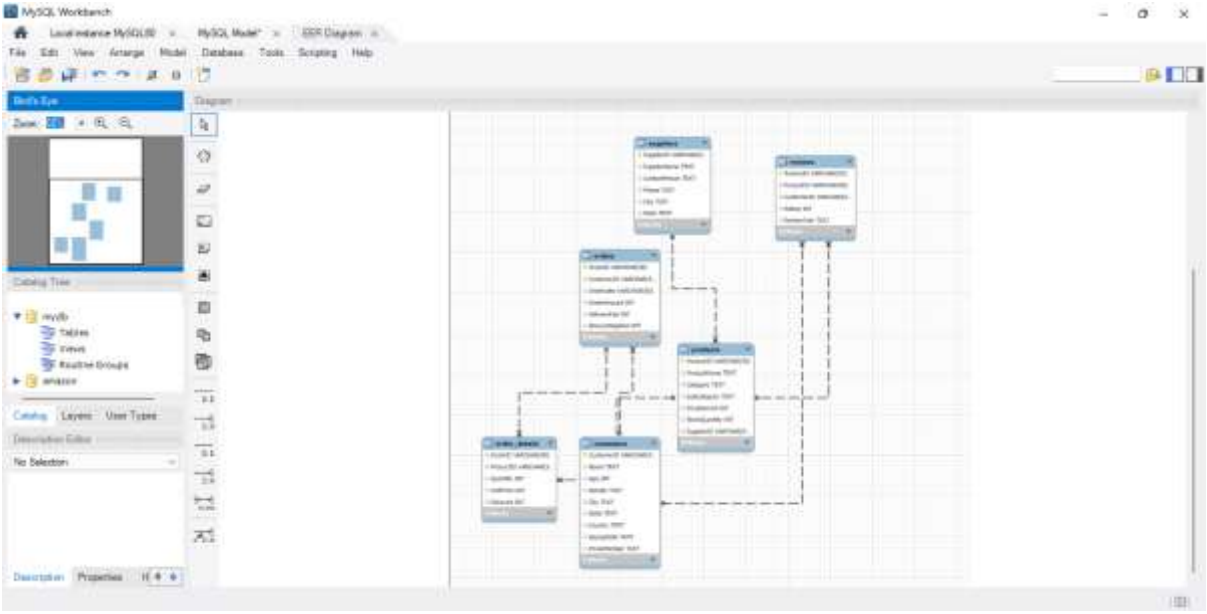
Amazon Fresh Analytics

Problem Statement:

Amazon Fresh is an e-commerce platform specializing in groceries and daily essentials. As the platform expands, its management seeks to optimize operations, enhance customer satisfaction, and boost revenue. The challenge lies in effectively managing large volumes of data, including customer profiles, product inventories, supplier information, orders, and customer reviews. The problem is to design a relational database for Amazon Fresh that organizes this data efficiently, allows for meaningful analysis, and supports critical business decisions. This project aims to solve key business problems, such as identifying top-performing products, analyzing customer purchasing patterns, and ensuring inventory management aligns with demand trends.

Task 1: ER Diagram Creation

The ER diagram was created in MySQL Workbench to represent the Amazon Fresh database structure. It includes entities such as Customers, Products, Orders, Suppliers, Order_Details, and Reviews, showing their relationships through primary and foreign keys.

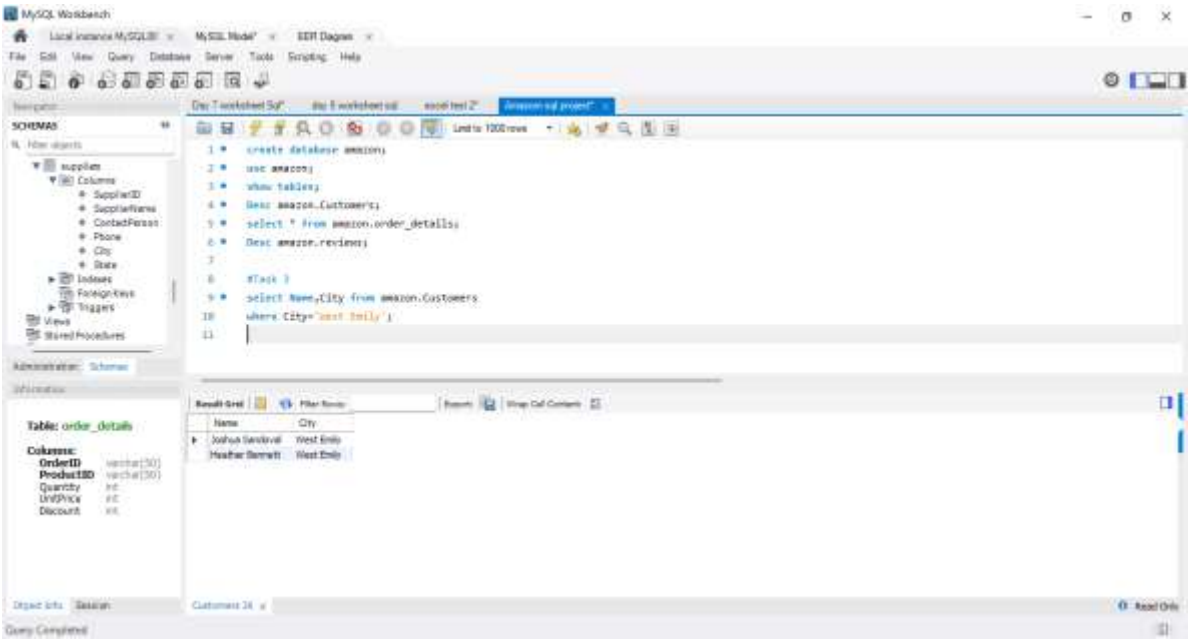


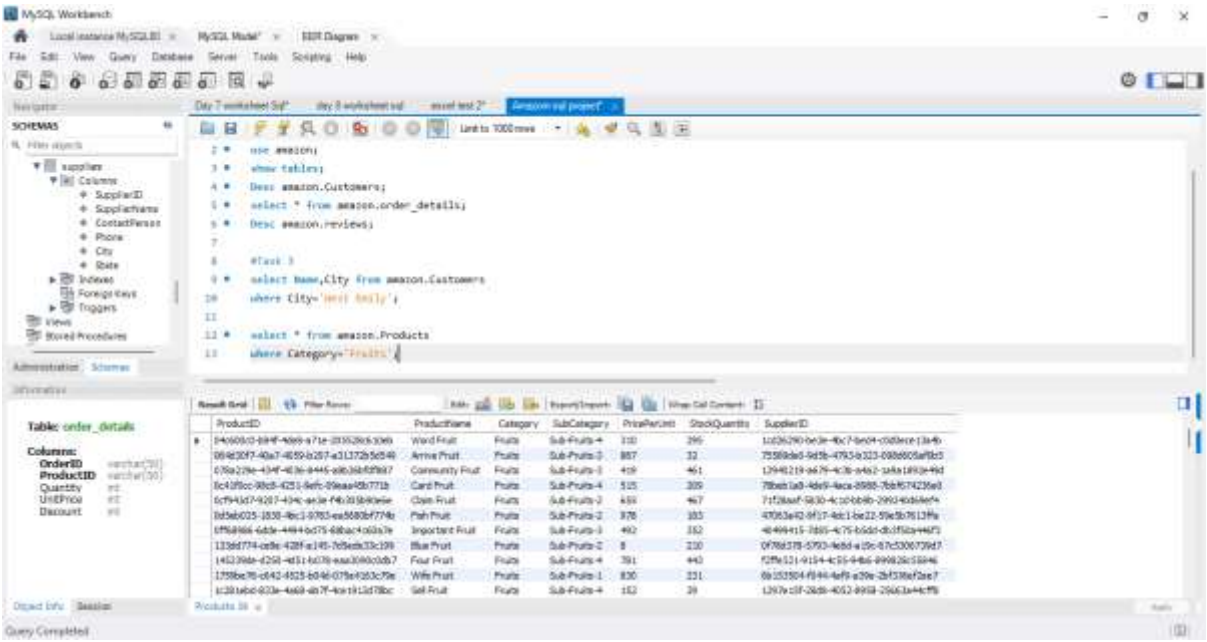
Task 2: Identifying Keys and Relationships

