Microsoft

Collect, Combine, and Transform Data Using Power Query in Excel and Power BI

Gil Raviv

Sample files on the web

Microsoft

Collect, Combine, and Transform Data Using Power Query in Excel and Power BI

Gil Raviv

construct the new columns from these unique values. As a result, this technique is not ideal if you have large datasets and want to combine the tables in an optimized way and with a shorter refresh time.

Exercise 4-6: Transposing Column Names Only

A performance-optimized method to transform column names into a column and set the stage for the merge of the conversion table is to transpose only the column names as an intermediate transformation phase and then apply the merge. Figure 4-8 shows how you can transpose the column names, apply the normalization, and transpose the column names back into the original table.

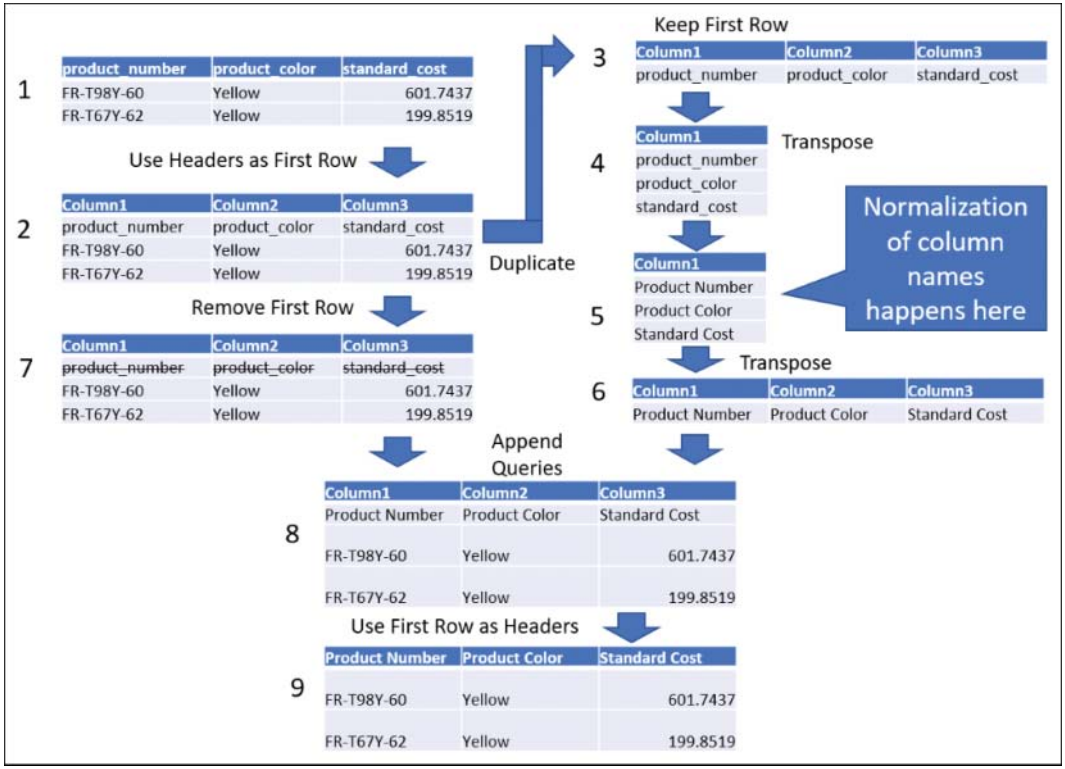

FIGURE 4-8 Normalization of column names involves nine transformation steps.

While the technique illustrated in Figure 4-8 is easy to implement when you combine a few specific files using Append Queries, applying it on a folder is quite advanced, as it relies on custom functions; doing this is therefore recommended here only for learning purposes. You should use this technique if the techniques described already in Exercises 4-4 and 4-5 are not relevant due to performance constraints, and only if you prefer to avoid using the M technique described in Exercise 4-7.

1. Start with the saved queries created in steps 1–9 of Exercise 4-4.
2. In Applied Steps, while the Products Sample query is still selected, delete the step Promoted Headers. Now the headers of the sample table are in the first row.

3. In the Queries pane, right-click Products Sample and select Duplicate. Now, while the new query, Products Sample (2), is selected, select Keep Rows on the Home tab and then select Keep Top Rows. In the Keep Top Rows dialog box, in the Number of Rows box, enter *1* and click OK.

