# **📘 Power BI Course – Lecture 1 Notes**

## **💰 Salary Expectations (India)**

* **Fresher / beginner Power BI Developer or Data Analyst**: ₹4 – ₹8 LPA (average).
* Depends on company size:
  + **MNCs / Big Organizations** → Higher package.
  + **Startups / Small Firms** → Moderate package.

## **📊 What is Power BI?**

* A **Microsoft product** for business intelligence & reporting.
* Helps to:
  + Transform, structure, and prepare raw data.
  + Create **interactive visuals & dashboards**.
  + Tell **data-driven stories** to support business decisions.
* **Why popular?**
  + Easy integration with other Microsoft products (Excel, Azure, Teams, etc.).
  + Widely used in industry.
* Competitors: **Tableau, QlikView, Looker**.

## **📚 Course Contents**

1. **Data Visualization (with Power BI Desktop)**
   1. Learn interface & default visuals.
   2. First step to get comfortable with tool.
2. **Power Query Editor**
   1. "Heart of Power BI".
   2. Perform **data cleaning & transformation** (ETL).
   3. Make raw data ready for reporting.
3. **DAX (Data Analysis Expressions)**
   1. Functional language in Power BI.
   2. Used to create **measures & calculated columns**.
4. **Project 1: Sales Data Analysis**
   1. Covers **Data Modeling** (very important for interviews).
   2. Content designed with interviews in mind.
5. **Project 2: Insurance Data Analysis**
   1. Features:
      1. Row-level security (RLS).
      2. Report & dashboard sharing.
      3. Using custom + default visuals.
6. **Project 3: UPI Transactions Analysis**
7. **Miscellaneous Section**
   1. Scenario-based questions.
   2. Interview questions.
   3. Additional exercises on Power Query & DAX.
8. **Interview Preparation**
   1. Provided PDF with ~30 important interview questions.

## **👨‍💻 Day-to-Day Work of a Data Analyst**

1. **Daily Meetings**
   1. Scrum calls, team updates, discussing challenges.
2. **Data Work**
   1. Extracting data from multiple sources.
   2. Cleaning, structuring, and analyzing.
3. **Documentation**
   1. Record processes, reports, and workflows.
4. **Client Interactions**
   1. Collect feedback on dashboards/reports.
   2. Requires good **oral & written communication**.
5. **Collaboration with Teams**
   1. Work with IT, business, and other departments.
   2. Understand **KPIs & business logic**.
6. **Ad-hoc Reports**
   1. Prepare quick insights on request (manager/client).
7. **Responding to Queries**
   1. Handle emails/questions on reports & KPIs.
8. **Incorporating Feedback**
   1. Apply corrections from clients or team members.
9. **Report Sharing & Maintenance**
   1. Share reports carefully (data = company asset).
   2. Ensure correct access permissions.
10. **Reconciliation & Data Validation**
    1. Check Power BI report data vs source data.
    2. Ensure **accuracy & reliability** for decision-making.

## **📝 Quick Summary**

* Salary (fresher): 4–8 LPA.
* Power BI = Microsoft BI tool for data cleaning, visualization, and decision-making.
* Course covers: Visualization → Power Query → DAX → Projects (Sales, Insurance, UPI) → Interview prep.
* Analyst daily work: Meetings, data prep, documentation, client interactions, teamwork, ad-hoc reporting, validation, and report sharing.

# **📘 Power BI Course – Power Query Editor (Lecture 2 Notes)**

## **🔑 Key Idea**

* **Power Query Editor = Heart of Power BI**
  + Used for **data transformation & cleaning**.
  + Ensures data is in the right format before visualization.
  + Impacts **report accuracy** & **performance optimization**.

## **🖥️ Opening Power Query Editor**

1. Open **Power BI Desktop**.
2. Import data (e.g., Excel workbook → UPI Data sheet).
3. Two options when loading data:
   1. **Load** → directly load into model.
   2. **Transform Data** → open Power Query Editor for cleaning.

## **📊 Report View (before loading data)**

* **Panes**: Filters | Visualizations | Data.
* If no data loaded → shows “You haven’t loaded any data yet”.

## **🛠️ Why Power Query Editor Matters**

* **Two critical steps in reporting**:
  + Data Preparation (Power Query).
  + Data Modeling.
* Tasks in Power Query Editor:
  + Remove nulls.
  + Replace values.
  + Merge & append queries.
  + Change column data types.
  + Structure raw data for reporting.

## **📑 Data Types in Power Query**

* Each column shows a **data type icon** before its name.
* Example:
  + 📅 Date → Date column.
  + ⏰ Time → Time column.
  + 1.23 → Decimal number.
  + 123 → Whole number.
  + ABC → Text.
  + ABC/123 → Any (undefined).

✅ Always **check and correct data types** after importing data.

* Correct data types → accurate relationships + reports.
* Wrong data types → errors in visuals & calculations.

## **⚙️ Auto-Detect vs Manual Data Types**

* By default, **Power BI auto-detects column types**.
* If auto-detect is turned **OFF** (via File → Options → Data Load → Type Detection):
  + All columns load as **“Any” (ABC123)**.
  + First row may appear as normal data (not headers).

### **Fixing it manually:**

1. Use **“Use First Row as Headers”**.
2. Assign correct data types for each column:
   1. Date → Date
   2. Time → Time
   3. Transaction / Balance → Decimal Number
   4. Age / Customer ID → Whole Number
   5. City / Gender / Bank → Text

## **💡 Apply / Discard Changes in Power Query**

* **Apply** → save changes, stay in Power Query.
* **Close** → discard changes, return to report.
* **Close & Apply** → save changes and return to report view.

## **✅ Key Takeaways**

* Power Query Editor is essential for **data preparation**.
* Always **verify column data types** after importing.
* Use **auto-detect** for convenience but double-check accuracy.
* If disabled, you must **define headers & data types manually**.
* Data preparation directly impacts **report correctness & performance**.