In this step, primary keys and foreign keys were identified for each table. Relationships were defined — for example, CustomerID links Customers and Orders, while ProductID connects Products and Order_Details.

Task 3: Basic Select Queries

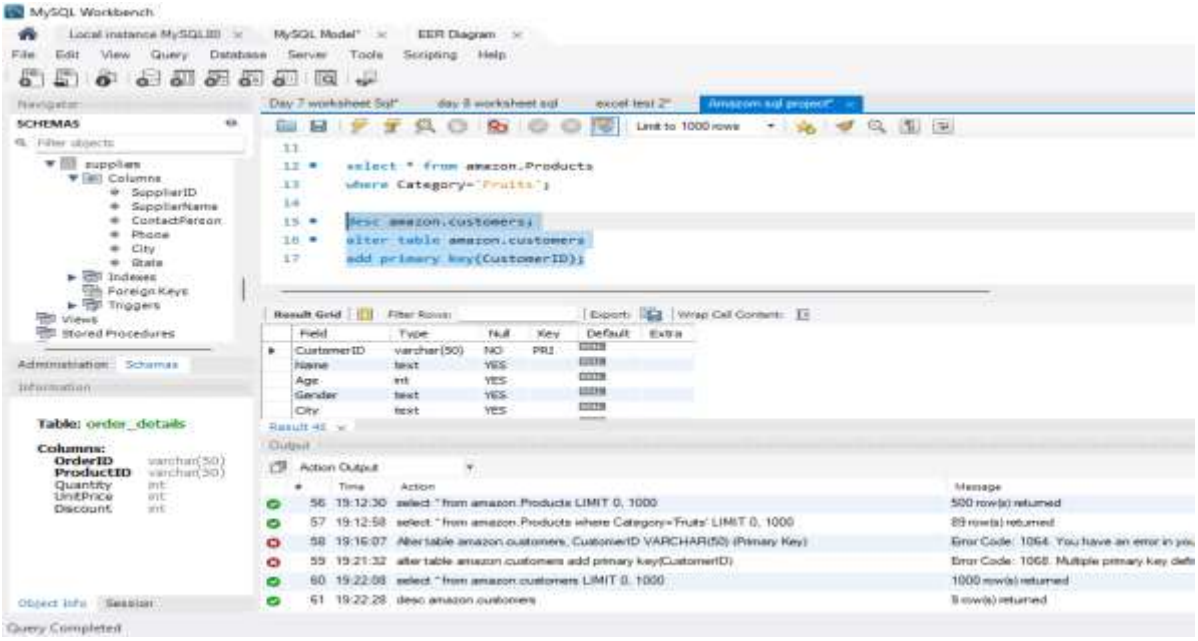
SQL queries were written to retrieve all customers from a specific city and to fetch all products under the ‘Fruits’ category. The results confirm correct filtering based on WHERE conditions.

