Power BI Ecommerce Project – Step by step process

Table of contents

1. Project overview & folder structure

Goal: Build a retail analytics report showing Total Sales, Month-to-date, Last Month, Avg Selling Price (ASP), Top Products, Customer metrics and product/customer drillthroughs.

Create a local folder for the project and store files exactly like this:

```
PowerBI_Project/
  data/
    Customers.xlsx
    Products.xlsx
    Sales.csv
    Returns.csv
  assets/
    logo.png
    theme.json
  deliverables/
    Retail_Report.pbix
```

Why: Keeping files organized makes it easy to (a) point Power BI to the correct data, (b) reproduce the project on another machine, and (c) update data later.

2. Tools to install

- **Power BI Desktop** (latest): the main tool.
- Optional but recommended later: **Tabular Editor** (advanced measure/model export), **DAX Formatter** (online) for readability.

Why: Power BI Desktop does the full ETL → model → visualization workflow. Tabular Editor helps inspect/ export model objects if needed.

---

3. Prepare/Available raw files

Customers.xlsx (sheet `Customers`)

- `CustomerID` (unique numeric)
- `CustomerName`
- `Region`
- `Segment`
- `JoinDate`

Products.xlsx (sheet `Products`)

- `ProductID` (unique numeric)
- `ProductName`
- `Brand`
- `Category`
- `UnitPrice`

Sales.csv

- `OrderID`
- `OrderDate` (yyyy-mm-dd or dd/mm/yyyy — must be correctly parsed)
- `CustomerID`
- `ProductID`
- `Quantity` (integer)
- `UnitPrice` (numeric)
- `SalesAmount` (optional; if present, okay — otherwise we'll compute)

Returns.csv

- `ReturnID`
- `ReturnDate`
- `OrderID`
- `ProductID`
- `QuantityReturned`
- `UnitPrice` (optional)

Why: Clear, consistent column names make transformations and DAX simpler and less error-prone.

---

4. Open Power BI & import raw data — literal clicks and purpose

1. **Open Power BI Desktop** (double-click the app).
2. On the ribbon: **Home** → **Get data** → choose connector:
    - For Customers.xlsx:
        - Click **Excel** → Browse to `PowerBI_Project/data/Customers.xlsx` → click **Open**.
        - In the Navigator popup, click the `Customers` sheet (or table) → click **Load** if you want to load immediately OR **Transform Data** to clean before loading.
        - **Why choose Transform?** If you want to clean or change types before loading. For beginners, it's safer to Transform — we'll do it in the next step.
    - For Products.xlsx: same steps as Customers.
    - For Sales.csv:
        - Home → Get data → **Text/CSV** → select `Sales.csv` → click **Transform Data**.
        - **Why**: transactional data often needs cleaning and derived columns.
    - For Returns.csv: same as Sales.
3. After selecting **Transform Data**, you'll enter **Power Query Editor** for ETL (next major section).

Why: Each "Get data" retrieves a data source into Power BI. Choosing Transform Data opens the Query Editor so you can clean data before it's loaded into the model. This is important for correctness and performance.

---

5. Power Query Editor — full ETL walkthrough (step-by-step, per file)

Power Query is where you visually record steps; Power BI saves each step (M code). Every operation is reproducible.

Open Power Query Editor: after Get Data → Transform Data, you arrive here. Or in Power BI Desktop: **Home → Transform data**.

General ETL best practice order (apply in Power Query):

1. Remove unwanted columns
2. Change data types
3. Trim / clean text
4. Fix date formats
5. Create calculated columns (like SalesAmount) if source doesn't have them
6. Merge / Lookup only when necessary
7. Rename queries to friendly names
8. Disable query load for staging queries you don't want in the final model

We will process each source.

5.1 Customers — step by step (literal clicks + why + M snippet)

Steps (literal clicks)

1. In Power Query left pane, click `Customers`.
2. Right-click `Customers` → **Rename** → type `Customers`.
3. Remove unnecessary columns (example: `Notes`):
   - Click the `Notes` column header → Home → **Remove Columns**.
   - **Why:** Removing unnecessary columns reduces model size.
4. Change data types:
   - Click `CustomerID` → from the left of the column name you'll see an icon (ABC/123/calendar) → set to **Whole Number**.
   - Click `JoinDate` → set to **Date**.
   - **Why:** Correct data types are required for proper calculations and date/time intelligence.
5. Trim customer names:
   - Select `CustomerName` → Transform → Format → **Trim** (removes leading/trailing spaces).
   - **Why:** Spaces cause duplicates and filter mismatches.
6. Remove duplicate customers (if necessary):
   - Select `CustomerID` → Home → **Remove Rows → Remove Duplicates**.
   - **Why:** Ensures dimension uniqueness.

