**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.

SELECT Name

FROM Customers C

WHERE NOT EXISTS (

SELECT DISTINCT MONTH(OrderDate)

FROM Orders

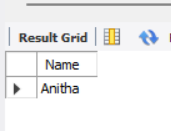
WHERE YEAR(OrderDate) = 2025

EXCEPT

SELECT DISTINCT MONTH(O.OrderDate)

FROM Orders O

WHERE C.CustomerID = O.CustomerID AND YEAR(O.OrderDate) = 2025

);

1. Retrieve the names of products that have been ordered **more than the average quantity** across all products.

SELECT P.ProductName

FROM Products P

JOIN OrderDetails OD ON P.ProductID = OD.ProductID

GROUP BY P.ProductID, P.ProductName

HAVING SUM(OD.Quantity) > (

SELECT AVG(TotalQty)

FROM (

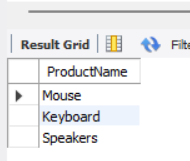
SELECT SUM(Quantity) AS TotalQty

FROM OrderDetails

GROUP BY ProductID

) AS AvgQuantities

);

****

1. Find customers who have **never ordered a product** priced above ₹1000.

SELECT DISTINCT C.Name

FROM Customers C

WHERE C.CustomerID NOT IN (

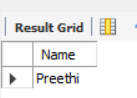
SELECT DISTINCT O.CustomerID

FROM Orders O

JOIN OrderDetails OD ON O.OrderID = OD.OrderID

JOIN Products P ON OD.ProductID = P.ProductID

WHERE P.Price > 1000

);

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

SELECT ProductName

FROM (

SELECT P.ProductName, SUM(P.Price \* OD.Quantity) AS Revenue

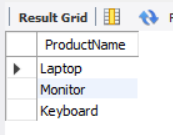
FROM Products P

JOIN OrderDetails OD ON P.ProductID = OD.ProductID

GROUP BY P.ProductID

ORDER BY Revenue DESC

LIMIT 3

) AS TopProducts;

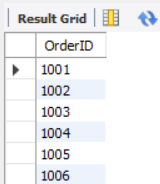
1. Find orders that contain **only one product** using a **correlated subquery**.

SELECT OrderID

FROM OrderDetails OD1

GROUP BY OD1.OrderID

HAVING COUNT(\*) = 1;



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

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

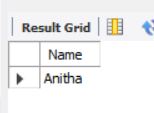
SELECT DISTINCT C2.Name

FROM Customers C1

JOIN Orders O1 ON C1.CustomerID = O1.CustomerID

JOIN Orders O2 ON O1.OrderDate = O2.OrderDate

JOIN Customers C2 ON O2.CustomerID = C2.CustomerID

WHERE C1.Name = 'John' AND C2.Name != 'John';

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

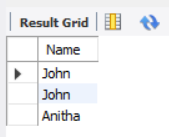
SELECT C.Name

FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

WHERE O.OrderDate = (

SELECT MAX(OrderDate) FROM Orders

);

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

SELECT ProductName

FROM Products

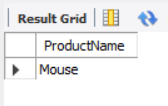
WHERE Price = (

SELECT MIN(Price)

FROM Products

WHERE Price > (

SELECT MIN(Price) FROM Products

);

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

SELECT C.Name

FROM Customers C

JOIN (

SELECT CustomerID, SUM(Amount) AS TotalSpent

FROM Orders

GROUP BY CustomerID

) AS T ON C.CustomerID = T.CustomerID

WHERE T.TotalSpent > 2 \* (

SELECT AVG(Total)

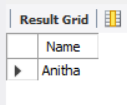
FROM (

SELECT SUM(Amount) AS Total

FROM Orders

GROUP BY CustomerID

) AS AvgSpending

);

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

SELECT C.Name

FROM Customers C

JOIN (

SELECT CustomerID, SUM(Amount) AS TotalAmount

FROM Orders

GROUP BY CustomerID

) AS T ON C.CustomerID = T.CustomerID

WHERE T.TotalAmount > ALL (

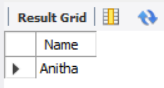
SELECT SUM(O.Amount)

FROM Orders O

JOIN Customers C2 ON O.CustomerID = C2.CustomerID

WHERE C2.City = 'Delhi'

GROUP BY O.CustomerID

);

**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.

SELECT C.Name

FROM Customers C

WHERE (

SELECT COUNT(\*)

FROM Orders O

WHERE O.CustomerID = C.CustomerID) > ( SELECT AVG(OrderCount)

