SQL AND BUSINESS ANALYTICS: WEEK 2 OVERVIEW

Exploring the Role of SQL in Enhancing Business Insights

Week 2 Documentation: SQL Implementation & Business Analytics

Project Title

Sales & Customer Analytics Using SQL — Week 2

Objective

In Week 2, we converted the cleaned dataset into a star-schema SQL database and performed key analytics to generate basic business insights, including a Single Customer View.

---

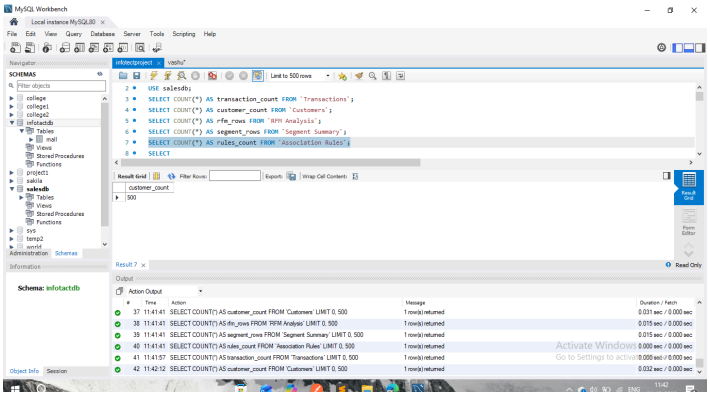
1. Database & Table Setup

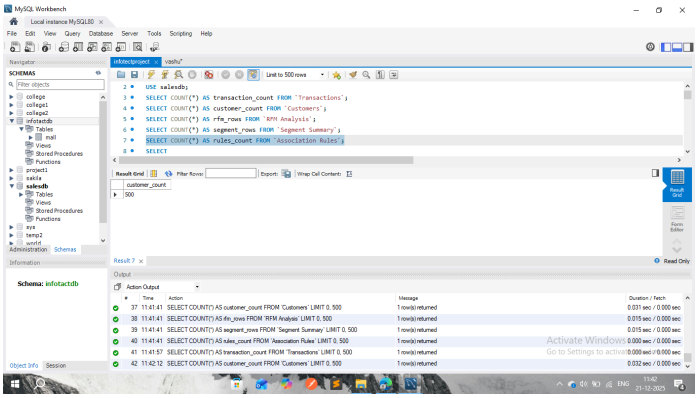
1.1 Tables Imported

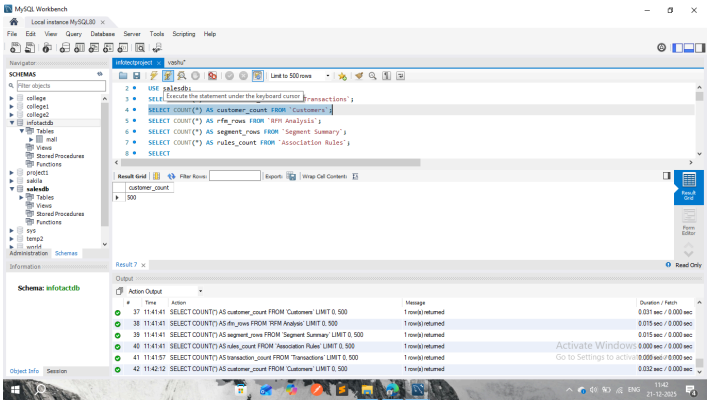
We used the MySQL Data Import Wizard to load the following cleaned CSV/Excel files:

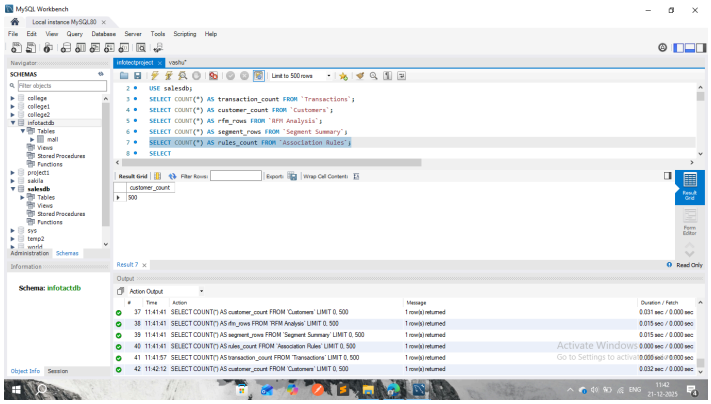
Table Name	Source File	Description
customers	Customers.csv	Customer master data
transactions	Transactions.csv	Sales transactions (fact table)
rfm_analysis	RFM Analysis.csv	RFM scoring data
segment_summary	Segment Summary.csv	Segment performance aggregates
association_rules	Association Rules.csv	Product association metrics

