Final Project Report

Retail Sales Analytics Using Excel, SQL, Power BI and Python

Project work done by

Thaslima Banu. S

Course

Data Analytics

Introduction:

Retail Sales Analytics aims to analyze sales performance, customer behaviour, product trends, and staff contribution using Excel, SQL, Python and Power BI. The goal is to build an interactive dashboard that supports data-driven decisions for improving profitability and operational efficiency.

Phase 1: Excel-Data Cleaning & Preparation

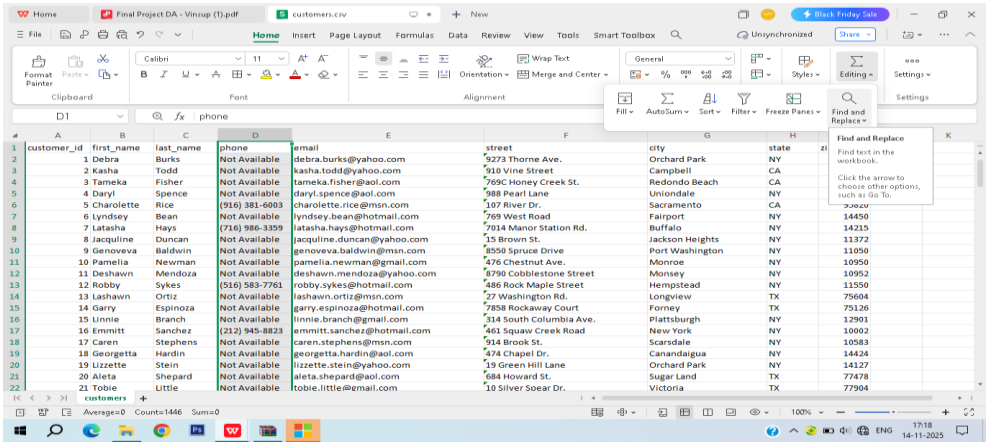
Q1. Standardize Column Headers – Replaces spaces with underscore.

Q2. Remove Duplicates: No Duplicate records were found while checking by – Data-Remove Duplicates.

Q3. Handle Missing Values

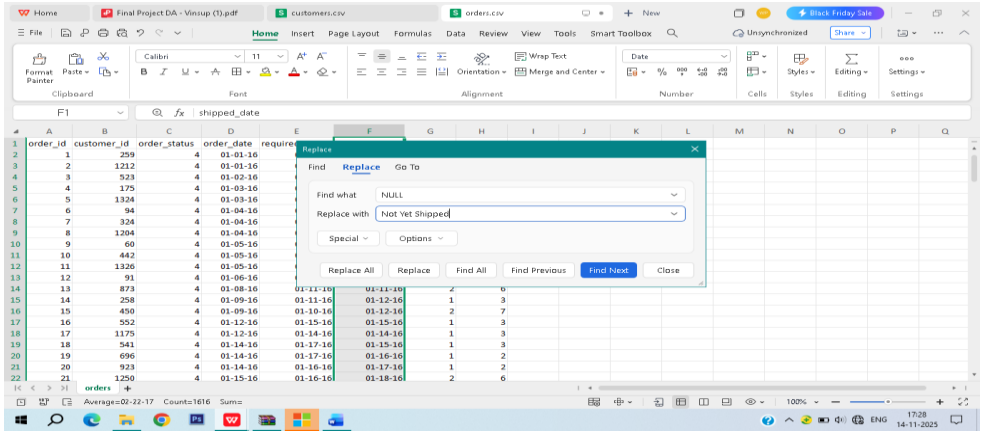
Customer Table:

1. Null values in phone column were replaced with 'Not Available' to maintain completeness while retaining customer records.
2. Replaced null values by selecting the column-Home-Find and Replace option-Find-NULL, Replace-Not Available.



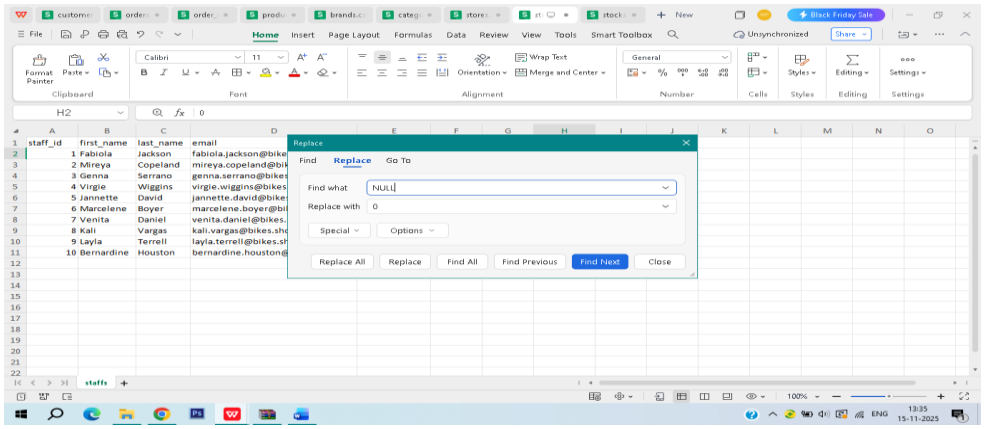
Orders Table:

3. In Ship_date – blank values are replaced with 'Not yet shipped'



Staffs Table:

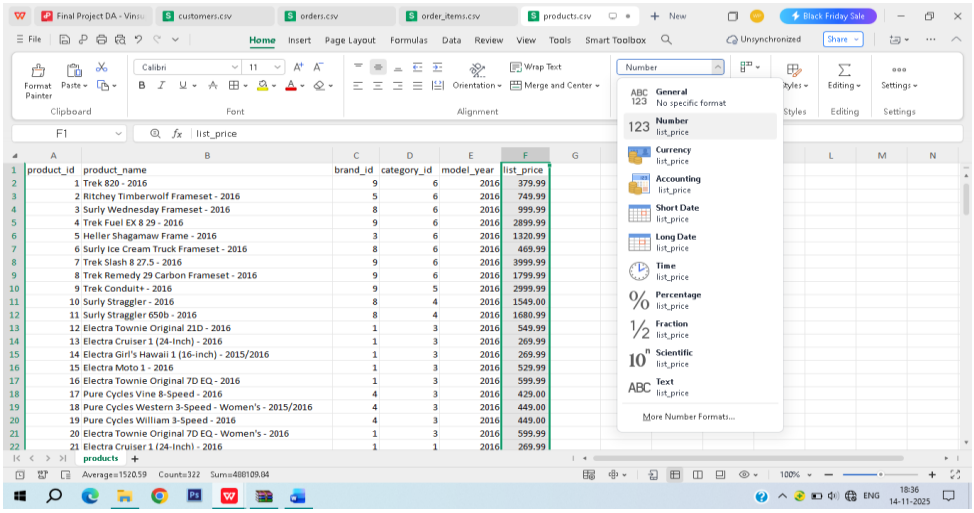
4. Replaced Null value in manager column with ‘0’.



Q4. Data Type Conversion.

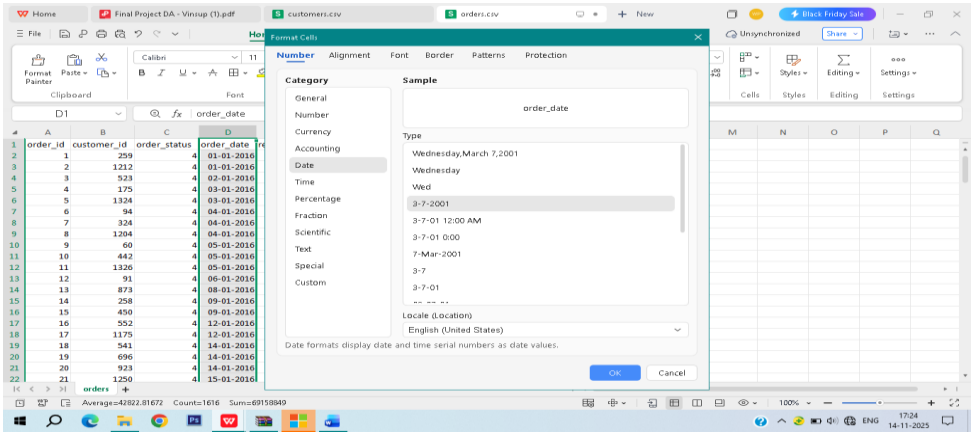
Products Table:

1. List_price changed to number



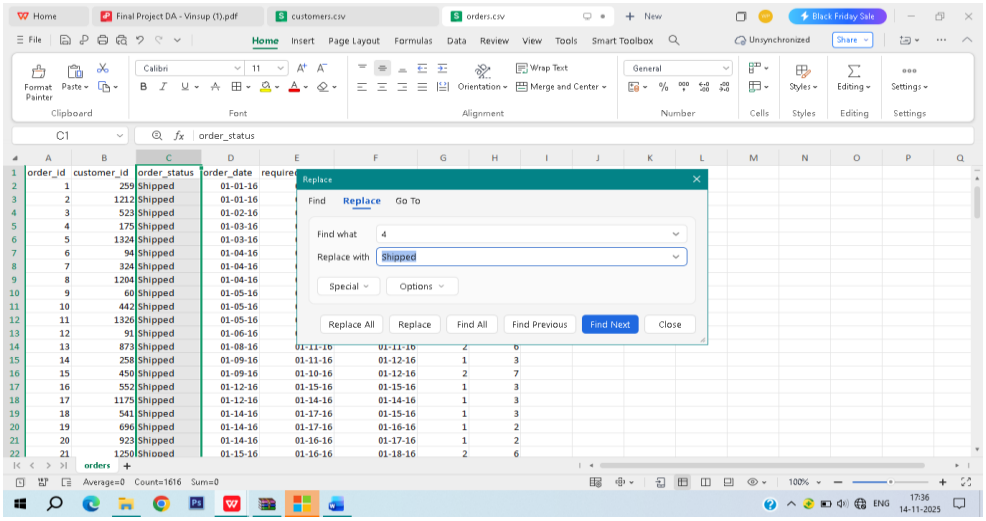
Orders Table:

2. List_price changed to number.
3. Quantity to Number
4. Standardized order_date, require_date, and shipped_date to Date Format by right clicking the column – Format cells – Date.

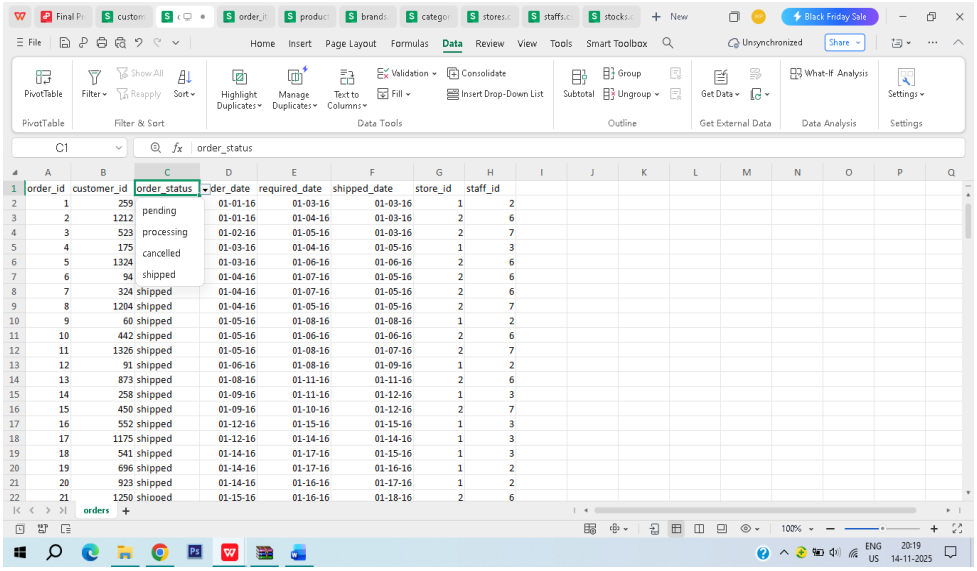


Q5. Data Validation

1. Standardized order_status values for consistency.
   i) 1 – Pending.
   ii) 2 – Processing.
   iii) 3 – Cancelled.
   iv) 4 – Shipped/Delivered.



