

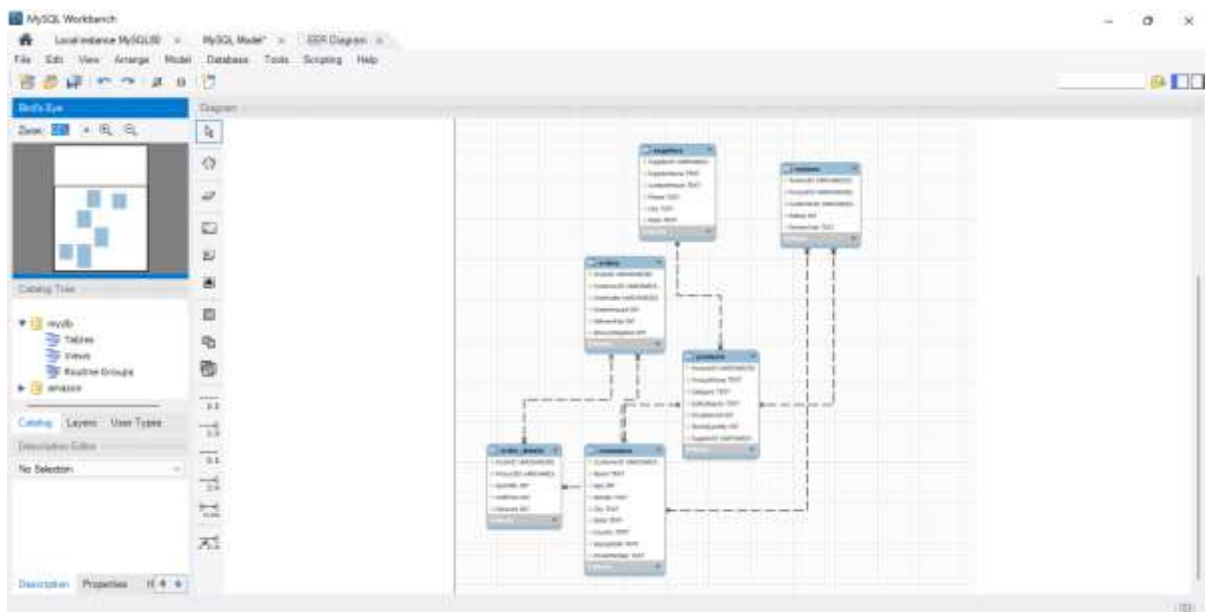
Amazon Fresh Analytics

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.

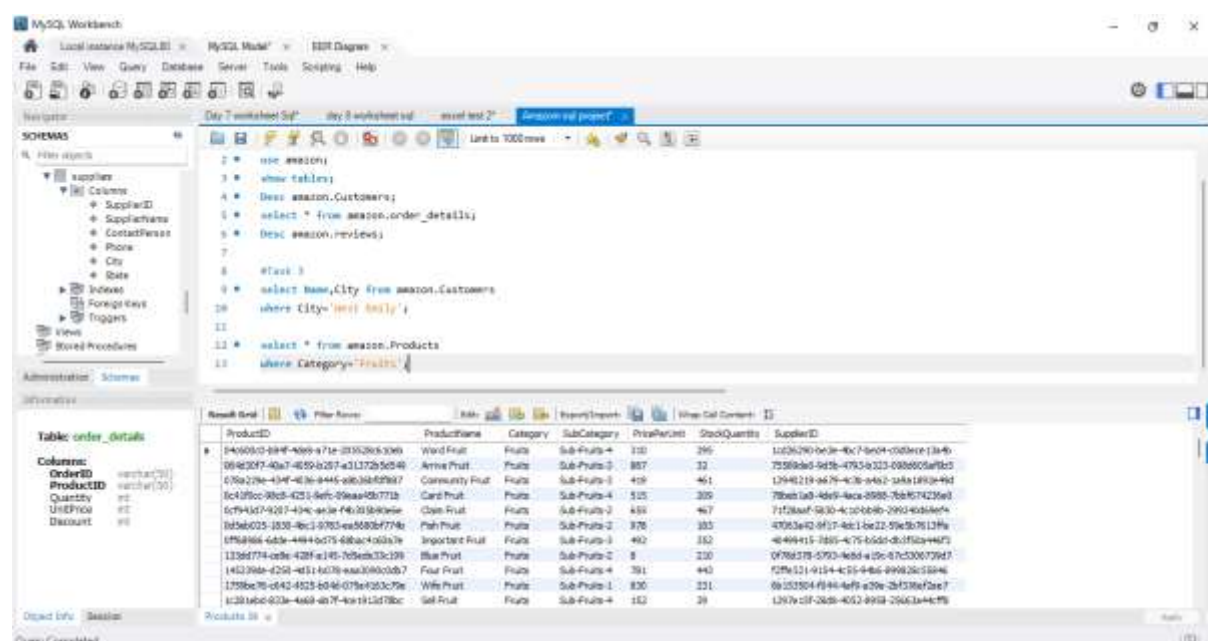
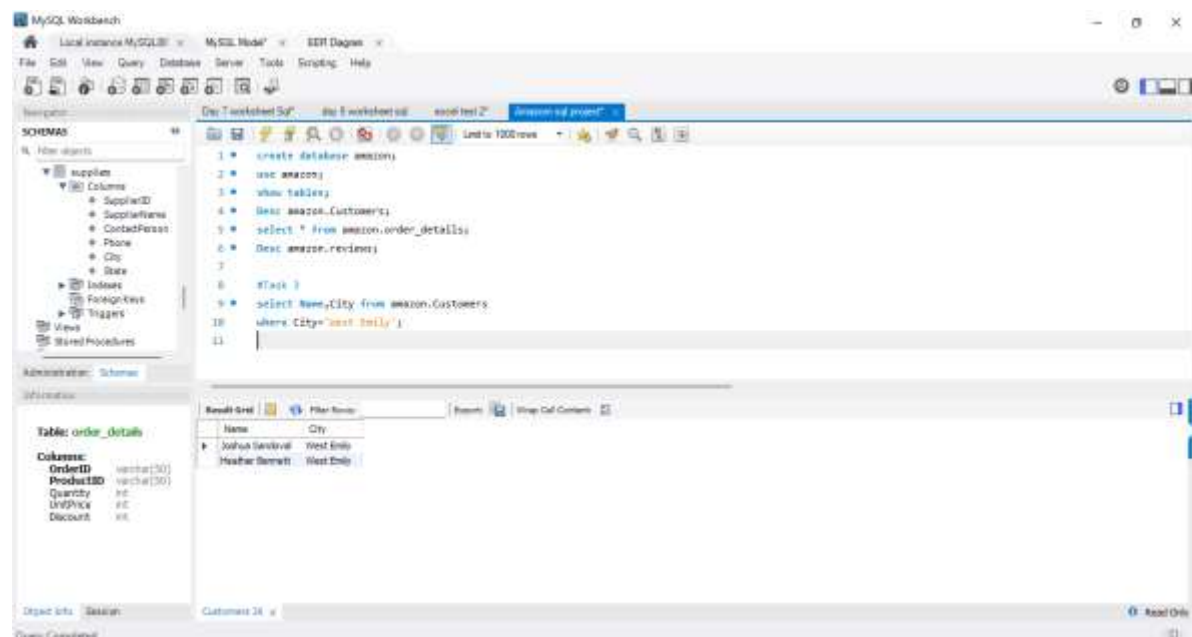


