RETAIL SALES ANALYISIS

Name :- Gunashree S

USN :- 2022408021

Program :- BCA

Section :- ‘A’ Section

Specialization :- AI, DS and SS

Title :- DBMS + Data Science Integrated Project

OBJECTIVE

The purpose of this project is to understand how databases and data science work together to extract useful business insights. This includes designing a retail sales database, querying it with SQL, visualizing results using Python, and interpreting patterns to support business decisions.

DATABASE DESIGN

The analysis is based on a relational database structured around three core entities: Products, Customers, and Sales.

ENTITY-RELATIONSHIP (ER) DIAGRAM

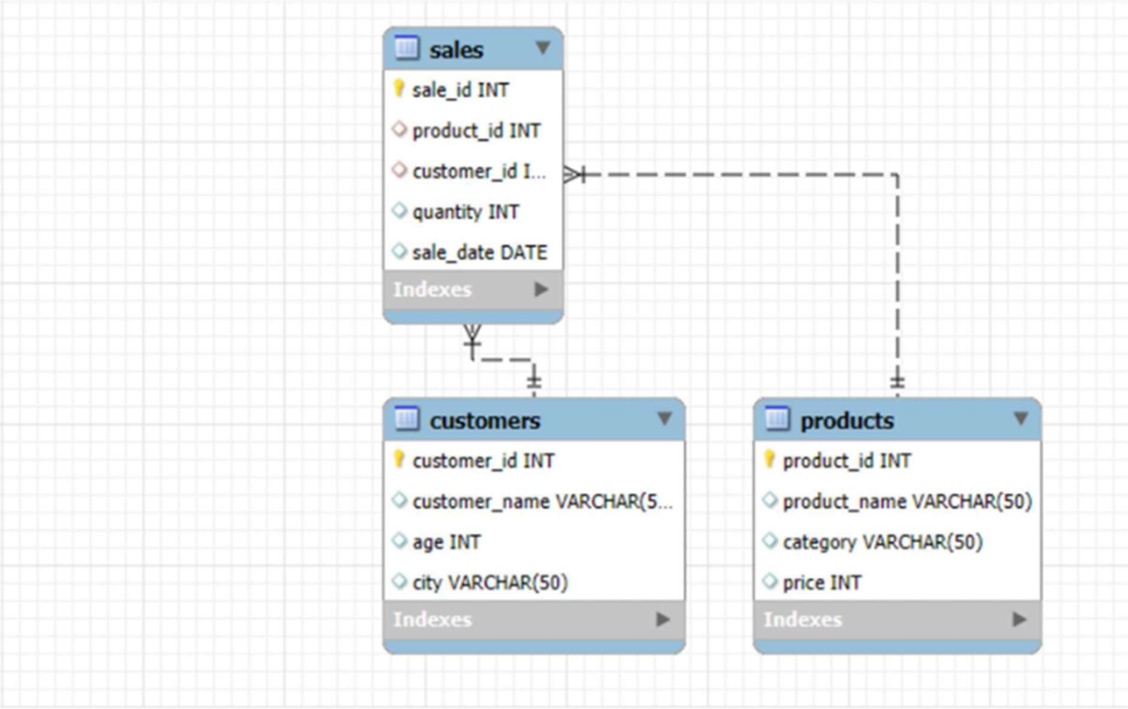


TABLE EXPLAINATIONS

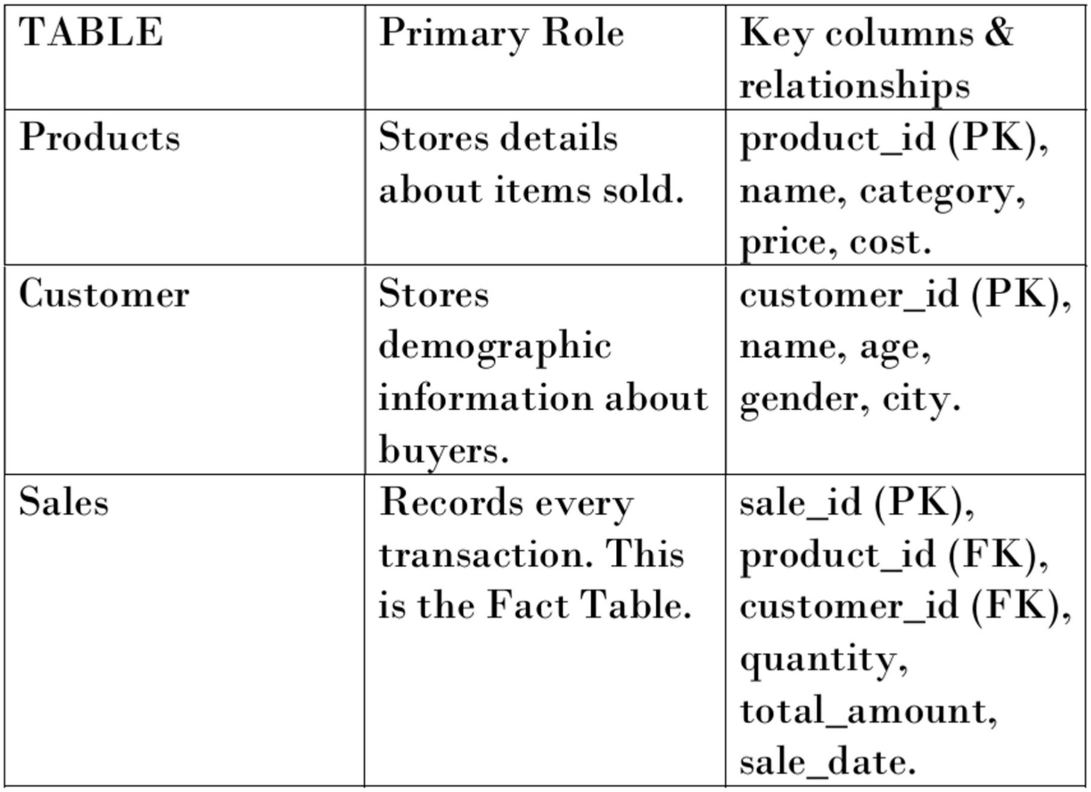
