🔄 1. Difference between “Merge” and “Append”

Feature	Merge	Append
Purpose	Combines **columns** from two tables (like SQL JOIN)	Combines **rows** from two or more tables (like UNION)
Example	Combine `Orders` and `Customers` on `CustID`	Stack `Orders_Jan` and `Orders_Feb` vertically
Result	One wider table (more columns)	One taller table (more rows)

👥 2. Split “Full Name” into “First Name” and “Last Name”

Select **Full Name → Home → Split Column → By Delimiter → Space**.
M-code:

```
= Table.SplitColumn(Source, "Full Name", Splitter.SplitTextByDelimiter(" ",
QuoteStyle.Csv), {"First Name", "Last Name"})
```

🔢 3. Purpose of “Pivot Columns”

Pivot Columns transforms unique values in one column into **new column headers**.
Example:

Product	Month	Quantity
Mouse	Jan	10
Mouse	Feb	15

👉 **Pivot on Month** → creates `Jan` and `Feb` columns with values from Quantity.

M-code:

```
= Table.Pivot(Source, List.Distinct(Source[Month]), "Month", "Quantity",
List.Sum)
```

↩️ 4. Undo a step in Power Query

In the **Applied Steps** pane:

- Click the **“X”** next to the unwanted step, or
- Press **Ctrl + Z** (undo shortcut).

💡 5. Purpose of “Reference” vs. “Duplicate”

Type	What It Does	When to Use
Duplicate	Makes a **copy** of the full query (independent)	When you need to modify separately
Reference	Creates a **linked version** of the query (depends on original)	When you want multiple transformations based on one source (without reloading data)

🔗 6. Merge Orders.csv and Customers.xlsx on CustID (inner join)

Home → Merge Queries → Choose Orders + Customers → Match CustID → Join Kind: Inner
M-code:

```
= Table.NestedJoin(Orders, {"CustID"}, Customers, {"CustID"}, "Customers", JoinKind.Inner)
```

Then expand the "Customers" column.

📊 7. Pivot Product column (show total Quantity per product)

Transform → Pivot Column

- Values column: `Quantity`
- Aggregate function: `Sum`

M-code:

```
= Table.Pivot(Source, List.Distinct(Source[Product]), "Product", "Quantity", List.Sum)
```

➕ 8. Append two tables (Orders_Jan + Orders_Feb)

Home → Append Queries → Two Tables → Select Orders_Jan & Orders_Feb
M-code:

```
= Table.Combine({Orders_Jan, Orders_Feb})
```

📩 9. Fill Down (replace nulls with previous Email)

Transform → Fill → Down
M-code:

```
= Table.FillDown(Source, {"Email"})
```

🌐 10. Extract domain from Email column

Add Column → Extract → Text After Delimiter → "@"
M-code:

```
= Table.AddColumn(Source, "Domain", each Text.AfterDelimiter([Email], "@"))
```

⚙️ 11. Dynamic merge using parameter (JoinType = "Inner")

Suppose you have a parameter `JoinType` (values like "Inner", "LeftOuter", etc.)

M-code:

```
= Table.NestedJoin(Orders, {"CustID"}, Customers, {"CustID"}, "Customers",
JoinKind.FromText(JoinType))
```

(Note: `JoinKind.FromText()` is a conceptual helper — in practice you'd use conditional logic:)

```
= Table.NestedJoin(Orders, {"CustID"}, Customers, {"CustID"}, "Customers",
    if JoinType = "Inner" then JoinKind.Inner
    else if JoinType = "LeftOuter" then JoinKind.LeftOuter
    else JoinKind.RightOuter)
```

🔁 12. Unpivot columns (Jan_Sales, Feb_Sales → Month, Sales)

Transform → Unpivot Columns (select all month columns).
M-code:

```
= Table.Unpivot(Source, {"Jan_Sales", "Feb_Sales"}, "Month", "Sales")
```

⚠️ 13. Handle errors in a custom column (try...otherwise)

For example, prevent division by zero:

```
= Table.AddColumn(Source, "Result", each try [Amount] / [Quantity] otherwise
null)
```

☑ If an error occurs, returns `null` (or any default value).

☎️ 14. Create a function to clean phone numbers

Removes spaces, dashes, parentheses:

```
CleanPhone = (phone as text) as text =>
    let
        clean = Text.Select(phone, {"0".."9"})
    in
        clean
```

You can then call:

```
= Table.AddColumn(Source, "CleanPhone", each CleanPhone([Phone]))
```

⚡ 15. Optimize a long query (10+ steps)

☑ Best Practices:

1. **Remove unnecessary columns early** (`Choose Columns`).
2. **Filter rows early** to minimize data volume.
3. **Combine related transformations** (e.g., multiple replaces into one `Table.ReplaceValue`).
4. **Avoid unnecessary "Changed Type" steps** (merge them).
5. **Use "Reference" queries** instead of repeating source connections.
6. **Disable load** for staging queries.
7. **Buffer** if reusing intermediate results:
8. `Buffered = Table.Buffer(PreviousStep)`
9. **Reduce custom column complexity** (calculate in DAX instead if possible).