Task 2: Identifying Keys and Relationships

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

Local instance MySQL80 x MySQL Model x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

Day 7 worksheet.sql day 8 worksheet.sql excel test 2 Amazon.sql project

Link to 1000 rows

1.1
1.2 *
1.3 select * from amazon.Products
1.4 where Category="Fruits";
1.5 *
1.6 desc amazon.customers;
1.7 alter table amazon.customers
add primary key(CustomerID);

Result Grid

Field	Type	Null	Key	Default	Extra
CustomerID	varchar(50)	NO	PK1		
Name	text	YES			
Age	int	YES			
Gender	text	YES			
City	text	YES			

Result 42

Output

Action Output

Time	Action	Message
56 19:12:30	select * from amazon.Products LIMIT 0, 1000	500 row(s) returned
57 19:12:58	select * from amazon.Products where Category="Fruits" LIMIT 0, 1000	89 row(s) returned
58 19:16:07	Alter table amazon.customers, CustomerID VARCHAR(50) (Primary Key)	Error Code: 1064: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'Alter table amazon.customers, CustomerID VARCHAR(50) (Primary Key)' at line 1
59 19:21:32	alter table amazon.customers add primary key(CustomerID)	Error Code: 1068: Multiple primary key defined
60 19:22:08	select * from amazon.customers LIMIT 0, 1000	1000 row(s) returned
61 19:22:28	desc amazon.customers	8 row(s) returned

Query Completed

MySQL Workbench

Local instance MySQL80 x MySQL Model x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

Day 7 worksheet.sql day 8 worksheet.sql excel test 2 Amazon.sql project customers - Table

Link to 1000 rows

18
19 *
20 desc amazon.customers;
21 alter table amazon.customers
modify Age INT Not Null;
22
23 *
24 select Name, age from amazon.customers
where age < 10;

Result Grid

Field	Type	Null	Key	Default	Extra
CustomerID	varchar(50)	NO	PK1		
Name	text	YES			
Age	int	NO			
Gender	text	YES			
City	text	YES			

