**DATE: 16.06.2025**

**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)

);

-- Customers Table

INSERT INTO Customers (CustomerID, Name, City) VALUES

(1, 'John', 'Delhi'),

(2, 'Alice', 'Mumbai'),

(3, 'Bob', 'Delhi'),

(4, 'Carol', 'Chennai'),

(5, 'David', 'Bangalore'),

(6, 'Eva', 'Kolkata');

-- Products Table

INSERT INTO Products (ProductID, ProductName, Price) VALUES

(101, 'Laptop', 1200.00),

(102, 'Mouse', 300.00),

(103, 'Keyboard', 700.00),

(104, 'Monitor', 1000.00),

(105, 'Webcam', 500.00),

(106, 'Smartphone', 1500.00);

--Orders Table

INSERT INTO Orders (OrderID, CustomerID, OrderDate, Amount) VALUES

(201, 1, '2025-06-18', 1500.00),

(202, 2, '2025-06-18', 700.00),

(203, 3, '2025-06-01', 500.00),

(204, 4, '2025-05-15', 2000.00),

(205, 5, '2025-06-05', 1000.00),

(206, 6, '2025-04-22', 900.00),

(207, 2, '2025-05-18', 1300.00),

(208, 1, '2025-02-10', 1800.00),

(209, 1, '2025-03-25', 750.00),

(210, 1, '2025-01-09', 600.00),

(211, 2, '2025-04-05', 1200.00),

(212, 4, '2025-06-15', 1100.00);

--OrderDetails Table

INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity) VALUES

(301, 201, 106, 1),

(302, 202, 103, 1),

(303, 203, 102, 2),

(304, 204, 101, 1),

(305, 205, 104, 1),

(306, 206, 105, 2),

(307, 207, 106, 1),

(308, 208, 101, 1),

(309, 209, 105, 1),

(310, 210, 102, 2),

(311, 211, 104, 1),

(312, 212, 103, 3),

(313, 203, 105, 1),

(314, 211, 102, 3),

(315, 208, 104, 1);

**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.Name

FROM Customers c

WHERE NOT EXISTS (

SELECT DISTINCT MONTH(DATE\_ADD(CURDATE(), INTERVAL -1 YEAR + INTERVAL 1 MONTH MONTH)) as m

FROM dual

WHERE NOT EXISTS (

SELECT 1

FROM Orders o

WHERE o.CustomerID = c.CustomerID

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

AND MONTH(o.OrderDate) = m

)

);

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

) AS avg\_qty

);

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

SELECT Name

FROM Customers

WHERE 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 Products

WHERE ProductID IN (

SELECT ProductID

FROM (

SELECT od.ProductID, SUM(od.Quantity \* p.Price) AS revenue

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY od.ProductID

ORDER BY revenue DESC

LIMIT 3

) AS top\_products

);

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 o2.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 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 Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID

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

SELECT AVG(total\_amount)

FROM (

SELECT SUM(Amount) AS total\_amount

FROM Orders

GROUP BY CustomerID

) AS avg\_spend

);

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

SELECT Name

FROM Customers

WHERE CustomerID IN (

SELECT o.CustomerID

FROM Orders o

GROUP BY o.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 o2.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 COUNT(\*) AS order\_count

FROM Orders

GROUP BY CustomerID

) AS avg\_orders

);

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(qty)

FROM (

SELECT SUM(Quantity) AS qty

FROM OrderDetails

GROUP BY ProductID

) AS avg\_qty

);

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 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.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.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.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 >= CURDATE() - INTERVAL 7 DAY

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 p.ProductName

FROM Products p

JOIN OrderDetails od ON p.ProductID = od.ProductID

GROUP BY p.ProductID

HAVING COUNT(DISTINCT od.OrderID) = (

SELECT MAX(order\_count)

FROM (

SELECT COUNT(DISTINCT OrderID) AS order\_count

FROM OrderDetails

GROUP BY ProductID

) AS counts

);