Note When you combine files from a folder, Power Query creates a query function, a sample query, a file query, and a parameter. Those artifacts help you combine multiple files from a folder based on a single file you select. The main transformation for each file is implemented in the function. To help you customize the transformation on each file, you can make changes on the sample query. In this exercise, the sample query is Transform Sample File from Appended Products, and you renamed it *Products Sample*. Changes made in the sample query will propagate to the function and, as a result, will be applied on all the files in the folder. But here is a crucial caveat: If you use Duplicate or Reference on sample queries, the transformations in the new query will not be propagated by Power Query when you combine the files from the folder—unless you apply an advanced technique to "wire" back the new query to the transformation sequence in the sample query. You will apply this wiring in steps 21 and 22.

4. On the Transform tab, select Transpose to set the stage for the normalization. This is the point where you will apply a merge sequence similar to the one described in steps 12–19 of Exercise 4-4. However, in this case, don't follow those steps. There are few variations here that are crucial.
5. While Products Sample (2) is selected, select Merge Queries on the Home tab.
6. When the Merge dialog box opens, select Column1 in Products Sample (2), and in the drop-down below Products Sample (2), select Header_Conversion. Then select the Source column, as you did in Exercise 4-4 (refer to Figure 4-6).
7. Ensure that Join Kind is set to Left Outer (All from First, Matching from Second) and click OK. In the Power Query Editor, the new Header_Conversion column is added, with *table* objects as values.
8. At this stage, you need to add an index column to ensure that the next step will not change the order of the column names in Column1, so on the Add Column tab, select Index Column. In step 13 you will learn why this step was important.
9. Select the Header_Conversion column and select Expand on the Transform tab.
10. In the Expand pane, deselect Source and click OK. The new column is transformed to the new column Header_Conversion.Target, with the Target values in rows where the product column names are mismatched and with null value in rows where you should keep the original product column names.
11. On the Add Column tab, select Conditional Column.
12. When the Add Conditional Column dialog box opens, set it as follows:
    a. In the New Column Name box, enter *New Column Names*.
    b. In the Column Name box, select Header_Conversion.Target.
    c. In the Value box, enter *null*.

d. In the left Output drop-down, select Select a Column.

e. In the right Output drop-down, select Column1.

f. In the left Otherwise drop-down, select Select a Column.

g. In the right Otherwise drop-down, select Header_Conversion.Target and click OK.

13. Ensure that the Index column is still in ascending order. If it isn't, you can sort the Index column in ascending order. The order of the column names is crucial here. In step 9 you expanded the table, and that step can reshuffle the order of the rows. Having the Index column that was added in step 8 helps you preserve the order.

14. Delete the Column1, Header_Conversion.Target, and Index columns and keep only the last column, New Column Names.

15. On the Transform tab, select Transpose. You now have the normalized column names.

    You have finished the normalization part of the Products Sample (2) query, which is illustrated in Figure 4-4 (in step 5 in the figure). From here, you can wrap up the remaining sequence of the transformation steps.

16. To remove the old headers from the table, go back to the original Products Sample query. While Products Sample query is selected in the Queries pane, select Remove Rows on the Home tab and then select Remove Top Rows. In the Remove Top Rows dialog box, in the Number of Rows box, enter *1* and click OK.

17. While Products Sample is still selected in the Queries pane, select Append Queries on the Home tab. The Append dialog box opens. Select Products Sample (2) as Table to Append and click OK.

18. In the Preview pane, you will notice that the normalized column names from Products Sample (2) are now located as the bottom row because you appended Products Sample (2) as the second table. You can switch the tables. In formula bar, note the following formula:

    ```
    = Table.Combine({#"Removed Top Rows", #"Products Sample (2)"})
    ```

    Change the formula by switching the positions of #"Removed Top Rows" and #"Products Sample (2)", as shown here:

    ```
    = Table.Combine({#"Products Sample (2)", #"Removed Top Rows"})
    ```

19. Now that you have the normalized column names in the first row, promote them as headers. To do this, on the Transform tab, select Use First Row As Headers and remove Changed Type from Applied Steps.