M code (Advanced Editor) — copy/paste example

Open Advanced Editor: **Home → Advanced Editor**, replace content with:

```
let
  Source =
Excel.Workbook(File.Contents("C:\\Path\\To\\PowerBI_Project\\data\\Customer
s.xlsx"), null, true),
  Customers_Sheet = Source{[Name="Customers"]}[Data],
  PromotedHeaders = Table.PromoteHeaders(Customers_Sheet,
[PromoteAllScalars=true]),
  ChangedTypes = Table.TransformColumnTypes(PromotedHeaders, {
      {"CustomerID", Int64.Type}, {"CustomerName", type text}, {"Region",
type text}, {"Segment", type text}, {"JoinDate", type date}
  }),
  Trimmed = Table.TransformColumns(ChangedTypes, {{"CustomerName",
Text.Trim, type text}}),
  RemovedDuplicates = Table.Distinct(Trimmed, {"CustomerID"})
in
  RemovedDuplicates
```

Replace `"C:\\Path\\To\\..."` with your actual path. Use forward slashes on Mac mapped environments if needed.

5.2 Products — step by step + M snippet

Steps

1. Select `Products` query.
2. Rename to `Products`.
3. Remove unused columns (e.g., `SupplierNotes`).
4. Ensure `ProductID` is Whole Number and `UnitPrice` is Decimal Number: change types.
5. If `Category` column contains hierarchical text (e.g., `Electronics>Audio>Headphones`), split it:
   - Select `Category` → Transform → Split Column → By Delimiter → choose `>` → split into `Category` and `SubCategory`.
   - **Why:** splitting allows easier grouping/aggregation.
6. Close & Apply later after all queries processed.

M snippet

```
let
  Source =
Excel.Workbook(File.Contents("C:\\Path\\To\\PowerBI_Project\\data\\Products
.xlsx"), null, true),
  Products_Sheet = Source{[Name="Products"]}[Data],
  PromotedHeaders = Table.PromoteHeaders(Products_Sheet,
[PromoteAllScalars=true]),
  ChangedTypes = Table.TransformColumnTypes(PromotedHeaders, {
      {"ProductID", Int64.Type}, {"ProductName", type text}, {"Brand", type
text}, {"Category", type text}, {"UnitPrice", type number}
  }),
  // Example split if Category contains a '>' delimiter
  SplitCategory = Table.SplitColumn(ChangedTypes, "Category",
Splitter.SplitTextByDelimiter(">", QuoteStyle.Csv),
{"Category","SubCategory"}),
  Trimmed = Table.TransformColumns(SplitCategory, {{"ProductName",
Text.Trim, type text}})
in
  Trimmed
```

5.3 Sales — the fact table (very important)

Why Sales is special: It's the large transactional table with the facts we will sum. We want to keep it as a fact table with foreign keys to dimension tables (ProductID, CustomerID) and add date parts for time-intelligence.

Steps (literal clicks)

1. In Power Query, click `Sales`.
2. Rename to `Sales`.
3. Remove any header garbage rows if present (Home → Remove Rows → Remove Top Rows).
4. Change data types:
   - `OrderDate` → **Date** (or Date/Time if time component exists).
   - `Quantity` → **Whole Number**
   - `UnitPrice` → **Decimal Number**
5. If `SalesAmount` not present, add it:
   - Add Column → Custom Column → Name: `SalesAmount` → Formula: `[Quantity] * [UnitPrice]` → OK.
   - **Why:** This is the numeric measure we will sum for Total Sales.
6. Add Date parts:
   - Select `OrderDate` → Add Column → Date → Year (creates `Year` column).
   - Add Column → Date → Month → Name of Month (or Month Number).
   - Add Column → Date → Start of Month (useful for grouping).
   - **Why:** Precomputing Year and Month simplifies visuals and reduces DAX complexity.
7. Remove unnecessary columns (e.g., internal flags).
8. Close & Apply when all queries done.

