SALES ANALYSIS PROJECT DOCUMENTATION

Prepared by: *Syed Karimoon*
Tools: Excel, MySQL, Python, Power BI

1. Abstract

This Sales Analysis project focuses on analyzing sales data to generate meaningful business insights using **Excel, MySQL, Python, and Power BI**. The project aims to understand **daily sales trends, top customers, and product performance** to support data-driven decision-making.

---

2. Business Problem

The organization faced challenges in analyzing sales performance due to manual reporting. There was no clear visibility into:

- Daily sales trends
- Top-performing customers
- Best-selling products

This project solves these issues using analytical tools and visualization.

---

3. Dataset Description

The dataset contains sales transaction data with the following columns:

- Order ID
- Order Date
- Customer Name
- Product Name
- Quantity
- Sales Amount

The data was initially available in Excel and later stored in MySQL for structured analysis.

Tools & Technologies Used:

- **Excel** – Initial data cleaning
- **MySQL** – Query-based analysis
- **Python (Pandas, Matplotlib)** – Data merging and analysis
- **Power BI** – Dashboard and visualization

This is the **Raw Sales Dataset:**

Order_ID	Order_Date	Customer_Name	Product	Category	Quantity	Unit_Price	Total_Sales
ORD100101	01-2025 00:00	Aarav	Laptop	Electronics	1	55000	55000
ORD100202	01-2025 00:00	Ananya	Mobile	Electronics	2	20000	40000
ORD100303	01-2025 00:00	Rohit	Tablet	Electronics	1	30000	30000
ORD100404	01-2025 00:00	Priya	Headphones	Accessories	3	2000	6000
ORD100505	01-2025 00:00	Karthik	Monitor	Electronics	2	15000	30000
ORD100606	01-2025 00:00	Sneha	Keyboard	Accessories	1	1500	1500
ORD100707	01-2025 00:00	Vikram	Mouse	Accessories	2	800	1600
ORD100808	01-2025 00:00	Pooja	Printer	Electronics	1	12000	12000
ORD100909	01-2025 00:00	Rahul	Camera	Electronics	1	25000	25000
ORD101010	01-2025 00:00	Neha	Smartwatch	Electronics	2	18000	36000
ORD101111	01-2025 00:00	Suresh	Laptop	Electronics	1	55000	55000
ORD101212	01-2025 00:00	Divya	Mobile	Electronics	2	20000	40000
ORD101313	01-2025 00:00	Arjun	Tablet	Electronics	1	30000	30000
ORD101414	01-2025 00:00	Kavya	Headphones	Accessories	3	2000	6000
ORD101515	01-2025 00:00	Manoj	Monitor	Electronics	2	15000	30000
ORD101616	01-2025 00:00	Ritu	Keyboard	Accessories	1	1500	1500
ORD101717	01-2025 00:00	Amit	Mouse	Accessories	2	800	1600
ORD101818	01-2025 00:00	Shalini	Printer	Electronics	1	12000	12000
ORD101919	01-2025 00:00	Naveen	Camera	Electronics	1	25000	25000
ORD102020	01-2025 00:00	Swathi	Smartwatch	Electronics	2	18000	36000

Raw Sales Dataset

Sales Dataset – Column Structure

Column Name	Description
Order_ID	Unique Order Number
Order_Date	Date of the Order
Customer_Name	Name of the Customer
Product_Name	Product Purchased
Quantity	Number of Units Sold
Sales	Total Sales Amount

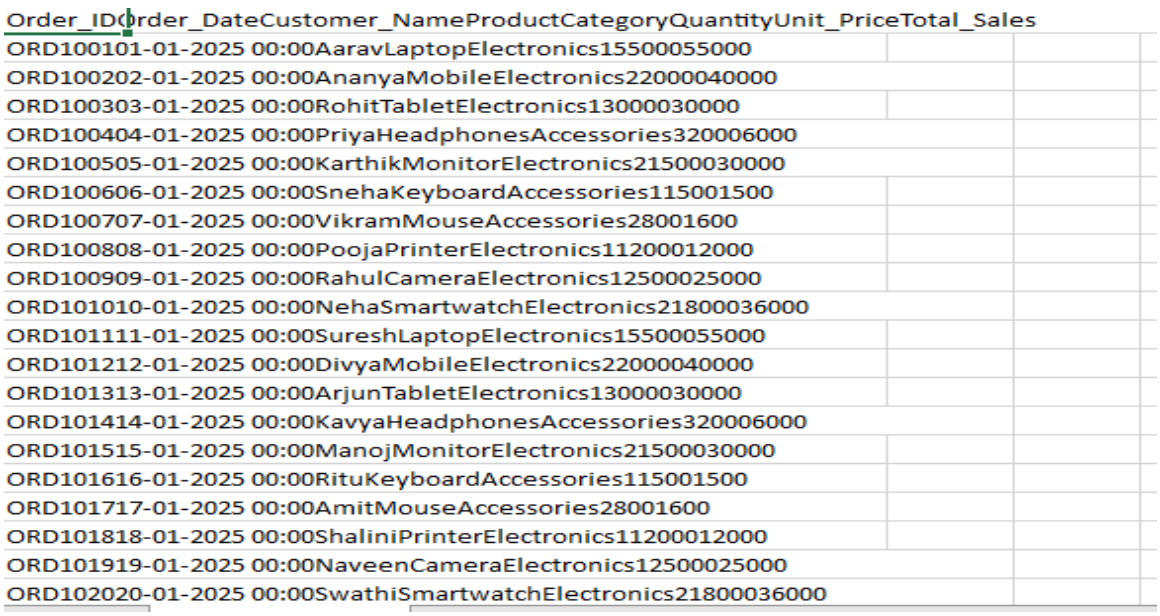
Dataset Column Structure

4. Data Cleaning and Preparation (Excel)

This section explains how the raw sales dataset was reviewed, validated, and prepared in **Microsoft Excel** before analysis. Even though the dataset did not contain missing or null values, standard data quality checks and transformations were performed to ensure accuracy and consistency.

4.1 RAW DATA REVIEW

- Imported the raw sales dataset into Excel
- Verified column names and data types
- Checked row count to ensure full data load



Order_IDOrder_DateCustomer_NameProductCategoryQuantityUnit_PriceTotal_Sales
ORD100101-01-2025 00:00AaravLaptopElectronics15500055000
ORD100202-01-2025 00:00AnanyaMobileElectronics22000040000
ORD100303-01-2025 00:00RohitTabletElectronics13000030000
ORD100404-01-2025 00:00PriyaHeadphonesAccessories320006000
ORD100505-01-2025 00:00KarthikMonitorElectronics21500030000
ORD100606-01-2025 00:00SnehaKeyboardAccessories115001500
ORD100707-01-2025 00:00VikramMouseAccessories28001600
ORD100808-01-2025 00:00PoojaPrinterElectronics11200012000
ORD100909-01-2025 00:00RahulCameraElectronics12500025000
ORD101010-01-2025 00:00NehaSmartwatchElectronics21800036000
ORD101111-01-2025 00:00SureshLaptopElectronics15500055000
ORD101212-01-2025 00:00DivyaMobileElectronics22000040000
ORD101313-01-2025 00:00ArjunTabletElectronics13000030000
ORD101414-01-2025 00:00KavyaHeadphonesAccessories320006000
ORD101515-01-2025 00:00ManojMonitorElectronics21500030000
ORD101616-01-2025 00:00RituKeyboardAccessories115001500
ORD101717-01-2025 00:00AmitMouseAccessories28001600
ORD101818-01-2025 00:00ShaliniPrinterElectronics11200012000
ORD101919-01-2025 00:00NaveenCameraElectronics12500025000
ORD102020-01-2025 00:00SwathiSmartwatchElectronics21800036000

Data Before Cleaning (Excel)

