**16/06/2025 Aarthi P**

**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)

);

INSERT INTO Customers (CustomerID, Name, City) VALUES

(1, 'John', 'Mumbai'),

(2, 'Aarthi', 'Chennai'),

(3, 'Rahul', 'Delhi'),

(4, 'Sneha', 'Bangalore'),

(5, 'Vikram', 'Kolkata');

-- Orders Table

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

Amount DECIMAL(10,2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

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

(101, 1, '2025-01-15', 1500.00),

(102, 2, '2025-02-10', 2500.00),

(103, 3, '2025-03-05', 300.00),

(104, 4, '2025-04-12', 1800.00),

(105, 5, '2025-05-20', 2200.00);

-- Products Table

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Price DECIMAL(10,2)

);

INSERT INTO Products (ProductID, ProductName, Price) VALUES

(201, 'Laptop', 55000.00),

(202, 'Smartphone', 15000.00),

(203, 'Headphones', 1200.00),

(204, 'Mouse', 500.00),

(205, 'Keyboard', 800.00);

-- 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)

);

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

(301, 101, 202, 2),

(302, 102, 201, 1),

(303, 103, 204, 3),

(304, 104, 203, 2),

(305, 105, 205, 4); (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

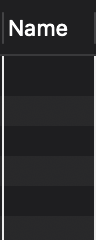
where CustomerID in (

select CustomerID from Orders

where year(OrderDate) = year(curdate())

group by CustomerID

having count(distinct month(OrderDate)) = 12);



*the table tured out empty as there is no customer who placed orders in every month.*

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

select ProductName from Products

where ProductID in(

select ProductID from OrderDetails

group by ProductID

having sum(Quantity)>(

select Avg(TotalQuantity) from (

select sum(Quantity) as TotalQuantity

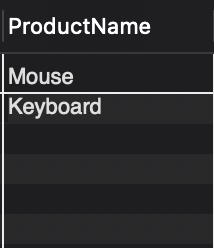
from OrderDetails

group by ProductID

) as ProductSums

)

);



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

select name from Customers

where CustomerID 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 OrderDetails

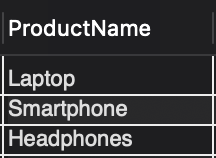
group by ProductID

order by sum(Quantity \*(

select price from Products

where Products.ProductID = OrderDetails.ProductID)) desc

Limit 3 );



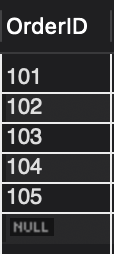
1. Find orders that contain **only one product** using a **correlated subquery**.

select 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

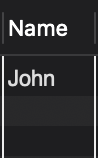
join Orders o on c.CustomerID = o.CustomerID

where o.OrderDate in (

select OrderDate from Customers c2

join Orders o2 on c2.CustomerID = o2.CustomerID

where c2.Name = 'John');



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

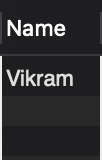
select Name from Customers

where CustomerID = (

select CustomerID from Orders

order by OrderDate Desc

limit 1);



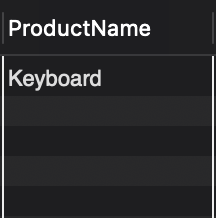
1. Write a query to find the product that has the **second lowest price** using a subquery.

select ProductName from Products

where Price = ( select distinct Price from Products

order by Price

limit 1 offset 1);



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

select Name from Customers c

join Orders o on c.CustomerID = o.CustomerID

group by c.CustomerID

having sum(o.Amount) > 2 \* (

select avg(TotalAmount) from (

select sum(o.Amount) as TotalAmount from Orders

group by CustomerID

) as AvgSpending

);



*the table tured out empty as there is no customer who has spent* ***more than double the average spending***

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

select Name from Customers c

join Orders o on c.CustomerID = o.CustomerID

group by c.CustomerID

having sum(Amount) > (

select max(DelhiTotal)

from (

select sum(Amount) as DelhiTotal from Orders o2

join Customers c2 on o2.CustomerID = c2.CustomerID

where c2.City = 'Delhi'

group by o2.CustomerID

) as DelhiTotals );



**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 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 AvgOrders

);



*the table tured out empty as there is no customer who has placed* ***more orders than the average*** *number of orders placed by all customers.*

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

select ProductName from Products p

join OrderDetails od on p.ProductID = od.ProductID

group by p.ProductID

having sum(Quantity) > (

select avg(TotalQty)

from (

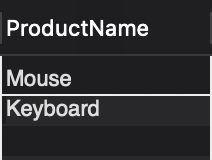
select sum(Quantity) as TotalQty

from OrderDetails

group by ProductID

) as AvgQty

);



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

select distinct 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**.

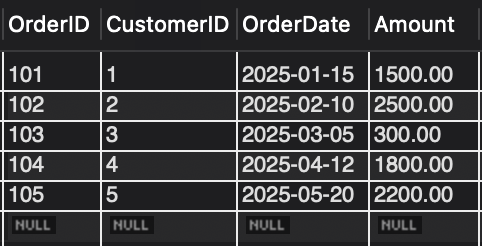
select o.\* from Orders o

where Amount = (

select max(Amount)

from Orders

where CustomerID = o.CustomerID);



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

select Name from Customers c

where not exists (

select \* from Orders o

join OrderDetails od on o.OrderID = od.OrderID

where o.CustomerID = c.CustomerID and 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 from Customers c

join Orders o on c.CustomerID = o.CustomerID

group by c.CustomerID

order by sum(o.Amount) desc

limit 5;



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

select Name from Customers c

where (

select count(distinct od.ProductID)

from Orders o

join OrderDetails od on o.OrderID = od.OrderID

where o.CustomerID = c.CustomerID

) = 1;



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

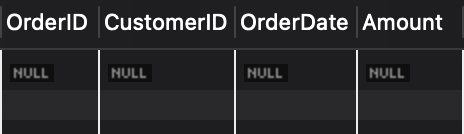
select \* from Orders

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 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

);



*the table tured out empty as there is no customer who has placed an order in the* ***last 7 days*** *but* ***not*** *in the* ***previous 30 days*** *before that.*

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(cnt)

from (

select count(distinct OrderID) as cnt from OrderDetails

group by ProductID

) as Sub

)

);

