Retail Sales & Customer Insights Report

-- Sayan Das

1. Introduction & Project Overview

This report summarizes our end-to-end process for creating a **Retail Sales & Customer Insights Dashboard**. The goal was to integrate multiple data sources—Sales (MySQL/CSV), Customers (NoSQL/JSON), and Products (SharePoint/CSV)—into a **Data Warehouse**, clean and transform the data, perform **SQL/Python-based** data analysis, and finally build an interactive **Power BI dashboard**.

Key Business Challenges (from the project requirements) included:

1. Identifying top-performing products.
2. Understanding purchasing patterns (e.g., seasonality, promotions).
3. Visualizing regional sales trends and optimizing marketing strategies.

---

2. Data Sources & Requirements

We worked with three main data files:

1. **Sales Data** (sales 1.csv):
    - Contains columns like SaleID, CustomerID, ProductID, SalesAmount, Quantity, Timestamp.
    - Some rows had missing values or invalid references to Customer or Product.
2. **Customer Data** (customers.json):
    - Includes CustomerID, FirstName, LastName, Gender, Region, SSN.
    - Required cleaning for inconsistent region names, gender labels, null values, and duplicate IDs.
3. **Product Data** (products.csv):
    - Contains ProductID, ProductName, Category.
    - Used to validate product references in the Sales table.

The **requirement PDF** outlined the phases:

1. **Project Initiation & Requirement Analysis**
2. **Data Warehouse Design & ETL**
3. **Data Analysis & SQL**
4. **Power BI Dashboard Development**

---

3. Data Cleaning & Transformation (Python Notebook)

We used Python (with **pandas**) to handle initial data cleaning. Below is an illustrative snippet showing some of the key steps:

```
import pandas as pd

# 1. Load Data
# ----------------------------
sales_df = pd.read_csv('sales 1.csv')
```

```
customers_df = pd.read_json('customers.json')
products_df = pd.read_csv('products.csv')

# 2. Remove Invalid Sales Rows
# ----------------------------
# Drop rows with no SalesAmount or Quantity
initial_count = sales_df.shape[0]
sales_df.dropna(subset=['SalesAmount', 'Quantity'], inplace=True)
final_count = sales_df.shape[0]
print("Rows before cleaning:", initial_count)
print("Rows after cleaning:", final_count)

# Filter out rows whose CustomerID or ProductID doesn't exist
print("Sales rows before filtering invalid IDs:", sales_df.shape[0])
sales_df = sales_df[
    (sales_df['CustomerID'].isin(customers_df['CustomerID'])) &
    (sales_df['ProductID'].isin(products_df['ProductID']))
]
print("Sales rows after filtering invalid IDs:", sales_df.shape[0])

# 3. Clean Customers Table
# ----------------------------
# Print initial customer count
initial_customers = len(customers_df)
print("Number of customers before cleaning:", initial_customers)

# Standardize gender labels
customers_df['Gender'] = customers_df['Gender'].replace({
    'M': 'Male', 'male': 'Male', 'F': 'Female', 'female': 'Female'
})

# Fix region names and fill null with "Unknown"
customers_df['Region'] = customers_df['Region'].replace({
    'Texaz': 'Texas',
    'Ohho': 'Ohio',
    'New Yorkk': 'New York',
    'NY': 'New York',
    'Nw York': 'New York',
    'california': 'California',
    'Californiya': 'California'
})
customers_df['Region'] = customers_df['Region'].fillna("Unknown")

# Replace null LastName with empty string
customers_df['LastName'] = customers_df['LastName'].fillna("Unknown").replace("Unknown",
"")

# Remove duplicate CustomerIDs, keep first
customers_df.drop_duplicates(subset=['CustomerID'], keep='first', inplace=True)

# Drop SSN column
if 'SSN' in customers_df.columns:
    customers_df.drop(columns=['SSN'], inplace=True)
```

```
# Merge FirstName & LastName into a single column
customers_df['Name'] = (customers_df['FirstName'].astype(str) + " " +
                        customers_df['LastName'].astype(str)).str.strip()

# Print final customer count
final_customers = len(customers_df)
print("Number of customers after cleaning:", final_customers)

# 4. Validate Timestamp & SaleID
# ----------------------------
# Convert Timestamp to datetime
sales_df['Timestamp'] = pd.to_datetime(sales_df['Timestamp'], errors='coerce')
invalid_timestamp_count = sales_df['Timestamp'].isna().sum()
print("Number of rows with invalid Timestamp:", invalid_timestamp_count)

# Check SaleID uniqueness
duplicate_saleid_count = sales_df['SaleID'].duplicated().sum()
print("Number of duplicate SaleID entries:", duplicate_saleid_count)
```

