**ANSWERS (16-06-2025)**

**-BY GETSY JACINTH**

**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)

);

**SAMPLE DATA :**

-- Insert Customers

INSERT INTO Customers (CustomerID, Name, City) VALUES

(1, 'Alice', 'Mumbai'),

(2, 'Bob', 'Delhi'),

(3, 'Charlie', 'Chennai');

-- Insert Products

INSERT INTO Products (ProductID, ProductName, Price) VALUES

(1, 'Laptop', 1500),

(2, 'Mouse', 300),

(3, 'Keyboard', 800),

(4, 'Monitor', 1200),

(5, 'Pen Drive', 500);

-- Insert Orders (Customer 1 has orders in every month of 2025)

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

(101, 1, '2025-01-05', 200),

(102, 1, '2025-02-05', 400),

(103, 1, '2025-03-05', 300),

(104, 1, '2025-04-05', 250),

(105, 1, '2025-05-05', 150),

(106, 1, '2025-06-05', 500),

(107, 1, '2025-07-05', 300),

(108, 1, '2025-08-05', 100),

(109, 1, '2025-09-05', 150),

(110, 1, '2025-10-05', 200),

(111, 1, '2025-11-05', 100),

(112, 1, '2025-12-05', 120),

-- Bob: orders with expensive product

(201, 2, '2025-06-06', 2000),

-- Charlie: only affordable products

(301, 3, '2025-04-04', 800);

-- Insert OrderDetails

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

(1, 101, 1, 1), -- Laptop

(2, 102, 2, 3), -- Mouse

(3, 103, 3, 2), -- Keyboard

(4, 104, 4, 1), -- Monitor

(5, 105, 5, 2), -- Pen Drive

(6, 106, 1, 1),

(7, 107, 2, 4),

(8, 108, 3, 2),

(9, 109, 5, 2),

(10, 110, 4, 1),

(11, 111, 2, 1),

(12, 112, 3, 1),

-- Bob ordered 1 laptop

(13, 201, 1, 1),

-- Charlie ordered mouse + keyboard

(14, 301, 2, 1),

(15, 301, 3, 1);

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

(501, 1, '2025-01-10', 50.00),

(502, 2, '2025-02-10', 30.00),

(503, 3, '2025-03-10', 25.00);

**Part A – Subqueries (20 marks)**

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

**ANSWER:**

SELECT c.Name

FROM Customers c

WHERE NOT EXISTS (

SELECT DISTINCT MONTH(m)

FROM (

SELECT DATE\_FORMAT(CURDATE(), '%Y-01-01') + INTERVAL seq MONTH AS m

FROM (

SELECT 0 AS seq UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL

SELECT 3 UNION ALL SELECT 4 UNION ALL SELECT 5 UNION ALL

SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL

SELECT 9 UNION ALL SELECT 10 UNION ALL SELECT 11

) AS months

) AS all\_months

WHERE NOT EXISTS (

SELECT 1

FROM Orders o

WHERE o.CustomerID = c.CustomerID

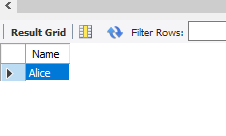
AND YEAR(o.OrderDate) = YEAR(CURDATE())

AND MONTH(o.OrderDate) = MONTH(m)

)

);

**OUTPUT:**



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

**ANSWER:**

SELECT p.ProductName

FROM Products p

JOIN OrderDetails od ON p.ProductID = od.ProductID

GROUP BY p.ProductID

HAVING SUM(od.Quantity) > (

SELECT AVG(TotalQty)

FROM (

SELECT SUM(Quantity) AS TotalQty

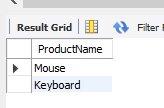
FROM OrderDetails

GROUP BY ProductID

) AS avg\_table

);

**OUTPUT:**

****

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

**ANSWER:**

SELECT DISTINCT c.Name

FROM Customers c

WHERE CustomerID NOT IN (

SELECT DISTINCT o.CustomerID

FROM Orders o

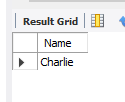
JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

WHERE p.Price > 1000

);

**OUTPUT:**

****

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

**ANSWER:**

SELECT ProductName

FROM Products

WHERE ProductID IN (

SELECT ProductID

FROM (

SELECT ProductID

FROM OrderDetails

GROUP BY ProductID

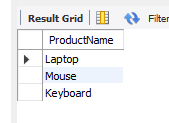
ORDER BY SUM(Quantity \* (SELECT Price FROM Products WHERE ProductID = OrderDetails.ProductID)) DESC

LIMIT 3

) AS top3

);

**OUTPUT:**



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

**ANSWER:**

SELECT o.OrderID

FROM Orders o

WHERE (

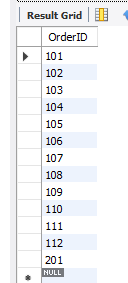
SELECT COUNT(\*)

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'**.