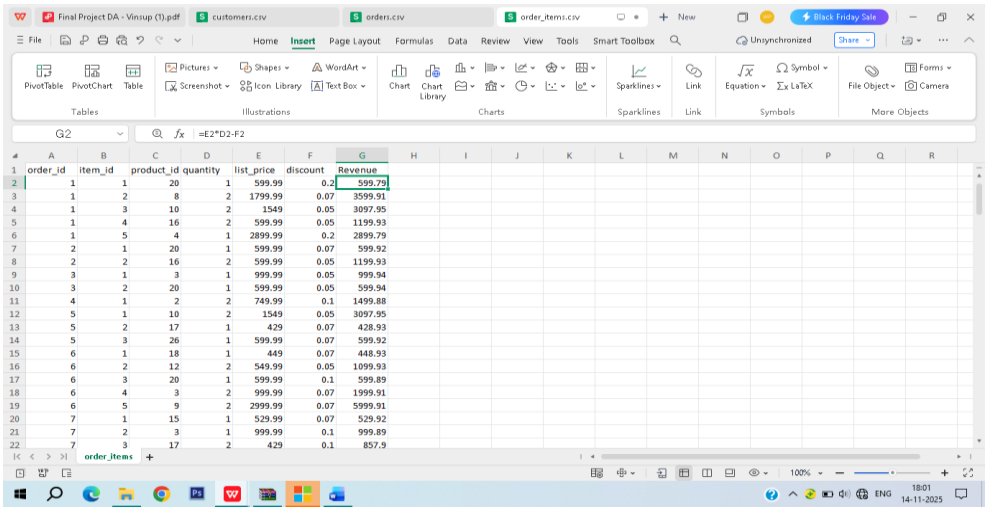
2. Select the order_status column- Data-Data validation-List.



Q6. Create Derived Column

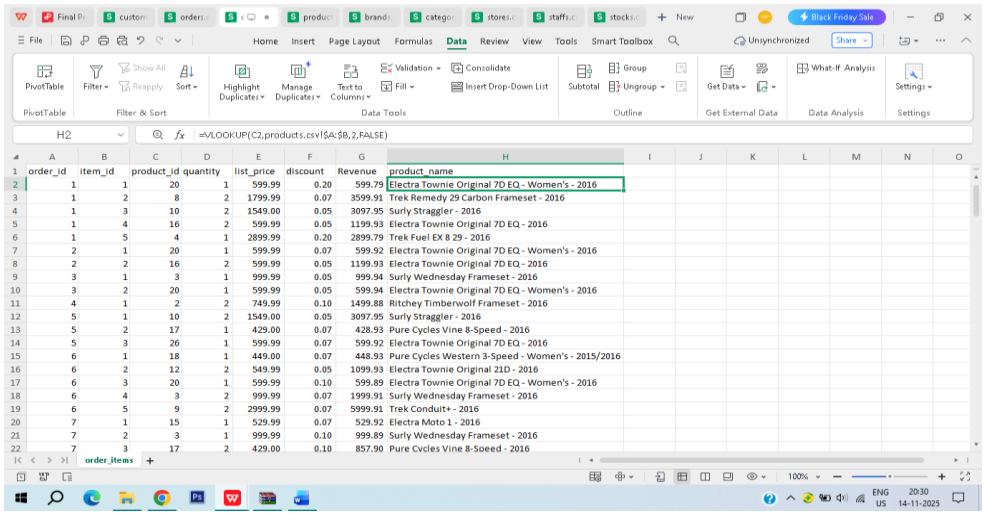
Order_Items Table:

1. Created Revenue Column
2. Revenue = List_price*Quantity-Discount.

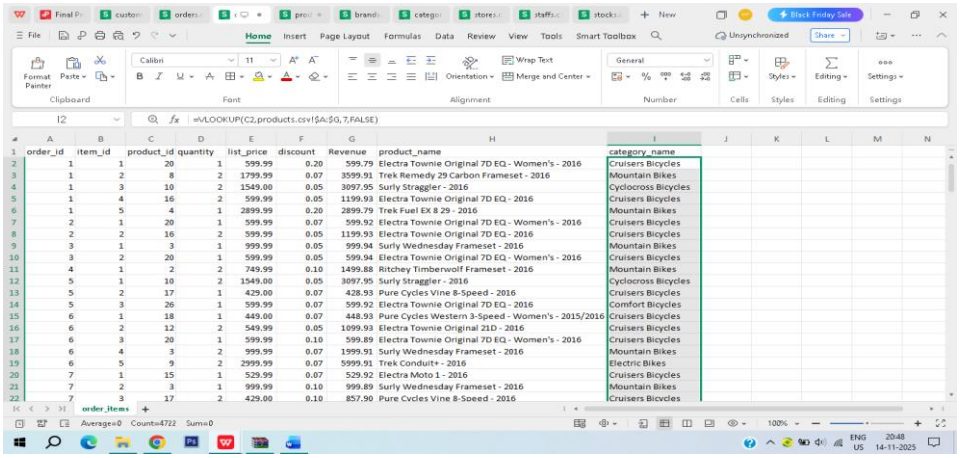


Q7. Merge Lookup Data

1. I used VLOOKUP in order_items, added new column product_name using product_id.
2. =VLOOKUP(C2,products.csv!$A:$B,2,FALSE)

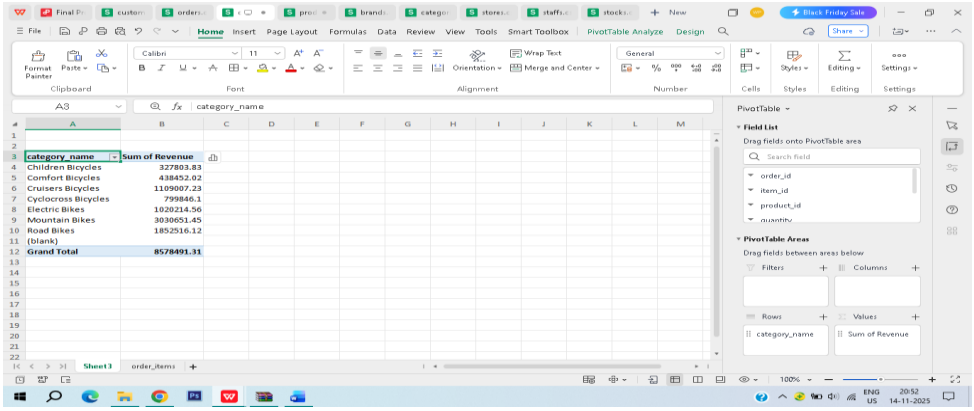


3. In products table added new column category_name using category_id .
4. In order_items table added new column category_name using product_id to create pivot table.



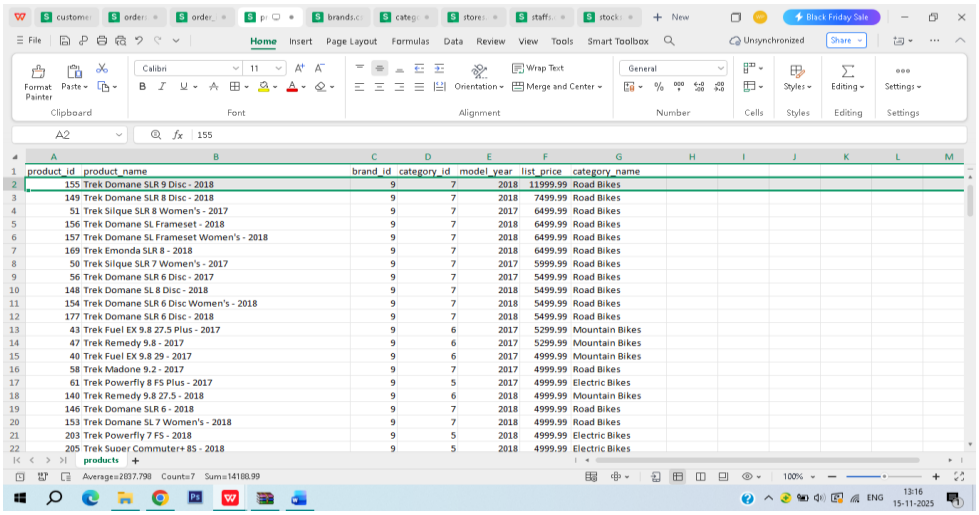
Q8. Create Pivot Table

1. In order_items table select all-insert-pivot table.
2. Rows: category_name
3. Values: Revenue
4. Renamed the pivot table and sheet name as 'Total sales by category'.



Q9. Sort and Filter Outliers

1. In products table select list_price column -sort-Largest to Smallest



2. To identify pricing outliers (much larger than majority values),
3. List_price values in the products table were sorted in descending order. The highest-priced products were Trek Domane SLR 9 Disc – 2018 with 11999.99 list_price, in Road Bikes category because they are significantly higher than the typical price range of most products.

Q10. Save Final CSVs

1. Click File
2. Select Save AS
3. Choose Other Formats
4. Save the file in CSV format wherever we want.

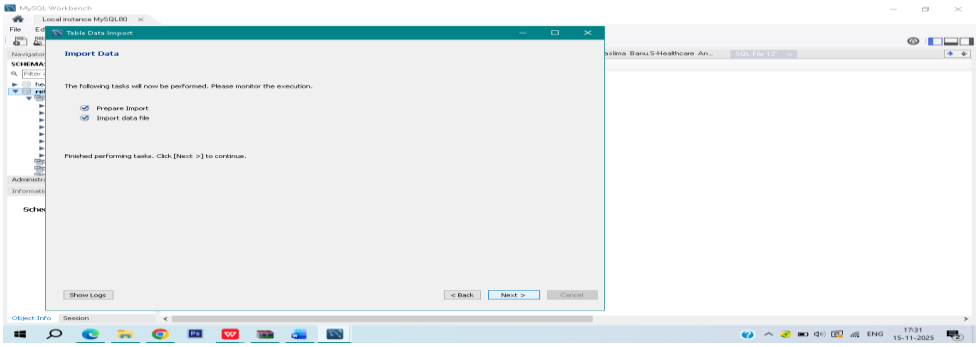
Phase 2: SQL-Database Management and Querying

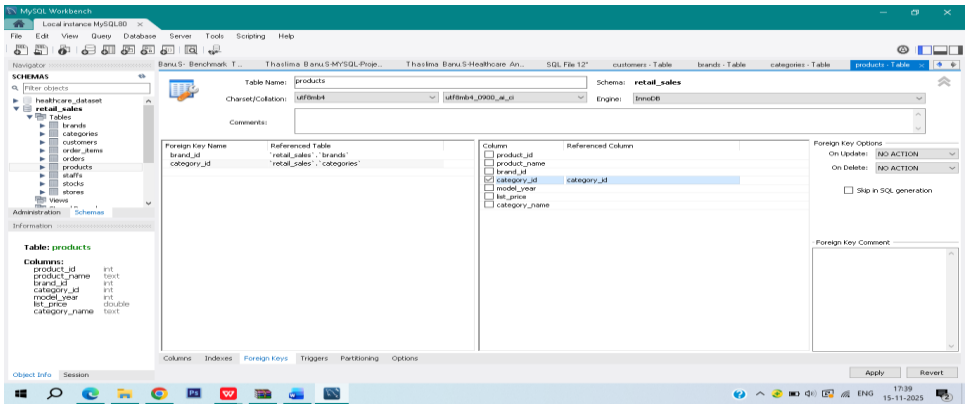
1. write the query on MySQL Workbench

```
CREATE database Retail_sales;
use retail_Sales;
```