Key Observations:

- We removed rows lacking SalesAmount or Quantity.
- We filtered out invalid foreign keys (CustomerID, ProductID).
- We standardized inconsistent region names (Texaz → Texas, Ohho → Ohio, etc.).
- We fixed gender labels (M/male → Male, F/female → Female).
- We handled null LastName values and removed duplicate CustomerIDs.
- We confirmed SaleID was unique and validated timestamps.

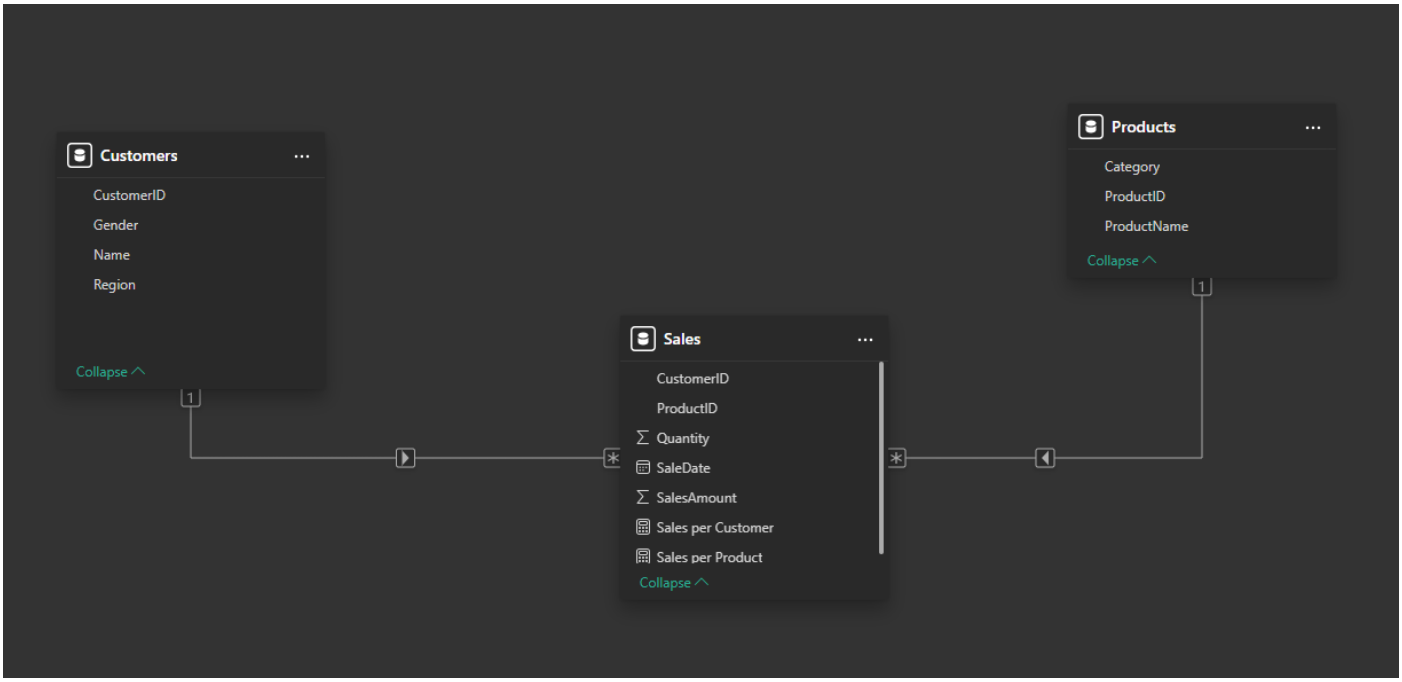
---

4. Data Warehouse Loading

After cleaning, we **loaded** the transformed tables into a **MySQL** data warehouse. Below is a snippet using **SQLAlchemy** and **PyMySQL**:

```
from sqlalchemy import create_engine

username = 'root'
password = '12345'
host = 'localhost'
port = '3306'
database = 'case2'

