**Tasks to Perform in Power Query**

**🔸 A. Understanding & Identifying Data Types**

1. **Recognize and understand all 12 Power Query data types:**

**Text**

**True/False (Boolean)**

**Decimal Number**

**Fixed Decimal Number**

**Whole Number**

**Percentage**

**Date**

**Time**

**DateTime**

**DateTime/Timezone**

**Duration**

**Binary**

1. **Identify current data types:**

**Observe column icon (e.g., ABC, 123)**

**Use the Home > Data Type dropdown**

1. **Understand the ‘Any’ data type:**

**Why it is assigned by default**

**Why it should be avoided (can’t be aggregated or calculated)**

**🔸 B. Changing and Managing Data Types**

1. **Change data types manually:**

**Right-click on column → Change Type**

**Home > Data Type dropdown**

**Best practice:**

**Add changes as new steps in Applied Steps (not modify existing ones)**

**2) Handle errors in conversion:**

**e.g., Converting “A.W.” to Decimal → fails**

**Remove or correct faulty steps**

**3)Test conversions:**

**Whole numbers ↔ Decimal numbers**

**Date ↔ DateTime**

**DateTime ↔ Time**

**Numbers ↔ Text**

**Text → Decimal (watch for errors)**

**Understand default conversion behavior:**

**Date → DateTime = adds 12:00 AM**

**Binary type cannot be used in visuals**

**🔸 C. Replacing Values**

1. **Replace values in columns:**

**Select Gender column → Transform > Replace Values**

**Replace "M" with "Male" and "F" with "Female"**

**Understand case sensitivity:**

**"m" ≠ "M" → Fix mismatches**

**Edit steps if needed**

**Capitalize values consistently:**

**Transform > Format > Capitalize Each Word**

**Use Applied Steps effectively:**

**Monitor changes**

**Add steps for traceability**

**Remove faulty steps using ❌**

**Understand Save vs Apply:**

**Apply = save steps inside Power Query**

**Save = saves the entire .pbix file**

**🔸 D. Merging Columns**

1. **Merge two columns into a new column:**

**Select First Name + Last Name → Add Column > Merge**

**Separator = space → Column name = Customer Name**

**Merge and replace original columns:**

**Use Transform > Merge instead of Add Column**

**Try various separators:**

**Space " ", Comma ", ", Hyphen "-", or No separator**

**Understand performance impact:**

**New columns increase memory usage → prefer replacing originals when possible**

**Manage Applied Steps:**

**Check step: "Inserted Merged Column"**

**Delete/repeat merges to test options**

**📝 Additional Practice:**

**Task # Description**

**1 Merge City and State with a comma separator**

**2 Merge Product Name and Product Code with hyphen**

**3 Merge Department and Role into one column (replace originals)**

**4 Delete and redo merge steps**

**5 Merge 3 columns (e.g., Title, First Name, Last Name)**

**🔸 E. Extracting Text from Columns**

1. **Use Extract Function:**

**Select Email Address → Transform > Extract > Text Before Delimiter**

**Delimiter = "@" → Results in username**

1. **Replace or keep original column:**

**Right-click → Rename Email Address → Username**

1. **Rename columns clearly:**

**Full Address → Street**

**OrderID\_2024 → OrderID**

1. **Try various extract types:**

**Text Before "@"**

**Text After "@"**

1. **Text Between Delimiters (e.g., "Name [Dept]")**

**📝 Additional Practice:**

**Task # Scenario**

**1 Extract username from email using "@"**

**2 Extract product code from P-2025-XL using "-"**

**3 Extract region from Name\_Region using "\_"**

**4 Rename extracted Email Address to Username**

**5 Compare keeping original column vs replacing it**

**🔸 F. Removing Duplicates**

1. **Load a new table:**

**Use Recent Sources → Load DimCurrency table**

1. **Identify duplicate values:**

**Visually inspect for duplicates in CurrencyAlternateKey**

1. **Remove duplicates from a column:**

