CODING ASSESSMENT

create database Ecommerce;

//create tables

CREATE TABLE customers (

customerID INT PRIMARY KEY ,

firstName VARCHAR(50) NOT NULL,

lastName VARCHAR(50) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

address VARCHAR(255) NOT NULL

);

CREATE TABLE products (

productID INT PRIMARY KEY ,

name VARCHAR(100) NOT NULL,

description TEXT,

price DECIMAL(10,2) NOT NULL,

stockQuantity INT NOT NULL

);

CREATE TABLE orders (

orderID INT PRIMARY KEY IDENTITY(1,1),

customerID INT NOT NULL,

orderDate DATETIME DEFAULT GETDATE(),

totalAmount DECIMAL(10,2) NOT NULL,

FOREIGN KEY (customerID) REFERENCES customers(customerID) ON DELETE CASCADE

);

CREATE TABLE order\_items (

orderItemID INT PRIMARY KEY ,

orderID INT NOT NULL,

productID INT NOT NULL,

quantity INT NOT NULL CHECK (quantity > 0),

itemAmount DECIMAL(10,2) NOT NULL,

FOREIGN KEY (orderID) REFERENCES orders(orderID) ON DELETE CASCADE,

FOREIGN KEY (productID) REFERENCES products(productID) ON DELETE CASCADE

);

CREATE TABLE cart (

cartID INT PRIMARY KEY IDENTITY(1,1),

customerID INT NOT NULL,

productID INT NOT NULL,

quantity INT NOT NULL CHECK (quantity > 0),

FOREIGN KEY (customerID) REFERENCES customers(customerID) ON DELETE CASCADE,

FOREIGN KEY (productID) REFERENCES products(productID) ON DELETE CASCADE

);

--Insert values

INSERT INTO customers (customerID, firstname, lastname, email, address)

VALUES

(1, 'John', 'Doe', 'johndoe@example.com', '123 Main St, City'),

(2, 'Jane', 'Smith', 'janesmith@example.com', '456 Elm St, Town'),

(3, 'Robert', 'Johnson', 'robert@example.com', '789 Oak St, Village'),

(4, 'Sarah', 'Brown', 'sarah@example.com', '101 Pine St, Suburb'),

(5, 'David', 'Lee', 'david@example.com', '234 Cedar St, District'),

(6, 'Laura', 'Hall', 'laura@example.com', '567 Birch St, County'),

(7, 'Michael', 'Davis', 'michael@example.com', '890 Maple St, State'),

(8, 'Emma', 'Wilson', 'emma@example.com', '321 Redwood St, Country'),

(9, 'William', 'Taylor', 'william@example.com', '432 Spruce St, Province'),

(10, 'Olivia', 'Adams', 'olivia@example.com', '765 Fir St, Territory');

INSERT INTO orders ( customerID, orderDate, totalAmount)

VALUES

(1, '2023-01-05', 1200.00),

(2, '2023-02-10', 900.00),

(3, '2023-03-15', 300.00),

(4, '2023-04-20', 150.00),

(5, '2023-05-25', 1800.00),

(6, '2023-06-30', 400.00),

(7, '2023-07-05', 700.00),

(8, '2023-08-10', 160.00),

(9, '2023-09-15', 140.00),

(10, '2023-10-20', 1400.00);

INSERT INTO products (productID, name, Description, price, stockQuantity)

VALUES

(1, 'Laptop', 'High-performance laptop', 1200.00, 10),

(2, 'Smartphone', 'Latest smartphone', 800.00, 10),

(3, 'Tablet', 'Portable tablet', 600.00, 15),

(4, 'Headphones', 'Noise-canceling headphones', 300.00, 20),

(5, 'TV', '4K Smart TV', 1500.00, 30),

(6, 'Coffee Maker', 'Automatic coffee maker', 50.00, 25),

(7, 'Refrigerator', 'Energy-efficient', 700.00, 10),

(8, 'Microwave Oven', 'Countertop microwave', 80.00, 15),

(9, 'Blender', 'High-speed blender', 70.00, 20),

(10, 'Vacuum Cleaner', 'Bagless vacuum cleaner', 120.00, 10);