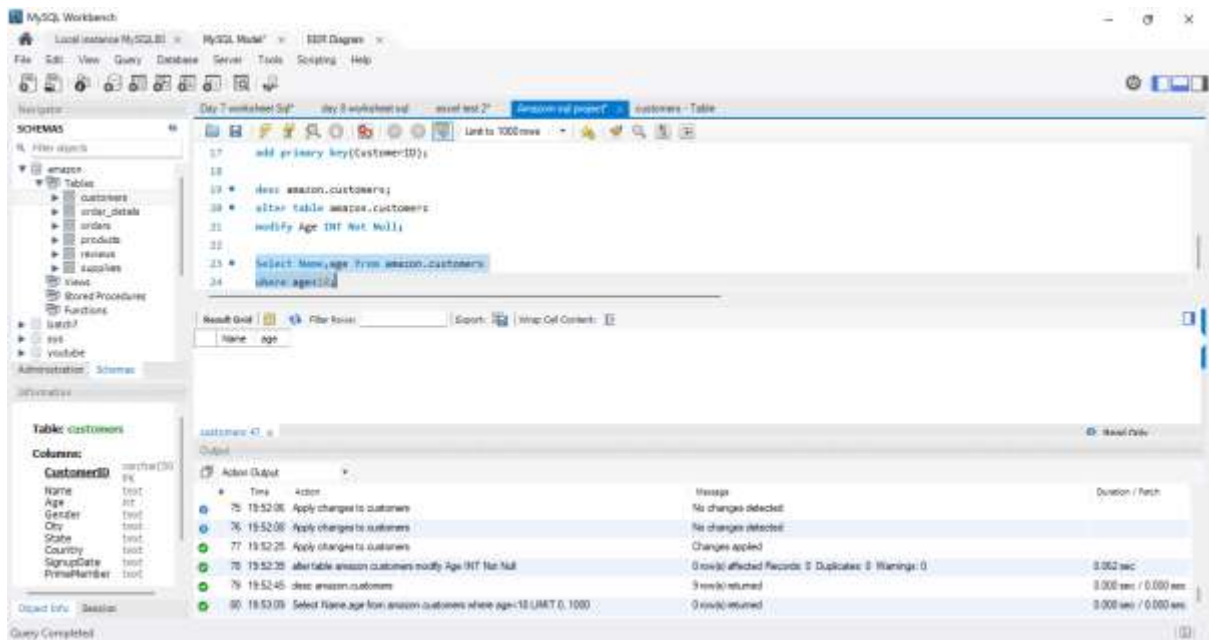
Result 43

Output

Action Output

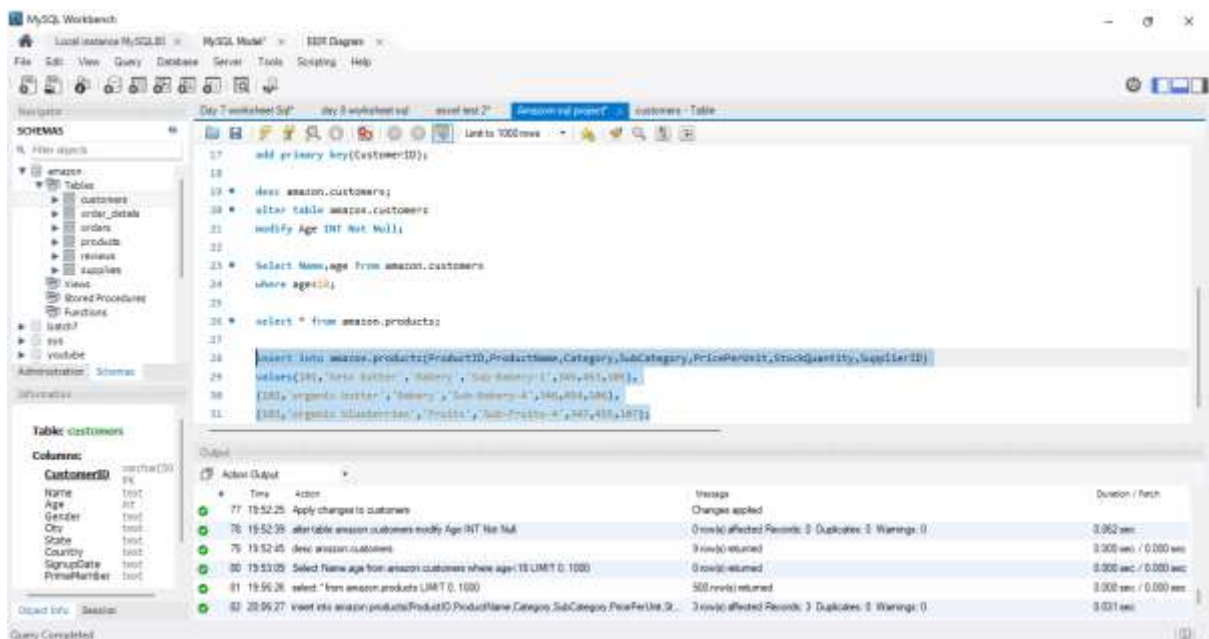
Time	Action	Message	Duration / Pctn
74 19:51:28	desc amazon.customers	8 row(s) returned	0.018 sec / 0.000 sec
75 19:52:08	Apply changes to customers	No changes detected	
76 19:52:08	Apply changes to customers	No changes detected	
77 19:52:25	Apply changes to customers	Changes applied	
78 19:52:39	alter table amazon.customers modify Age INT Not Null	3 row(s) affected; Records: 0 Duplicates: 0 Warnings: 0	0.062 sec
79 19:52:45	desc amazon.customers	8 row(s) returned	0.008 sec / 0.000 sec