Order_ID	Order_Date	Customer_Name	Product	Category	Quantity	Unit_Price	Total_Sales
ORD1001	01-01-2025	Aarav	Laptop	Electronics	1	55000.00	55000.00
ORD1002	02-01-2025	Ananya	Mobile	Electronics	2	20000.00	40000.00
ORD1003	03-01-2025	Rohit	Tablet	Electronics	1	30000.00	30000.00
ORD1004	04-01-2025	Priya	Headphones	Accessories	3	2000.00	6000.00
ORD1005	05-01-2025	Karthik	Monitor	Electronics	2	15000.00	30000.00
ORD1006	06-01-2025	Sneha	Keyboard	Accessories	1	1500.00	1500.00
ORD1007	07-01-2025	Vikram	Mouse	Accessories	2	800.00	1600.00
ORD1008	08-01-2025	Pooja	Printer	Electronics	1	12000.00	12000.00
ORD1009	09-01-2025	Rahul	Camera	Electronics	1	25000.00	25000.00
ORD1010	10-01-2025	Neha	Smartwatch	Electronics	2	18000.00	36000.00
ORD1011	11-01-2025	Suresh	Laptop	Electronics	1	55000.00	55000.00
ORD1012	12-01-2025	Divya	Mobile	Electronics	2	20000.00	40000.00
ORD1013	13-01-2025	Arjun	Tablet	Electronics	1	30000.00	30000.00
ORD1014	14-01-2025	Kavya	Headphones	Accessories	3	2000.00	6000.00
ORD1015	15-01-2025	Manoj	Monitor	Electronics	2	15000.00	30000.00
ORD1016	16-01-2025	Ritu	Keyboard	Accessories	1	1500.00	1500.00
ORD1017	17-01-2025	Amit	Mouse	Accessories	2	800.00	1600.00
ORD1018	18-01-2025	Shalini	Printer	Electronics	1	12000.00	12000.00
ORD1019	19-01-2025	Naveen	Camera	Electronics	1	25000.00	25000.00
ORD1020	20-01-2025	Swathi	Smartwatch	Electronics	2	18000.00	36000.00
ORD1021	21-01-2025	Deepak	Laptop	Electronics	1	55000.00	55000.00

Data After Cleaning (Excel)

4.2 DATA VALIDATION CHECKS

The following checks were performed to confirm data quality:

1. NULL & MISSING VALUE CHECK

- Used **Filter** and **Go To Special → Blanks**
- Confirmed that **no null or missing values** were present in any column

2. DUPLICATE CHECK

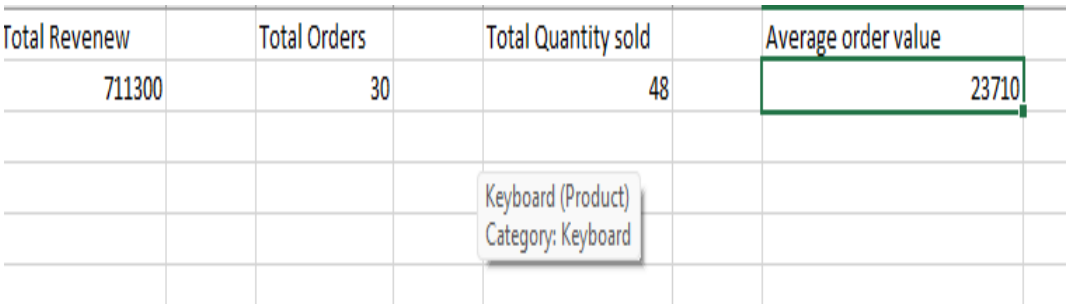
- Checked Order_ID for duplicates using:
  - Conditional Formatting → Highlight Duplicates
- Result: **No duplicate records found**

3. Data Type Validation

Column Name	Expected Type	Validation
Order_Date	Date	Correct date format
Quantity	Number	Numeric values
Sales	Currency	Numeric values
Customer_Name	Text	Valid text
Product_Name	Text	Valid text

4. ADDITIONAL METRICS

- Total Revenue
- Total orders
- Total Quantity sold
- Average order value



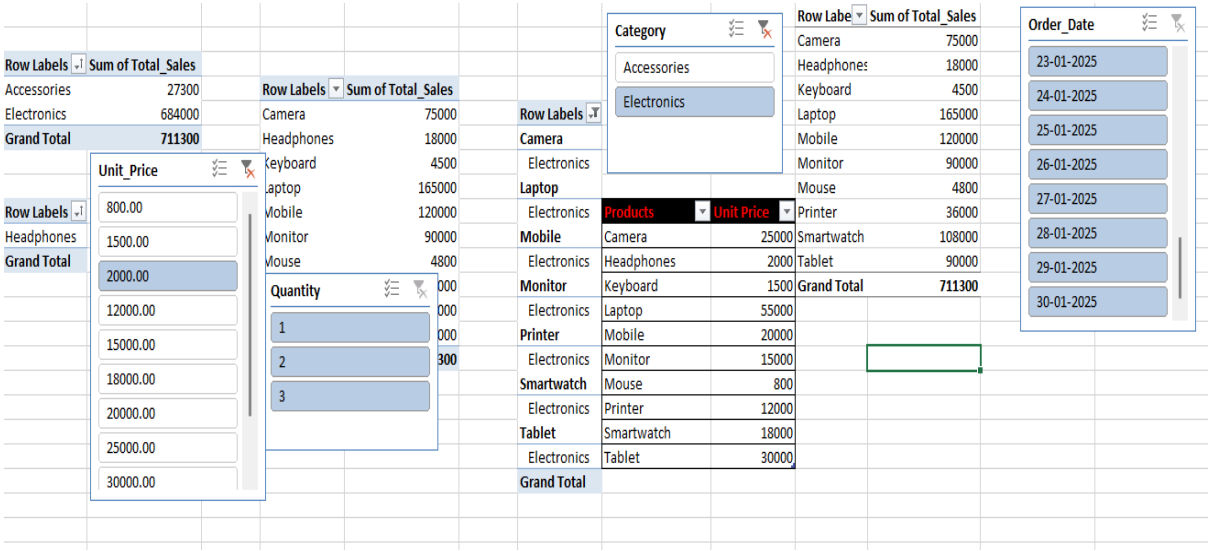
Calculated values

4.3 PIVOT TABLE PREPARATION

Pivot tables were created to summarize sales data efficiently.

PIVOT TABLES CREATED:

- Total Sales by Product
- Total Sales by Category
- Product vs Category Mapping
- Date-Wise Sales Analysis
- Interactive Dashboard Design
- Business Insights Derived



Pivot Tables

4.4. INTERACTIVE DASHBOARD DESIGN

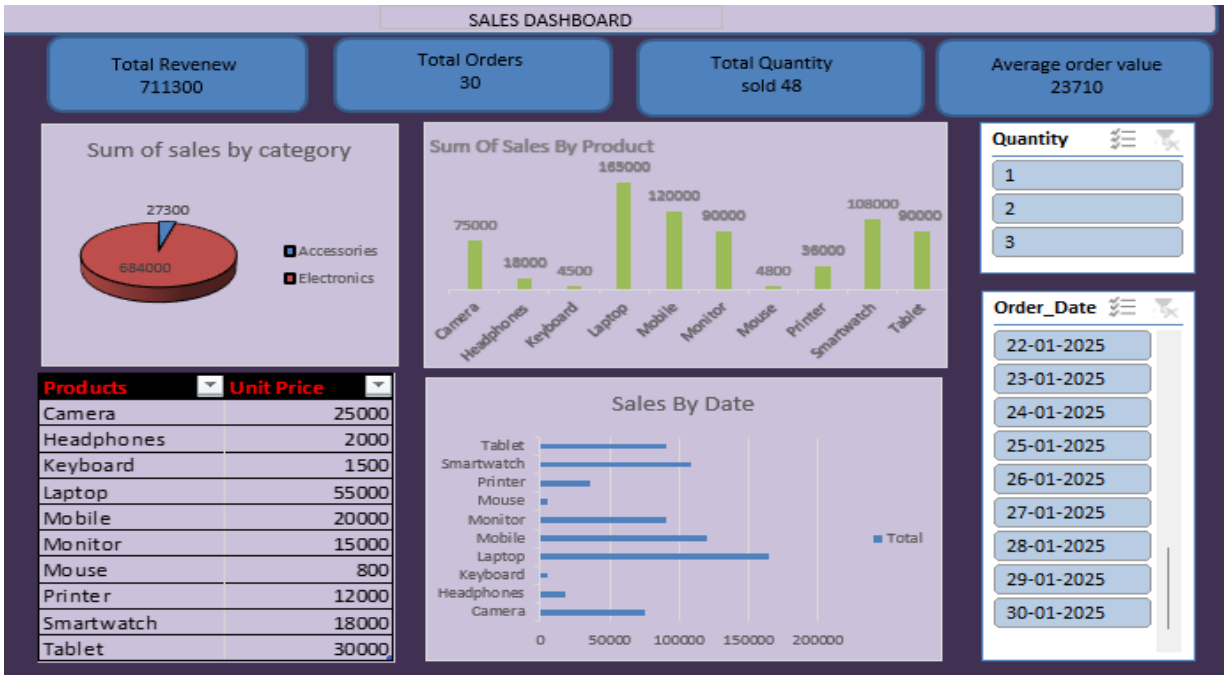
An interactive Excel dashboard was created using:

COMPONENTS:

- Multiple pivot tables
- Slicers for:
  - Category
  - Unit Price
  - Quantity
  - Order Date
- Linked pivot tables for dynamic filtering

Dashboard Features:

- Single click filtering
- Real-time pivot updates
- Easy comparison of products and categories



Full Excel dashboard

4.5. BUSINESS INSIGHTS DERIVED

- Electronics category dominates total sales
- High-priced items like Laptop and Mobile generate maximum revenue
- Quantity-based filtering shows bulk purchases significantly impact sales
- Date slicers help identify peak sales days

4.6. EXCEL AS A FOUNDATION TOOL

The Excel analysis served as the **foundation layer** for:

- MySQL data modelling
- Power BI dashboard creation
- Python-based data analysis

---

5. MySQL Implementation and Analysis

Goal:

Import your **Excel table** into **MySQL** so you can:

- Practice SQL queries
- Use it as a **resume project**
- Later connect it to **Power BI**

METHOD USED: Excel → CSV → MySQL

STEP 1: PREPARE YOUR EXCEL FILE

1. Open Excel file
2. Make sure:
    - Row 1 = Column names (headers)
      Example:
    - Customer_ID | Customer_Name | Product | Category | Sales | Quantity | Order_Date |
3. Click File → Save As
4. Choose Save as type:
   CSV (Comma delimited) (*.csv)
5. Name it:
6. sales_data.csv
7. Save it (click Yes)

STEP 2: OPEN MYSQL WORKBENCH

1. Open MySQL Workbench
2. Click on your connection (example: Local instance MySQL)
3. Enter root password
4. You will see the SQL Editor screen

STEP 3: CREATE A DATABASE

In the SQL editor, type this:

```
CREATE DATABASE excel_sql_project;
```

Click the **Execute button**

Now select the database:

```
USE excel_sql_project;
```

STEP 4: CREATE TABLE

Your MySQL table must match Excel columns

STEP 5: IMPORT CSV INTO MYSQL

1. In **MySQL Workbench**
2. On the left side → **Schemas**
3. Expand excel_sql_project
4. Right-click **Tables**
5. Click **Table Data Import Wizard**

WIZARD STEPS:

STEP 1:

- Select file:
  sales_data.csv
- Click **Next**

Step 2:

- File type: **CSV**
- Encoding: **utf-8**
- Check **"First row contains column names"**
- Click **Next**

STEP 3:

- Choose **Existing Table**
- Select: sales_data
- Click **Next**

STEP 4:

- Review mapping
- Click **Next**

Step 5:

- Click **Import**

You will see **Import completed successfully**

STEP 6:

Verify Data

Run this SQL:

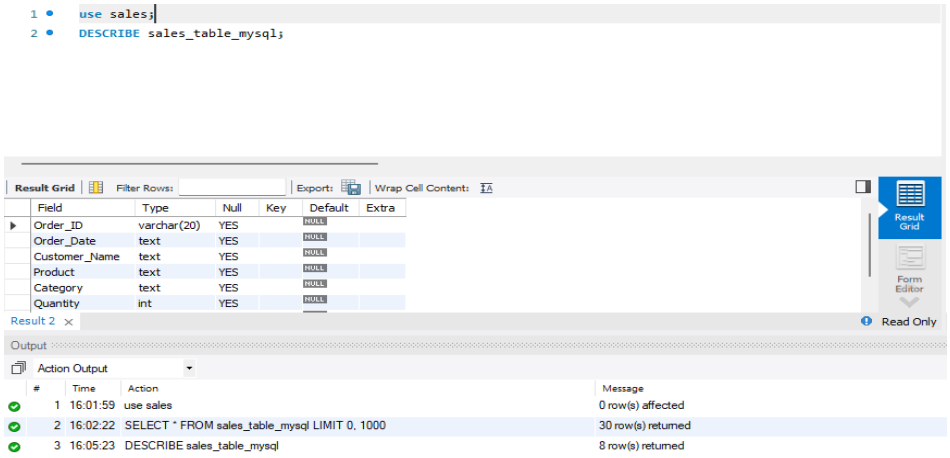
```
SELECT * FROM sales_data;
```

You should see all the table **rows from Excel**

Check Table Structure:

Run:

```
DESCRIBE your_table_name;
```



The cleaned data was imported into MySQL and normalized into multiple tables:

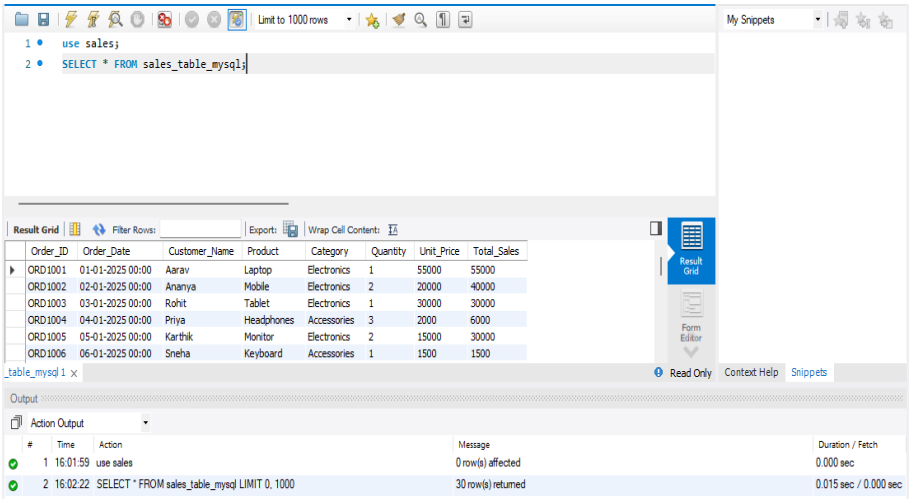
- Customers
- Products
- Orders

WHAT IS NORMALIZATION?

Normalization = breaking one big table into smaller tables to:

1. Remove data duplication
2. Improve data consistency
3. Make database professional
4. Match **real-world industry design**

Current Table (Unnormalized)



```
use sales;
SELECT * FROM sales_table_mysql;
```

Order_ID	Order_Date	Customer_Name	Product	Category	Quantity	Unit_Price	Total_Sales
ORD1001	01-01-2025 00:00	Aarav	Laptop	Electronics	1	55000	55000
ORD1002	02-01-2025 00:00	Ananya	Mobile	Electronics	2	20000	40000
ORD1003	03-01-2025 00:00	Rohit	Tablet	Electronics	1	30000	30000
ORD1004	04-01-2025 00:00	Priya	Headphones	Accessories	3	2000	6000
ORD1005	05-01-2025 00:00	Karthik	Monitor	Electronics	2	15000	30000
ORD1006	06-01-2025 00:00	Sneha	Keyboard	Accessories	1	1500	1500

```
1  16:01:59  use sales                                  0 row(s) affected     0.000 sec
2  16:02:22  SELECT * FROM sales_table_mysql LIMIT 0, 1000  30 row(s) returned  0.015 sec / 0.000 sec
```

Problems:

- Customer name repeated many times
- Product & category repeated

Normalization Process:

Now We Will Normalize to **3rd Normal Form (3NF)**

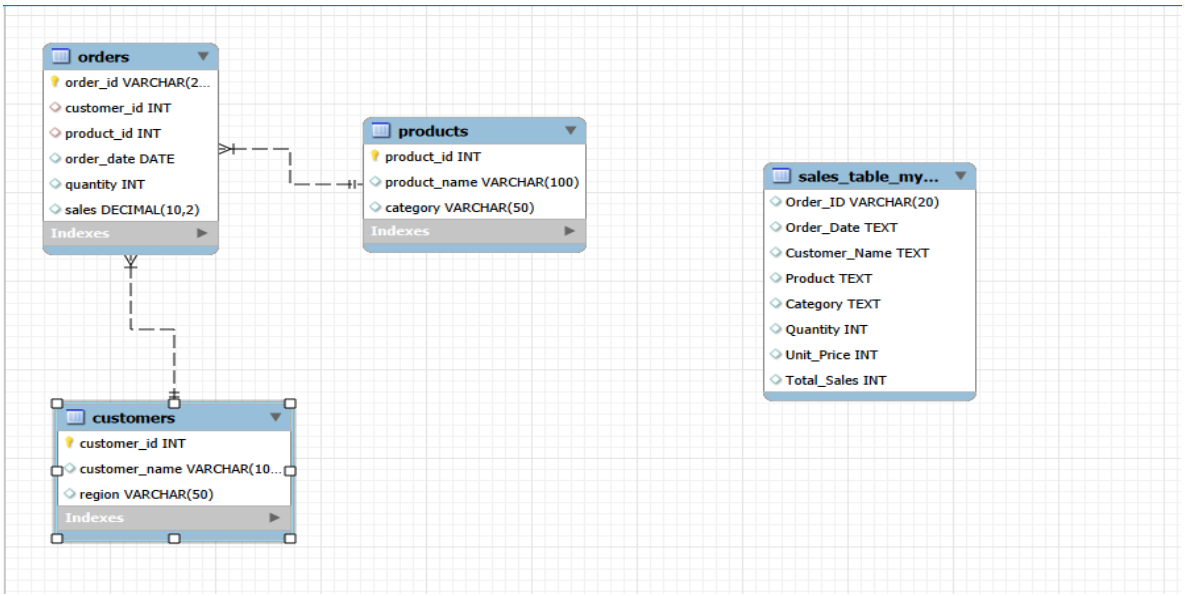
We will create:

1 Customers table
2. Products table
3.Orders (Fact) table

```
use sales;
/*  Table Normalization  */

/* 1.Customers Table */

CREATE TABLE customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_name VARCHAR(100)
);
INSERT INTO customers (customer_name)
SELECT DISTINCT Customer_Name
FROM sales_table_mysql;

/* Product table*/

CREATE TABLE products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    product_name VARCHAR(100),
    category VARCHAR(50)
);
INSERT INTO products (product_name, category)
SELECT DISTINCT Product, Category
FROM sales_table_mysql;
```

```sql
CREATE TABLE orders (
    order_id VARCHAR(20),
    customer_id INT,
    product_id INT,
    order_date DATE,
    quantity INT,
    Total_sales DECIMAL(10,2),

    PRIMARY KEY (order_id),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
INSERT INTO orders (order_id, customer_id, product_id, order_date, quantity, sales)
SELECT
    s.Order_ID,
    c.customer_id,
    p.product_id,
    STR_TO_DATE(s.Order_Date, '%d-%m-%Y %H:%i'),
    s.Quantity,
    s.Total_Sales
FROM sales_table_mysql s
JOIN customers c
    ON s.Customer_Name = c.customer_name
JOIN products p
```

MODEL VIEW:



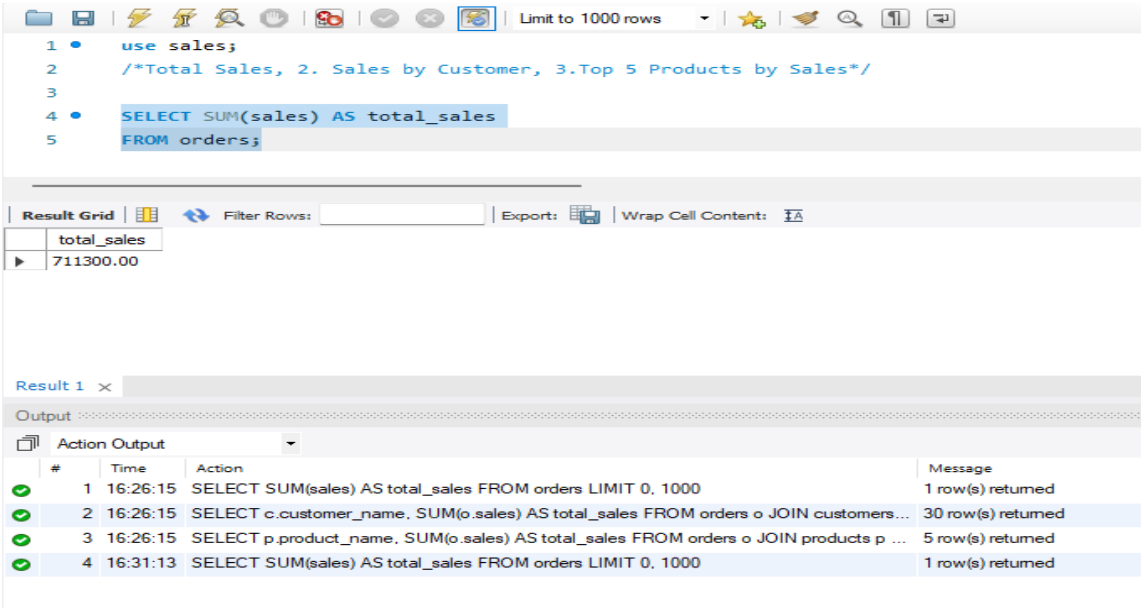
SQL queries were used for:

- Table joins
- Daily sales calculation
- Top customers analysis
- Product-wise sales analysis
- CTEs and window functions

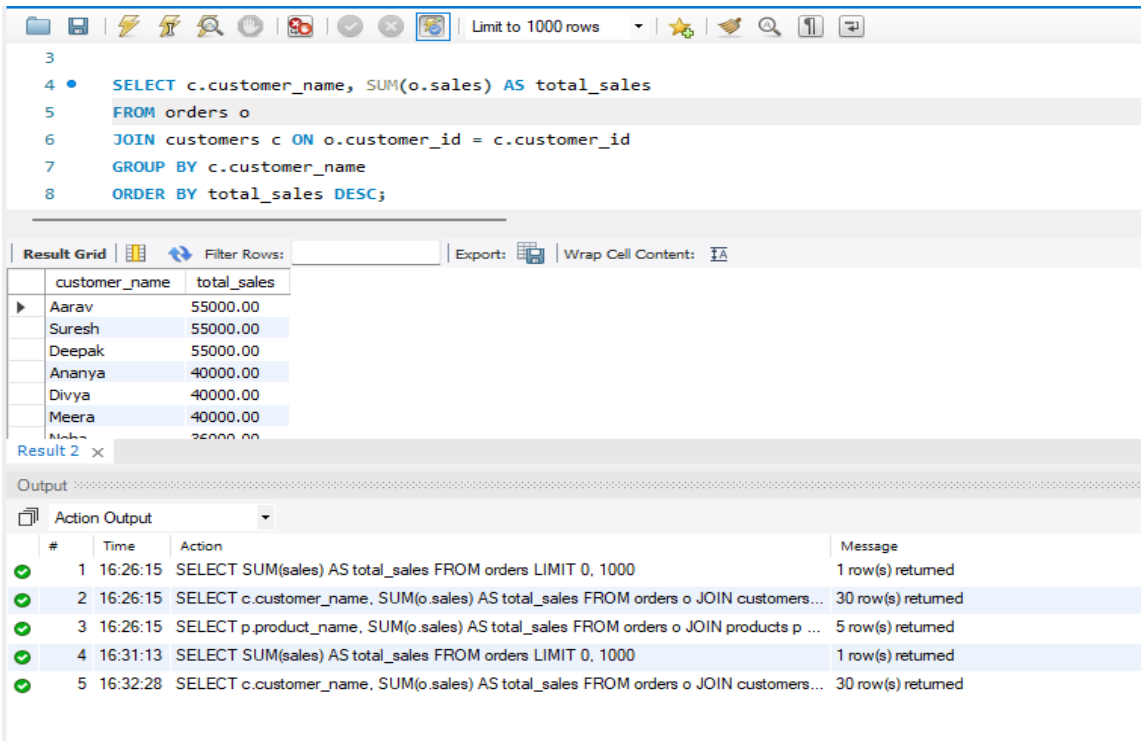
Screenshot Template:

The following screenshots demonstrate the execution of various SQL queries in MySQL Workbench. Each screenshot includes the query input, output result, and validation details to ensure correctness and error-free execution.