**Right-click column → Remove Duplicates**

**Verify outcome:**

**Ensure values like "USD" or "GBP" are now unique**

**🔸 G. Extracting Characters from a Column Using Delimiters (Extended)**

**Select the Email Address Column**

**Identify the column that contains values with structured patterns (e.g., email)**

**Use the Extract Function**

**Go to Transform > Extract > Text Before Delimiter**

**Enter the delimiter (e.g., "@") to extract everything before it**

**Replace or Create New Column**

**Choose whether to:**

**Replace the existing column**

**Create a new column**

**✅ In this scenario, you replace the column since only the username is required**

**Rename the Column Appropriately**

**Right-click on the transformed column → Rename**

**Change column name from Email Address to Username**

**Confirm Extraction Result**

**Verify that values now contain only the part before "@"**

**Example: "john.doe@email.com" → "john.doe"**

**Repeat Extraction Practice Using Other Delimiters**

**Try:**

**Extracting product ID from "P-2024-XL" using "-"**

**Extracting region from "Name\_Region" using "\_"**

**Compare Extraction Approaches**

**Use Extract as New Column vs. Replace Column**

**Practice when each method is more useful**

**🔸 H. Removing Duplicate Records from Tables**

1. **Load a New Table into Power Query**

* Use **Home > Recent Sources**
* Choose an already used source and select a different table (e.g., DimCurrency)

1. **Identify Duplicate Values in a Column**

* Scroll through a suspected column (e.g., CurrencyAlternateKey)
* Look for repeated entries (e.g., GBP, USD)

1. **Remove Duplicates**

* Select the column → Right-click → **Remove Duplicates**
* Observe the step Removed Duplicates in the **Applied Steps pane**

1. **Validate the Output**

* Confirm that duplicate values are removed and each item appears only once

1. **Understand the Context for Deduplication**

* Removing duplicates is critical in **dimension tables** like:
  + Currency table
  + Region table
  + Customer category

1. **Test Duplicate Removal in Other Tables**

If Any

### 🔸 I. Removing Unused or Unnecessary Columns

1. **Identify Columns Not Needed for Reporting**

* Review tables like DimCustomer or others to spot unused columns:
  + E.g., CustomerAlternateKey, Title, MiddleName, NameStyle, NumberChildrenAtHome, etc.

1. **Multi-select Columns for Deletion**

* Hold **Ctrl** and **click on column headers** (not inside cells)
* Select all irrelevant columns in one go

1. **Remove Unwanted Columns**

* Go to **Home tab** → Click **Remove Columns**
* Watch how the table becomes more focused and performance-ready

1. **Apply and Save**

* Use **File > Apply** to record the transformation
* Then **Save the Power BI file** to preserve changes

1. **Practice Exercise:**

* Load another large table (e.g., Sales, Orders)
* Remove any field that:
  + Is always null
  + Is never used in visuals
  + Exists for technical/debugging purposes (like IDs from old systems)

**Section J: Filtering Unnecessary Rows for Optimization**

Filtering unnecessary rows is a powerful way to optimize data models, improve performance, and ensure only relevant data is retained in Power BI reports.

### 🔵 ****Concepts to Understand****

* ✅ The goal of filtering is to **remove irrelevant or unused rows** that do not contribute to your visuals, calculations, or relationships.
* ✅ Filtering reduces the dataset size, improving load time and memory efficiency.
* ✅ You can apply filters based on **text values**, **numeric keys**, or **date ranges**.
* ✅ Advanced filter logic (e.g., combining multiple conditions) allows precise row-level control.
* ✅ Numeric field filtering is more reliable when working with relationships or foreign keys.

### 📚 ****Techniques and Best Practices****

#### ⚫ Filter Text Fields Using "Text Filters"

1. Go to the target column (e.g., CurrencyName).
2. Click the dropdown ↓ > **Text Filters** > **Equals**.
3. Switch to **Advanced** mode to apply multiple conditions.
4. Add multiple filters using OR, e.g.:
   * CurrencyName = "Australian Dollar"
   * OR CurrencyName = "Euro"
   * OR CurrencyName = "US Dollar"
   * OR CurrencyName = "Canadian Dollar"
   * OR CurrencyName = "GBP"
   * OR CurrencyName = "Deutschmark"

#### ⚫ Filter Numeric Fields Using "Number Filters"

1. Delete previous text filter step from Applied Steps.
2. Select the CurrencyKey column (numeric field).
3. Click dropdown ↓ > **Number Filters** > **Equals**.
4. Switch to **Advanced** mode.
5. Enter numeric keys for filtering, using OR logic:
   * CurrencyKey = 6
   * OR CurrencyKey = 19
   * OR CurrencyKey = 27
   * OR CurrencyKey = 34
   * OR CurrencyKey = 85
   * OR CurrencyKey = 100

#### ✅ Results

* Only the required rows for selected currencies remain.
* Table size is optimized, and only meaningful currencies are retained.

