16/6/25 MYSQL MCQ QUIZ

1. Q1. What is a key characteristic of SQL vs NoSQL?

- A. SQL vs NoSQL ensures data duplication
- B. SQL vs NoSQL is used only in NoSQL databases
- **C. SQL VS NOSQL IMPROVES DATA INTEGRITY**
- D. SQL vs NoSQL is not related to database design

2. Q2. What is a key characteristic of Advantages of SQL?

- A. Advantages of SQL ensures data duplication
- B. Advantages of SQL is used only in NoSQL databases
- **C. ADVANTAGES OF SQL IMPROVES DATA INTEGRITY**
- D. Advantages of SQL is not related to database design

3. Q3. What is a key characteristic of Disadvantages of SQL?

- A. Disadvantages of SQL ensures data duplication
- B. Disadvantages of SQL is used only in NoSQL databases
- **C. DISADVANTAGES OF SQL IMPROVES DATA INTEGRITY**
- D. Disadvantages of SQL is not related to database design

4. Q4. What is a key characteristic of System Databases in SQL Server?

- A. System Databases in SQL Server ensures data duplication
- B. System Databases in SQL Server is used only in NoSQL databases
- **C. SYSTEM DATABASES IN SQL SERVER IMPROVES DATA INTEGRITY**
- D. System Databases in SQL Server is not related to database design

5. Q5. What is a key characteristic of Managing Databases?

- A. Managing Databases ensures data duplication
- B. Managing Databases is used only in NoSQL databases
- **C. MANAGING DATABASES IMPROVES DATA INTEGRITY**
- D. Managing Databases is not related to database design

6. Q6. What is a key characteristic of 1NF?

- A. 1NF ensures data duplication
- B. 1NF is used only in NoSQL databases
- **C. 1NF IMPROVES DATA INTEGRITY**
- D. 1NF is not related to database design

7. Q7. What is a key characteristic of 2NF?

- A. 2NF ensures data duplication
- B. 2NF is used only in NoSQL databases
- **C. 2NF IMPROVES DATA INTEGRITY**
- D. 2NF is not related to database design

8. Q8. What is a key characteristic of 3NF?

- A. 3NF ensures data duplication
- B. 3NF is used only in NoSQL databases
- **C. 3NF IMPROVES DATA INTEGRITY**
- D. 3NF is not related to database design

9. Q9. What is a key characteristic of BCNF?

- A. BCNF ensures data duplication
- B. BCNF is used only in NoSQL databases
- **C. BCNF IMPROVES DATA INTEGRITY**
- D. BCNF is not related to database design

10. Q10. What is a key characteristic of Identifying System Databases?

- A. Identifying System Databases ensures data duplication
- B. Identifying System Databases is used only in NoSQL databases
- **C. IDENTIFYING SYSTEM DATABASES IMPROVES DATA INTEGRITY**
- D. Identifying System Databases is not related to database design

11. Q11. What is a key characteristic of Database Files?

- A. Database Files ensures data duplication
- B. Database Files is used only in NoSQL databases
- **C. DATABASE FILES IMPROVES DATA INTEGRITY**
- D. Database Files is not related to database design

12. Q12. What is a key characteristic of Creating Databases?

- A. Creating Databases ensures data duplication
- B. Creating Databases is used only in NoSQL databases
- **C. CREATING DATABASES IMPROVES DATA INTEGRITY**
- D. Creating Databases is not related to database design

13. Q13. What is a key characteristic of Renaming Databases?

- A. Renaming Databases ensures data duplication
- B. Renaming Databases is used only in NoSQL databases

- **C. RENAMING DATABASES IMPROVES DATA INTEGRITY**
- D. Renaming Databases is not related to database design

14. Q14. What is a key characteristic of Dropping Databases?

- A. Dropping Databases ensures data duplication
- B. Dropping Databases is used only in NoSQL databases
- **C. DROPPING DATABASES IMPROVES DATA INTEGRITY**
- D. Dropping Databases is not related to database design

15. Q15. What is a key characteristic of Data Types?

- A. Data Types ensures data duplication
- B. Data Types is used only in NoSQL databases
- **C. DATA TYPES IMPROVES DATA INTEGRITY**
- D. Data Types is not related to database design

16. Q16. What is a key characteristic of Creating Tables?

- A. Creating Tables ensures data duplication
- B. Creating Tables is used only in NoSQL databases
- **C. CREATING TABLES IMPROVES DATA INTEGRITY**
- D. Creating Tables is not related to database design