2. Database was created – It will visible on the left side of the workbench under Schemas.

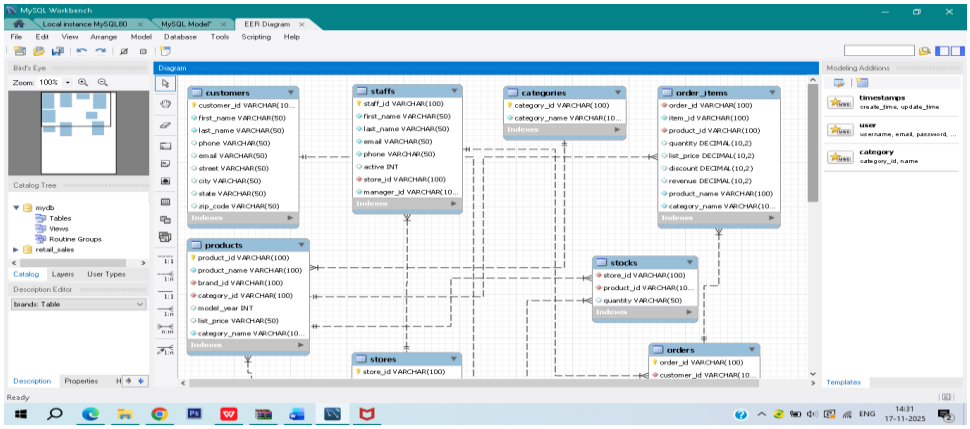
3. Right Click the database created – Table data import wizard – Click Browse – Select the saved csv file - Import tables



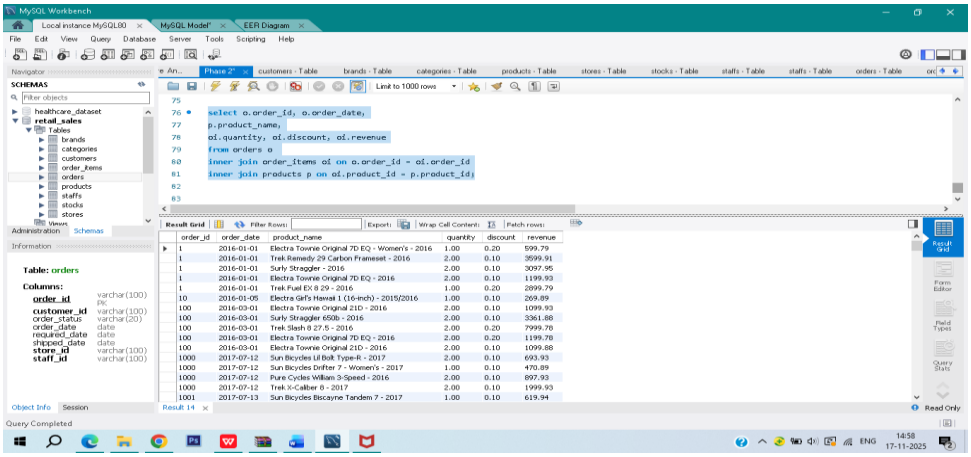


Q2.ER Diagram

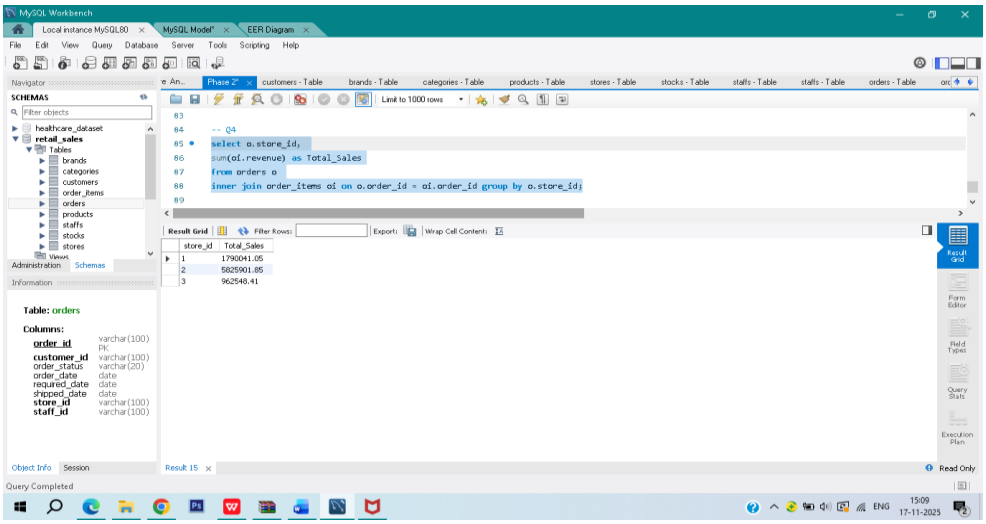
Relationship Flow	Cardinality	Rationale
Customers - Orders	One-to-many(1:*)	A customer can place many orders.
Staffs – Orders	One-to-many(1:*)	A staff member can process many orders.
Orders – OrderItems	One-to-many(1:*)	One order contains many individual items.
Products - OrderItems	One-to-many(1:*)	One product can be sold across many order items.
Brands - Products	One-to-many(1:*)	One brand can produce many products.
Categories - Products	One-to-One(1:1)	One category can contain many products.
Customer–Customer_Segments	One-to-One(1:1)	Each unique customer is assigned a single, specific segment.