### ✍️ ****Practice Tasks****

| **Task #** | **Description** |
| --- | --- |
| 1 | Apply text filter on CurrencyName to keep 6 specific currencies. |
| 2 | Remove the text filter step from Applied Steps. |
| 3 | Apply numeric filter on CurrencyKey for the same currencies. |
| 4 | Compare results of text vs numeric filtering. |
| 5 | Load another table (e.g., Products) and practice filtering by product ID. |

### ✨ ****Tips for Mastery****

* Prefer filtering by **keys (IDs)** when joining tables or modeling relationships.
* Keep Power Query steps modular by clearly naming each filtering step.
* Validate filters by checking row counts before and after.
* Combine filters logically using OR and avoid AND unless rows have multiple matching conditions.

✅ **Section K: Calculating Age Using Dates in Power Query**

### 🎯 Goal:

Learn how to dynamically calculate and cleanly display a person's age based on a birthdate column using Power Query.

### ✅ Key Concepts to Understand

* Age is a dynamic field and cannot be hardcoded.
* Birthdate must be in **Date** or **Date/Time** data type.
* Age is calculated as the **difference between the birthdate and the current date**.
* Use **duration**, **conversion to years**, and **rounding** to calculate age correctly.
* Avoid unnecessary intermediate columns for better optimization.

### 🔹 Step-by-Step Transformation

#### 1. **Select Birthdate Column**

* Go to dimCustomer table.
* Select the BirthDate column.

#### 2. **Extract Date Information**

* Use Add Column > Date dropdown.
  + You can extract:
    - Year → gives year part of date
    - Month → gives month number
    - Quarter, Week, Day, etc.
* **Note**: Date functions only work on Date or DateTime columns. They will not work on text or non-date fields.

#### 3. **Calculate Age (Duration)**

* Select the BirthDate column.
* Go to Transform > Date > Age.
  + This gives a **duration** column.
  + Power Query will mark this with the **Duration** data type.

#### 4. **Convert Duration into Total Years**

* With the new column selected, go to:
  + Transform > Duration > Total Years
* Result will be decimal values like 49.69

#### 5. **Round Down Age**

* Still in Transform, go to:
  + Rounding > Round Down
  + Now you'll have age as whole numbers (e.g., 49 instead of 49.69)

#### 6. **Clean Up**

* If you used Add Column, delete intermediate columns like Duration or Decimal Age.
* If using Transform, you can simply overwrite existing columns.
* Optionally, **remove the BirthDate** column to save memory.

### 🛠 Best Practices

* Use **Transform** instead of Add Column to avoid storing unnecessary columns.
* Always **round age down** to reflect real-world grouping logic.
* Remove the **BirthDate** column if not required after age calculation to keep the dataset clean.

### 📌 Summary of Applied Steps

1. Select BirthDate
2. Apply Transform > Date > Age
3. Apply Transform > Duration > Total Years
4. Apply Transform > Rounding > Round Down
5. Remove BirthDate column (optional)

### 🧪 Practice Tasks

| **Task #** | **Practice Scenario** |
| --- | --- |
| 1 | Calculate age from a BirthDate field in your dataset |
| 2 | Extract just the year from a StartDate for employee records |
| 3 | Use Round Down on a decimal age column |
| 4 | Try using Add Column path and compare result with Transform |
| 5 | Remove intermediate columns to optimize your data model |

### ✅ Section L: Creating Conditional Columns Using IF Logic in Power Query

#### ✅ Key Concepts:

* **Purpose**: Group detailed data (e.g., age) into high-level categories (e.g., age buckets)
* **Tool**: Conditional Column tool under "Add Column" tab
* **Use Case**: Transform raw ages into meaningful groups (e.g., "30+", "40+", "50+")

#### 🔹 Step-by-Step: Create Age Buckets

1. **Select Column**:
   * Go to the Age column (created in earlier steps).
2. **Open Conditional Column Dialog**:
   * Navigate to **Add Column** tab > click **Conditional Column**
3. **Name the New Column**:
   * Set name to: Age Buckets
4. **Define Conditions (Initial Mistake Example)**:
   * If Age >= 30 → then 30+
   * If Age >= 40 → then 40+
   * If Age >= 50 → then 50+
   * Else → Others
5. **Understand the Error**:
   * Logic reads top-down, so all ages >= 30 match the first condition and never get checked against 40 or 50
6. **Fix the Logic - Use Descending Order**:
   * First clause: If Age >= 50 → 50+
   * Second clause: If Age >= 40 → 40+
   * Third clause: If Age >= 30 → 30+
   * Else → Others
7. **Apply and Review**:
   * Apply changes
   * Verify values by checking that age 49 appears under 40+, age 33 under 30+, age 55 under 50+, etc.