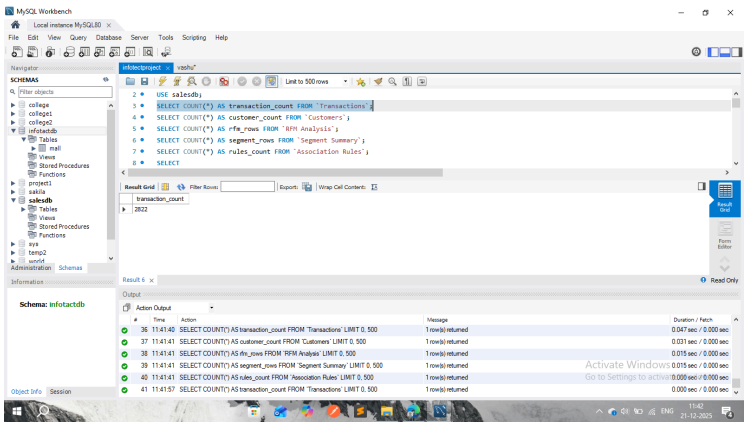
SCHREENSHOOTS











---

2. Data Cleaning & Preliminary Checks

2.1 Studied the Tables

We reviewed table structures using:

- DESCRIBE transactions;
- DESCRIBE customers;
- DESCRIBE rfm_analysis;
- DESCRIBE segment_summary;
- DESCRIBE association_rules;

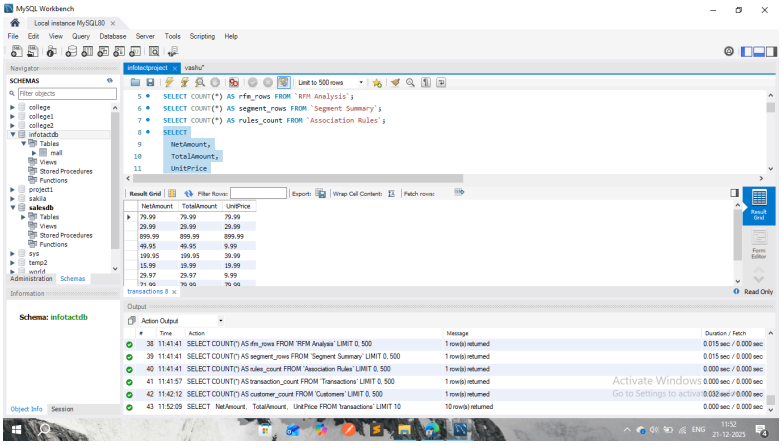
2.2 Cleaned Numeric Columns

Sales amount fields originally contained currency symbols ($). We removed $ from:

- NetAmount
- TotalAmount
- UnitPrice

to ensure they are numeric and aggregatable.

Screenshot:



2.3 Check for NULL Keys

We verified key columns do not contain NULL values:

- SELECT COUNT(*) FROM transactions WHERE CustomerID IS NULL;
- SELECT COUNT(*) FROM transactions WHERE ProductID IS NULL;
- SELECT COUNT(*) FROM customers WHERE CustomerID IS NULL;

✓ No NULL primary key values found.

---

3. Table Enhancements: Data Types & Keys

3.1 Convert Key Columns to Fixed Types

To enable foreign keys, we changed TEXT to VARCHAR for keys:

ALTER TABLE transactions MODIFY COLUMN CustomerID VARCHAR(50);
ALTER TABLE transactions MODIFY COLUMN ProductID VARCHAR(50);
ALTER TABLE customers MODIFY COLUMN CustomerID VARCHAR(50);

3.2 Add Primary/Unique Keys

ALTER TABLE customers ADD PRIMARY KEY (CustomerID);

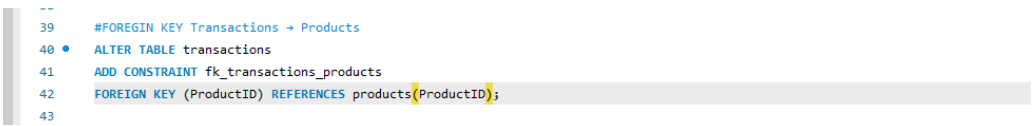
Screenshot: Customers primary key added

3.3 Add Foreign Keys

ALTER TABLE transactions
ADD CONSTRAINT fk_transactions_customers
FOREIGN KEY (CustomerID)
REFERENCES customers(CustomerID);

ALTER TABLE transactions
ADD CONSTRAINT fk_transactions_products
FOREIGN KEY (ProductID)
REFERENCES products(ProductID);

