**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

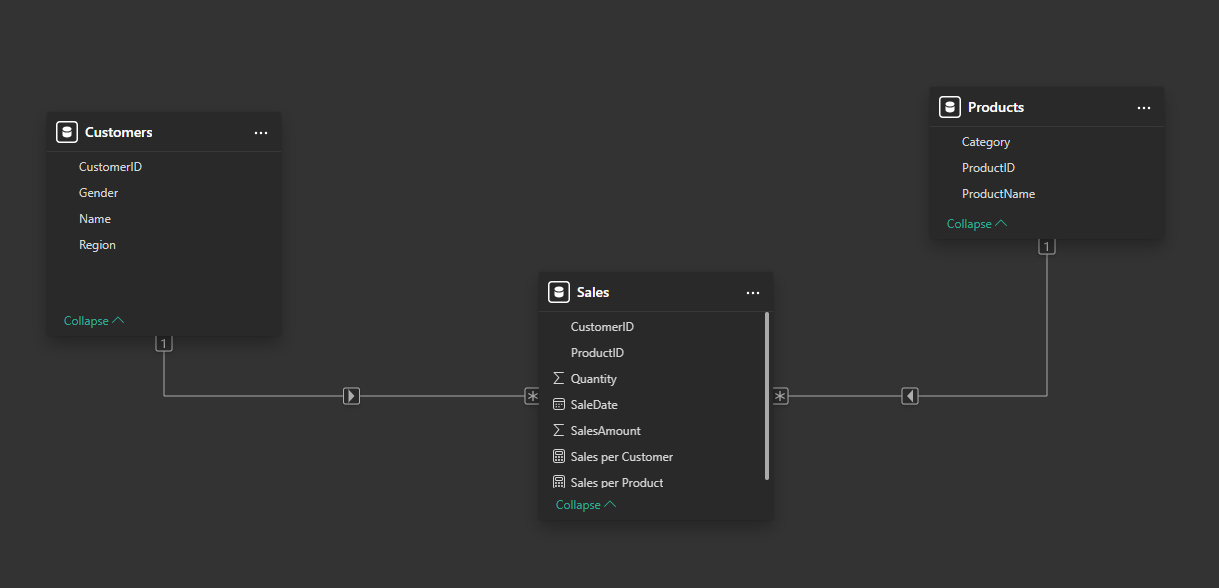
sales\_df.to\_sql('fact\_sales', engine, if\_exists='replace', index=False)

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.")

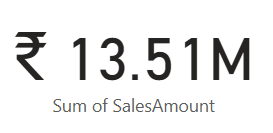
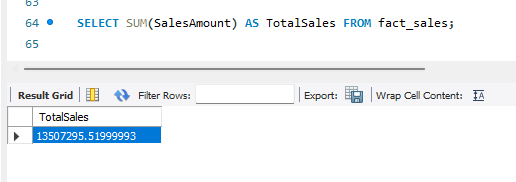
**Schema**:



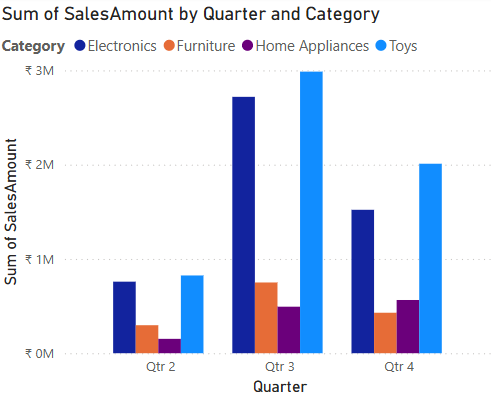
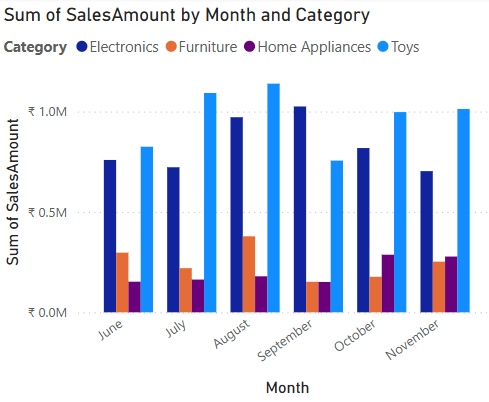
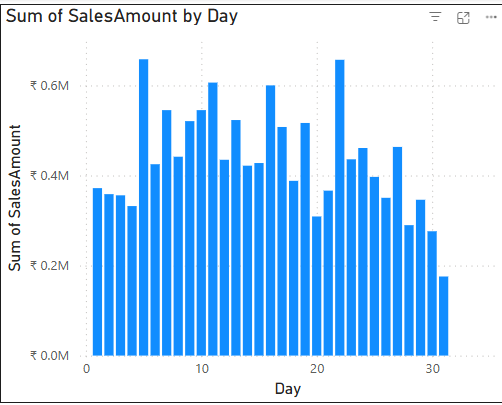
**5. Data Analysis & KPIs**

