Add Missing Columns to a Table

In Power-Query

STRUGGLING TO EXCEL

Let us do it the regular way...

- Just add a blank column called “Test Add” and inspect its code. The M-Code will read as follows:

```
Table.AddColumn(Source, "Test Add", each null)
```

- Now, modify the code, so it reads like this:

```
Table.HasColumns(Source, "Test Add")
```

- It will result in a ‘FALSE’ because that column does not exist.
- If you change the name of the column to something that already exists in your table, it will result in a ‘TRUE’

Let us put it together...

- Modify the code as follows:

```
if Table.HasColumns(Source, "Test Add") then Source else Table.AddColumn(Source, "Test Add", each null)
```

- Now this code, will add the column only if it does not already exist in the table.
- Okay great!
- But what if we must repeat this with 10 or a 100 more column names?
- Hmmm...

Let us scale it up!!!

- Insert a blank query and name it “Table_AddMissingColumns”.
- And paste the following code into the advanced editor:

```
let
    func = (table as table, columns as list, optional blanks
as any) as table =>
    let
        result = List.Accumulate(columns, table,
            (state, current) =>
                if
                    Table.HasColumns(state, current)
                then
                    state
                else
                    Table.AddColumn(state, current, each
blanks, type any)
            )
    in
        result
in
    func
```

Call this function...

- Now, you can call this function in any query.
- Right clicking the last step in your query and click on ‘Insert Step After’.
- Prefix the code of the new step with the name of the new function we added, and then suffix it with a list of columns you must have in your table.
- Here is an example:

```
Table_AddMissingColumns(#"Last Step",{"Col 1", "Col 2", "Col 3"})
```

- Now you will see that all these columns have been added to your table. If they existed already, their values will be retained safely.

The juicy details...

- You should learn about List.Accumulate from Microsoft Learn. Here is my explanation...
- The first two arguments are simple. You pass it a list, with an initial seed value for the "current" state.
- The last argument of this function, is tricky. It is a function itself. It starts off with the seed value for 'state' and the first value in the list (first argument) for 'current'.
- The logic that follows '=>' determines what you do with this framework.
- The return value of this function (third argument) becomes the 'state' for the next iteration, with the next value of the list becoming 'current'.
- Our function starts with the initial table, iteratively check whether a list of columns exists, and adds them when they are missing.

Stay curious...

Comment and share with a friend, if you found this helpful.

Follow me for more helpful tips.

STRUGGLING TO EXCEL