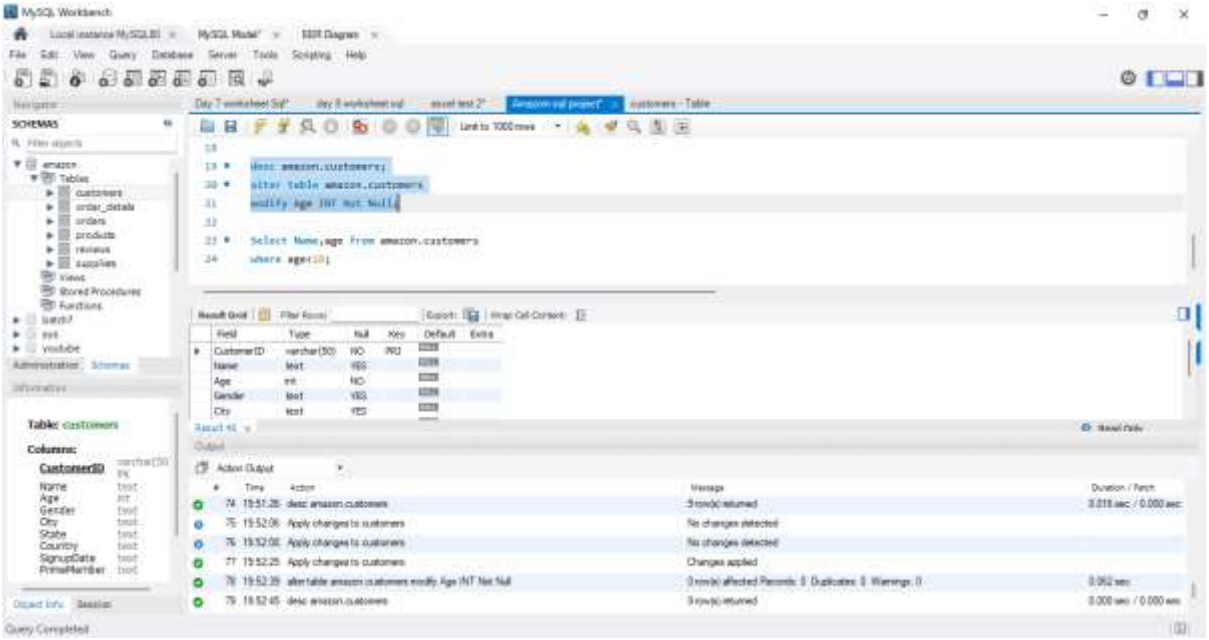


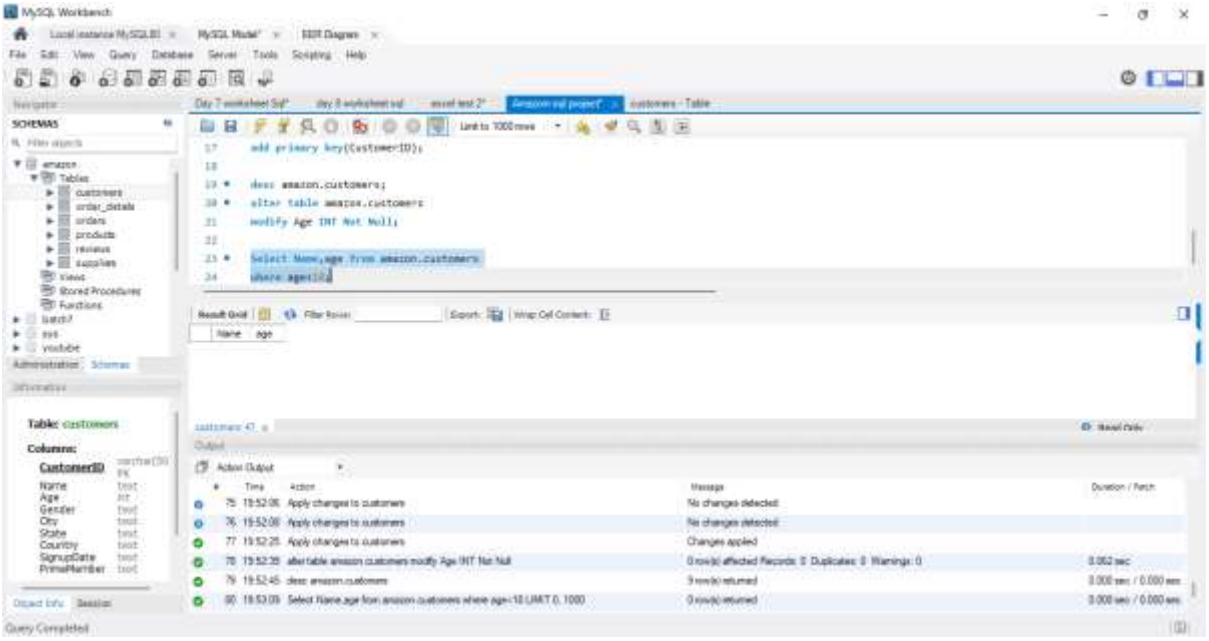


Task 4: Creating Customers Table with Constraints (DDL)

The Customers table was created using DDL statements, applying constraints such as PRIMARY KEY (CustomerID), CHECK constraint (Age > 18), NOT NULL (Age), and UNIQUE constraint (Name).

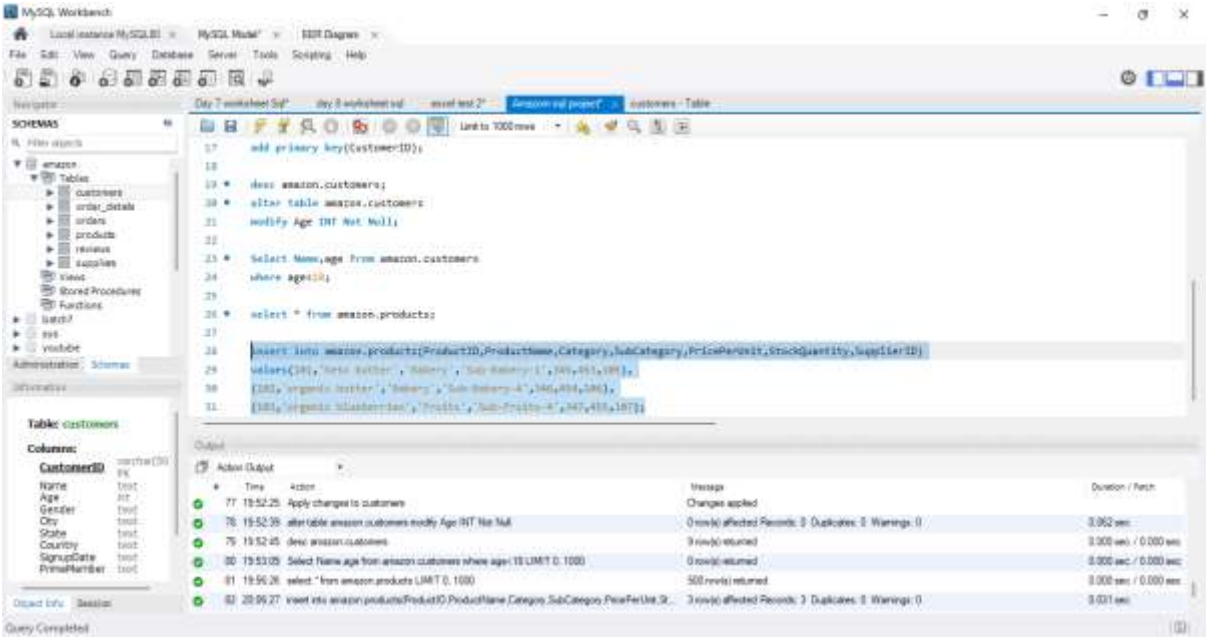

MySQL Workbench
select * from amazon.Products
where Category='Fruits';
desc amazon.customers;
alter table amazon.customers
add primary key(CustomerID);
Table: order_details
Columns:
OrderID varchar(50)
ProductID varchar(30)
Quantity int
UnitPrice int
Discount int
Query Completed


MySQL Workbench
desc amazon.customers;
alter table amazon.customers
modify Age INT Not Null;
Select Name,age from amazon.customers
where age<10;
Table: customers
Columns:
CustomerID
Name
Age
Gender
City
State
Country
SignupDate
PrimeMember
Query Completed



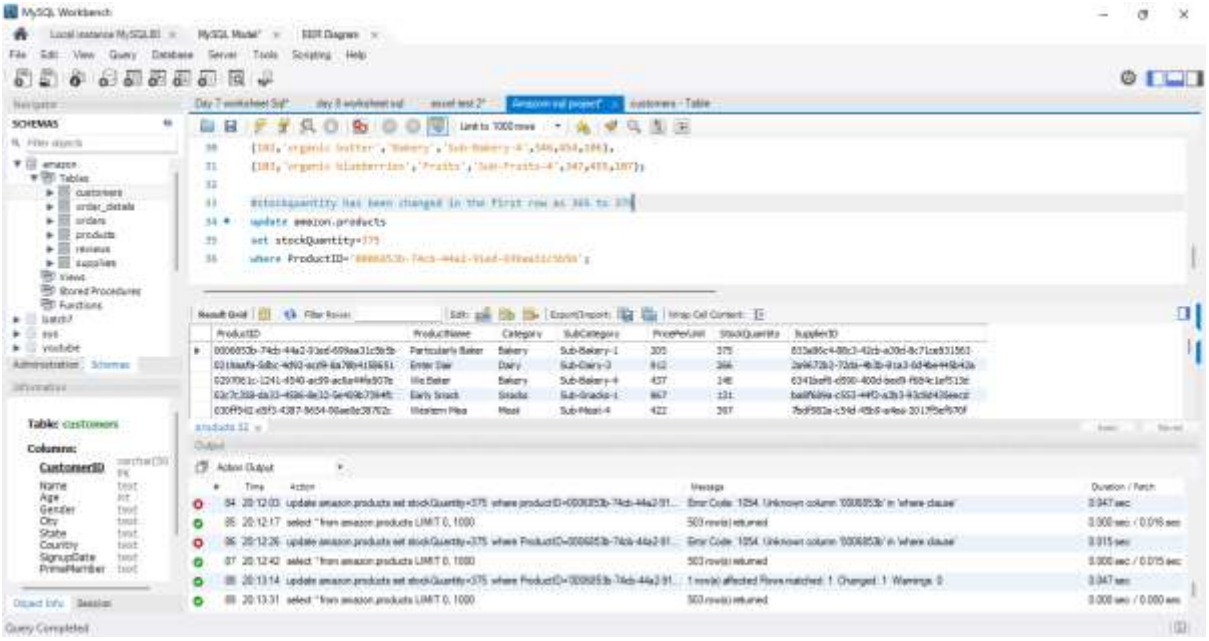
Task 5: Inserting Records (DML)

Three new records were inserted into the Products table using INSERT statements. Each record includes ProductID, ProductName, Category, Price, and StockQuantity.



