**Instructions:**

* Answer all questions using **MySQL**.
* Use appropriate **subqueries**, **joins**, and **aggregate functions** wherever applicable.
* Make sure to use proper **aliasing**, **GROUP BY**, **HAVING**, **DISTINCT**, etc., as needed.
* Data

-- Customers Table

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100),

City VARCHAR(100)

);

-- Orders Table

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

Amount DECIMAL(10,2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

-- Products Table

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Price DECIMAL(10,2)

);

-- OrderDetails Table

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT,

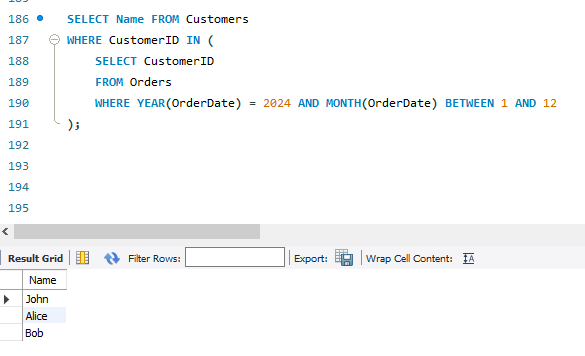
FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

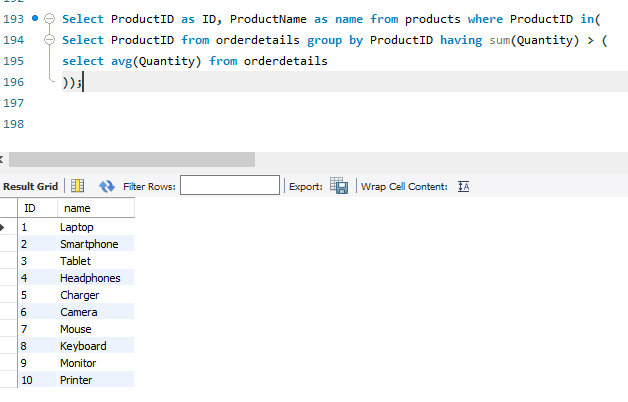
FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

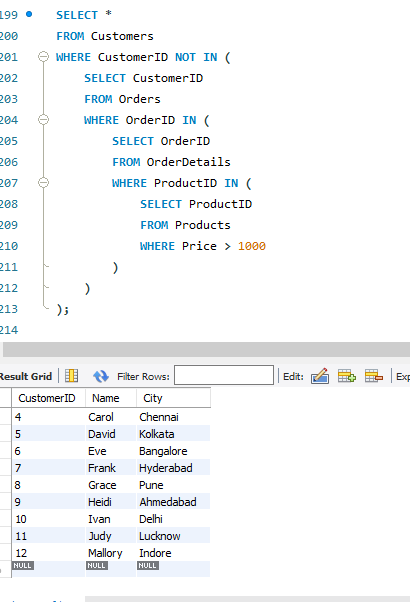
);

**Part A – Subqueries (20 marks)**

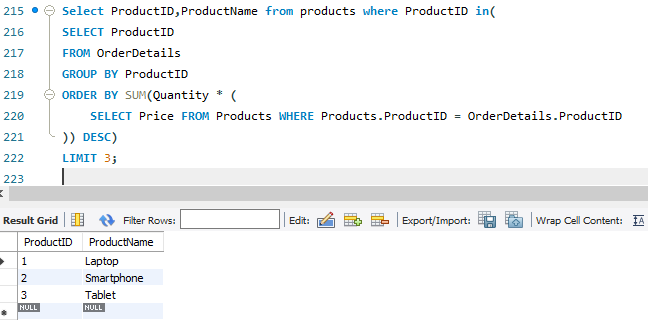
1. Write a query to find customers who have placed orders in **every month** of the current year.



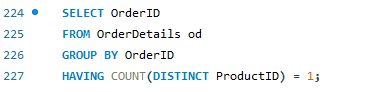
1. Retrieve the names of products that have been ordered **more than the average quantity** across all products. 
2. Find customers who have **never ordered a product** priced above ₹1000.

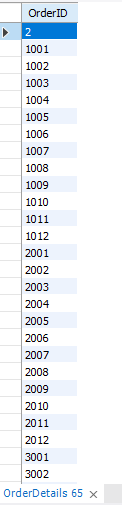
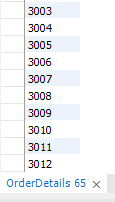


1. List the **top 3 products by total revenue** using a subquery.



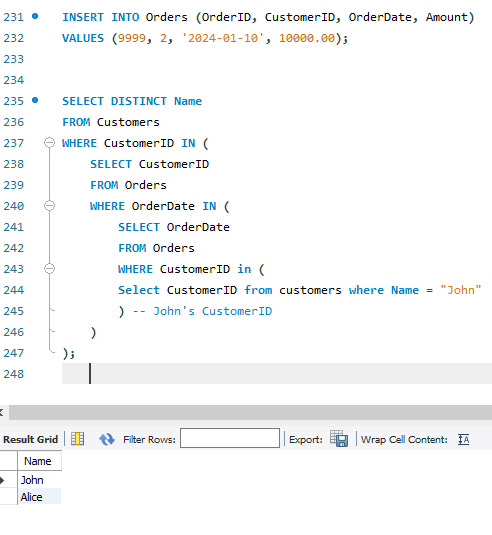
1. Find orders that contain **only one product** using a **correlated subquery**.



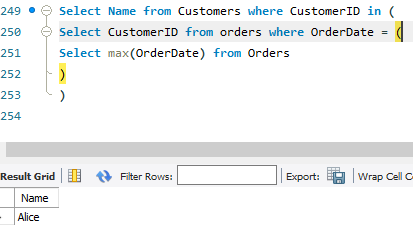
 

**Part B – Correlated & Nested Subqueries (25 marks)**

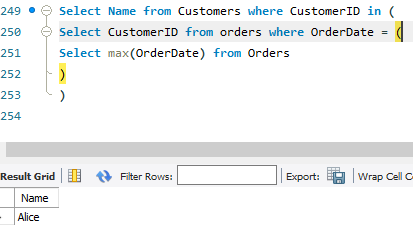
1. Retrieve the names of customers who placed an order on the **same date as 'John'**.



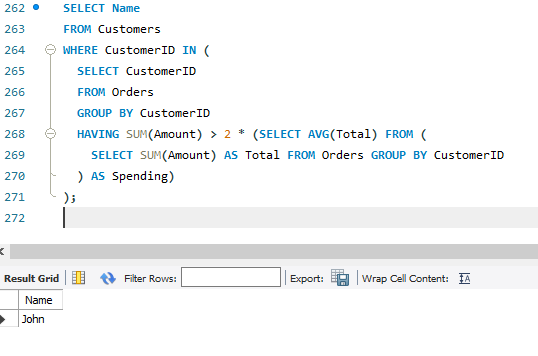
1. Find the name of the customer who placed the **most recent order**.



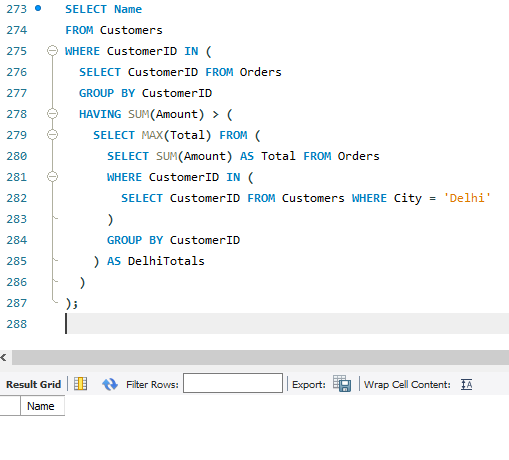
1. Write a query to find the product that has the **second lowest price** using a subquery.



1. Display customer names who have spent **more than double the average spending**.

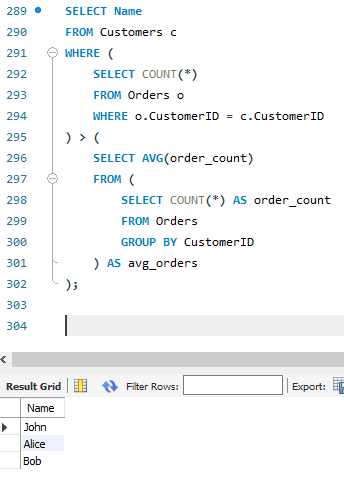


1. List customers whose **total order amount is more than the total order amount of any customer from 'Delhi'**.

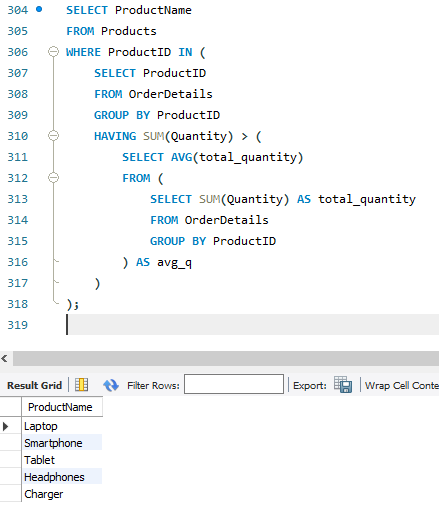


**Part C – Join + Subquery Mix (30 marks)**

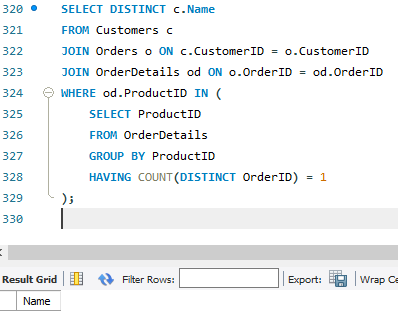
1. Use a correlated subquery to find customers who have placed **more orders than the average** number of orders placed by all customers.



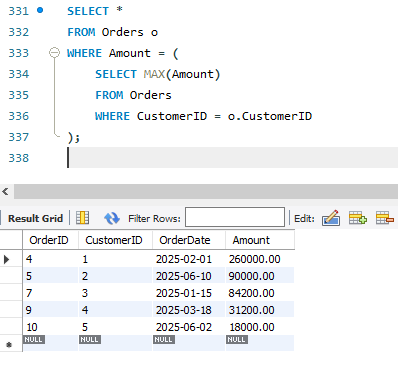
1. Find all products whose **total sales quantity** is higher than the average total quantity sold per product.



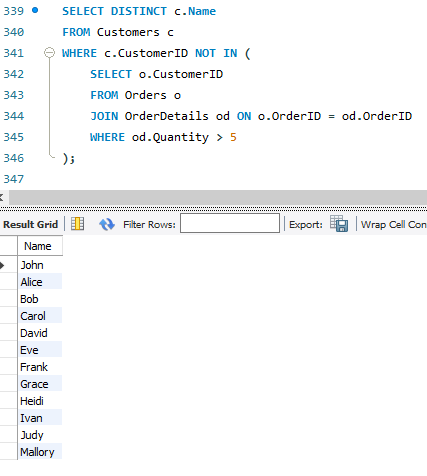
1. Get customers who have ordered at least **one product that no one else has ordered**.



1. Retrieve all orders where the total order amount is equal to the **maximum order amount for that customer**.

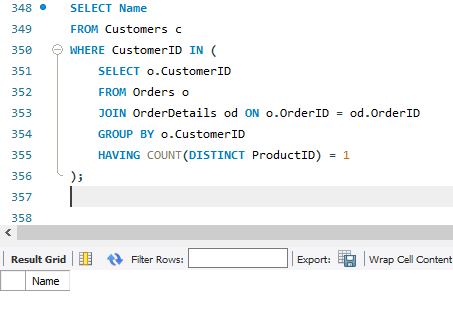


1. Write a query to list customers who have **never placed an order with a quantity greater than 5**.

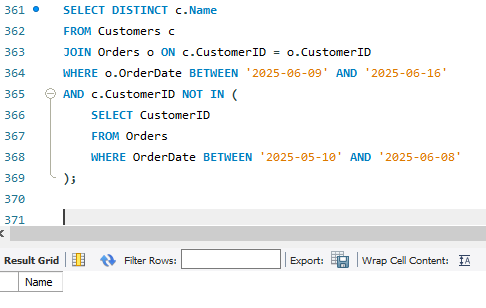


**Part D – Joins & Set Operations (25 marks)**

1. Use a subquery to list the **top 5 customers by total spending**.
2. Find all customers who have only ordered **one unique product** using subqueries.



1. List all orders where the amount is **not in the top 10 highest order amounts**.
2. Retrieve customer names who placed an order in the **last 7 days** but **not** in the **previous 30 days** before that.



1. Write a query to list all products ordered in the **highest number of distinct orders**.