INSERT INTO Order\_Items (orderItemID, orderID, productID, quantity, itemAmount)

VALUES

(1, 1, 1, 2, 1600.00),

(2, 1, 3, 1, 300.00),

(3, 2, 2, 3, 1800.00),

(4, 3, 5, 2, 1800.00),

(5, 4, 4, 4, 600.00),

(6, 4, 6, 1, 50.00),

(7, 5, 1, 1, 800.00),

(8, 5, 2, 2, 1200.00),

(9, 6, 10, 2, 240.00),

(10, 6, 9, 3, 210.00);

INSERT INTO cart (customerID, productID, quantity)

VALUES

(1, 1, 2),

(1, 3, 1),

(2, 2, 3),

(3, 4, 4),

(3, 5, 2),

(4, 6, 1),

(5, 1, 1),

(6, 10, 2),

(6, 9, 3),

(7, 7, 2);

//queries

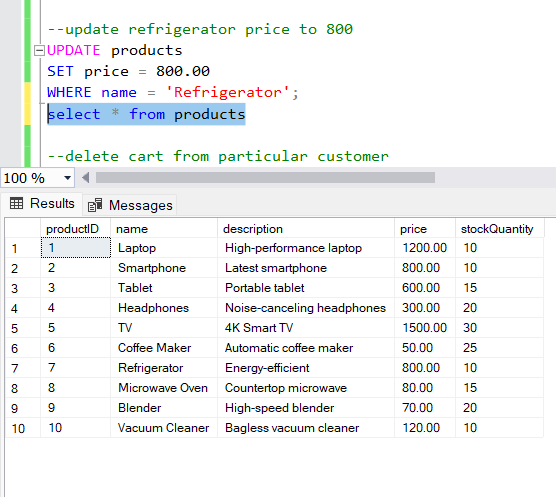
1. Update refrigerator product price to 800.

UPDATE products

SET price = 800.00

WHERE name = 'Refrigerator';

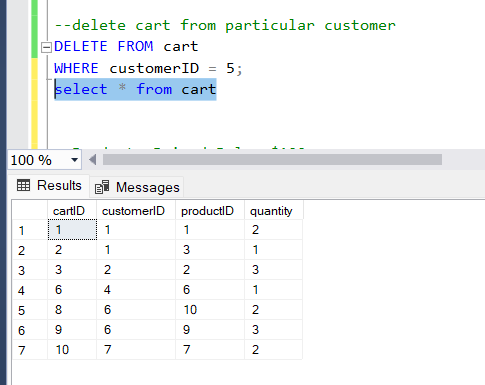
select \* from products



2.delete cart from particular customer

DELETE FROM cart

WHERE customerID = 3;

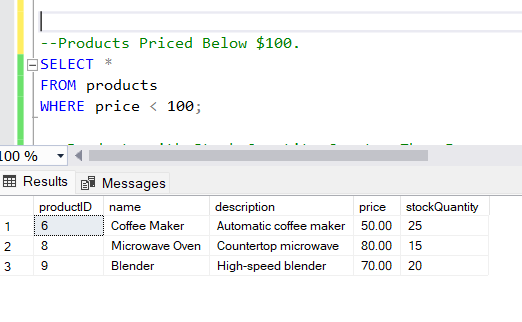


3. Products Priced Below $100.

SELECT \*

FROM products

WHERE price < 100;

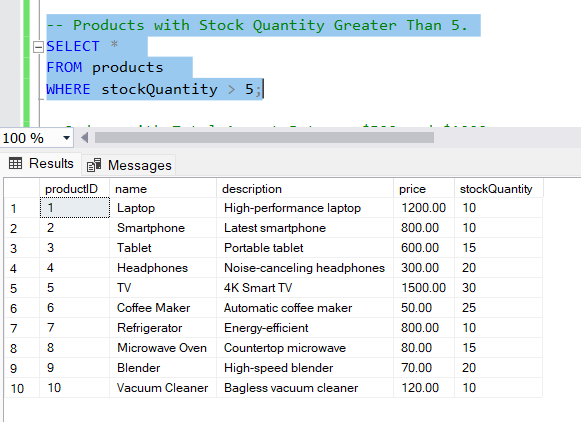


-- Products with Stock Quantity Greater Than 5.

