```
-- Create the database


CREATE DATABASE ecommerce;
USE ecommerce;

-- Create the customers table


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

-- Create the products table


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

-- Create the 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)
);

-- Insert sample data into the customers table



INSERT INTO customers (name, email, address)
VALUES
    ('Alice Johnson', 'alice@example.com', '123 Maple Street'),
    ('Bob Smith', 'bob@example.com', '456 Oak Avenue'),
    ('Charlie Lee', 'charlie@example.com', '789 Pine Road');

    SELECT * FROM CUSTOMERS;

-- Insert sample data into the products table


INSERT INTO products (name, price, description)
VALUES
```

```
    ('Product A', 20.00, 'Description for Product A'),
    ('Product B', 35.00, 'Description for Product B'),
    ('Product C', 50.00, 'Description for Product C');

    SELECT * FROM PRODUCTS;

-- Insert sample data into the orders table


INSERT INTO orders (customer_id, order_date, total_amount)
VALUES
    (1, CURDATE(), 55.00),
    (2, CURDATE() - INTERVAL 10 DAY, 85.00),
    (1, CURDATE() - INTERVAL 25 DAY, 75.00),
    (3, CURDATE() - INTERVAL 35 DAY, 120.00);

    SELECT *FROM ORDERS ;

-- Queries

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


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

-- 2. Get the total amount of all orders placed 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;

-- 3. Update the price of Product C to 45.00.


UPDATE products
SET price = 45.00
WHERE name = 'Product C';

-- 4. Add a new column discount to the products table.


ALTER TABLE products
ADD COLUMN discount DECIMAL(5, 2) DEFAULT 0;

-- 5. Retrieve the top 3 products with the highest price.


SELECT name, price
FROM products
```

```
ORDER BY price DESC
LIMIT 3;

-- 6. Get the names of customers who have 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';

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


SELECT c.name AS customer_name, o.order_date
FROM orders o
JOIN customers c ON o.customer_id = c.id;

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


SELECT id, customer_id, order_date, total_amount
FROM orders
WHERE total_amount > 150.00;

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



-- Step 1: Create the order_items table

CREATE TABLE order_items (
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT DEFAULT 1,
    price DECIMAL(10, 2),
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);

-- Step 2: Modify the orders table to remove product-related information
(if needed)

-- Step 3: Insert sample data into order_items table

INSERT INTO order_items (order_id, product_id, quantity, price)
VALUES
    (1, 1, 2, 20.00),
    (2, 2, 1, 35.00),
    (3, 3, 1, 45.00),
    (4, 1, 1, 20.00);

-- 10. Retrieve the average total of all orders.

SELECT AVG(total_amount) AS average_order_total
FROM orders;
```