# SQL Task

## Task:

Create a database and tables to manage a simple e-commerce system.
The system should have three tables: customers, orders, and products.

## Requirements:

- Create a database named ecommerce.
- Create three tables: customers, orders, and products.
- Insert some sample data into the tables.

```sql
-- Create the database
CREATE DATABASE ecommercetask2;

-- Use the database
USE ecommercetask2;

-- Create customers table
CREATE TABLE customers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100),
    address VARCHAR(255)
);

-- Create products table
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    price DECIMAL(10, 2),
    description TEXT
);
```

```sql
-- Create orders table
CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    total_amount DECIMAL(10, 2),
    FOREIGN KEY (customer_id) REFERENCES customers(id)
);
```

Output

Action Output

| # | Time | Action |
|---|---|---|
| 1 | 19:18:25 | CREATE DATABASE ecommercetask2 |
| 2 | 19:18:25 | USE ecommercetask2 |
| 3 | 19:18:25 | CREATE TABLE customers ( id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), email V... |
| 4 | 19:18:25 | CREATE TABLE products ( id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), price DE... |
| 5 | 19:18:25 | CREATE TABLE orders ( id INT AUTO_INCREMENT PRIMARY KEY, customer_id INT, order_date DAT... |

```sql
-- Insert customers
INSERT INTO customers (name, email, address) VALUES
('Alice', 'alice@example.com', '123 Apple St'),
('Bob', 'bob@example.com', '456 Banana Ave'),
('Charlie', 'charlie@example.com', '789 Cherry Blvd');

-- Insert products
INSERT INTO products (name, price, description) VALUES
('Product A', 50.00, 'Description A'),
('Product B', 100.00, 'Description B'),
('Product C', 40.00, 'Description C'),
('Product D', 200.00, 'Description D'),
('Product E', 150.00, 'Description E');

-- Insert orders
INSERT INTO orders (customer_id, order_date, total_amount) VALUES
(1, CURDATE(), 150.00),
(2, CURDATE() - INTERVAL 10 DAY, 250.00),
(1, CURDATE() - INTERVAL 40 DAY, 75.00),
(3, CURDATE() - INTERVAL 5 DAY, 180.00);
```

Output

Action Output

| # | Time | Action |
|---|---|---|
| 1 | 19:21:24 | INSERT INTO customers (name, email, address) VALUES ('Alice', 'alice@example.com', '123 Apple St'), ('Bob', 'b... |
| 2 | 19:21:24 | INSERT INTO products (name, price, description) VALUES ('Product A', 50.00, 'Description A'), ('Product B', 100.... |
| 3 | 19:21:24 | INSERT INTO orders (customer_id, order_date, total_amount) VALUES (1, CURDATE(), 150.00), (2, CURDATE() ... |

## Table Structure:

### Customers

Id (primary key, auto-increment): unique identifier for each customer
Name: customer's name
Email: customer's email address
Address: customer's address

### Orders

Id (primary key, auto-increment): unique identifier for each order
customer_id (foreign key referencing customers.id): a customer who placed the order
order_date: date the order was placed
total_amount: total amount of the order