SELECT \*

FROM products

WHERE stockQuantity > 5;

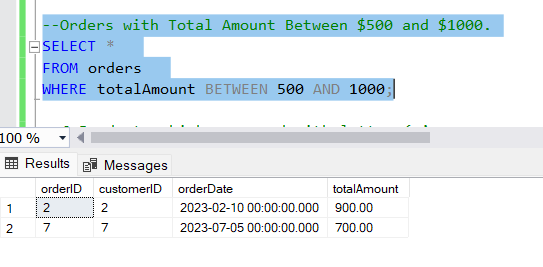


--Orders with Total Amount Between $500 and $1000.

SELECT \*

FROM orders

WHERE totalAmount BETWEEN 500 AND 1000;

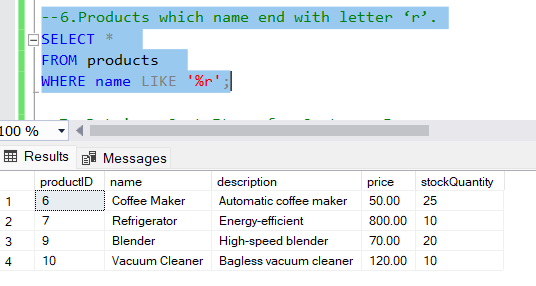


--6.Products which name end with letter ‘r’.

SELECT \*

FROM products

WHERE name LIKE '%r';



--7. Retrieve Cart Items for Customer 5.

SELECT \*

FROM cart

WHERE customerID = 5;

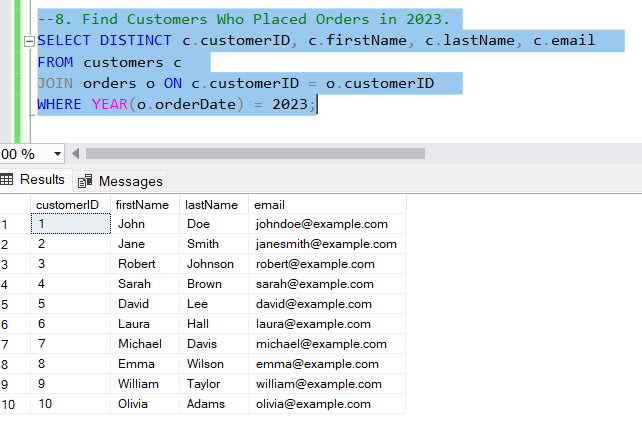
--8. Find Customers Who Placed Orders in 2023.

SELECT DISTINCT c.customerID, c.firstName, c.lastName, c.email

FROM customers c

JOIN orders o ON c.customerID = o.customerID

WHERE YEAR(o.orderDate) = 2023;

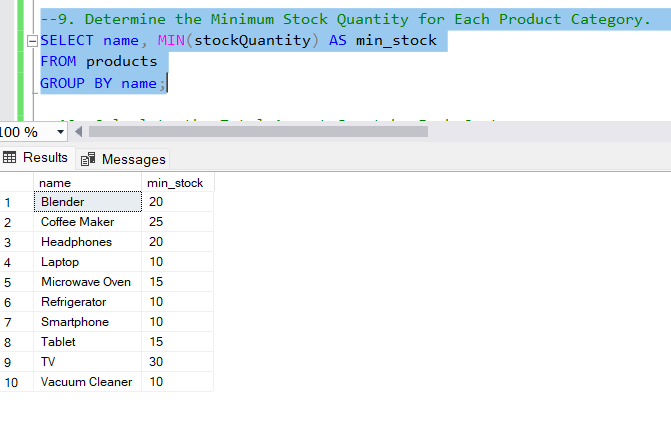


--9. Determine the Minimum Stock Quantity for Each Product Category.

