1. What is the difference between "Merge" and "Append" in Power Query?

   Merge is similar to Join in SQL. Here is what it does:
   - Combines tables horizontally
   - Matches rows based on common key columns
   - Creates relationships between tables

   Example can be adding customer details to orders table

   Append is similar to Union in SQL. Here are the specifications of Append:

   - Combines tables vertically
   - Stack tables with similar column structures
   - No matching is required, just combines all rows

   Example can be combining January and February sales data

   Key differences: Merge adds width – columns. Append adds height – rows

2. How do you split a "Full Name" column into "First Name" and "Last Name"?

   Select Full Name column → transform tab → split column → by delimiter: *space* → at the leftmost delimiter → Ok → Rename resulting columns to 'First Name' and 'Last Name'

3. What is "Pivot Columns" used for?

   Pivot Columns are mainly used for transforming data from long form to wide formats. Here are the steps:
   - Takes unique values from a column and creates new columns from them
   - Spreads row data horizontally
   - Across multiple columns
   - Opposite of 'Unpivot Columns'

   Common usage: converting transactional data into summary tables or reports where categories become column headers.

4. How do you undo a step in Power Query?

   Applied Steps pane → right-click on the required step → Delete → confirm *or* just simply click x next to any step taken

5. What is the purpose of "Reference" vs. "Duplicate" in queries?

Reference and Duplicate have various roles:
Reference:

- Creates a pointer to the original query
- Shares the same data source and initial steps
- Changes to original query affect the reference
- Lighter on memory since there is no data duplication

Mainly used for building related queries from same source

Duplicate:

- Creates a complete copy of the query
- Independent of the original query
- Changes made to original don’t affect the duplicate
- Uses more memory since store separate data

Mainly used for creating similar but separate queries

Key differences: Reference shares data/steps, Duplicate creates an independent copy

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

   In Orders query:
   Home → Merge Queries → First table: Orders → Second table: Customers → clicking CustID in both tables to select the join key → Join kind: Inner → Ok

7. Pivot the Product column to show total Quantity per product.

   Select product column → transform tab → Pivot column → Values Column: Quantity → Advanced Options → Aggregate Value Function: Sum → Ok

8. Append two tables with identical columns (e.g., Orders_Jan.csv + Orders_Feb.csv).

   Home → Append queries → two tables → Primary table: Orders_Jan → Table to append: Orders_Feb → Ok

9. Use "Fill Down" to replace nulls in the Email column with the previous value.

   Home tab → Transform data → select Email column → Transform tab → Fill → Down

10. Extract the domain (e.g., "example.com") from the Email column.

    Home tab → Transform data → select Customer table → Add column → Custom column → new column name → custom column formula field: Text.AfterDelimiter([Email], “@”) → Ok

