Knowledge is Power

DIU

Estd. 7th April, 1995

Dhaka International University

CSEDIU 1996

Department of CSE

Course Name : Database Management Systems Lab

Course No. : 0612-304

Submitted to:

MD. AYON MIA
Lecturer
Department of CSE
Dhaka International University

Submitted by:

Shihab Uddin Soumik (35)
Tasnuba Kawsar (40)
MD. Nure Lokman (43)
Abdullah Ibrahim (12)

Project Proposal on:

Contents

Database Design and Implementation of a Scalable E-Commerce System Using SQL

Objectives:

- ➢ Design a normalized, efficient, and scalable relational database for a large-scale ecommerce platform.
- ➢ Ensure data integrity, accuracy, and consistency across all transactions and operations.
- ➢ Enable smooth handling of customer activities, vendor operations, product management, and order processing.
- ➢ Implement secure login systems, product reviews, and payment tracking using SQL queries and constraints.
- ➢ Support future scalability with flexible and modular table relationships. **Context:**

As online shopping grows rapidly, there's a need for powerful, secure, and flexible e-commerce platforms. This project focuses on building a complete website that manages users, supports multiple retailers, and handles product listings with advanced search options. It includes real-time inventory and order tracking, secure payment systems, and strong security features like authentication and encryption. To enhance user experience, it offers reviews, wishlists, personalized recommendations, and admin tools to monitor the platform. Designed for scalability and future growth, it's ready to support new technologies like AI, AR, Blockchain, and progressive web apps, making it a reliable foundation for a modern digital marketplace.

Database Entities and Relationships:

1. Entity: Users & Address

- **Attributes:**
  - user_id **(PK)** – Unique, NOT NULL
  - user_name – NOT NULL
  - email – Unique, NOT NULL
  - address – Unique, NOT NULL

- **2. Entity: Products, Product Images**

- **Attributes:**
  - product_code **(PK)** – Unique, NOT NULL
  - product_name – NOT NULL
  - Product_quantity
  - category_name – NOT NULL
  - price – NOT NULL, positive

3. Entity: Cart & Wishlist (Weak)

- **Attributes:**
  - user_id – NOT NULL
  - product_code – NOT NULL
  - product_name – NOT NULL
  - product_quantity – NOT NULL, >1