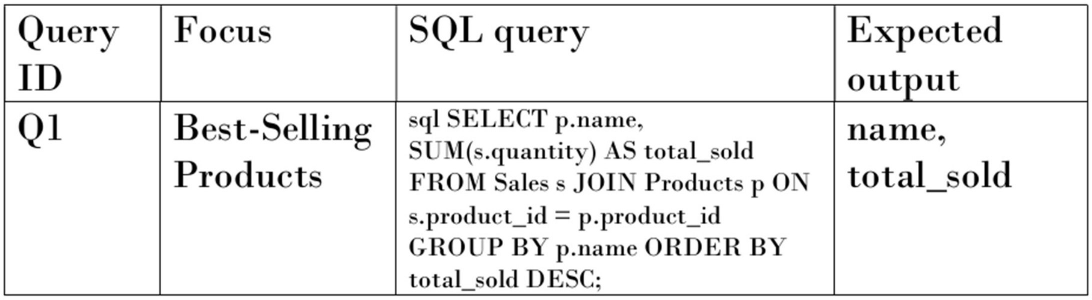


TABLE	Primary Role	Key columns & relationships
Products	Stores details about items sold.	product_id (PK), name, category, price, cost.
Customer	Stores demographic information about buyers.	customer_id (PK), name, age, gender, city.
Sales	Records every transaction. This is the Fact Table.	sale_id (PK), product_id (FK), customer_id (FK), quantity, total_amount, sale_date.

LIST OF INSIGHTS FROM SQL QUERIES

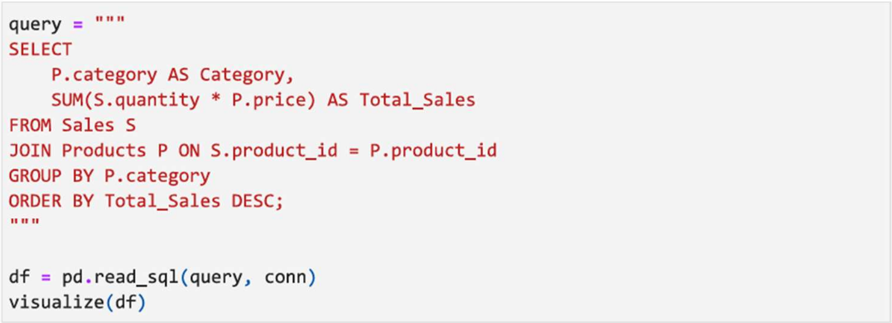


Query ID	Focus	SQL query	Expected output
Q1	Best-Selling Products	sql SELECT p.name, SUM(s.quantity) AS total_sold FROM Sales s JOIN Products p ON s.product_id = p.product_id GROUP BY p.name ORDER BY total_sold DESC;	name, total_sold

