Takaisin välilehdelle

Tehty: Käy oppitunti läpi loppuun asti

Tasks

In this exercise, you will learn about some common advanced transforms that you may have a need for, which include **Conditional Columns**, **Fill Down**, **Unpivot**, **Merge Queries**, and **Append Queries**.

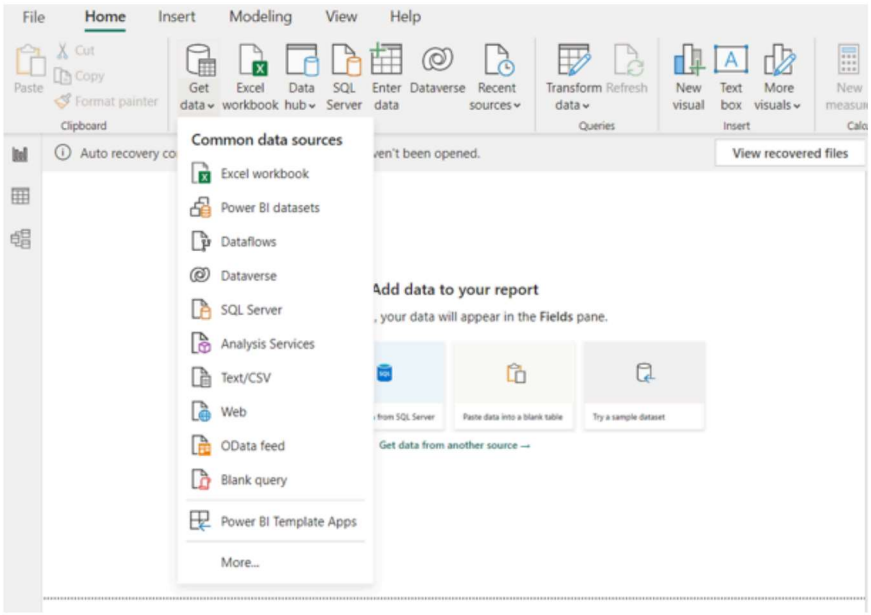
Task 1 - Add Conditional Columns

Using the Power Query Editor *Conditional Columns* functionality is a great way to add new columns to your query that follow logical if/then/else statements. This concept of if/then/else is common across many programming languages, including Excel formulas. Let's review a real-world scenario where you would be required to do some data cleansing on a file before it could be used.

In this example, you will be provided with a file of all the counties in the United States, and you must create a new column that extracts the state name from the county column and places it in its own column.

Step 1: Click link to download the file called **FIPS_CountyName.zip,** and unpack it to **C:\PBExams.**

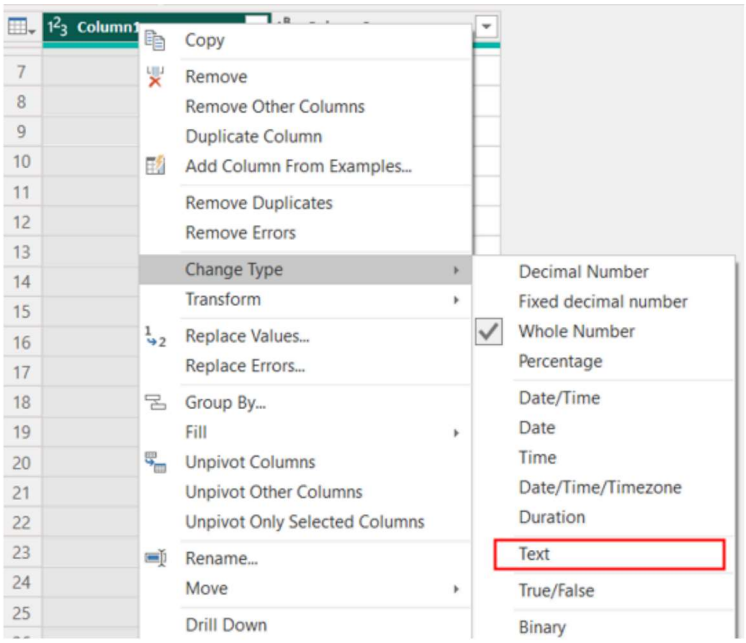
Step 2: Launch **Power BI Desktop**, and select **Get data** at **Home** ribbon and select **Text/CSV**



Step 3: Select on the **FIPS_CountyName.txt** file that is available in **C:\PBExams,** and click **Open**.

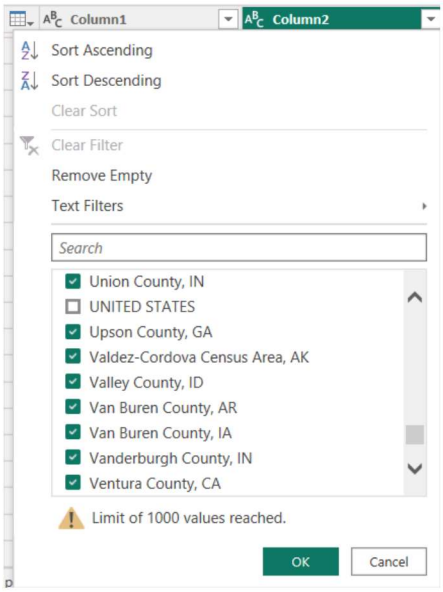
Step 4: In the **Navigator** window choose **Transform Data**.

Step 5: Start by changing the data type of *Column1* to *Text*.

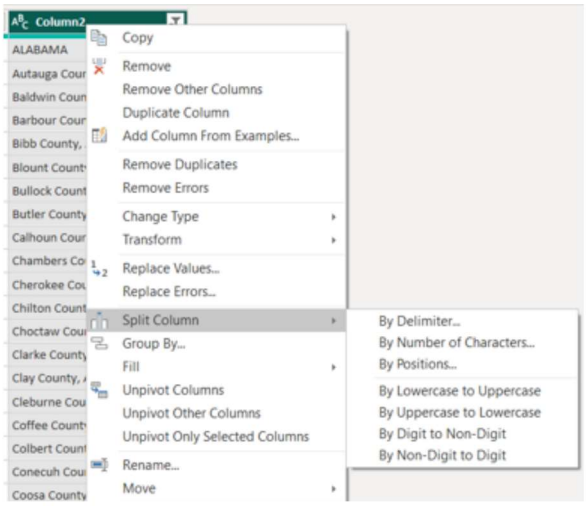


When you do this, you will be prompted to replace an existing type conversion. You can accept this by clicking **Replace current**. Notice that many of the values had leading zeros that were not visual before changing the data type.

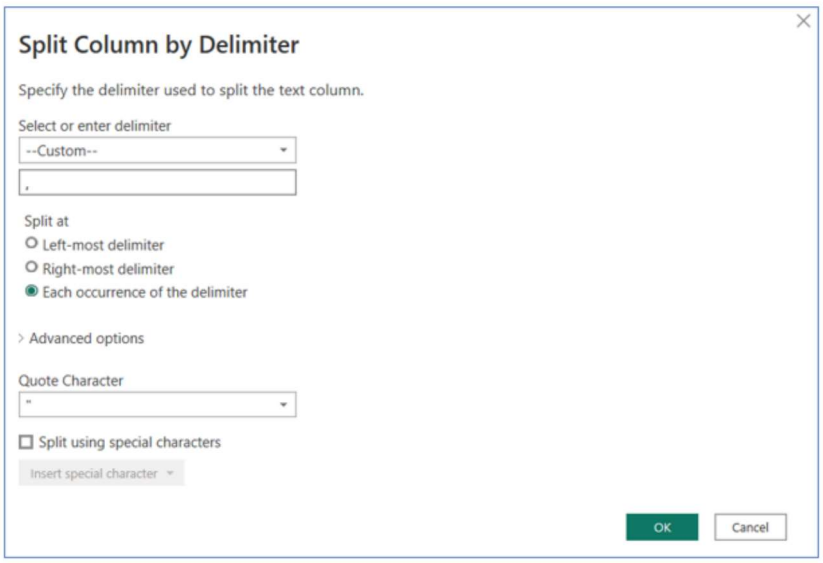
Step 6: Now, on *Column2*, filter out the value *UNITED STATES* from the column by clicking the arrow next to the column header and unchecking *UNITED STATES*. Then, click **OK**.