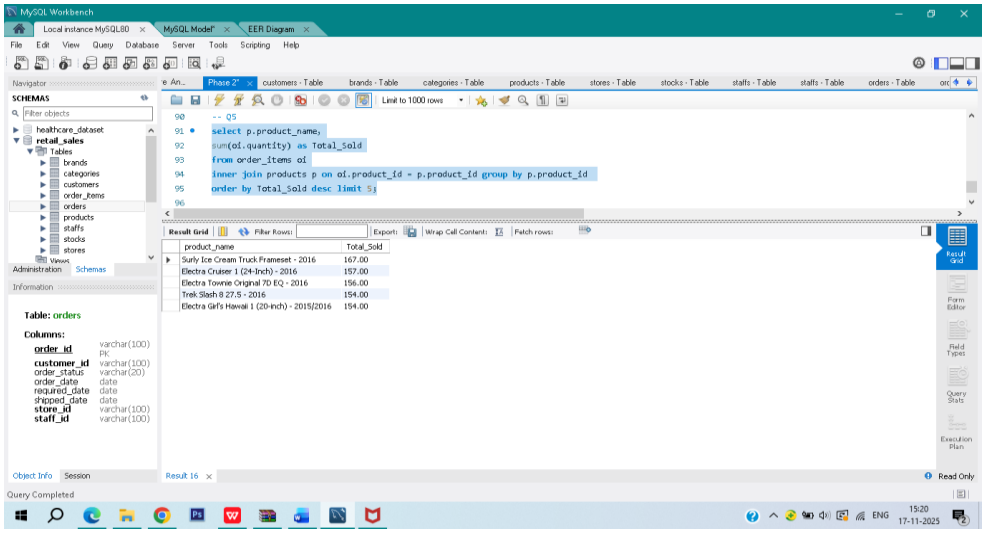
Q3. Inner Join for Order Details



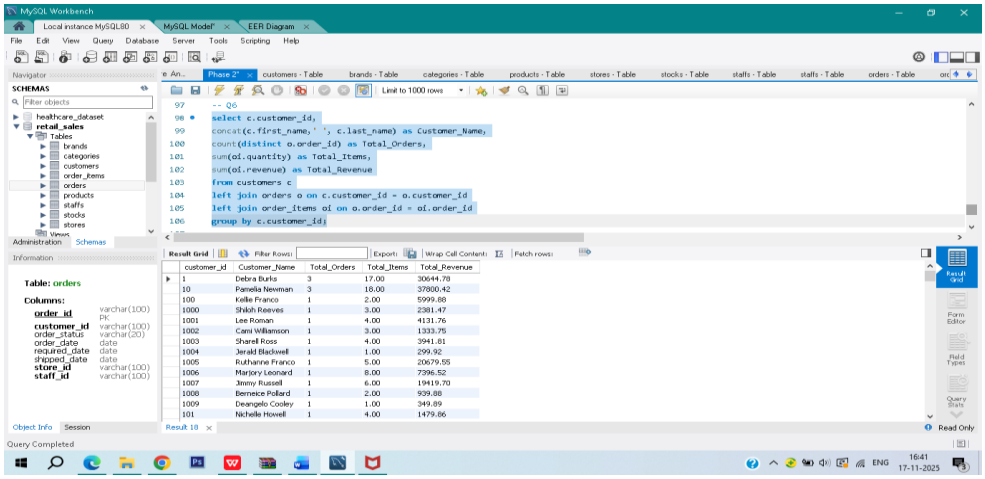
Q4. Total Sales by Store



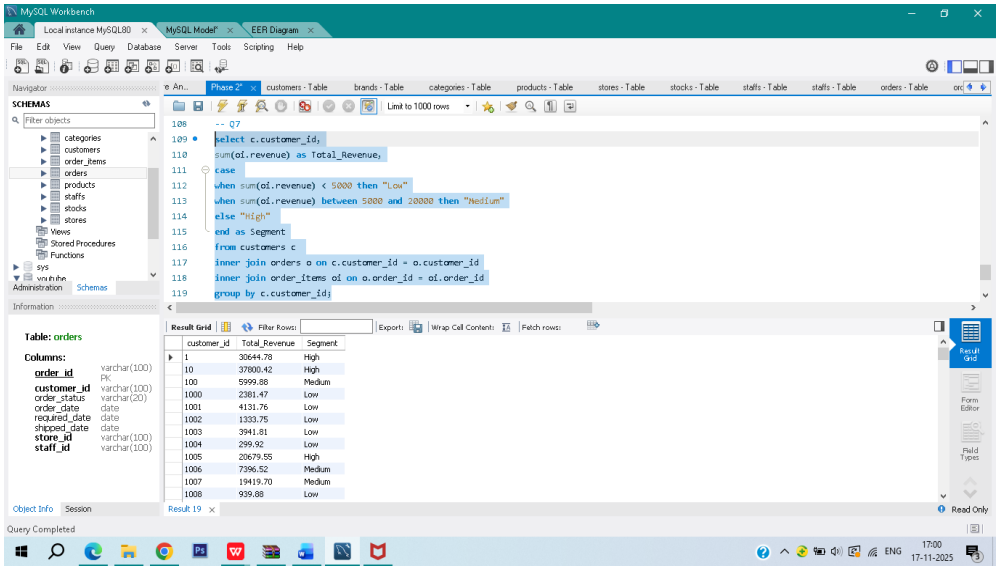
Q5. Top 5 Selling Products



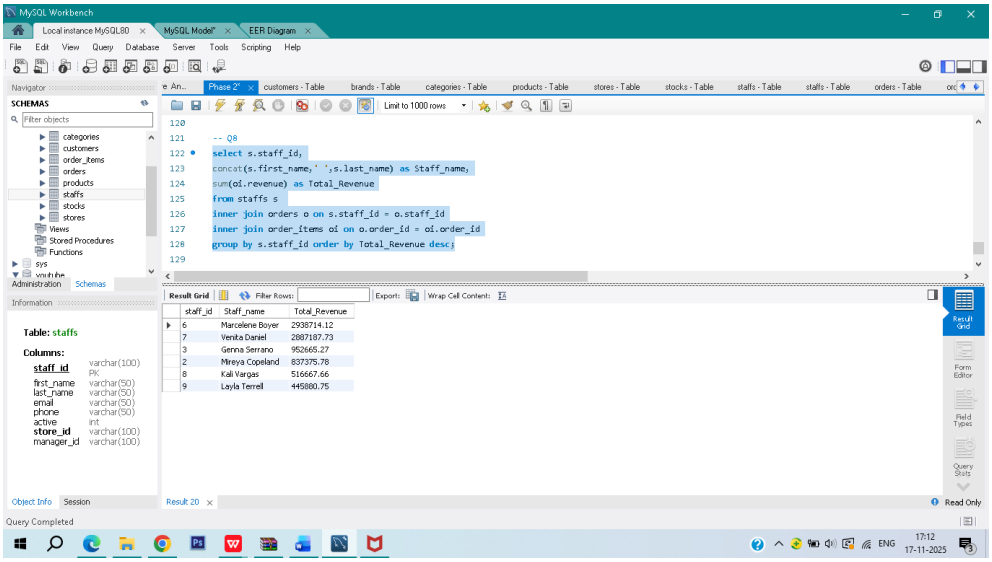
Q6. Customer Purchase Summary



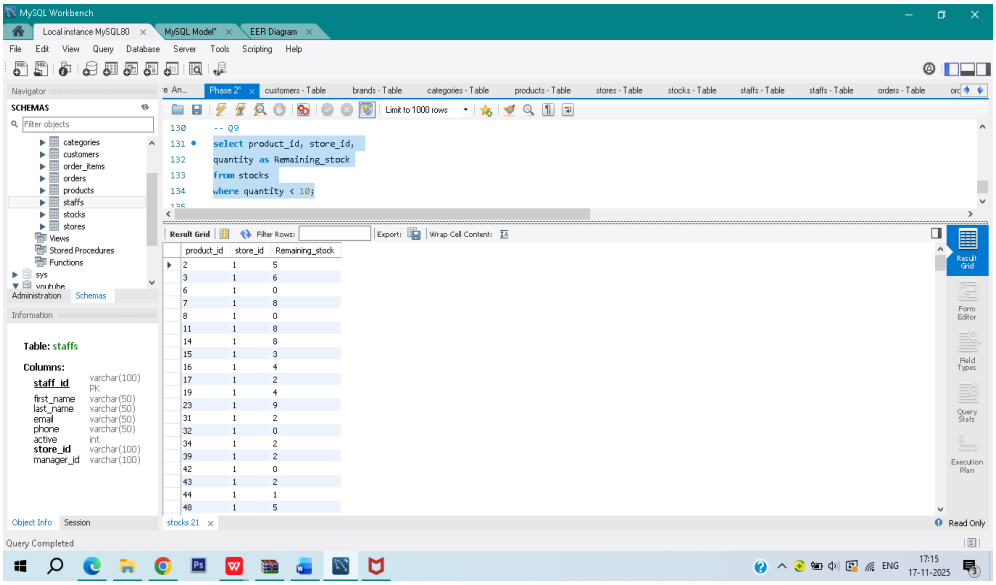
Q7. Segment Customers by Total Spend



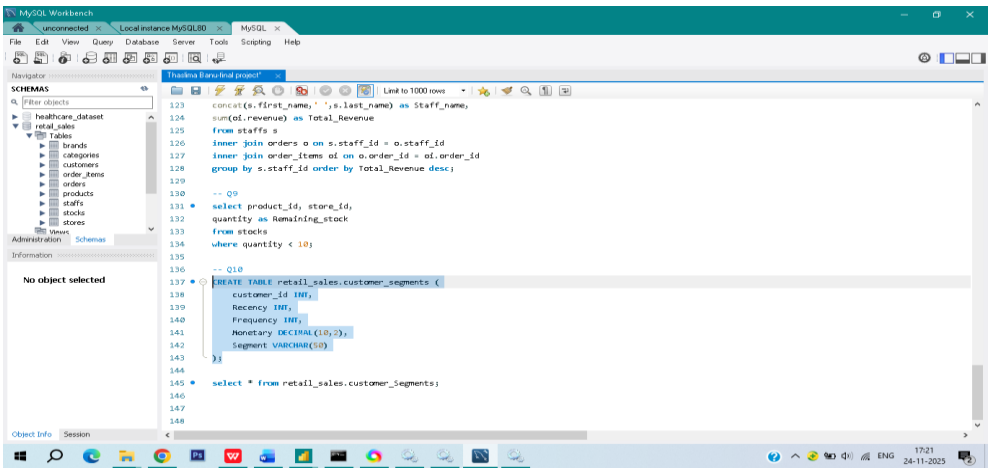
Q8. Staff Performance Analysis



Q9. Stock Alert Query



Q10. Final Segmentation Table



Phase 3: Python and ML Tasks

1. Open Command prompt.
2. Type command python -m notebook.
3. It will open Jupyter notebook.
4. Install pandas, numpy, matplotlib by using the word pip before the libraries.
5. Import pandas as pd, numpy as np, matplotlib.pyplot as plt.

Q1. Load Data from SQL

Steps:

1.pip install sqlalchemy

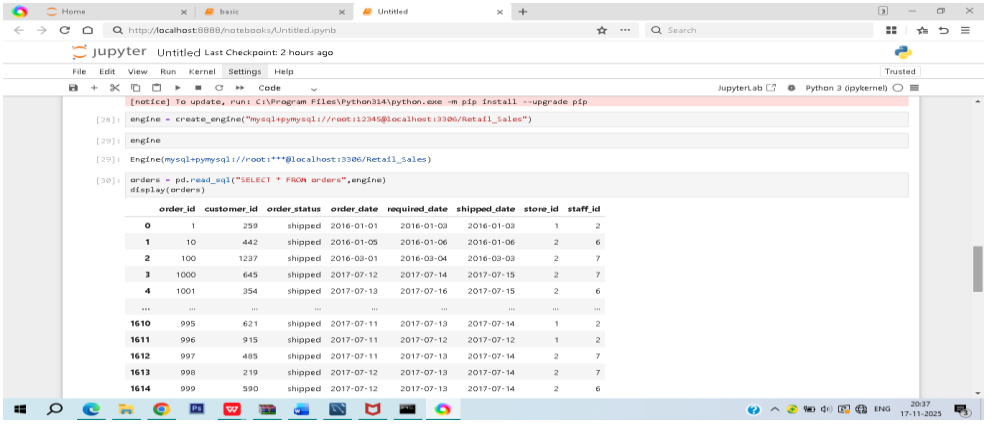
2.from sqlalchemy import create_engine

3.engine=create_engine("mysql+pymysql://root:*****@localhost:3306/Retail_Sales")

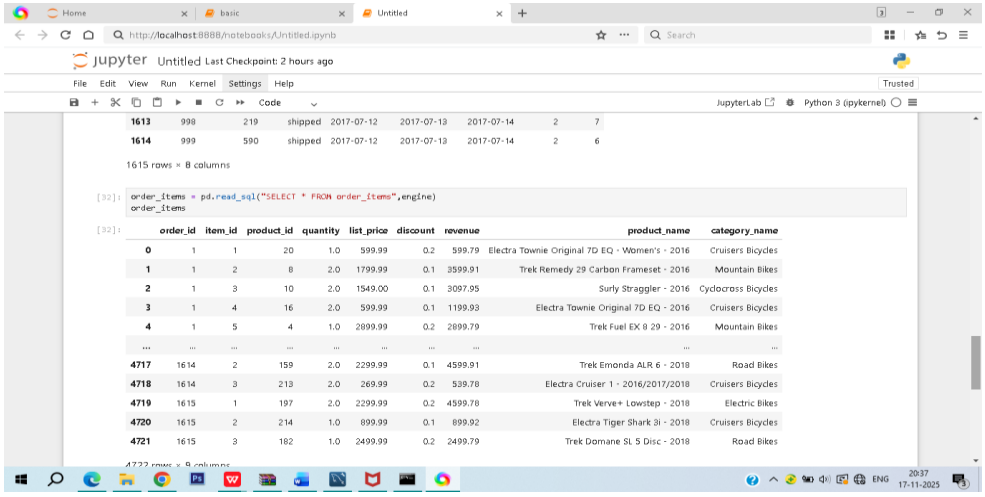
4. orders = pd.read_sql ("SELECT * FROM orders", engine)

display(orders)

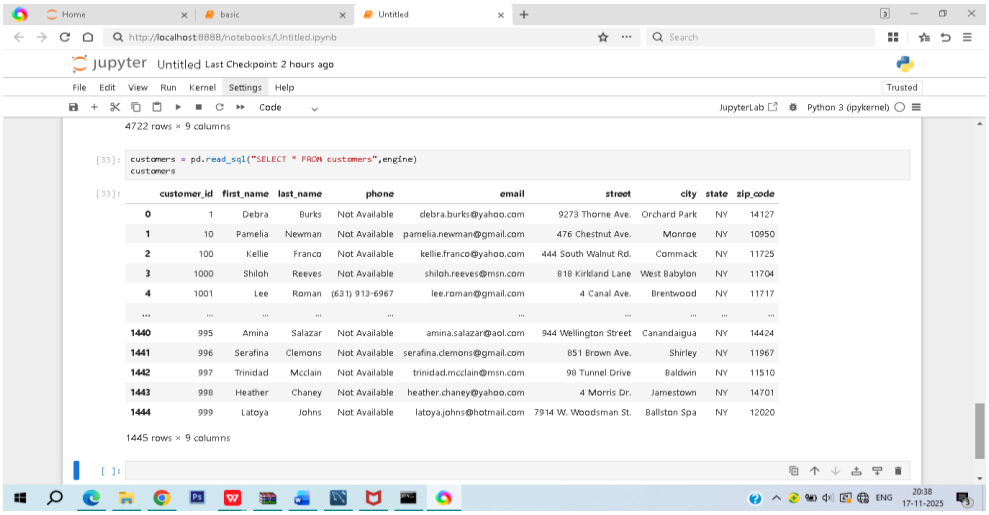
Orders:



Order_Items:



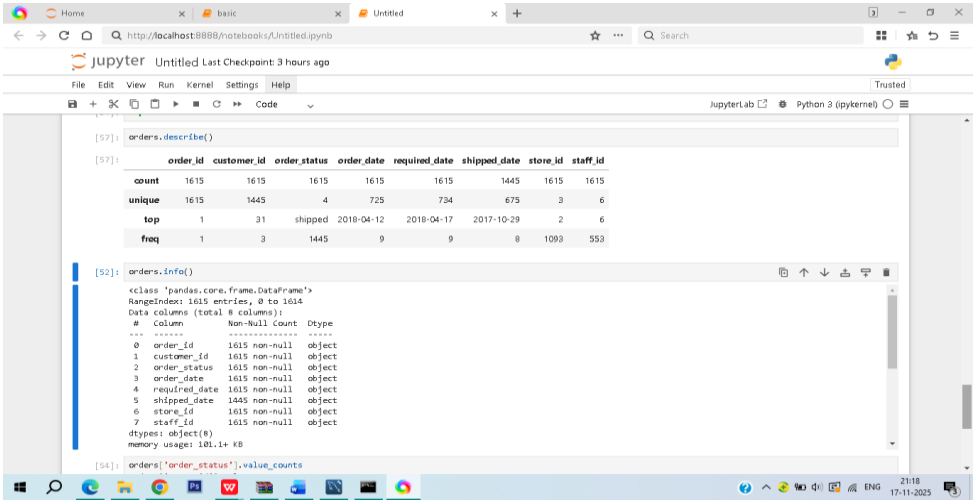
Customers



Q2. Basic EDA (Exploratory Data Analysis)

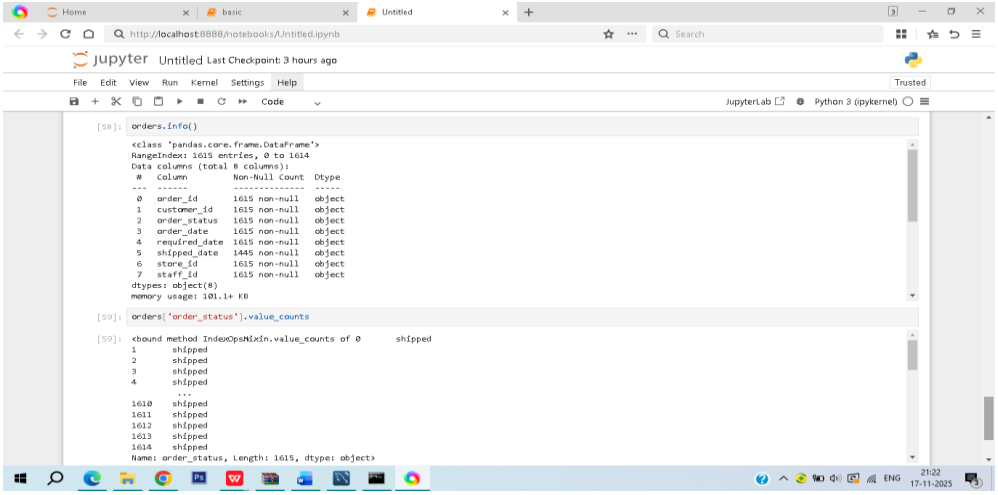
i) Orders:

Order.describe()