 Total Sales Calculation



 Sales by Customer



🔹 Top 5 Products by Sales

```sql
SELECT p.product_name, SUM(o.sales) AS total_sales
FROM orders o
JOIN products p ON o.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_sales DESC
LIMIT 5;
```

product_name	total_sales
Laptop	165000.00
Mobile	120000.00
Smartwatch	108000.00
Tablet	90000.00
Monitor	90000.00

Result 3

Output

Action Output

#	Time	Action	Message
1	16:26:15	SELECT SUM(sales) AS total_sales FROM orders LIMIT 0, 1000	1 row(s) returned
2	16:26:15	SELECT c.customer_name, SUM(o.sales) AS total_sales FROM orders o JOIN customers...	30 row(s) returned
3	16:26:15	SELECT p.product_name, SUM(o.sales) AS total_sales FROM orders o JOIN products p ...	5 row(s) returned
4	16:31:13	SELECT SUM(sales) AS total_sales FROM orders LIMIT 0, 1000	1 row(s) returned
5	16:32:28	SELECT c.customer_name, SUM(o.sales) AS total_sales FROM orders o JOIN customers...	30 row(s) returned
6	16:33:57	SELECT p.product_name, SUM(o.sales) AS total_sales FROM orders o JOIN products p ...	5 row(s) returned

🔹 Date-wise Sales Analysis

Day wise total sales

```sql
SELECT
    DATE(order_date) AS order_day,
    SUM(sales) AS daily_sales
FROM orders
GROUP BY order_day
ORDER BY order_day;
```

order_day	daily_sales
2025-01-04	6000.00
2025-01-05	30000.00
2025-01-06	1500.00
2025-01-07	1600.00
2025-01-08	12000.00

Result 4

Output

Action Output

#	Time	Action	Message
1	16:26:15	SELECT SUM(sales) AS total_sales FROM orders LIMIT 0, 1000	1 row(s) returned
2	16:26:15	SELECT c.customer_name, SUM(o.sales) AS total_sales FROM orders o JOIN customers...	30 row(s) returned
3	16:26:15	SELECT p.product_name, SUM(o.sales) AS total_sales FROM orders o JOIN products p ...	5 row(s) returned
4	16:31:13	SELECT SUM(sales) AS total_sales FROM orders LIMIT 0, 1000	1 row(s) returned
5	16:32:28	SELECT c.customer_name, SUM(o.sales) AS total_sales FROM orders o JOIN customers...	30 row(s) returned
6	16:33:57	SELECT p.product_name, SUM(o.sales) AS total_sales FROM orders o JOIN products p ...	5 row(s) returned
7	16:40:10	SELECT DATE(order_date) AS order_day, SUM(sales) AS daily_sales FROM order...	30 row(s) returned

Highest Sales Day

```sql
SELECT
    DATE(order_date) AS order_day,
    SUM(sales) AS total_sales
FROM orders
GROUP BY order_day
ORDER BY total_sales DESC
LIMIT 1;
```

order_day	total_sales
2025-01-01	55000.00

Result 5

Output

Action Output

#	Time	Action	Message
2	16:26:15	SELECT c.customer_name, SUM(o.sales) AS total_sales FROM orders o JOIN custom...	30 row(s) returned
3	16:26:15	SELECT p.product_name, SUM(o.sales) AS total_sales FROM orders o JOIN products ...	5 row(s) returned
4	16:31:13	SELECT SUM(sales) AS total_sales FROM orders LIMIT 0, 1000	1 row(s) returned
5	16:32:28	SELECT c.customer_name, SUM(o.sales) AS total_sales FROM orders o JOIN custom...	30 row(s) returned
6	16:33:57	SELECT p.product_name, SUM(o.sales) AS total_sales FROM orders o JOIN products ...	5 row(s) returned
7	16:40:10	SELECT DATE(order_date) AS order_day, SUM(sales) AS daily_sales FROM ord...	30 row(s) returned
8	16:41:59	SELECT DATE(order_date) AS order_day, SUM(sales) AS total_sales FROM ord...	1 row(s) returned

Lowest Sales Day

```sql
/*Lowest Sales Day*/
SELECT
    DATE(order_date) AS order_day,
    SUM(sales) AS total_sales
FROM orders
GROUP BY order_day
ORDER BY total_sales ASC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

order_day	total_sales
2025-01-06	1500.00

Result 6

Output

Action Output

#	Time	Action	Message
4	16:31:13	SELECT SUM(sales) AS total_sales FROM orders LIMIT 0, 1000	1 row(s) returned
5	16:32:28	SELECT c.customer_name, SUM(o.sales) AS total_sales FROM orders o JOIN custome...	30 row(s) returned
6	16:33:57	SELECT p.product_name, SUM(o.sales) AS total_sales FROM orders o JOIN products ...	5 row(s) returned
7	16:40:10	SELECT DATE(order_date) AS order_day, SUM(sales) AS daily_sales FROM ord...	30 row(s) returned
8	16:41:59	SELECT DATE(order_date) AS order_day, SUM(sales) AS total_sales FROM orde...	1 row(s) returned
9	16:46:43	SELECT DATE(order_date) AS order_day, SUM(sales) AS total_sales FROM orde...	1 row(s) returned

Advance CTE to calculate day wise sales analysis

Limit to 1000 rows

```sql
/*Advanced SQL (CTE) for Your Sales Project*/
WITH daily_sales AS (
    SELECT
        DATE(order_date) AS order_day,
        SUM(sales) AS total_sales
    FROM orders
    GROUP BY DATE(order_date)
)
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_day	total_sales
2025-01-26	1500.00
2025-01-27	1600.00
2025-01-28	12000.00
2025-01-29	25000.00
2025-01-30	36000.00

Result 8

Output

Action Output

#	Time	Action	Message
6	16:33:57	SELECT p.product_name, SUM(o.sales) AS total_sales FROM orders o JOIN products ...	5 row(s) returned
7	16:40:10	SELECT DATE(order_date) AS order_day, SUM(sales) AS daily_sales FROM ord...	30 row(s) returned
8	16:41:59	SELECT DATE(order_date) AS order_day, SUM(sales) AS total_sales FROM orde...	1 row(s) returned
9	16:46:43	SELECT DATE(order_date) AS order_day, SUM(sales) AS total_sales FROM orde...	1 row(s) returned
10	16:51:12	WITH daily_sales AS (SELECT DATE(order_date) AS order_day, SUM(sal...	30 row(s) returned
11	16:52:32	WITH daily_sales AS (SELECT DATE(order_date) AS order_day, SUM(sal...	30 row(s) returned

Window Function: Rank Days by Sales

```sql
WITH daily_sales AS (
    SELECT
        DATE(order_date) AS order_day,
        SUM(sales) AS total_sales
    FROM orders
    GROUP BY DATE(order_date)
)
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_day	total_sales	sales_rank
2025-01-01	55000.00	1
2025-01-11	55000.00	1
2025-01-21	55000.00	1
2025-01-02	40000.00	4
2025-01-12	40000.00	4
2025-01-22	40000.00	4

Result 7

Output

Action Output

#	Time	Action	Message
5	16:32:28	SELECT c.customer_name, SUM(o.sales) AS total_sales FROM orders o JOIN custome...	30 row(s) returned
6	16:33:57	SELECT p.product_name, SUM(o.sales) AS total_sales FROM orders o JOIN products ...	5 row(s) returned
7	16:40:10	SELECT DATE(order_date) AS order_day, SUM(sales) AS daily_sales FROM ord...	30 row(s) returned
8	16:41:59	SELECT DATE(order_date) AS order_day, SUM(sales) AS total_sales FROM orde...	1 row(s) returned
9	16:46:43	SELECT DATE(order_date) AS order_day, SUM(sales) AS total_sales FROM orde...	1 row(s) returned
10	16:51:12	WITH daily_sales AS (SELECT DATE(order_date) AS order_day, SUM(sal...	30 row(s) returned

 Window Function: compares each day with Previous day