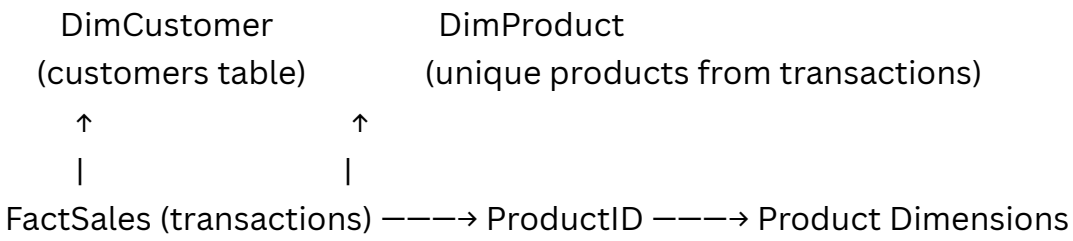
Screenshot: Foreign key constraints



```sql
#FOREGIN KEY Transactions → Customers
ALTER TABLE transactions
MODIFY COLUMN CustomerID VARCHAR(50);
ALTER TABLE transactions
MODIFY COLUMN ProductID VARCHAR(50);
ALTER TABLE customers
MODIFY COLUMN CustomerID VARCHAR(50);
ALTER TABLE customers
ADD PRIMARY KEY (CustomerID);
ALTER TABLE transactions
ADD CONSTRAINT fk_transactions_customers
FOREIGN KEY (CustomerID) REFERENCES customers(CustomerID);
```

---

4. Star Schema Diagram (Text Representation)



✔ Fact table: transactions
✔ Dimension tables: customers, products (derived)

