**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.
2. Retrieve the names of products that have been ordered **more than the average quantity** across all products.
3. Find customers who have **never ordered a product** priced above ₹1000.
4. List the **top 3 products by total revenue** using a subquery.
5. 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'**.
2. Find the name of the customer who placed the **most recent order**.
3. Write a query to find the product that has the **second lowest price** using a subquery.
4. Display customer names who have spent **more than double the average spending**.
5. 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.
2. Find all products whose **total sales quantity** is higher than the average total quantity sold per product.
3. Get customers who have ordered at least **one product that no one else has ordered**.
4. Retrieve all orders where the total order amount is equal to the **maximum order amount for that customer**.
5. 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.
3. List all orders where the amount is **not in the top 10 highest order amounts**.
4. Retrieve customer names who placed an order in the **last 7 days** but **not** in the **previous 30 days** before that.
5. Write a query to list all products ordered in the **highest number of distinct orders**.

**ANSWERS**

**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 m

FROM (

SELECT DISTINCT MONTH(OrderDate) AS m

FROM Orders

WHERE YEAR(OrderDate) = YEAR(CURDATE())

) AS months

WHERE NOT EXISTS (

SELECT 1

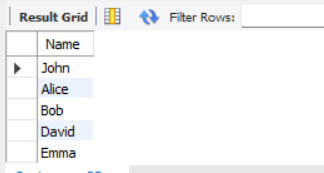
FROM Orders o

WHERE MONTH(o.OrderDate) = months.m

AND YEAR(o.OrderDate) = YEAR(CURDATE())

AND o.CustomerID = c.CustomerID

)

);

**Output :**

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

HAVING SUM(od.Quantity) > (

SELECT AVG(total\_qty)

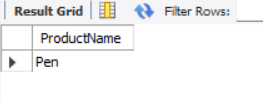
FROM (

SELECT SUM(Quantity) AS total\_qty

FROM OrderDetails

GROUP BY ProductID

) AS avg\_qty

);

**Output :**

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

SELECT C.Name

FROM Customers C

WHERE C.CustomerID NOT IN (

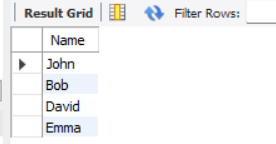
SELECT DISTINCT O.CustomerID

FROM Orders O

JOIN OrderDetails OD ON O.OrderID = OD.OrderID

JOIN Products P ON OD.ProductID = P.ProductID

WHERE P.Price > 1000

);

**Output :**

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

SELECT ProductName

FROM Products

WHERE ProductID IN (

SELECT ProductID

FROM (

SELECT OD.ProductID

FROM OrderDetails OD

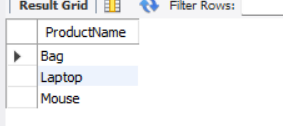
JOIN Products P ON OD.ProductID = P.ProductID

GROUP BY OD.ProductID

ORDER BY SUM(OD.Quantity \* P.Price) DESC

LIMIT 3

) AS TopProducts

);

**Output :**

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

SELECT O.OrderID

FROM Orders O

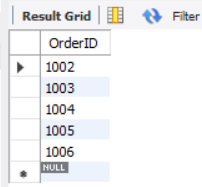
WHERE (

SELECT COUNT(\*)

FROM OrderDetails OD

WHERE OD.OrderID = O.OrderID

) = 1;



**Output :**

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

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

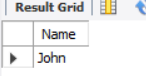
SELECT DISTINCT c2.Name

FROM Orders o1

JOIN Customers c1 ON o1.CustomerID = c1.CustomerID

JOIN Orders o2 ON o1.OrderDate = o2.OrderDate

JOIN Customers c2 ON o2.CustomerID = c2.CustomerID

WHERE c1.Name = 'John';

**Output :**

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

SELECT C.Name

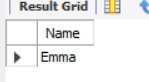
FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

WHERE O.OrderDate = (

SELECT MAX(OrderDate)

FROM Orders

);

**Output :**

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)

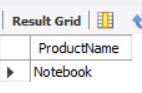
FROM Products

WHERE Price > (

SELECT MIN(Price)

FROM Products

)

);

**Output :**

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

SELECT C.Name

FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

GROUP BY C.CustomerID, C.Name

HAVING SUM(O.Amount) > (

2 \* (

SELECT AVG(TotalSpent)