17. Q17. What is a key characteristic of Modifying Tables?

- A. Modifying Tables ensures data duplication
- B. Modifying Tables is used only in NoSQL databases
- **C. MODIFYING TABLES IMPROVES DATA INTEGRITY**
- D. Modifying Tables is not related to database design

18. Q18. What is a key characteristic of Renaming Tables?

- A. Renaming Tables ensures data duplication
- B. Renaming Tables is used only in NoSQL databases
- **C. RENAMING TABLES IMPROVES DATA INTEGRITY**
- D. Renaming Tables is not related to database design

19. Q19. What is a key characteristic of Dropping Tables?

- A. Dropping Tables ensures data duplication
- B. Dropping Tables is used only in NoSQL databases
- **C. DROPPING TABLES IMPROVES DATA INTEGRITY**
- D. Dropping Tables is not related to database design

20. Q20. What is a key characteristic of Insert/Update/Delete?

- A. Insert/Update/Delete ensures data duplication
- B. Insert/Update/Delete is used only in NoSQL databases
- **C. INSERT/UPDATE/DELETE IMPROVES DATA INTEGRITY**
- D. Insert/Update/Delete is not related to database design

21. Q21. What is a key characteristic of Retrieving Data?

- A. Retrieving Data ensures data duplication
- B. Retrieving Data is used only in NoSQL databases
- **C. RETRIEVING DATA IMPROVES DATA INTEGRITY**
- D. Retrieving Data is not related to database design

22. Q22. What is a key characteristic of Filtering: WHERE, IN, AND, OR, LIKE?

- A. Filtering: WHERE, IN, AND, OR, LIKE ensures data duplication
- B. Filtering: WHERE, IN, AND, OR, LIKE is used only in NoSQL databases
- **C. FILTERING: WHERE, IN, AND, OR, LIKE IMPROVES DATA INTEGRITY**
- D. Filtering: WHERE, IN, AND, OR, LIKE is not related to database design

23. Q23. What is a key characteristic of Aliases?

- A. Aliases ensures data duplication
- B. Aliases is used only in NoSQL databases
- **C. ALIASES IMPROVES DATA INTEGRITY**
- D. Aliases is not related to database design

24. Q24. What is a key characteristic of DISTINCT?

- A. DISTINCT ensures data duplication
- B. DISTINCT is used only in NoSQL databases
- **C. DISTINCT improves data integrity**
- D. DISTINCT is not related to database design

25. Q25. What is a key characteristic of BETWEEN?

- A. BETWEEN ensures data duplication
- B. BETWEEN is used only in NoSQL databases
- **C. BETWEEN IMPROVES DATA INTEGRITY**
- D. BETWEEN is not related to database design

26. Q26. What is a key characteristic of Data Integrity?

- A. Data Integrity ensures data duplication
- B. Data Integrity is used only in NoSQL databases
- **C. DATA INTEGRITY IMPROVES DATA INTEGRITY**
- D. Data Integrity is not related to database design

27. Q27. What is a key characteristic of String Functions?

- A. String Functions ensures data duplication
- B. String Functions is used only in NoSQL databases
- **C. STRING FUNCTIONS IMPROVES DATA INTEGRITY**
- D. String Functions is not related to database design

28. Q28. What is a key characteristic of Date Functions?

- A. Date Functions ensures data duplication
- B. Date Functions is used only in NoSQL databases
- **C. DATE FUNCTIONS IMPROVES DATA INTEGRITY**
- D. Date Functions is not related to database design

29. Q29. What is a key characteristic of Math Functions?

- A. Math Functions ensures data duplication
- B. Math Functions is used only in NoSQL databases
- **C. MATH FUNCTIONS IMPROVES DATA INTEGRITY**
- D. Math Functions is not related to database design

30. Q30. What is a key characteristic of System Functions?

- A. System Functions ensures data duplication
- B. System Functions is used only in NoSQL databases
- **C. SYSTEM FUNCTIONS IMPROVES DATA INTEGRITY**
- D. System Functions is not related to database design

31. Q31. What is a key characteristic of Aggregate Functions?

- A. Aggregate Functions ensures data duplication
- B. Aggregate Functions is used only in NoSQL databases
- **C. AGGREGATE FUNCTIONS IMPROVES DATA INTEGRITY**
- D. Aggregate Functions is not related to database design

32. Q32. What is a key characteristic of GROUP BY?

- A. GROUP BY ensures data duplication
- B. GROUP BY is used only in NoSQL databases
- **C. GROUP BY IMPROVES DATA INTEGRITY**
- D. GROUP BY is not related to database design