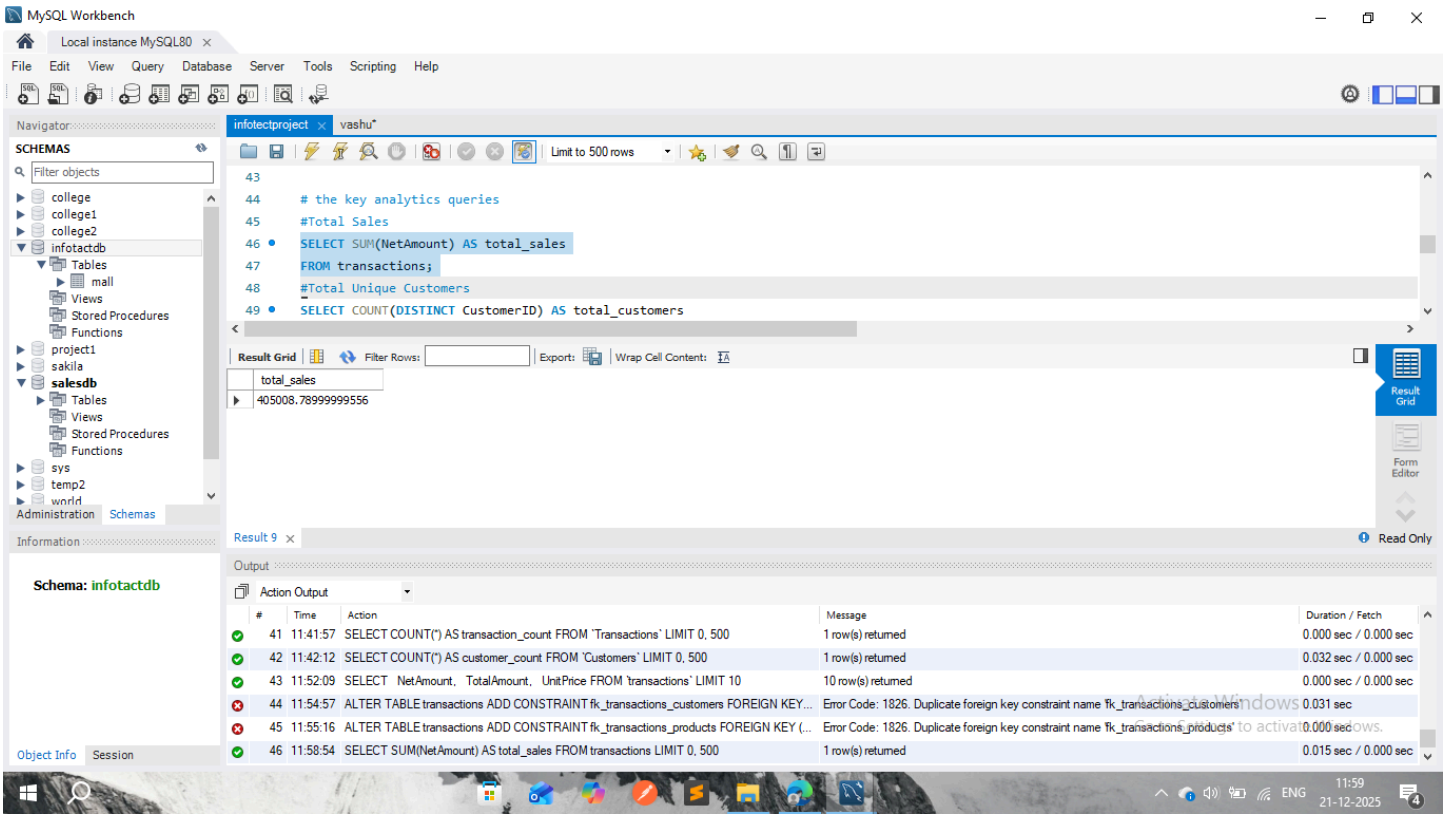
---

5. Basic Business Analytics Queries (Week 2)

5.1 Total Sales Revenue

SELECT SUM(NetAmount) AS total_sales FROM transactions;

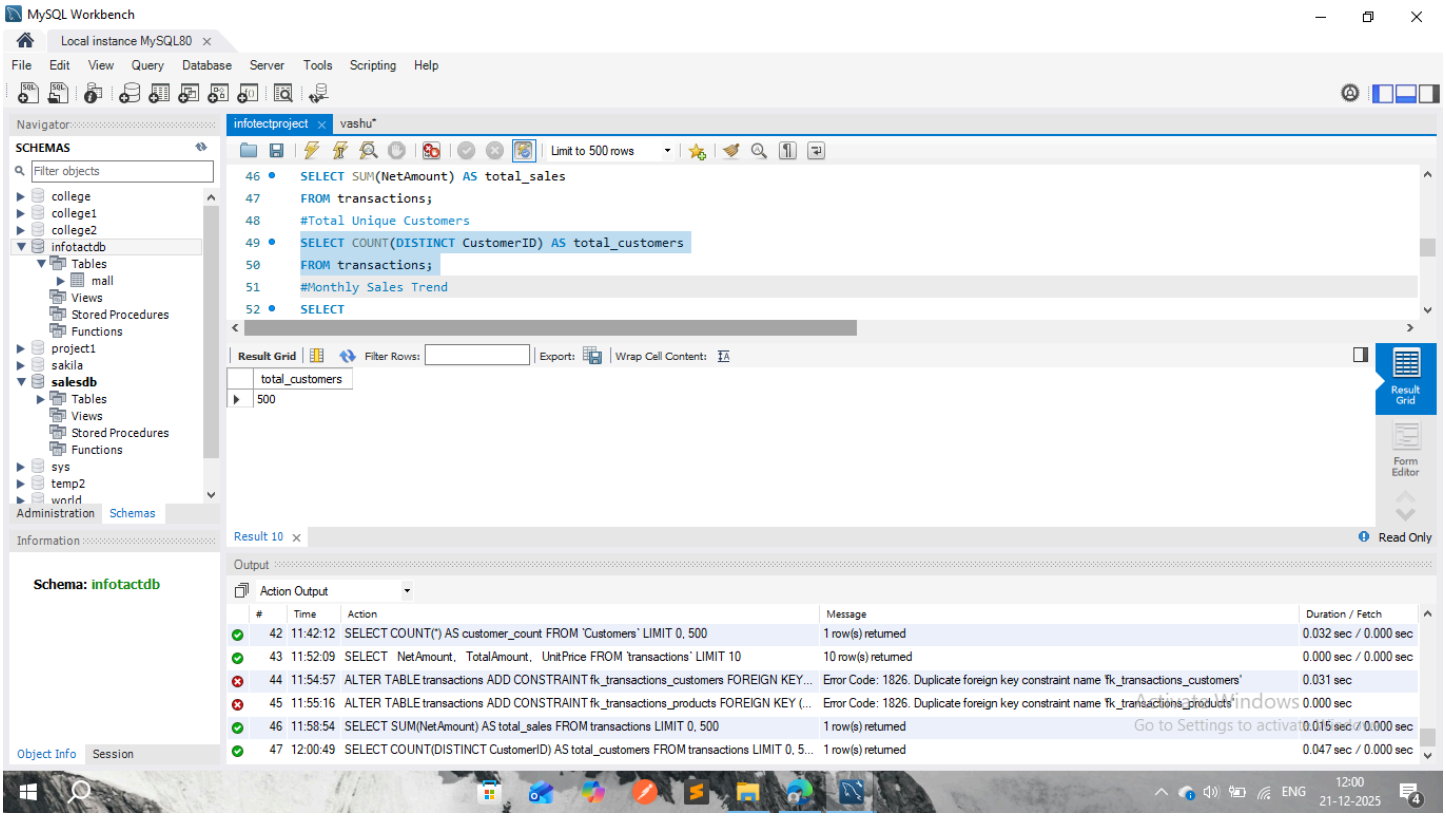
Result



5.2 Total Unique Customers

SELECT COUNT(DISTINCT CustomerID) AS total_customers FROM transactions;