#### 🟢 Additional Notes:

* **Conditional Columns Can Use**:
  + Numbers (e.g., age, sales, income)
  + Text (e.g., gender, city, region)
* **Use Else Clause**:
  + Always define an Else clause to catch unexpected or out-of-range values
* **Editing Conditional Column**:
  + You can edit existing logic by clicking the gear ⚙️ icon in the Applied Steps pane

#### 🧪 Practice Tasks for Conditional Columns

| **Task #** | **Description** |
| --- | --- |
| 1 | Create age buckets with logic: 50+ / 40+ / 30+ / Others |
| 2 | Create spending buckets: <1000 = Low, 1000-5000 = Mid, >5000 = High |
| 3 | Use conditional column to classify regions (e.g., North, South, etc.) |
| 4 | Apply conditional logic to text column: Gender ("M" = Male, "F" = Female) |
| 5 | Practice removing and redoing conditional logic |

✅ **Outcome**: After this section, you'll know how to:

* Group numeric data into logical categories
* Fix misapplied conditional logic by reordering conditions
* Use conditional logic to create dynamic categorization for both numbers and text

**Section M: Unpivoting Columns for Better Tabular Data Structure**

✅ **Context and Challenge**

* In Power Query, we’ve primarily worked with transformations on individual columns.
* But some scenarios—like those in our **Internet Sales table**—require restructuring the whole table.
* The Internet Sales table does **not** promote the first row as headers by default.
* Column headers are generic (Column1, Column2…), while the actual headers (e.g., currency keys) are in the **first row**.

✅ **Problem Identification**

* The first row contains **currency IDs**, but Power Query fails to recognize this automatically due to numeric values.
* These IDs match with values in our **DimCurrency** table (e.g., 29, 6, 98, 1939, 100).
* Manually renaming these IDs to currency names is unsustainable.
* New currencies may appear over time, making this format difficult to maintain.
* **From a data modeling standpoint**, storing currency types across multiple columns violates tidy data principles.

✅ **Best Practice: Unpivoting**

* **Goal**: Turn multiple currency columns into two columns: one for CurrencyKey, one for SalesAmount.
* **Tool**: Use **Unpivot Columns** feature in Power Query.

✅ **Steps to Fix the Data Structure**

1. **Promote Headers**
   * Click the small icon on the top-left of the table preview.
   * Select **"Use First Row as Headers"**.
   * Power Query promotes the first row and automatically assigns data types.
2. **Multi-select Currency Columns to Unpivot**
   * Identify currency columns (e.g., 6, 29, 98, 1939, 100).
   * Hold Ctrl and Shift to select all desired columns.
3. **Apply Unpivot Transformation**
   * Right-click on one of the selected columns.
   * Choose **"Unpivot Only Selected Columns"**.
4. **Rename Columns**
   * The resulting columns will be:
     + **Attribute** → rename to CurrencyKey
     + **Value** → rename to SalesAmount

✅ **Benefits**

* The dataset is now in a **clean tabular format**.
* It allows for efficient joins (e.g., to the DimCurrency table).
* It avoids problems with expanding columns for new currencies.

### Section N: Merging Queries to Bring in Currency Names

✅ **Context and Objective**

* Previously, we transformed the Internet Sales table by unpivoting currency columns into two: CurrencyKey and SalesAmount.
* The CurrencyKey corresponds to entries in the DimCurrency table.
* Our goal: Replace CurrencyKey with actual **currency codes** and optionally **currency names**, eliminating the need for a separate dimension table.

✅ **Problem with the Current State**

* The Internet Sales table has many duplicated rows due to null values in unpivoted columns.
* A sale should only be recorded in one currency, but due to nulls, multiple rows are mistakenly created.

✅ **Step 1: Remove Invalid Rows**

* Go to SalesAmount column and filter **out null values**:
  + Click the dropdown > Uncheck (null) > Click OK.
* Result: Only valid sales records with actual currency values remain.

✅ **Step 2: Check Data Types Before Merge**

* Currency keys must match in **data type** to enable merging.
* Internet Sales: Change CurrencyKey column to **Whole Number**.
* DimCurrency: Ensure CurrencyKey is also of **Whole Number** type.

✅ **Step 3: Perform the Merge**

1. Go to **Home** ribbon > Click **Merge Queries** > Select **Merge Queries (not as new)**.
2. First Table: FactInternetSales (already selected)
3. Second Table: DimCurrency
4. Select CurrencyKey in both tables.
5. **Join Type**: Choose **Left Outer Join** (All from first, matching from second)
6. Click OK.