# **📘 Power BI Notes – Data Profiling (Lecture 3)**

## **🔑 What is Data Profiling?**

* **Definition**: Process of analyzing the data to understand its quality, structure, and patterns.
* **Purpose**:
  + Helps in **understanding data better**.
  + Makes it easier to create **accurate reports & dashboards**.
  + Detects issues like **nulls, duplicates, or incorrect data types** early.

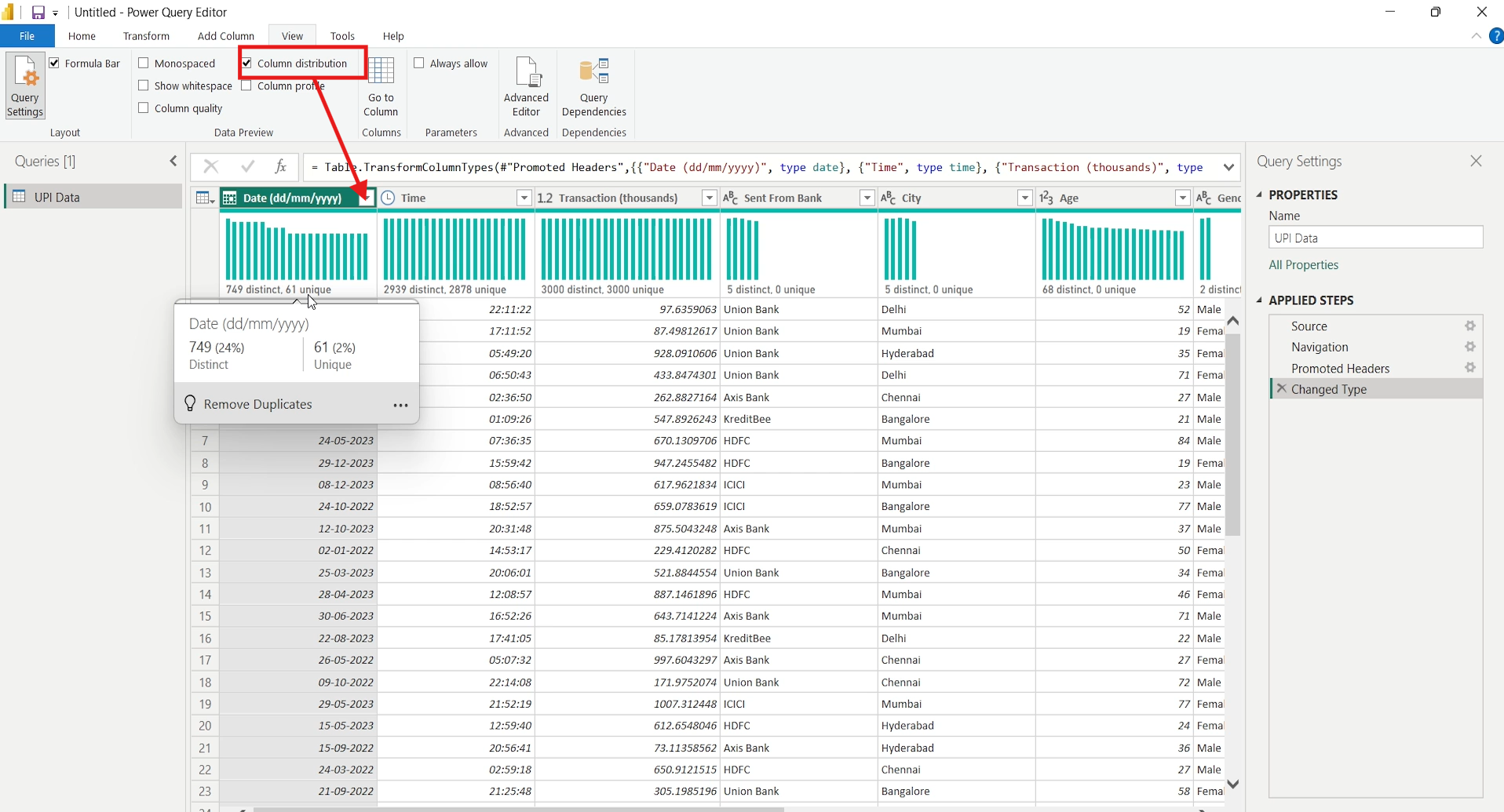
## **🖥️ Enabling Data Profiling in Power Query**

1. Open **Power Query Editor** → Transform Data.
2. Go to **View tab**.
3. Enable/disable options:
   1. **Column Distribution**
   2. **Column Profile**
   3. **Column Quality**

⚡ By default: profiling is done on **Top 1000 rows**.

👉 Change to **Entire Dataset** for full accuracy.