Note Removing Changed Type is important here. The Size column for Accessories includes only numbers. Hence, Power Query converted the type of the Size column to Whole Number. But for other products, such as bikes, textual values are valid sizes. If you don't remove the type conversion at this stage, you will receive errors in cells of the Size column that should contain textual values.

20. In the Queries pane, select the query Appended Products to review the combined table. You will notice an error that can be fixed by removing the last step, Changed Type, from Applied Steps.

    Now when you scroll down in the Preview pane and reach the rows from C04E04 - Bikes.xlsx, you find out, as shown in Figure 4-9, that the values in the Name column and the Product Number column were swapped. The normalization didn't work well.

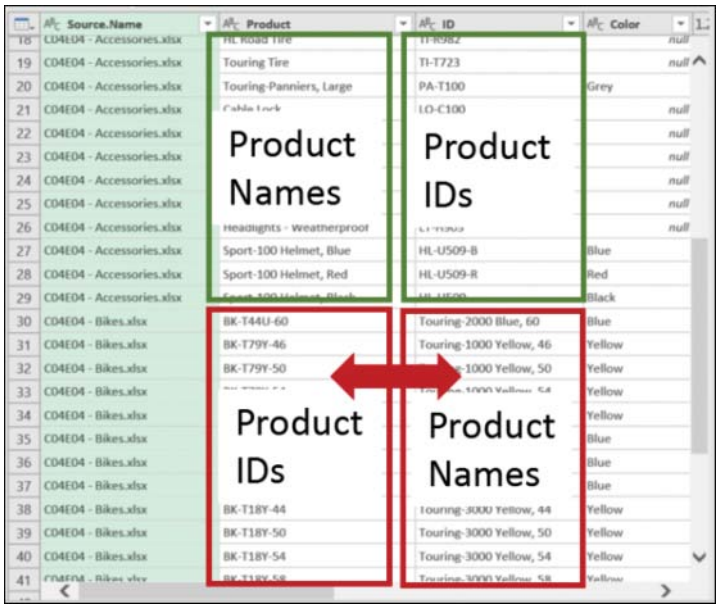

FIGURE 4-9 In the Appended Products query, product IDs and product names are swapped in the Bikes rows.

Tip Auditing the end results is an important step in using Power Query. The Preview pane can often be very misleading, as it shows partial data. In this case, everything worked fine for the Accessories data, but when you take a careful look at the appended results, you find that product ID and product name values are swapped for Bikes. Always take a careful look at the results data.

Let's pause here for a moment to understand what happened and why the normalization hasn't worked. The good news is that the Products Sample (2) query is now performing well on a single file. When you select Products Sample (2), you can see that in the last step in Applied Steps, the headers are normalized as expected. But the bad news is that the header normalization was done on only a single file, and now, when this sample query is reused via the custom function on all the files in the folder, it rigidly transforms all column names by the data in the same single file.

Why is the same single file used? Look at the Products Sample (2) query. When the Source step is selected in Applied Steps, the formula bar shows that the query loads the Excel workbook

from the #"Sample File Parameter1" parameter. This parameter was created by Power Query when you combined the files from the folder. You can select this parameter in the Queries pane to find that the parameter's current value is Sample File. You can find the Sample File query below #"Sample File Parameter1" in the Queries pane. This query returns the content of the workbook C04E04 - Accessories.xlsx.

Why were the product ID values and product names swapped for Bikes in the Appended Products query? This happened because the Accessories file, C04E04 - Accessories.xlsx, contains the two columns in the reverse order from C04E04 - Bikes.xlsx. You cannot run the normalization in the Accessories file and apply it on other files if the orders of the columns are different.

To fix this issue, you need to learn how to reuse the logic in Products Sample (2) per file in the folder. Recall from step 3 of this exercise that Products Sample (2) was duplicated from Products Sample, which is a sample query. As a sample query, Products Sample runs on each file in the folder and invokes Products Sample (2). But since Products Sample (2) is currently not a sample query, the files from the folder are not propagated to Products Sample (2). This query always runs its transformation sequence on the same file: C04E04 - Accessories.xlsx.

You need to find a way to convert Products Sample (2) into a reusable query that will accept a file as a parameter from the Products Sample query and normalize the headers for each file. To solve this challenge, you can turn Product Sample (2) into a function (also referred to as a custom function) that can be called and reused for each file in the folder. Carry out the following additional steps to create a function from Products Sample (2):

21. In the Queries pane, right-click Products Sample (2) and select Create Function. The Create Function dialog box opens. In the Function Name box, enter *FnNormalizeColumnNames* and click OK. As shown in Figure 4-10, you now have a new function that can be used on all the files.