Query Completed



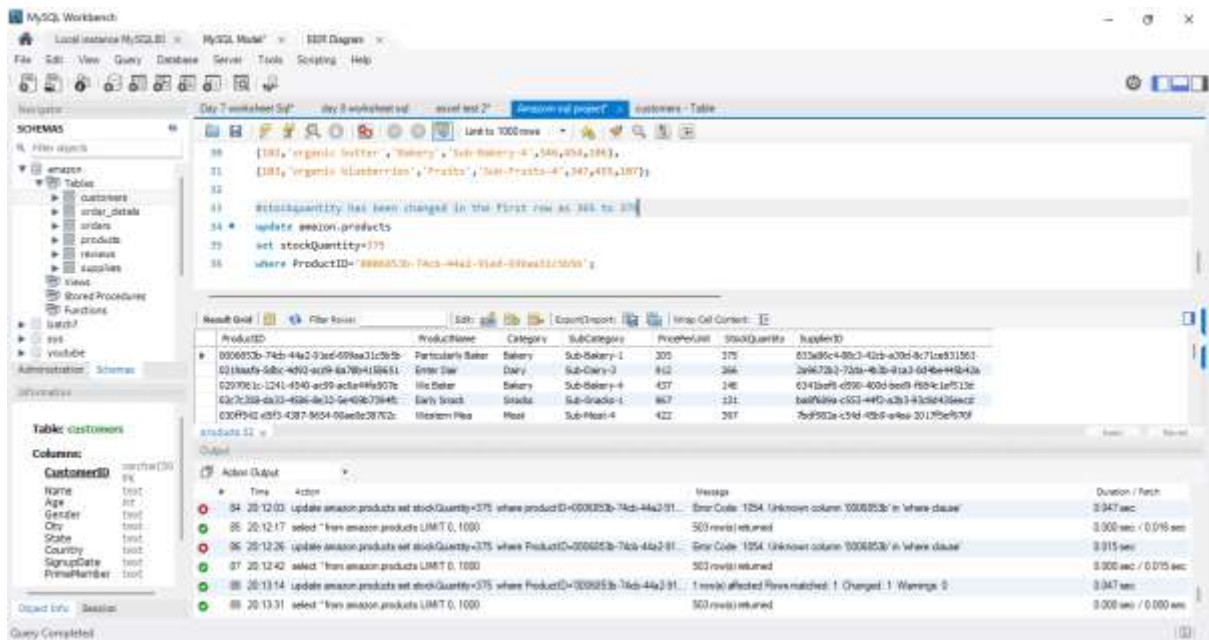
Task 5: Inserting Records (DML)

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



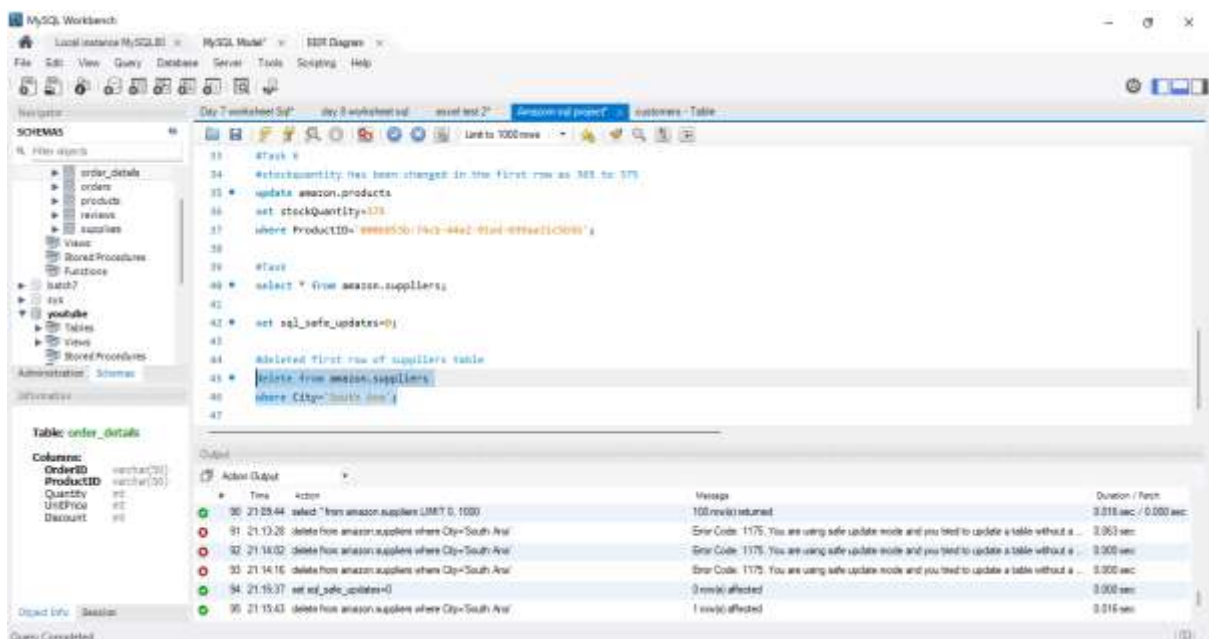
Task 6: Updating Product Stock

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.