M snippet

```
let
  Source =
Csv.Document(File.Contents("C:\\Path\\To\\PowerBI_Project\\data\\Sales.csv"
), [Delimiter=",", Columns=6, Encoding=1252, QuoteStyle=QuoteStyle.None]),
  PromotedHeaders = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
  ChangedTypes = Table.TransformColumnTypes(PromotedHeaders, {
      {"OrderID", type text}, {"OrderDate", type date}, {"CustomerID",
Int64.Type}, {"ProductID", Int64.Type}, {"Quantity", Int64.Type},
{"UnitPrice", type number}
  }),
  AddedSalesAmount = Table.AddColumn(ChangedTypes, "SalesAmount", each
[Quantity] * [UnitPrice], type number),
  AddedYear = Table.AddColumn(AddedSalesAmount, "Year", each
Date.Year([OrderDate]), Int64.Type),
  AddedMonth = Table.AddColumn(AddedSalesAmount, "Month", each
Date.Month([OrderDate]), Int64.Type),
  AddedYearMonth = Table.AddColumn(AddedSalesAmount, "YearMonth", each
Date.ToText([OrderDate], "yyyy-MM"), type text)
in
  AddedYearMonth
```

5.4 Returns — process returns (optional: convert to negative sales)

Steps

1. Select `Returns` query.
2. Verify types: `ReturnDate` as Date, `QuantityReturned` as Whole Number.
3. Compute `ReturnsAmount`: Add Column → Custom Column → `[QuantityReturned] * [UnitPrice]`.
4. Optionally, create a `NegativeReturns` table or plan to subtract returns from Sales in DAX.

M snippet

```
let
  Source =
Csv.Document(File.Contents("C:\\Path\\To\\PowerBI_Project\\data\\Returns.cs
v"), [Delimiter=",", Columns=6, Encoding=1252,
QuoteStyle=QuoteStyle.None]),
  PromotedHeaders = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
  ChangedTypes = Table.TransformColumnTypes(PromotedHeaders, {
      {"ReturnID", type text}, {"ReturnDate", type date}, {"OrderID", type
text}, {"ProductID", Int64.Type}, {"QuantityReturned", Int64.Type},
{"UnitPrice", type number}
  }),
  AddedReturnAmount = Table.AddColumn(ChangedTypes, "ReturnAmount", each
[QuantityReturned] * [UnitPrice], type number)
in
  AddedReturnAmount
```

5.5 Final ETL housekeeping

- In Power Query, **right-click** queries that are intermediate helper queries (e.g., aggregated staging tables) and choose **Disable load** if you do not want them stored as tables in the model.
- Rename queries to sensible names if you haven't (e.g., `DimCustomers`, `DimProducts`, `FactSales`, `FactReturns`).

Why: Only tables you use in visuals should be loaded to minimize memory footprint.

5.6 Close & Apply

- When ready: click **Home → Close & Apply**.
- Power BI applies all transforms and loads the tables into the internal tabular model.

Why: This moves the prepared, clean data into the model where we create relationships and measures.

---

6 Data modelling — build relationships & date table (exact clicks + rationale)

Open **Model** view (left-side vertical icons → Model icon).

6.1 Create relationships (drag & drop)

1. Drag `Sales[CustomerID]` → drop onto `Customers[CustomerID]`.
   - If the relationship dialog opens, ensure: **Cardinality: Many to One (*:1)**, **Cross filter direction: Single**. Click **OK**.
2. Drag `Sales[ProductID]` → drop onto `Products[ProductID]`. Set cardinality to Many→One.
3. (If you created a Date table) Drag `Sales[OrderDate]` → `Date[Date]`. Mark Date table as date table: Select Date table → Modeling → Mark as date table → choose Date column → OK.

Why: Relationships allow filters on dimension tables (like Category or Customer) to filter the Fact (Sales) correctly. The star schema (one central fact, multiple dimensions) is the recommended pattern for performance and clarity.

6.2 Cardinality & filter direction explained (simple)

- **Many-to-One (*:1)**: Many rows in Sales for a single product in Products — typical for Fact→Dimension.
- **Cross filter direction = Single**: Filters flow from dimension → fact. Only use Both when necessary (it can create ambiguous filter contexts and reduce performance).