Step 7: Remove the state abbreviation from *Column2* by right-clicking on the column header and selecting **Split Column | By Delimiter**.



Choose ***-- Custom --*** for the delimiter type, and *type* a comma before clicking **OK**, as shown in figure below.



Step 8: Rename the column names *Column1*, *Column2.1*, and *Column2.2* to **County Code**, **County Name**, and **State Abbreviation**, respectively.

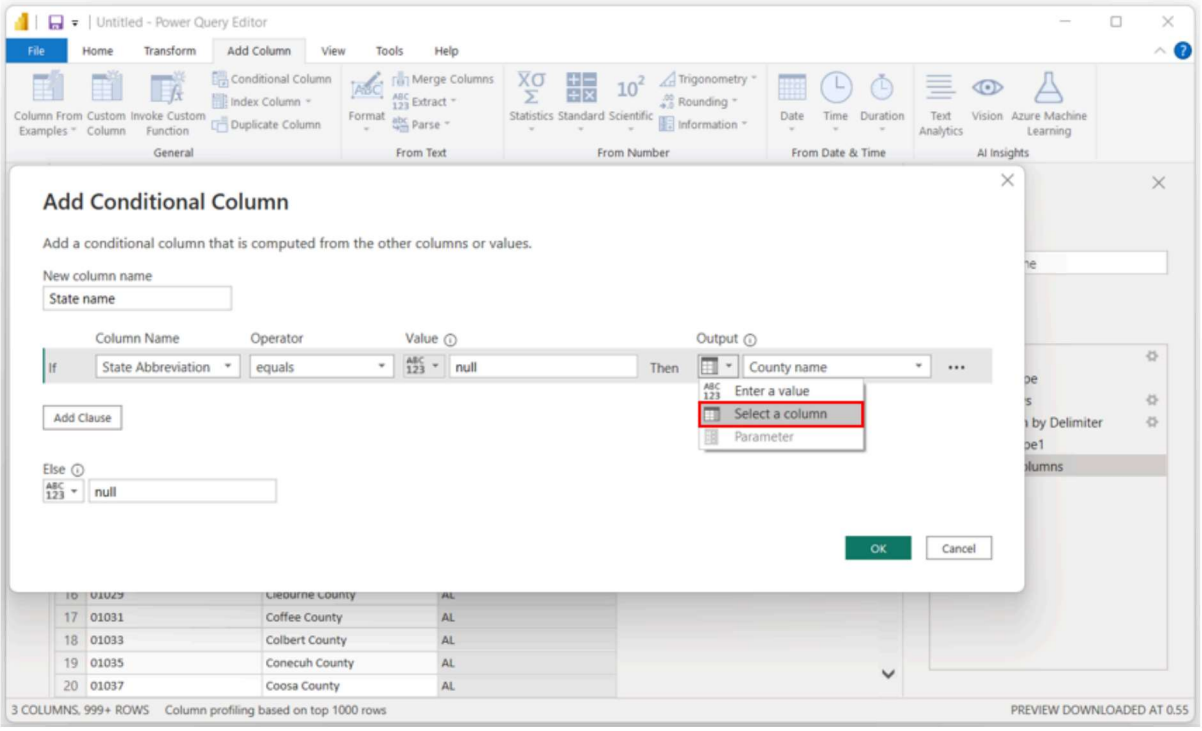
	County Code	County name	State Abbreviation
1	01000	ALABAMA	*null*
2	01001	Autauga County	AL
3	01003	Baldwin County	AL

Step 9: To isolate the full state name into its own column, you will need to implement **Conditional Column**. Go to the **Add Column** ribbon and select **Conditional Column**.

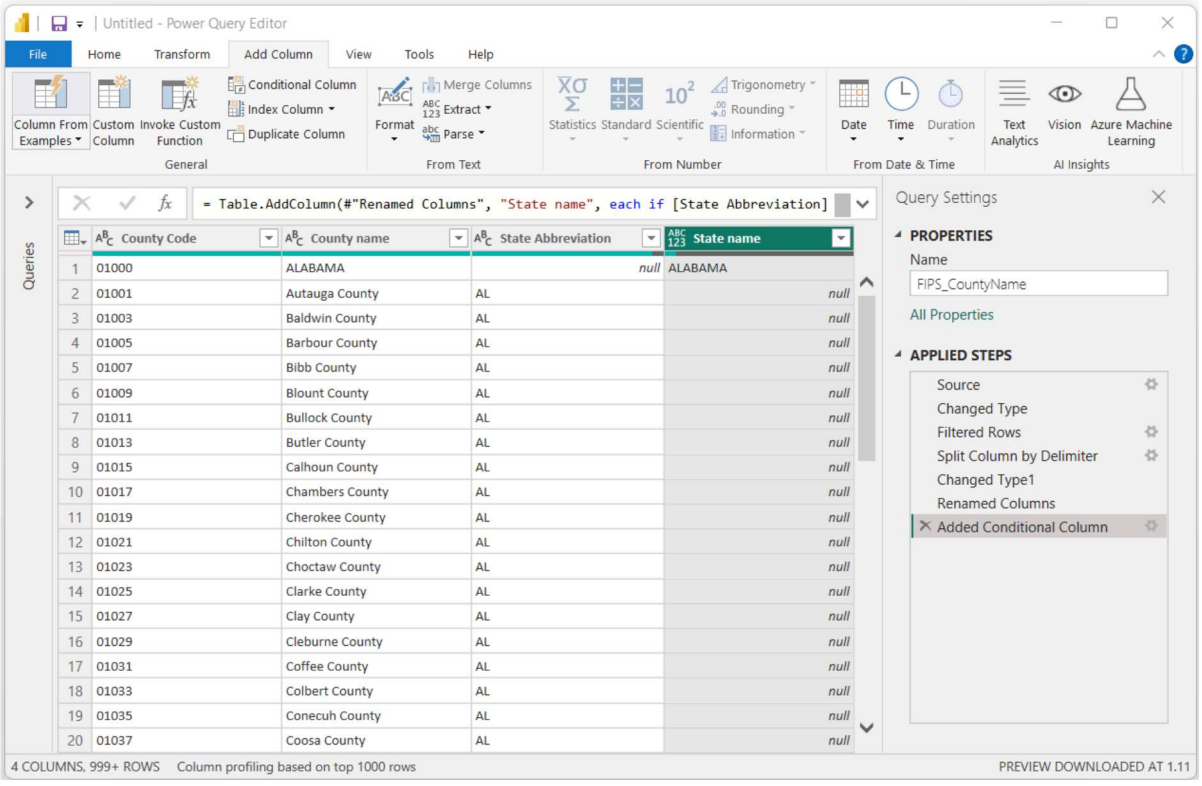
'Change the **New column name** property to **State Name** and implement the logic

> If State Abbreviation equals null Then return County Name Else return null

as shown in beow. To return the value from another column, you must select the icon in the Output property, then choose **Select a column**. Once this is complete, click **OK**:



This results in a new column called **State Name**, which has the fully spelled-out state name only appearing on rows where the *State Abbreviation* is null:



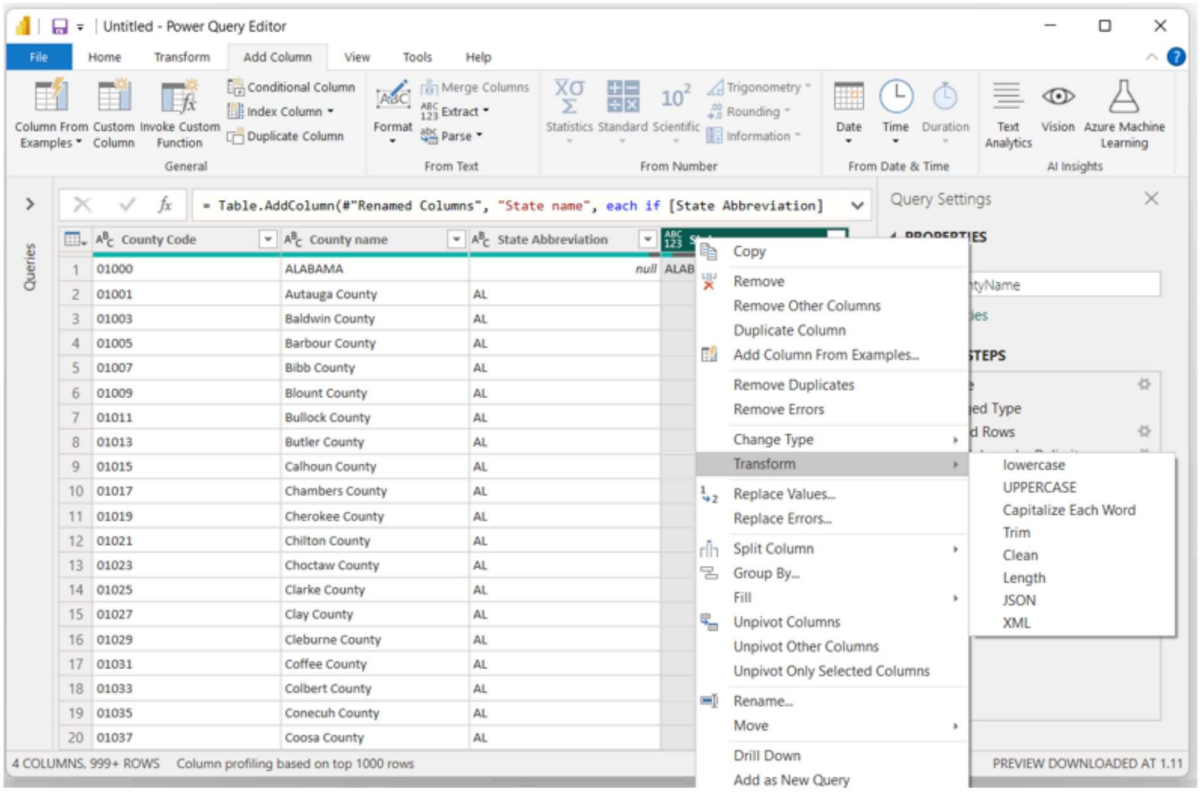
	County Code	County name	State Abbreviation	State name
1	01000	ALABAMA	null	ALABAMA
2	01001	Autauga County	AL	null
3	01003	Baldwin County	AL	null
4	01005	Barbour County	AL	null
5	01007	Bibb County	AL	null
6	01009	Blount County	AL	null
7	01011	Bullock County	AL	null
8	01013	Butler County	AL	null
9	01015	Calhoun County	AL	null
10	01017	Chambers County	AL	null
11	01019	Cherokee County	AL	null
12	01021	Chilton County	AL	null
13	01023	Choctaw County	AL	null
14	01025	Clarke County	AL	null
15	01027	Clay County	AL	null
16	01029	Cleburne County	AL	null
17	01031	Coffee County	AL	null
18	01033	Colbert County	AL	null
19	01035	Conecuh County	AL	null
20	01037	Coosa County	AL	null

You have now learned how to leverage the capabilities of the *Conditional Column* transform in the *Power Query Editor*.

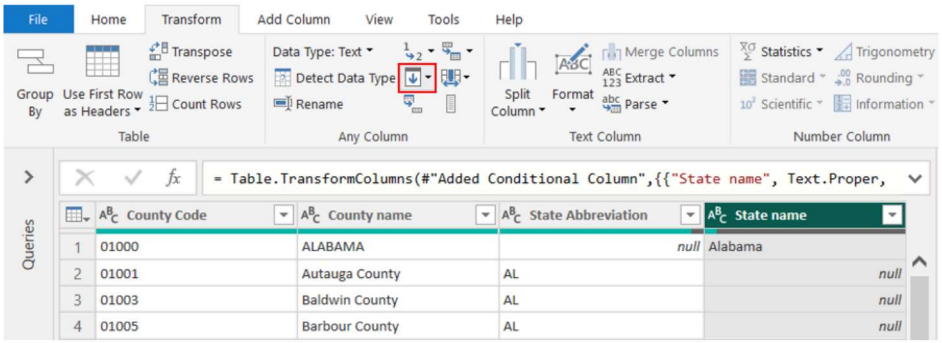
Task 2 - Fill Down

Fill Down is a rather unique transform in how it operates. By selecting **Fill Down** on a particular column, a value will replace all null values below it until another non-null appears. When another non-null value is present, that value will then fill down to all subsequent null values. To examine this transform, you will pick up from where you left off in the **Conditional Column** example in the previous task:

Step 1: Right-click on the **State Name** column header and select **Transform | Capitalize Each Word**. This transform should be self-explanatory.



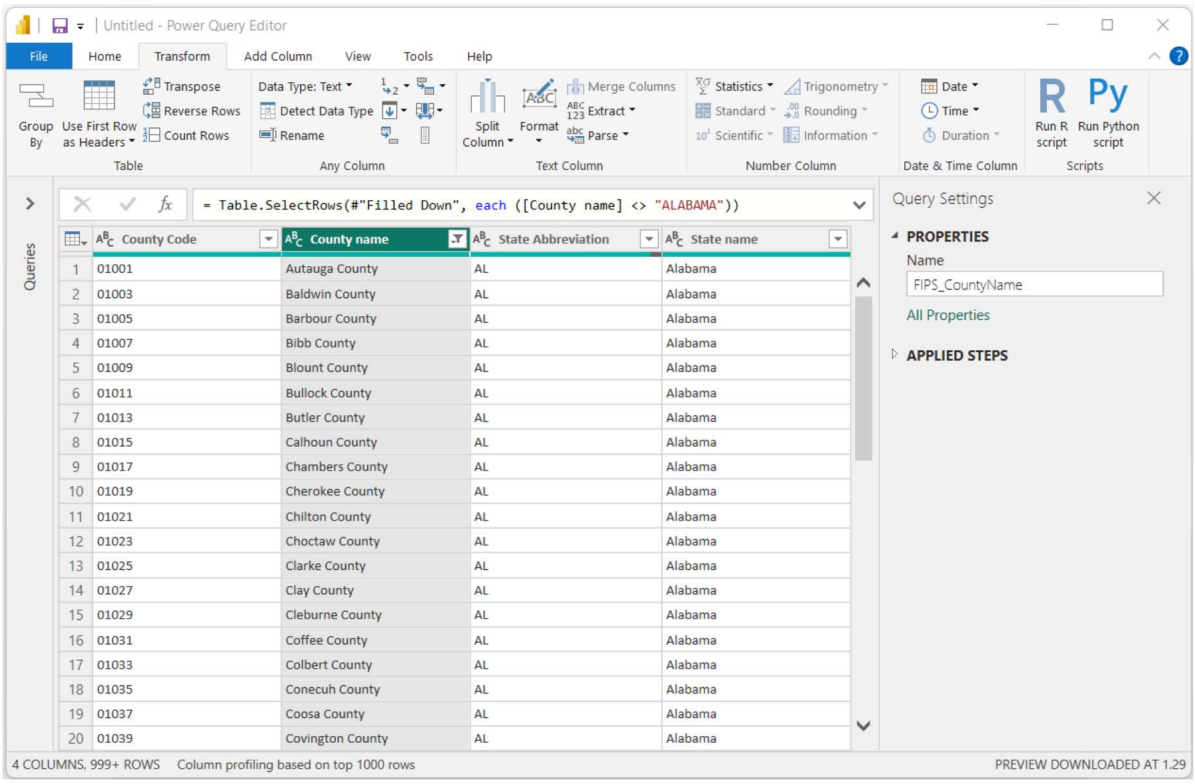
Step 2: Select the **State Name** column and, in the Transform ribbon, select **Fill | Down.**