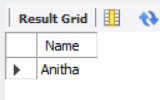
FROM (

SELECT CustomerID, COUNT(\*) AS OrderCount

FROM Orders

GROUP BY CustomerID

) AS OrderCounts

);

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

SELECT P.ProductName

FROM Products P

JOIN OrderDetails OD ON P.ProductID = OD.ProductID

GROUP BY P.ProductID, P.ProductName

HAVING SUM(OD.Quantity) > (

SELECT AVG(TotalQty)

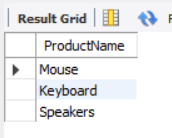
FROM (

SELECT SUM(Quantity) AS TotalQty

FROM OrderDetails

GROUP BY ProductID

) AS ProductQuantities

);

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

SELECT DISTINCT C.Name

FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

JOIN OrderDetails OD ON O.OrderID = OD.OrderID

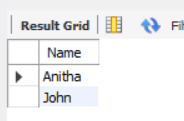
WHERE OD.ProductID IN (

SELECT ProductID

FROM OrderDetails

GROUP BY ProductID

HAVING COUNT(DISTINCT OrderID) = 1

);

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

SELECT O.\*

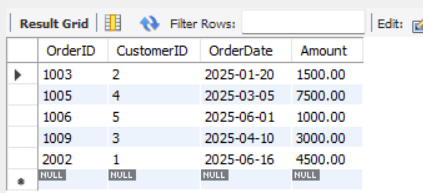
FROM Orders O

WHERE Amount = (

SELECT MAX(Amount)

FROM Orders O2

WHERE O.CustomerID = O2.CustomerID

);

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

SELECT DISTINCT C.Name

FROM Customers C

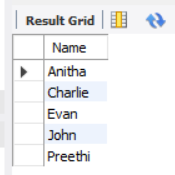
WHERE C.CustomerID NOT IN (

SELECT DISTINCT O.CustomerID

FROM Orders O

JOIN OrderDetails OD ON O.OrderID = OD.OrderID

WHERE OD.Quantity > 5

);

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

1. Use a subquery to list the **top 5 customers by total spending**.

SELECT Name, TotalSpent

FROM (

SELECT C.Name, SUM(O.Amount) AS TotalSpent

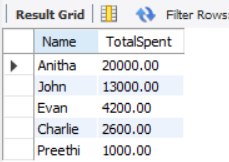
FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

GROUP BY C.CustomerID

ORDER BY TotalSpent DESC

LIMIT 5

) AS TopCustomers;

1. Find all customers who have only ordered **one unique product** using subqueries.

SELECT DISTINCT C.Name

FROM Customers C

WHERE C.CustomerID IN (

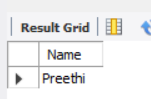
SELECT O.CustomerID

FROM Orders O

JOIN OrderDetails OD ON O.OrderID = OD.OrderID

GROUP BY O.CustomerID

HAVING COUNT(DISTINCT OD.ProductID) = 1

);

1. List all orders where the amount is **not in the top 10 highest order amounts**.

SELECT O.\*

FROM Orders O

LEFT JOIN (

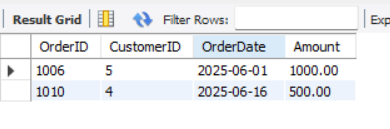
SELECT DISTINCT Amount

FROM Orders

ORDER BY Amount DESC

LIMIT 10

) AS Top10 ON O.Amount = Top10.Amount

WHERE Top10.Amount IS NULL;

1. Retrieve customer names who placed an order in the **last 7 days** but **not** in the **previous 30 days** before that.

SELECT DISTINCT C.Name

FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

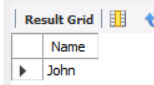
WHERE O.OrderDate BETWEEN '2025-06-09' AND '2025-06-16'

AND C.CustomerID NOT IN (

SELECT CustomerID

FROM Orders

WHERE OrderDate BETWEEN '2025-05-10' AND '2025-06-08'

);

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

SELECT ProductName

FROM Products P

JOIN OrderDetails OD ON P.ProductID = OD.ProductID

GROUP BY P.ProductID, P.ProductName

HAVING COUNT(DISTINCT OD.OrderID) = (

SELECT MAX(OrderCount)

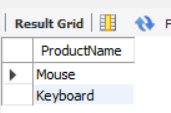
FROM (

SELECT COUNT(DISTINCT OrderID) AS OrderCount

FROM OrderDetails

GROUP BY ProductID

) AS OrderCounts

);