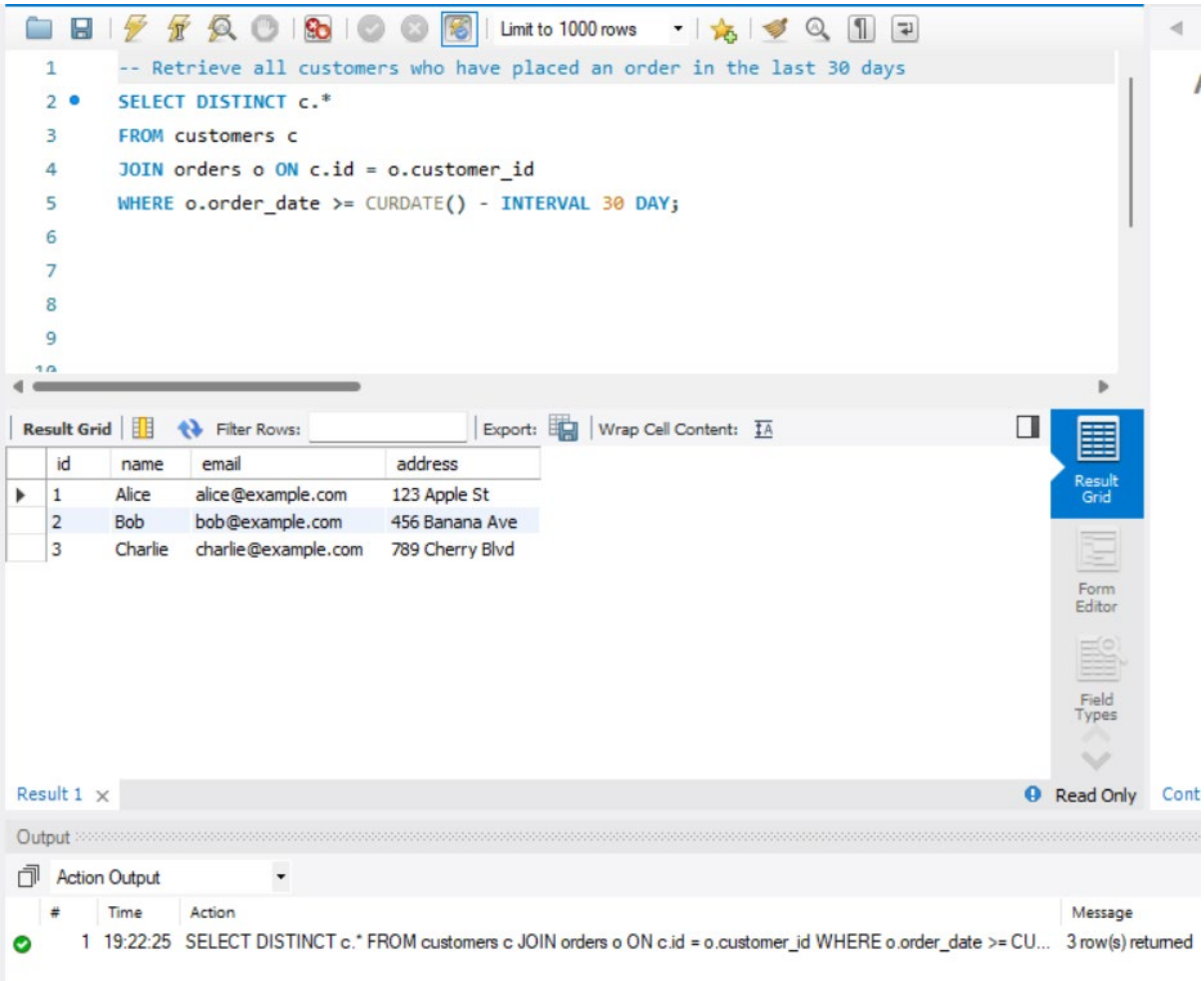
### Products

Id (primary key, auto-increment): unique identifier for each product
Name: product's name