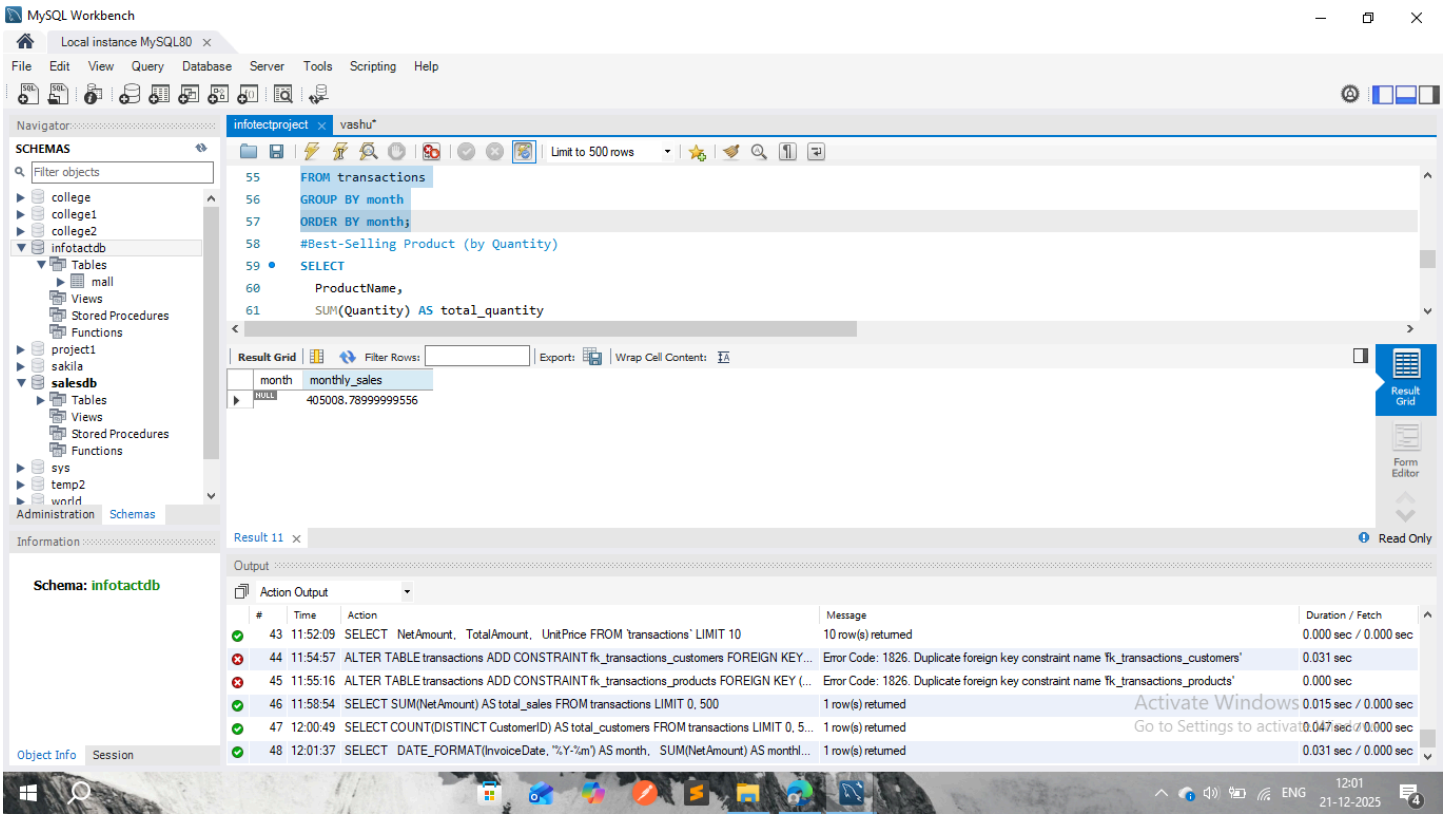
Result



5.3 Monthly Sales Trend (Report)

```
SELECT
  DATE_FORMAT(InvoiceDate, '%Y-%m') AS month,
  SUM(NetAmount) AS monthly_sales
FROM transactions
GROUP BY month
ORDER BY month;
```