✅ **Step 4: Expand and Keep Only Needed Columns**

1. After merge, you'll see a new column (like DimCurrency as a table).
2. Click the **expand icon** (two arrows) on the new column.
3. Uncheck all columns except CurrencyAlternateKey (currency code).
4. Uncheck "Use original column name as prefix."
5. Click OK.

✅ **Step 5: Final Clean-Up**

* Rename CurrencyAlternateKey to CurrencyCode.
* Remove the original CurrencyKey column from Internet Sales.
* (Optional) If you want full currency names, go back to the expand icon and check CurrencyName.

✅ **Important Power Query Tip**

* Always make sure the **final transformation step is selected** before you click "Close & Apply."
* If an earlier step is selected, Power Query may only apply transformations **up to that step**, causing incorrect data to load into Power BI.

✅ **Outcome**

* The Internet Sales table now has:
  + A clean SalesAmount column
  + A readable CurrencyCode (and optionally CurrencyName)
  + No dependency on the external currency dimension table

**Section O: Using Parameters for Scalable Data Management**

✅ **What Are Parameters in Power BI?**

* Parameters act like **variables** in Power Query.
* They allow you to create **dynamic, reusable templates** for changing values (e.g., file paths, filter values).
* Extremely useful for **large organizations** operating in multiple regions or with multiple data sources.

✅ **Why Use Parameters?**

* Optimize data refresh and report performance.
* Secure data segregation by region, department, or user.
* Make reports scalable across departments or markets.
* Avoid hardcoding file paths in multiple tables.

✅ **Creating a Parameter in Power Query**

1. Go to **Home > Manage Parameters > New Parameter**.
2. Fill in:
   * **Name**: source
   * **Description** (optional): "Manages the data source path"
   * **Required**: ✅
   * **Type**: Text
   * **Suggested Values**: Any value
   * **Current Value**: Leave blank for now or paste the full file path, e.g., C:\Users\YourName\Documents\Data.xlsx

✅ **Link the Parameter to Tables**  
For each table (e.g., Customer, Currency, InternetSales):

1. Right-click the table > **Advanced Editor**.
2. Locate the File.Contents path in the Source step.
3. Replace the hardcoded path with the parameter name: source (no quotes).
4. Click **Done**.

**Alternative method:** In the Query Steps panel, click on Source, edit the file path by deleting the text and typing source.

✅ **Why This Matters**

* If the file is moved or changed (e.g., to a different regional folder), you only need to update **one parameter**, not every table.

✅ **Example Use Case**

* Germany regional manager gets their version of the report using the same dataset structure but filtered for German customers.
* Simply change the source parameter to the new file location.

✅ **Handling Missing Files**  
If the file is moved or renamed:

1. Power BI will show an error like "Could not find the file."
2. Go to the parameter value.
3. Update the path to the correct location.
4. All tables will update and refresh automatically.

✅ **Key Benefits**

* **Centralized file management**
* **Faster development and updates**
* **Avoid repetitive manual edits**
* **Supports enterprise-level scalability**

### Section P: Using Query Step Properties to Document Transformations

✅ **Why Query Properties Matter**

* Power Query provides a feature called **Step Properties**, which allows you to add **comments** to transformation steps.
* This becomes especially useful in **team environments** or when dealing with **complex queries**.
* Without comments, it can be difficult to understand the purpose behind each transformation—especially when there are 25+ steps.

✅ **Best Practices**

* Not every step needs a comment—some like Change Data Type or Rename Column are self-explanatory.
* Focus on **important transformations** like:
  + Unpivoting
  + Filtering
  + Merging tables
  + Expanding columns

✅ **How to Add Comments Using Step Properties**

1. **Locate the Step**
   * Hover over the step (e.g., Unpivoted Only Selected Columns).
2. **Open Step Properties**
   * Right-click on the step and choose **Properties**.
3. **Add a Description**
   * Example for unpivot:

"Unpivoting currency codes so that sales amount in each currency appears in a single column."

* + Example for filtering:

"Filtering out rows where SalesAmount is null to remove incomplete sales records."

* + Example for merging:

"Merging Internet Sales with Currency table using CurrencyKey to retrieve readable currency codes."

1. **Save the Comment**
   * Click **OK**. A small info icon ("i") will appear next to the step.
   * Hovering over the icon reveals your comment.

✅ **View All Comments via Advanced Editor**

* You can also open the **Advanced Editor** (right-click on the query name) to see M code with the commented steps.
* This view helps when you want to review or add comments **after completing** your transformations.
* Especially useful if you have 30–50 steps and don’t want to click through each individually.

✅ **Summary**

* Adding comments helps others (and your future self) understand the logic behind each transformation.
* It's a small effort that adds **long-term maintainability** and **collaboration value**.