# Create MySQL engine
engine = create_engine(f'mysql+pymysql://{username}:{password}@{host}:{port}/{database}')

# Load fact and dimension tables
sales_df.to_sql('fact_sales', engine, if_exists='replace', index=False)
customers_df.to_sql('dim_customers', engine, if_exists='replace', index=False)
products_df.to_sql('dim_products', engine, if_exists='replace', index=False)

print("Data successfully loaded into MySQL data warehouse.")
```

Schema:



---

5. Data Analysis & KPIs

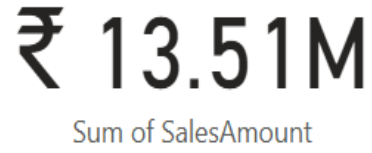
SQL Queries & KPIs

1. **Total Sales Revenue**

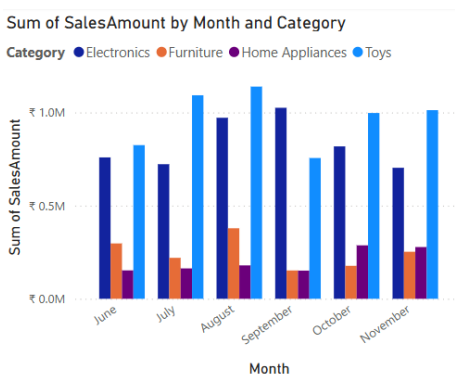
```
63
64 •    SELECT SUM(SalesAmount) AS TotalSales FROM fact_sales;
65
```

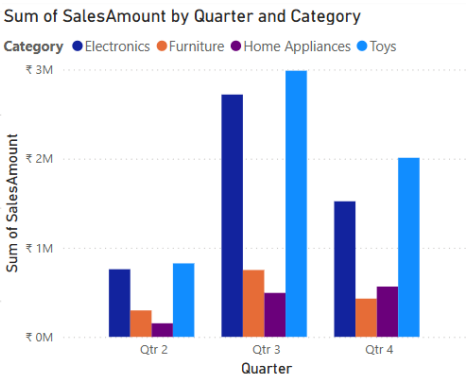
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