Q2	Best-Selling Categories	sql SELECT p.category, SUM(s.total_amount) AS total_sales FROM Sales s JOIN Products p ON s.product_id = p.product_id GROUP BY p.category;	category, total_sales
Q3	Customer Value (Average Purchase)	sql SELECT c.name, AVG(s.total_amount) AS avg_purchase FROM Sales s JOIN Customers c ON s.customer_id = c.customer_id GROUP BY c.name;	name, avg_purchase
Q4	Daily Sales Trend	sql SELECT s.sale_date, SUM(s.total_amount) AS daily_sales FROM Sales s GROUP BY s.sale_date ORDER BY s.sale_date;	sale_date, daily_sales

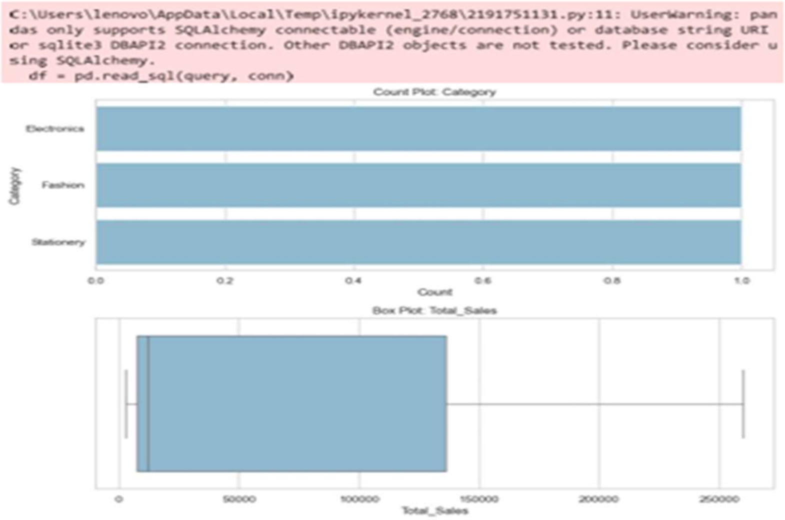
QUERY ID	FOCUS	SQL FUNCTIONS	Expected Output
Q1	Best-Selling Products	Sum()	product_name, total_sold
Q2	Best-Selling Categories	Sum()	category_name, total_sales
Q3	Customer Value (Average Purchase)	Avg()	customer_name, avg_purchase
Q4	Daily Sales Trend	Sum(), Date()	sale_date, daily_sales
Q5	Age vs. Purchase Value	Avg()	customer_age, avg_purchase (by age group)

SCREENSHOTS OF VISUALIZATIONS

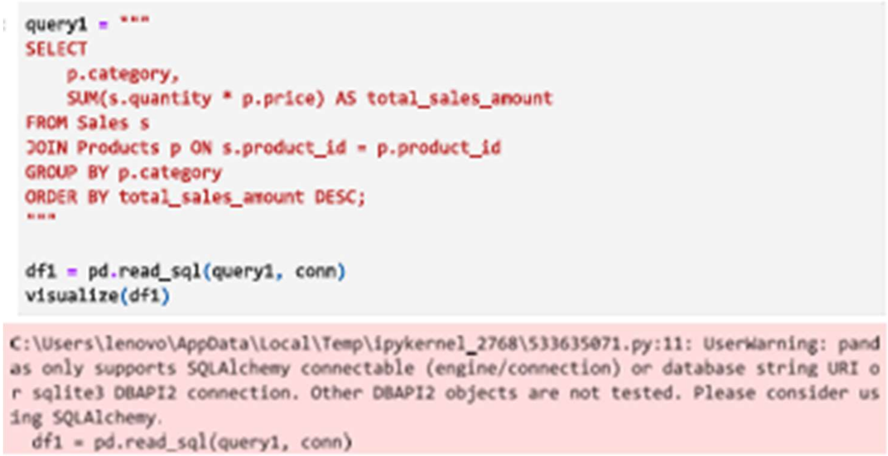


```
query = """
SELECT
    P.category AS Category,
    SUM(S.quantity * P.price) AS Total_Sales
FROM Sales S
JOIN Products P ON S.product_id = P.product_id
GROUP BY P.category
ORDER BY Total_Sales DESC;
"""

df = pd.read_sql(query, conn)
visualize(df)
```