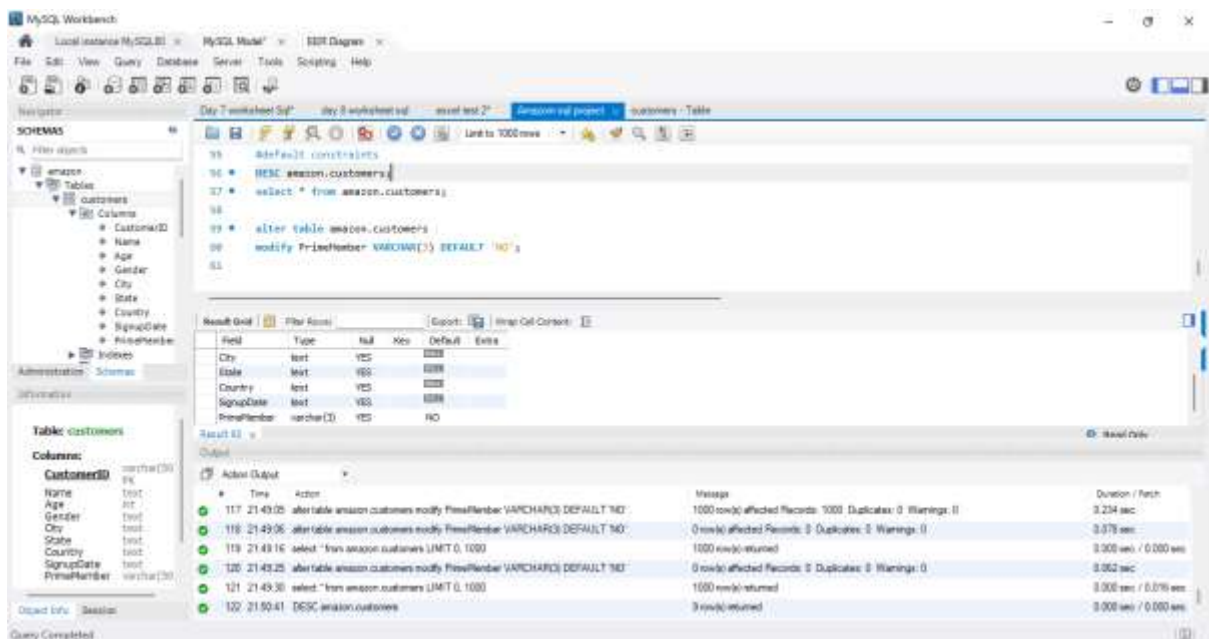
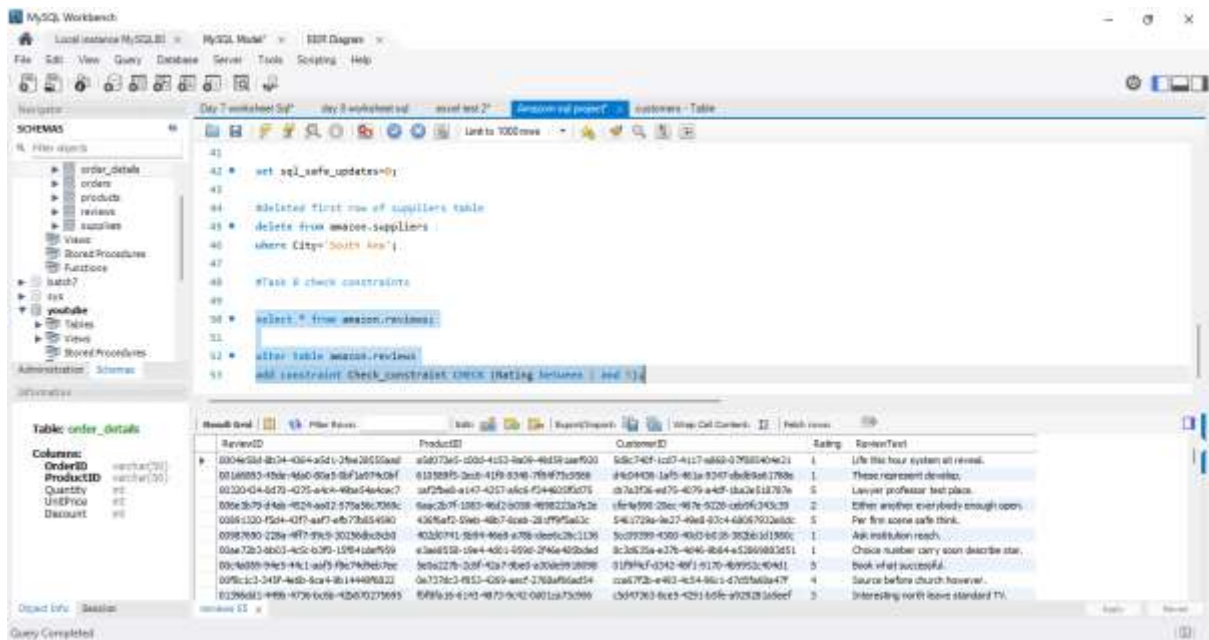
Task 7: Deleting Supplier Record

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.

```

72 group by ProductName
73 order by OrderAmount asc)
74
75 select pr.ProductID,
76 pr.ProductName, r.Rating, o.OrderDate, od.Quantity * od.UnitPrice as TotalSales
77 from
78 amazon.products as pr
79 join amazon.order_details as od on pr.ProductID = od.ProductID
80 join amazon.orders as o on od.OrderID = o.OrderID
81 join amazon.reviews as r on pr.ProductID = r.ProductID
82 where o.OrderDate > '2024-01-01'
83 group by pr.ProductID, pr.ProductName, od.Quantity, od.UnitPrice, r.Rating, o.OrderDate
84 having avg(r.Rating) > 4
85 order by TotalSales DESC
86

```

ProductID	ProductName	Rating	OrderDate	TotalSales
1759326-4546-48c-act0-730c0f7d58	Able Miso	5	2025-01-01	9950
721a36b-799b-4670-bac0-e794c73462e	Snack Miso	5	2025-01-01	9900
rac092b-4346-4346-b0-2d-3d45c36063	Hot Miso	5	2025-01-01	8040
86124685-e871-4734b-055-4228e486c0f	Snack Miso	5	2025-01-01	9600
86060f01-4a11-411a-00a5-053a32fa688	Candied Fruit	5	2025-01-01	8740
33084620-9463-4495-000f-4038b2734ee	Snack Miso	5	2025-01-01	9710
cc582ac3-2c31-40a4-e604-c536da0a0f2	Miso Dair	5	2025-01-01	8360
180687f4-c0c6-4a59-8a3b-08847600312	Yeast Dair	5	2025-01-01	8990
ef7ab3ac-3bc3-4622-4607-89a2f6a02ba8	Yeast Dair	5	2025-01-01	8540
1a6a449b-567a-4d4d-888a-03a4719b4c79	Pythagorean	5	2025-01-01	8410