## **📊 Column Distribution**



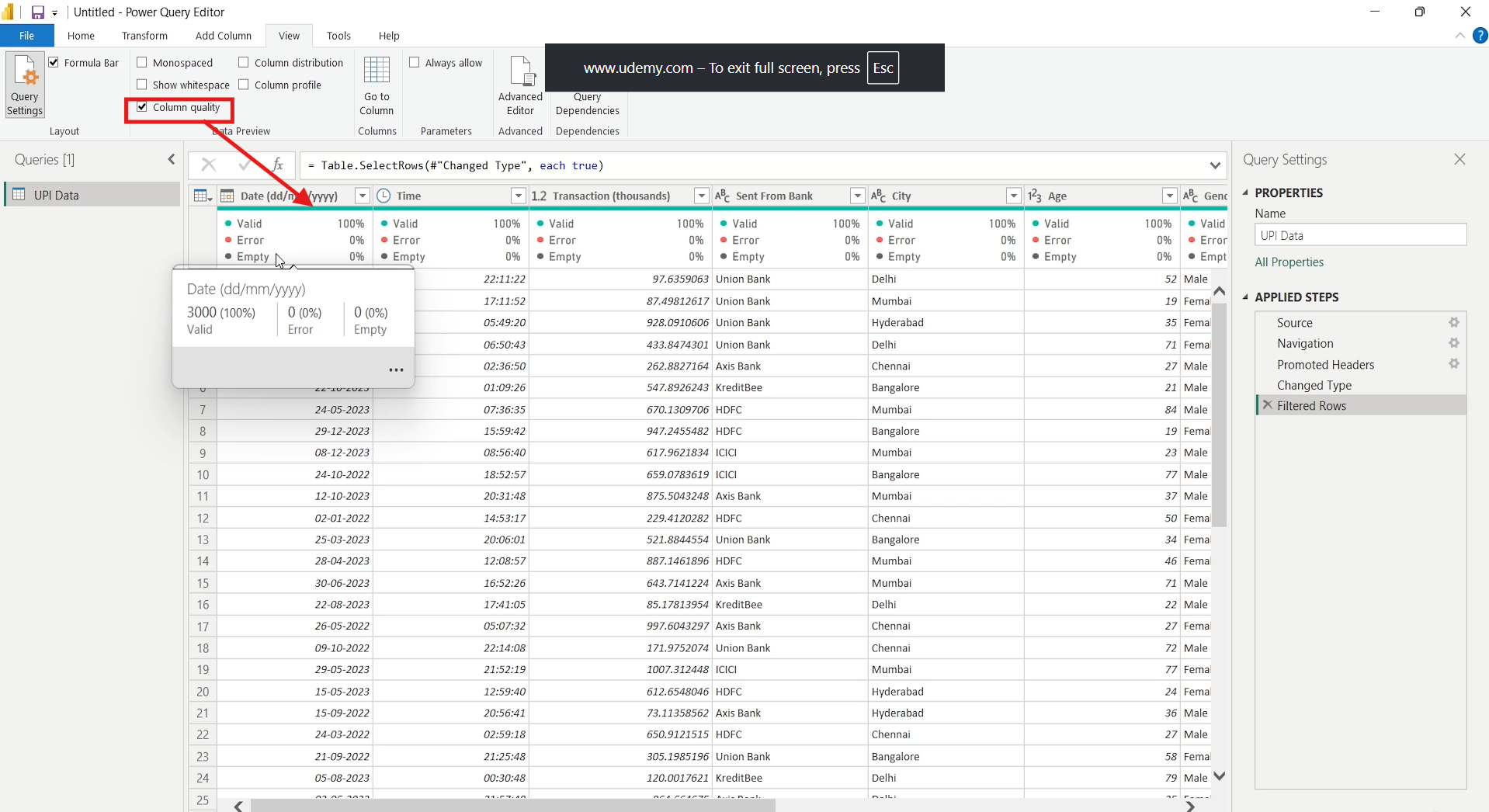
* Shows **distinct values** & **unique values** for each column.
* **Distinct values** → total number of different values in the column.
* **Unique values** → values that occur **only once** in the column.
* Example (Date column):
  + Distinct = 749 (different dates).
  + Unique = 61 (dates that appear only once).

💡 **Primary Key Rule**:

For a column to qualify as a **Primary Key**:

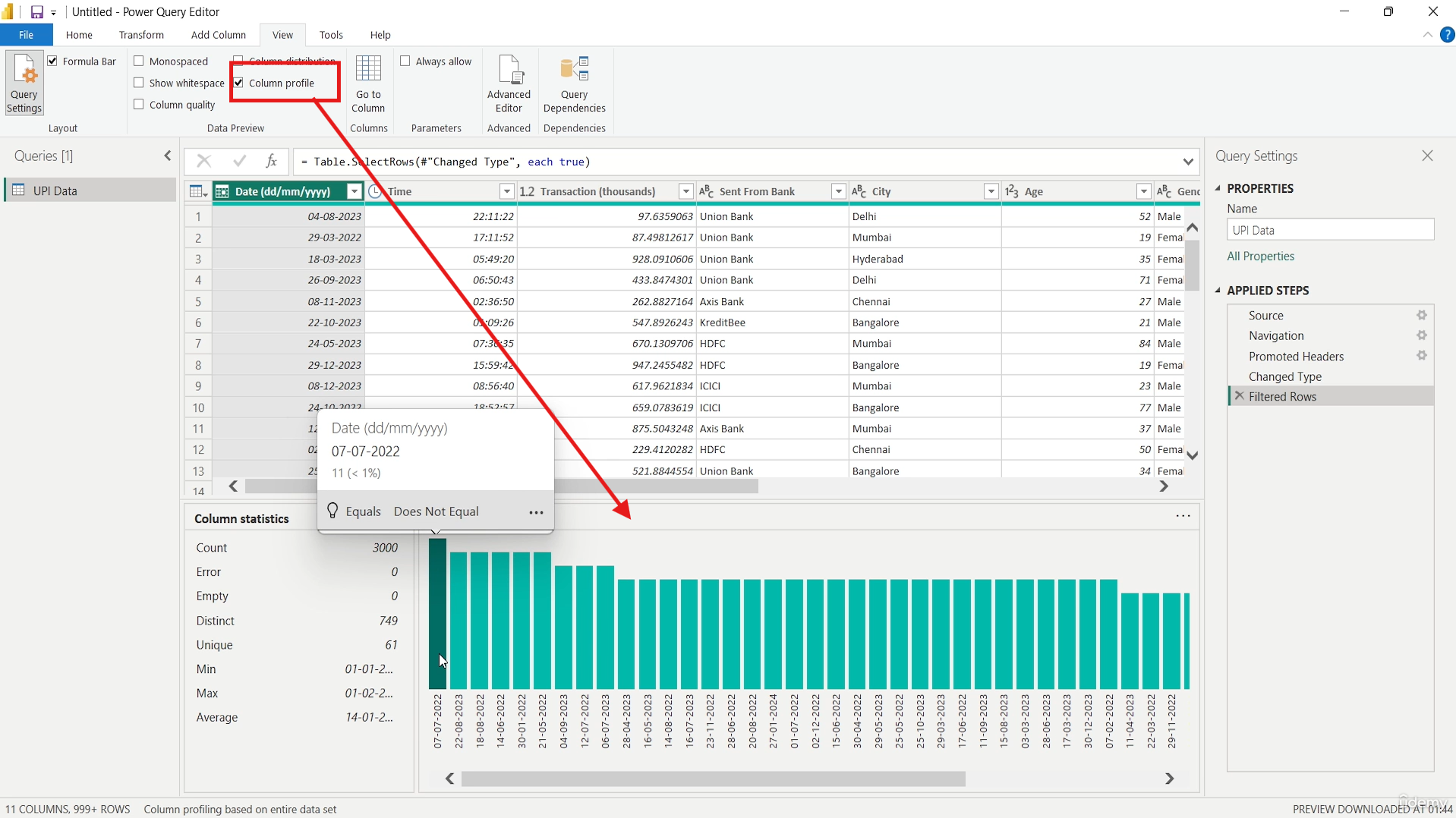
* No nulls.
* Distinct count = Unique count.