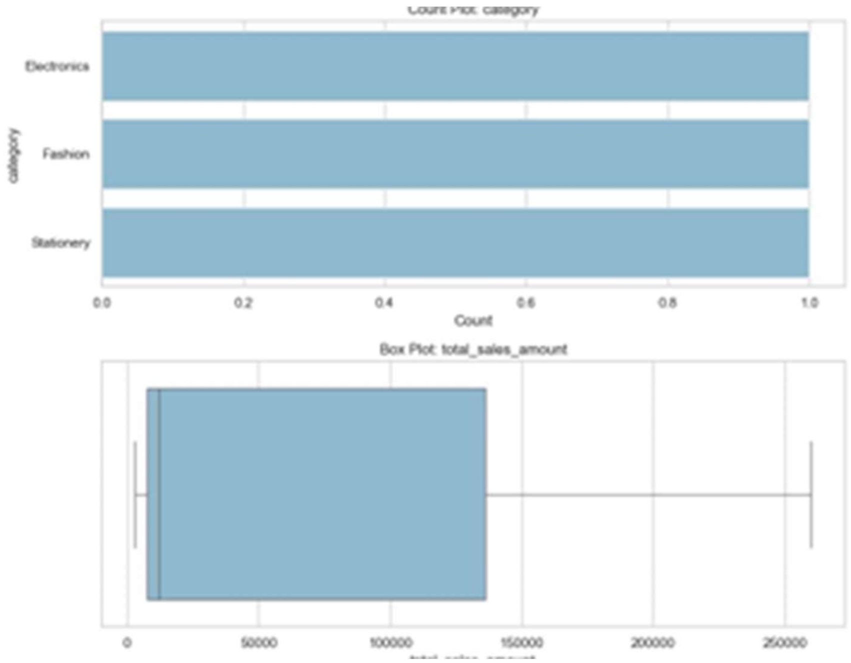
```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_2768\2191751131.py:11: UserWarning: pan
das only supports SQLAlchemy connectable (engine/connection) or database string URI
or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider u
sing SQLAlchemy.
  df = pd.read_sql(query, conn)
```

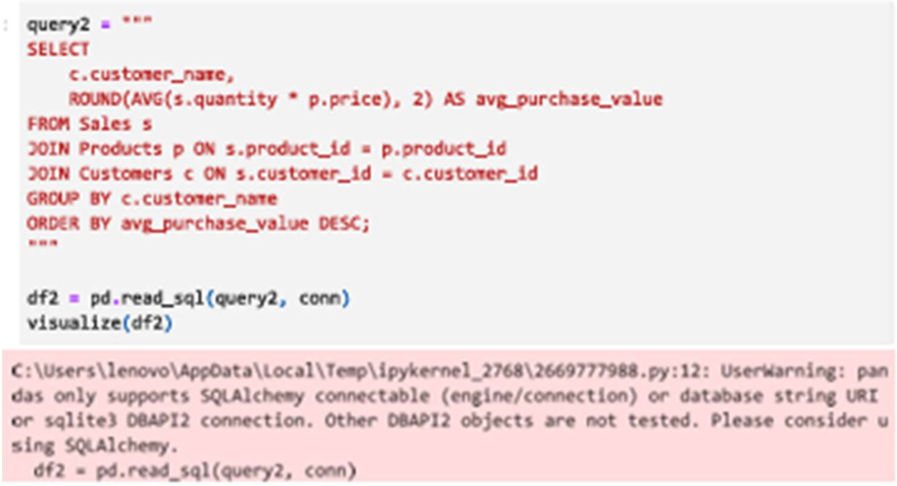


```
query1 = """
SELECT
    p.category,
    SUM(s.quantity * p.price) AS total_sales_amount
FROM Sales s
JOIN Products p ON s.product_id = p.product_id
GROUP BY p.category
ORDER BY total_sales_amount DESC;
"""

df1 = pd.read_sql(query1, conn)
visualize(df1)
```

```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_2768\533635071.py:11: UserWarning: pand
as only supports SQLAlchemy connectable (engine/connection) or database string URI o
r sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider us
ing SQLAlchemy.
  df1 = pd.read_sql(query1, conn)
```