**Section Q: Data Profiling in Power Query**

✅ **What is Data Profiling?**

* Introduced in June 2020, data profiling is an essential tool in Power Query.
* It provides an at-a-glance overview of your dataset.
* Helps identify:
  + Blank/null values
  + Errors
  + Value distributions
  + Statistical summaries (min, max, mean, etc.)

✅ **Why It Matters**

* Quickly understand data quality before building visuals.
* Detect anomalies or incorrect values in transactional tables.
* Often tested in Power BI certification exams (Day One topics).

✅ **Data Profiling Tools Available**

1. **Column Quality**
   * Shows percentages of **Valid**, **Error**, and **Empty** values per column.
   * Hover tooltips reveal absolute row counts.
   * Helps you spot gaps in data instantly.
2. **Column Distribution**
   * Displays number of **distinct** and **unique** values.
   * Useful for identifying repeated vs. one-time entries.
   * Example: 180 distinct dates, but only 4 are unique (appear only once).
3. **Column Profile**
   * Most detailed profiling option.
   * Displays summary statistics:
     + Count, Min, Max, Mean, Std Dev, etc.
   * Auto-adjusts based on data type (numeric, text, date).
   * Preview limited to top 1,000 rows (does not affect actual dataset).

✅ **How to Enable Profiling**

1. Go to the **View** tab in Power Query.
2. Toggle on:
   * **Column Quality**
   * **Column Distribution**
   * **Column Profile**

✅ **Best Practice: Use with Fact Tables**

* Use profiling on **FactInternetSales** or similar transactional tables.
* High-frequency updates make them prone to anomalies.

✅ **Extra Power: Export Column Profile as Table**  
You can extract column profiling info into a new query for reporting.

🔧 **Steps:**

1. Go to **Home > New Source > Blank Query**.
2. Rename the query to DataProfile.
3. Right-click > **Advanced Editor**.
4. Replace the default code with:
5. let
6. Source = Table.Profile(FactInternetSales)
7. in
8. Source
9. Click **Done**. You now have a table with profiling metrics:
   * ColumnName, Min, Max, Average, StdDev, Count, NullCount, etc.

🧠 **Note:**

* Fields like mean, standard deviation appear only for numeric columns.
* For non-applicable data types, null is shown.

✅ **Benefits of Profiling Table**

* Automate data audits.
* Embed in Power BI reports for live quality monitoring.
* Great for large datasets with frequent refreshes.

### Section R: Web Scraping with Power Query

✅ **What Is Web Scraping?**

* Web scraping extracts data/content from websites.
* In Power BI, it can be done directly from Power Query—no coding required!

✅ **Important Notes**

* Web scraping works best on websites with structured/tabular content (like Amazon or currency sites).
* News sites and heavily scripted pages may not load properly in Power Query's web preview.

✅ **Example 1: Scraping Data from Amazon**

1. Go to [Amazon.com](https://www.amazon.com), search for a term (e.g., "Power BI").
2. Copy the URL and go to **Power Query > New Source > Web**.
3. Paste the URL and click **OK**.
4. Choose between:
   * **HTML tables** (may include a lot of nulls)
   * **Suggested tables** (automatically structured options)
5. Pick the table that includes book title, reviews, format, etc.
6. You can rename columns like:
   * Title
   * Format
   * Review Count
7. If the structure is inconsistent, try **Add Table Using Examples**:
   * Type a sample value (e.g., title or price).
   * Power Query will auto-detect similar patterns.
   * Note: This preview uses an outdated browser and may not work for all sites.

✅ **Example 2: Importing Currency Exchange Rates**

1. Visit the [European Central Bank](https://www.ecb.europa.eu) exchange rates page.
2. Copy the URL and go to **Power Query > New Source > Web**.
3. Paste the URL, choose from the **HTML tables**.
4. Select the first table (which includes:
   * Currency code
   * Currency name
   * Exchange rate
5. Clean the table:
   * Remove unnecessary columns (like Chart).
   * Rename columns appropriately.
   * Rename table to ExchangeRates.

✅ **Merging Currency Rates with Existing Data**

* If your data model only includes 6 currencies out of 32:
  1. Go to **DimCurrency** table in Power Query.
  2. Use **Merge Queries** with ExchangeRates.
  3. Match on CurrencyAlternateKey and CurrencyCode.
  4. Use **Left Outer Join**.
  5. Expand only the ExchangeRate column.

✅ **Handling Missing Exchange Rates**

* Some rows like Euro or Deutschmark may have null for exchange rate.
* Replace null with 1:
  + Go to **Transform > Replace Values**.
  + Replace null with 1.