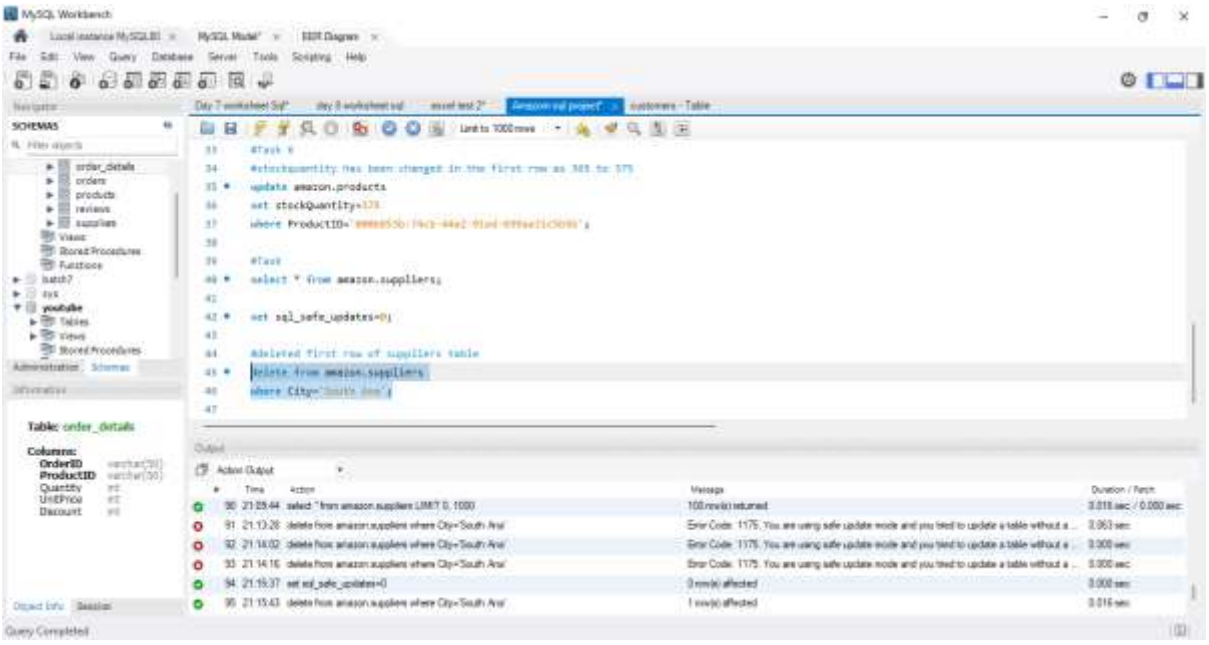
Task 6: Updating Product Stock

An UPDATE statement was executed to modify the stock quantity of a specific product using the ProductID condition. The result shows the successful update of the product's inventory.



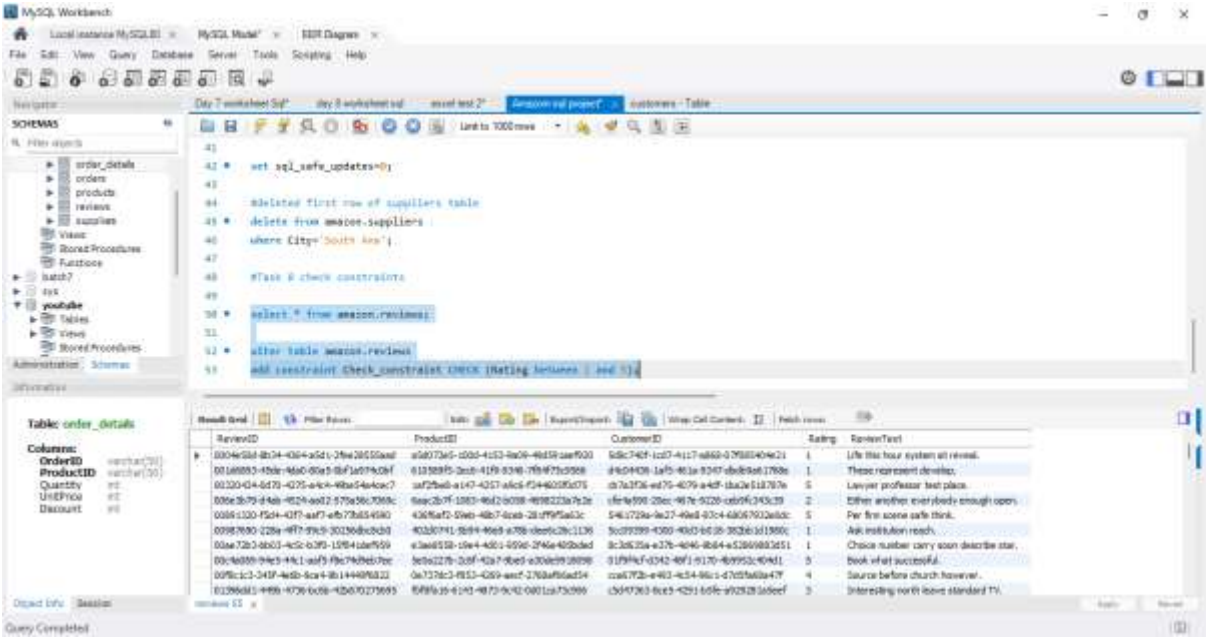
Task 7: Deleting Supplier Record

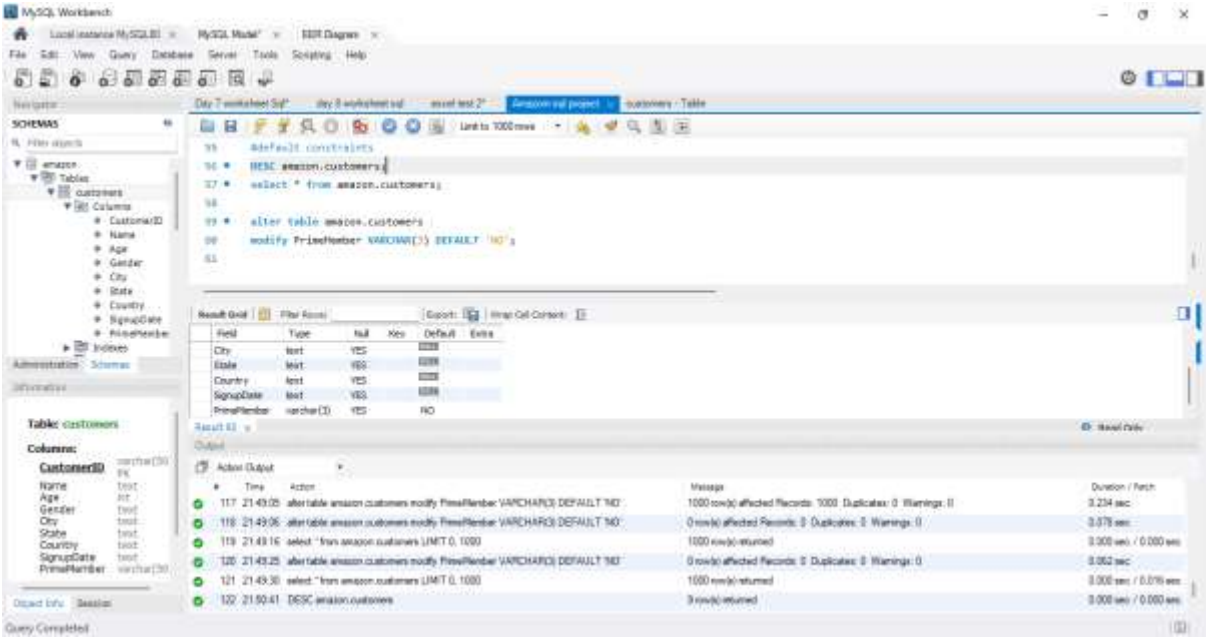
A DELETE statement was used to remove a supplier from the Suppliers table based on their city. This demonstrates the use of conditional deletion in SQL.



