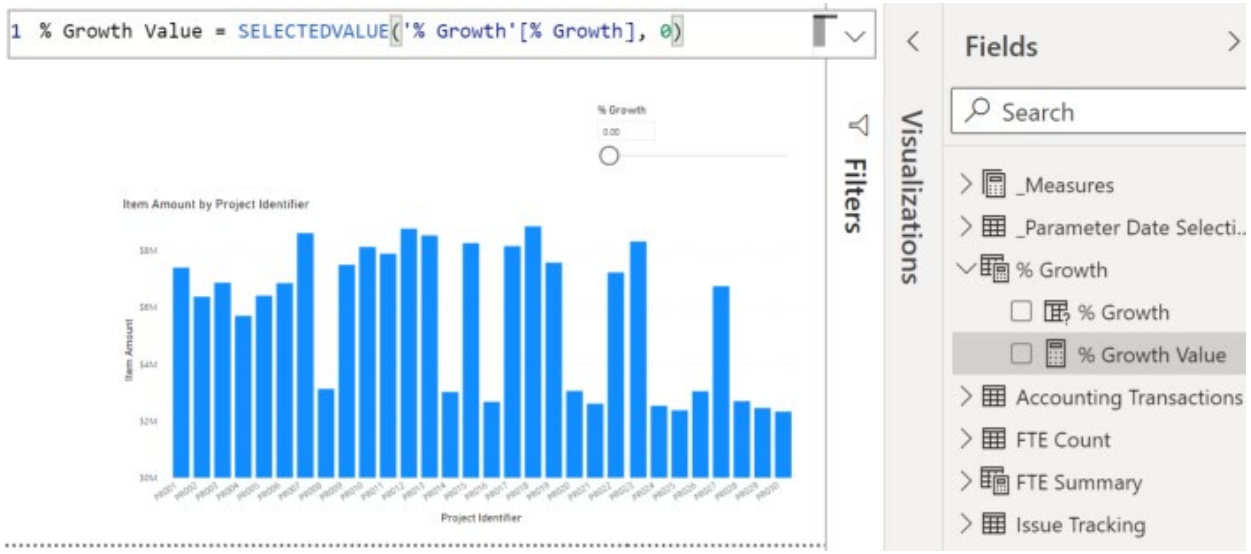
```
GENERATESERIES(<startValue>, <endValue>[, <incrementValue>])
```



WHAT IF PARAMETERS

There is also a measure that is created that is uses SELECTEDVALUE to pull a value from a slicer automatically put on the page.

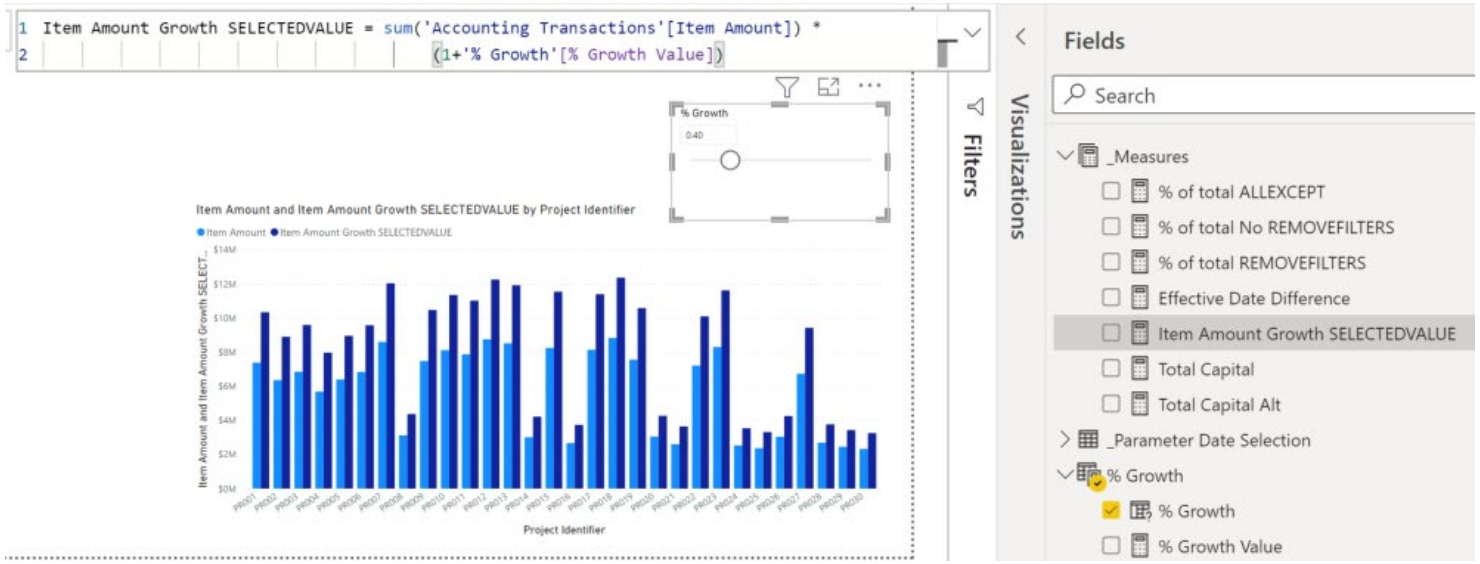
```
SELECTEDVALUE(<columnName>[, <alternateResult>])
```