	County Code	County name	State Abbreviation	State name
1	01000	ALABAMA	null	Alabama
2	01001	Autauga County	AL	null
3	01003	Baldwin County	AL	null
4	01005	Barbour County	AL	null

This will take the value in the **State Name** column and replace all non-null values until there is another **State Name** value that it can switch to. After performing this transform, scroll through the results to ensure that the value of *Alabama* switches to *Alaska* when appropriate.

To finish this example, filter out any null values that appear in the **State Abbreviation** column. The final result should look like figure below:



	County Code	County name	State Abbreviation	State name
1	01001	Autauga County	AL	Alabama
2	01003	Baldwin County	AL	Alabama
3	01005	Barbour County	AL	Alabama
4	01007	Bibb County	AL	Alabama
5	01009	Blount County	AL	Alabama
6	01011	Bullock County	AL	Alabama
7	01013	Butler County	AL	Alabama
8	01015	Calhoun County	AL	Alabama
9	01017	Chambers County	AL	Alabama
10	01019	Cherokee County	AL	Alabama
11	01021	Chilton County	AL	Alabama
12	01023	Choctaw County	AL	Alabama
13	01025	Clarke County	AL	Alabama
14	01027	Clay County	AL	Alabama
15	01029	Cleburne County	AL	Alabama
16	01031	Coffee County	AL	Alabama
17	01033	Colbert County	AL	Alabama
18	01035	Conecuh County	AL	Alabama
19	01037	Coosa County	AL	Alabama
20	01039	Covington County	AL	Alabama

In this task, you learned how you can use **Fill Down** to replace all of the null values below a non-null value. You can also use **Fill Up** to do the opposite, which would replace all the null values above a non-null value. One important thing to note is that the data must be sorted properly for **Fill Down** or **Fill Up** to be successful. In the next task, you will learn about another advanced transform, known as *Unpivot*.

Task 3 - Unpivot

The **Unpivot transform** is an incredibly powerful transform that allows you to reorganize your dataset into a more structured format best suited for BI. Let's discuss this by visualizing a practical example to help understand the purpose of **Unpivot**. Imagine you are provided with a file that contains the populations of US states and territories by decades, as shown below.

Year	Alabama	Alaska	American Samoa	Arizona
1960	3,266,740	226,167	20,051	1,302,161
1970	3,444,165	300,382	27,159	1,770,900
1980	3,893,888	401,851	32,297	2,718,215
1990	4,040,587	550,043	46,773	3,665,228
2000	4,447,100	626,932	57,291	5,130,632
2010	4,779,736	710,231	55,519	6,392,017
2020	5,024,279	733,391	49,710	7,151,502

The problem with data stored like this is you cannot very easily answer simple questions. For example, how would you answer questions like, *What was the total population for all states and territories in the US in 2020?* or *What was the average state and territory population in 2010?* With the data stored in this format, simple reports are made rather difficult to design. This is where the Unpivot transform can be a lifesaver.

Using **Unpivot**, you can change this dataset into something more acceptable for an analytics project:

Name	Year	Population
Minnesota	1960	3413864
Colorado	1960	1753947
Rhode Island	1960	859488
Mississippi	1960	2178141
Oklahoma	1960	2328284
North Carolina	1960	4556155
South Carolina	1960	2382594
Iowa	1960	2757537
Massachusetts	1960	5148578
Kansas	1960	2178611
South Dakota	1960	680514
New Jersey	1960	6066782
American Samoa	1960	20051
Louisiana	1960	3257022
Vermont	1960	389881
Montana	1960	674767
Pennsylvania	1960	11319366
Kentucky	1960	3038156

Data stored in this format can now easily answer the questions posed earlier by simply dragging a few columns into your visuals. Accomplishing this in other programming languages would often require fairly complex logic, while the **Power Query Editor** does it in just a few clicks.

There are three different methods for selecting the **Unpivot** transform that you should be aware of, and they include the following options:

1. **Unpivot Columns**: Turns any selected column's headers into row values and the data in those columns into a corresponding row. With this selection, any new columns that may get added to the data source *will* automatically be included in the Unpivot transform.
2. **Unpivot Other Columns**: Turns all column headers that *are not* selected into row values and the data in those columns into a corresponding row. With this selection, any new columns that may get added to the data source *will* automatically be included in the Unpivot transform.
3. **Unpivot Only Selected Columns**: Turns any selected columns' headers into row values and the data in those columns into a corresponding row. With this selection, any new columns that may get added to the data source *will not* be included in the Unpivot transform.

Let's walk through two examples of using the **Unpivot** transform to show you the first two of these methods, and provide an understanding of how this complex problem can be solved with little effort in Power BI. The third method mentioned for doing Unpivot will not be shown since it's so similar to the first option.

Unpivot Columns

Step 1: Click link to download the file called **Income Per Person.xlsx,** and save it to **C:\PBExams.**

Launch a new instance of **Power BI Desktop**, and use the Excel Workbook connector to import the workbook called **Income Per Person.xlsx** from **C:\PBExams**.

Once you select this workbook, choose the spreadsheet called *Data* in the **Navigator** window, and then select **Transform Data** to launch the **Power Query Editor**.

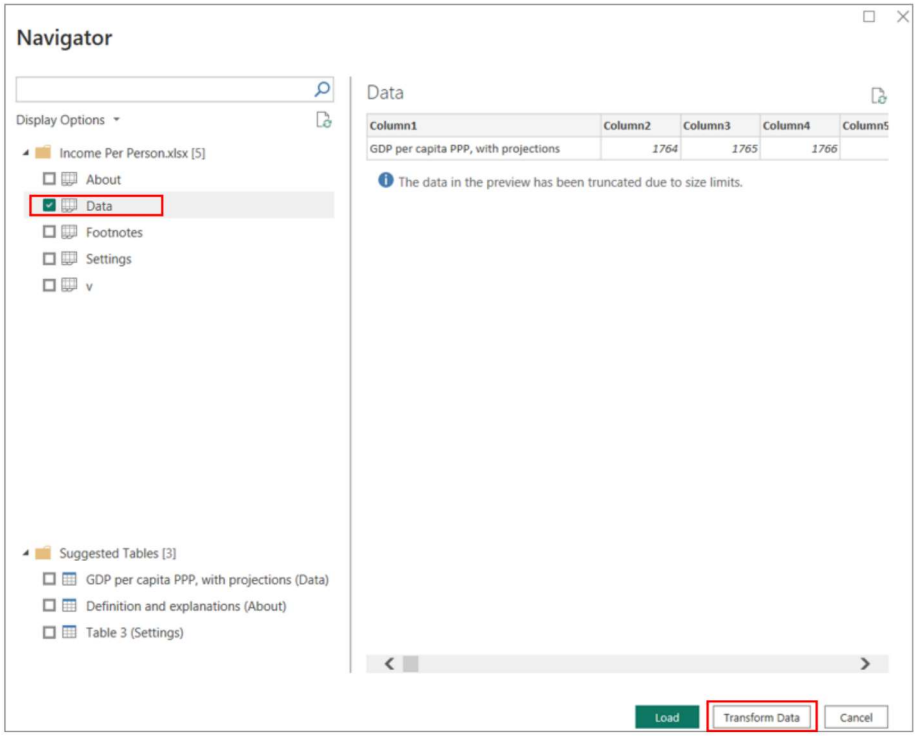


Figure shows what our data looks like before the **Unpivot** operation:

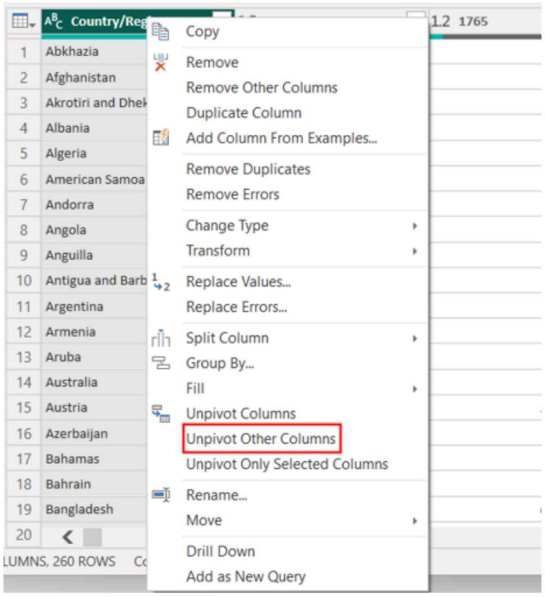
	Column1	Column2	Column3	Column4
1	GDP per capita PPP, with projections	1764	1765	
2	Abkhazia	null	null	
3	Afghanistan	null	null	
4	Akrotiri and Dhekelia	null	null	
5	Albania	null	null	
6	Algeria	null	null	
7	American Samoa	null	null	
8	Andorra	null	null	
9	Angola	null	null	
10	Anguilla	null	null	
11	Antigua and Barbuda	null	null	
12	Argentina	null	null	
13	Armenia	null	null	
14	Aruba	null	null	
15	Australia	null	null	
16	Austria	1403,59853	1404,448052	140
17	Azerbaijan	null	null	
18	Bahamas	null	null	
19	Bahrain	null	null	
20				

Step 2: Now, make the first row of data into column headers by selecting the transform called **Use First Row as Headers** on the **Home** ribbon.

Step 3: Rename the **GDP per capita PPP**, with projections column to *Country/Region*.

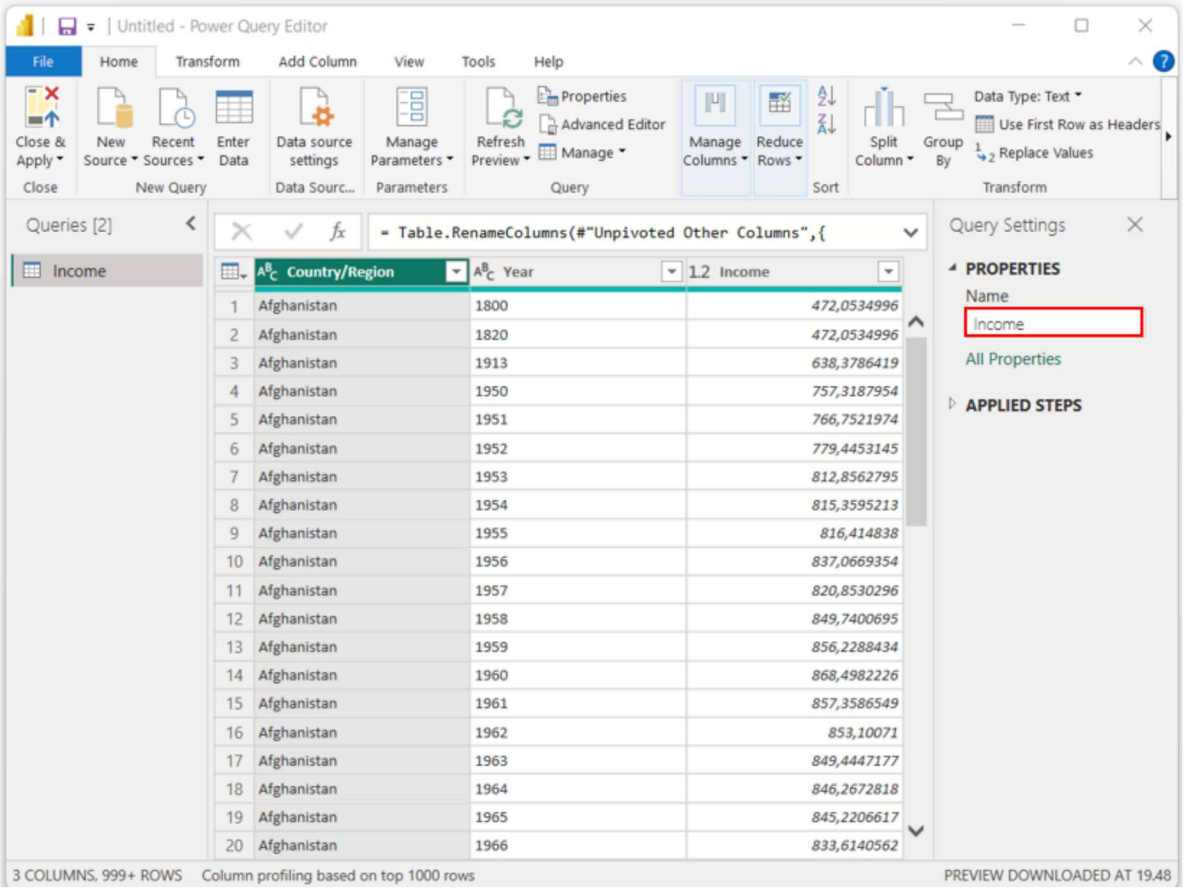
	Country/Region	1.2 1764	1.2 1765	1.2 1766	1.2
1	Abkhazia	null	null	null	
2	Afghanistan	null	null	null	
3	Akrotiri and Dhekelia	null	null	null	
4	Albania	null	null	null	
5	Algeria	null	null	null	
6	American Samoa	null	null	null	
7	Andorra	null	null	null	
8	Angola	null	null	null	
9	Anguilla	null	null	null	
10	Antigua and Barbuda	null	null	null	
11	Argentina	null	null	null	
12	Armenia	null	null	null	
13	Aruba	null	null	null	
14	Australia	null	null	null	
15	Austria	1403,59853	1404,448052	1405,298088	
16	Azerbaijan	null	null	null	
17	Bahamas	null	null	null	
18	Bahrain	null	null	null	
19	Bangladesh	614,52257	614,3627258	614,2029232	
20					

If you look closely at the column headers, you can tell that most of the column names are actually years and the values inside those columns are the income for those years. This is not the ideal way to store this data because it would be incredibly difficult to answer a question like, *What is the average income per person for Belgium?* To make it easier to answer this type of question, right-click on the **Country/Region** column and select **Unpivot Other Columns**.



Step 4: Rename the columns Attribute and *Value* to *Year* and *Income*, respectively.

Step 5: To finish this first part , you should also rename this query **Income**. The results of these first steps can be seen ibelow.