**SQL Queries & KPIs**

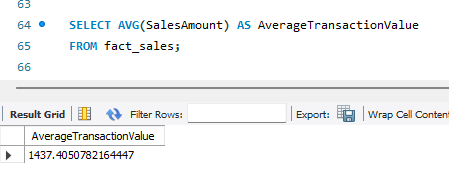
1. **Total Sales Revenue**



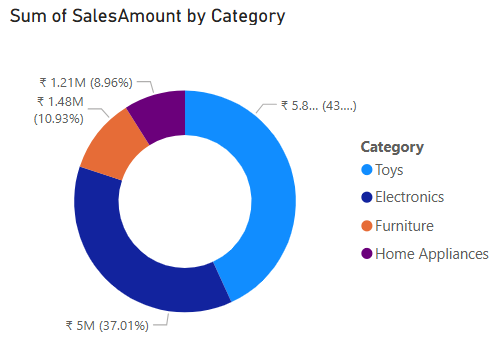
1. **Sales Growth Rate**

**** ****

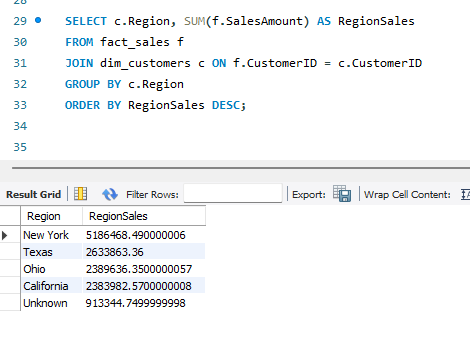
1. **Average Transaction Value (ATV)**

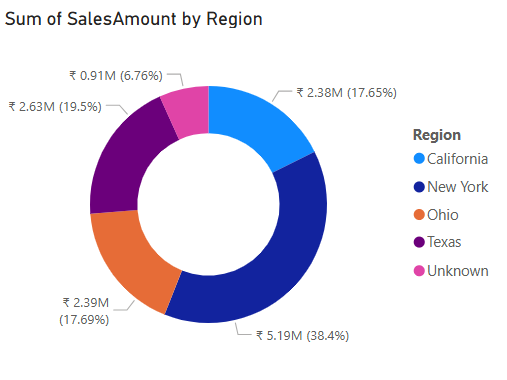
****

1. **Sales by Product Category**

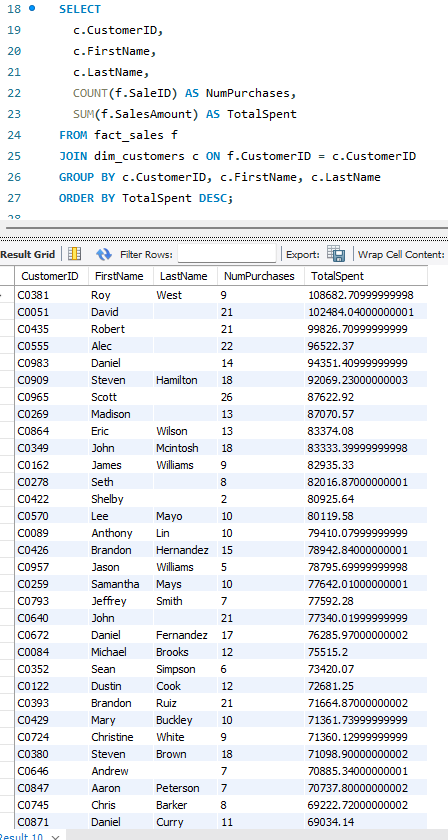


1. **Sales by Region**

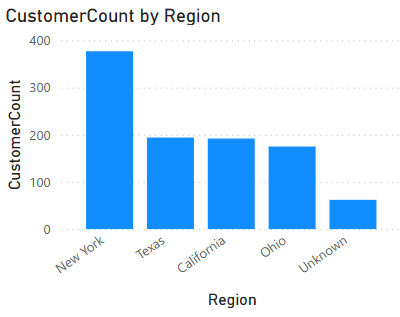
****

****

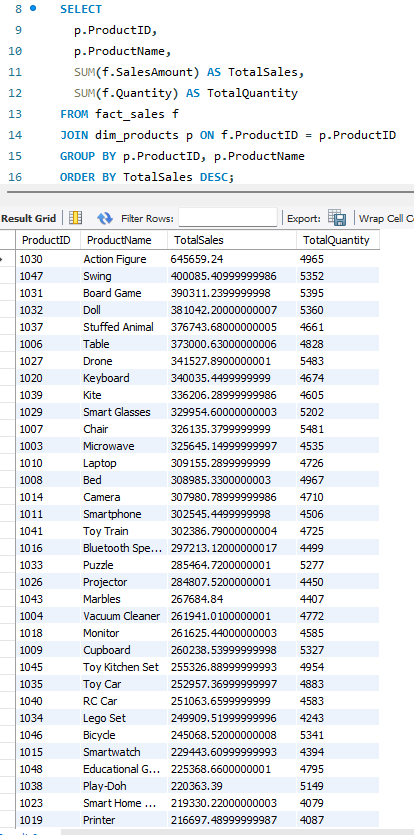
1. **Customer Lifetime Value (CLV)**

****

1. **Customer Demographics Analysis.**