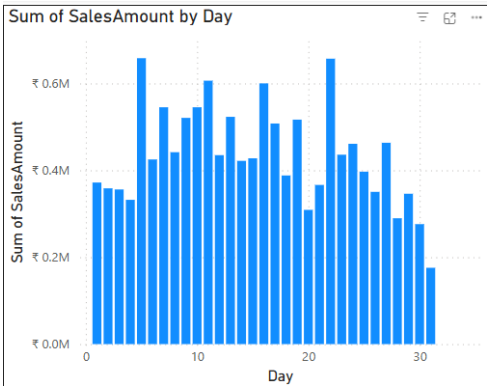
TotalSales
13507295.51999993



2. **Sales Growth Rate**







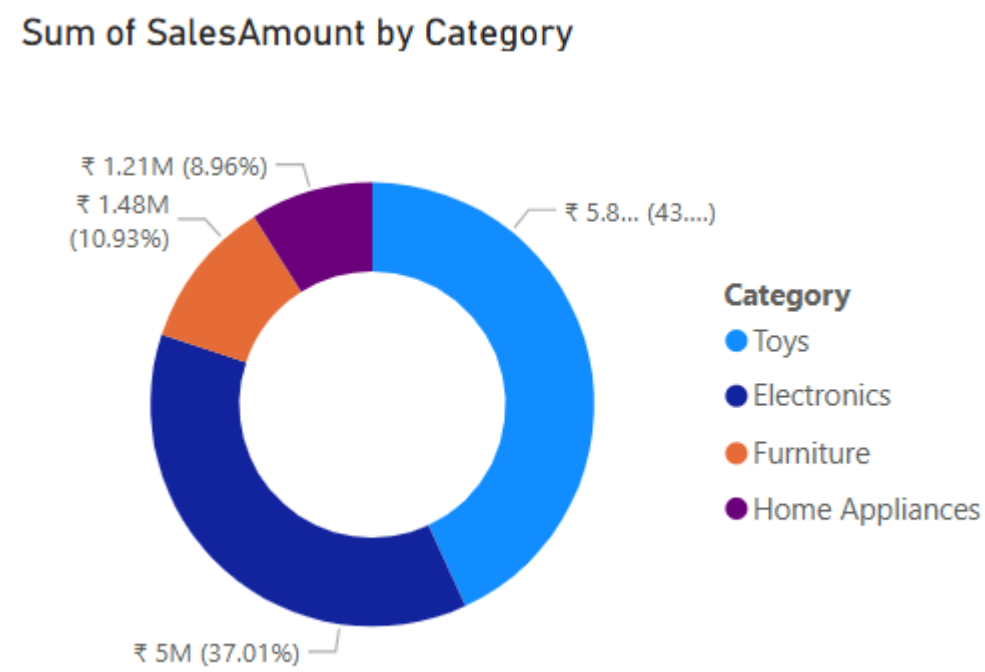
3. **Average Transaction Value (ATV)**

```
63
64 •   SELECT AVG(SalesAmount) AS AverageTransactionValue
65     FROM fact_sales;
66
```

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

AverageTransactionValue
1437.4050782164447

4. Sales by Product Category

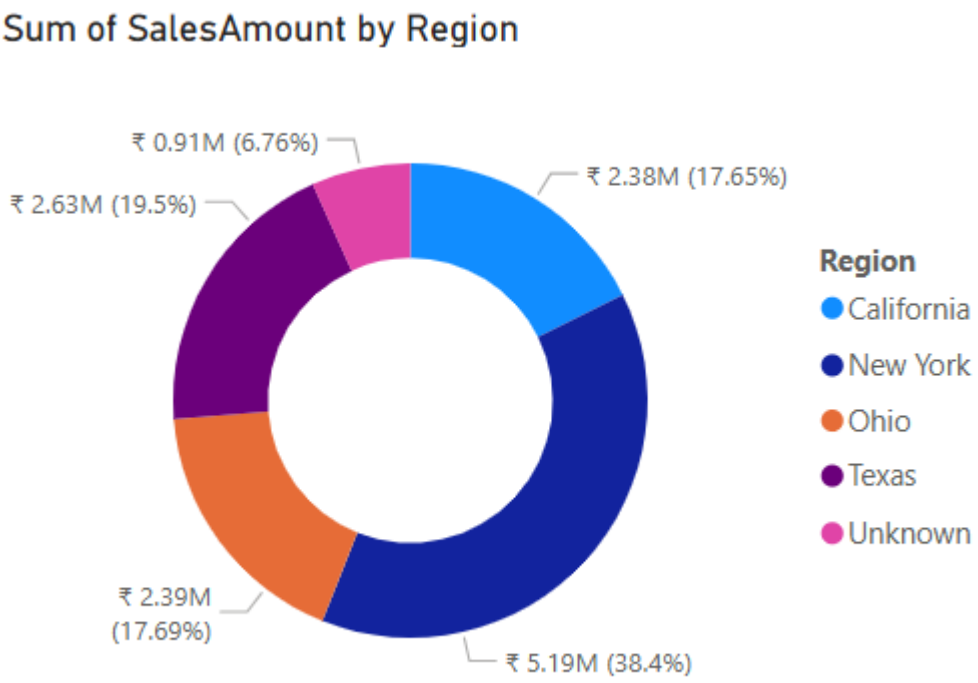


5. Sales by Region

```
29 •   SELECT c.Region, SUM(f.SalesAmount) AS RegionSales
30     FROM fact_sales f
31     JOIN dim_customers c ON f.CustomerID = c.CustomerID
32     GROUP BY c.Region
33     ORDER BY RegionSales DESC;
34
35
```

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

Region	RegionSales
New York	5186468.490000006
Texas	2633863.36
Ohio	2389636.3500000057
California	2383982.5700000008
Unknown	913344.7499999998