## **🛠️ Column Quality**



* Displays **data quality indicators**:
  + ✅ Valid values (%).
  + ❌ Errors (%).
  + ⚪ Empty values (%).
* Example (Date column):
  + Valid = 100%
  + Error = 0%
  + Empty = 0%

## **📑 Column Profile**



* Gives a **detailed summary** of a selected column:
  + **Value distribution** (bar chart).
  + **Column statistics** (depends on data type):

### **For Date columns:**

* Count, Distinct, Unique, Min Date, Max Date, Average Date.

### **For Numeric columns (decimal/whole number):**

* Count, Errors, Empty, Distinct, Unique.
* Min, Max, Average, Standard Deviation.
* Zero count, Not a Number (NaN), Even/Odd count (for integers).

### **For Text columns:**

* Distinct & Unique count.
* Min & Max (alphabetically sorted).
* Value distribution (frequency of categories).

## **✅ Why Data Profiling is Important?**

* Helps detect **duplicates, nulls, errors**.
* Assists in finding **primary keys**.
* Gives a **clearer picture** of dataset before modeling.
* Improves **accuracy** and **performance** of reports.

# **📘 Power BI Notes – Column Distribution Example (Lecture 4)**

## **🔑 Recap**

* Last lecture → **Data Profiling** (Distribution, Quality, Profile).
* This lecture → Focus on **Column Distribution** with an example.

## **🖥️ Example Setup**

* Table: **Column Distribution**
* Column: **Column1**
* Records: **11 total**

Values:

A, A, (Blank), B, B, B, C, D, E, E, F

## **📊 Distinct Count**

* **Definition**: Number of **different values** present in a column (ignores frequency).
* In this example:
  + A
  + Blank
  + B
  + C
  + D
  + E
  + F
* **Count = 7**

✅ Distinct = All unique categories, regardless of how many times they repeat.

## **📑 Unique Count**

* **Definition**: Number of values that occur **only once** in the entire column.
* In this example:
  + Blank (appears once)
  + C (appears once)
  + D (appears once)
  + F (appears once)
* **Count = 4**

❌ A (2 times), B (3 times), E (2 times) → Not included.

## **⚖️ Distinct vs Unique (Quick Difference)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Concept** | **Meaning** | **Example in this dataset** | **Count** |
| **Distinct** | Total number of different values (categories). | A, Blank, B, C, D, E, F | 7 |
| **Unique** | Values that occur only once. | Blank, C, D, F | 4 |

## **🛠️ Relevance to Primary Key**

* **Primary Key Requirements**:
  + No null values.
  + Distinct Count = Unique Count.

👉 If both match → column can uniquely identify each row → valid **Primary Key**.

## **✅ Key Takeaways**

* **Distinct = Different categories** (ignores frequency).
* **Unique = Values occurring only once**.
* Primary key must have **Distinct = Unique** and no nulls.
* Column Distribution is a quick way to check **data uniqueness** in Power Query

# **📘 Power BI Notes – Combining Queries (Appending) (Lecture 5)**

## **🔑 What is Combining Queries?**

* In Power Query, **Combine** can mean:
  + **Append Queries** → Stacks tables **row-wise** (like SQL UNION ALL).
  + **Merge Queries** → Joins tables **column-wise** (like SQL JOIN).

👉 This lecture focuses on **Appending Queries**.

## **🖥️ Where to Find?**

* In **Power Query Editor → Home tab → Combine group**.
* Options:
  + **Append Queries** → modifies the selected table.
  + **Append Queries as New** → creates a new query (✅ safer, keeps base tables unchanged).

## **📊 Example – Student Details**

### **Table 1: Student\_Details\_1**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Phone** |
| 1 | A | X |
| 2 | B | Y |
| 3 | C | Z |

### **Table 2: Student\_Details\_2**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Phone** | **Address** |
| 9 | P | M | PM9 |
| 12 | Q | N | QN12 |

## **📑 Appending Result**

After appending, Power Query creates a **new table with all columns from both tables**:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Phone** | **Address** |
| 1 | A | X | Null |
| 2 | B | Y | Null |
| 3 | C | Z | Null |
| 9 | P | M | PM9 |
| 12 | Q | N | QN12 |

⚡ Note:

* Columns missing in a table → filled with **Null**.
* Final schema = **Union of all columns**.

## **✅ Key Takeaways**

* **Append Queries** is used to **stack multiple tables** into one.
* If column names differ → Power BI **creates new columns**.
* If a column is missing in one table → its values = **Null**.
* Safer option: use **Append Queries as New** (keeps source intact).
* Works similar to SQL **UNION ALL** (keeps duplicates).

