Ecommerce SQL Challenge

# Database and Tables Creation

CREATE DATABASE ecommerce\_db;  
USE ecommerce\_db;  
  
  
CREATE TABLE customers (

customer\_id INT PRIMARY KEY,

first\_name VARCHAR(100),

last\_name VARCHAR(100),

email VARCHAR(100),

password VARCHAR(100)

);  
  
CREATE TABLE products (  
 product\_id INT PRIMARY KEY,  
 name VARCHAR(100),  
 price DECIMAL(10,2),  
 description VARCHAR(255),  
 stockQuantity INT  
);  
  
  
 CREATE TABLE cart (  
 cart\_id INT PRIMARY KEY,  
 customer\_id INT,  
 product\_id INT,  
 quantity INT,  
 FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id),  
 FOREIGN KEY (product\_id) REFERENCES products(product\_id)  
);  
  
  
CREATE TABLE orders (  
 order\_id INT PRIMARY KEY,  
 customer\_id INT,  
 order\_date DATE,  
 total\_price DECIMAL(10,2),  
 shipping\_address VARCHAR(255),  
 FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)  
);  
  
  
CREATE TABLE order\_items (  
 order\_item\_id INT PRIMARY KEY,  
 order\_id INT,  
 product\_id INT,  
 quantity INT,  
 item\_amount DECIMAL(10,2),  
 FOREIGN KEY (order\_id) REFERENCES orders(order\_id),  
 FOREIGN KEY (product\_id) REFERENCES products(product\_id)  
);

**Insert Data into Tables**

INSERT INTO products (product\_id, name, description, price, stockQuantity) VALUES  
(1, 'Laptop', 'High-performance laptop', 800.00, 10),  
(2, 'Smartphone', 'Latest smartphone', 600.00, 15),  
(3, 'Tablet', 'Portable tablet', 300.00, 20),  
(4, 'Headphones', 'Noise-canceling', 150.00, 30),  
(5, 'TV', '4K Smart TV', 900.00, 5),  
(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 customers (customer\_id, first\_name, last\_name, email, password) 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'),

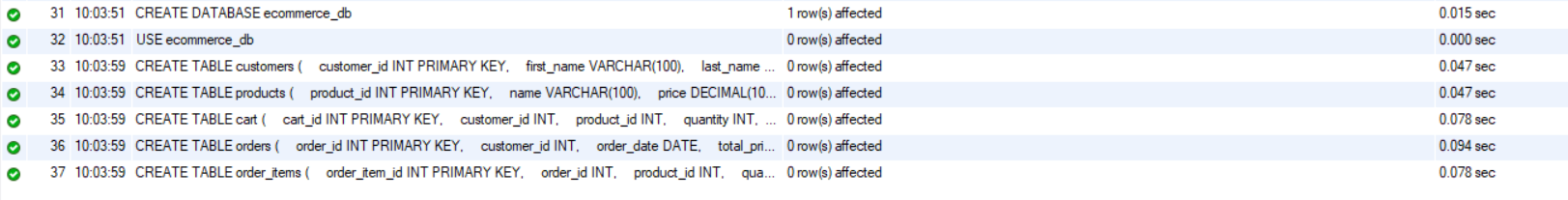
(8, 'Emma', 'Wilson', 'emma@example.com', '321 Redwood St, Country'),