WHAT IF PARAMETERS

We can now include this value in a new measure.

```
Item Amount Growth SELECTEDVALUE = sum('Accounting Transactions'[Item Amount]) *
                                   (1+'% Growth'[% Growth Value])
```



GOVERNMENT OF CANADA FISCAL YEAR CONVERSION

GOC FISCAL YEAR

One of the issues working for the GoC is that we use a non-calendar year described in various formats (e.g. 2021-22, FY2021-2022 etc.).

The problem with this is that to cross reference to a normal date we typically would build a manual cross reference table which is manual and not very flexible.

In the next few slides we are going to use the tricks that we have learned, plus some new DAX code to automatically add a new column to a date table which shows the GoC Fiscal date.

Full details (for a different data set) are posted on the Data Action Lab blog here:

https://www.data-action-lab.com/2018/10/22/goc-power-bi-tips-1-automatically-converting-a-date-to-goc-fiscal-year/

GOC FISCAL YEAR

We are going to add this new column to our “XREF date” table. The first thing to do is to calculate the actual fiscal year (not the fiscal month or period). To do this we need to do some initial setting up.

1. Select the “XREF Date” table
2. Let’s add in a separate column for “Year” by using the following DAX:

```
Year = YEAR('XREF Date'[Date])
```

4. Ok so let’s do the same for the month number:

```
Month = MONTH('XREF Date'[Date])
```

GOC FISCAL YEAR

5. Now we will use the following (fairly complex) DAX to calculate the fiscal year

```
Fiscal Year =
            if('XREF Date'[Date]>3,'XREF Date'[Year]
            /*(1) For a given year if [Month] < 3 then it is in previous FY. E.g. Jan 2018 is in FY 2017-18 whereas Apr
            2018 is in FY 2018-19*/
            &"-"& /*(2) Insert separator*/
            format(if(right('XREF Date'[Year],2)="99", "00",
            /*(3) this deals with change in century (e.g. 1999-2000 rollover). The format command ensures two digits if
                                                                                                  result<
            10 (e.g. 03)*/
            (right('XREF Date'[Year],2)+1)),"00"), /*(4) this deals with the condition where [Month] <=3 - see (1)*/
            (('XREF Date'[Year])-1) /*(5) this returns the "Year-1" is the month is Jan - Mar*/
            &"-"& /*(6) Insert separator*/
            format(right('XREF Date'[Year],2),"00")) /*(7) Inserts final 2 digits*/
```

GOC FISCAL YEAR

6. Phew! Ok that is the hard bit done, but we still need to add in the fiscal quarter and month (period)
7. We can now add in a new column for the quarter into the table by creating a new column with the following DAX:

```
Quarter = QUARTER('XREF Date'[Date])
```

8. Now we need to work out what the equivalent Fiscal Quarter is by using the following DAX. EXERCISE – work out and share with the class how the logic works!

```
Fiscal Quarter # = if('XREF Date'[Quarter]=1,
                      'XREF Date'[Quarter]+3,
                      'XREF Date'[Quarter]-1)
```

GOC FISCAL YEAR

9. Almost there!!!
10. In a very similar way, we will now use our “Month” column to calculate the fiscal period

```
Fiscal Period = if(and('XREF Date'[Month]<=12,'XREF Date'[Month]>3),
                  'XREF Date'[Month]-3,
                  'XREF Date'[Month]+9)
```

11. EXERCISE – work out and share with the class how the logic works!
12. Finally, you can add in a day column to use as you want to