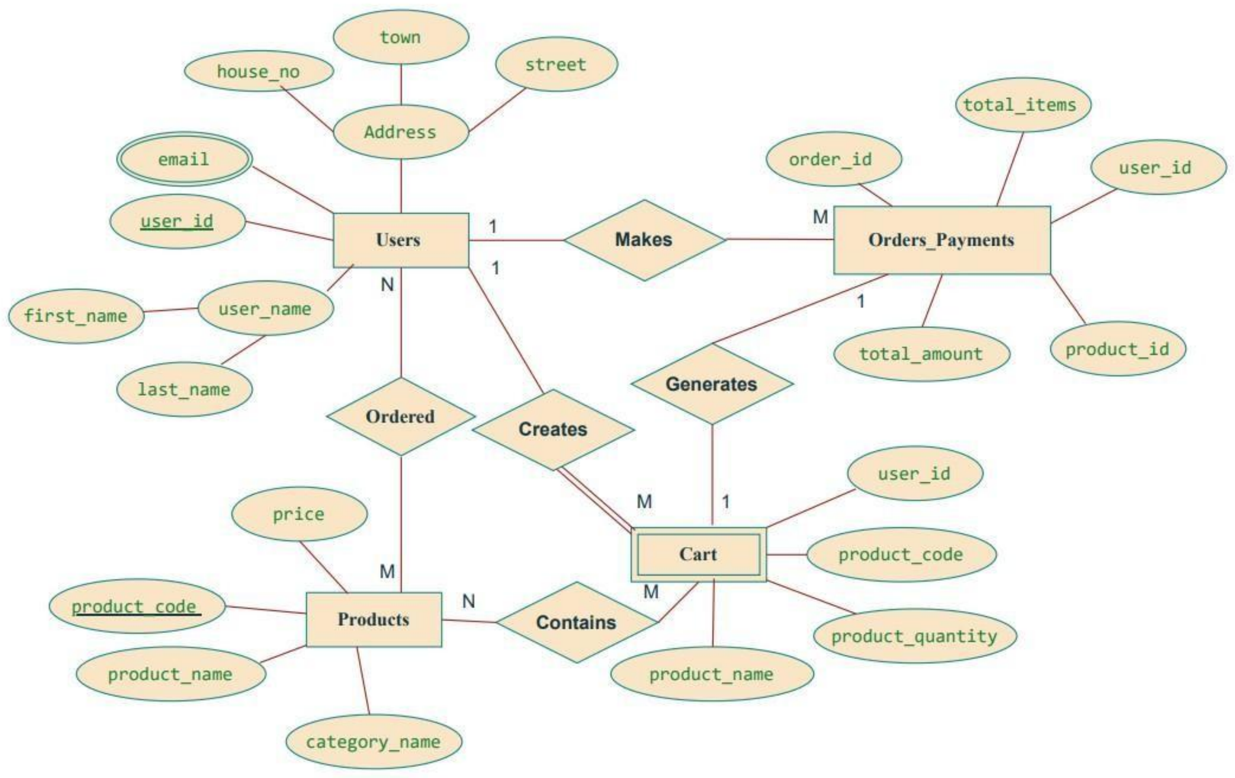
4.Entity: Orders & Payments

- **Attributes:**
  - order_id **(PK)** – Unique
  - user_id – NOT NULL
  - product_id **(FK → Products.product_id)** – NOT NULL
  - total_amount – NOT NULL, positive
  - total_items – NOT NULL, positive

The ER diagram:

Relationship	Entities Involved	Type	Description
Enrolls	Student ↔ Course	Many-to-Many	A student can enroll in many courses, and a course can have many students.
Teaches	Teacher ↔ Course	Many-to-Many	A teacher can teach many courses, and a course can be taught by many.
HasResult	Student → Result	One-to-Many	A student can have many results.
CourseResult	Course → Result	One-to-Many	A course can have many results (for different students).
ResultDetails	Result → Details	One-to-One	Each result can have detailed marks, grade, etc.

The ER diagram:



Mapping

1. Users:

user_id	F_name	L_name	town	street	House_no

2. user_Email:

user_id	u_Email

3. Address:

House_no	town	street

4. Order_Payment:

order_id	user_id	Total_amount	total_item	Product_id

5. Makes:

User_id	order_id

6.Ordered:

user_id	Product_code

7.Cart:

user_id	Product_code

Future Improvements:

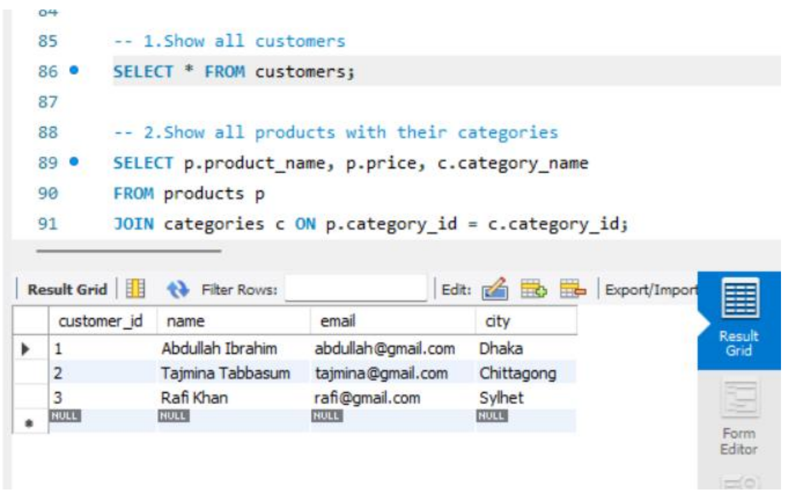
- ➢ Partitioning Large Tables (e.g., orders, payments) for faster query performance.
- ➢ Advanced Stored Procedures for returns, cancellations, and bulk operations.
- ➢ Dynamic Reporting Views for business intelligence dashboards.
- ➢ Database Auditing & Logging for all user actions.
- ➢ Migration Scripts to integrate with external payment APIs or microservices.