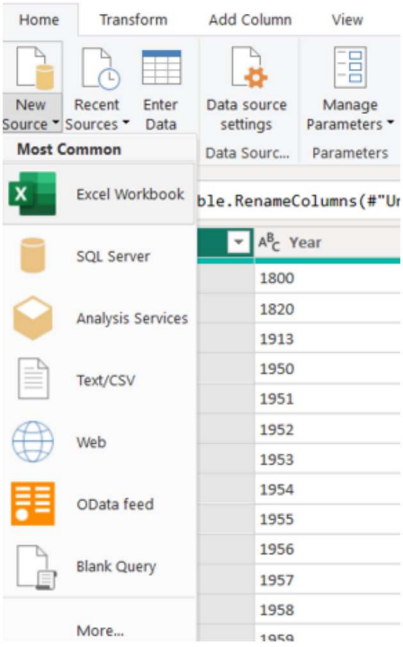
Unpivot other Columns

This first method walked you through what can often be the fastest method for performing an Unpivot transform.

In the following, you will learn how to use the **Unpivot Columns** method as well:

Step 1: Click link to download the file called **Total Population.xlsx,** and save it to **C:\PBExams.**

Remain in the **Power Query Editor**, and select **New Source** from the **Home** ribbon.



Use the Excel Workbook connector to import the **Total Population.xlsx** workbook from **C:\PBExams**.

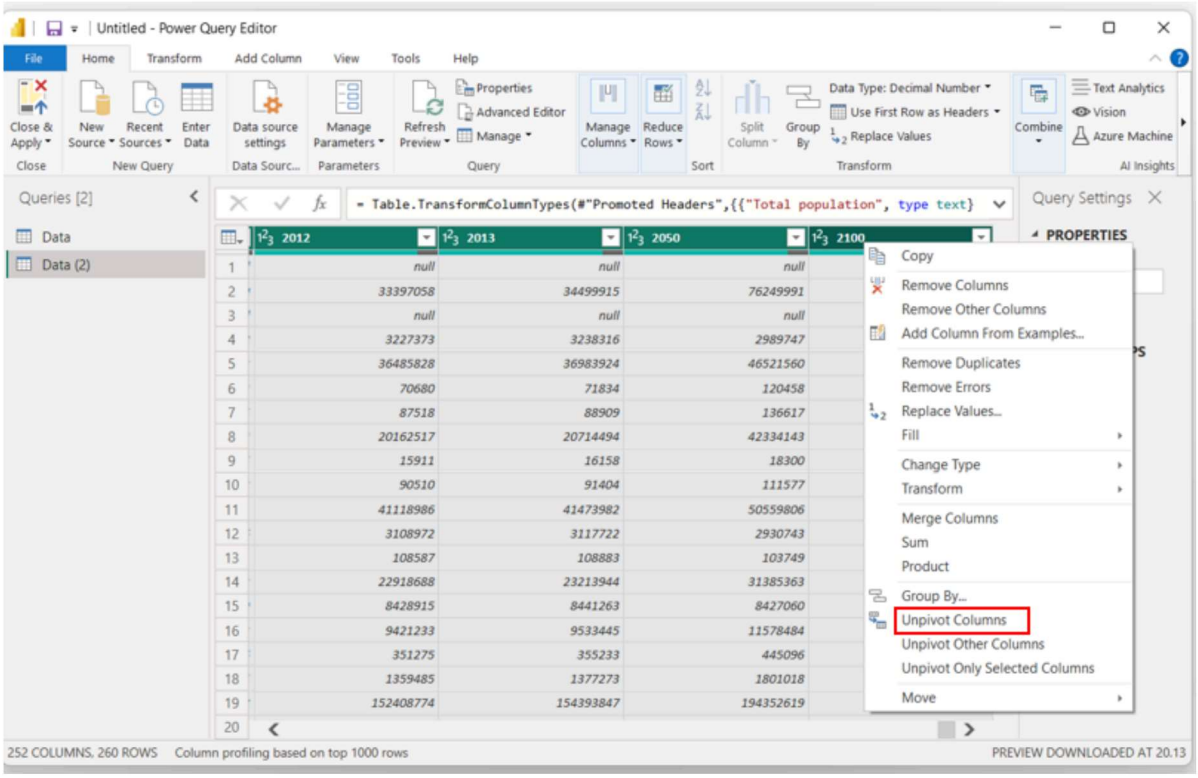
Once you select this workbook, choose the spreadsheet called **Data** in the Navigator window, and then select **OK**. Figure shows the dataset before **Unpivot** has been added:

	Column1	Column2	Column3	Column4
1	Total population	1086	1100	1290
2	Abkhazia	null	null	null
3	Afghanistan	null	null	null
4	Akrotiri and Dhekelia	null	null	null
5	Albania	null	null	null
6	Algeria	null	null	null
7	American Samoa	null	null	null
8	Andorra	null	null	null
9	Angola	null	null	null
10	Anguilla	null	null	null
11	Antigua and Barbuda	null	null	null
12	Argentina	null	null	null
13	Armenia	null	null	null
14	Aruba	null	null	null
15	Australia	null	null	null
16	Austria	null	null	null
17	Azerbaijan	null	null	null
18	Bahamas	null	null	null
19	Bahrain	null	null	null
20				

Step 2: Like the last example, you will again need to make the first row of data into column headers by selecting the transform called **Use First Row as Headers** on the **Home** ribbon.

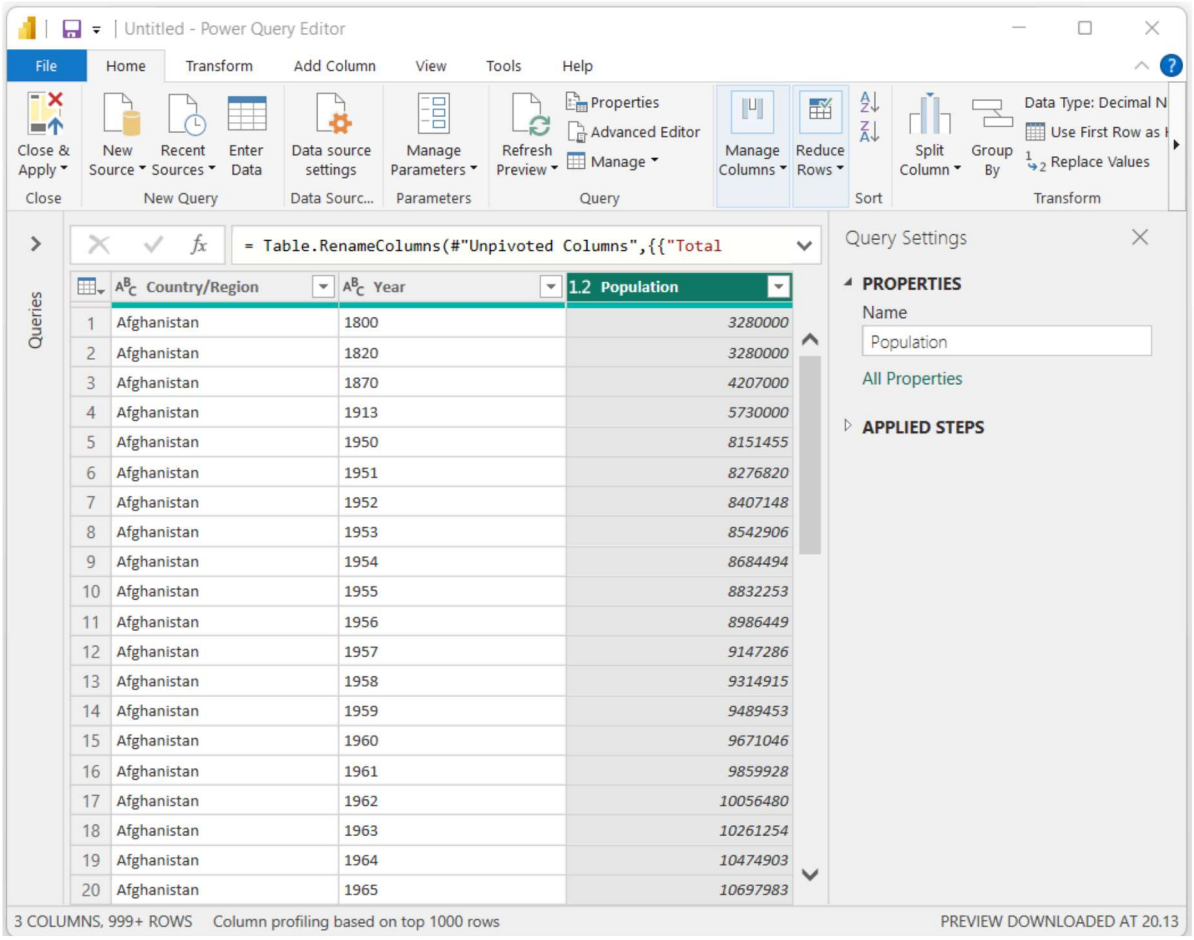
Step 3: Rename the column Total population to *Country/Region*.