```sql
use sales;
/* Window Function: Compare Each Day with Previous Day*/
WITH daily_sales AS (
    SELECT
        DATE(order_date) AS order_day,
        SUM(sales) AS total_sales
    FROM orders
    GROUP BY DATE(order_date)
```

order_day	total_sales	prev_day_sales	daily_change
2025-01-01	55000.00	NULL	NULL
2025-01-02	40000.00	55000.00	-15000.00
2025-01-03	30000.00	40000.00	-10000.00
2025-01-04	6000.00	30000.00	-24000.00
2025-01-05	30000.00	6000.00	24000.00
2025-01-06	1500.00	30000.00	-28500.00

Result 11

Output

Action Output

#	Time	Action	Message
9	16:46:43	SELECT DATE(order_date) AS order_day, SUM(sales) AS total_sales FROM orde...	1 row(s) returned
10	16:51:12	WITH daily_sales AS (SELECT DATE(order_date) AS order_day, SUM(sal...	30 row(s) returned
11	16:52:32	WITH daily_sales AS (SELECT DATE(order_date) AS order_day, SUM(sal...	30 row(s) returned
12	16:54:51	WITH daily_sales AS (SELECT DATE(order_date) AS order_day, SUM(sal...	30 row(s) returned
13	16:56:50	WITH daily_sales AS (SELECT DATE(order_date) AS order_day, SUM(sal...	30 row(s) returned
14	16:58:51	WITH daily_sales AS (SELECT DATE(order_date) AS order_day, SUM(sal...	30 row(s) returned

---

6. Python Analysis

Python was used to perform advanced analysis using:

- MySQL-connector for database connection
- pandas for data manipulation
- matplotlib for visualization

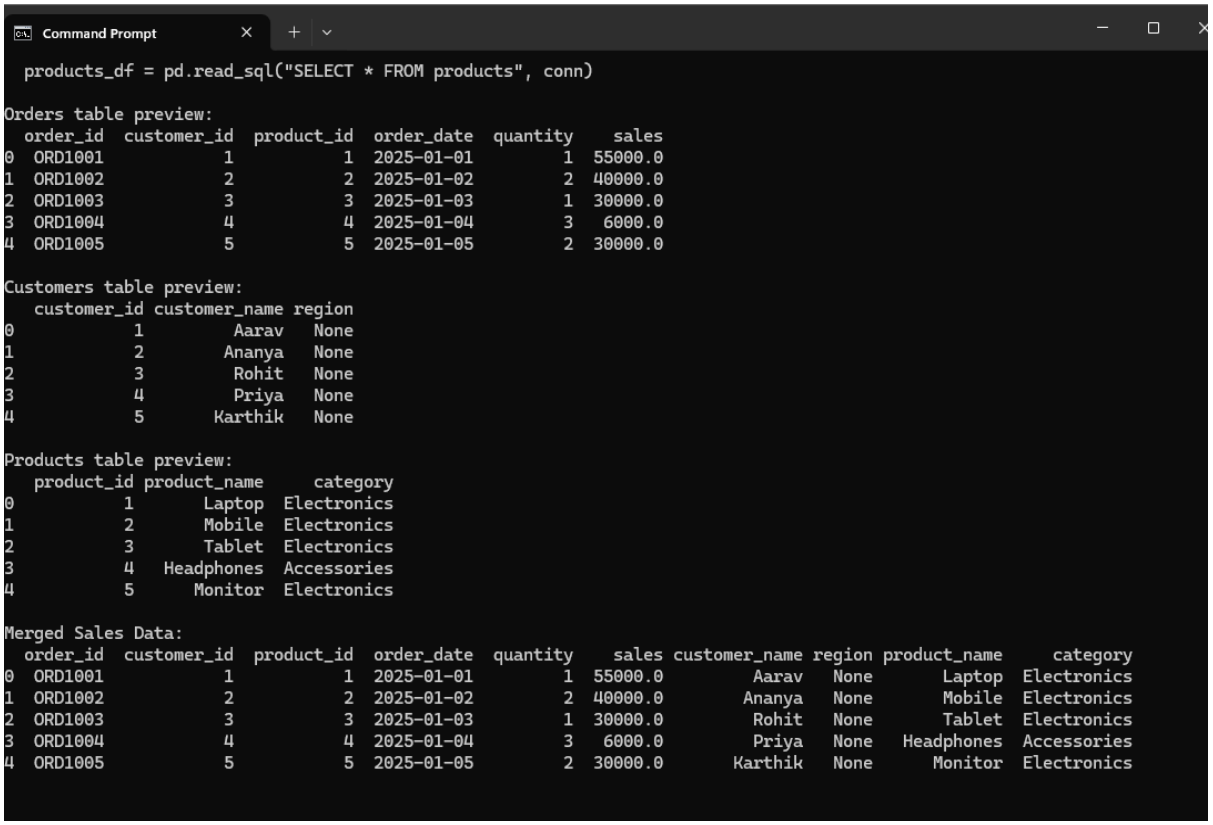
Analysis performed:

The following screenshots demonstrate Python-based data analysis performed using data manipulation and visualization techniques. The analysis includes merging datasets, calculating total and average sales values, and generating graphical representations to identify the top 5 customers. Each screenshot validates correct data processing and meaningful business insights

Python MySQL Connection Output:

```
C:\WINDOWS\py.exe
MySQL connected successfully
C:\Users\lenovo\Desktop\sales_python-project\sales_analysis.py:15: UserWarning: pandas only supports SQLAlchemy connecta
ble (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please
 consider using SQLAlchemy.
  orders_df = pd.read_sql("SELECT * FROM orders", conn)
C:\Users\lenovo\Desktop\sales_python-project\sales_analysis.py:16: UserWarning: pandas only supports SQLAlchemy connecta
ble (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please
 consider using SQLAlchemy.
  customers_df = pd.read_sql("SELECT * FROM customers", conn)
C:\Users\lenovo\Desktop\sales_python-project\sales_analysis.py:17: UserWarning: pandas only supports SQLAlchemy connecta
ble (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please
 consider using SQLAlchemy.
  products_df = pd.read_sql("SELECT * FROM products", conn)
```

Merged Sales Data



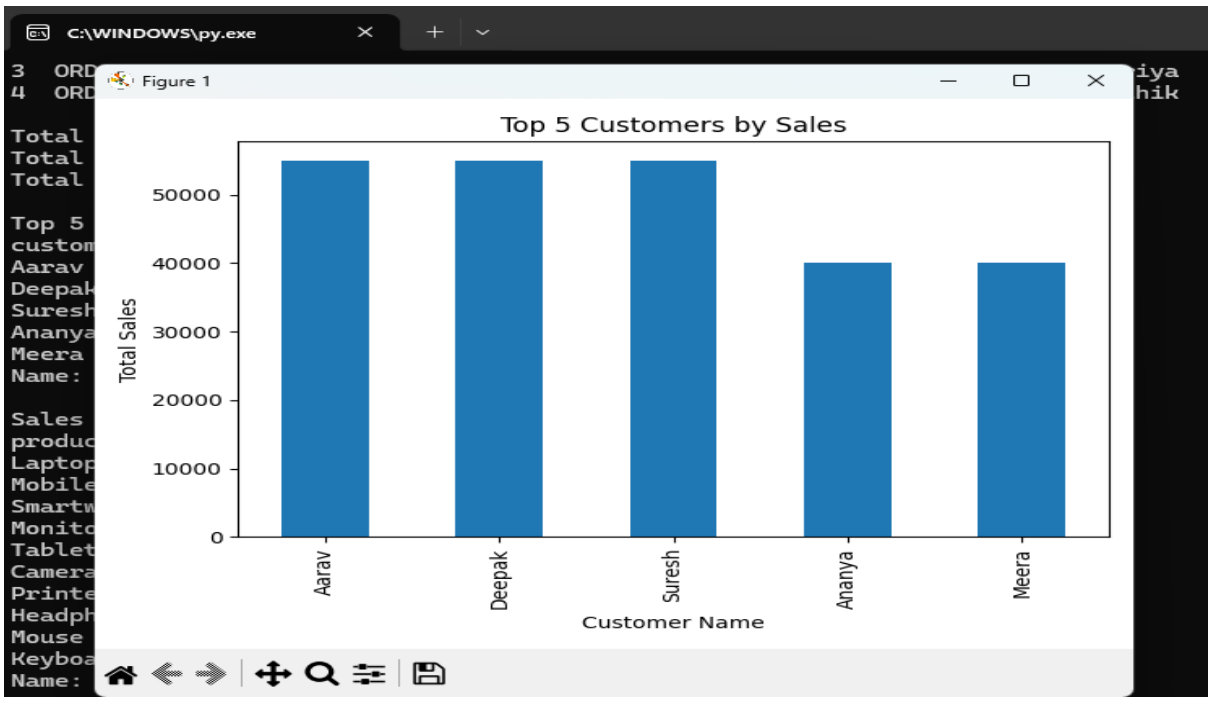
```
Command Prompt
  products_df = pd.read_sql("SELECT * FROM products", conn)