33. Q33. What is a key characteristic of Customizing Result Sets?

- A. Customizing Result Sets ensures data duplication
- B. Customizing Result Sets is used only in NoSQL databases

- **C. CUSTOMIZING RESULT SETS IMPROVES DATA INTEGRITY**
- D. Customizing Result Sets is not related to database design

34. Q34. What is a key characteristic of Inner Join?

- A. Inner Join ensures data duplication
- B. Inner Join is used only in NoSQL databases
- **C. INNER JOIN IMPROVES DATA INTEGRITY**
- D. Inner Join is not related to database design

35. Q35. What is a key characteristic of Left Join?

- A. Left Join ensures data duplication
- B. Left Join is used only in NoSQL databases
- **C. LEFT JOIN IMPROVES DATA INTEGRITY**
- D. Left Join is not related to database design

36. Q36. What is a key characteristic of Right Join?

- A. Right Join ensures data duplication
- B. Right Join is used only in NoSQL databases
- **C. RIGHT JOIN IMPROVES DATA INTEGRITY**
- D. Right Join is not related to database design

37. Q37. What is a key characteristic of Full Outer Join?

- A. Full Outer Join ensures data duplication
- B. Full Outer Join is used only in NoSQL databases
- **C. FULL OUTER JOIN IMPROVES DATA INTEGRITY**
- D. Full Outer Join is not related to database design

38. Q38. What is a key characteristic of Cross Join?

- A. Cross Join ensures data duplication
- B. Cross Join is used only in NoSQL databases
- **C. CROSS JOIN IMPROVES DATA INTEGRITY**
- D. Cross Join is not related to database design

39. Q39. What is a key characteristic of GROUP BY with Joins?

- A. GROUP BY with Joins ensures data duplication
- B. GROUP BY with Joins is used only in NoSQL databases
- **C. GROUP BY WITH JOINS IMPROVES DATA INTEGRITY**
- D. GROUP BY with Joins is not related to database design

40. Q40. What is a key characteristic of Aggregate Functions with Joins?

- A. Aggregate Functions with Joins ensures data duplication
- B. Aggregate Functions with Joins is used only in NoSQL databases
- **C. AGGREGATE FUNCTIONS WITH JOINS IMPROVES DATA INTEGRITY**
- D. Aggregate Functions with Joins is not related to database design

41. Q41. What is a key characteristic of Equi Join?

- A. Equi Join ensures data duplication
- B. Equi Join is used only in NoSQL databases
- **C. EQUI JOIN IMPROVES DATA INTEGRITY**
- D. Equi Join is not related to database design

42. Q42. What is a key characteristic of Self Join?

- A. Self Join ensures data duplication
- B. Self Join is used only in NoSQL databases
- **C. SELF JOIN IMPROVES DATA INTEGRITY**
- D. Self Join is not related to database design

43. Q43. What is a key characteristic of HAVING, GROUPING SETS?

- A. HAVING, GROUPING SETS ensures data duplication
- B. HAVING, GROUPING SETS is used only in NoSQL databases
- **C. HAVING, GROUPING SETS IMPROVES DATA INTEGRITY**
- D. HAVING, GROUPING SETS is not related to database design

44. Q44. What is a key characteristic of Subqueries?

- A. Subqueries ensures data duplication
- B. Subqueries is used only in NoSQL databases
- **C. SUBQUERIES IMPROVES DATA INTEGRITY**
- D. Subqueries is not related to database design

45. Q45. What is a key characteristic of EXISTS, ANY, ALL?

- A. EXISTS, ANY, ALL ensures data duplication
- B. EXISTS, ANY, ALL is used only in NoSQL databases
- **C. EXISTS, ANY, ALL IMPROVES DATA INTEGRITY**
- D. EXISTS, ANY, ALL is not related to database design

46. Q46. What is a key characteristic of Nested Subqueries?

- A. Nested Subqueries ensures data duplication
- B. Nested Subqueries is used only in NoSQL databases
- **C. NESTED SUBQUERIES IMPROVES DATA INTEGRITY**
- D. Nested Subqueries is not related to database design

47. Q47. What is a key characteristic of Correlated Subqueries?

- A. Correlated Subqueries ensures data duplication
- B. Correlated Subqueries is used only in NoSQL databases
- **C. CORRELATED SUBQUERIES IMPROVES DATA INTEGRITY**
- D. Correlated Subqueries is not related to database design