✅ **Result**

* You now have a live, refreshable currency feed.
* Each data refresh in Power BI will pull updated exchan

**Section S: Currency Standardization Using Exchange Rates**

✅ **Goal:**  
Convert all sales amounts in different currencies into a single base currency (e.g., Euro) to enable unified financial analysis.

✅ **Scenario Overview**

* Your sales data includes a SalesAmount and a CurrencyCode for each row.
* Sales are recorded in local currencies across different regions.
* You need to standardize them into one currency (Euro) using the exchange rates we brought in from the web in Section R.

✅ **Step-by-Step Transformation Process**

1. **Merge Exchange Rate into Sales Table**
   * Open the FactInternetSales table.
   * Go to **Home > Merge Queries**.
   * Select CurrencyCode from FactInternetSales.
   * Merge it with CurrencyAlternateKey in the DimCurrency table.
   * Use a **Left Outer Join**.
   * Click **OK**.
2. **Expand the Merged Table**
   * Click the **expand icon** in the new column.
   * Select only the ExchangeRate column.
   * Uncheck *Use original column name as prefix*.
   * Click **OK**.
3. **Convert Sales to Base Currency (Euro)**
   * Go to **Add Column > Standard > Divide**.
   * Choose SalesAmount as the base column.
   * Divide by the ExchangeRate column.
   * Result: A new column with Euro-standardized sales.
4. **Clean Up Columns**
   * Remove the original SalesAmount and ExchangeRate columns.
   * Rename the new column to SalesAmount\_Euro.

✅ **Why Add a New Column Instead of Overwriting?**

* Power Query’s **Transform > Standard > Divide** only allows dividing by a single constant value, not a column.
* The workaround is to use **Add Column > Standard > Divide** to perform column-to-column math.

✅ **Final Steps**

* Click **Close & Apply** to load the changes back into Power BI.
* The dataset now contains Euro-normalized sales ready for cross-region comparison.

✅ **What You’ve Learned**

* Merging currency rates with transaction tables.
* Performing dynamic currency conversion using Power Query.
* Preparing data for flexible currency-based reporting in Power BI.

**Section 3A: Introduction to Data Modeling in Power BI**

✅ **Why Data Modeling Is Needed**

* Raw data often comes in disorganized formats, making it unsuitable for reporting.
* In the **Power Query** chapter, we transformed various tables (FactInternetSales, DimCurrency, DimCustomer) to make them analysis-ready.
* Tasks included:
  + Pivoting multiple currency columns into a single one.
  + Filtering nulls, changing data types, merging tables.
  + Removing unnecessary columns.
* These steps helped shape raw transactional data into meaningful structure.

✅ **What Is Data Modeling?**  
Data modeling is the process of **transforming and shaping data** to support accurate, efficient, and insightful reporting.

Without proper modeling:

* Data retrieval is slow.
* Reports may be inaccurate.
* Resources are wasted with large, inefficient tables.

✅ **Dimension vs. Fact Tables**

* **Dimension Tables** (e.g., DimCustomer, DimCurrency)
  + Contain **descriptive attributes** (e.g., customer name, currency name).
  + Each row is uniquely identified by a **primary key**.
  + Do not grow rapidly.
* **Fact Tables** (e.g., FactInternetSales)
  + Contain **transactional data** (e.g., each row is a sale).
  + Contain **foreign keys** linking to dimension tables.
  + Grow continuously as more transactions occur.

Example:  
Instead of embedding customer names, addresses, and currency details in the fact table, we link these via keys—keeping FactInternetSales lean and performant.

✅ **Why Not Just Use One Big Table?**

* Technically possible but not recommended.
* Combining all details into FactInternetSales adds ~10 extra columns.
* This leads to:
  + Increased memory usage.
  + Slower dashboard performance.
  + Complex, hard-to-maintain data models.
* Efficient modeling ensures reports are **fast**, **responsive**, and **scalable**.

✅ **Keys in Data Modeling**

* **Primary Key** = Unique identifier in a dimension table.
* **Foreign Key** = Matching identifier in a fact table that connects to the dimension.

✅ **Power BI Views Overview**

1. **Report View**
   * Design visuals and reports using available fields.
2. **Data View**
   * Explore underlying data in tables (read-only Excel-like view).
3. **Model View**
   * Diagram view showing how tables are **related through relationships**.

**Section 3B: Understanding Relationships in Power BI**

✅ **Why Relationships Matter**

* When importing **multiple tables** into Power BI, you’ll likely want them to interact.
* **Relationships** allow separate tables to “talk” to each other—this enables filtering and aggregating data across tables.
* Power BI automatically **detects relationships** (based on primary and foreign keys) during data load.