Step 4: This time, multi-select all the columns except *Country/Region*, then right-click on one of the selected columns and choose **Unpivot Columns**, as shown in figure. The easiest way to multi-select these columns is to select the first year column and then hold *Shift* before clicking the last column:



Step 5: Rename the columns from *Attribute* and *Value* to *Year* and *Population*, respectively, to match the results.

Step 6: Finalize this query by renaming it *Population* in the **Query Settings** pane.



In this task, you learned about two different methods for performing an **Unpivot**. To complete the data cleansing process on these two datasets, it's recommended that you continue to the next section on merging queries.

Task 4 - Merge Queries

Another common requirement when building BI solutions is the need to join two tables together to form a new outcome that includes some columns from both tables in the result. Fortunately, Power BI makes this task very simple with the **Merge Queries** feature. Using this feature requires that you select two tables and then determine which column or columns will be the basis of how the two queries are merged.

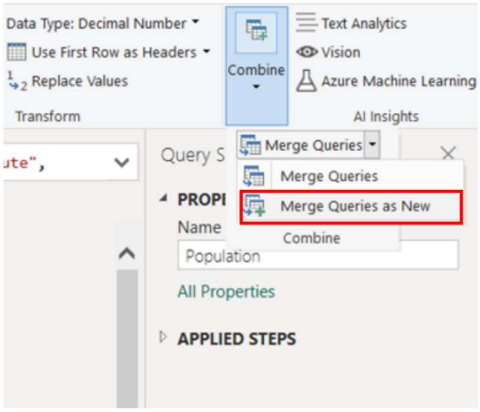
After determining the appropriate columns for your join, you will select a join type. The join types are listed here with the description that is provided within the product:

- **Left Outer** (all rows from the first table, only matching rows from the second)
- **Right Outer** (all rows from the second table, only matching rows from the first)
- **Full Outer** (all rows from both tables)
- **Inner** (only matching rows from both tables)
- **Left Anti** (rows only in the first table)
- **Right Anti** (rows only in the second table)

Many of you may already be very familiar with these different join terms from the SQL programming. However, if these terms are new to you, I recommend reviewing *Visualizing Merge Join Types in Power BI*, courtesy of Jason Thomas in the *Power BI Data Story Gallery*: https://community.powerbi.com/t5/Data-Stories-Gallery/Visualizing-Merge-Join-Types-in-Power-BI/m-p/219906. This visual aid is a favorite of many users that are new to these concepts.

To examine the **Merge Queries** option, you will pick up from where you left off with the **Unpivot** example in the previous task:

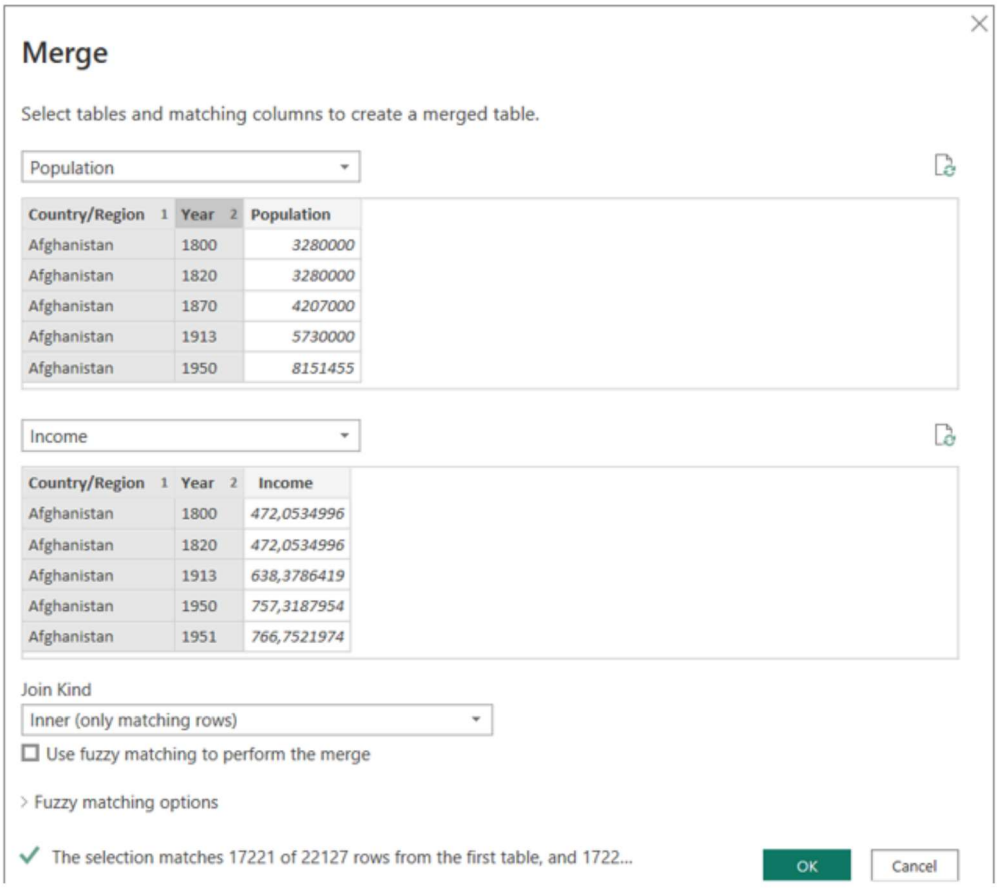
Step 1: With the **Population** query selected, find and select **Merge Queries | Merge Queries as New** on the **Home** ribbon.



Step 2: In the **Merge** dialog box, select the **Income** query from the drop-down selection in the middle of the screen.

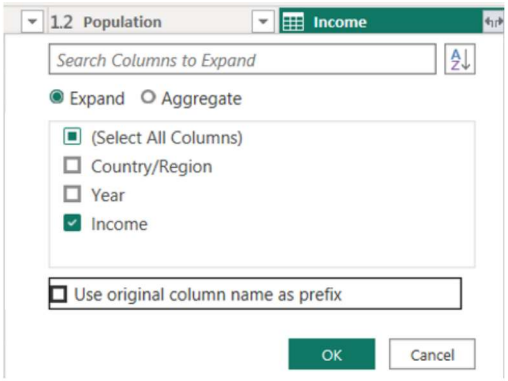
Step 3: Then, multi-select the **Country** and **Year** columns on the **Population** query, and do the same under the **Income** query. This defines which columns will be used to join the two queries together. Ensure that the number indicators next to the column headers match, as demonstrated in Figure below. If they don't, you could accidentally attempt to join the incorrect columns.

Step 4: Next, select **Inner (only matching rows)** for **Join Kind**. This join type will return rows only when the columns you chose to join on have values that exist in both queries. Before you click **OK**, confirm that your screen matches the figure below*:*



Step 5: Once you select **OK**, this will create a new query called **Merge** that combines the results of the two queries. Go ahead and rename this query *Country Statistics*.