6. Customer Lifetime Value (CLV)

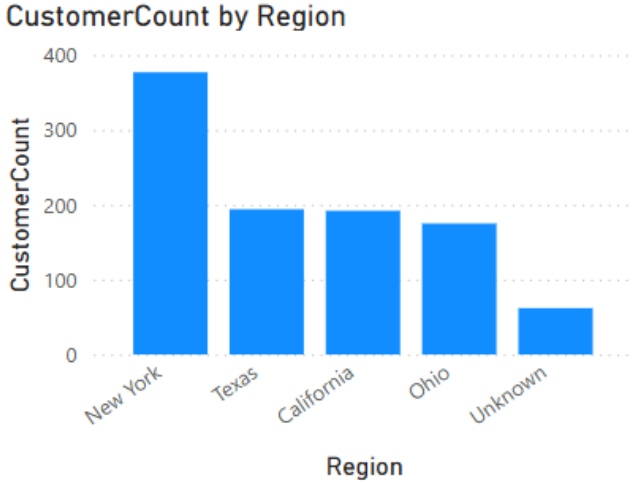
```sql
SELECT
  c.CustomerID,
  c.FirstName,
  c.LastName,
  COUNT(f.SaleID) AS NumPurchases,
  SUM(f.SalesAmount) AS TotalSpent
FROM fact_sales f
JOIN dim_customers c ON f.CustomerID = c.CustomerID
GROUP BY c.CustomerID, c.FirstName, c.LastName
ORDER BY TotalSpent DESC;
```

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

CustomerID	FirstName	LastName	NumPurchases	TotalSpent
C0381	Roy	West	9	108682.70999999998
C0051	David		21	102484.04000000001
C0435	Robert		21	99826.70999999999
C0555	Alec		22	96522.37
C0983	Daniel		14	94351.40999999999
C0909	Steven	Hamilton	18	92069.23000000003
C0965	Scott		26	87622.92
C0269	Madison		13	87070.57
C0864	Eric	Wilson	13	83374.08
C0349	John	Mcintosh	18	83333.39999999998
C0162	James	Williams	9	82935.33
C0278	Seth		8	82016.87000000001
C0422	Shelby		2	80925.64
C0570	Lee	Mayo	10	80119.58
C0089	Anthony	Lin	10	79410.07999999999
C0426	Brandon	Hernandez	15	78942.84000000001
C0957	Jason	Williams	5	78795.69999999998
C0259	Samantha	Mays	10	77642.01000000001
C0793	Jeffrey	Smith	7	77592.28
C0640	John		21	77340.01999999999
C0672	Daniel	Fernandez	17	76285.97000000002
C0084	Michael	Brooks	12	75515.2
C0352	Sean	Simpson	6	73420.07
C0122	Dustin	Cook	12	72681.25
C0393	Brandon	Ruiz	21	71664.87000000002
C0429	Mary	Buckley	10	71361.73999999999
C0724	Christine	White	9	71360.12999999999
C0380	Steven	Brown	18	71098.90000000002
C0646	Andrew		7	70885.34000000001
C0847	Aaron	Peterson	7	70737.80000000002
C0745	Chris	Barker	8	69222.72000000002
C0871	Daniel	Curry	11	69034.14

Result 10

7. Customer Demographics Analysis.



8. **Top-Selling Products**

```
SELECT
  p.ProductID,
  p.ProductName,
  SUM(f.SalesAmount) AS TotalSales,
  SUM(f.Quantity) AS TotalQuantity
FROM fact_sales f
JOIN dim_products p ON f.ProductID = p.ProductID
GROUP BY p.ProductID, p.ProductName
ORDER BY TotalSales DESC;
```