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.

```

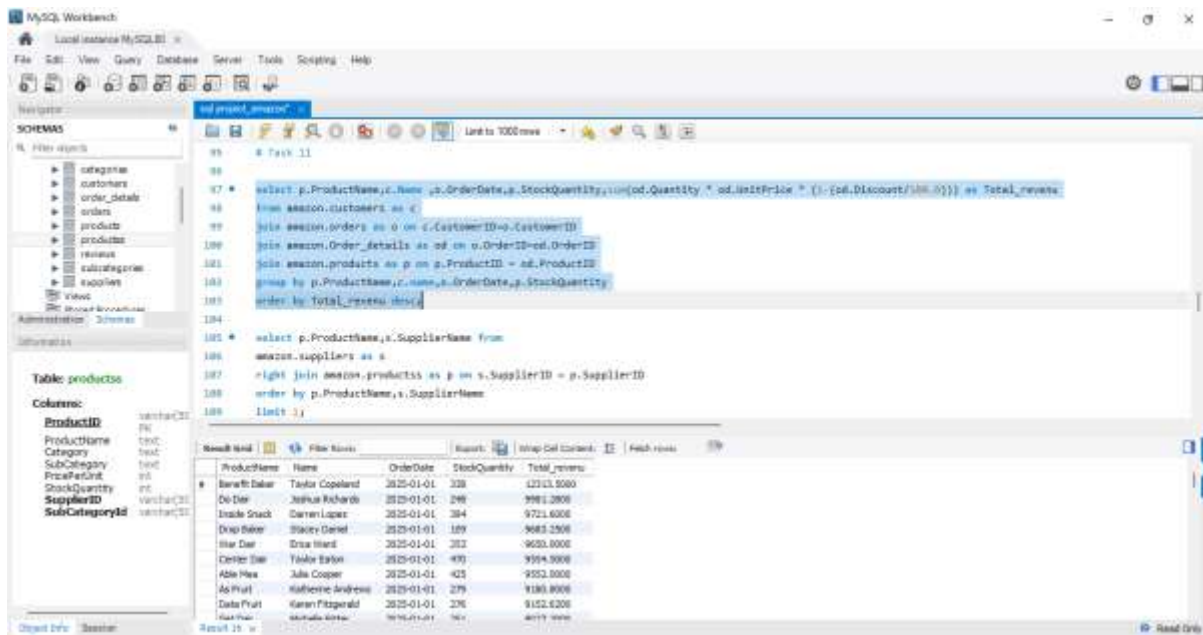
87 Order by
88 TotalSales DESC)
89
90 use amazon;
91
92 select c.Name, c.CustomerID, sum(od.Quantity*od.UnitPrice) as Total_spent,
93 rank() over (order by sum(od.Quantity*od.UnitPrice) desc) as Total_rank
94 from amazon.customers as c
95 join amazon.orders as o on c.CustomerID = o.CustomerID
96 join amazon.order_details as od on o.OrderID = od.OrderID
97 group by c.Name, c.CustomerID
98 having sum(od.Quantity*od.UnitPrice) > 5000
99 order by Total_spent desc)
100

```

Name	CustomerID	Total_spent	Total_rank
Christopher Douglas	33653a75-9a09-409b-a376-c415766a6128	84404	1
Stacy Webb	128123ee-1301-4079-92d9-1193f640299b	83899	2
Sean Allen	226097ed-111f-45c2-9a4d-42295431f112	81215	3
Robert Corbett	666f12b7-5e79-4676-6796-a0793e643d3e	79909	4
Ryan Lee	42c2495c-3846-462c-4733-028a2858a6ac	78544	5
Joel Fowler	4d5b1421-8c13-46d3-99fe-e5a3b11a753e	77712	6
Alexander Moore	c9f4c090-20ac-467e-4225-cab9c342c336	77204	7
Joshua Marshall	79c2f0c4-3d64-4fed-aacc-700b780809a3	72449	8
Julie Cooper	a6c7fa05-6184-429a-9190-4b23a3e54322	72413	9
Robert Williamson	a17f7f0a-8b4c-4a8a-03a4-719b4c79b4c7	71708	10