FROM (

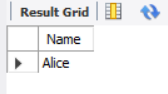
SELECT SUM(Amount) AS TotalSpent

FROM Orders

GROUP BY CustomerID

) AS Sub

)

);

**Output :**

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 Orders O ON C.CustomerID = O.CustomerID

GROUP BY C.CustomerID, C.Name

HAVING SUM(O.Amount) > ALL (

SELECT SUM(O2.Amount)

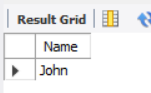
FROM Customers C2

JOIN Orders O2 ON C2.CustomerID = O2.CustomerID

WHERE C2.City = 'Delhi'

GROUP BY C2.CustomerID

);



**Output :**

**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)

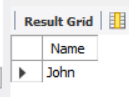
FROM (

SELECT COUNT(\*) AS OrderCount

FROM Orders

GROUP BY CustomerID

) AS Sub

);

**Output :**

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

HAVING SUM(od.Quantity) > (

SELECT AVG(TotalQuantity)

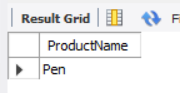
FROM (

SELECT SUM(Quantity) AS TotalQuantity

FROM OrderDetails

GROUP BY ProductID

) AS ProductTotals

);

**Output :**

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

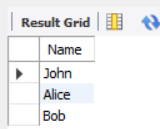
WHERE od.ProductID IN (

SELECT ProductID

FROM OrderDetails

GROUP BY ProductID

HAVING COUNT(DISTINCT OrderID) = 1

);

**Output :**

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

SELECT o.\*

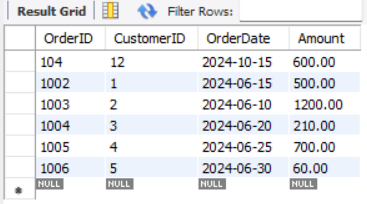
FROM Orders o

WHERE o.Amount = (

SELECT MAX(o2.Amount)

FROM Orders o2

WHERE o2.CustomerID = o.CustomerID

);

**Output :**

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 o.CustomerID

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

WHERE od.Quantity > 5

);

**Output :** 

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

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

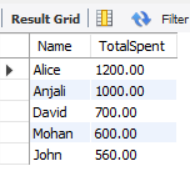
SELECT c.Name, SUM(o.Amount) AS TotalSpent

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID

ORDER BY TotalSpent DESC

LIMIT 5;

**Output :**

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

SELECT c.Name

FROM Customers c

WHERE c.CustomerID IN (

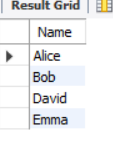
SELECT o.CustomerID

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

GROUP BY o.CustomerID

HAVING COUNT(DISTINCT od.ProductID) = 1

);

**Output :**

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

SELECT \*

FROM Orders

WHERE Amount NOT IN (

SELECT Amount FROM (

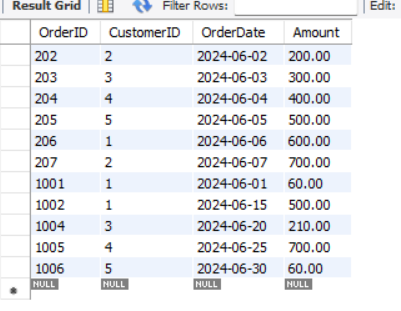
SELECT Amount

FROM Orders

ORDER BY Amount DESC

LIMIT 10

) AS TopAmounts

);

**Output :**

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 >= CURDATE() - INTERVAL 7 DAY

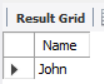
AND c.CustomerID NOT IN (

SELECT o2.CustomerID

FROM Orders o2

WHERE o2.OrderDate >= CURDATE() - INTERVAL 37 DAY

AND o2.OrderDate < CURDATE() - INTERVAL 7 DAY

);

**Output :**

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

SELECT ProductName

FROM Products

WHERE ProductID IN (

SELECT ProductID

FROM OrderDetails

GROUP BY ProductID

HAVING COUNT(DISTINCT OrderID) = (

SELECT MAX(OrderCount)

FROM (

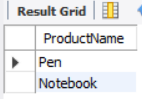
SELECT COUNT(DISTINCT OrderID) AS OrderCount

FROM OrderDetails

GROUP BY ProductID

) AS ProductOrderCounts

)

);

**Output :**