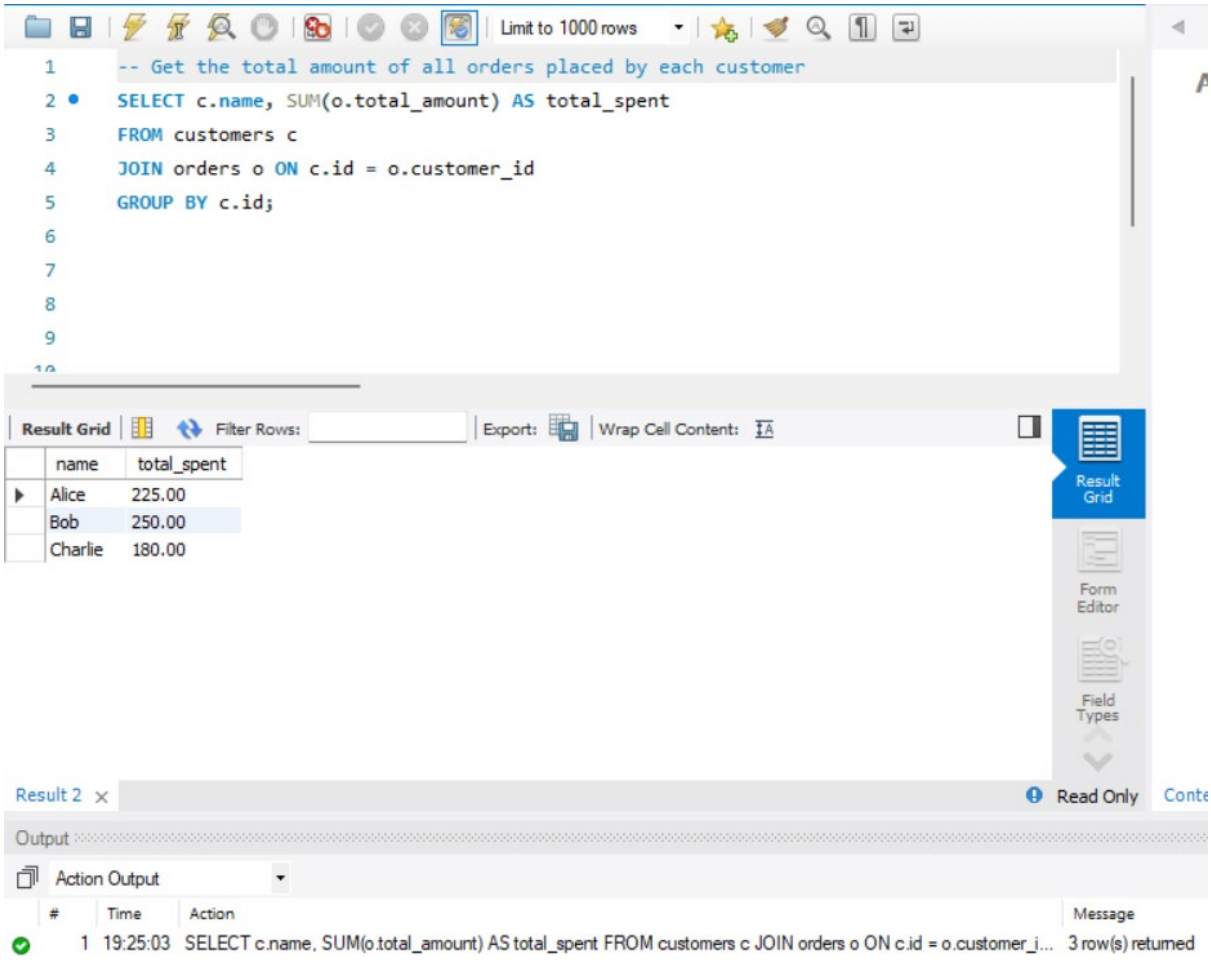
Price: product's price
Description: product's description