Orders table preview:
  order_id  customer_id  product_id  order_date  quantity    sales
0  ORD1001            1           1  2025-01-01         1  55000.0
1  ORD1002            2           2  2025-01-02         2  40000.0
2  ORD1003            3           3  2025-01-03         1  30000.0
3  ORD1004            4           4  2025-01-04         3   6000.0
4  ORD1005            5           5  2025-01-05         2  30000.0

Customers table preview:
   customer_id customer_name region
0            1         Aarav   None
1            2        Ananya   None
2            3         Rohit   None
3            4         Priya   None
4            5       Karthik   None

Products table preview:
   product_id product_name     category
0           1       Laptop  Electronics
1           2       Mobile  Electronics
2           3       Tablet  Electronics
3           4   Headphones  Accessories
4           5      Monitor  Electronics

Merged Sales Data:
  order_id  customer_id  product_id  order_date  quantity    sales customer_name region product_name     category
0  ORD1001            1           1  2025-01-01         1  55000.0         Aarav   None       Laptop  Electronics
1  ORD1002            2           2  2025-01-02         2  40000.0        Ananya   None       Mobile  Electronics
2  ORD1003            3           3  2025-01-03         1  30000.0         Rohit   None       Tablet  Electronics
3  ORD1004            4           4  2025-01-04         3   6000.0         Priya   None   Headphones  Accessories
4  ORD1005            5           5  2025-01-05         2  30000.0       Karthik   None      Monitor  Electronics
```

pandas for data manipulation:

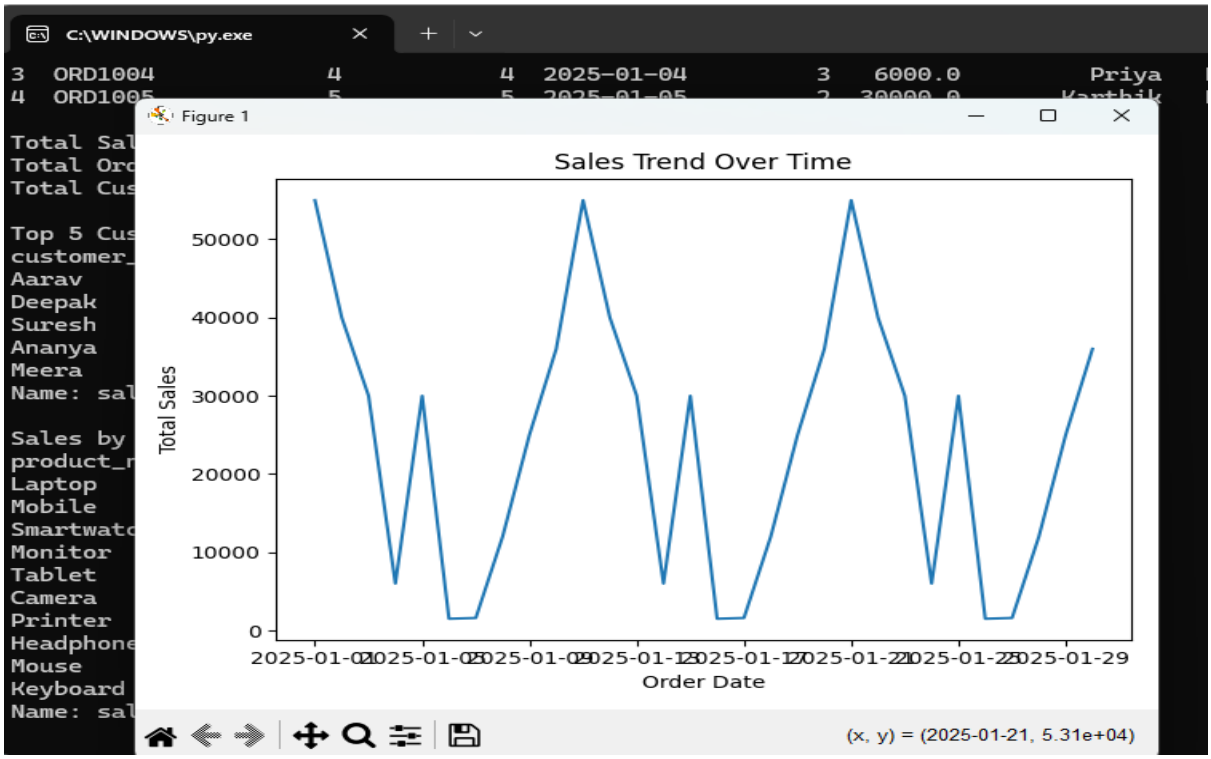
```
Total Sales: 711300.0
Total Orders: 30
Total Customers: 30

Top 5 Customers by Sales:
customer_name
Aarav     55000.0
Deepak    55000.0
Suresh    55000.0
Ananya    40000.0
Meera     40000.0
Name: sales, dtype: float64