Task 8: Adding SQL Constraints

Constraints were added using ALTER TABLE commands — a CHECK constraint ensures ratings are between 1 and 5 in the Reviews table, and a DEFAULT constraint was added to the PrimeMember column (default: ‘No’).

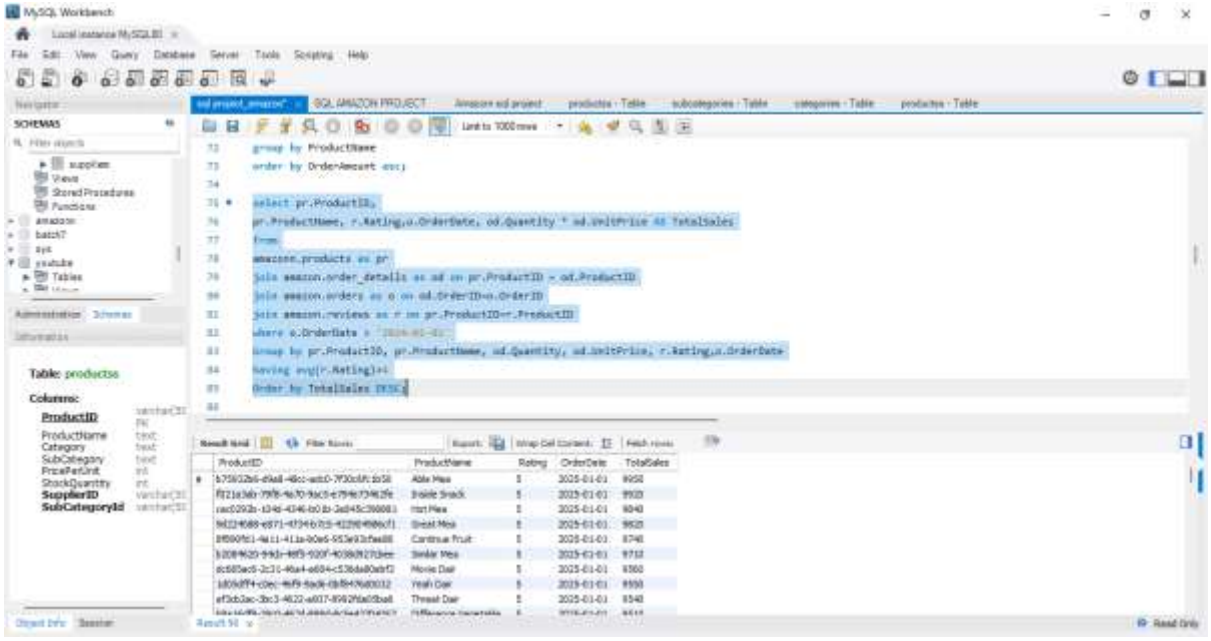




Task 9: Using WHERE, GROUP BY, HAVING, ORDER BY Clauses

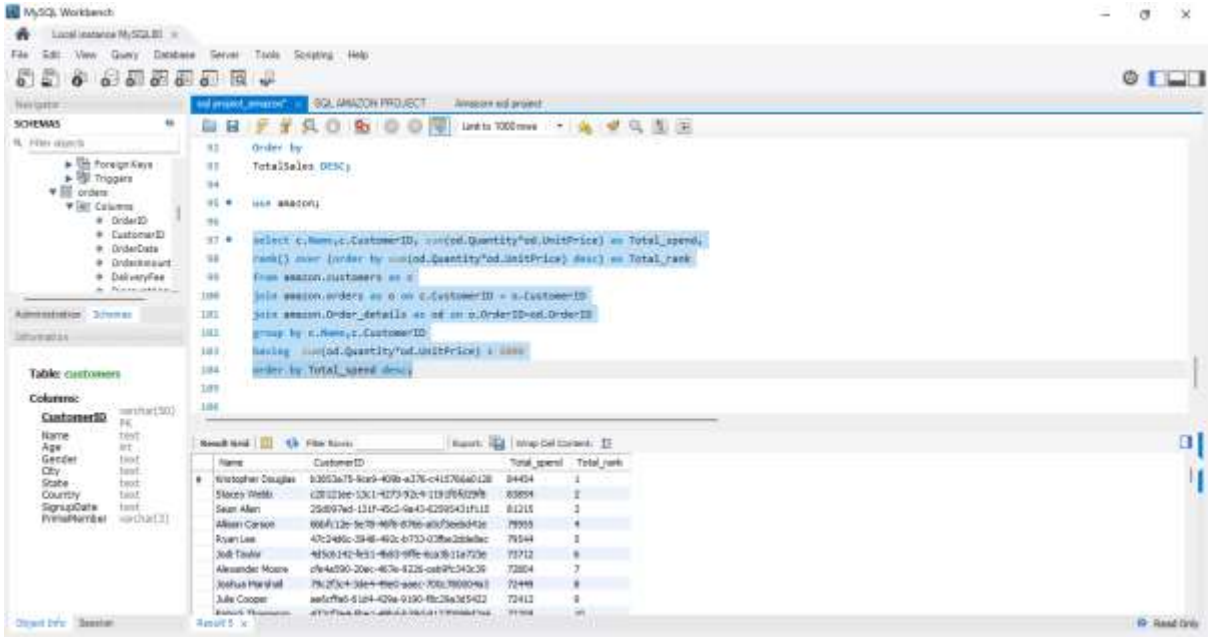
Queries were written to:

- Retrieve orders placed after 2024-01-01 (WHERE).
- List products with an average rating above 4 (HAVING).
- Rank products by total sales (GROUP BY and ORDER BY).