✅ **Enabling Auto-Detect for Relationships**

* Auto-detect is enabled by default in Power BI.
* To check or modify:
  1. Go to **File > Options and Settings > Options**.
  2. Under **Data Load**, ensure:
     + ✅ *Auto-detect new relationships after data is loaded* is checked.
* It’s recommended to **keep this on**, especially for complex models with many tables.

✅ **How Relationships Work**

* Example: FactInternetSales and DimCustomer are related by CustomerKey.
  + In DimCustomer: CustomerKey is **unique** ➡️ One
  + In FactInternetSales: CustomerKey appears **multiple times** ➡️ Many
* This forms a **one-to-many (1:*N*) relationship**.

✅ **Cross Filter Direction**

* By default, Power BI sets **single-directional filtering** (from dimension ➡️ fact table).
* You can view this by double-clicking the relationship line.
  + **Single** means: Filtering Customer will affect InternetSales, but not vice versa.
* It's usually best to keep direction **single**, unless a **bi-directional** setup is absolutely needed.

✅ **How to Create Relationships**

1. **Drag & Drop**: Drag a key from one table to a matching key in another.
2. **Manage Relationships**:
   * Go to **Model View > Manage Relationships > New**.
   * Select source and target tables and keys.

✅ **What Happens Without a Relationship?**

* If no relationship exists:
  + Power BI **doesn’t know how to aggregate or filter data** correctly.
  + It will **duplicate total values** across rows.
* For example: A total sales figure of €23 million might be **repeated for each customer**, giving misleading results.

✅ **Active vs Inactive Relationships**

* Only **one active relationship** is allowed between two tables.
* If a second relationship is created:
  + It becomes **inactive** (shown as a **dotted line**).
  + Inactive relationships require **DAX functions** (like USERELATIONSHIP()) to use them.
* ⚠️ Be cautious with **many-to-many** relationships—they can cause **unpredictable behavior**.

✅ **Best Practices for Relationships**

* Prefer **One-to-Many** and **One-to-One** cardinality.
* Avoid **Many-to-Many**, unless absolutely necessary.
* Stick to **active relationships** unless advanced DAX modeling is involved.

**Section 3C: Implementing Row-Level Security (RLS)**

✅ **Why Secure Data Access?**

* In large organizations, different departments or regions should only access **relevant data**.
* Example: Sales managers in **Germany** shouldn’t see **U.S. sales data**.
* **Row-Level Security (RLS)** lets you restrict access to data based on user roles.

✅ **Scenario Example**

* A multinational company operates in:
  + United States
  + Canada
  + France
  + Germany
  + Australia
  + United Kingdom
* Each regional manager should only see **their own country’s** sales.

✅ **Step 1: Import Geographical Data**

* Use existing Excel data file.
* Load the DimSalesTerritory table using **Power Query**.
* Remove unwanted rows (e.g., where values are “Any”) using **Text Filters ≠ “Any”**.
* Apply and close the transformation.

✅ **Step 2: Verify Relationships**

* Ensure DimSalesTerritory is connected to FactInternetSales via SalesTerritoryKey.

✅ **Step 3: Create Roles**

1. Go to **Modeling > Manage Roles**.
2. Click **Create**.
3. Name the role (e.g., United States).
4. Select the DimSalesTerritory table.
5. Click the ellipsis ... next to it, and **Add Filter** on SalesTerritoryCountry.
6. Enter "United States" (include quotation marks).
7. Verify the DAX expression.

Repeat for each country:

* Canada
* France
* Germany
* Australia
* United Kingdom

Click **Save** once all roles are created.

✅ **Step 4: Test Roles in Power BI**

* Create a table visual with:
  + SalesTerritoryCountry from DimSalesTerritory
  + SalesAmountEuro from FactInternetSales
* Go to **Modeling > View as Roles**.
* Choose a role (e.g., Germany).
  + The data will filter to **only Germany’s sales**.
* Use **Stop Viewing** to return to default view.

✅ **What Happens Behind the Scenes?**

* In **Data View**, Power BI **filters all tables** according to the active role.
* Even foreign key tables like FactInternetSales show only the rows that match the selected region.
* This is not just visual—it applies to the underlying data model.

✅ **RLS Scope**

* Currently defined **within Power BI Desktop**.
* When you publish to **Power BI Service**, you’ll assign users or groups to these roles.
* This ensures users only access data **relevant to their role** when viewing shared reports.