11. Write M-code to merge queries dynamically based on a parameter (e.g., JoinType = "Inner").

```
let
    JoinType = "Inner",
    LeftTable = Orders,
    RightTable = Customers,
    JoinKeys = {"CustID"},

    Orders = Table.FromRecords({
        [OrderID = 1001, CustID = 101, Product = "Laptop", Quantity = 1],
        [OrderID = 1002, CustID = 102, Product = "Mouse", Quantity = 3],
        [OrderID = 1003, CustID = 101, Product = "Keyboard", Quantity = 2],
        [OrderID = 1004, CustID = 103, Product = "Monitor", Quantity = 1]
    }),

    Customers = Table.FromRecords({
        [CustID = 101, Name = "Alice", Email = "alice@example.com"],
        [CustID = 102, Name = "Bob", Email = "bob@example.com"],
        [CustID = 103, Name = "Charlie", Email = "charlie@example.com"]
    }),

 MergedTable =
        if JoinType = "Inner" then
            Table.NestedJoin(LeftTable, JoinKeys, RightTable, JoinKeys,
"RightData", JoinKind.Inner)
        else if JoinType = "Left" then
            Table.NestedJoin(LeftTable, JoinKeys, RightTable, JoinKeys,
"RightData", JoinKind.LeftOuter)
        else if JoinType = "Right" then
            Table.NestedJoin(LeftTable, JoinKeys, RightTable, JoinKeys,
"RightData", JoinKind.RightOuter)
        else if JoinType = "Full" then
            Table.NestedJoin(LeftTable, JoinKeys, RightTable, JoinKeys,
"RightData", JoinKind.FullOuter)
```

```
        else if JoinType = "LeftAnti" then
            Table.NestedJoin(LeftTable, JoinKeys, RightTable, JoinKeys,
"RightData", JoinKind.LeftAnti)
        else if JoinType = "RightAnti" then
            Table.NestedJoin(LeftTable, JoinKeys, RightTable, JoinKeys,
"RightData", JoinKind.RightAnti)
        else
            error "Invalid JoinType. Use: Inner, Left, Right, Full, LeftAnti, or
RightAnti",

FinalTable =
        if List.Contains({"LeftAnti", "RightAnti"}, JoinType) then
            MergedTable
        else
            Table.ExpandTableColumn(MergedTable, "RightData",
Table.ColumnNames(RightTable))
in
    FinalTable

DynamicMerge = (leftTable as table, rightTable as table, joinKeys as list,
joinType as text) =>
let
    ValidJoinTypes = {"Inner", "Left", "Right", "Full", "LeftAnti",
"RightAnti"},
    ValidatedJoinType = if List.Contains(ValidJoinTypes, joinType) then
joinType else error "Invalid join type",

    JoinKindMap = [
        Inner = JoinKind.Inner,
        Left = JoinKind.LeftOuter,
        Right = JoinKind.RightOuter,
        Full = JoinKind.FullOuter,
        LeftAnti = JoinKind.LeftAnti,
        RightAnti = JoinKind.RightAnti
    ],

    SelectedJoinKind = Record.Field(JoinKindMap, ValidatedJoinType),

    MergedTable = Table.NestedJoin(leftTable, joinKeys, rightTable,
joinKeys, "RightData", SelectedJoinKind),

    FinalTable =
        if List.Contains({"LeftAnti", "RightAnti"}, ValidatedJoinType) then
            MergedTable
        else
            Table.ExpandTableColumn(MergedTable, "RightData",
Table.ColumnNames(rightTable))
in
    FinalTable,

ExampleUsage = DynamicMerge(Table1, Table2, {"CustID"}, "Inner")

AdvancedDynamicMerge = (
    leftTable as table,
    rightTable as table,
    joinKeys as list,
    joinType as text,
    optional columnsToExpand as list,
    optional newColumnNames as list
) =>
let
    BasicMerge = DynamicMerge(leftTable, rightTable, joinKeys, joinType),

    FinalResult =
        if List.Contains({"LeftAnti", "RightAnti"}, joinType) then
            BasicMerge
        else if columnsToExpand <> null then
            let
                ExpandedColumnNames = if newColumnNames <> null then
newColumnNames else columnsToExpand,
                ExpandedTable = Table.ExpandTableColumn(
                    Table.NestedJoin(leftTable, joinKeys, rightTable, joinKeys,
"RightData",
                        Record.Field([
                            Inner = JoinKind.Inner,
                            Left = JoinKind.LeftOuter,
                            Right = JoinKind.RightOuter,
                            Full = JoinKind.FullOuter
                        ], joinType)
                    ),
                    "RightData",
                    columnsToExpand,
                    ExpandedColumnNames
                )
            in
                ExpandedTable
        else
            BasicMerge
in
    FinalResult
```

12. Unpivot a table with columns like "Jan_Sales," "Feb_Sales" into a "Month" and "Sales" format.

    Transform data → selecting Product, Region columns → transform tab → unpivot columns → unpivot other columns → rename columns

13. Handle errors in a custom column (e.g., division by zero) using try...otherwise.

    Add column → Custom column → *try…otherwise* → Ok
    try…otherwise part can be

    ```
    = try [Sales] / [Quantity] otherwise null
    = try Text.Upper([Name]) otherwise "Invalid"
    = try ([Sales] / [Quantity]) * [Price] otherwise 0
    ```

14. Create a function in Power Query to clean phone numbers (e.g., remove dashes).

    Transform data → Add column → custom column → = Text.Replace(Text.Replace(Text.Replace([PhoneColumn], "-", ""), "(", ""), ")", "") || = Text.Select([PhoneColumn], {"0".."9"})
    Function:

    ```
    = let
        step1 = Text.Replace([PhoneColumn], "-", ""),
        step2 = Text.Replace(step1, "(", ""),
        step3 = Text.Replace(step2, ")", ""),
        step4 = Text.Replace(step3, " ", ""),
        step5 = Text.Replace(step4, ".", "")
      in step5
    ```

15. Optimize a query with 10+ steps—identify bottlenecks and simplify.

**Identify Bottlenecks**

- **Redundant Transformations**: Each Text.Replace step processes the entire column anew, leading to multiple passes over the data.
- **Scalability**: For a large dataset, these sequential replacements can slow performance due to repeated scanning.

## Simplify the Query

- **Combine Replacements**: Use a single Text.Replace with a list of characters to remove, reducing the number of steps to 1.
- **Efficient Function**: Leverage Text.Clean or a custom replacement logic to handle multiple characters in one pass.

  M code:

```
let
   step1 =
Text.Replace(Text.Replace(Text.Replace(Text.Replace(Text.Replace([PhoneColumn], "-", ""), "(", ""), ")", ""), " ", ""), ".", "")
in
   step1
```