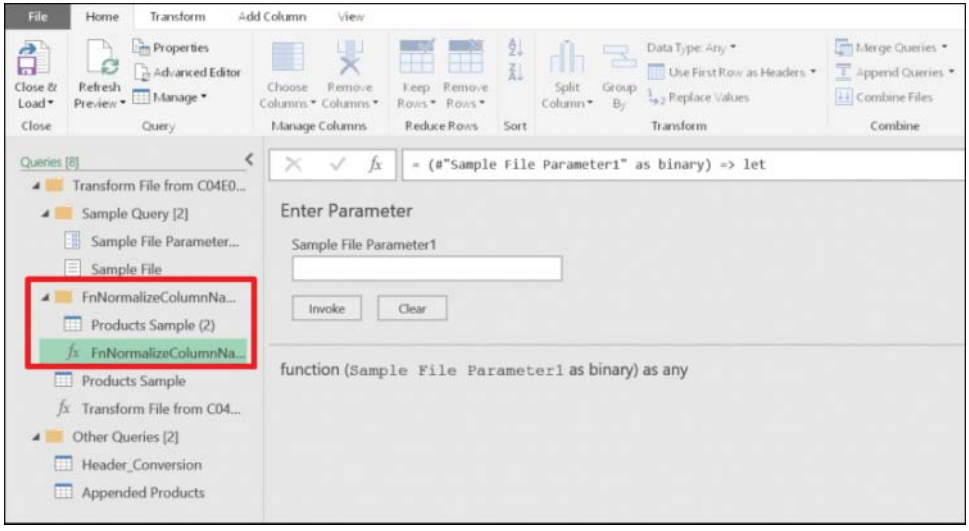

FIGURE 4-10 Products Sample (2) has been converted into the function FnNormalizeColumnNames with Products Sample (2) as its sample query.

Now you have a function, *FnNormalizeColumnNames*, that can be invoked inside the Products Sample query instead of invoking the Products Sample (2) query.

22. In the Queries pane, select the Products Sample query. In Applied Steps, select Appended Query. In the formula bar, replace the following formula:

```
= Table.Combine({#"Products Sample (2)", #"Removed Top Rows"})
```

with the following formula:

```
= Table.Combine({FnNormalizeColumnNames(#"Sample File Parameter1"), #"Removed Top Rows"})
```

The function *FnNormalizeColumnNames* receives an Excel workbook's contents from the folder as an argument and returns the normalized columns. The table #"Removed Top Rows" is the last step in Products Samples, before you applied the append. It includes the original rows without the header.

Why do you use #"Sample File Parameter1" here? Recall that this parameter was always pointing to the Accessories file when it was called inside Sample Products (2). But now, in the context of a function, this parameter is reassigned with the correct file, as Power Query iterates over the files in the folder.

23. Move back to the appended results in the Appended Products query and scroll down the Preview pane to confirm that all the columns of C04E04 - Bikes.xlsx are aligned correctly. Close the Power Query Editor and load the data to your report.

You can download the solution files C04E06 - Solution - Transpose Headers.xlsx and C04E06 - Solution - Transpose Headers.pbix from https://aka.ms/DataPwrBIPivot/downloads.

Exercise 4-7: Using M to Normalize Column Names

This exercise shows the most efficient method to apply a conversion table and normalize column names. In this exercise, you will use M and apply the conversion directly on the column names.

Recall that in Exercise 4-3, you applied a simple text manipulation on the column names by applying the M formula *Table.TransformColumnNames* with the relevant text manipulation as the second argument. For example, you used *Text.Proper* to capitalize each word.

In this final exercise on normalizing column names, you will create a custom function that will normalize the column names according to the rules in the conversion table.

Before you create the function, you need to understand one more transformation change that will simplify your code and speed up the lookup in the conversion table. Recall that earlier your conversion table was constructed as pairs of Source and Target values. Now you will transpose that table and promote the first row as headers. Thus, the new conversion table will include the Source values as headers, and the Target values as the first row, as shown in Figure 4-11.