SELECT name, MIN(stockQuantity) AS min\_stock

FROM products

GROUP BY name;



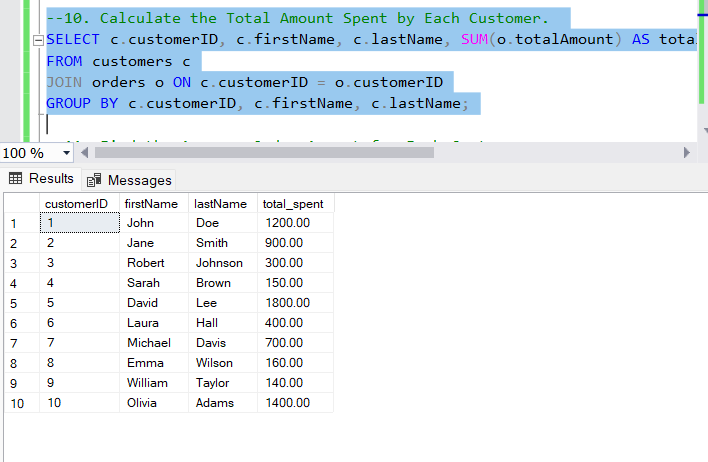
--10. Calculate the Total Amount Spent by Each Customer.

SELECT c.customerID, c.firstName, c.lastName, SUM(o.totalAmount) AS total\_spent

FROM customers c

JOIN orders o ON c.customerID = o.customerID

GROUP BY c.customerID, c.firstName, c.lastName;



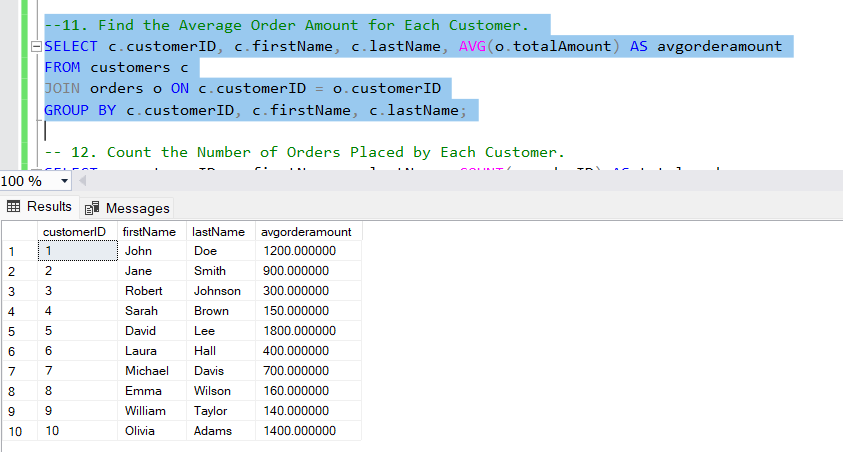
--11. Find the Average Order Amount for Each Customer.

SELECT c.customerID, c.firstName, c.lastName, AVG(o.totalAmount) AS avgorderamount

FROM customers c

JOIN orders o ON c.customerID = o.customerID

GROUP BY c.customerID, c.firstName, c.lastName;



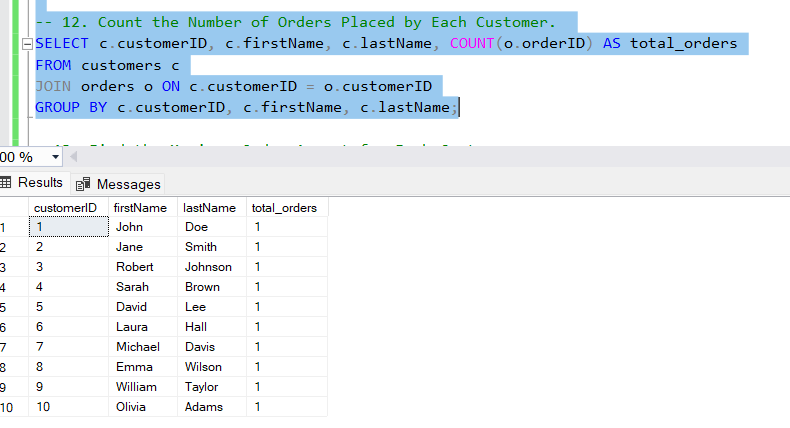
-- 12. Count the Number of Orders Placed by Each Customer.

