🛒 SQL Task1: E-Commerce System

📌 Task Overview

Design and implement a simple e-commerce system using MySQL.

📋 Table Structures

Create customers table

```
CREATE TABLE customers (
    id INT AUTO_INCREMENT PRIMARY KEY, -- Unique customer ID
    name VARCHAR(100) NOT NULL, -- Customer's name
    email VARCHAR(255) NOT NULL UNIQUE, -- Customer's email
    address VARCHAR(255) -- Customer's address
);
```

Create products table

```
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY, -- Unique product ID
    name VARCHAR(100) NOT NULL, -- Product's name
    price DECIMAL(10,2) NOT NULL, -- Product's price
    description TEXT -- Product's description
);
```

Create orders table

```
CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY, -- Unique order ID
    customer_id INT NOT NULL, -- Customer who placed the order
```

```
    order_date DATE NOT NULL, -- Date of the order
    total_amount DECIMAL(10,2) NOT NULL, -- Total amount of the order
    FOREIGN KEY (customer_id) REFERENCES customers(id)
);
```

📦 Sample Data

Insert sample customers

```
INSERT INTO customers (name, email, address) VALUES
('Alice Smith', 'alice@example.com', '123 Park Ave'),
('Bob Jones', 'bob@example.com', '456 Maple St'),
('Carol Lee', 'carol@example.com', '789 Oak Rd');
```

Insert sample products

```
INSERT INTO products (name, price, description) VALUES
('Product A', 25.00, 'Basic product A'),
('Product B', 35.00, 'Popular product B'),
('Product C', 40.00, 'Premium product C');
```

Insert sample orders

```
INSERT INTO orders (customer_id, order_date, total_amount) VALUES
(1, CURDATE(), 60.00),
(2, DATE_SUB(CURDATE(), INTERVAL 10 DAY), 25.00),
(1, DATE_SUB(CURDATE(), INTERVAL 40 DAY), 40.00),
(3, CURDATE(), 160.00);
```

🔍 Queries and Results

📅 1. Customers who ordered in the last 30 days

```
SELECT DISTINCT c.*
FROM customers c
JOIN orders o ON c.id = o.customer_id
WHERE o.order_date >= CURDATE() - INTERVAL 30 DAY;
```

```
mysql> SELECT DISTINCT c.*
    -> FROM customers c
    -> JOIN orders o ON c.id = o.customer_id
    -> WHERE o.order_date >= CURDATE() - INTERVAL 30 DAY;
+----+-------------+-------------------+--------------+
| id | name        | email             | address      |
+----+-------------+-------------------+--------------+
|  1 | Alice Smith | alice@example.com | 123 Park Ave |
|  2 | Bob Jones   | bob@example.com   | 456 Maple St |
|  3 | Carol Lee   | carol@example.com | 789 Oak Rd   |
+----+-------------+-------------------+--------------+
3 rows in set (0.21 sec)
```

💰 2. Total amount spent by each customer

SELECT c.name, SUM(o.total_amount) AS total_spent
FROM customers c
JOIN orders o ON c.id = o.customer_id
GROUP BY c.id, c.name;

```
mysql> SELECT c.name, SUM(o.total_amount) AS total_spent
    -> FROM customers c
    -> JOIN orders o ON c.id = o.customer_id
    -> GROUP BY c.id, c.name;
+-------------+-------------+
| name        | total_spent |
+-------------+-------------+
| Alice Smith |      100.00 |
| Bob Jones   |       25.00 |
| Carol Lee   |      160.00 |
+-------------+-------------+
3 rows in set (0.05 sec)
```

🧾 3. Update price of Product C to 45.00

UPDATE products

SET price = 45.00

WHERE name = 'Product C';

```
mysql> UPDATE products
    -> SET price = 45.00
    -> WHERE name = 'Product C';
Query OK, 1 row affected (0.11 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> SELECT * FROM products WHERE name = 'product C';
+----+-----------+-------+-------------------+----------+
| id | name      | price | description       | discount |
+----+-----------+-------+-------------------+----------+
|  3 | Product C | 45.00 | Premium product C |     0.00 |
+----+-----------+-------+-------------------+----------+
1 row in set (0.00 sec)
```