6.3 Create a Date table (if not present) — why and how

Why: DAX time intelligence functions like `SAMEPERIODLASTYEAR` require a proper date table that has contiguous dates (no gaps), and it should be marked as the model's date table.

Quick method: create in Power BI (Modeling → New table)

Literal steps:

1. In Power BI Desktop -> Modeling -> New table.
2. In the formula bar, paste:

```
Date =
ADDCOLUMNS(
  CALENDAR(DATE(2018,1,1), DATE(2026,12,31)),
  "Year", YEAR([Date]),
  "MonthNumber", MONTH([Date]),
  "MonthName", FORMAT([Date], "MMMM"),
  "YearMonth", FORMAT([Date], "yyyy-MM"),
  "Quarter", "Q" & FORMAT([Date], "Q")
)
```

3. Replace dates to cover your data range.
4. Then: Modeling → Mark as date table → select `Date[Date]`.

Why: This table drives accurate time-intelligence and consistent axes.

---

7 DAX measures — create, explain, annotate (all core measures)

Where to create: In Report or Model view, right-click a table (preferable a table named `Measures` or `FactSales`) → **New measure** → paste DAX.

I'll list each measure, explain it simply, and show the code.

7.1 Total Sales

What it does: Sums `SalesAmount` across filtered context (slicers/time selection apply).
Create: Right-click `FactSales` (or any table) → New measure → name and paste:

```
Total Sales = SUM( FactSales[SalesAmount] )
```

Why: Base metric used everywhere.

7.2 Total Quantity

```
Total Quantity = SUM( FactSales[Quantity] )
```

Why: Used for volume and to compute ASP.

7.3 Average Selling Price (ASP)

What: Sales per unit, safe division to avoid divide-by-zero errors.

```
ASP = DIVIDE( [Total Sales], [Total Quantity] )
```

Why: Gives price-level insights, avoids dividing by zero.

7.4 Total Returns

```
Total Returns = SUM( FactReturns[ReturnAmount] )
```

Why: For Net Sales.

7.5 Net Sales

```
Net Sales = [Total Sales] - [Total Returns]
```

Why: Real revenue after returns.

7.6 Sales MTD (Month To Date)

Pre-req: `Date` table must be marked as date table.

```
Sales MTD = TOTALMTD( [Total Sales], 'Date'[Date] )
```

Why: MTD helps track current month progress.

7.7 Last Month Sales

```
Last Month Sales = CALCULATE( [Total Sales], PREVIOUSMONTH('Date'[Date]) )
```

Why: Compare current to last month.

7.8 Sales LY (Last Year)

```
Sales LY = CALCULATE( [Total Sales], SAMEPERIODLASTYEAR('Date'[Date]) )
```

Why: Year-over-year comparisons.

7.9 % Change vs Last Year

```
Sales % vs LY = DIVIDE( [Total Sales] - [Sales LY], [Sales LY] )
```

Why: Express performance as percentage change.

7.10 Product Rank by Sales (Top N)

```
Product Rank by Sales =
RANKX( ALL( Products[ProductName] ), [Total Sales], , DESC, DENSE )
```

Why: Allows Top N visuals and rank filtering.

7.11 Distinct Customers (count of unique customers)

```
Total Customers = DISTINCTCOUNT( FactSales[CustomerID] )
```

Why: Tracks customer reach.

7.12 Example dynamic Top N filtered measure (Top 10 by slicer)

1. Create a parameter table (Home → Enter data) named `TopN` with single column `TopNValue` and values (5,10,20). Or create a numeric slicer.
2. Then:

```
Top N Sales =
VAR TopNValue = SELECTEDVALUE( TopN[TopNValue], 10 )
RETURN
CALCULATE(
  [Total Sales],
  TOPN( TopNValue, VALUES( Products[ProductName] ), [Total Sales], DESC )
)
```

Why: Enables user-controlled top N selection.

---

8 Build the report pages — step-by-step visuals and exact fields

I'll create three pages: `Dashboard`, `Product Detail`, `Customer Detail`. Follow exactly.