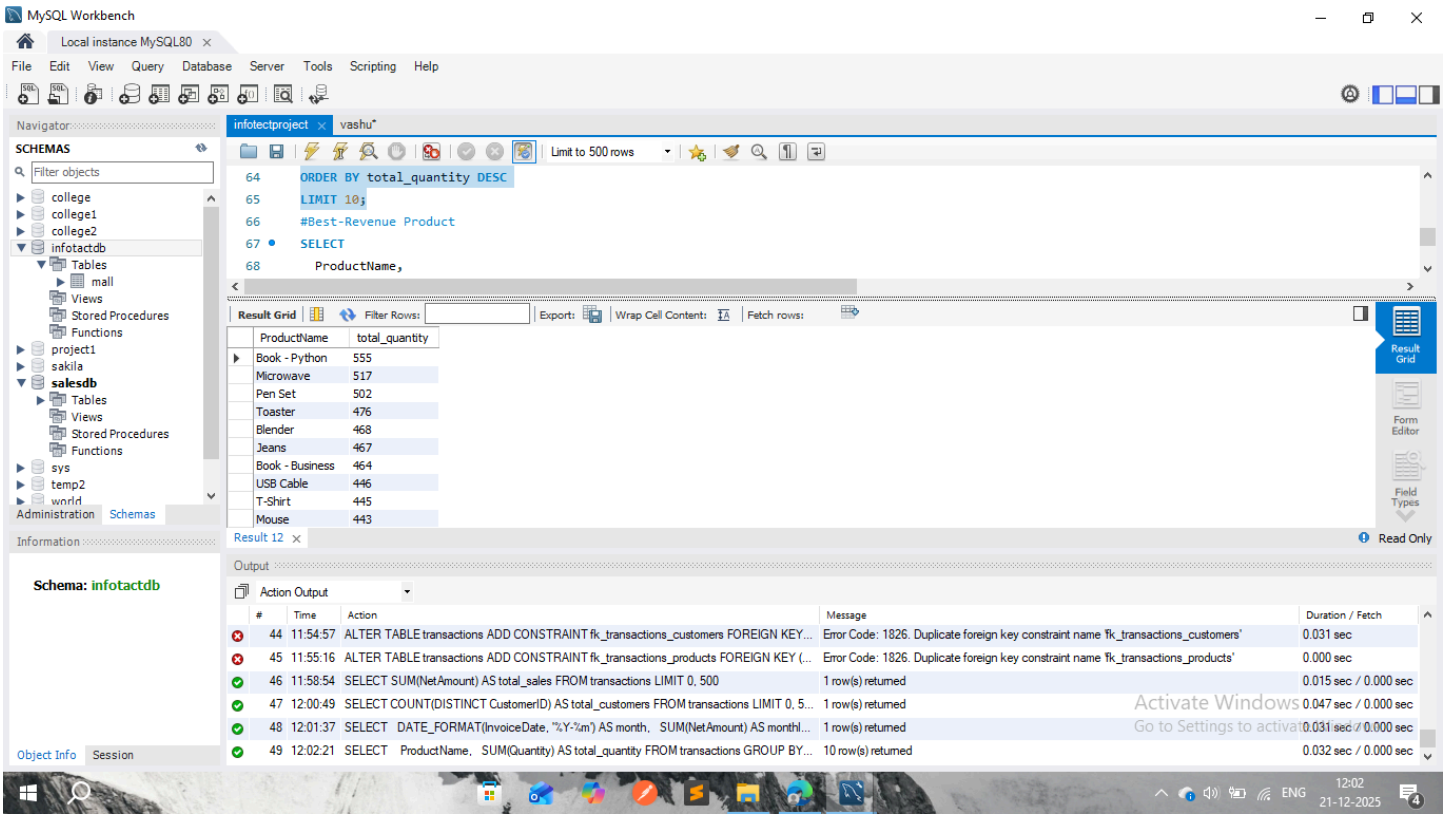
Result



5.4 Best-Selling Product (by Quantity)

```
SELECT
  ProductName,
  SUM(Quantity) AS total_quantity
FROM transactions
GROUP BY ProductName
ORDER BY total_quantity DESC
LIMIT 10;
```