## Queries to Write:

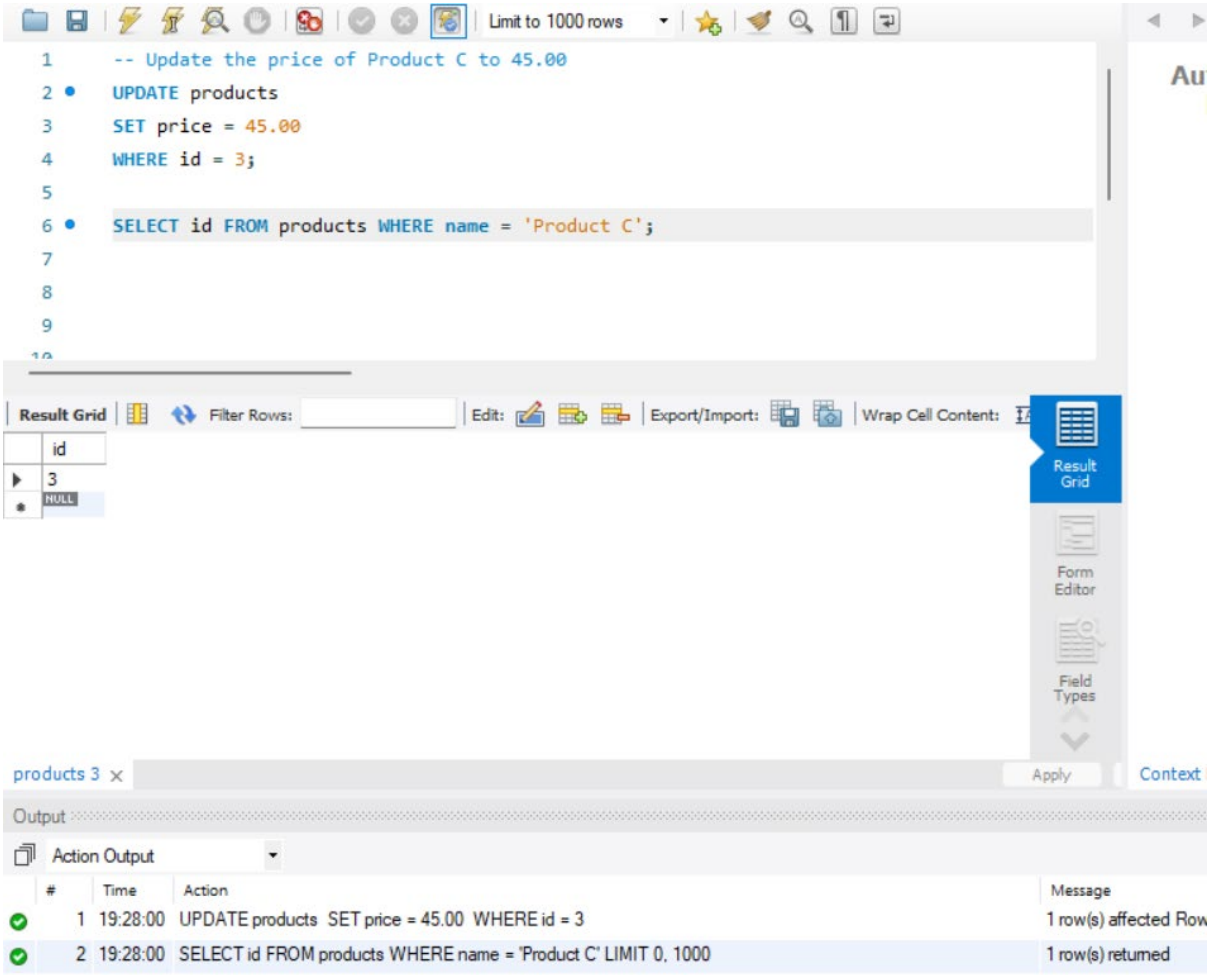
- Retrieve all customers who have placed an order in the last 30 days.



- Get the total amount of all orders placed by each customer.



- Update the price of Product C to 45.00.



- Add a new column discount to the products table.

```
-- Add a new column discount to the products table
ALTER TABLE products
ADD COLUMN discount DECIMAL(5, 2) DEFAULT 0.00;
```

Output

Action Output

| # | Time | Action |
|---|---|---|
| 1 | 19:29:29 | ALTER TABLE products ADD COLUMN discount DECIMAL(5, 2) DEFAULT 0.00 |

- Retrieve the top 3 products with the highest price.

```
-- Retrieve the top 3 products with the highest price
SELECT * FROM products
ORDER BY price DESC
LIMIT 3;
```

Result Grid

| id | name | price | description | discount |
|---|---|---|---|---|
| 4 | Product D | 200.00 | Description D | 0.00 |
| 5 | Product E | 150.00 | Description E | 0.00 |
| 2 | Product B | 100.00 | Description B | 0.00 |
| NULL | NULL | NULL | NULL | NULL |