# **📘 Power BI Notes – Merge Queries (Inner Join) (Lecture 6)**

## **🔹 What is Merging Queries?**

* **Merging** = combining two tables based on a **common column** (similar to SQL joins).
* Different from **Appending** (which stacks rows like SQL UNION).
* Done in **Power Query Editor** under **Home → Combine → Merge Queries**.

## **🔹 Example Setup**

1. **All Student Details table**

Columns:

* 1. ID
  2. Name
  3. Phone Number
  4. Address

→ Contains 7 students (IDs: 1, 2, 3, 4, 5, 9, 12).

→ Some addresses are NULL.

1. **Student Marks table**

Columns:

* 1. ID
  2. Subject
  3. Marks

→ Contains marks for **only 4 students** (IDs: 1, 2, 3, 4).

→ Not all subjects available for each student.

## **🔹 How Merge Works**

* To merge, Power BI asks:
  + **First Table (Left Table)** → e.g., All Student Details.
  + **Second Table (Right Table)** → e.g., Student Marks.
  + **Matching Column** → ID (common in both).
  + **Join Type** → e.g., Inner Join.

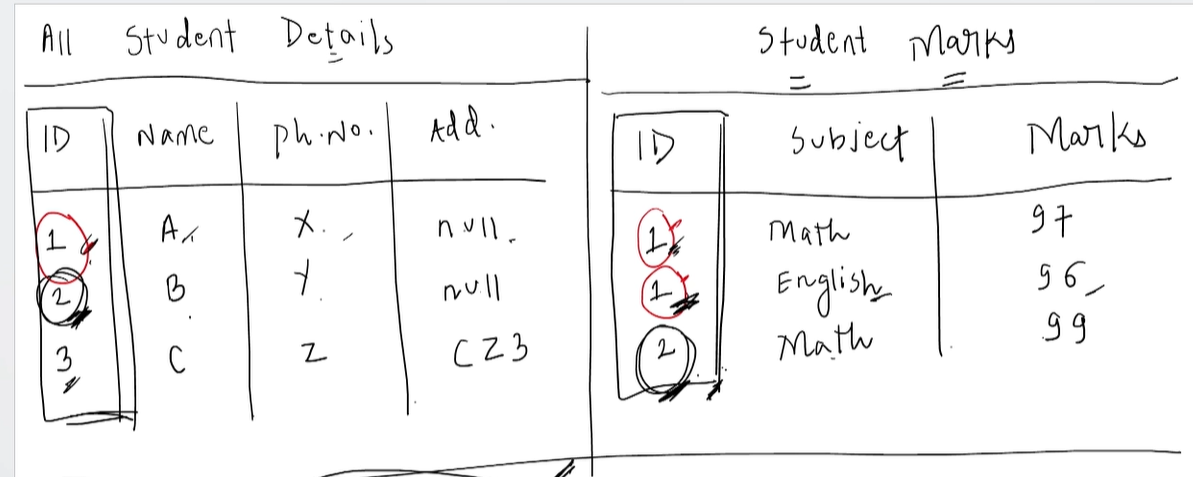
## **🔹 Inner Join in Power BI**

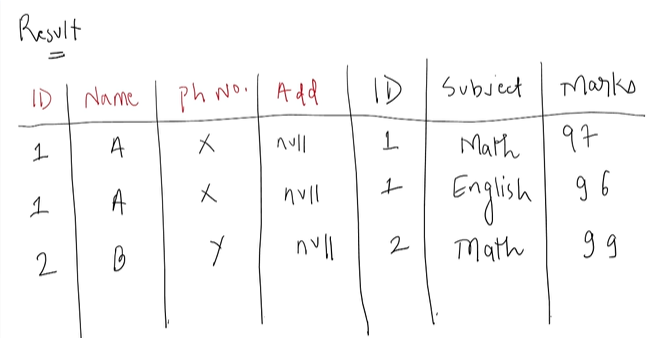
* **Definition:** Keeps only rows where the key column (ID) is present in **both tables**.
* Extra columns from second table can be **expanded** after merge.

**Steps in Power BI:**

1. Select **All Student Details** → Merge Queries as New.
2. Choose **Student Marks** as second table.
3. Select **ID** column from both.
4. Choose **Inner Join (Only matching rows)**.
5. Expand second table → select **Subject & Marks** only.

## **🔹 Example Result (Inner Join Output)**





✅ Only IDs **1, 2, 3, 4** appear (common in both).

❌ IDs **5, 9, 12** are excluded (not in Student Marks).

# 📘 Notes: Left Outer Join in Power BI (Power Query)

## 🔹 What is Left Outer Join?

* Combines two tables based on a common column.
* **Rule:**
  + Takes **all records from the Left Table** (first table).
  + Adds **matching records from the Right Table** (second table).
  + If no match is found in the Right Table → NULL values are added.

## 🔹 Tables Used in Example

### Left Table → **All Student Details**

| **ID** | **Name** | **Phone** | **Address** |
| --- | --- | --- | --- |
| 1 | A | X | NULL |
| 2 | B | Y | NULL |
| 3 | C | Z | CZ3 |

### Right Table → **Student Marks New**

| **ID** | **Subject** | **Marks** |
| --- | --- | --- |
| 1 | English | 79 |
| 1 | Math | 97 |
| 2 | English | 89 |
| 6 | Math | 78 |
| 15 | English | 88 |

## 🔹 Applying Left Outer Join on Column ID

### Step 1: Matches

* **ID 1**
  + Left: (1, A, X, NULL)
  + Right has two matches: (1, English, 79), (1, Math, 97)
  + → Two records in result.
* **ID 2**
  + Left: (2, B, Y, NULL)
  + Right: (2, English, 89)
  + → One record in result.
* **ID 3**
  + Left: (3, C, Z, CZ3)
  + Right: ❌ No match
  + → Record included with NULL for Subject & Marks.

### Step 2: Non-Matching IDs

* Right Table IDs **6** and **15** don’t exist in Left Table.
* ❌ Not included in result (because join is Left Outer).