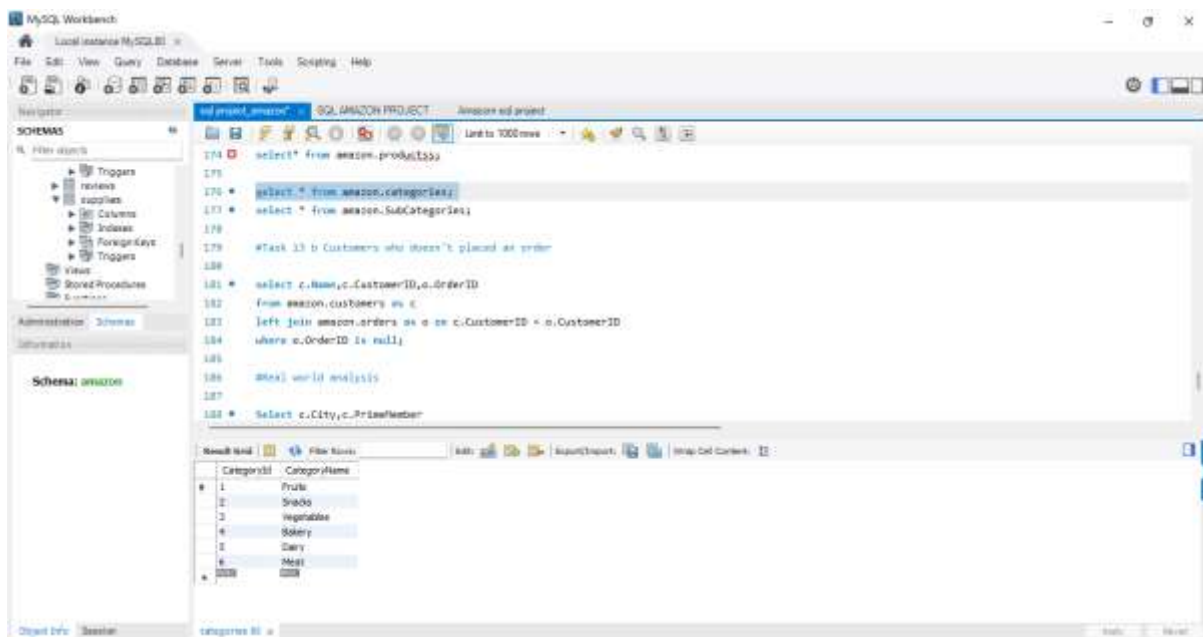
Task 11: Complex Joins and Aggregations

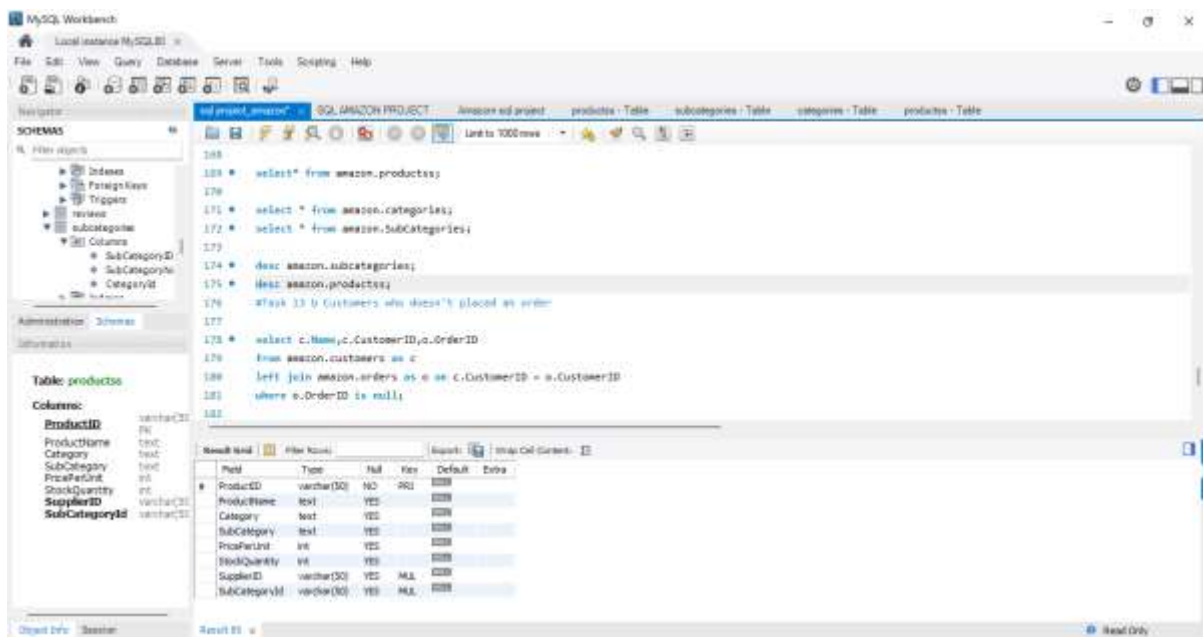
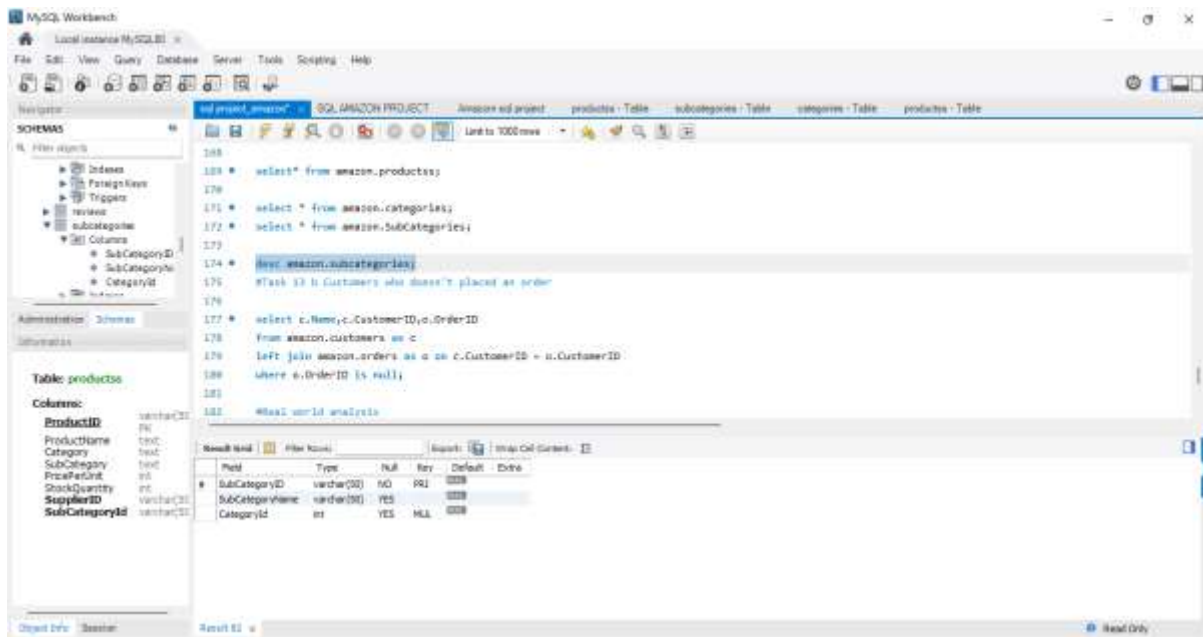
JOIN operations were used between Orders and Order_Details tables to compute total revenue per order. Additional queries identified top customers and suppliers with maximum stock.



