POWERBI - Tutorial

What I'll cover (short roadmap)

1. Power BI: Desktop vs Service vs Mobile — roles, when to use which, live example workflows.
2. Power Query & Data Transformation — connecting sources, Query Settings, M Query Editor, custom/conditional columns, hands-on steps.
3. Data Modelling & Relationships — relationship types, direction, cardinality, calendar (date) table, summarization, hands-on steps.
4. Advanced DAX & Measures — calculated columns vs measures, YTD/MTD, quick measures, performance tips, hands-on steps.

1) Power BI: Desktop vs Service vs Mobile — theory + real example + hands-on

Theory — roles & differences

- **Power BI Desktop**
  - Full report authoring tool (Windows app). You connect to sources, shape data in Power Query, define model and DAX measures, design visuals, and create pages.
  - Use when building or editing reports and data models.
- **Power BI Service (app.powerbi.com)**
  - Cloud platform for publishing, sharing, scheduling refreshes, dataflows, and app distribution. Contains workspace management, dashboards, dataset refresh, row-level security, and collaboration features.
  - Use when sharing reports, scheduling refreshes, building dashboards, or embedding.
- **Power BI Mobile (iOS/Android)**

- Optimized consumption of dashboards and reports on mobile device alerts, bookmarks and natural language Q&A.
- Use when stakeholders need to view KPIs on the go.

Real-world workflow example

- Data engineer provides a nightly CSV dump to an Azure Blob storage.
- Business analyst loads CSV in **Power BI Desktop**, transforms and models it, builds visuals and measures.
- Analyst publishes the report to **Power BI Service**, configures a gateway (if data is on-prem), sets scheduled refresh and permissions, and pins key visuals to a dashboard for executives.
- Executives view the dashboard and receive mobile push notifications on the **Power BI Mobile** app.

Hands-on (create & publish a new PBIX)

1. Open **Power BI Desktop**.
2. File → Import → Excel workbook → select **`powerbi_sample_data.xlsx`**.
3. In the Navigator dialog, check **`Sales`**, **`Customers`**, **`Products`**, **`Calendar`** and click **Transform Data** to go to Power Query.
4. After modelling & visuals (steps below), save: File → Save As → **`PowerBI_Tutorial.pbix`**.
5. Publish to Power BI Service: Home → Publish → choose workspace.
6. In Service, open the dataset → Settings → Schedule Refresh (if you later host file in OneDrive/SharePoint or setup Gateway).

---

2) Power Query & Data Transformation

Theory

Power Query is the ETL layer inside Power BI Desktop. It:

- Connects to many sources (Excel, CSV, SQL, Web, APIs, Azure, etc.).
- Applies stepwise transformations; each step generates M code.
- Is *declarative* — changes are recorded as query steps (applied steps).
- Provides the **Advanced Editor** with M language for custom transformations.
- Query Settings pane lists steps and lets you rename, reorder (careful), or delete steps.

Key capabilities:

- Data type changes, filter rows, remove duplicates, merge/append queries, fill-down, grouping, splitting columns.
- Add **Custom Column** (use M expressions) or **Conditional Column** (UI builder for if/then/else).
- Query folding: when source supports it (like SQL), transformations may be pushed to source for performance.

Real-time example

You receive **`Sales`** data where Product names are inconsistent (e.g., extra spaces, mixed case), some dates are text, and you need a **`SalesAmount`** column.

Hands-on steps (Power BI Desktop — using the provided Excel)

1. Home → Get Data → Excel → open **`powerbi_sample_data.xlsx`**.
2. In Navigator, click **Transform Data**.
3. In Power Query:
    - Rename query **`SalesRaw`** → then right-click → Duplicate → rename duplicated query **`Sales`**.
    - Ensure column **`Date`** is Date type: select column → Data Type → Date.
    - Normalize **`Product`** text: Transform → Format → Trim → Format → Capitalize Each Word (or Lowercase).
    - Remove duplicates if needed: Home → Remove Rows → Remove Duplicates (select relevant columns).
    - Create **`SalesAmount`** if not present: Add Column → Custom Column → formula:

      ```
      css

      [Quantity] * [UnitPrice]
      ```

    - Add a **Conditional Column** for **`SalesCategory`** (example business rule):
        - Add Column → Conditional Column:
            - If **`SalesAmount`** >= 200 then "Large"
            - Else if **`SalesAmount`** >= 50 then "Medium"
            - Else "Small"

- Rename steps in Query Settings so they read clearly: "Changed Type", Product", "Added SalesAmount", "Added SalesCategory".
- Use **Advanced Editor** to inspect M code. Example snippet for adding custom column:

```
m

#"Added SalesAmount" = Table.AddColumn(#"PreviousStep", "SalesAmount", each
[Quantity] * [UnitPrice], type number)
```

4. Click **Close & Apply**.

M Query Editor tips

- Every step returns a table; steps take previous step as input.
- Use **`Table.TransformColumns`**, **`Table.SelectRows`**, **`Table.AddColumn`**.
- For complex logic, you can create functions inside queries.
- To see the raw M: Home → Advanced Editor.

---

3) Data Modelling & Relationships

Theory

- **Tables** (queries) in the model are connected by relationships, often via keys (e.g., **`Sales[CustomerID]`** → **`Customers[CustomerID]`**).
- Cardinality: One-to-Many, Many-to-One, Many-to-Many.
- Cross-filter direction: Single or Both. Default best practice: keep Single direction where possible for performance and clarity; use Both only when required (and with caution).
- Star schema: Preferred modelling pattern. One central Fact table (Sales) and multiple Dimension tables (Customers, Products, Calendar).
- **Calendar (Date) table**: Crucial for time intelligence (YTD, MTD).
  - Should be a contiguous list of dates with columns for year, month, quarter, fiscal year, etc.
  - Mark it as a Date table in Power BI: Model view → select the date table → Table tools → Mark as date table → choose Date column.