## 🔹 Final Result (Left Outer Join Output)

| **ID** | **Name** | **Phone** | **Address** | **Subject** | **Marks** |
| --- | --- | --- | --- | --- | --- |
| 1 | A | X | NULL | English | 79 |
| 1 | A | X | NULL | Math | 97 |
| 2 | B | Y | NULL | English | 89 |
| 3 | C | Z | CZ3 | NULL | NULL |

## 🔹 Key Takeaways

* **Inner Join** → Only matching records.
* **Left Outer Join** → All records from Left Table + matching from Right Table.
* Non-matching Left records → stay in result with NULL values for Right table columns.
* Non-matching Right records → ❌ ignored.
* In Power Query:
  + Use **Merge Queries as New** → Choose Left Outer join type → Expand columns as needed.

# 📘 Notes: Right Outer Join in Power BI (Power Query)

## 🔹 What is Right Outer Join?

* Combines two tables based on a common column.
* **Rule:**
  + Takes **all records from the Right Table** (second table).
  + Adds **matching records from the Left Table** (first table).
  + If no match is found in the Left Table → NULL values are added.

## 🔹 Tables Used in Example

### Left Table → **All Student Details**

| **ID** | **Name** | **Phone** | **Address** |
| --- | --- | --- | --- |
| 1 | A | X | NULL |
| 2 | B | Y | NULL |
| 3 | C | Z | CZ3 |

### Right Table → **Student Marks New**

| **ID** | **Subject** | **Marks** |
| --- | --- | --- |
| 1 | English | 79 |
| 1 | Math | 97 |
| 2 | English | 89 |
| 6 | Math | 78 |
| 15 | English | 88 |

## 🔹 Applying Right Outer Join on Column ID

### Step 1: Matches

* **ID 1**
  + Right table has (1, English, 79), (1, Math, 97).
  + Matches with Left table (1, A, X, NULL).
  + → Two records in result.
* **ID 2**
  + Right table (2, English, 89).
  + Matches with Left table (2, B, Y, NULL).
  + → One record in result.

### Step 2: Non-Matching IDs (Right side only)

* **ID 6**
  + Present only in Right Table.
  + → (6, Math, 78) with **NULLs from Left Table**.
* **ID 15**
  + Present only in Right Table.
  + → (15, English, 88) with **NULLs from Left Table**.

## 🔹 Final Result (Right Outer Join Output)

| **ID** | **Name** | **Phone** | **Address** | **Subject** | **Marks** |
| --- | --- | --- | --- | --- | --- |
| 1 | A | X | NULL | English | 79 |
| 1 | A | X | NULL | Math | 97 |
| 2 | B | Y | NULL | English | 89 |
| 6 | NULL | NULL | NULL | Math | 78 |
| 15 | NULL | NULL | NULL | English | 88 |

## 🔹 Key Takeaways

* **Left Outer Join** → All rows from Left Table + matching from Right Table.
* **Right Outer Join** → All rows from Right Table + matching from Left Table.
* Non-matching Left records → ❌ ignored.
* Non-matching Right records → stay in result with NULL values for Left table columns.
* In Power Query:
  + Use **Merge Queries as New** → Choose Right Outer join type → Expand columns.

# 📘 Notes: Left Anti Join & Right Anti Join (Power BI)

## 🔹 1. Left Anti Join

* **Definition:** Returns only rows that exist **in the Left Table but NOT in the Right Table**.
* Matching IDs between tables are excluded.
* Columns from the Right Table appear as **NULL** in output.

### Example

**Left Table → All Student Details**

| **ID** | **Name** | **Phone** | **Address** |
| --- | --- | --- | --- |
| 1 | A | X | NULL |
| 2 | B | Y | NULL |
| 3 | C | Z | CZ3 |
| 5 | E | V | EV5 |
| 9 | F | W | FW9 |
| 12 | G | T | GT12 |

**Right Table → Student Marks New**

| **ID** | **Subject** | **Marks** |
| --- | --- | --- |
| 1 | English | 79 |
| 1 | Math | 97 |
| 2 | English | 89 |
| 3 | Science | 80 |
| 6 | Math | 78 |
| 15 | English | 88 |

**Left Anti Join Output**

| **ID** | **Name** | **Phone** | **Address** | **Subject** | **Marks** |
| --- | --- | --- | --- | --- | --- |
| 5 | E | V | EV5 | NULL | NULL |
| 9 | F | W | FW9 | NULL | NULL |
| 12 | G | T | GT12 | NULL | NULL |

✅ Explanation:  
Only **5, 9, 12** are unique to Left Table.  
Other IDs (1, 2, 3) exist in Right Table → excluded.

## 🔹 2. Right Anti Join

* **Definition:** Returns only rows that exist **in the Right Table but NOT in the Left Table**.
* Matching IDs between tables are excluded.
* Columns from the Left Table appear as **NULL** in output.

### Example

**Right Anti Join Output**

| **ID** | **Name** | **Phone** | **Address** | **Subject** | **Marks** |
| --- | --- | --- | --- | --- | --- |
| 6 | NULL | NULL | NULL | Math | 78 |
| 15 | NULL | NULL | NULL | English | 88 |

**✅ Explanation:**  
Only **6, 15** are unique to Right Table.  
Other IDs (1, 2, 3) exist in Left Table → excluded.

## 🔹 Key Takeaways

* **Left Anti Join → Unique rows from Left Table**.
* **Right Anti Join → Unique rows from Right Table**.
* Used for **finding mismatched records** or **data cleaning**.
* In Power BI Power Query:
  + Go to **Home → Merge Queries as New**.
  + Select join type **Left Anti (rows only in first)** or **Right Anti (rows only in second)**.

# 📘 Power BI – Full Outer Join Notes

## 🔹 What is Full Outer Join?

* Combines results of **Inner Join + Left Anti Join + Right Anti Join**.
* Includes:
  1. All matching rows (like Inner Join).
  2. All records present **only in Left Table**.
  3. All records present **only in Right Table**.