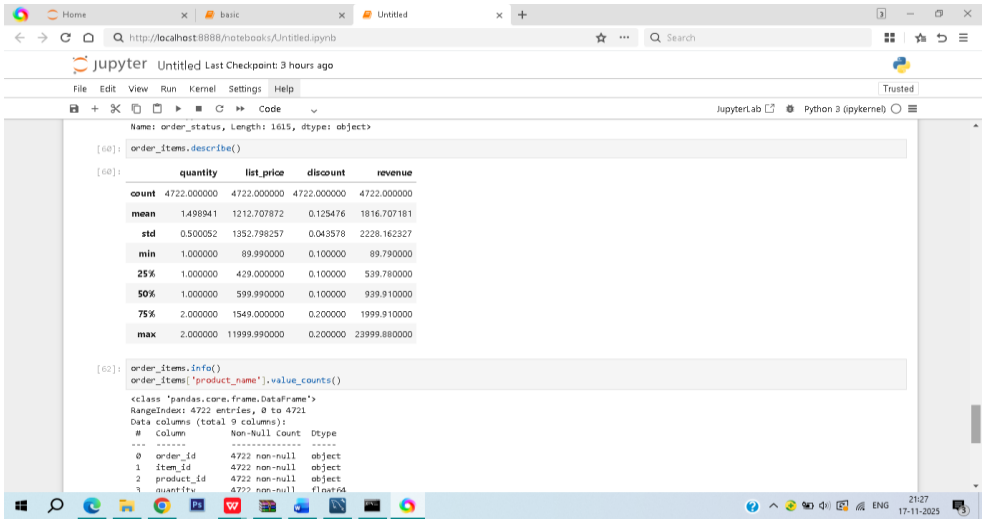
orders.info ()

orders['order_status'].value_counts



Order_items:

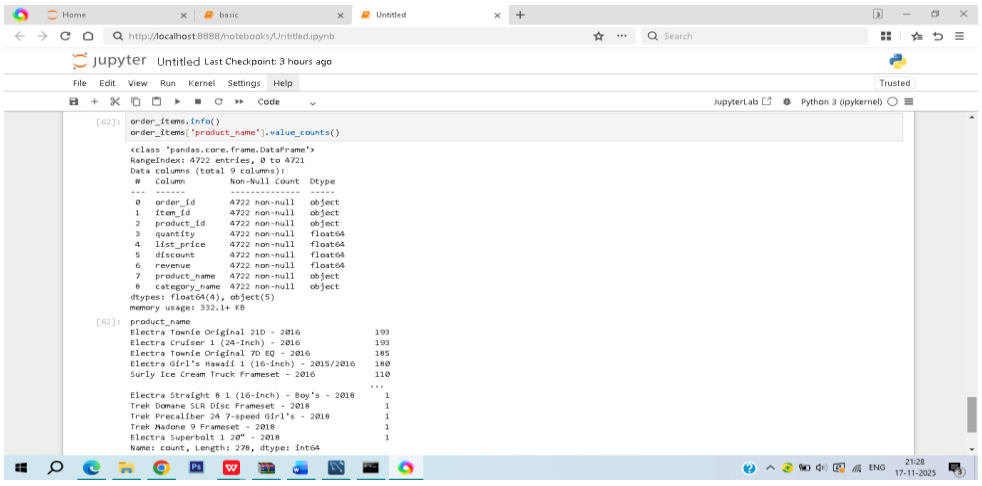
order_items.describe()



	quantity	list_price	discount	revenue
count	4722.000000	4722.000000	4722.000000	4722.000000
mean	1.498941	1212.707872	0.125476	1816.707181
std	0.500052	1352.798257	0.043578	2228.162327
min	1.000000	89.990000	0.100000	89.790000
25%	1.000000	429.000000	0.100000	539.780000
50%	1.000000	599.990000	0.100000	939.910000
75%	2.000000	1549.000000	0.200000	1999.910000
max	2.000000	11999.990000	0.200000	23999.880000

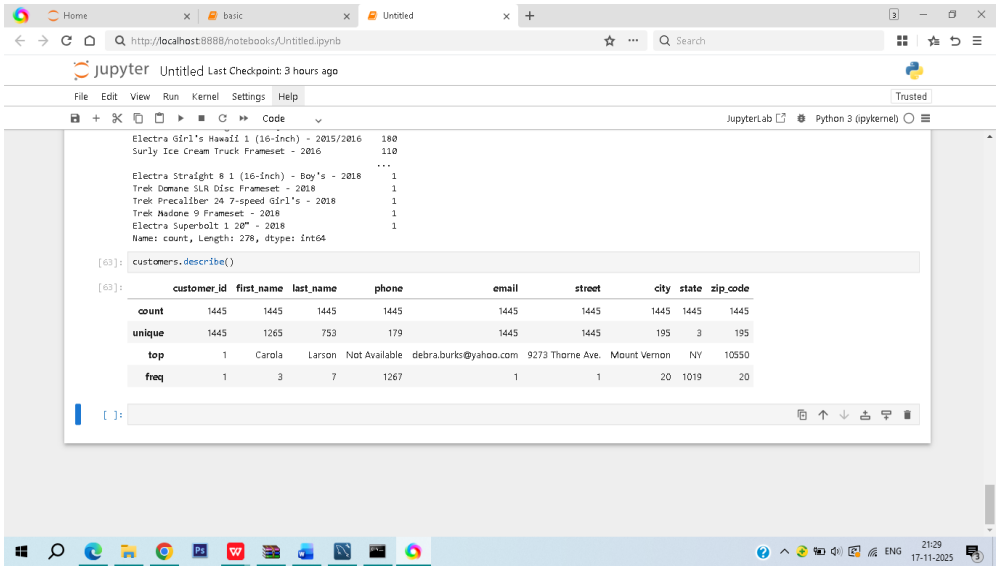
order_items.info ()

order_items[‘product_name’].value_counts( )



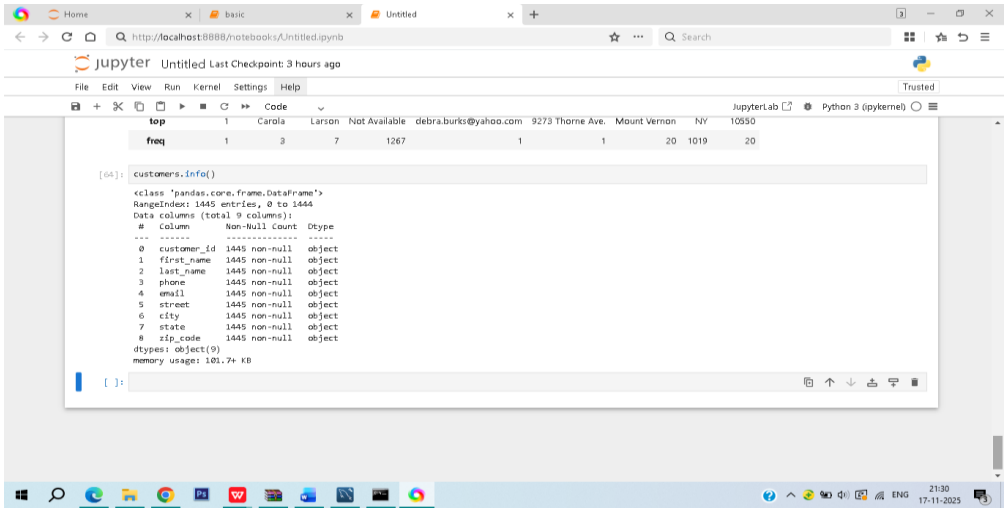
Customers:

customers. Describe()

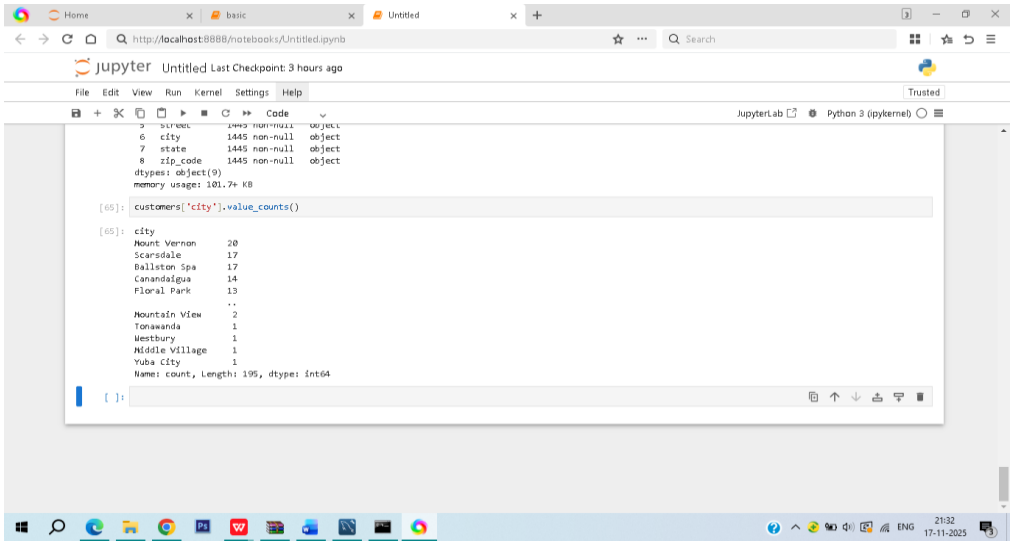


	customer_id	first_name	last_name	phone	email	street	city	state	zip_code
count	1445	1445	1445	1445	1445	1445	1445	1445	1445
unique	1445	1265	753	179	1445	1445	195	3	195
top	1	Carola	Larson	Not Available	debra.burks@yahoo.com	9273 Thorne Ave.	Mount Vernon	NY	10550
freq	1	3	7	1267	1	1	20	1019	20

customers.info ()



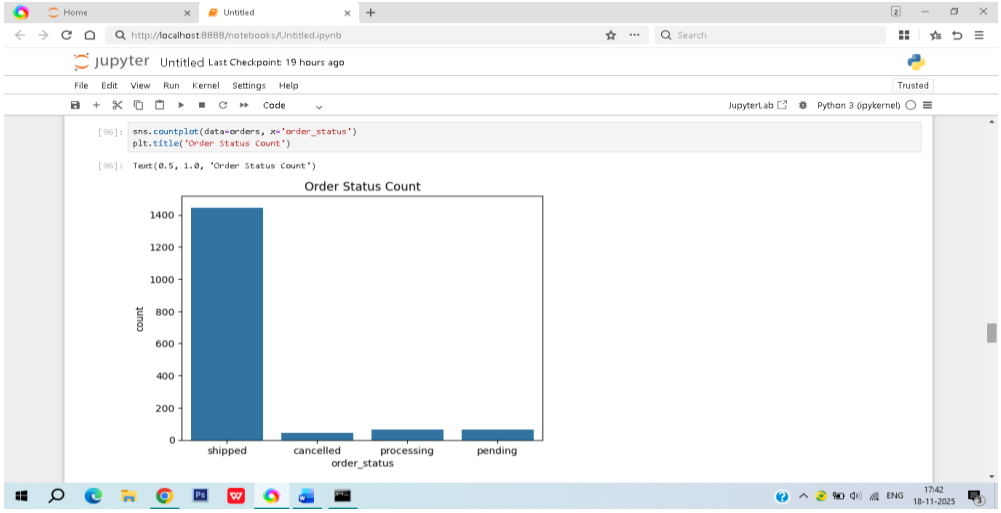
Customers['city'].value_counts ()



ii) Bar Chart – Order Status Distribution using Seaborn.Countplot

sns.countplot(data=orders, x='order_status')

plt.title('Order Status Count')



Q3. Calculate RFM Features for Customers

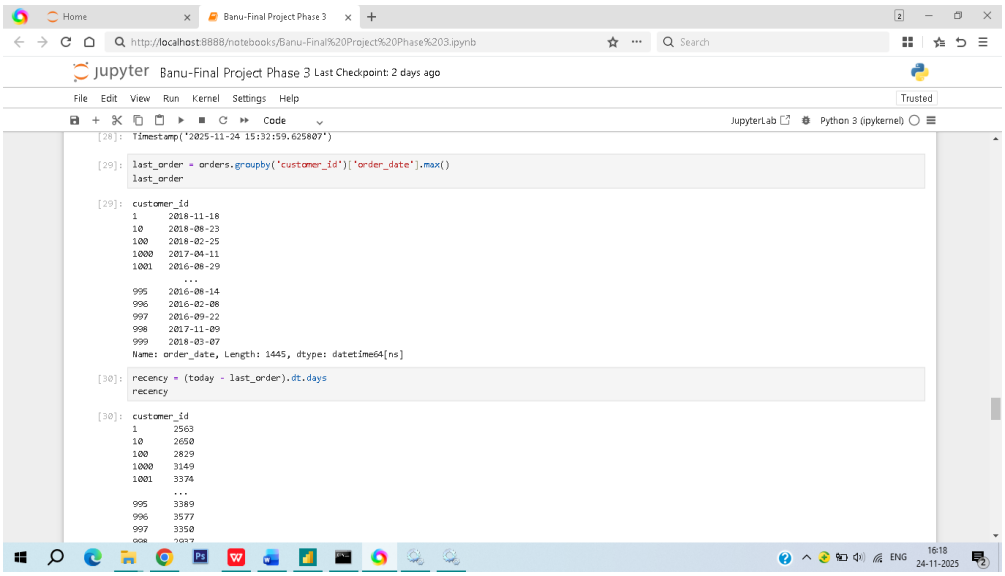
Steps:

1. Calculate Recency

i) Determine the max order date for each customer

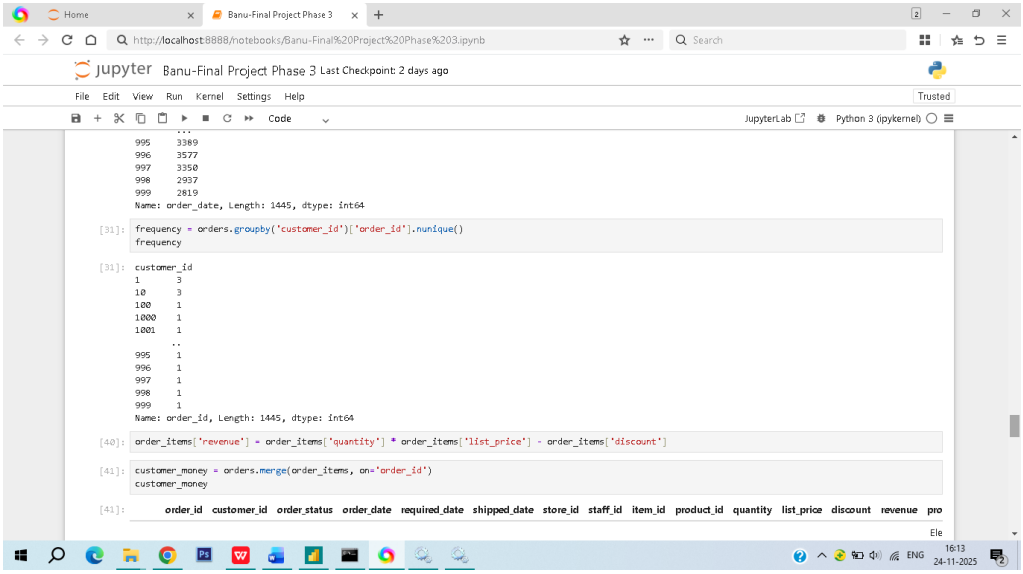
ii) Define a reference

iii) Calculate the difference in days between the Reference Date and the customer's max order date.



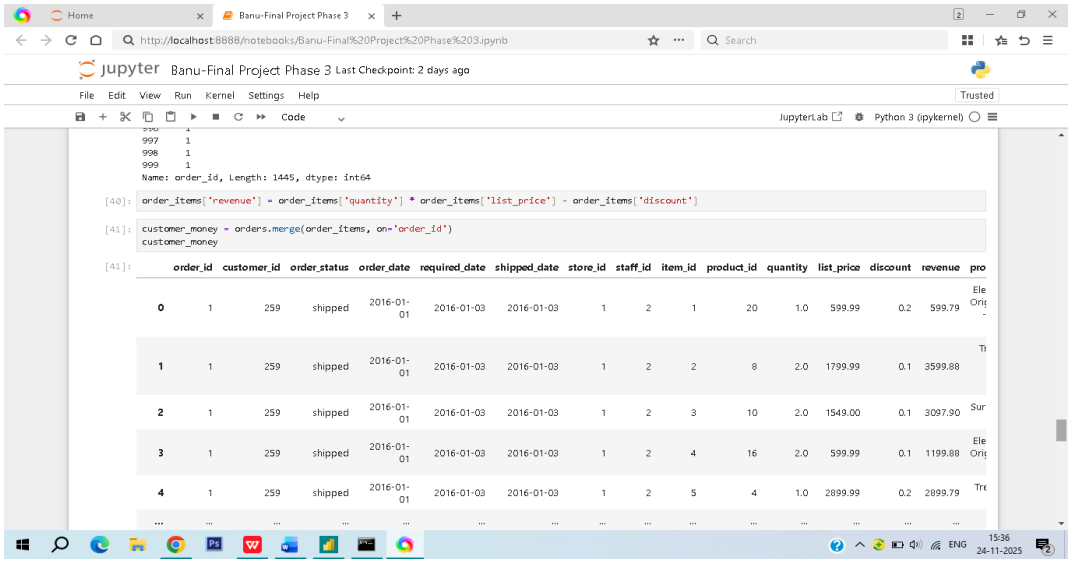
2. Calculate Frequency

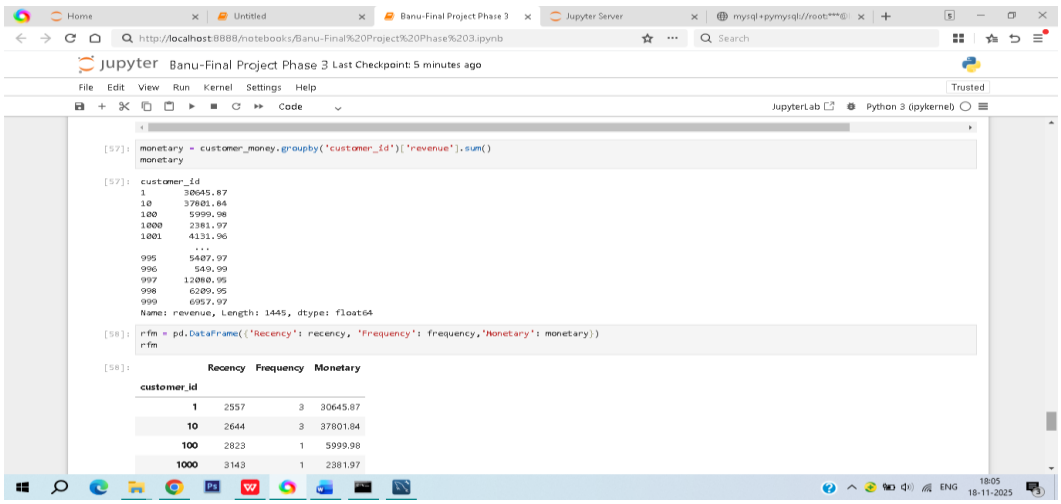
To determine the total number of transactions made by each customer's, Count the number of unique order_id associated with each customer_id from the orders table. It will show the dataset showing the Number of Orders placed by each customer.



3. Calculate Monetary value

To calculate Total net purchase value for each customer, Group the order_items table by customer_id and sum the calculated Total Revenue, it will give the result ,the dataset where each customer is associated with their Total value of all purchases.



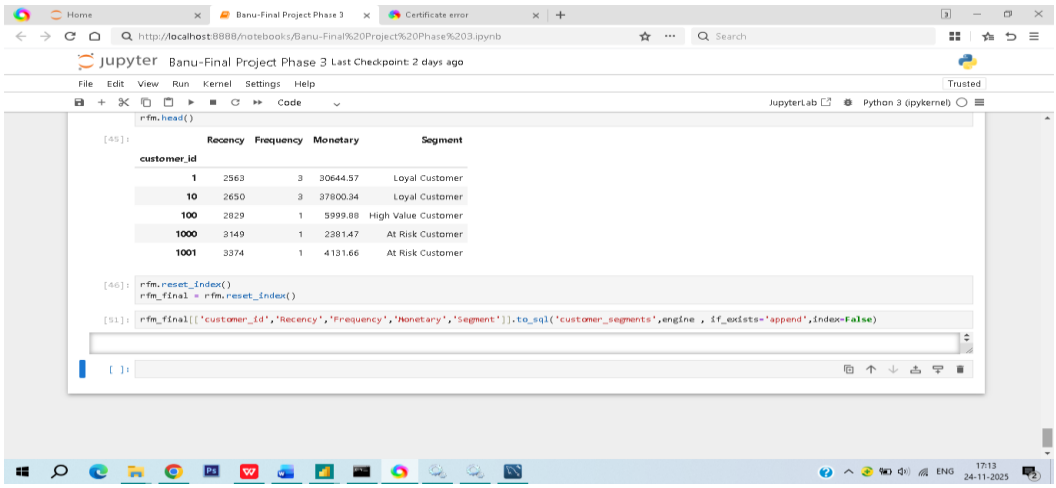


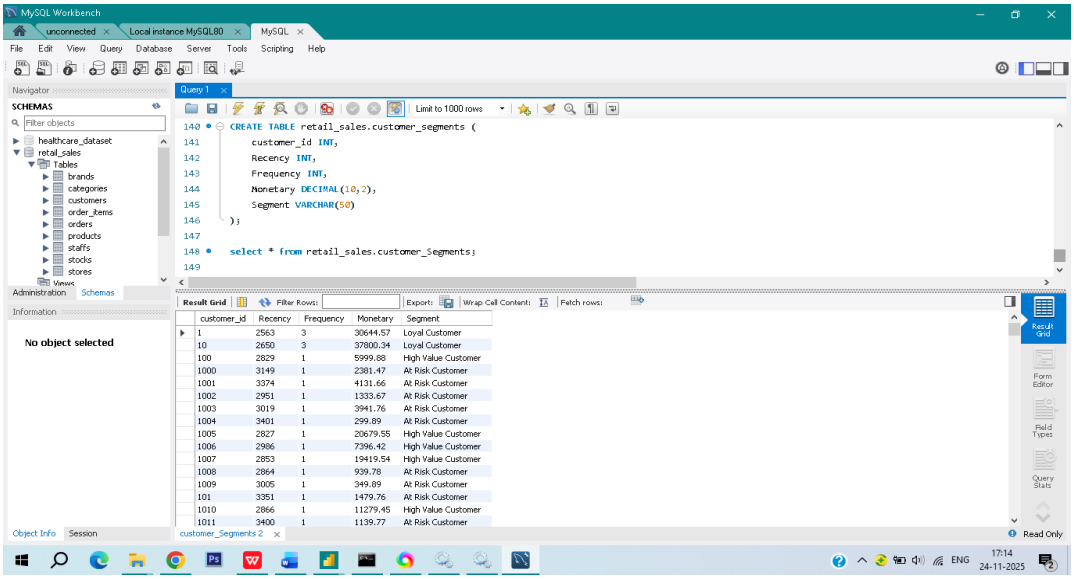
Q4. Export Segmentation Results to SQL

rfm.reset_index()

rfm_final = rfm.reset_index()

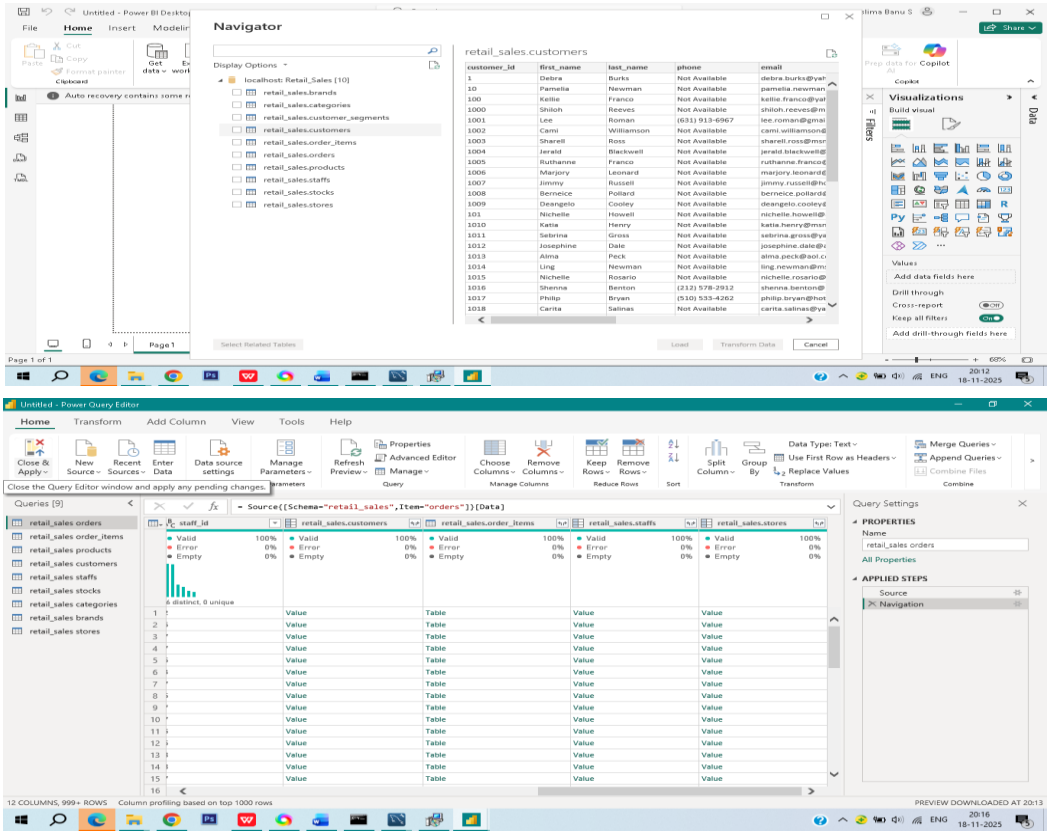
rfm_final[['customer_id','Recency','Frequency','Monetary','Segment']].to_sql('customer_segments',engine , if_exists='append',index=False)