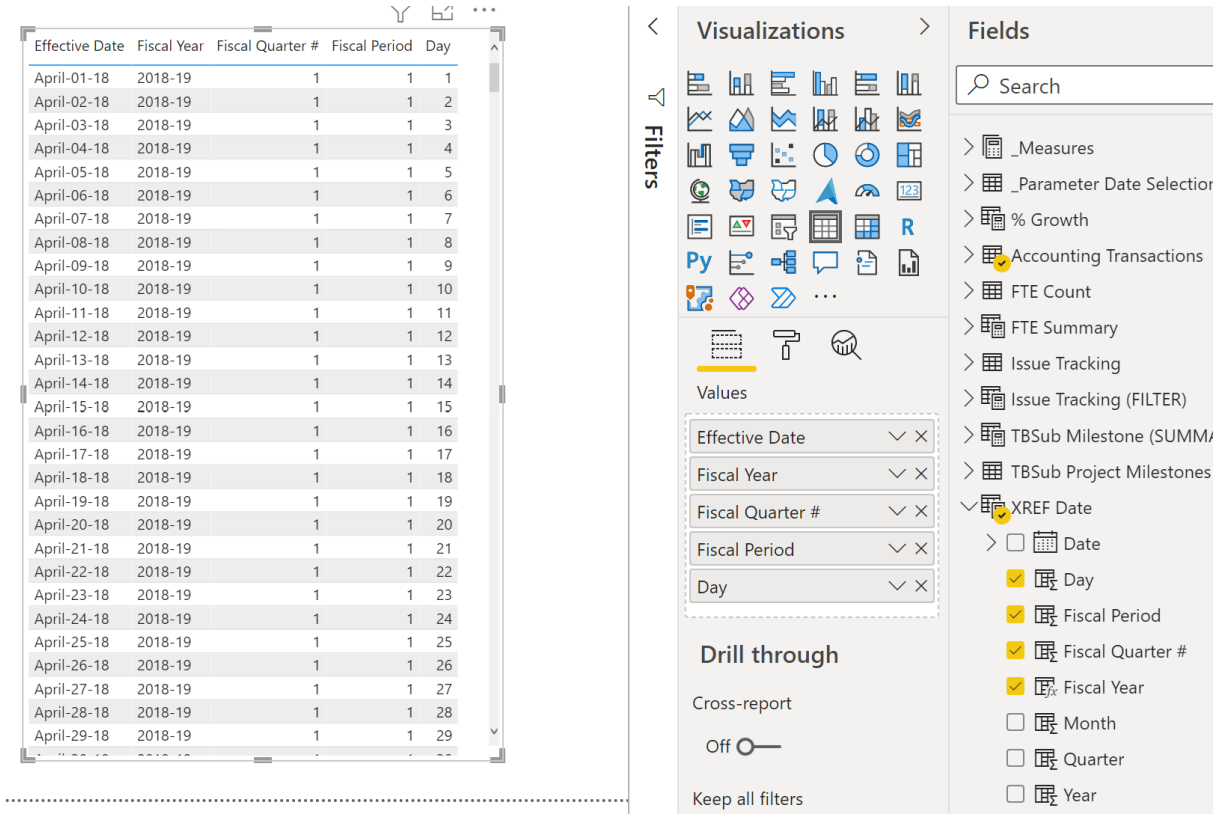
```
Day = DAY('XREF Date'[Date])
```

GOC FISCAL YEAR

Let's test it out in a table. Create a table and add in "Effective Date" from our transactions table followed by:

- "Fiscal Year"
- "Fiscal Quarter #"
- "Fiscal Period"
- "Day"

From the "XREF Date" table. Make sure you select "Don't summarize" in the value dropdown.



Effective Date	Fiscal Year	Fiscal Quarter #	Fiscal Period	Day
April-01-18	2018-19	1	1	1
April-02-18	2018-19	1	1	2
April-03-18	2018-19	1	1	3
April-04-18	2018-19	1	1	4
April-05-18	2018-19	1	1	5
April-06-18	2018-19	1	1	6
April-07-18	2018-19	1	1	7
April-08-18	2018-19	1	1	8
April-09-18	2018-19	1	1	9
April-10-18	2018-19	1	1	10
April-11-18	2018-19	1	1	11
April-12-18	2018-19	1	1	12
April-13-18	2018-19	1	1	13
April-14-18	2018-19	1	1	14
April-15-18	2018-19	1	1	15
April-16-18	2018-19	1	1	16
April-17-18	2018-19	1	1	17
April-18-18	2018-19	1	1	18
April-19-18	2018-19	1	1	19
April-20-18	2018-19	1	1	20
April-21-18	2018-19	1	1	21
April-22-18	2018-19	1	1	22
April-23-18	2018-19	1	1	23
April-24-18	2018-19	1	1	24
April-25-18	2018-19	1	1	25
April-26-18	2018-19	1	1	26
April-27-18	2018-19	1	1	27
April-28-18	2018-19	1	1	28
April-29-18	2018-19	1	1	29

Questions?