Table Create And Data Insert Code:

```
DROP DATABASE IF EXISTS D_87_Ecommerce;
CREATE DATABASE D_87_Ecommerce;
USE D_87_Ecommerce;

CREATE TABLE Users (
 user_id INT AUTO_INCREMENT PRIMARY KEY,
 user_name VARCHAR(50),
 address VARCHAR(150),
email VARCHAR(50)
 );

CREATE TABLE Products (
 product_code INT AUTO_INCREMENT PRIMARY KEY,
 product_name VARCHAR(100),
 category_name VARCHAR(100),
 price INT
 );

CREATE TABLE Cart (
 cart_id INT AUTO_INCREMENT PRIMARY KEY,
 product_quantity INT,
 user_id INT,
 product_code INT,
 FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE,
 FOREIGN KEY (product_code) REFERENCES Products(product_code) ON DELETE CASCADE
 );

CREATE TABLE Orders_Payments (
order_id INT AUTO_INCREMENT PRIMARY KEY,
user_id INT,
product_code INT,
total_items INT,
total_amount INT,
FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE,
FOREIGN KEY (product_code) REFERENCES Products(product_code) ON
DELETE CASCADE
);

INSERT INTO Users (user_name, address, email) VALUES
('Ali Ahmed', 'Dhaka', 'ali.ahmed@gmail.com'),
('Bilal Hossain', 'Chittagong', 'bilal_hossain@yahoo.com'),
('Chitra Das', 'Sylhet', 'chitra.das@outlook.com'),
 ('Dipto Rahman', 'Rajshahi', 'dipto.rahman@hotmail.com'),
('Ema Khan', 'Khulna', 'ema.khan@gmail.com'),
('Farhan Sayeed', 'Barisal', 'farhan.sayeed@icloud.com'),
('Gulshan Ara', 'Rangpur', 'gulshan.ara@gmail.com'),
('Habib Ullah', 'Comilla', 'habibullah.bd@yahoo.com'),
('Ishrat Jahan', 'Mymensingh', 'ishrat.jahan@protonmail.com'),
('Jamal Uddin', 'Narayanganj', 'jamal.uddin@gmail.com');

INSERT INTO Products (product_name, category_name, price) VALUES
('Walton LED TV 32 Inch', 'Television', 18500),
('Singer Refrigerator 240L', 'Refrigerator', 34500),
('Vision Rice Cooker 1.8L', 'Kitchen Appliance', 2450),
('Philips Electric Iron', 'Home Appliance', 1890),
('Sharp Microwave Oven 25L', 'Kitchen Appliance', 14500),
('Sony Bluetooth Speaker', 'Audio Device', 5600),
('Hitachi Air Conditioner 1.5 Ton', 'Cooling Appliance', 62500),
('Panasonic Ceiling Fan', 'Fan', 2950),
('Nova Hair Dryer', 'Personal Care', 1250),
('Kiam Blender 3-in-1', 'Kitchen Appliance', 3550);

INSERT INTO Cart (product_quantity, user_id, product_code) VALUES
(1, 1, 1),
(2, 2, 3),
(1, 3, 4),
(1, 4, 2),
(3, 5, 10),
(1, 6, 7),
(2, 7, 5),
(1, 8, 9),
(1, 9, 6),
(2, 10, 8);

INSERT INTO Orders_Payments (user_id, product_code, total_items, total_amount) VALUES
(1, 1, 1, 18500),
(2, 3, 2, 4900),
(3, 4, 1, 1890),
(4, 2, 1, 34500),
(5, 10, 3, 10650),
(6, 7, 1, 62500),
(7, 5, 2, 29000),
(8, 9, 1, 1250),
(9, 6, 1, 5600),
(10, 8, 2, 5900);
```

Run Query

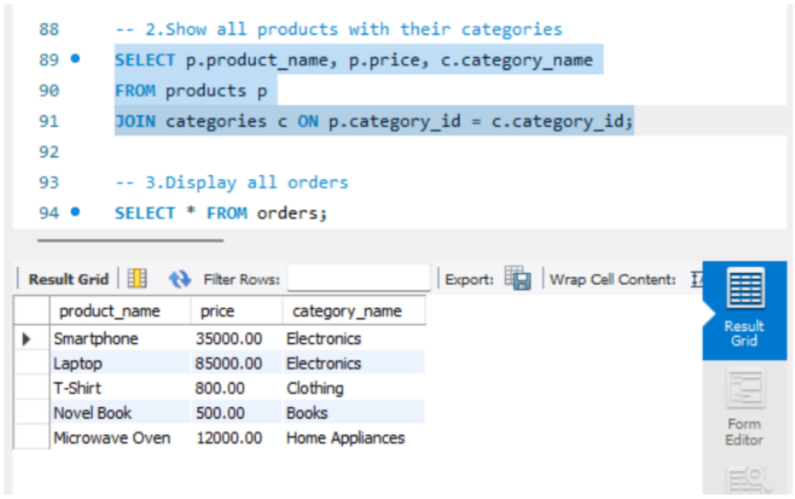
1.Show all customers



```
-- 1.Show all customers
SELECT * FROM customers;

-- 2.Show all products with their categories
SELECT p.product_name, p.price, c.category_name
FROM products p
JOIN categories c ON p.category_id = c.category_id;
```