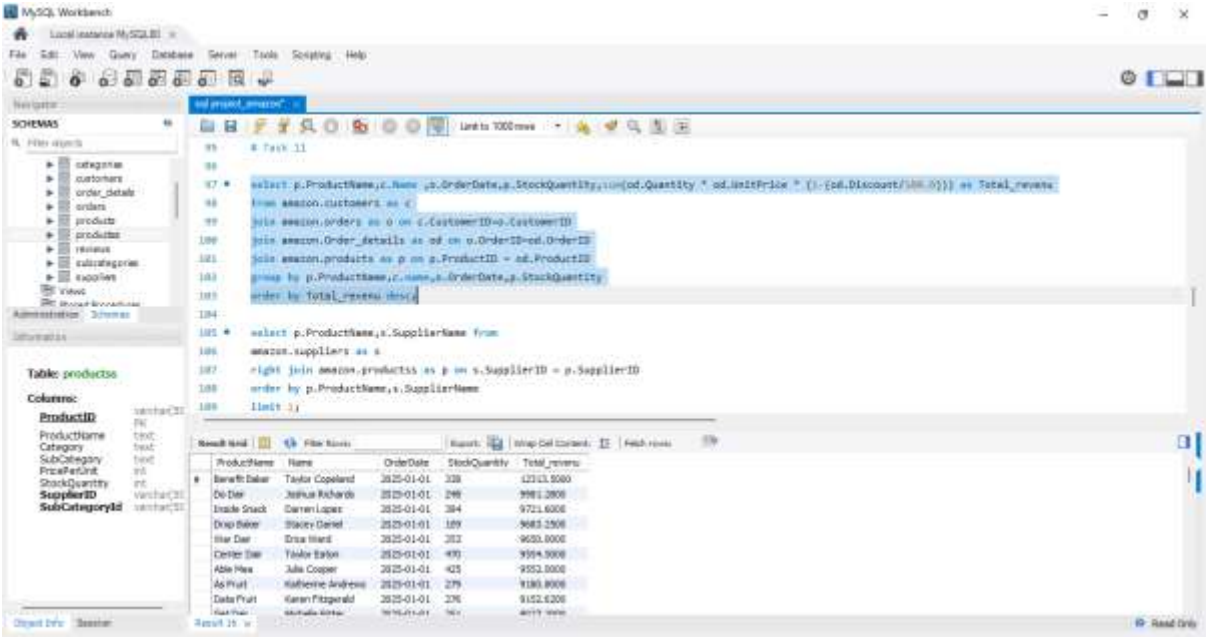
Task 10: Identifying High-Value Customers

Calculations were performed to determine total spending per customer. Customers were ranked using ORDER BY and filtered to find those who spent more than ₹5,000.



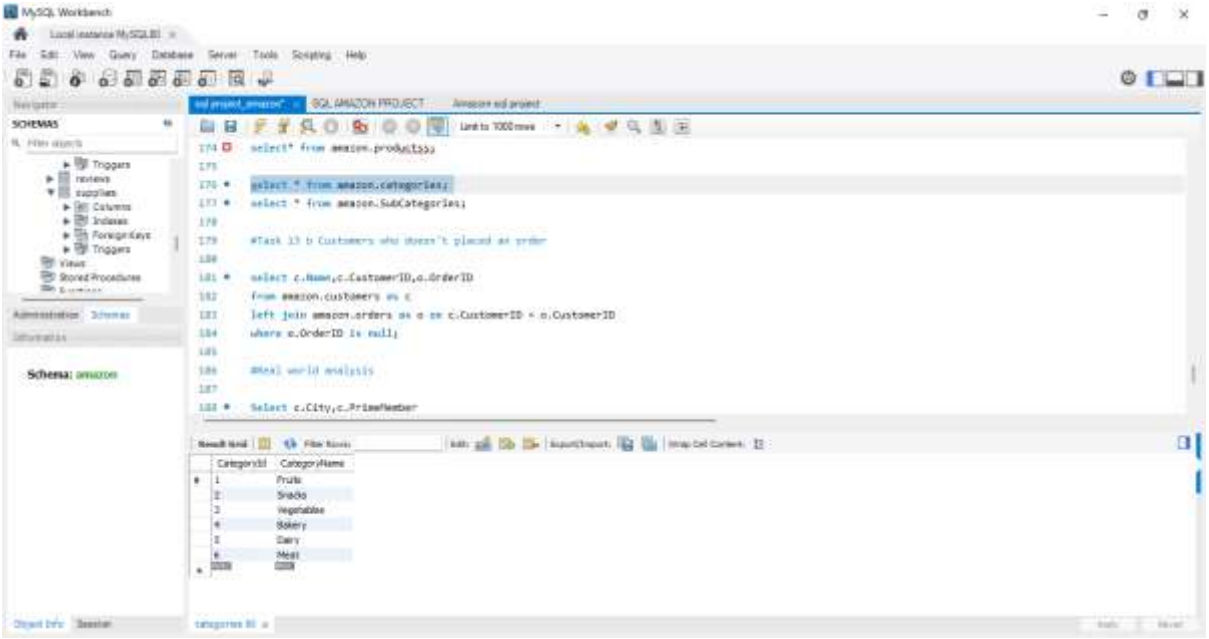
Task 11: Complex Joins and Aggregations

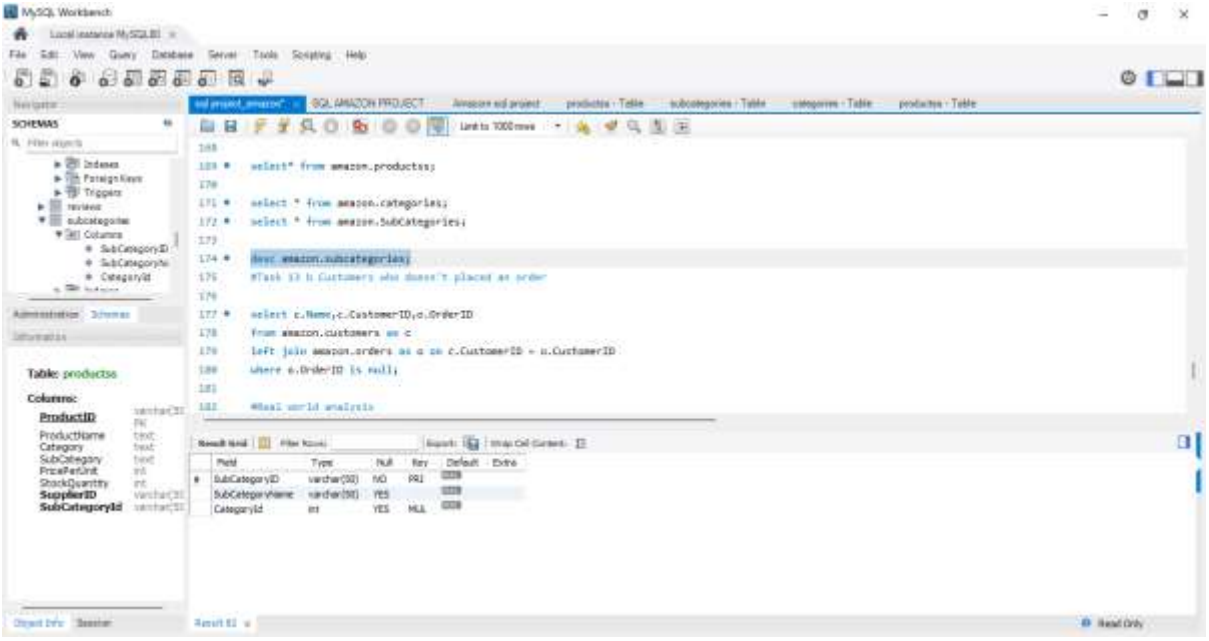
JOIN operations were used between Orders and Order_Details tables to compute total revenue per order. Additional queries identified top customers and suppliers with maximum stock.