Result



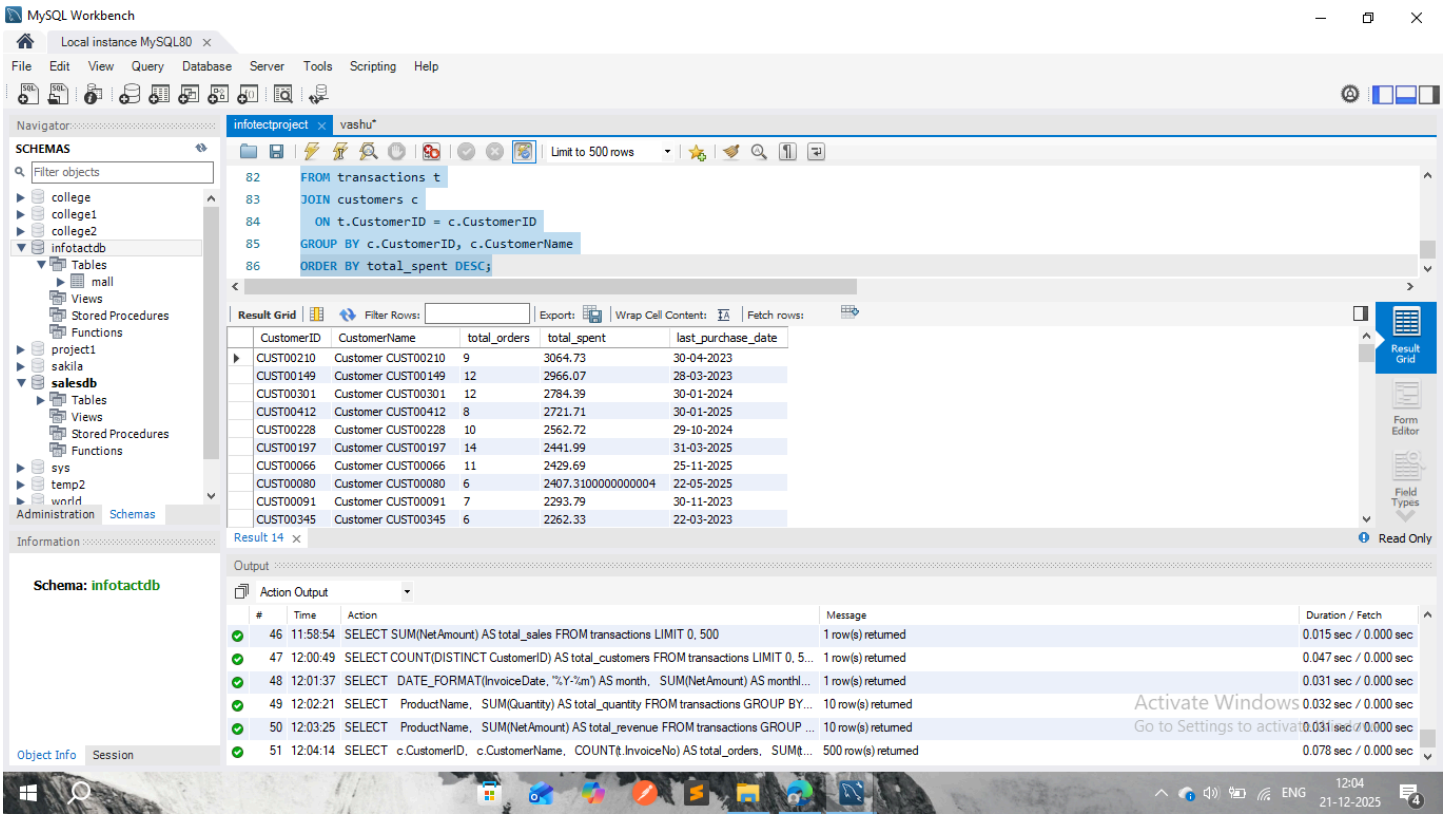
---

6. Single Customer View (Customer 360)

Query:

```
SELECT
  c.CustomerID,
  c.CustomerName,
  COUNT(t.InvoiceNo) AS total_orders,
  SUM(t.NetAmount) AS total_spent,
  MAX(t.InvoiceDate) AS last_purchase_date
FROM transactions t
JOIN customers c
  ON t.CustomerID = c.CustomerID
GROUP BY c.CustomerID, c.CustomerName
ORDER BY total_spent DESC;
```

Result



---

7. Additional Analytics (Optional)

RFM Segment Summary

```
SELECT Customer_Segment,
    COUNT(*) AS customers_in_segment,
    AVG(Monetary) AS avg_spend
FROM rfm_analysis
GROUP BY Customer_Segment;
```

Result

Top Association Rules

```
SELECT *
FROM association_rules
ORDER BY Lift DESC
LIMIT 10;
```

Result

---

8. Validation & Things Checked

Check	Status
Totals in SQL = Totals in CSV/Excel	✓ Verified
No null key values	✓ Verified
Data types corrected (numeric/DATE)	✓ Yes
Foreign keys established	✓ Yes
Queries producing expected results	✓ Yes

---

9. SQL Scripts Used (DDL + Analytics)

(List all scripts you ran; optionally attach a .sql file in your submission.)