8.1 Page: Dashboard (layout, KPIs, trend, top lists)

Page setup

- New page: Click the + icon at bottom → double-click page name → rename to `Dashboard`.

Top row: KPI cards

1. Insert → **Card** → place at top-left.
2. Fields pane → drag measure `[Total Sales]` onto card (it becomes a single big number).
3. Duplicate card (select card, Ctrl+C, Ctrl+V) for `Total Customers`, `ASP`, `Net Sales`.
4. Format cards: Visual → Format pane → Turn on title → set title (e.g., "Total Sales") → Increase font sizes.

Middle row: Trend line

1. Visuals → **Line chart**.
2. Axis: `Date[Date]` or `Date[YearMonth]` (continuous)
   Values: `[Total Sales]`
   Legend (optional): `Products[Category]` to show categories over time.
3. Format: Data labels off, enable tooltips.

Right column: Top Products bar

1. Visuals → **Clustered bar chart**.
2. Axis = `Products[ProductName]`
   Values = `[Total Sales]`
3. Filter the visual for top 10: Visual filters pane → Add Products[ProductName] → choose Top N → enter 10 → By value `[Total Sales]` → Apply filter.
4. Sort by `[Total Sales]` descending.

Bottom: Matrix or Table for details

1. Visuals → **Matrix**
   Rows = `Products[Category]`, `Products[ProductName]`
   Values = `[Total Sales]`, `[Total Quantity]`
2. Format to show subtotals, set column widths.

Slicers (global filters)

- Insert → **Slicer** → drag `Date[Year]` → for month/year selection.
- Insert additional slicers for `Products[Brand]`, `Products[Category]`, `Customers[Region]`.

Why: This page gives a quick overview: KPIs for immediate health, trend to see time patterns, top products for prioritization, and detailed table.

8.2 Product Detail (drillthrough)

Create drillthrough page

1. New page → rename `Product Detail`.
2. In **Visualizations → Drillthrough** pane, drag `Products[ProductID]` (or `Products[ProductName]`) into Drillthrough fields.
   - This creates a target page for product-level insight.

Add visuals

- Line chart: Axis `Date[Date]`, Values `[Total Sales]` to show sales trend for selected product.
- Card: `[Total Sales]`, `[Total Returns]`, `ASP`.
- Table: Customer breakdown: `Customers[CustomerName]`, `[Total Sales]` (shows who bought this product).

Why: Drillthrough allows user to right-click a product on any page and open this page filtered to that product for deep analysis.

8.3 Customer Detail

- New page → rename `Customer Detail`.
- Add Drillthrough field: `Customers[CustomerID]`.
- Visuals: Card `[Total Sales]`, Line chart Sales by month, Table of Products purchased.
- Add a slicer for `Customers[Segment]` or `Customers[Region]`.

---

9 Interactivity: slicers, drilldown, bookmarks, tooltips (how & why)

9.1 Slicers

- Insert → Slicer, then drop a field such as `Date[Year]`.
- Format: Single select vs Multiselect (Format → Selection controls).
- Sync slicers across pages: View → Sync slicers → check pages to sync.

Why: Slicers provide user-driven filters.

9.2 Drilldown (hierarchy)

- Build a hierarchy: In Fields pane, right-click `Products[Category]` → create hierarchy and add `Products[SubCategory]` → add `Products[ProductName]`.
- Add hierarchy to a column chart axis. Use the drill controls (double down arrow) on the visual to drill into details.

Why: Drilldown lets users explore across levels without separate visuals.

9.3 Drillthrough (already set above)

- After adding `Products[ProductID]` to Drillthrough pane, users can right-click a product on any visual → Drillthrough → Product Detail.

Why: Focused, context-aware deep dive.

9.4 Bookmarks & Buttons (storytelling)

- Create a view you want to save (e.g., Top 10 Products).
- View → Bookmarks pane → Add → name it `Top10`.
- Insert → Button (Blank) → Action → Bookmark → choose `Top10`.
- Click the button to return to the saved state.

Why: Bookmarks store report states and power interactive navigation.

9.5 Report Tooltips (custom tooltips)

- New page → Format → Page information → Turn on Tooltip and set page size (e.g., Tooltip).
- Build visuals to show inside tooltip (e.g., mini chart).
- On target visual → Format → Tooltip → Report page → choose this tooltip page.