customer_id	name	email	city
1	Abdullah Ibrahim	abdullah@gmail.com	Dhaka
2	Tajmina Tabbasum	tajmina@gmail.com	Chittagong
3	Rafi Khan	rafi@gmail.com	Sylhet
NULL	NULL	NULL	NULL

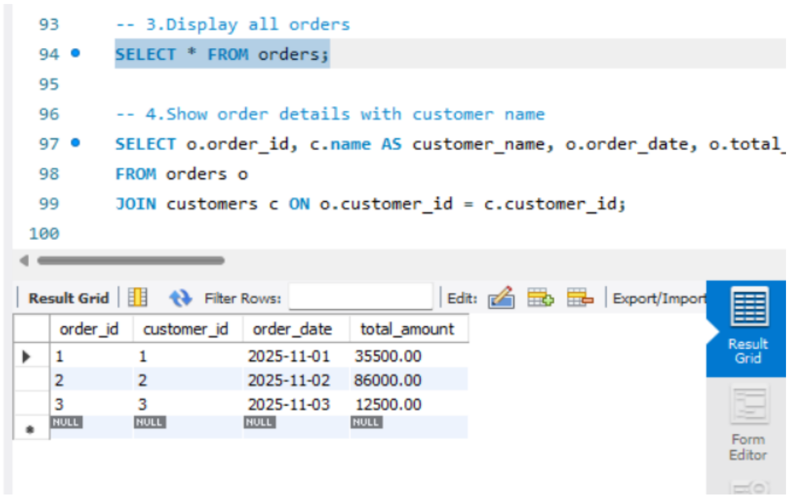
2.Show all products with their categories



```
-- 2.Show all products with their categories
SELECT p.product_name, p.price, c.category_name
FROM products p
JOIN categories c ON p.category_id = c.category_id;

-- 3.Display all orders
SELECT * FROM orders;
```

product_name	price	category_name
Smartphone	35000.00	Electronics
Laptop	85000.00	Electronics
T-Shirt	800.00	Clothing
Novel Book	500.00	Books
Microwave Oven	12000.00	Home Appliances

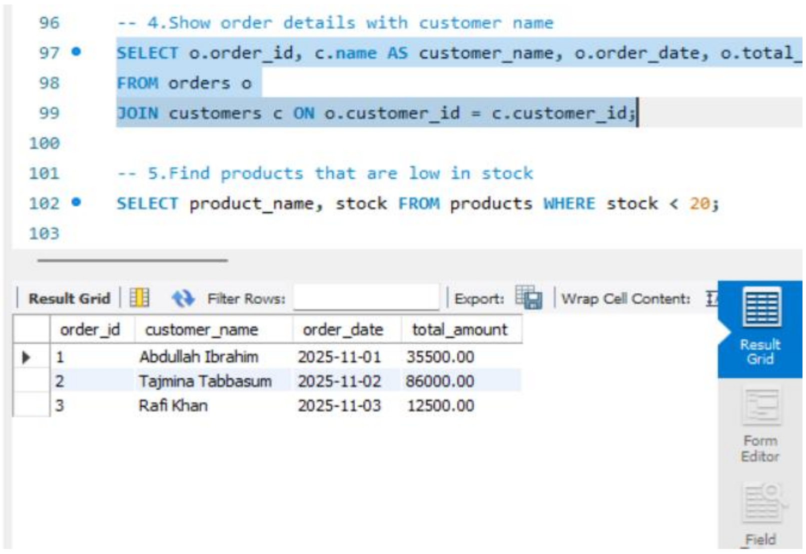
3.Display all orders



```
-- 3.Display all orders
SELECT * FROM orders;

-- 4.Show order details with customer name
SELECT o.order_id, c.name AS customer_name, o.order_date, o.total_
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id;
```

order_id	customer_id	order_date	total_amount
1	1	2025-11-01	35500.00
2	2	2025-11-02	86000.00
3	3	2025-11-03	12500.00
NULL	NULL	NULL	NULL