Phase 4: Power BI – Visualization & Dashboarding

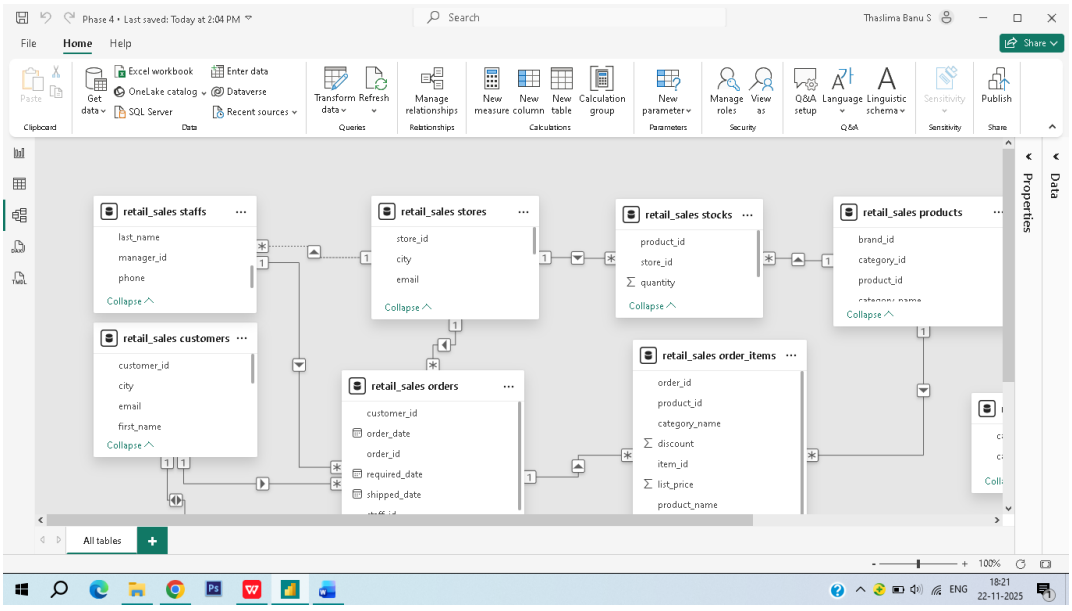
1. In Power BI – Click Get data – More – Search for MySQL Database.
2. Click on Connect.
3. Select Database - Enter Username and Password
4. Click connect.
5. Select the Tables we need to Transform
6. Load the Tables – check errors and rectify it.



Q2. Create Relationships Between Tables

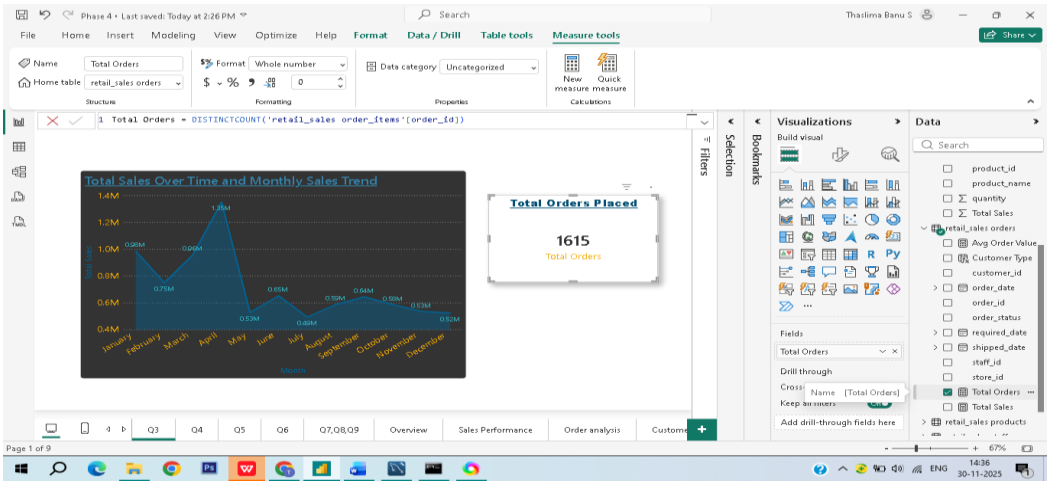
Relationship Flow	Cardinality	Rationale
Customers - Orders	One-to-many(1:*)	A customer can place many orders.
Staffs - Orders	One-to-many(1:*)	A staff member can process many orders.
Orders – OrderItems	One-to-many(1:*)	One order contains many individual items.
Products - OrderItems	One-to-many(1:*)	One product can be sold across many order items.
Brands - Products	One-to-many(1:*)	One brand can produce many products.
Categories - Products	One-to-One(1:1)	One category can contain many products.
Customer – CustomerSegments	One-to-One(1:1)	Each unique customer is assigned a single, specific segment.

Click on Model View – to view the ER Diagram

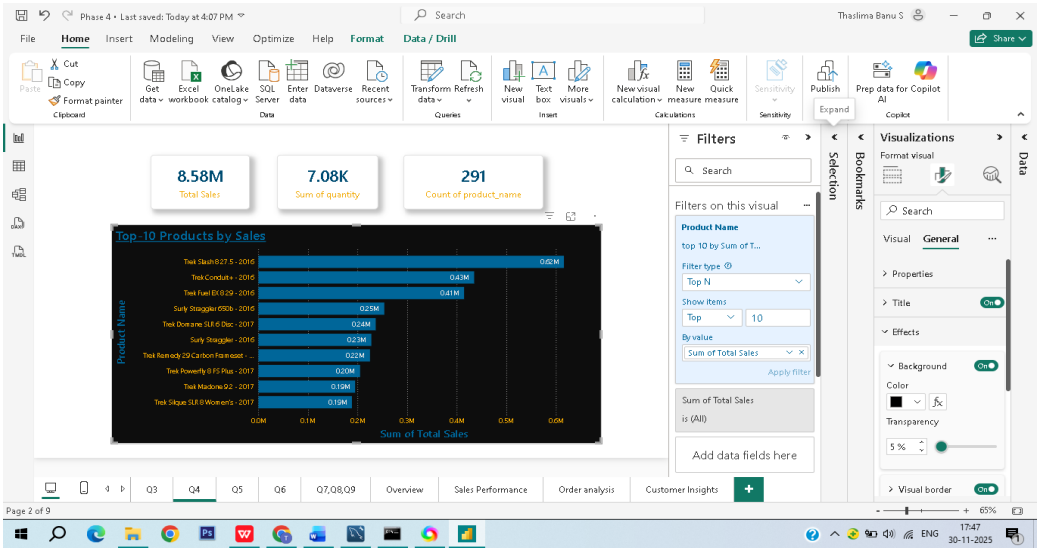


Q3. Sales Overview Report

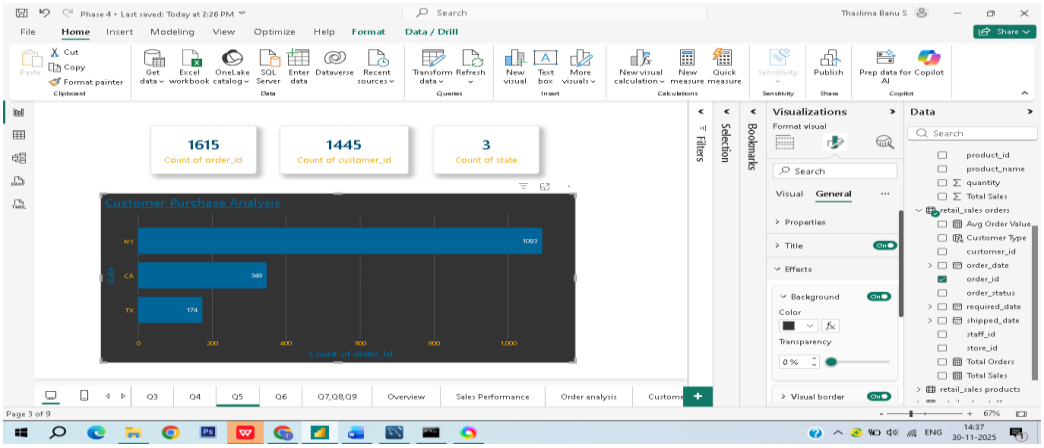
To Calculate TotalOrders Placed – click new measure – Enter the Dax formula - Total Orders = DISTINCTCOUNT('retail_sales orders'[order_id])