## 🔹 Example Tables

**Left Table: All Student Details**  
(ID, Name, Phone Number, Address)

**Right Table: Student Marks**  
(ID, Subject, Marks)

**Join Key → ID**

## 🔹 Full Outer Join Output Structure

Columns in output:

* From Left Table: ID, Name, Phone, Address
* From Right Table: ID, Subject, Marks

## 🔹 Step-by-Step Logic

1. **Inner Join Part**
   * Records with **matching IDs** in both tables appear.
   * If multiple matches exist, duplicates are generated.
   * Example:
     + ID = 1 in Left matches ID = 1 twice in Right → two rows in output.
     + ID = 2 in Left matches ID = 2 twice in Right → two rows in output.
2. **Left Table Only**
   * Records with IDs **only in Left Table** appear.
   * Example:
     + ID = 9, 12 in Left but missing in Right → output includes them with Right-side columns as NULL.
3. **Right Table Only**
   * Records with IDs **only in Right Table** appear.
   * Example:
     + ID = 6, 15 in Right but missing in Left → output includes them with Left-side columns as NULL.

## 🔹 Implementation in Power BI (Power Query Editor)

1. Click **Transform Data → Power Query Editor**.
2. Select **All Student Details (Left Table)**.
3. Go to **Merge Queries → Merge Queries as New**.
4. Select ID column from both tables.
5. Choose **Full Outer Join (All rows from both)**.
6. Expand Right Table columns (ID, Subject, Marks) → Click OK.
7. Output contains:
   * Matches (like Inner Join).
   * Left-only rows (Right-side = NULL).
   * Right-only rows (Left-side = NULL).
8. Rename result as **Full Outer Join**.

## 🔹 Example Output (Simplified)

| **ID** | **Name** | **Phone** | **Address** | **Subject** | **Marks** |
| --- | --- | --- | --- | --- | --- |
| 1 | A | X | … | English | 79 |
| 1 | A | X | … | Math | 97 |
| 2 | B | Y | … | English | 89 |
| 3 | C | Z | … | NULL | NULL |
| 6 | NULL | NULL | NULL | Math | 78 |
| 9 | D | P | … | NULL | NULL |
| 12 | E | Q | … | NULL | NULL |
| 15 | NULL | NULL | NULL | English | 88 |

## 🔹 Summary

* **Inner Join part** + Left-only + Right-only = **Full Outer Join**.
* Ensures **no data loss** from either table.
* Useful when you want to see a **complete dataset** across both tables.

# 📘 Power BI – Group By in Power Query Editor

| **Concept** | **Details** | **Example** |
| --- | --- | --- |
| **Where to find** | Home tab → Group By OR Transform tab → Group By | – |
| **Data Example** | UPI transactions dataset with columns: Sent From Bank, Transaction Amount, City, Age, Gender, Received Bank, Balance, Customer ID, Age Group | – |
| **Goal 1** | Find **total transaction amount per bank** | Access Bank → ₹X, HDFC → ₹Y, ICICI → ₹Z |
| **Steps (Basic Group By)** | 1. Select **Sent From Bank** column2. Choose **Basic** mode3. Operation: **Sum** on Transaction Amount column4. New column name: **Sum of Transactions** | Output: 2 columns → Bank Name & Sum of Transactions |
| **Applied Steps** | Power Query records each step on the **right-hand pane**. You can remove steps (e.g., "Grouped Rows") to go back to original data. | – |
| **Goal 2** | Find **total transactions for each bank in each city** | Axis Bank–Delhi, Axis Bank–Mumbai, etc. |
| **Steps (Advanced Group By)** | 1. Select **Advanced** mode2. Group by **Bank** + **City**3. Operation: **Sum** on Transaction Amount column4. New column: **Total Transactions** | Output: 3 columns → Bank, City, Total Transactions |
| **Sorting** | You can sort by Bank or City to easily view results. | Axis Bank (Delhi, Mumbai, …), HDFC (Delhi, Chennai, …) |
| **Discard Changes** | If you don’t want to apply Group By: Home tab → **Close & Discard**. | Data returns to original state. |

## 🔑 Key Takeaways

* **Group By** is used for **aggregation** (Sum, Count, Average, etc.).
* Use **Basic mode** when grouping by **1 column**.
* Use **Advanced mode** when grouping by **multiple columns**.
* **Applied Steps** keeps history — you can undo anytime.
* You can **discard changes** to restore original dataset.

# 📘 Power BI – Pivot, Unpivot & Transpose (Power Query Editor)

| **Function** | **What It Does** | **Example (Before)** | **Example (After)** | **Key Use Case** |
| --- | --- | --- | --- | --- |
| **Transpose** | Converts **rows → columns** and **columns → rows** | Table: 4 rows × 3 columns (ID, Marks, Student) | Table: 3 rows × 4 columns (ID, Marks, Student become rows) | Useful when you want to **flip table orientation** |
| **Pivot** | Converts **unique values in a column → new columns** and aggregates other column values | Table: Month, Day, Sales, Profit | Month values (Jan, Feb, Mar, Apr) become **columns** with aggregated **Sales** | Used to create a **summary table** (like Excel Pivot Table) |
| **Unpivot** | Converts **columns → rows** by creating **attribute-value pairs** | Table: Items, Jan, Feb, Mar | "Attribute" column (Jan/Feb/Mar) + "Value" column (numbers) | Useful for bringing data into **long format** for analysis |

## 🔎 Step-by-Step Details

### 1. **Transpose**

* Rows ↔ Columns swap.
* If table = 4 rows × 3 columns, after transpose → 3 rows × 4 columns.
* Example:
  + Before: Columns = **ID, Marks, Student**
  + After: Rows = ID row, Marks row, Student row.