products 4

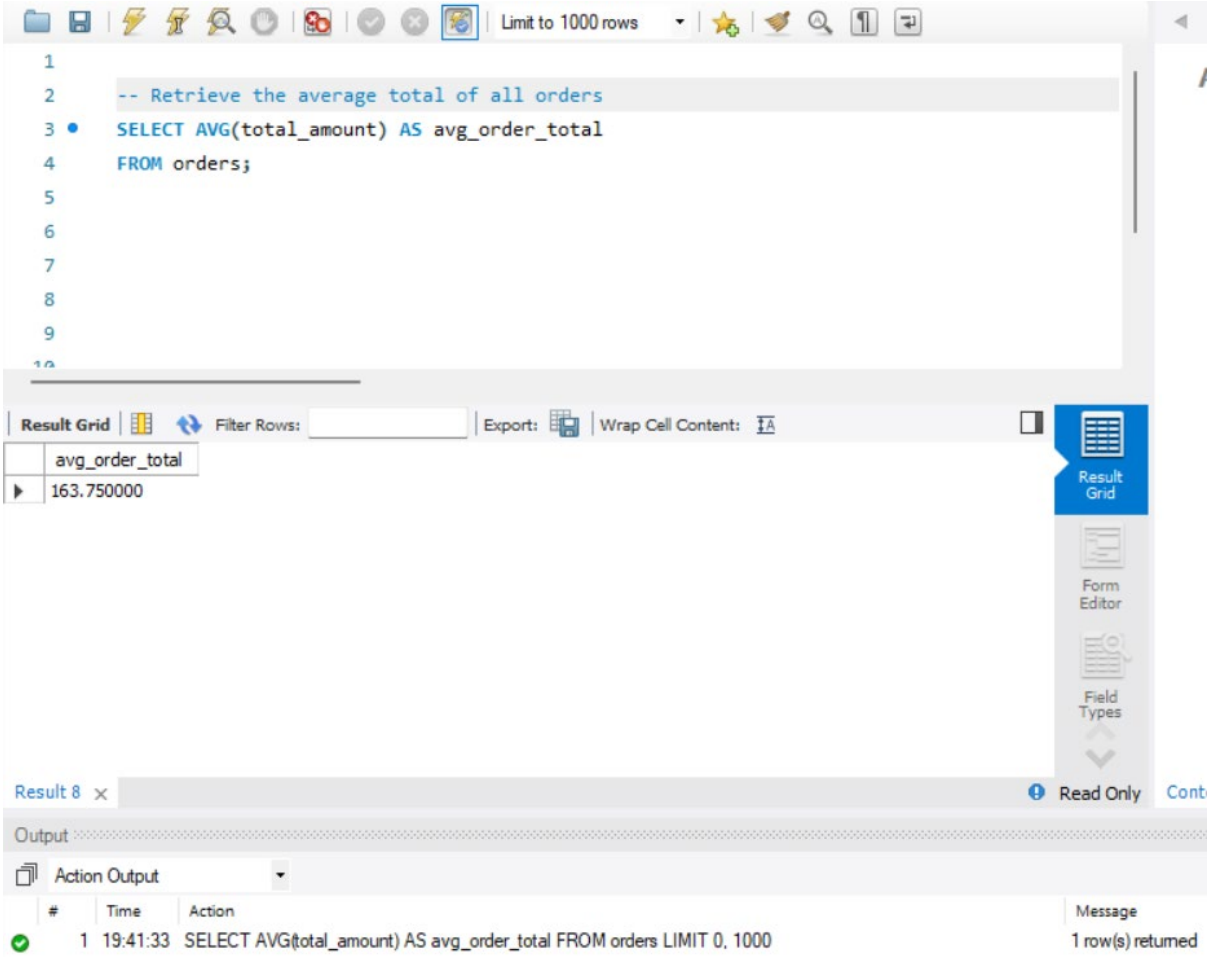
Output

Action Output

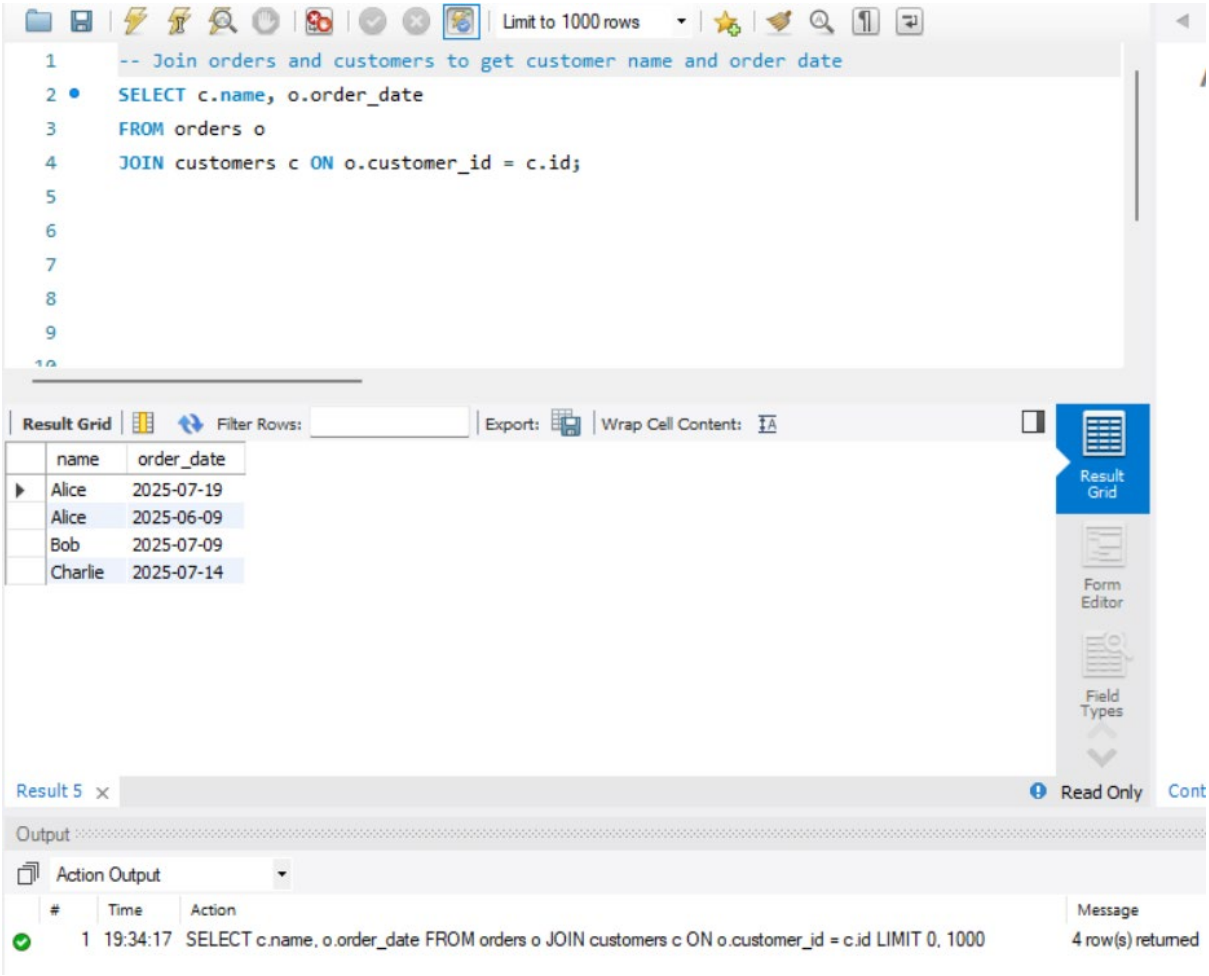
| # | Time | Action | Message |
|---|---|---|---|
| 1 | 19:31:31 | SELECT * FROM products ORDER BY price DESC LIMIT 3 | 3 row(s) returned |

- Get the names of customers who have ordered Product A.