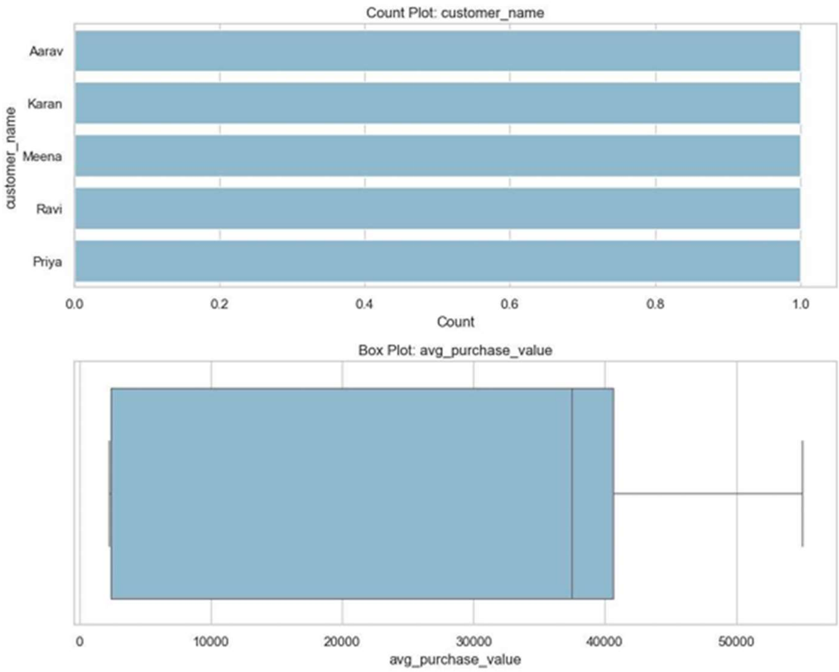


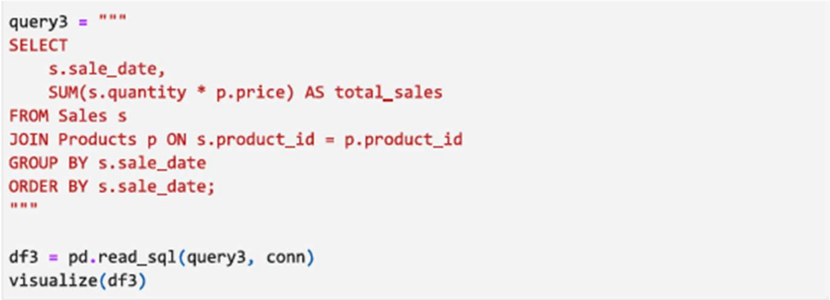


```
query2 = """
SELECT
    c.customer_name,
    ROUND(AVG(s.quantity * p.price), 2) AS avg_purchase_value
FROM Sales s
JOIN Products p ON s.product_id = p.product_id
JOIN Customers c ON s.customer_id = c.customer_id
GROUP BY c.customer_name
ORDER BY avg_purchase_value DESC;
"""

df2 = pd.read_sql(query2, conn)
visualize(df2)
```

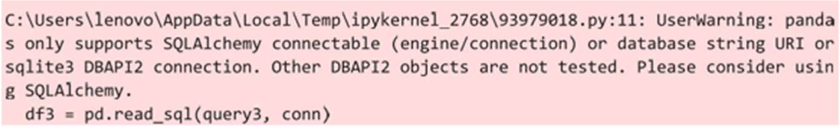
```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_2768\2669777988.py:12: UserWarning: pan
das only supports SQLAlchemy connectable (engine/connection) or database string URI
or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider u
sing SQLAlchemy.
  df2 = pd.read_sql(query2, conn)
```