Step 6: You will also notice that there is a column called *Income* that has a value of **Table** for each row. This column is actually representative of the entire Income query that you joined to. To choose which columns you want from this query, click the **Expand** button on the column header. After clicking the **Expand** button, uncheck *Country*, *Year*, and *Use original column name as prefix*, then click **OK**.

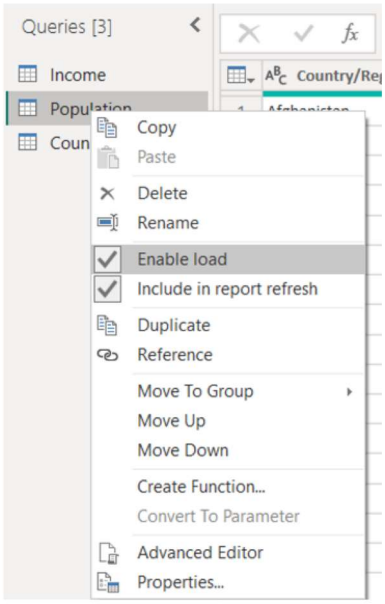


Step 7: Rename the column called **Income.1** to *Income*. Figure shows this step completed:

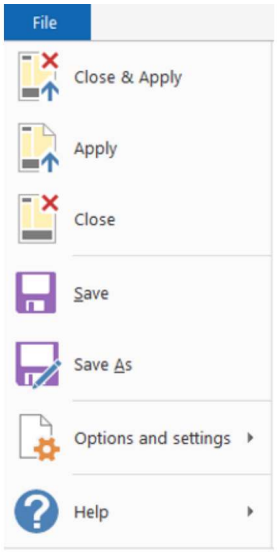
	Country/Region	Year	1.2 Population	1.2 Income
1	Afghanistan	1800	3280000	472,0534996
2	Afghanistan	1820	3280000	472,0534996
3	Afghanistan	1913	5730000	638,3786419
4	Afghanistan	1950	8151455	757,3187954
5	Afghanistan	1951	8276820	766,7521974
6	Afghanistan	1952	8407148	779,4453145
7	Afghanistan	1953	8542906	812,8562795
8	Afghanistan	1954	8684494	815,3595213
9	Afghanistan	1955	8832253	816,414838
10	Afghanistan	1956	8986449	837,0669354

Step 8: Finally, since you chose the option **Merge Queries as New** in Step1, you can disable the load option for the original queries that you started with. To do this, right-click on the **Population** query in the **Queries** pane and click **Enable load** to disable it.

Do the same thing for the **Income** query. Disabling these queries means that the only query that will be loaded into your Power BI data model is the new one, called **Country Statistics**:



To begin using this dataset in a report, you would click **Close & Apply**.



By default, merging queries together relies on exact matching values between your join column(s). However, you may work with data that does not always provide perfect matching values. For example, a user enters data and misspells their country as Unite States instead of United States. In those cases, you may consider the more advanced feature called **fuzzy matching**. With fuzzy matching, *Power BI* can perform an approximate match and still join on these two values based on their similarity. In this section, you learned how the *Merge Queries* option is ideal for joining two queries together. In the next task, you will learn how to solve the problem of performing a union of two or more queries.

Task 5 - Append Query

Occasionally, you will work with multiple datasets that need to be appended to each other. Here's a scenario: you work for a customer service department for a company that provides credit or loans to customers. You are regularly provided with .csv and .xlsx files that give summaries of customer complaints regarding credit cards and student loans. You would like to analyze both of these data extracts at the same time but, unfortunately, the credit card and student loan complaints are provided in two separate files. In this example, you will learn how to solve this problem by performing an append operation on these two different files:

Step 1: Click link to download the file called **Student Loan Complaints.xlsx,** and save it to **C:\PBExams.**

Click link to download the file called **Credit Card Complaints.csv,** and save it to **C:\PBExams.**

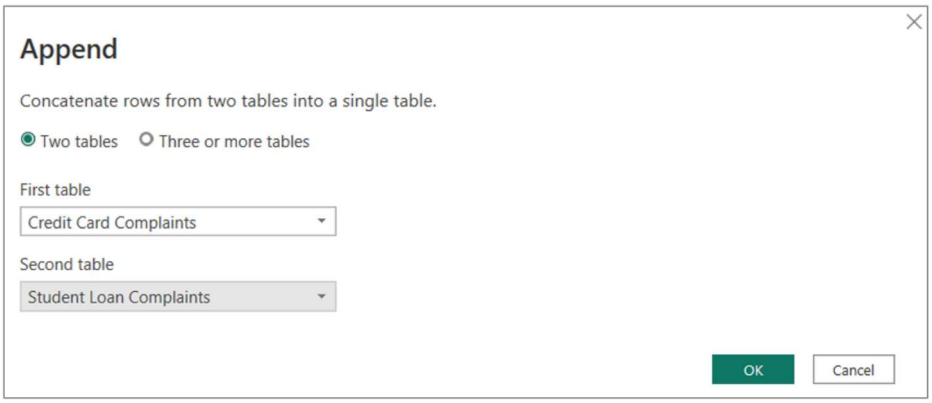
Step 2: Launch a new instance of **Power BI Desktop**, and use the Excel Workbook connector to import the workbook called ***Student Loan Complaints.xlsx*** from **C:\PBExams**.

Once you select this workbook, choose the spreadsheet called **Student Loan Complaints** in the **Navigator** window, and then select **Transform Data** to launch the **Power Query Editor**.

Step 3: Import the credit card data by selecting **New Source | Text/CSV**, then choose the file called **Credit Card Complaints.csv** from **C:\PBExams.** Click **OK** to bring this data into the **Power Query Editor.**

Step 4: With the **Credit Card Complaints** query selected, find and select **Append Queries | Append Queries as New** on the **Home** ribbon.

Step 5: Select **Student Loan Complaints** as the table to append to, then select **OK**, as shown in below.



Step 6: Rename the newly created query **All Complaints** and view the results, as seen in below.

	Complaint ID	Product	Sub-product	Issue	Sub-issue	State
1	1429682	Credit card		Credit line increase/decrease		NC
2	1430686	Credit card		Billing statement		PA
3	1429869	Credit card		Advertising and marketing		IL
4	1427544	Credit card		Billing disputes		WI
5	1427873	Credit card		APR or interest rate		NJ
6	1428112	Credit card		Credit determination		NY
7	1430459	Credit card		Billing disputes		NJ
8	1425175	Credit card		Privacy		NC
9	1425203	Credit card		Other		FL
10	1424483	Credit card		Advertising and marketing		NY
11	1424843	Credit card		Billing disputes		FL
12	1422868	Credit card		Advertising and marketing		MN
13	1423026	Credit card		Late fee		VA
14	1423176	Credit card		Customer service / Customer relations		CA
15	1423531	Credit card		Closing/Cancelling account		WA
16	1424194	Credit card		Other		CA
17	1424522	Credit card		Late fee		MD
18	1421071	Credit card		Identity theft / Fraud / Embezzlement		CA
19	1420918	Credit card		Billing disputes		MD
20						

Step 7: Similar to the previous example, you would likely want to disable the load option for the original queries that you started with. To do this, right-click on the **Student Load Complaints** query in the **Queries** pane and click **Enable load** to disable it.

Step 8: Do the same to the **Credit Card Complaints** query, and then select **Close & Apply.**

End-of-Exercise

Olet suorittanut 100 % oppitunnista

100%

◄ Exercise 3 - Transformation basics

Siirry...

Exercise 5 - R programming language (not working, can be omitted) ►

PowerBI

Suomi (fi)
- Deutsch (de)
- English (en)
- Français (fr)
- Suomi (fi)
- Svenska (sv)

Hanki mobiilisovellus

Metropolia