Why: Rich hover information without cluttering main page.

---

10 Formatting & themes — make it professional

1. Use a theme (View → Themes → Browse for themes → choose `assets/theme.json` if you have). Theme sets palette and fonts.
2. Format pane on each visual:
   - **Title**: Toggle On → set text, font size, alignment.
   - **Background**: Choose transparent or subtle fill.
   - **Data labels**: On for KPI visuals, off for cluttered charts.
   - **Legend & axis**: Place legend at top or right; format axis labels (rotate if long).
3. Use consistent fonts and sizes. Keep whitespace and alignment (View → Snap to grid → On).
4. Add logo: Insert → Image → upload `assets/logo.png` → place in header.

Why: Good formatting improves readability and trust.

---

11 Performance & optimization — practical rules and checks

Simple rules

- Remove unused columns before loading into model.
- Prefer measures to calculated columns when possible (measures compute on the fly and save memory).
- Use integers for keys (CustomerID, ProductID) instead of text.
- Avoid complex row-by-row iterators (like many `FILTER`-inside-`SUMX`) in measures; use aggregations.

Query folding

- When connecting to a database, check each Power Query step — if it is folded, Power Query sends work to the DB (faster). Right-click a step → **View Native Query** (if available).
- Avoid steps that break folding early: e.g., invoking a custom function that the DB cannot translate.

Performance Analyzer

- View → Performance analyzer → Start recording → Refresh visuals → see time per visual and DAX query durations.
- Focus optimization on the slowest visuals (top 20% causing 80% of slowness).

Why: Faster visuals improve user experience and reduce resource consumption on the service.

---

12 Publish, schedule refresh & security (RLS basics)

12.1 Publish

- File → Save As → `PowerBI_Project/deliverables/Retail_Report.pbix`.
- Home → Publish → Sign in → Choose workspace.
- Once published, go to **Power BI Service** → Workspace → Reports and Datasets.

12.2 Schedule refresh

- In workspace → Datasets → Click `...` → Settings → Data source credentials → set credentials.
- Gateway: if data is on-prem, configure On-premises data gateway.
- Schedule refresh: set frequency (daily/weekly) and time slots.

12.3 Row Level Security (RLS) basics

- Modeling → Manage roles → Create role (e.g., `RegionManager`).
- Add filter on `Customers` table: `[Region] = "West"` or better use a mapping table linking user email to region and filter: `Customers[Region] = LOOKUPVALUE(Access[Region], Access[UserEmail], USERPRINCIPALNAME())`.
- Test role: Modeling → View as roles → pick role.
- In Power BI Service: go to Dataset → Security → Add users to role.

Why: RLS restricts data visible to different users — crucial for compliance and privacy.

13 Validation checklist & troubleshooting common errors

Before publishing perform these checks:

- Totals check: `Total Sales` should equal sum of Sales CSV `SalesAmount` (or your external report).
- Row counts: `FactSales` row count expected.
- Date range: Date table covers min–max of Sales dates.
- Time-intelligence measures test: `Sales LY` should produce reasonable numbers for prior year.
- Drillthrough works: Right-click a product → Drillthrough.
- Filters: Slicers filter visuals correctly.
- Performance Analyzer: no visual exceeding acceptable time (e.g., 3–5 seconds).

Common problems & fixes

- Blank visuals: likely broken relationship or mismatched data types (e.g., ProductID text vs number).
- Wrong totals: check duplicated joins or incorrect relationship direction.
- Time intelligence returns blank: Date table not marked as date table.
- Slow visuals: convert heavy calculated columns to Power Query if static; add aggregations.

14 Appendix — Full example M scripts and DAX measures (copy-paste ready)

Full M script — Customers (example)

```
let
  Source =
Excel.Workbook(File.Contents("C:\\Path\\To\\PowerBI_Project\\data\\Customer
s.xlsx"), null, true),
  Customers_Sheet = Source{[Name="Customers"]}[Data],
  PromotedHeaders = Table.PromoteHeaders(Customers_Sheet,
[PromoteAllScalars=true]),
  ChangedTypes = Table.TransformColumnTypes(PromotedHeaders, {
      {"CustomerID", Int64.Type}, {"CustomerName", type text}, {"Region",
type text}, {"Segment", type text}, {"JoinDate", type date}
  }),
  Trimmed = Table.TransformColumns(ChangedTypes, {{"CustomerName",
Text.Trim, type text}}),
  RemovedDuplicates = Table.Distinct(Trimmed, {"CustomerID"})
in
  RemovedDuplicates
```