- CREATE TABLE ... for all tables
- ALTER TABLE ... to add foreign keys
- Aggregation and analytics SELECT queries

(Include the full script text here or as an appendix.)

📦 Week 2 Final Deliverables Checklist

- ✓ SQL table creation scripts
- ✓ Data loaded into tables successfully
- ✓ Basic analytics SQL queries
- ✓ Single Customer View SQL
- ✓ Screenshots of query results
- ✓ Validation tests and documentation

Summary Statement (for Reviewers)

“In Week 2, we converted cleaned datasets into a star-schema SQL database and performed basic sales and customer analytics, including a Single Customer View for customer-level insights.”

Creating a SQL Command Documentation

This document provides an overview of the SQL commands used to manage and analyze the salesdb database. The commands are grouped into categories for easier reference and

understanding.

Database Setup

Create and Use Database

```
CREATE DATABASE salesdb;
USE salesdb;
```

Data Counting Queries

Count Rows in Tables

```
SELECT COUNT(*) AS transaction_count FROM `Transactions`;
SELECT COUNT(*) AS customer_count FROM `Customers`;
SELECT COUNT(*) AS rfm_rows FROM `RFM Analysis`;
SELECT COUNT(*) AS segment_rows FROM `Segment Summary`;
SELECT COUNT(*) AS rules_count FROM `Association Rules`;
```

Data Retrieval

Sample Data from Transactions

```
SELECT
  NetAmount,
  TotalAmount,
  UnitPrice
FROM `transactions`
LIMIT 10;
```

Describe Table Structures

```
DESCRIBE `transactions`;
DESCRIBE `customers`;
DESCRIBE `rfm_analysis`;
DESCRIBE `segment_summary`;
DESCRIBE `association_rules`;
```

Data Integrity Checks

Check for NULL Keys

```
SELECT COUNT(*) FROM transactions WHERE CustomerID IS NULL;
SELECT COUNT(*) FROM transactions WHERE ProductID IS NULL;
SELECT COUNT(*) FROM customers WHERE CustomerID IS NULL;
```

Foreign Key Constraints

Define Foreign Keys

```
# Adjust column types for Foreign Key Constraints
ALTER TABLE transactions
MODIFY COLUMN CustomerID VARCHAR(50);
ALTER TABLE transactions
MODIFY COLUMN ProductID VARCHAR(50);
ALTER TABLE customers
MODIFY COLUMN CustomerID VARCHAR(50);

# Add Primary Key and Foreign Keys
ALTER TABLE customers
ADD PRIMARY KEY (CustomerID);

ALTER TABLE transactions
ADD CONSTRAINT fk_transactions_customers
FOREIGN KEY (CustomerID) REFERENCES customers(CustomerID);

ALTER TABLE transactions
ADD CONSTRAINT fk_transactions_products
FOREIGN KEY (ProductID) REFERENCES products(ProductID);
```

Analytical Queries

Key Analytics

Total Sales

```
SELECT SUM(NetAmount) AS total_sales
FROM transactions;
```

Total Unique Customers

```
SELECT COUNT(DISTINCT CustomerID) AS total_customers
FROM transactions;
```

Monthly Sales Trend

```
SELECT
  DATE_FORMAT(InvoiceDate, '%Y-%m') AS month,
  SUM(NetAmount) AS monthly_sales
FROM transactions
```

```
GROUP BY month
ORDER BY month;
```

Best-Selling Product (by Quantity)

```
SELECT
  ProductName,
  SUM(Quantity) AS total_quantity
FROM transactions
GROUP BY ProductName
ORDER BY total_quantity DESC
LIMIT 10;
```

Best-Revenue Product

```
SELECT
  ProductName,
  SUM(NetAmount) AS total_revenue
FROM transactions
GROUP BY ProductName
ORDER BY total_revenue DESC
LIMIT 10;
```

Single Customer View (Customer 360)

```
SELECT
  c.CustomerID,
  c.CustomerName,
  COUNT(t.InvoiceNo) AS total_orders,
  SUM(t.NetAmount) AS total_spent,
  MAX(t.InvoiceDate) AS last_purchase_date
FROM transactions t
JOIN customers c
  ON t.CustomerID = c.CustomerID
GROUP BY c.CustomerID, c.CustomerName
ORDER BY total_spent DESC;
```

This document serves as a comprehensive guide to the SQL commands for managing and analyzing data within the salesdb database. These commands facilitate the creation of structured insights and ensure data integrity.