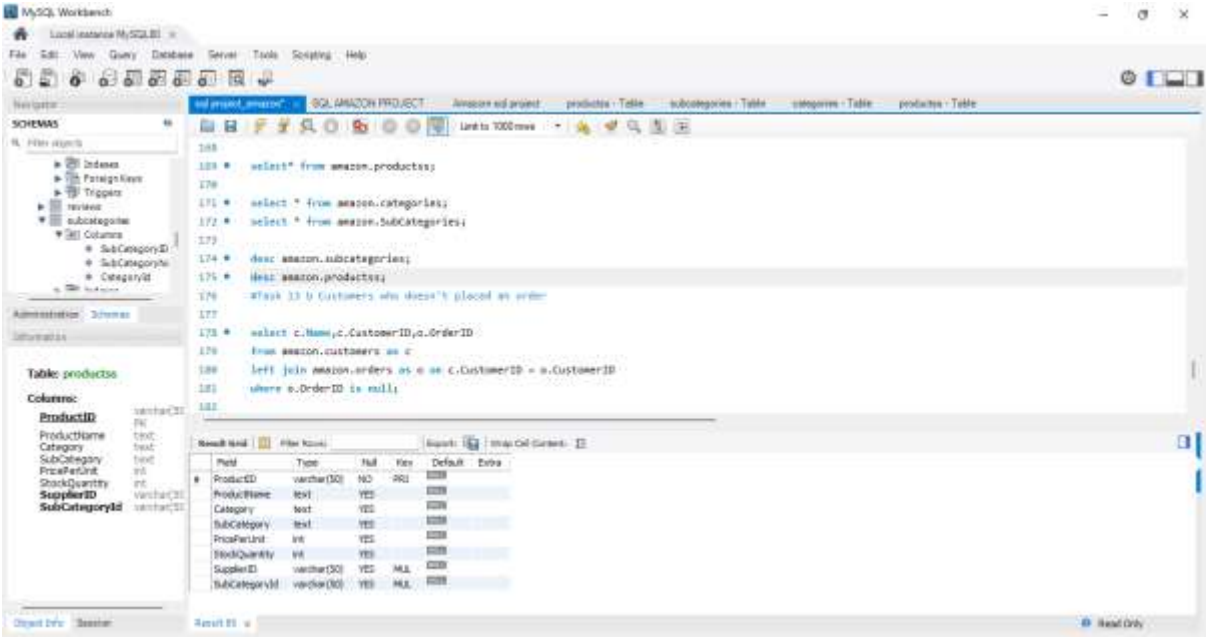


Task 12: Normalization to 3NF

The Products table was normalized by separating category and subcategory details into a new table. Foreign keys were added to maintain referential integrity, ensuring a well-structured 3NF database.



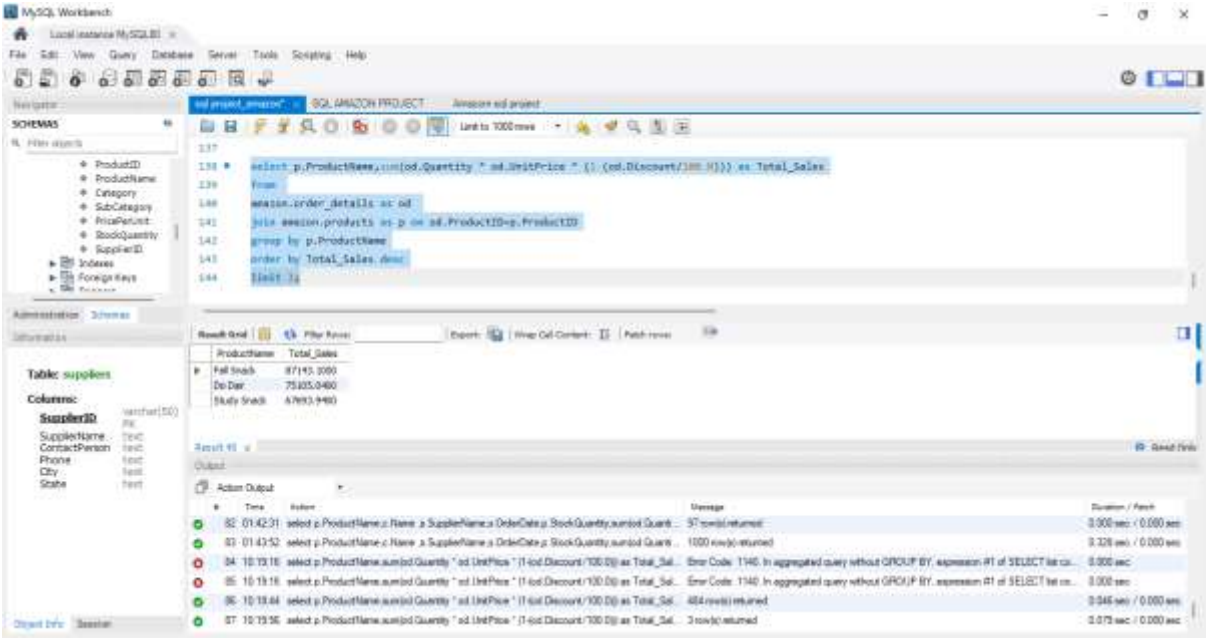


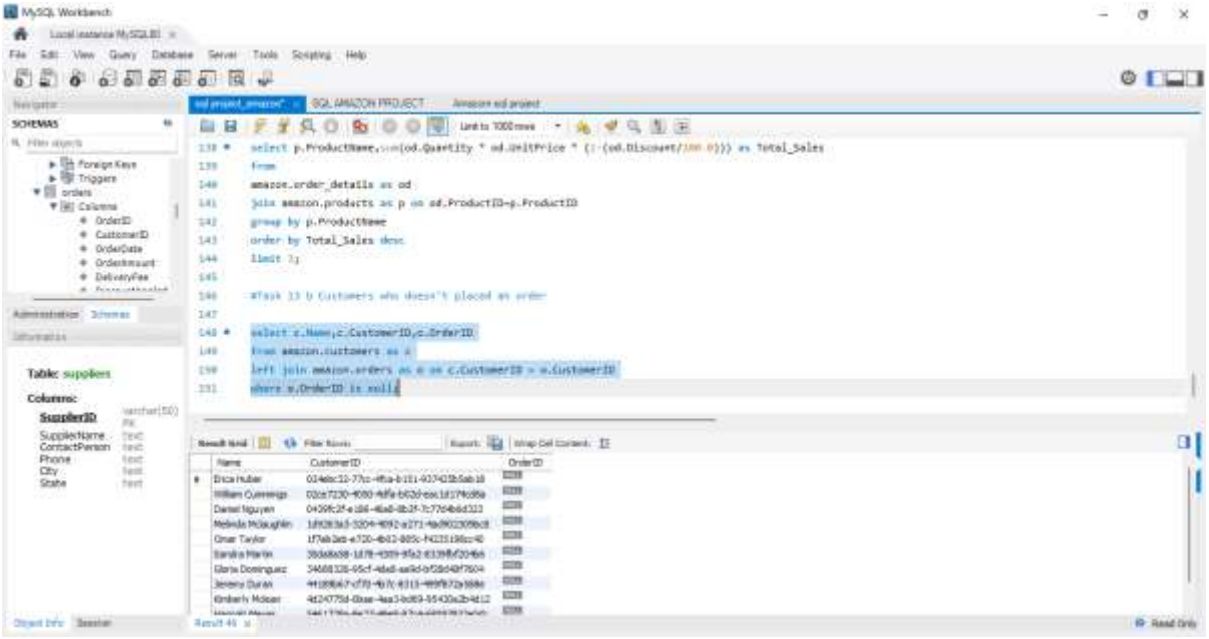


Task 13: Subqueries and Nested Queries

Subqueries were used to identify:

- Top 3 products based on total sales revenue.
- Customers who have not placed any orders.
  This demonstrates advanced SQL techniques for data analysis.