****

1. **Top-Selling Products**

****

1. **Product Return Rate**

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

1. **Regional Sales**

Cannot be determined as Cost Price is not given.

**6. Power BI Dashboard Development**

**Data Preparation in Power BI**

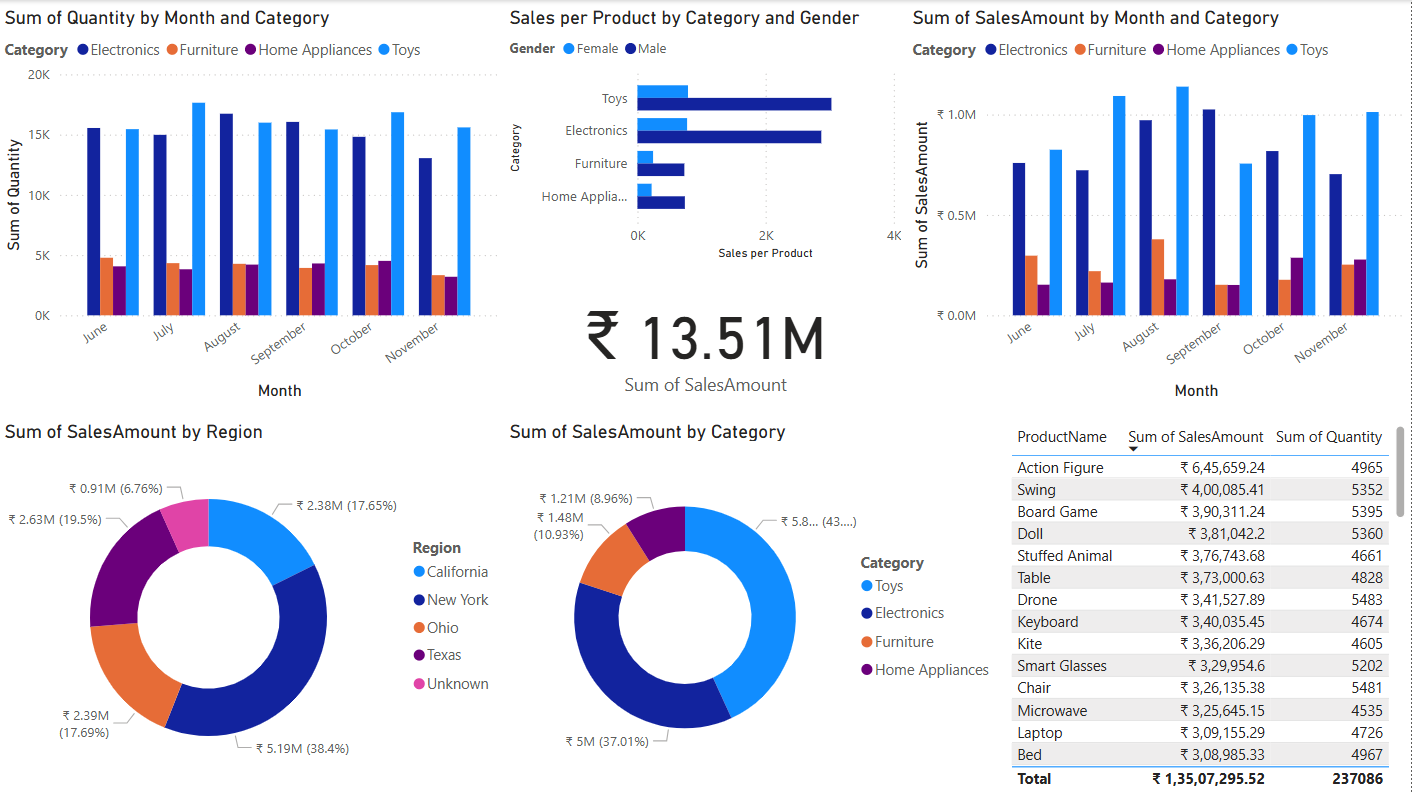
* We imported the fact and dimension tables from MySQL into Power BI.
* Used **Power Query** transformations to ensure final consistency (e.g., region name fixes, last name merges).

**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.



**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.