➕ 4. Add discount column to products table

ALTER TABLE products
ADD COLUMN discount DECIMAL(5,2) DEFAULT 0.00;

```
mysql> DESC products;
+-------------+---------------+------+-----+---------+----------------+
| Field       | Type          | Null | Key | Default | Extra          |
+-------------+---------------+------+-----+---------+----------------+
| id          | int           | NO   | PRI | NULL    | auto_increment |
| name        | varchar(100)  | NO   |     | NULL    |                |
| price       | decimal(10,2) | NO   |     | NULL    |                |
| description | text          | YES  |     | NULL    |                |
| discount    | decimal(5,2)  | YES  |     | 0.00    |                |
+-------------+---------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

🏆 5. Top 3 most expensive products

SELECT *
FROM products
ORDER BY price DESC
LIMIT 3;

```
mysql> SELECT *
    -> FROM products
    -> ORDER BY price DESC
    -> LIMIT 3;
+----+-----------+-------+-------------------+----------+
| id | name      | price | description       | discount |
+----+-----------+-------+-------------------+----------+
|  3 | Product C | 45.00 | Premium product C |     0.00 |
|  2 | Product B | 35.00 | Popular product B |     0.00 |
|  1 | Product A | 25.00 | Basic product A   |     0.00 |
+----+-----------+-------+-------------------+----------+
3 rows in set (0.10 sec)
```

🛍️ 6. Customers who ordered Product A

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';

```
mysql> 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';
+-------------+
| name        |
+-------------+
| Alice Smith |
| Bob Jones   |
+-------------+
2 rows in set (0.01 sec)

mysql>
```

🔗 7. Orders with customer names

SELECT c.name, o.order_date

FROM orders o

JOIN customers c ON o.customer_id = c.id;

```
mysql> SELECT c.name, o.order_date
    -> FROM orders o
    -> JOIN customers c ON o.customer_id = c.id;
+-------------+------------+
| name        | order_date |
+-------------+------------+
| Alice Smith | 2025-04-25 |
| Alice Smith | 2025-03-16 |
| Bob Jones   | 2025-04-15 |
| Carol Lee   | 2025-04-25 |
+-------------+------------+
4 rows in set (0.00 sec)
```

💳 8. Orders over 150.00

SELECT *

FROM orders

WHERE total_amount > 150.00;

```
mysql> -- 8. Retrieve orders with a total amount greater than 150.00
mysql> SELECT *
    -> FROM orders
    -> WHERE total_amount > 150.00;
+----+-------------+------------+--------------+
| id | customer_id | order_date | total_amount |
+----+-------------+------------+--------------+
|  4 |           3 | 2025-04-25 |       160.00 |
+----+-------------+------------+--------------+
1 row in set (0.00 sec)
```

🧩 9. Normalize: Create order_items table and update relationships

CREATE TABLE order_items (

id INT AUTO_INCREMENT PRIMARY KEY,

order_id INT NOT NULL,

```
    product_id INT NOT NULL,
    quantity INT NOT NULL DEFAULT 1,
    price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);
INSERT INTO order_items (order_id, product_id, quantity, price) VALUES
(1, 1, 2, 25.00), -- Order 1, Product A, 2 units
(1, 2, 1, 35.00), -- Order 1, Product B, 1 unit
(2, 1, 1, 25.00), -- Order 2, Product A, 1 unit
(3, 3, 1, 40.00), -- Order 3, Product C, 1 unit
(4, 3, 4, 40.00); -- Order 4, Product C, 4 units
```

```
SELECT order_id, SUM(price * quantity) AS calculated_total
FROM order_items
GROUP BY order_id;
```

```
mysql> SELECT order_id, SUM(price * quantity) AS calculated_total
    -> FROM order_items
    -> GROUP BY order_id;
+----------+------------------+
| order_id | calculated_total |
+----------+------------------+
|        1 |            85.00 |
|        2 |            25.00 |
|        3 |            40.00 |
|        4 |           160.00 |
+----------+------------------+
4 rows in set (0.08 sec)
```

📉 10. Average order total

SELECT AVG(total_amount) AS average_order_total

FROM orders;

```
mysql> -- 10. Retrieve the average total of all orders
mysql> SELECT AVG(total_amount) AS average_order_total
    -> FROM orders;
+---------------------+
| average_order_total |
+---------------------+
|           71.250000 |
+---------------------+
1 row in set (0.00 sec)

mysql>
```