**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 c.CustomerID, c.Name

FROM Customers c

WHERE NOT EXISTS (

SELECT DISTINCT MONTH(CURDATE()) - m.month\_val

FROM (

SELECT 1 AS month\_val UNION SELECT 2 UNION SELECT 3 UNION SELECT 4 UNION SELECT 5 UNION SELECT 6

UNION SELECT 7 UNION SELECT 8 UNION SELECT 9 UNION SELECT 10 UNION SELECT 11 UNION SELECT 12

) m

WHERE NOT EXISTS (

SELECT 1 FROM Orders o

WHERE o.CustomerID = c.CustomerID

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

AND MONTH(o.OrderDate) = m.month\_val

)

);

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(quantity\_sum) FROM (

SELECT SUM(Quantity) AS quantity\_sum

FROM OrderDetails

GROUP BY ProductID

) avg\_table

);

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

SELECT c.CustomerID, c.Name

FROM Customers c

WHERE NOT EXISTS (

SELECT 1

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

WHERE c.CustomerID = o.CustomerID

AND p.Price > 1000

);

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

SELECT ProductID, ProductName, total\_revenue

FROM (

SELECT p.ProductID, p.ProductName, SUM(od.Quantity \* p.Price) AS total\_revenue

FROM Products p

JOIN OrderDetails od ON p.ProductID = od.ProductID

GROUP BY p.ProductID

ORDER BY total\_revenue DESC

LIMIT 3

) AS TopProducts;

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

SELECT o.OrderID

FROM Orders o

WHERE 1 = (

SELECT COUNT(\*)

FROM OrderDetails od

WHERE od.OrderID = o.OrderID

);

**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 c.Name

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

WHERE o.OrderDate IN (

SELECT OrderDate FROM Orders o2

JOIN Customers c2 ON o2.CustomerID = c2.CustomerID

WHERE 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 DISTINCT Price

FROM Products

ORDER BY Price ASC

LIMIT 1 OFFSET 1

);

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

HAVING SUM(o.Amount) > 2 \* (

SELECT AVG(total\_spending)

FROM (

SELECT CustomerID, SUM(Amount) AS total\_spending

FROM Orders

GROUP BY CustomerID

) avg\_spend

);

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

SELECT c.CustomerID, c.Name

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID

HAVING SUM(o.Amount) > ANY (

SELECT SUM(o2.Amount)

FROM Orders o2

JOIN Customers c2 ON o2.CustomerID = c2.CustomerID

WHERE c2.City = 'Delhi'

GROUP BY c2.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(order\_count) FROM (

SELECT CustomerID, COUNT(\*) AS order\_count

FROM Orders

GROUP BY CustomerID

) counts

);

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(total\_qty) FROM (

SELECT SUM(Quantity) AS total\_qty

FROM OrderDetails

GROUP BY ProductID

) q

);

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

SELECT DISTINCT c.CustomerID, 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 o.Amount = (

SELECT MAX(o2.Amount) FROM Orders o2 WHERE o2.CustomerID = o.CustomerID

);

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

SELECT DISTINCT c.CustomerID, 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

);

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

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

SELECT c.CustomerID, 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;

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

SELECT c.CustomerID, c.Name

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderDetails od ON o.OrderID = od.OrderID

GROUP BY c.CustomerID

HAVING COUNT(DISTINCT od.ProductID) = 1;

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 Orders

ORDER BY Amount DESC

LIMIT 10

);

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 CURDATE() - INTERVAL 7 DAY AND CURDATE()

AND c.CustomerID NOT IN (

SELECT CustomerID FROM Orders

WHERE OrderDate BETWEEN CURDATE() - INTERVAL 37 DAY AND CURDATE() - INTERVAL 8 DAY

);

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

SELECT ProductID, ProductName

FROM Products

WHERE ProductID IN (

SELECT ProductID

FROM OrderDetails

GROUP BY ProductID

HAVING COUNT(DISTINCT OrderID) = (

SELECT MAX(order\_count) FROM (

SELECT COUNT(DISTINCT OrderID) AS order\_count

FROM OrderDetails

GROUP BY ProductID

) x

)

);