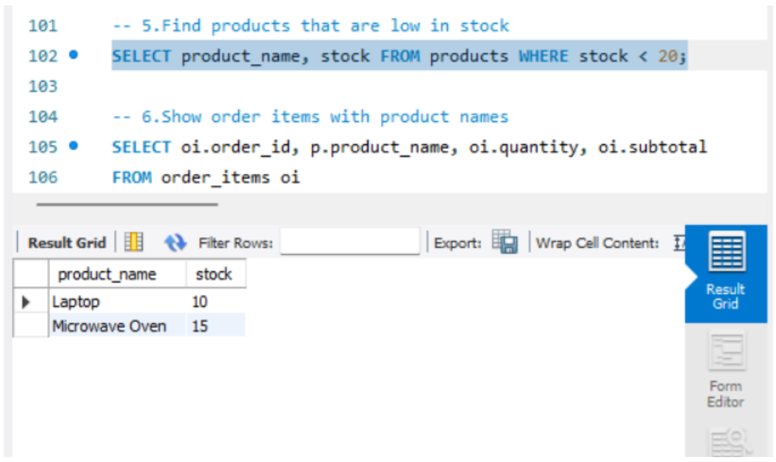
4.**Show order details with customer name**



```
-- 4.Show order details with customer name
SELECT o.order_id, c.name AS customer_name, o.order_date, o.total_
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id;

-- 5.Find products that are low in stock
SELECT product_name, stock FROM products WHERE stock < 20;

```

order_id	customer_name	order_date	total_amount
1	Abdullah Ibrahim	2025-11-01	35500.00
2	Tajmina Tabbasum	2025-11-02	86000.00
3	Rafi Khan	2025-11-03	12500.00

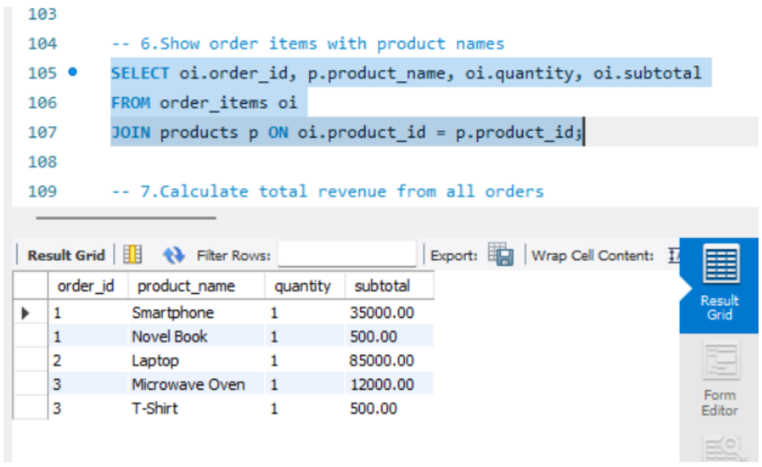
5.**Find products that are low in stock**



```
-- 5.Find products that are low in stock
SELECT product_name, stock FROM products WHERE stock < 20;

-- 6.Show order items with product names
SELECT oi.order_id, p.product_name, oi.quantity, oi.subtotal
FROM order_items oi
```