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

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

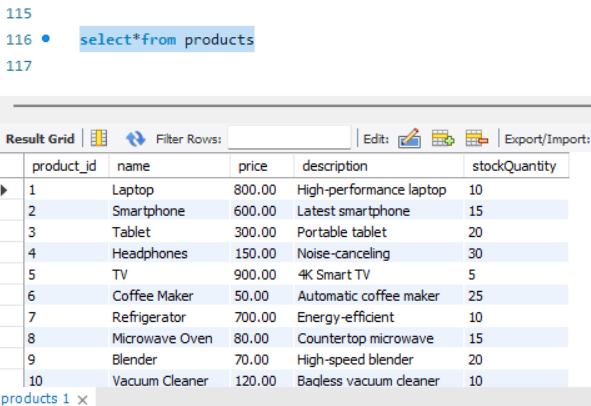
INSERT INTO orders (order\_id, customer\_id, order\_date, total\_price, shipping\_address) VALUES  
(1, 1, '2023-01-05', 1200.00, '123 Main St, City'),  
(2, 2, '2023-02-10', 900.00, '456 Elm St, Town'),  
(3, 3, '2023-03-15', 300.00, '789 Oak St, Village'),  
(4, 4, '2023-04-20', 150.00, '101 Pine St, Suburb'),  
(5, 5, '2023-05-25', 1800.00, '234 Cedar St, District'),  
(6, 6, '2023-06-30', 400.00, '567 Birch St, County'),  
(7, 7, '2023-07-05', 700.00, '890 Maple St, State'),  
(8, 8, '2023-08-10', 160.00, '321 Redwood St, Country'),  
(9, 9, '2023-09-15', 140.00, '432 Spruce St, Province'),  
(10, 10, '2023-10-20', 1400.00, '765 Fir St, Territory');  
  
  
INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity, item\_amount) 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 (cart\_id, customer\_id, product\_id, quantity) VALUES  
(1, 1, 1, 2),  
(2, 1, 3, 1),  
(3, 2, 2, 3),  
(4, 3, 4, 4),  
(5, 3, 5, 2),  
(6, 4, 6, 1),  
(7, 5, 1, 1),  
(8, 6, 10, 2),  
(9, 6, 9, 3),  
(10, 7, 7, 2);

**For table creation**

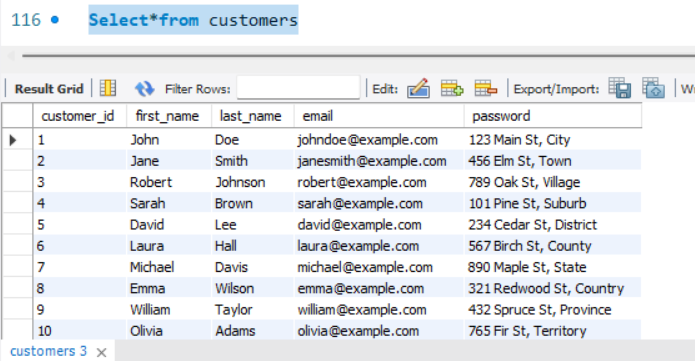


**Inserting values**

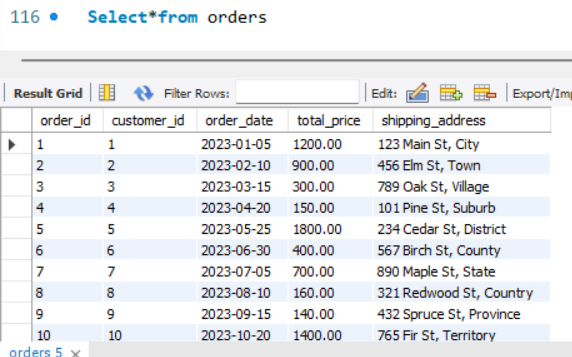
Product Table



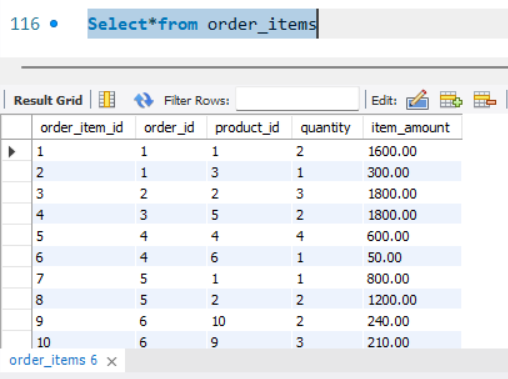
Customers Table



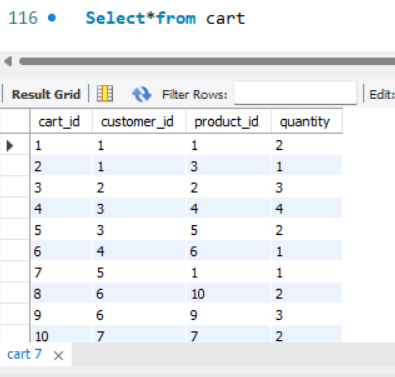
Orders table



Orders Items Table



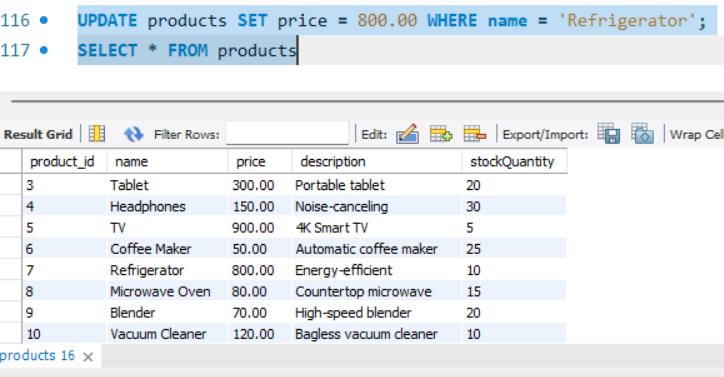
Cart table



# SQL Queries for Given Questions

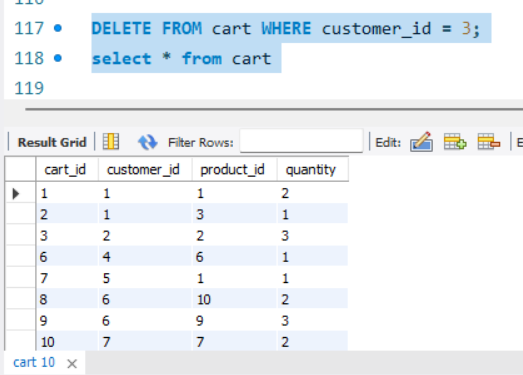
1. Update refrigerator product price to 800.

UPDATE products SET price = 800.00 WHERE name = 'Refrigerator';



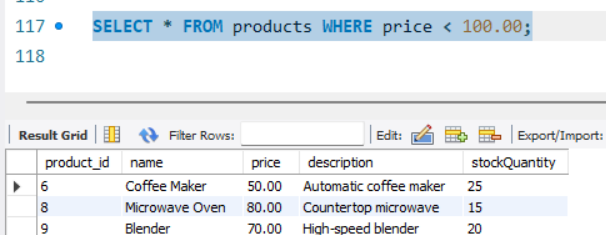
2. Remove all cart items for a specific customer (e.g., customer\_id = 3).

DELETE FROM cart WHERE customer\_id = 3;



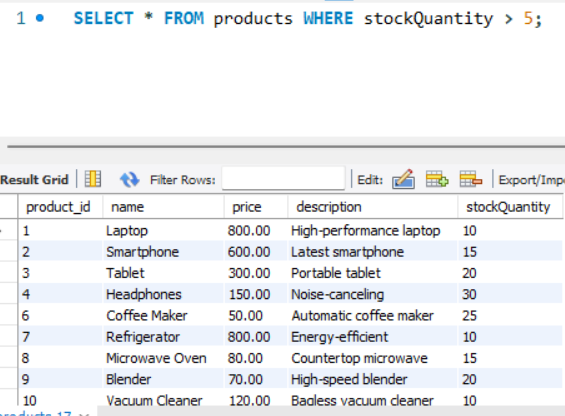
3. Retrieve Products Priced Below $100.

SELECT \* FROM products WHERE price < 100.00;



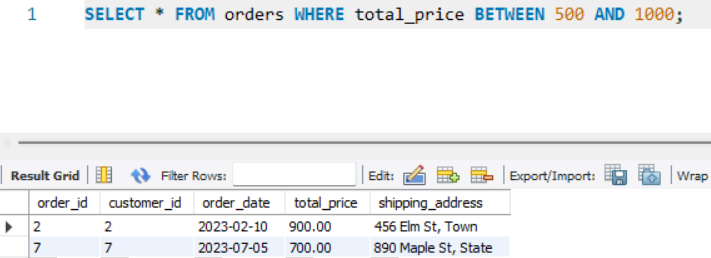
4. Find Products with Stock Quantity Greater Than 5.

SELECT \* FROM products WHERE stockQuantity > 5;



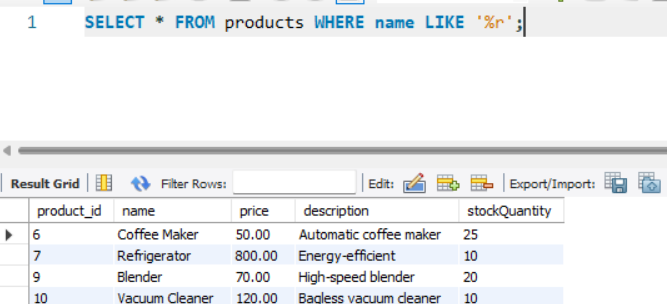
5. Retrieve Orders with Total Amount Between $500 and $1000.

SELECT \* FROM orders WHERE total\_price BETWEEN 500 AND 1000;



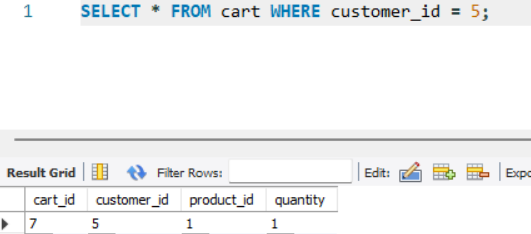
6. Find Products which name end with letter ‘r’.

SELECT \* FROM products WHERE name LIKE '%r';



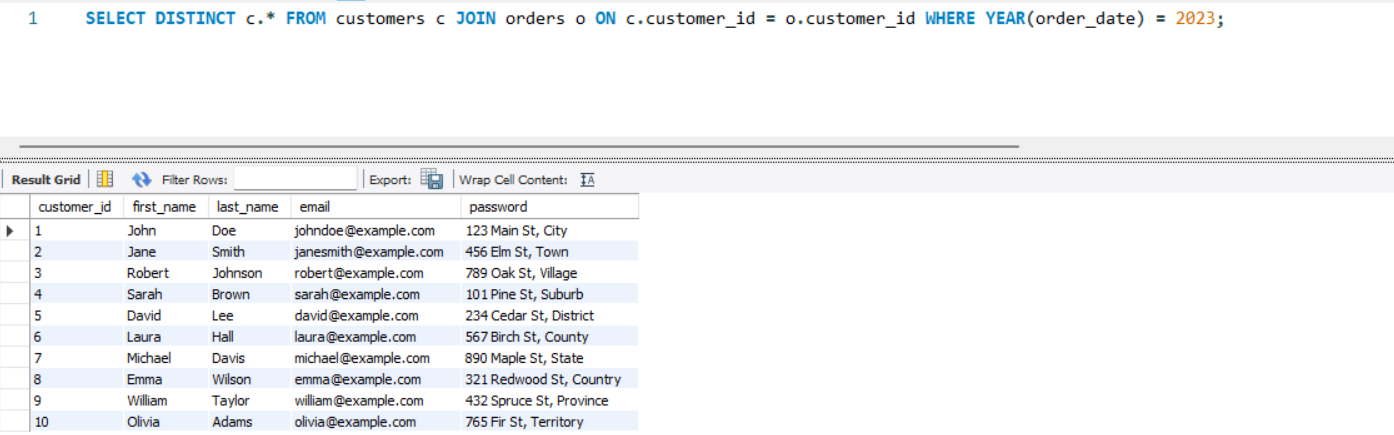
7. Retrieve Cart Items for Customer 5.

SELECT \* FROM cart WHERE customer\_id = 5;



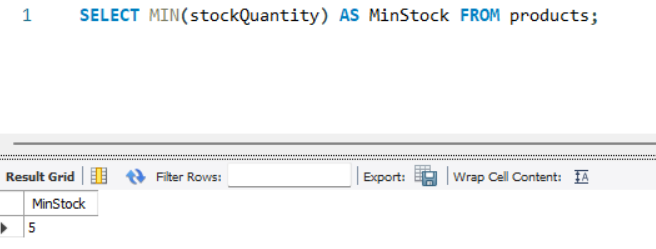
8. Find Customers Who Placed Orders in 2023.

SELECT DISTINCT c.\* FROM customers c JOIN orders o ON c.customer\_id = o.customer\_id WHERE YEAR(order\_date) = 2023;



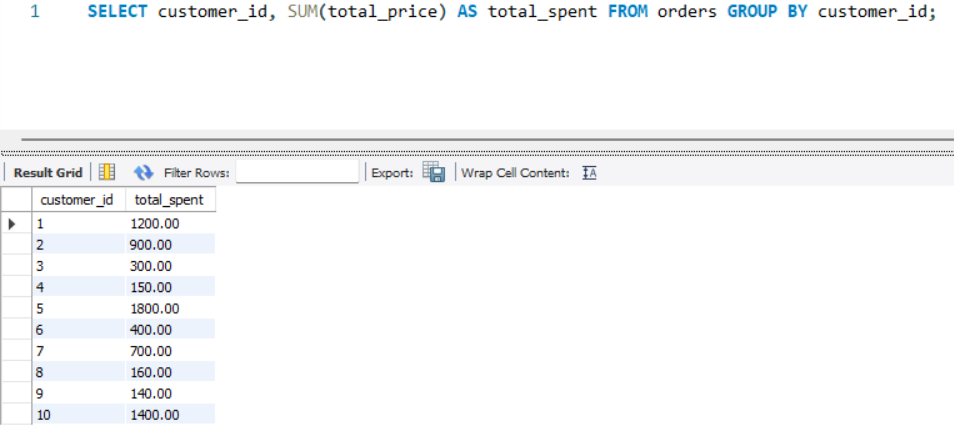
9. Determine the Minimum Stock Quantity for Each Product .

SELECT MIN(stockQuantity) AS MinStock FROM products;



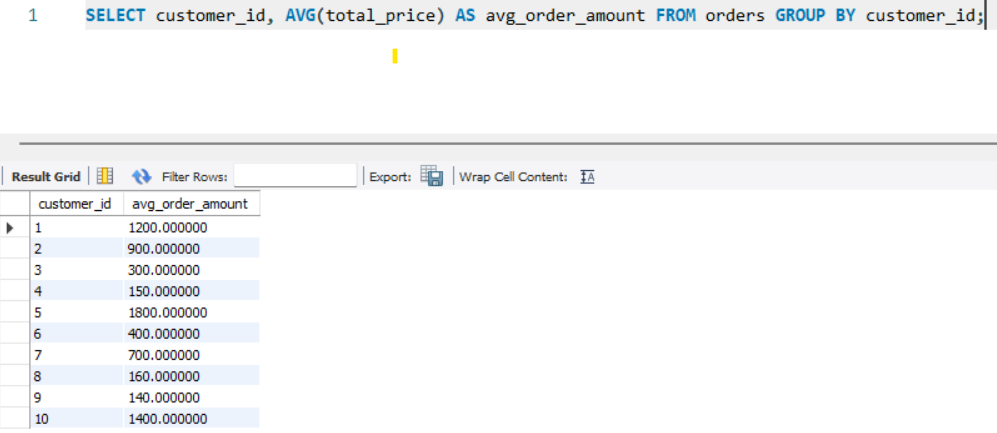
10. Calculate the Total Amount Spent by Each Customer.

SELECT customer\_id, SUM(total\_price) AS total\_spent FROM orders GROUP BY customer\_id;



11. Find the Average Order Amount for Each Customer.

SELECT customer\_id, AVG(total\_price) AS avg\_order\_amount FROM orders GROUP BY customer\_id;



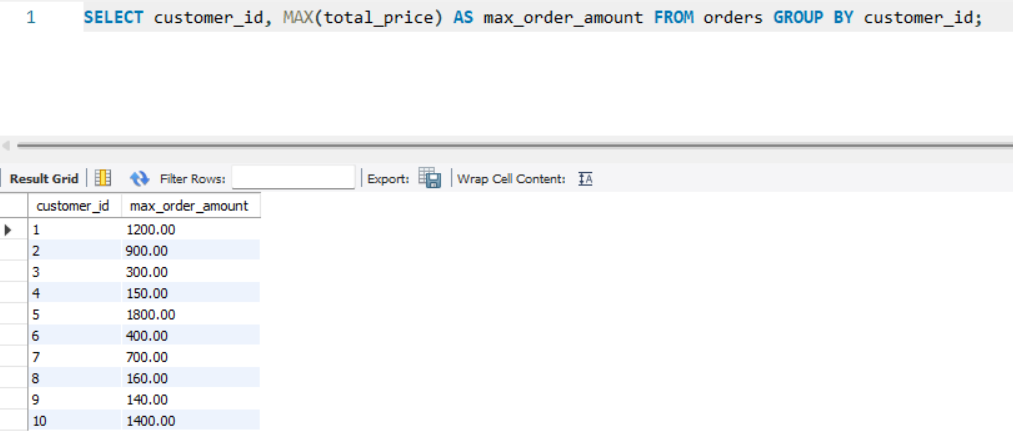
12. Count the Number of Orders Placed by Each Customer.

SELECT customer\_id, COUNT(order\_id) AS order\_count FROM orders GROUP BY customer\_id;



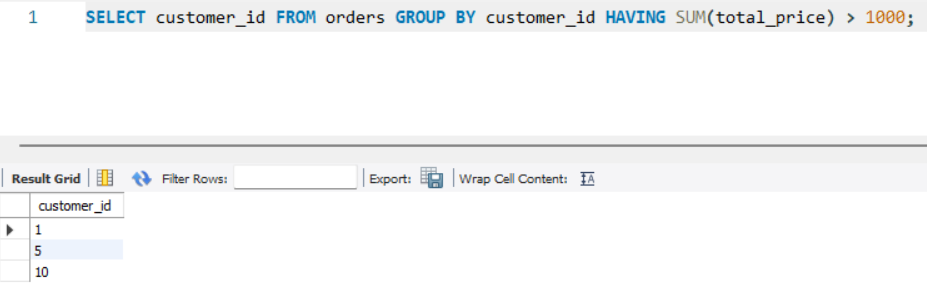
13. Find the Maximum Order Amount for Each Customer.

SELECT customer\_id, MAX(total\_price) AS max\_order\_amount FROM orders GROUP BY customer\_id;



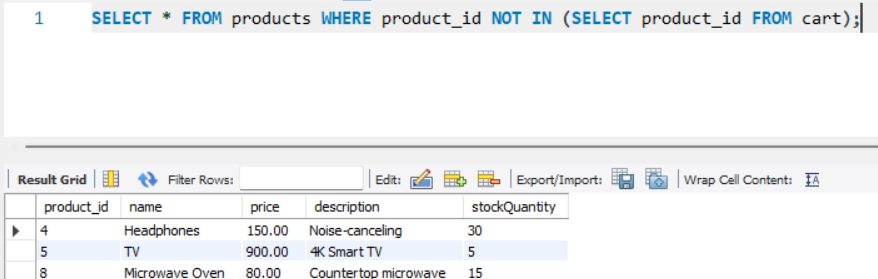
14. Get Customers Who Placed Orders Totaling Over $1000.

SELECT customer\_id FROM orders GROUP BY customer\_id HAVING SUM(total\_price) > 1000;



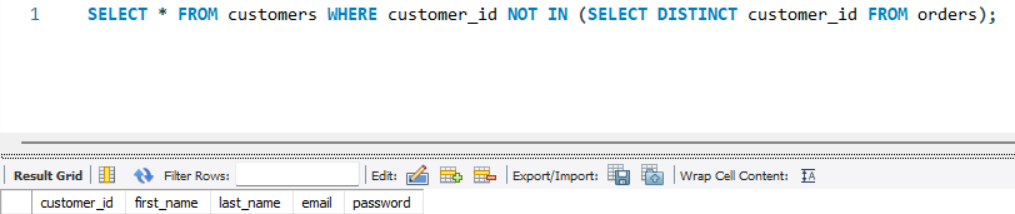
15. Subquery to Find Products Not in the Cart.

SELECT \* FROM products WHERE product\_id NOT IN (SELECT product\_id FROM cart);



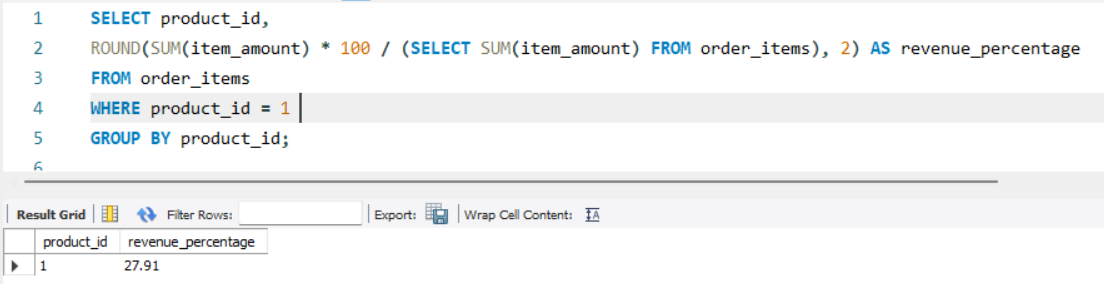
16. Subquery to Find Customers Who Haven't Placed Orders.

SELECT \* FROM customers WHERE customer\_id NOT IN (SELECT DISTINCT customer\_id FROM orders);

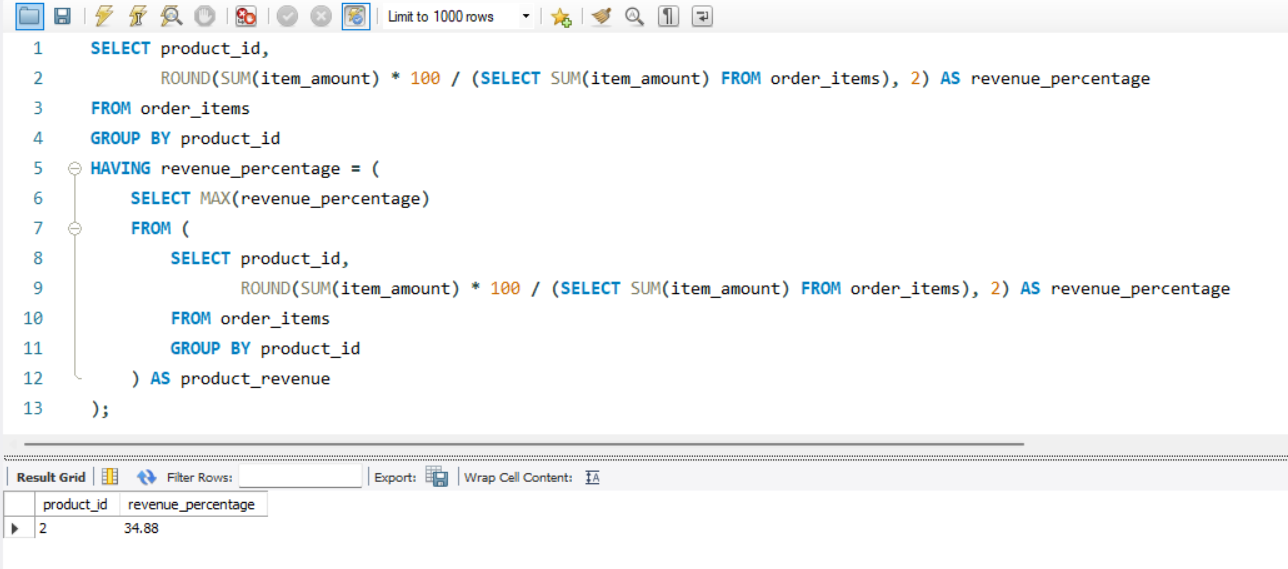


17. Subquery to Calculate the Percentage of Total Revenue for a Product .

SELECT product\_id,   
ROUND(SUM(item\_amount) \* 100 / (SELECT SUM(item\_amount) FROM order\_items), 2) AS revenue\_percentage   
FROM order\_items   
WHERE product\_id = 1   
GROUP BY product\_id;

Only for the particular product (product id=1)

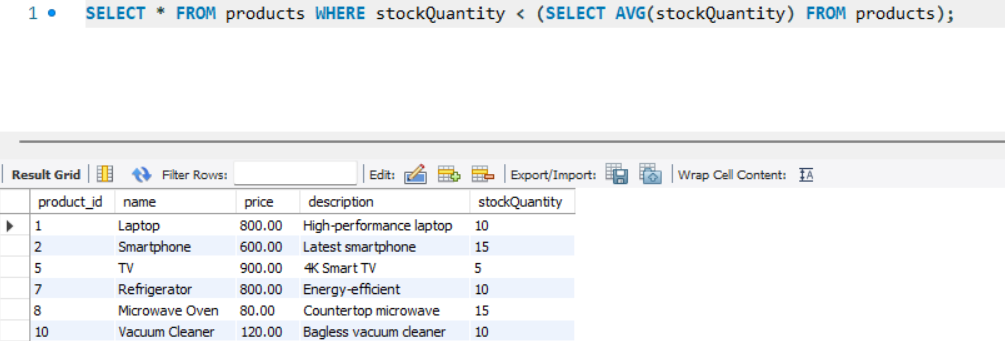
Example : Using Max() to get the total revenue of the product



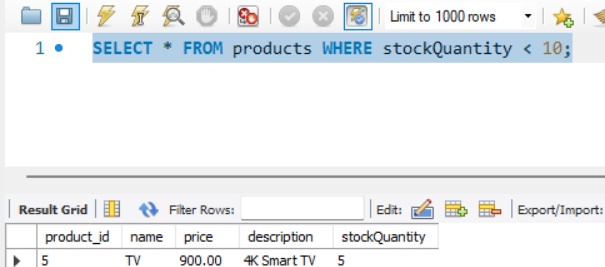
18. Subquery to Find Products with Low Stock .

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

Using Avg() to find product with low stock



Example : Subquery – Products with Stock Below a Threshold (e.g., < 10)



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

SELECT \* FROM customers WHERE customer\_id IN (

SELECT customer\_id

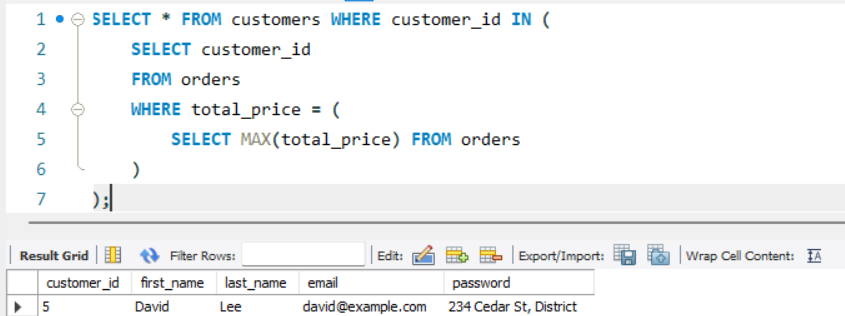
FROM orders

WHERE total\_price = (

SELECT MAX(total\_price) FROM orders

));

Using max() function to find the High-Value Orders



Example : For customers greater than 1000 (high value order)