SELECT c.customerID, c.firstName, c.lastName, COUNT(o.orderID) AS total\_orders

FROM customers c

JOIN orders o ON c.customerID = o.customerID

GROUP BY c.customerID, c.firstName, c.lastName;



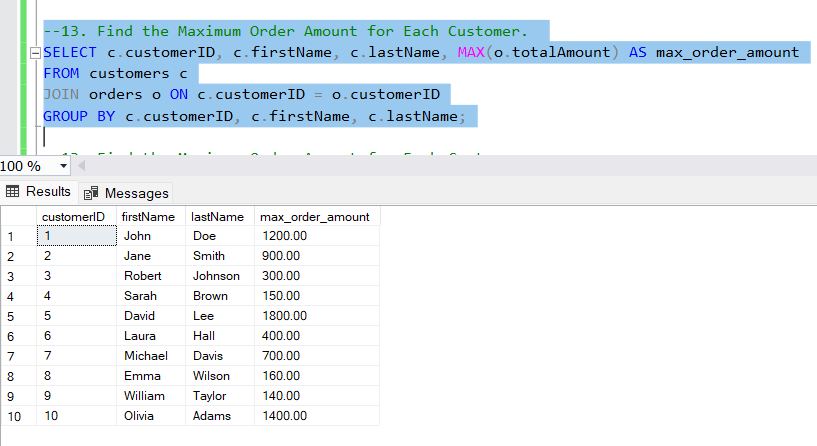
--13. Find the Maximum Order Amount for Each Customer.

SELECT c.customerID, c.firstName, c.lastName, MAX(o.totalAmount) AS max\_order\_amount

FROM customers c

JOIN orders o ON c.customerID = o.customerID

GROUP BY c.customerID, c.firstName, c.lastName;



--14. Get Customers Who Placed Orders Totaling Over $1000.

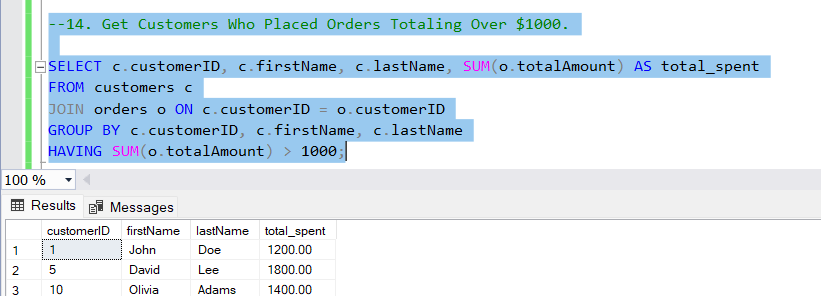
SELECT c.customerID, c.firstName, c.lastName, SUM(o.totalAmount) AS total\_spent

FROM customers c

JOIN orders o ON c.customerID = o.customerID

GROUP BY c.customerID, c.firstName, c.lastName

HAVING SUM(o.totalAmount) > 1000;

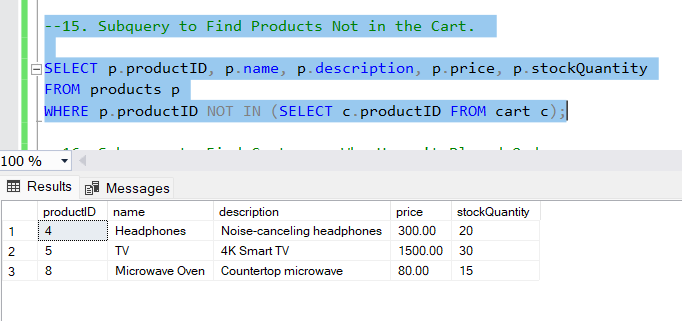


--15. Subquery to Find Products Not in the Cart.