Sales by Product:
product_name
Laptop         165000.0
Mobile         120000.0
Smartwatch     108000.0
Monitor         90000.0
Tablet          90000.0
Camera          75000.0
Printer         36000.0
Headphones      18000.0
Mouse            4800.0
Keyboard         4500.0
Name: sales, dtype: float64
```

Python Sales Visualization:

TOP 5 Customers by Sales:



Sales Trend Over Time:



6.Power BI Dashboard

The following screenshots demonstrate data modelling and report development performed in Power BI. The model view illustrates table relationships and data structure, while the report view presents interactive visualizations created to analyze sales performance and business insights.

IMPORTING DATA FROM MYSQL INTO POWER BI

OVERVIEW

This step involves importing structured sales data from a **MySQL database** into **Power BI Desktop** for data modelling and visualization.

DATA SOURCE

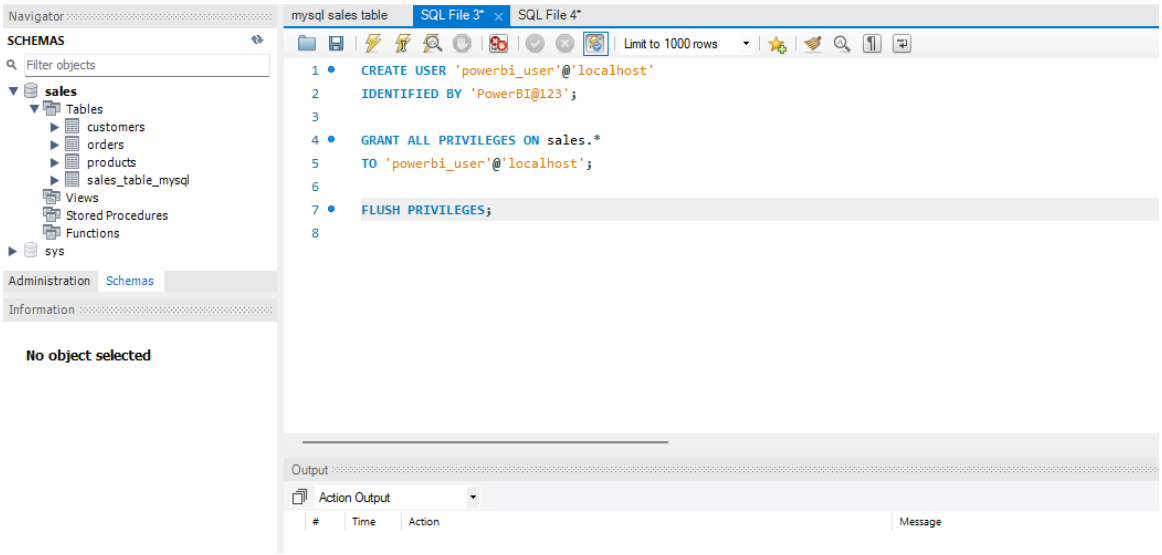
- Database: MySQL
- Connection Mode: Import
- Tables Imported: Sales-related tables required for analysis

IMPORT PROCESS

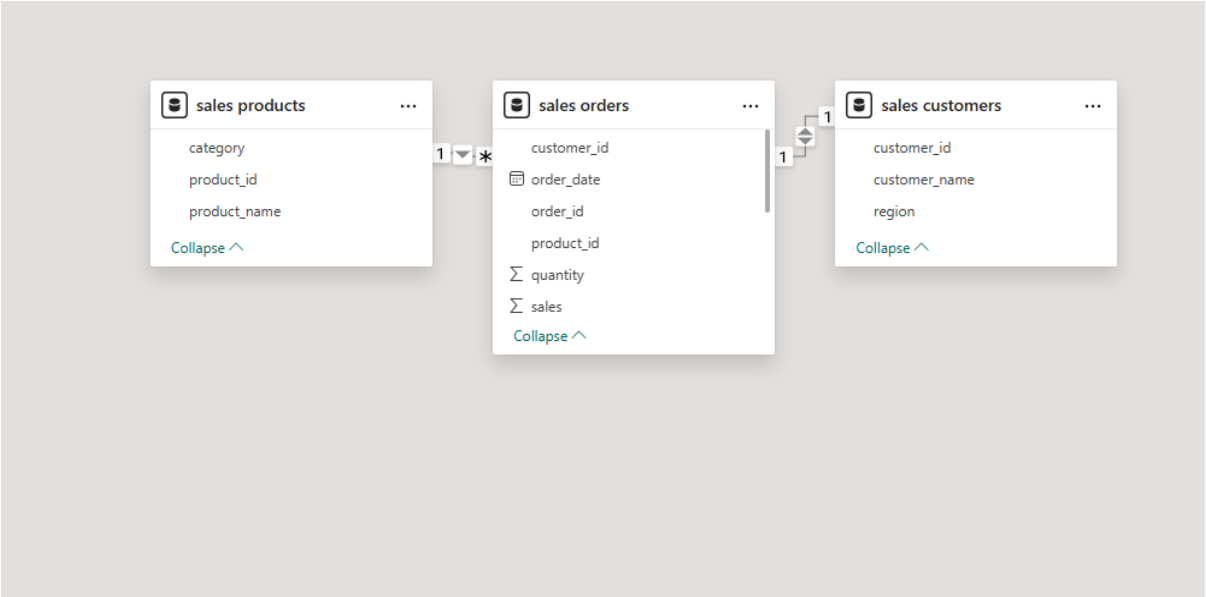
- Power BI Desktop is connected to the MySQL database using database credentials.
- Required tables are selected and loaded into Power BI.
- Data is imported successfully without errors.

DATA VALIDATION

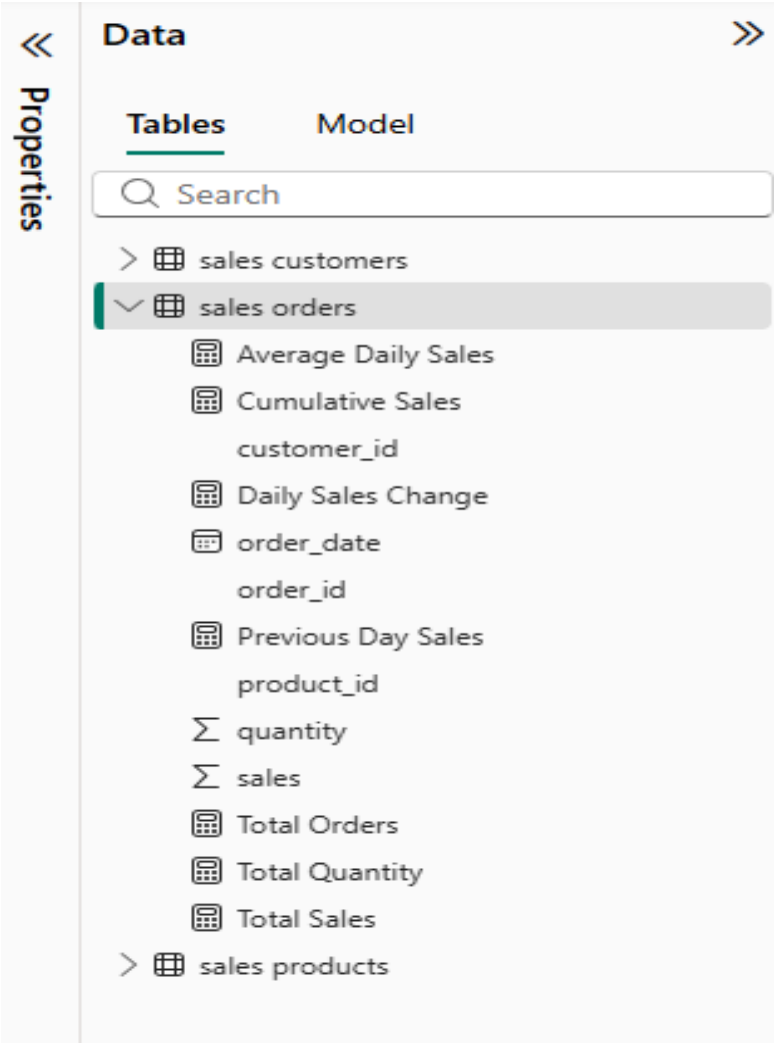
- Imported data matches the source MySQL tables.
- Row counts and column values are verified.
- No data loss or inconsistency observed.



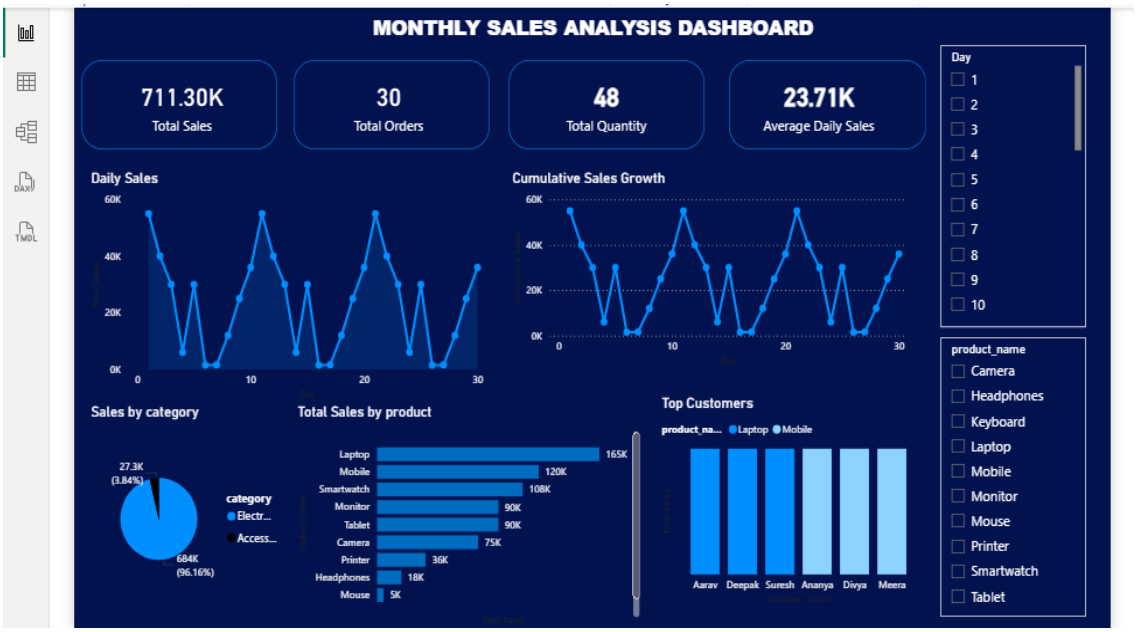
- Creating relationships in Model View



- Writing DAX measures (Total Sales, Daily Sales)



- Designing dashboard for business users



---

7. Key Insights

- Daily sales trends highlight peak and low-performance days
- Top 5 customers contribute a significant share of total revenue
- Certain products consistently outperform others

---

8. Conclusion

This project demonstrates a complete end-to-end data analysis workflow using multiple tools. It shows how raw sales data can be transformed into actionable insights for business decision-making.

---

9. Future Scope

- Sales forecasting
- Automated data refresh
- Real-time dashboard integration