Full M script — Products

```
let
  Source =
Excel.Workbook(File.Contents("C:\\Path\\To\\PowerBI_Project\\data\\Products
.xlsx"), null, true),
  Products_Sheet = Source{[Name="Products"]}[Data],
  PromotedHeaders = Table.PromoteHeaders(Products_Sheet,
[PromoteAllScalars=true]),
  ChangedTypes = Table.TransformColumnTypes(PromotedHeaders, {
      {"ProductID", Int64.Type}, {"ProductName", type text}, {"Brand", type
text}, {"Category", type text}, {"UnitPrice", type number}
  }),
  SplitCategory = Table.SplitColumn(ChangedTypes, "Category",
Splitter.SplitTextByDelimiter(">", QuoteStyle.Csv),
{"Category","SubCategory"}),
  Trimmed = Table.TransformColumns(SplitCategory, {{"ProductName",
Text.Trim, type text}})
in
  Trimmed
```

Full M script — Sales

```
let
  Source =
Csv.Document(File.Contents("C:\\Path\\To\\PowerBI_Project\\data\\Sales.csv"
), [Delimiter=",", Columns=6, Encoding=1252, QuoteStyle=QuoteStyle.None]),
  PromotedHeaders = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
  ChangedTypes = Table.TransformColumnTypes(PromotedHeaders, {
      {"OrderID", type text}, {"OrderDate", type date}, {"CustomerID",
Int64.Type}, {"ProductID", Int64.Type}, {"Quantity", Int64.Type},
{"UnitPrice", type number}
  }),
  AddedSalesAmount = Table.AddColumn(ChangedTypes, "SalesAmount", each try
[Quantity] * [UnitPrice] otherwise null, type number),
  AddedYear = Table.AddColumn(AddedSalesAmount, "Year", each
Date.Year([OrderDate]), Int64.Type),
  AddedMonth = Table.AddColumn(AddedSalesAmount, "Month", each
Date.Month([OrderDate]), Int64.Type),
  AddedYearMonth = Table.AddColumn(AddedSalesAmount, "YearMonth", each
Date.ToText([OrderDate], "yyyy-MM"), type text)
in
  AddedYearMonth
```

Full M script — Returns

```
let
  Source =
Csv.Document(File.Contents("C:\\Path\\To\\PowerBI_Project\\data\\Returns.cs
v"), [Delimiter=",", Columns=6, Encoding=1252,
QuoteStyle=QuoteStyle.None]),
  PromotedHeaders = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
  ChangedTypes = Table.TransformColumnTypes(PromotedHeaders, {
      {"ReturnID", type text}, {"ReturnDate", type date}, {"OrderID", type
text}, {"ProductID", Int64.Type}, {"QuantityReturned", Int64.Type},
{"UnitPrice", type number}
  }),
  AddedReturnAmount = Table.AddColumn(ChangedTypes, "ReturnAmount", each
try [QuantityReturned] * [UnitPrice] otherwise null, type number)
in
  AddedReturnAmount
```

DAX measures — copy & paste

```
Total Sales = SUM( FactSales[SalesAmount] )

Total Quantity = SUM( FactSales[Quantity] )

ASP = DIVIDE( [Total Sales], [Total Quantity] )

Total Returns = SUM( FactReturns[ReturnAmount] )

Net Sales = [Total Sales] - [Total Returns]

Sales MTD = TOTALMTD( [Total Sales], 'Date'[Date] )

Last Month Sales = CALCULATE( [Total Sales], PREVIOUSMONTH( 'Date'[Date] )
)

Sales LY = CALCULATE( [Total Sales], SAMEPERIODLASTYEAR( 'Date'[Date] ) )

Sales % vs LY = DIVIDE( [Total Sales] - [Sales LY], [Sales LY] )

Product Rank by Sales = RANKX( ALL( Products[ProductName] ), [Total Sales],
, DESC, DENSE )

Total Customers = DISTINCTCOUNT( FactSales[CustomerID] )
```