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

ProductID	ProductName	TotalSales	TotalQuantity
1030	Action Figure	645659.24	4965
1047	Swing	400085.40999999986	5352
1031	Board Game	390311.2399999998	5395
1032	Doll	381042.20000000007	5360
1037	Stuffed Animal	376743.68000000005	4661
1006	Table	373000.63000000006	4828
1027	Drone	341527.8900000001	5483
1020	Keyboard	340035.4499999999	4674
1039	Kite	336206.28999999986	4605
1029	Smart Glasses	329954.60000000003	5202
1007	Chair	326135.3799999999	5481
1003	Microwave	325645.14999999997	4535
1010	Laptop	309155.2899999999	4726
1008	Bed	308985.3300000003	4967
1014	Camera	307980.78999999986	4710
1011	Smartphone	302545.4499999998	4506
1041	Toy Train	302386.79000000004	4725
1016	Bluetooth Spe...	297213.12000000017	4499
1033	Puzzle	285464.7200000001	5277
1026	Projector	284807.5200000001	4450
1043	Marbles	267684.84	4407
1004	Vacuum Cleaner	261941.0100000001	4772
1018	Monitor	261625.44000000003	4585
1009	Cupboard	260238.53999999998	5327
1045	Toy Kitchen Set	255326.88999999993	4954
1035	Toy Car	252957.36999999997	4883
1040	RC Car	251063.6599999999	4583
1034	Lego Set	249909.51999999996	4243
1046	Bicycle	245068.52000000008	5341
1015	Smartwatch	229443.60999999993	4394
1048	Educational G...	225368.6600000001	4795
1038	Play-Doh	220363.39	5149
1023	Smart Home ...	219330.22000000003	4079
1019	Printer	216697.48999999987	4087

9. **Product Return Rate**

Can not be determined as there are no data reguarding this.

10. **Regional Sales**

Cannot be determined as Cost Price is not given.

6. Power BI Dashboard Development

Data Preparation in Power BI

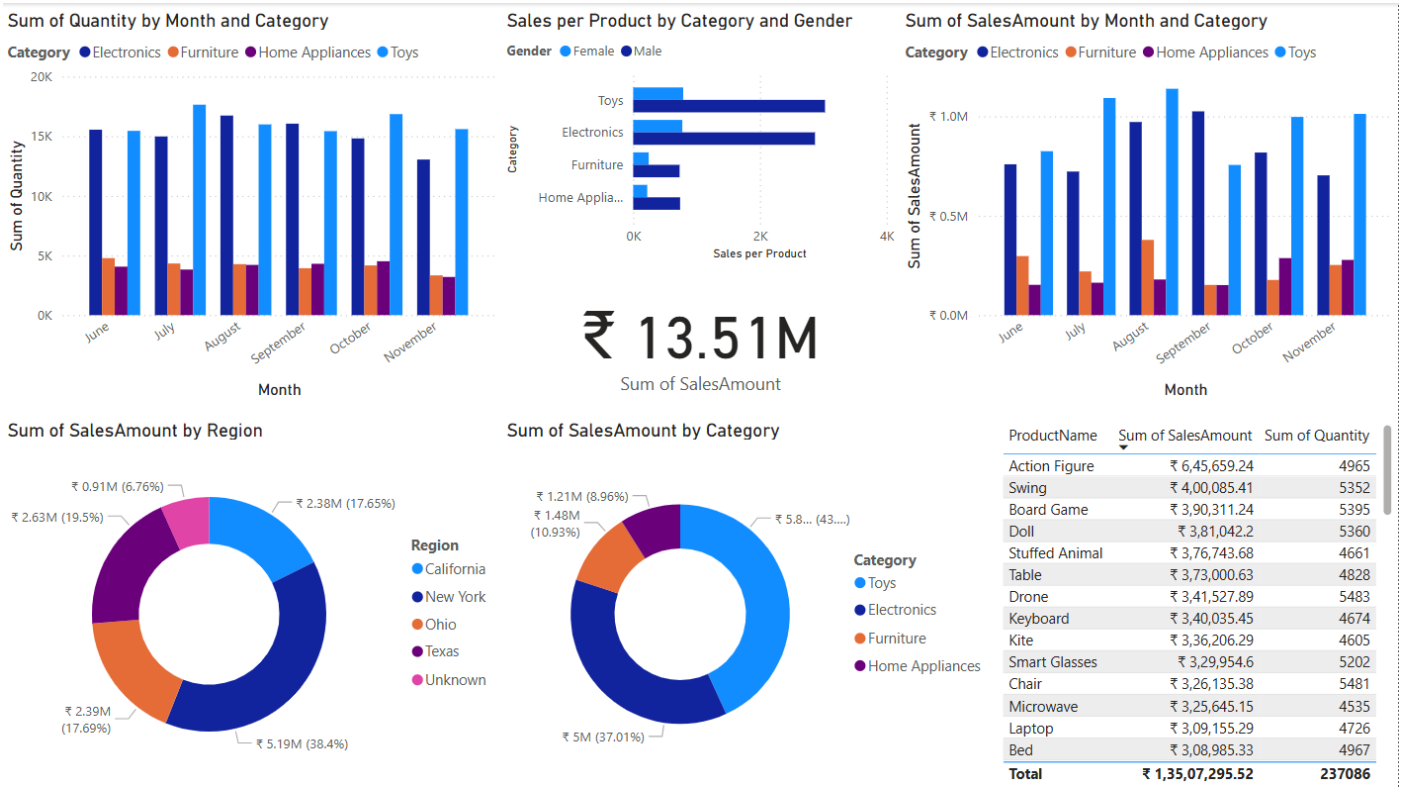
- We imported the fact and dimension tables from MySQL into Power BI.
- Used **Power Query** transformations to ensure final consistency (e.g., region name fixes, last name merges).