Real-time example

Create relationships:

- **`Sales[CustomerID]`** → **`Customers[CustomerID]`** (Many-to-One)

- **`Sales[Product]`** → **`Products[Product]`** (Many-to-One)
- **`Sales[Date]`** → **`Calendar[Date]`** (Many-to-One)

Hands-on steps

1. In Model view (left side), drag **`CustomerID`** from **`Sales`** to **`CustomerID`** in **`Customers`**.
2. For **`Date`**: drag **`Date`** from **`Sales`** to **`Date`** in **`Calendar`**.
3. Check relationship properties: cardinality should be Many-to-One (Sales → Customers), cross filter direction usually Single (from Customers to Sales).
4. If **`Sales.Date`** contains time as well, create a date-only column in Power Query (Transform → Date → Date Only) before modelling, or in DAX: **`DateOnly = DATEVALUE([DateTimeColumn])`** (but prefer Power Query).
5. Mark **`Calendar`** as the official date table: Table Tools → Mark as date table → choose **`Date`** column.
6. Create a simple summarized table visual:
    - Insert a Matrix/Table visual: Rows = **`Calendar[Year]`**, Columns = **`Calendar[MonthName]`**, Values = **`SUM(Sales[SalesAmount])`**.
    - Use **`Summarize`** or **`Group By`** in Power Query for pre-aggregations if needed.

Summarization & Storage

- Aggregations: use aggregated tables or Power BI Aggregations feature for very large datasets.
- Summarize behavior: By default, numeric columns in visuals aggregate (sum, avg); you can change default summarization in modelling pane.

---

4) Advanced DAX & Measures

Theory: Calculated columns vs Measures

- **Calculated column**
    - Computed row-by-row when data is refreshed and stored in the model.
    - Use when result is needed for filtering, relationships, or row-level context (e.g., categorization).
    - Example: **`Sales[MarginClass] = IF([UnitPrice] - [StandardCost] > 20, "High", "Low")`**.
- **Measure**

- Calculated on the fly at query time using current filter context.
- Not stored per row; ideal for aggregations (SUM, AVERAGE, CALCULA
- Example: **`Total Sales = SUM(Sales[SalesAmount])`**.

Common time-intelligence measures

- **YTD** (Year-to-date)

```
dax


Total Sales YTD =
CALCULATE(
  [Total Sales],
  DATESYTD( 'Calendar'[Date] )
)
```

- **MTD** (Month-to-date)

```
dax


Total Sales MTD =
CALCULATE(
  [Total Sales],
  DATESMTD( 'Calendar'[Date] )
)
```

- **Previous Year**

```
dax


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

Quick Measures

- Quick Measures are template-based measures you can create via UI (Right-click table → New Quick Measure).
- Useful to learn the DAX produced; convert Quick Measures into custom DAX later.

Performance tips

- Prefer Measures over calculated columns for aggregates.
- Use variables (**`VAR`**) in DAX to avoid repeated calculation and improve readability.

- Reduce cardinality of columns used as keys when possible.
- Avoid **`FILTER`** over whole large tables unless necessary; use **`CALCULATETABLE`** logic.

Hands-on measures using the sample file

1. Open **Report view**.
2. In the Fields pane, right-click **`Sales`** → **New measure**:
   - Create **`Total Sales`**:

     ```
     dax

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

3. Create **`Total Qty`**:

   ```
   dax

   Total Qty = SUM(Sales[Quantity])
   ```

4. Create YTD:

   ```
   dax

   Total Sales YTD = CALCULATE([Total Sales], DATESYTD('Calendar'[Date]))
   ```

5. Create MTD:

   ```
   dax

   Total Sales MTD = CALCULATE([Total Sales], DATESMTD('Calendar'[Date]))
   ```

6. Create a measure for Average Order Value:

   ```
   dax

   AOV = DIVIDE([Total Sales], DISTINCTCOUNT(Sales[OrderID]), 0)
   ```

7. Build visuals:
   - Line chart: Axis = **`Calendar[Date]`**, Values = **`Total Sales YTD`**.
   - Card visuals: show **`Total Sales`** and **`AOV`**.
   - Matrix: Rows = **`Customers[Region]`**, Values = **`Total Sales`**, **`Total Qty`**.

Example: Compare YTD vs Last Year YTD % change

Create measure:

```
dax

Total Sales LY YTD = CALCULATE([Total Sales], SAMEPERIODLASTYEAR('Calendar'[Date]))
YTD % Change = DIVIDE([Total Sales] - [Total Sales LY YTD], [Total Sales LY YTD], 0)
```

Extra practical tips & checklist before saving your PBIX

- Validate relationships in Model view; avoid ambiguous many-to-many unless intentional.
- Mark the calendar as the date table.
- Rename measures with clear, consistent naming (prefix totals with **`Total`**).
- Document assumptions in a text box on the first report page (source, refresh cadence, currency).
- Save your PBIX: File → Save As → **`PowerBI_Tutorial.pbix`**.

Short checklist to perform now (copy/paste)

1. Download the sample file.
2. Open Power BI Desktop → Get Data → Excel → select **`powerbi_sample_data.xlsx`** → Transform Data.
3. In Power Query: set types, trim product text, add **`SalesAmount`**, add **`SalesCategory`** (conditional). Close & Apply.
4. Model view: create relationships (Sales→Customers, Sales→Products, Sales→Calendar). Mark **`Calendar`** as date table.
5. Report view: create measures (**`Total Sales`**, **`Total Sales YTD`**, **`AOV`**) and build visuals.
6. Save as **`PowerBI_Tutorial.pbix`** and publish if needed.

If you'd like, I can: