


1. In **Power Query**, the **Applied Steps** pane (on the right side of the editor) is basically a **history and control panel** for all the transformations you've made to your data.

Here's what it's for:

Purpose

1. **Shows the transformation sequence**
 - Every time you perform an action (filter rows, rename columns, change data types, etc.), Power Query records it as a step in this list.
 - Steps are executed in the order shown — top to bottom.
2. **Lets you edit or remove transformations**
 - You can click the  (gear) icon next to certain steps to edit them.
 - You can delete a step (X button) if it's no longer needed.
 - You can reorder steps if you want to change the sequence (though this may sometimes cause errors if later steps depend on earlier ones).
3. **Keeps your process reproducible**
 - Instead of applying changes manually every time, Power Query stores them here so they're automatically applied when the query refreshes.
4. **Makes debugging easier**
 - If you get an error, you can step through transformations one-by-one to see where it starts.



In

short:

The **Applied Steps** pane is like a *recipe card* for your data cleaning — it lists each “ingredient” (step) in order, so Power Query can rebuild the final dataset automatically each time you refresh.

2. In **Power Query**, removing duplicate rows is straightforward — it's basically telling Power Query, “*Keep only the first occurrence of each unique row or set of columns.*”

Here's how you can do it:

Method 1 – Remove duplicates from all columns

1. Select the **table** or make sure no specific column is highlighted.
2. Go to the **Home** tab.
3. Click **Remove Rows** → **Remove Duplicates**.
4. Power Query will compare **all columns** and keep only the first unique occurrence.

Method 2 – Remove duplicates based on specific columns

1. Select one or more columns by holding **Ctrl** (or **Shift**) and clicking their headers.
2. Go to **Home** → **Remove Rows** → **Remove Duplicates**.

3. Power Query will keep the first occurrence for each unique combination of the selected columns.

Extra Notes

- The step will appear in the **Applied Steps** pane as "Removed Duplicates".
- Behind the scenes, Power Query generates an M function like:
- `Table.Distinct(Source, {"Column1", "Column2"})`
 - If no columns are listed, it defaults to **all columns**.
- This operation is **case-sensitive** unless you transform text to lower/upper case beforehand.

 **Tip:** If you want to see what's being removed, you can **Group By** those columns first or sort by them before removing duplicates.

3. In **Power Query**, the **Filter** icon (the little ▼ dropdown arrow that appears in column headers) works a lot like Excel's filter — it lets you choose which rows to keep based on column values.

What it does

1. **Displays unique values in that column**
 - When you click the filter icon, Power Query scans the column and lists its distinct values.
2. **Lets you choose what to keep or remove**
 - You can check/uncheck values, use search, or apply conditions (e.g., "Text Filters", "Number Filters", "Date Filters").
3. **Applies a filter step**
 - When you apply a filter, Power Query creates a new **Applied Step** (usually called "Filtered Rows").
 - Only rows meeting your filter condition remain in the dataset.

Examples

- Keep only rows where Status = "Active".
- Remove all rows where Quantity is less than 10.
- Show only records from the last 30 days.


Behind the scenes

Filtering adds an **M function** like:

`Table.SelectRows(Source, each [Status] = "Active")`

or, for numeric conditions:

`Table.SelectRows(Source, each [Quantity] >= 10)`

 **Tip:** Unlike Excel, Power Query filtering is **static** unless you use dynamic conditions

(like filtering by `DateTime.LocalNow()`), so your filter will always follow the same rule when you refresh.

4. In Power Query (inside Power BI Desktop), clicking Close & Apply basically says:


“I’m done transforming this data — now load it into the Power BI model.”

What happens step-by-step

1. Applies your transformations
 - All steps in the Applied Steps pane are executed on the source data.
2. Loads the processed data into the Power BI data model.
 - The final query result is stored in Power BI’s in-memory storage (VertiPaq engine).
3. Closes the Power Query Editor window and returns you to the main Power BI interface.
4. Triggers a refresh for those queries so you can use the updated data in your visuals.

Key notes

- In Power BI Desktop, “Close & Apply” loads the data into the report model.
- In Excel Power Query, the equivalent button is Close & Load, which loads the data into a worksheet or the Excel Data Model.
- If you click Close (without “Apply”), your changes are discarded unless you’ve already applied them earlier.
- If your dataset is large, “Close & Apply” might take a while because it processes and loads all rows.

 **Tip:** If you want to save transformations but not load data yet, you can use Apply without closing, or Close without applying (and choose “Discard Changes” if you want to drop them).

10. In Power Query, there are several ways to handle null values in a column like Price, depending on your goal:

1 Remove rows with null prices

- When: If rows without a price are useless for your analysis.
- How (interface):
 1. Click the filter icon in the Price column.
 2. Uncheck (null).
- M code:
- `Table.SelectRows(PreviousStep, each [Price] <> null)`

2 Replace nulls with a default value

- When: If you want to fill missing prices with a placeholder (e.g., 0).
 - How (interface):
 1. Select the Price column.
 2. Transform → Replace Values →
Value to Find: null → *Replace With:* 0
 (Or Transform → Replace Errors if they're error values, not nulls.)
 - M code:
 - `Table.ReplaceValue(PreviousStep, null, 0, Replacer.ReplaceValue, {"Price"})`
-

3 Fill nulls with the previous or next value

- When: If missing prices should inherit from neighboring rows.
 - How (interface):
 1. Select Price.
 2. Transform → Fill → Down (or Up).
 - M code:
 - `Table.FillDown(PreviousStep, {"Price"})`
-

4 Replace nulls with a calculated value

- When: If you want to use, for example, the average price.
 - M code example:
 - let
 - `AvgPrice = List.Average(List.RemoveNulls(PreviousStep[Price])),`
 - `Filled = Table.ReplaceValue(PreviousStep, null, AvgPrice,`
`Replacer.ReplaceValue, {"Price"})`
 - in
 - `Filled`
-

💡 Tip: Always decide whether missing prices are *bad data* (remove them) or *incomplete data* (fill them) — that choice depends on your analysis needs.

Do you want me to make you an M code block that automatically replaces null prices with the average price of the column? That's a common approach.