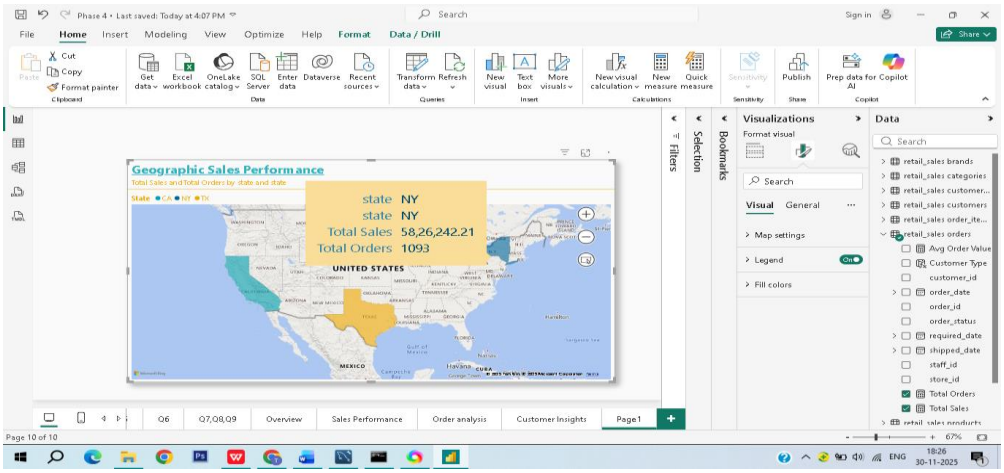
Q4. Top Products by Sales



Q5. Customer Purchase Analysis



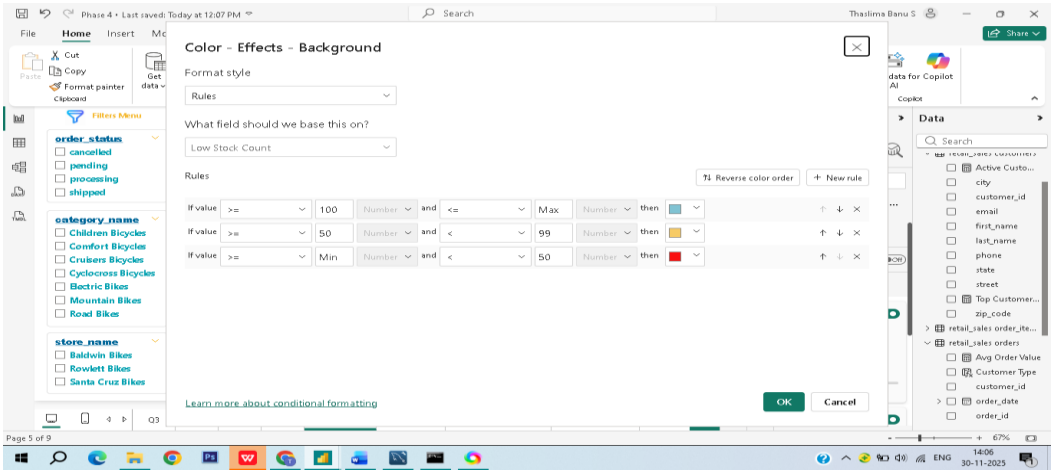
Q6. Sales by Store Map



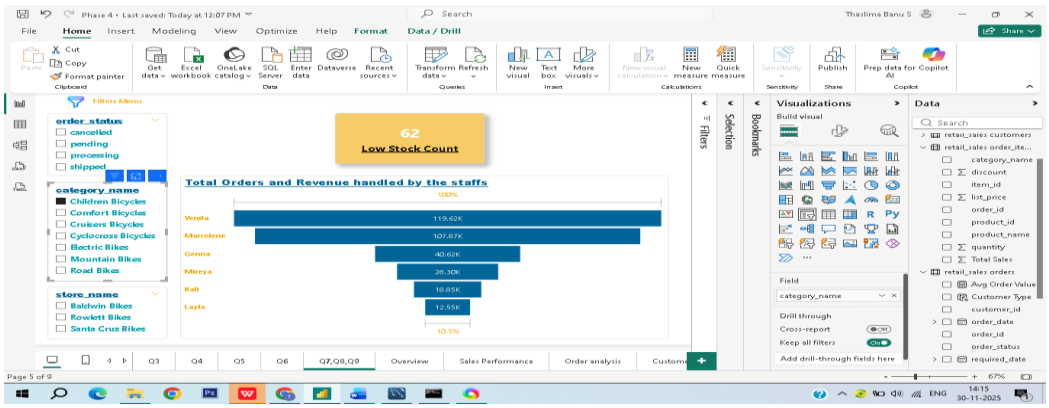
Q7. Low Stock Alert Dashboard

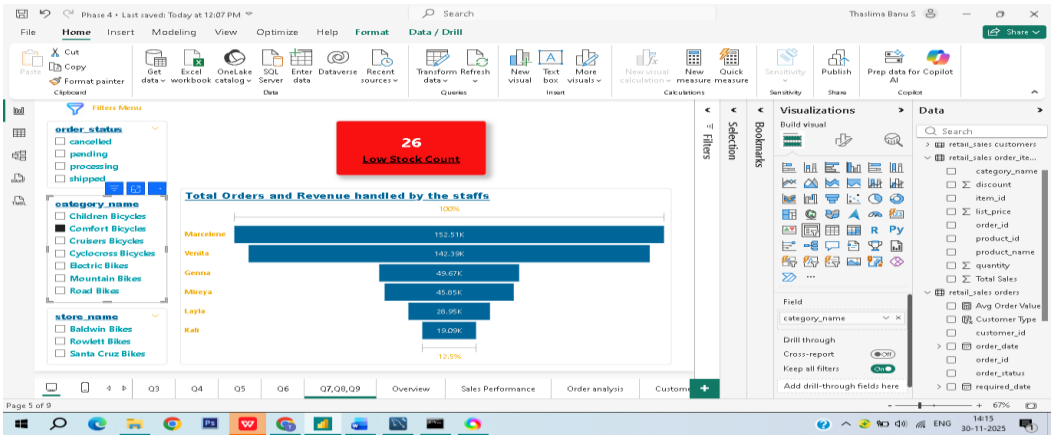
Apply Conditional Formatting

1. Select the card – in Format Visuals – Select format your visuals – Go for General – Effects – Background - Click on the fx

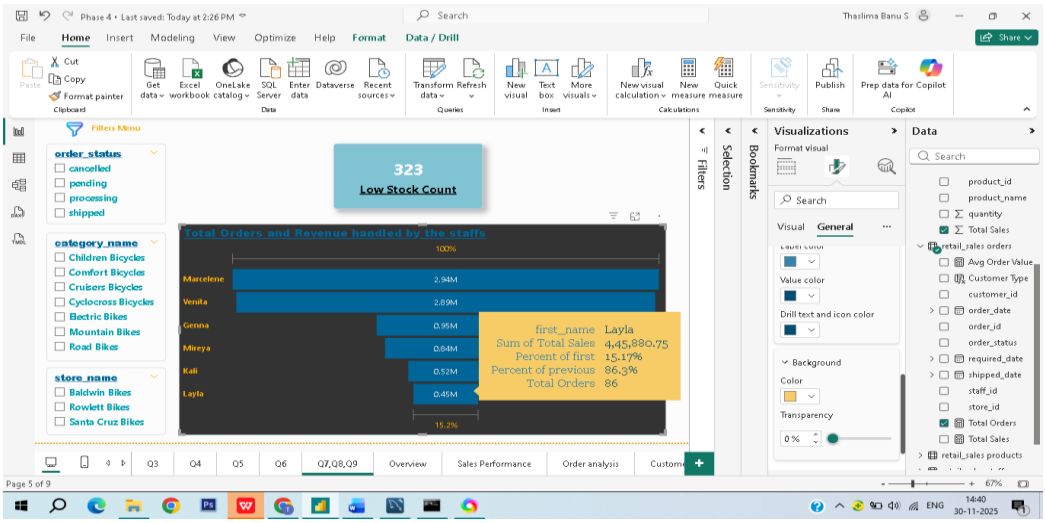


Q8. Interactive Filters and Slicers

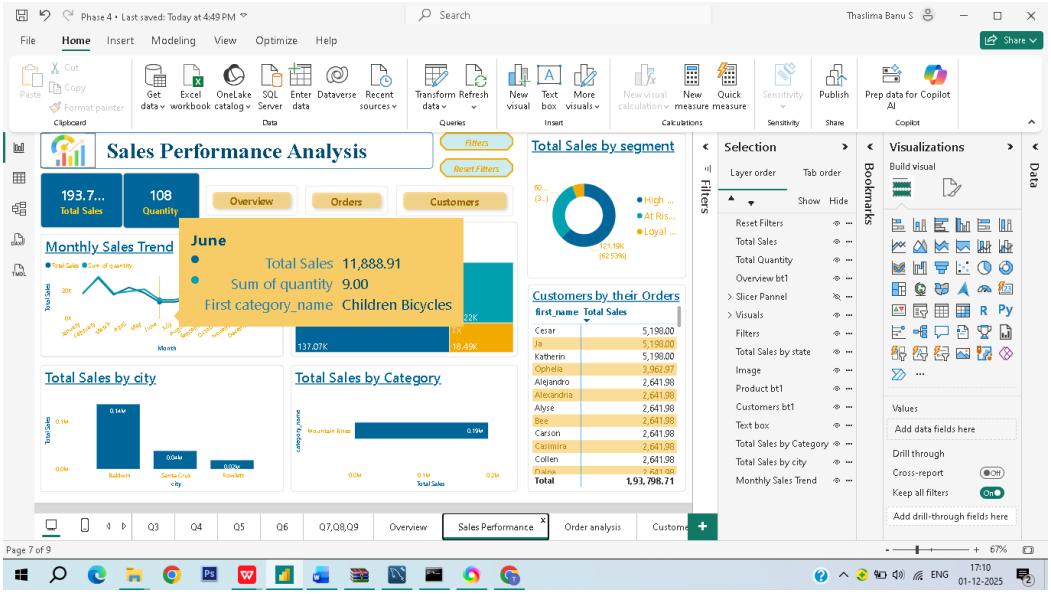


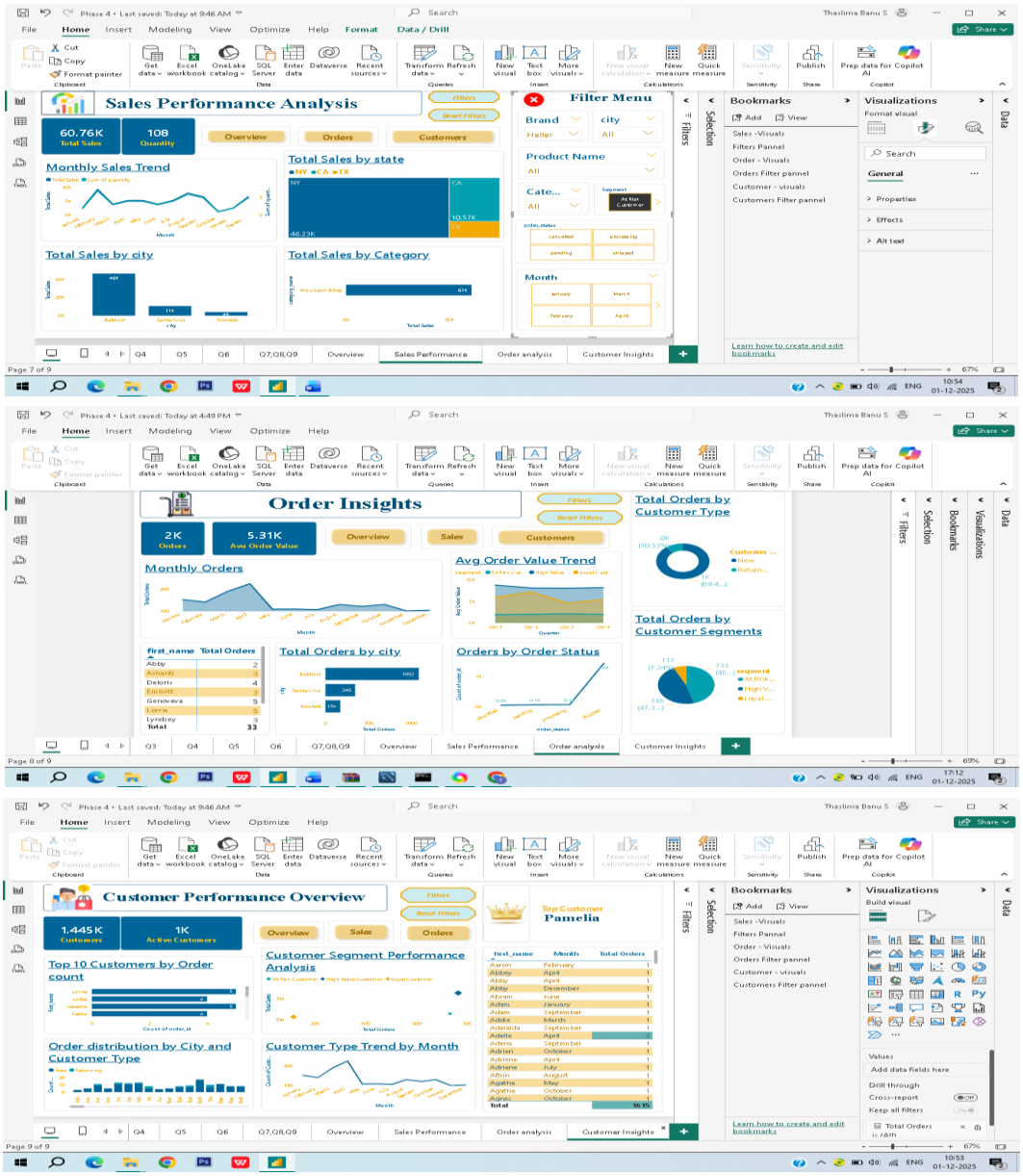


Q9. Staff Performance Report



Q10. Consolidated Dashboard Page





Conclusion:

This Retail Sales Analytics project integrated multiple analytical tools such as Excel, SQL, Python, and Power BI to analyze sales performance and customer behaviour for a retail business. The dataset was cleaned and standardized in Excel, and a structured database was created in MySQL to run meaningful analytical queries, including top-selling products, sales by store, customer segmentation, and staff performance. Python was used for exploratory data analysis and RFM customer segmentation, enabling a deeper understanding of purchasing patterns. Finally, Power BI visualizations provided an interactive dashboard offering insights on total orders, top products, sales trends, store performance, low stock alerts, and customer contribution.

Overall, this project demonstrates the complete data analytics workflow—from raw data to actionable insights—supporting better decision-making and business improvement.