Create the database

```
CREATE DATABASE ecommerce;
```

Use the created database

```
USE ecommerce;
```

Create the customers table

```
CREATE TABLE customers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    address VARCHAR(255)
);
```

Create the products table

```
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    description TEXT
);
```

Create the orders table

```
CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    order_date DATE NOT NULL,
    total_amount DECIMAL(10, 2),
    FOREIGN KEY (customer_id) REFERENCES customers(id)
);
```

Insert sample data into customers

```
INSERT INTO customers (name, email, address) VALUES
('John Doe', 'john.doe@example.com', '123 Maple St'),
('Jane Smith', 'jane.smith@example.com', '456 Oak St'),
('Alice Johnson', 'alice.johnson@example.com', '789 Pine St');
```

Insert sample data into products

```
INSERT INTO products (name, price, description) VALUES
('Product A', 25.00, 'Description of Product A'),
('Product B', 30.00, 'Description of Product B'),
('Product C', 40.00, 'Description of Product C');
```

Insert sample data into orders

```
INSERT INTO orders (customer_id, order_date, total_amount) VALUES
(1, CURDATE(), 100.00),
(2, CURDATE() - INTERVAL 10 DAY, 150.00),
(3, CURDATE() - INTERVAL 40 DAY, 200.00);
```

Query 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;
```

Query 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.name;
```

Query 3: Update the price of Product C to 45.00

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

Query 4: Add a new column discount to the products table

```
ALTER TABLE products
ADD COLUMN discount DECIMAL(10, 2) DEFAULT 0.00;
```

Query 5: Retrieve the top 3 products with the highest price

```
SELECT name, price
FROM products
ORDER BY price DESC
LIMIT 3;
```

Query 6: Get the names of customers who have ordered Product A

Assuming a separate order_items table exists for this

```
CREATE TABLE order_items (
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);
```

Insert sample data into order_items

```
INSERT INTO order_items (order_id, product_id, quantity) VALUES
(1, 1, 2), -- John ordered Product A
(2, 2, 1),
(2, 3, 3);

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';
```

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

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

Query 8: Retrieve the orders with a total amount greater than 150.00

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

Query 9: Normalize the database by creating a separate table for order items

This was done earlier by creating the `order_items` table.

Update the `orders` table to remove product details, as it now references `order_items`.

Query 10: Retrieve the average total of all orders

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