**ANSWER:**

SELECT DISTINCT c2.Name

FROM Customers c1

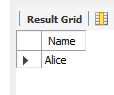
JOIN Orders o1 ON c1.CustomerID = o1.CustomerID

JOIN Orders o2 ON o1.OrderDate = o2.OrderDate

JOIN Customers c2 ON o2.CustomerID = c2.CustomerID

WHERE c1.Name = 'John' AND c1.CustomerID != c2.CustomerID;

**OUTPUT:**

****

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

**ANSWER:**

SELECT c.Name

FROM Customers c

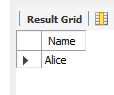
JOIN Orders o ON c.CustomerID = o.CustomerID

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.

**ANSWER:**

SELECT ProductName

FROM Products

WHERE Price = (

SELECT DISTINCT Price

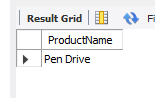
FROM Products

ORDER BY Price ASC

LIMIT 1 OFFSET 1

);

**OUTPUT:**

****

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

**ANSWER:**

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\_spent)

FROM (

SELECT SUM(Amount) AS total\_spent

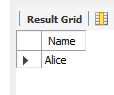
FROM Orders

GROUP BY CustomerID

) AS avg\_spending

);

**OUTPUT:**

****

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

**ANSWER:**

SELECT 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 Customers c2

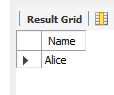
JOIN Orders o2 ON c2.CustomerID = o2.CustomerID

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.

**ANSWER:**

SELECT c.Name

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID

HAVING COUNT(o.OrderID) > (

SELECT AVG(order\_count)

FROM (

SELECT COUNT(\*) AS order\_count

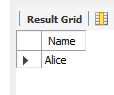
FROM Orders

GROUP BY CustomerID

) AS avg\_orders

);

**OUTPUT:**

****

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

**ANSWER:**

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

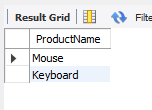
FROM OrderDetails

GROUP BY ProductID

) AS avg\_qty

);

**OUTPUT:**

****

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

**ANSWER:**

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

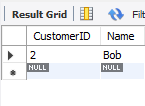
FROM OrderDetails

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**.

**ANSWER:**

SELECT o.OrderID, o.Amount

FROM Orders o

WHERE o.Amount = (

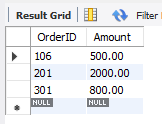
SELECT MAX(o2.Amount)

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**.

**ANSWER:**

SELECT DISTINCT c.Name

FROM Customers c

WHERE c.CustomerID NOT IN (

SELECT DISTINCT o.CustomerID

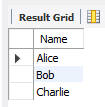
FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

WHERE od.Quantity > 5

);

**OUTPUT:**



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

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

**ANSWER:**

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

FROM Customers c

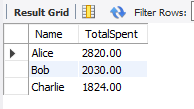
JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID

ORDER BY TotalSpent DESC

LIMIT 5;

**OUTPUT:**

****

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

**ANSWER:**

SELECT DISTINCT c.Name

FROM Customers c

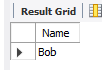
JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderDetails od ON o.OrderID = od.OrderID

GROUP BY c.CustomerID

HAVING COUNT(DISTINCT od.ProductID) = 1;

**OUTPUT:**



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

**ANSWER:**

SELECT o.\*

FROM Orders o

LEFT JOIN (

SELECT DISTINCT Amount

FROM Orders

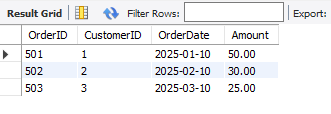
ORDER BY Amount DESC

LIMIT 10

) AS top10 ON o.Amount = top10.Amount

WHERE top10.Amount IS NULL;

**OUTPUT:**



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

**ANSWER:**

-- Add a new recent order (assuming today is 2025-06-16)

INSERT INTO Orders (OrderID, CustomerID, OrderDate, Amount) VALUES (401, 3, CURDATE(), 999);

INSERT INTO OrderDetails VALUES (16, 401, 3, 1);

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 (

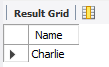
SELECT DISTINCT o2.CustomerID

FROM Orders o2

WHERE o2.OrderDate BETWEEN CURDATE() - INTERVAL 37 DAY AND CURDATE() - INTERVAL 8 DAY

);

**OUTPUT:**

****

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

**ANSWER:**

SELECT 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

) AS sub

)

);