SELECT p.productID, p.name, p.description, p.price, p.stockQuantity

FROM products p

WHERE p.productID NOT IN (SELECT c.productID FROM cart c);



--16. Subquery to Find Customers Who Haven't Placed Orders.

SELECT c.customerID, c.firstName, c.lastName, c.email

FROM customers c

WHERE c.customerID NOT IN (SELECT o.customerID FROM orders o);

--17. Subquery to Calculate the Percentage of Total Revenue for a Product.

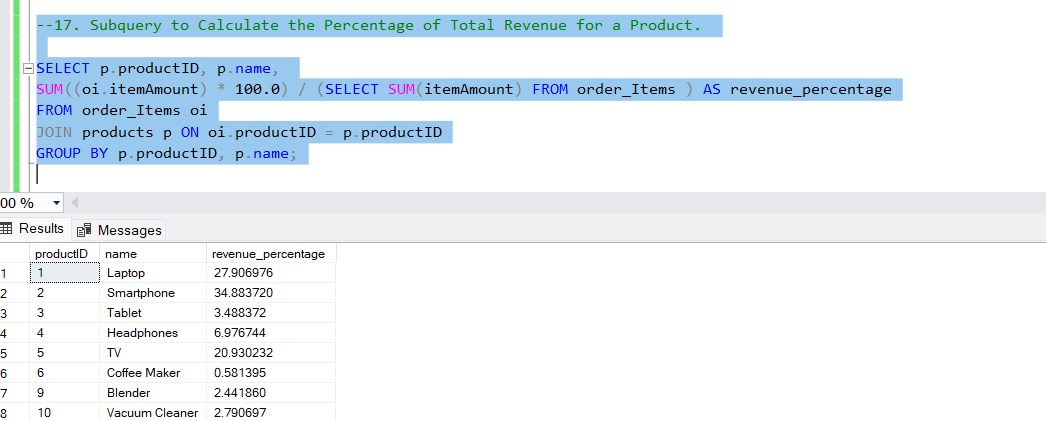
SELECT p.productID, p.name,

SUM((oi.itemAmount) \* 100.0) / (SELECT SUM(itemAmount) FROM order\_Items ) AS revenue\_percentage

FROM order\_Items oi

JOIN products p ON oi.productID = p.productID

GROUP BY p.productID, p.name;

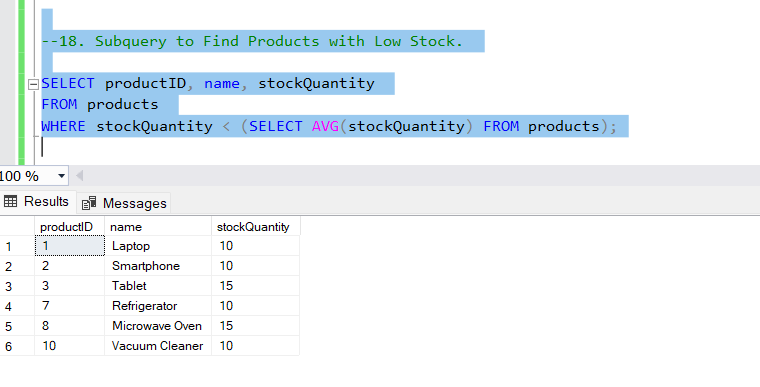


--18. Subquery to Find Products with Low Stock.

SELECT productID, name, stockQuantity

FROM products

WHERE stockQuantity < (SELECT AVG(stockQuantity) FROM products);



--19. Subquery to Find Customers Who Placed High-Value Orders.

SELECT customerID, firstName, lastName, email

FROM customers

WHERE customerID IN (

SELECT DISTINCT o.customerID

FROM orders o

WHERE o.totalAmount > (SELECT AVG(totalAmount) FROM orders)

);