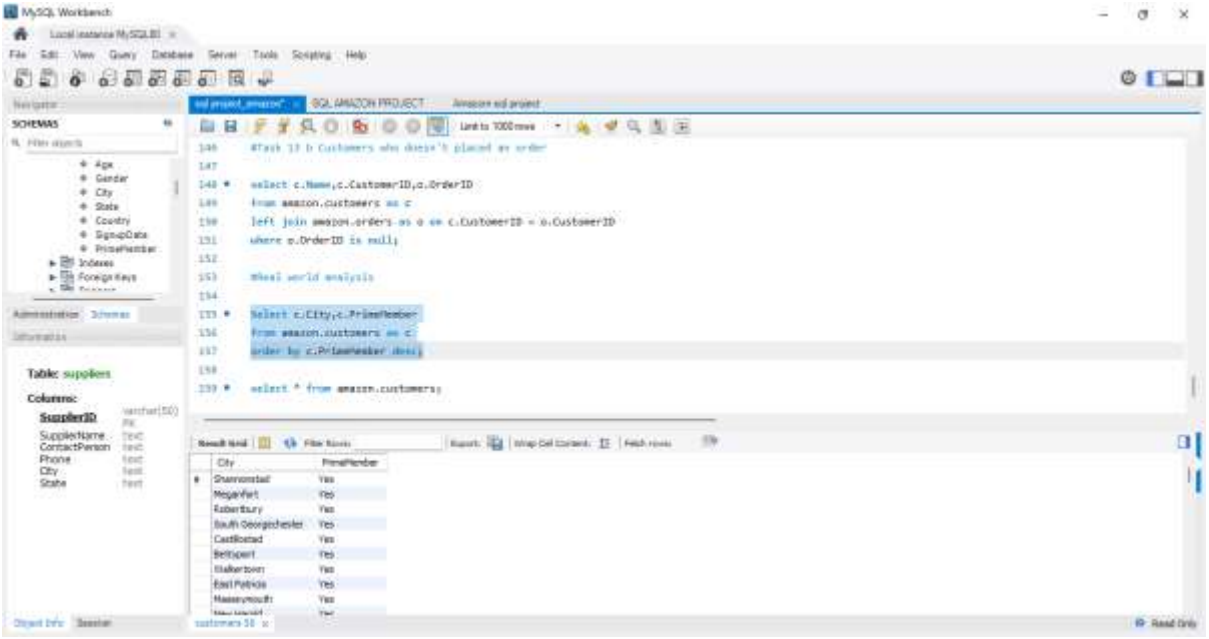


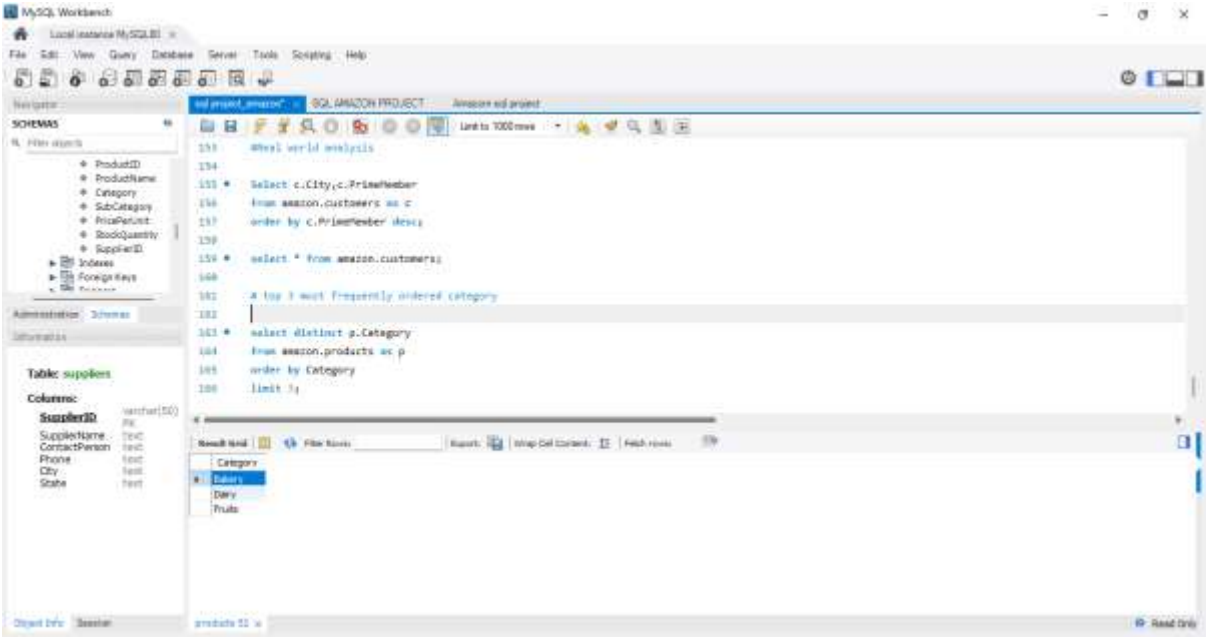


Task 14: Real-World Analysis

Analytical queries were written to find:

- Cities with the highest number of Prime Members.
- The top 3 most frequently ordered product categories.
  These insights help understand customer trends and business performance.


MySQL Workbench
SCHEMAS
Table: suppliers
Columns:
SupplierID
SupplierName
ContactPerson
Phone
City
State
#Task 13 b Customers who dosn't placed an order
select c.Name,c.CustomerID,o.OrderID
from amazon.customers as c
left join amazon.orders as o on c.CustomerID = o.CustomerID
where o.OrderID is null;
#Real world analysis
Select c.City,c.PrimeMember
from amazon.customers as c
order by c.PrimeMember desc;
select * from amazon.customers;
City
PrimeMember
Yes


MySQL Workbench
SCHEMAS
Table: suppliers
Columns:
SupplierID
SupplierName
ContactPerson
Phone
City
State
#Real world analysis
Select c.City,c.PrimeMember
from amazon.customers as c
order by c.PrimeMember desc;
select * from amazon.customers;
top 3 most frequently ordered category
select distinct p.Category
from amazon.products as p
order by Category
limit 3;
Category
Bakery
Dairy
Fruits