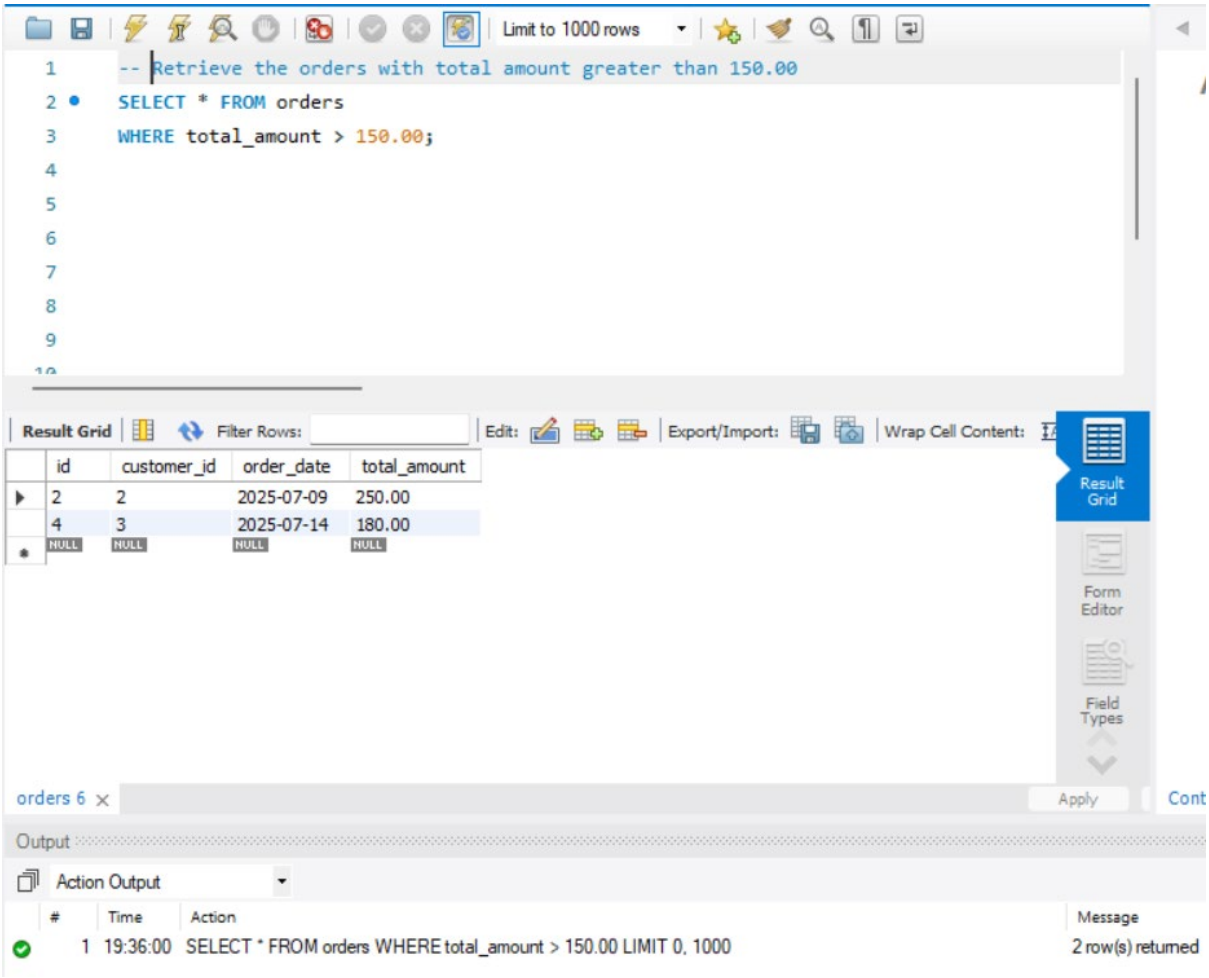
```
-- Get the names of customers who have ordered Product A
-- Requires normalization with order items first (see below)
-- So this query comes after normalization
```



- Join the orders and customers tables to retrieve the customer's name and order date for each order.



- Retrieve the orders with a total amount greater than 150.00.



- Normalize the database by creating a separate table for order items and updating the orders table to reference the order items table.

```sql
-- Normalize: Create order_items and update structure
CREATE TABLE order_items (
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);

-- Example insert into order_items
INSERT INTO order_items (order_id, product_id, quantity) VALUES
(1, 1, 2),  -- 2 of Product A in Order 1
(2, 2, 1),  -- 1 of Product B in Order 2
(2, 1, 1),  -- 1 of Product A in Order 2
(3, 3, 1),  -- 1 of Product C in Order 3
(4, 4, 1);  -- 1 of Product D in Order 4
```

Output

Action Output

| # | Time | Action | Message |
|---|---|---|---|
| 1 | 19:38:50 | CREATE TABLE order_items ( id INT AUTO_INCREMENT PRIMARY KEY, order_id INT, product_id INT,... | 0 row(s) affecte |
| 2 | 19:38:50 | INSERT INTO order_items (order_id, product_id, quantity) VALUES (1, 1, 2), -- 2 of Product A in Order 1 (2, 2, 1), ... | 5 row(s) affecte |

- Retrieve the average total of all orders.

```
-- Get names of customers who have ordered Product A (after normalization)
SELECT DISTINCT c.name
FROM customers c
JOIN orders o ON c.id = o.customer_id
JOIN order_items oi ON o.id = oi.order_id
JOIN products p ON oi.product_id = p.id
WHERE p.name = 'Product A';
```

Result Grid

| name |
|---|
| Alice |
| Bob |

Result 7

Output

Action Output

| # | Time | Action | Message |
|---|---|---|---|
| 1 | 19:40:09 | SELECT DISTINCT c.name FROM customers c JOIN orders o ON c.id = o.customer_id JOIN order_items oi ON ... | 2 row(s) |