### 2. **Pivot**

* Choose a column whose **unique values** should become **new columns**.
* Must specify **which column to aggregate** and **how** (Sum, Average, etc.).
* Example:
  + Before:
    - Month | Day | Sales | Profit
    - Jan | 3 | 88 | 99
    - Jan | 3 | 88 | 99
  + After (Pivot on **Month**, Aggregate **Sales by Sum**):
    - Day | Profit | Jan | Feb | Mar | Apr
    - 3 | 99 | 176 | ... | ... | ...

### 3. **Unpivot**

* Select multiple **columns → turns them into rows**.
* Creates 2 new columns:
  + **Attribute** = former column name (e.g., Jan, Feb, Mar).
  + **Value** = data inside those columns.
* Example:
  + Before:
    - Item | Jan | Feb | Mar
    - Pen | 50 | 40 | 60
  + After:
    - Item | Attribute | Value
    - Pen | Jan | 50
    - Pen | Feb | 40
    - Pen | Mar | 60

## 🔑 Key Takeaways

* **Transpose** → Swap rows/columns.
* **Pivot** → Wider table (columns created from unique values).
* **Unpivot** → Longer table (columns melted into rows).
* **Interview Tip:** Pivot/Unpivot is commonly asked because it tests understanding of **reshaping data**.

# **📘 Power BI – Merge Queries (All Joins)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Join Type (Power BI)** | **SQL Equivalent** | **What It Does** | **Example Result (with Student Tables)** |
| **Inner Join** | INNER JOIN | Returns only matching rows from both tables. | Students 1, 2, 3, 4 (since only they exist in both **All Student Details** and **Student Marks**). |
| **Left Outer Join** | LEFT JOIN | Keeps all rows from **first (left) table**, adds matching data from second table; if no match → NULL. | Students 1–7 appear. For IDs 5, 9, 12 → Marks = NULL. |
| **Right Outer Join** | RIGHT JOIN | Keeps all rows from **second (right) table**, adds matching data from first table; if no match → NULL. | Students 1–4 appear with full details. If extra IDs were in Marks (not in Details), those would still show. |
| **Full Outer Join** | FULL OUTER JOIN | Keeps all rows from both tables. If no match, fills with NULL. | Students 1–7 from Details + all from Marks. Missing values are NULL. |
| **Left Anti Join** | LEFT ANTI | Returns rows from **first table** that have **no match** in the second table. | Students 5, 9, 12 (exist in Details but not in Marks). |
| **Right Anti Join** | RIGHT ANTI | Returns rows from **second table** that have **no match** in the first table. | If Marks had IDs not in Details (say ID 15), those would appear here. |

## **🔹 Visual Example**

Imagine:

**All Student Details → IDs = {1, 2, 3, 4, 5, 9, 12}**

**Student Marks → IDs = {1, 2, 3, 4}**

**Inner Join** → {1, 2, 3, 4}

**Left Outer Join** → {1, 2, 3, 4, 5, 9, 12} (Marks = NULL for 5, 9, 12)

**Right Outer Join** → {1, 2, 3, 4}

**Full Outer Join** → {1, 2, 3, 4, 5, 9, 12}

**Left Anti Join** → {5, 9, 12}

**Right Anti Join** → {} (empty, unless Marks had IDs not in Details)

# 📘 Power Query Editor – Difference between **Transform** and **Add Column** Tabs

| **Feature** | **Transform Tab** | **Add Column Tab** |
| --- | --- | --- |
| **Effect on Data** | Modifies / overwrites the **existing column** | Creates a **new column** while keeping the original column intact |
| **Use Case** | When you only need the **transformed result** and don’t want to keep the original column | When you need to **preserve the original column** and create new calculated columns |
| **Example – Extract Text Before Delimiter (,)** | Selecting Location column → Transform → Extract → Text Before Delimiter (,) → **Location column replaced** with only **City values** | Selecting Location column → Add Column → Extract → Text Before Delimiter (,) → A **new column** created with **City values**, original Location column still available |
| **Applied Steps** | Shows transformation as step → modifies existing column | Shows step → adds new column along with original |
| **Risk** | You might lose the original data if you overwrite accidentally | Safer, because the original data stays as reference |

## 📝 Example Walkthrough (From Your Exercise)

### Dataset (Location column contains values like):

| Mumbai, Maharashtra, 500001 |

| Delhi, Delhi, 110011 |

| Kolkata, West Bengal, 700001|

### Using **Transform → Extract → Text Before Delimiter (",")**

* **Result:** Location column = Mumbai, Delhi, Kolkata
* (Original column replaced → only cities left).

### Using **Add Column → Extract → Text Before Delimiter (",")**

* **Result:**
  + Original Location column stays intact.
  + New column created = Mumbai, Delhi, Kolkata.

Then you repeated the same with:

* **Text Between Delimiters (",", ",")** → New column = Maharashtra, Delhi, West Bengal
* **Text After Delimiter (",")** → New column = 500001, 110011, 700001

So final clean table becomes:

| **City** | **State** | **Pin Code** |
| --- | --- | --- |
| Mumbai | Maharashtra | 500001 |
| Delhi | Delhi | 110011 |
| Kolkata | West Bengal | 700001 |

## 🔑 Key Takeaway

Use **Transform** → if you want to **overwrite** the existing column.

* Use **Add Column** → if you want to **preserve original data** and add new derived columns.

| **Stage** | **Dataset View** |
| --- | --- |
| **Before (Original Data)** | Location Mumbai, Maharashtra, 500001 Delhi, Delhi, 110011 Kolkata, West Bengal, 700001 |
| **Transform → Extract (Before “,”)** | Location Mumbai Delhi Kolkata |
| **Add Column → Extract (Before “,”)** | Location |

### 📝 Quick Notes

* **Transform** → Replaces original column (no backup).
* **Add Column** → Keeps original + adds new columns (City, State, Pin Code).
* Best practice: Use **Add Column** when cleaning, unless you’re sure original column isn’t needed.