product_name	stock
Laptop	10
Microwave Oven	15

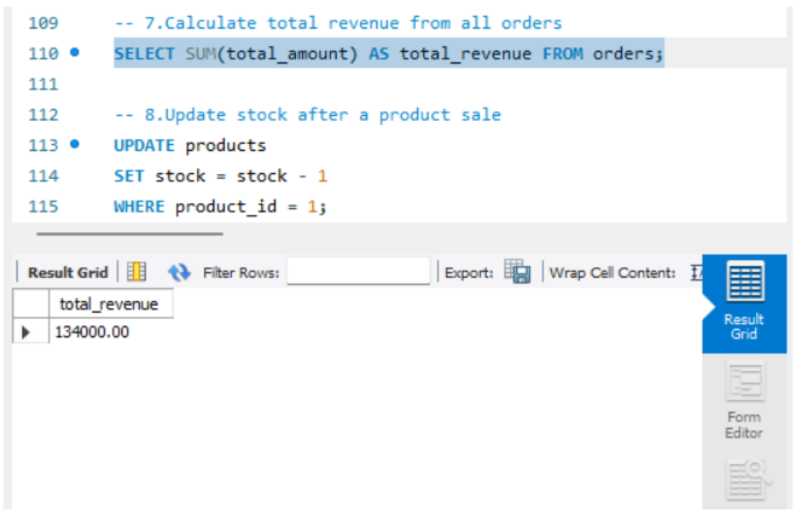
6.**Show order items with product names**



```

-- 6.Show order items with product names
SELECT oi.order_id, p.product_name, oi.quantity, oi.subtotal
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id;

-- 7.Calculate total revenue from all orders
```

order_id	product_name	quantity	subtotal
1	Smartphone	1	35000.00
1	Novel Book	1	500.00
2	Laptop	1	85000.00
3	Microwave Oven	1	12000.00
3	T-Shirt	1	500.00

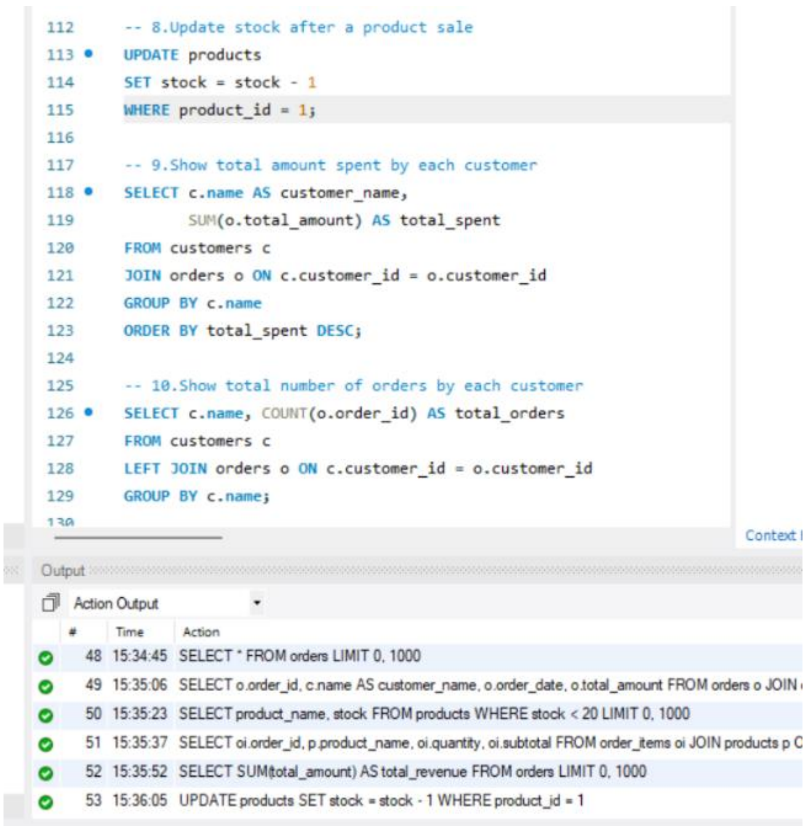
7.Calculate total revenue from all orders



```sql
-- 7.Calculate total revenue from all orders
SELECT SUM(total_amount) AS total_revenue FROM orders;

-- 8.Update stock after a product sale
UPDATE products
SET stock = stock - 1
WHERE product_id = 1;
```

```
total_revenue
134000.00
```