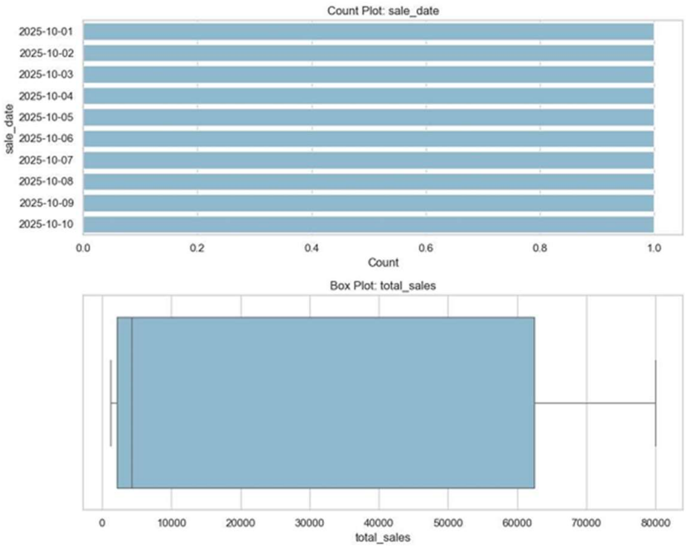


```
query3 = """
SELECT
    s.sale_date,
    SUM(s.quantity * p.price) AS total_sales
FROM Sales s
JOIN Products p ON s.product_id = p.product_id
GROUP BY s.sale_date
ORDER BY s.sale_date;
"""

df3 = pd.read_sql(query3, conn)
visualize(df3)
```



```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_2768\93979018.py:11: UserWarning: panda
s only supports SQLAlchemy connectable (engine/connection) or database string URI or
sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider usin
g SQLAlchemy.
  df3 = pd.read_sql(query3, conn)
```



OBSERVATIONS AND INSIGHTS

Customer Name Frequency Analysis (Q1 Analysis)

- **Bar Chart / Count Plot for product_name :-** This plot shows that every unique product name is represented exactly once in the aggregated result. This confirms the SQL query's successful grouping, as the GROUP BY clause ensures one row per product, but provides no direct insight into sales volume. The purpose of this specific visualization is simply to validate the structure of the data returned from the database.
- **Bar Chart for total_sold :-** This chart is the primary result of the query, clearly illustrating the sales performance hierarchy. The bars, sorted from highest to lowest, immediately identify the best-selling products in terms of units sold. This visualization is critical for business decision-making, highlighting the top performers that drive sales volume and should be prioritized in inventory and marketing efforts.

Category and Total Sales Analysis (Q2 Analysis)

- **Bar Chart / Count Plot for category_name :-** This plot confirms that each unique category is present exactly once in the resulting table, which is a result of the GROUP BY clause. It serves as a data integrity check, validating that all sales for a category have been properly summed into a single entry.

- **Bar Chart for total_sales (or revenue) :-** This is the most significant visualization, showing which business segments generate the most revenue. By highlighting the dominant categories, this chart is essential for budget allocation and resource planning, guiding where the company should focus its investment efforts.

Customer Name and Avg Purchase Analysis (Q3 Analysis)

- **Bar Chart / Count Plot for customer_name :-** This graph primarily serves as a quick check to ensure the SQL query successfully isolated a single, unique record for each customer in the aggregated table.
- **Bar Chart for avg_purchase :-** This is a highly valuable chart, as it clearly identifies your most financially valuable customers based on their average transaction size. This information is key for developing personalized retention strategies and VIP programs to maximize future revenue from these high-value accounts.

Sale Date and Daily Sales Analysis (Q4 Analysis)

- **Time Series / Line Plot for daily sales :-** This is the most crucial visualization, as it reveals the sales trend and seasonality over the observed period. By plotting sales value against time, you can immediately identify peaks, troughs, and consistent patterns (e.g., higher sales on weekends). This plot is essential for forecasting, identifying anomalies, and understanding business cycles to optimize operations.

CONCLUSION

The integrated analysis, leveraging SQL for efficient data aggregation and Python for detailed visualization, successfully transformed raw sales data into actionable business intelligence. The project established a clear understanding of the sales environment, identifying a strong concentration of revenue within specific product categories and an identifiable difference in spending habits across customer demographics. These findings provide a solid foundation for strategic decision-making, allowing the business to move beyond descriptive statistics to prescriptive actions that optimize inventory, marketing spend, and operational scheduling.