Data Modeling

- Created relationships between **fact_sales** (Fact) and **dim_customers**, **dim_products** (Dimensions).
- DAX measures for Sales per Customer, Sales per product, etc.
    - Sales per Customer = COUNT(Sales[CustomerID])
    - Sales per Product = COUNT(Sales[ProductID])

Visualizations (see the provided dashboard screenshot):

1. **Bar Charts**:
    - **Sum of Quantity by Month and Category** helps identify which categories peak in certain months.
    - **Sales per Product by Category and Gender** indicates how each product category performs across different genders.
2. **Cards & KPI**:
    - **Total Sales** Card (e.g., `₹ 13.51M`) is a quick at-a-glance measure of overall performance.
3. **Pie/Donut Charts**:
    - **SalesAmount by Region** reveals top regions (e.g., Ohio, Texas, California, New York).
    - **SalesAmount by Category** quickly shows the highest-earning product categories (e.g., Electronics, Furniture).
4. **Tables**:
    - Shows individual product performance (e.g., Action Figure, Board Game, Car, Laptop, etc.) with revenue and quantity sold.



ProductName	Sum of SalesAmount	Sum of Quantity
Action Figure	₹ 6,45,659.24	4965
Swing	₹ 4,00,085.41	5352
Board Game	₹ 3,90,311.24	5395
Doll	₹ 3,81,042.2	5360
Stuffed Animal	₹ 3,76,743.68	4661
Table	₹ 3,73,000.63	4828
Drone	₹ 3,41,527.89	5483
Keyboard	₹ 3,40,035.45	4674
Kite	₹ 3,36,206.29	4605
Smart Glasses	₹ 3,29,954.6	5202
Chair	₹ 3,26,135.38	5481
Microwave	₹ 3,25,645.15	4535
Laptop	₹ 3,09,155.29	4726
Bed	₹ 3,08,985.33	4967
Total	**₹ 1,35,07,295.52**	**237086**

---

7. Observations & Insights from the Dashboard

1. **Overall Sales**: The total revenue stands at around **₹13.51M**, indicating a substantial volume of transactions.
2. **Top Categories**:
    - **Electronics** and **Toys** appear as strong categories, driving a large share of revenue.
    - **Home Appliances** and **Furniture** also contribute but at slightly lower levels.
3. **Regional Breakdown**:
    - **New York** is leading in sales, each contributing roughly 18–19% of total revenue.
    - All other states follow closely behind.
4. **Monthly Trends**:
    - Certain months show higher quantity sold but no abnormal behaviour seen.
5. **Product Insights**:
    - Items like **Action Figure** and **Swing** show high revenue, suggesting they are top sellers.

These insights help **marketing teams** target top regions, **inventory managers** stock high-demand items, and **executives** track overall performance.

---

8. Conclusion & Recommendations

Conclusion:

- We successfully **cleaned** and **consolidated** disparate datasets into a consistent **star schema**.
- The **data warehouse** approach ensures a single source of truth for sales, customers, and products.
- Our **Power BI dashboard** provides interactive, real-time insights, enabling quick decision-making on product performance, regional strategies, and customer segmentation.

Recommendations:

1. Focus on top-earning regions (e.g., New York) with targeted campaigns.
2. Focus on high selling catagories like Toys and Electronics.
3. On Average men are buying more than women so targeted ads can help.