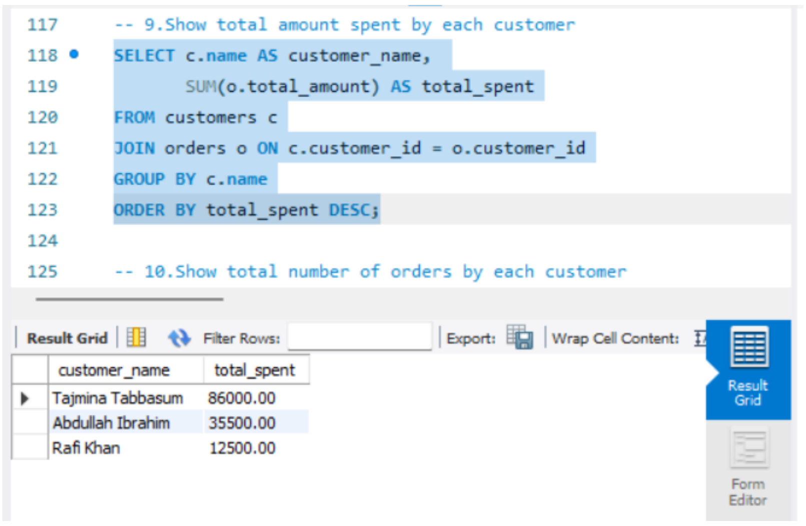
8.Update stock after a product sale



```sql
-- 8.Update stock after a product sale
UPDATE products
SET stock = stock - 1
WHERE product_id = 1;

-- 9.Show total amount spent by each customer
SELECT c.name AS customer_name,
       SUM(o.total_amount) AS total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.name
ORDER BY total_spent DESC;

-- 10.Show total number of orders by each customer
SELECT c.name, COUNT(o.order_id) AS total_orders
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.name;
```

```
   #   Time      Action
✓  48  15:34:45  SELECT * FROM orders LIMIT 0, 1000
✓  49  15:35:06  SELECT o.order_id, c.name AS customer_name, o.order_date, o.total_amount FROM orders o JOIN
✓  50  15:35:23  SELECT product_name, stock FROM products WHERE stock < 20 LIMIT 0, 1000
✓  51  15:35:37  SELECT oi.order_id, p.product_name, oi.quantity, oi.subtotal FROM order_items oi JOIN products p O
✓  52  15:35:52  SELECT SUM(total_amount) AS total_revenue FROM orders LIMIT 0, 1000
✓  53  15:36:05  UPDATE products SET stock = stock - 1 WHERE product_id = 1
```

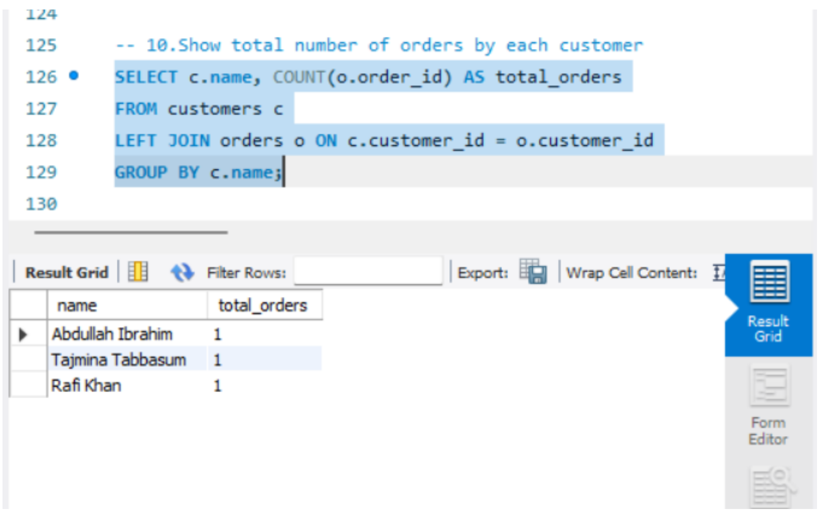
9.Show total amount spent by each customer



```sql
-- 9.Show total amount spent by each customer
SELECT c.name AS customer_name,
       SUM(o.total_amount) AS total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.name
ORDER BY total_spent DESC;

-- 10.Show total number of orders by each customer
```

customer_name	total_spent
Tajmina Tabbasum	86000.00
Abdullah Ibrahim	35500.00
Rafi Khan	12500.00

10.Show total number of orders by each customer



```sql
-- 10.Show total number of orders by each customer
SELECT c.name, COUNT(o.order_id) AS total_orders
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.name;
```

name	total_orders
Abdullah Ibrahim	1
Tajmina Tabbasum	1
Rafi Khan	1

Conclusion:

This SQL-focused on database project demonstrates the core backend structure required to power a fully functional e-commerce application. With a strong relational schema, integrity constraints, and automation over triggers and stored procedures, the system ensures consistent, scalable, and secure data management. It lays the groundwork for seamless front-end integration and can be extended to support advanced business logic and real-world deployments.