```
-- Step 1: Create the `ecommerce` database
CREATE DATABASE ecommerce;

-- Step 2: Use the `ecommerce` database
USE ecommerce;

-- Step 3: Create the `customers` table
CREATE TABLE customers (
    id INT AUTO_INCREMENT PRIMARY KEY,  -- Unique identifier for each customer
    name VARCHAR(100) NOT NULL,         -- Customer's name
    email VARCHAR(100) NOT NULL,        -- Customer's email address
    address VARCHAR(255)                -- Customer's address
);

-- Step 4: Create the `products` table
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,  -- Unique identifier for each product
    name VARCHAR(100) NOT NULL,         -- Product's name
    price DECIMAL(10, 2) NOT NULL,      -- Product's price
    description TEXT                    -- Product's description
);

-- Step 5: Create the `orders` table
CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,  -- Unique identifier for each order
    customer_id INT,                    -- Reference to the customer who placed the order
    order_date DATE,                    -- Date when the order was placed
    total_amount DECIMAL(10, 2),        -- Total amount of the order
    FOREIGN KEY (customer_id) REFERENCES customers(id) -- Foreign key to the customers table
);

-- Step 6: Insert sample data into `customers` table
```

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

-- Step 7: Insert sample data into `products` table
INSERT INTO products (name, price, description)
VALUES
   ('Product A', 25.50, 'Description of Product A'),
   ('Product B', 35.00, 'Description of Product B'),
   ('Product C', 40.00, 'Description of Product C');

-- Step 8: Insert sample data into `orders` table
INSERT INTO orders (customer_id, order_date, total_amount)
VALUES
   (1, '2024-12-01', 100.00),
   (2, '2024-12-15', 200.00),
   (3, '2024-12-20', 150.00);

-- Step 9: Retrieve all customers who have placed an order in the last 30 days
SELECT DISTINCT c.name, c.email, c.address
FROM customers c
JOIN orders o ON c.id = o.customer_id
WHERE o.order_date >= CURDATE() - INTERVAL 30 DAY;

-- Step 10: 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;
```

```
-- Step 11: Update the price of Product C to 45.00
UPDATE products
SET price = 45.00
WHERE name = 'Product C';

-- Step 12: Add a new column `discount` to the `products` table
ALTER TABLE products
ADD COLUMN discount DECIMAL(5, 2) DEFAULT 0;

-- Step 13: Retrieve the top 3 products with the highest price
SELECT * FROM products
ORDER BY price DESC
LIMIT 3;

-- Step 14: 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';

-- Step 15: 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;

-- Step 16: Retrieve the orders with a total amount greater than 150.00
SELECT * FROM orders
WHERE total_amount > 150.00;
```

```
-- Step 17: Normalize the database by creating a separate table for `order_items` and update the `orders` table

-- Create the `order_items` table to store individual products in each order
CREATE TABLE order_items (
    id INT AUTO_INCREMENT PRIMARY KEY,  -- Unique identifier for each order item
    order_id INT,                   -- Reference to the order
    product_id INT,                 -- Reference to the product
    quantity INT DEFAULT 1,         -- Quantity of the product ordered
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);


-- Step 18: Retrieve the average total of all orders
SELECT AVG(total_amount) AS average_order_total
FROM orders;
```