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.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'amazon' selected. The 'amazon' database is open, showing tables like 'products', 'order_details', and 'suppliers'. The 'suppliers' table is highlighted, showing columns: SupplierID, SupplierName, ContactPerson, Phone, City, and State.

The main editor contains the following SQL query:

```

137
138 select p.ProductName, sum(od.Quantity * od.UnitPrice * (1 - (od.Discount/100))) as Total_Sales
139 from
140 amazon.order_details as od
141 join amazon.products as p on od.ProductID=p.ProductID
142 group by p.ProductName
143 order by Total_Sales desc
144 limit 10

```

The 'Result Grid' shows the execution results:

Product Name	Total Sales
Fall Snack	87145.000
De Dip	75105.940
Stacy Snack	67893.940

The 'Output' pane shows the execution log with details for each statement, including execution time and error messages.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'amazon' selected. The 'amazon' database is open, showing tables like 'products', 'order_details', and 'suppliers'. The 'suppliers' table is highlighted, showing columns: SupplierID, SupplierName, ContactPerson, Phone, City, and State.

The main editor contains the following SQL query:

```

138
139 select p.ProductName, sum(od.Quantity * od.UnitPrice * (1 - (od.Discount/100))) as Total_Sales
140 from
141 amazon.order_details as od
142 join amazon.products as p on od.ProductID=p.ProductID
143 group by p.ProductName
144 order by Total_Sales desc
145 limit 10
146
147 --Task 13 b Customers who doesn't placed an order
148
149 select c.Name, c.CustomerID, c.OrderID
150 from amazon.customers as c
151 left join amazon.orders as o on c.CustomerID = o.CustomerID
152 where o.OrderID is null

```

The 'Result Grid' shows the execution results:

Name	CustomerID	OrderID
Brian Huber	0346c12-772-4fa-b11-407402b5ab18	0000
William Cummings	03ce7120-4080-4a9e-b020-eac1a179c09e	0000
Daniel Nguyen	0409c2f-e284-4ba0-b23f-7c7704b4d323	0000
Medele Mokughin	1f02b3d3-0209-4692-a271-4a902309b0d1	0000
Oran Taylor	17fa83e5-e720-4b42-b05c-f4c11198c04c	0000
Bertha Parker	2636a589-1a18-c019-95d2-43398f0294e5	0000
Glenn Dominguez	24d88328-65cf-4a4d-ea8d-a92864976094	0000
Jerome Smith	44088647-cf70-467b-4313-4699f72b388e	0000
Emberly McLean	4d24775d-48ae-4ea3-b069-95430a2b24d2	0000
Elizabeth Wilson	54811110a-8a71-42a4b1-d7c0-b2a09101017a7c7d	0000

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

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SQL AMAZON PROJECT Amazon sqldb project

146 #Task 13 In Customers who didn't placed an order

147

148 * select c.Name,c.CustomerID,o.OrderID

149 from amazon.customers as c

150 left join amazon.orders as o on c.CustomerID = o.CustomerID

151 where o.OrderID is null;

152

153 #task world analysis

154

155 * select c.City,c.PrimeMember

156 from amazon.customers as c

157 order by c.PrimeMember desc;

158

159 * select * from amazon.customers;

Result grid

City	PrimeMember
Shannonville	Yes
MojaveFort	Yes
Essexbury	Yes
South Georgetown	Yes
Castleton	Yes
Bellport	Yes
Stakerston	Yes
East Petrolia	Yes
Massachusetts	Yes
Massachusetts	Yes
Massachusetts	Yes

customers 10 x

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SQL AMAZON PROJECT Amazon sqldb project

153 #task world analysis

154

155 * select c.City,c.PrimeMember

156 from amazon.customers as c

157 order by c.PrimeMember desc;

158

159 * select * from amazon.customers;

160

161 # task 3 most frequently ordered category

162

163 * select distinct p.Category

164 from amazon.products as p

165 order by Category

166 limit 3;

Result grid

Category
Electronics
Books
Home & Garden

products 10 x