48. Q48. What is a key characteristic of UNION, INTERSECT, EXCEPT, MERGE?

- A. UNION, INTERSECT, EXCEPT, MERGE ensures data duplication
- B. UNION, INTERSECT, EXCEPT, MERGE is used only in NoSQL databases
- **C. UNION, INTERSECT, EXCEPT, MERGE IMPROVES DATA INTEGRITY**
- D. UNION, INTERSECT, EXCEPT, MERGE is not related to database design

16/6/25 Practise Question

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

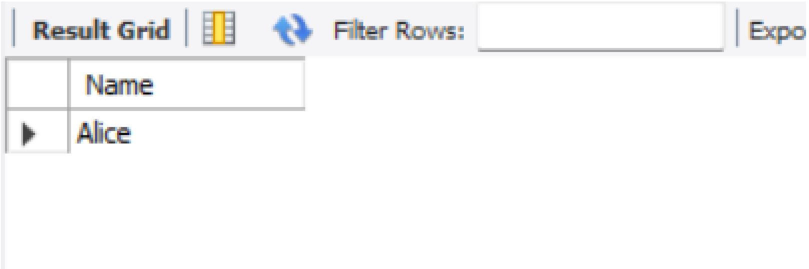
```
-- Customers Table
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(100),
    City VARCHAR(100)
);

-- 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.

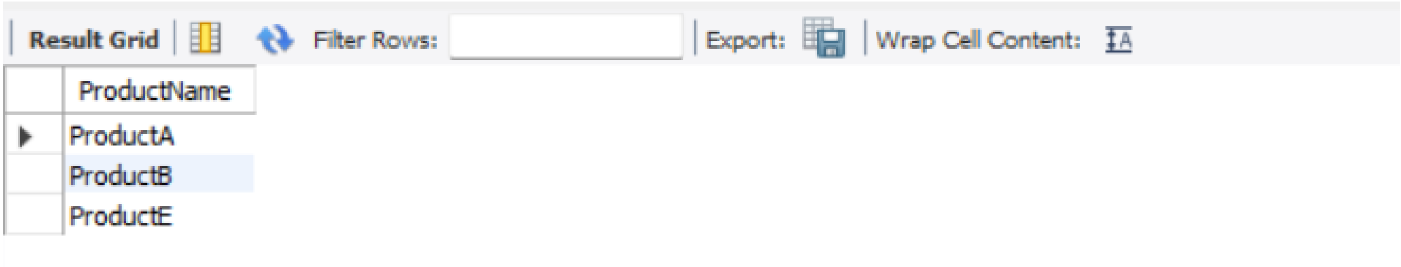
```
SELECT Name
FROM Customers
WHERE NOT EXISTS (
SELECT 1
FROM (
SELECT DISTINCT MONTH(OrderDate) AS M
FROM Orders
WHERE YEAR(OrderDate) = YEAR(CURDATE())
) AS months
WHERE NOT EXISTS (
SELECT 1 FROM Orders
WHERE Orders.CustomerID = Customers.CustomerID
AND YEAR(OrderDate)=YEAR(CURDATE())
AND MONTH(OrderDate)=months.M ));
```



Name
Alice

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

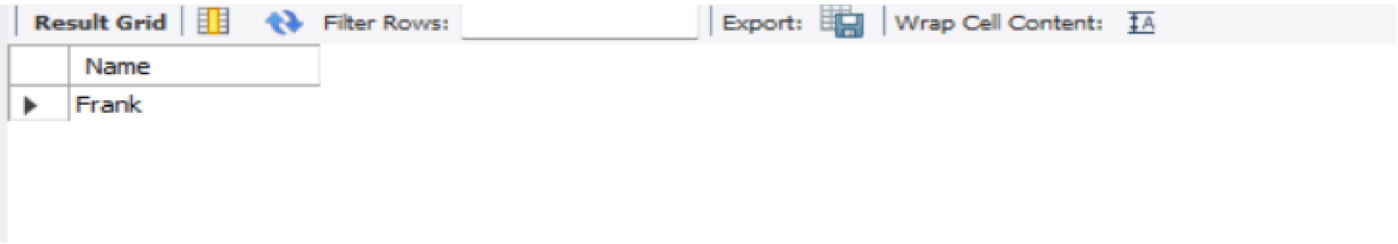
```
SELECT ProductName
FROM Products
WHERE (SELECT AVG(Quantity) FROM OrderDetails) < (
    SELECT SUM(Quantity)
    FROM OrderDetails
    WHERE ProductID = Products.ProductID);
```



ProductName
ProductA
ProductB
ProductE

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

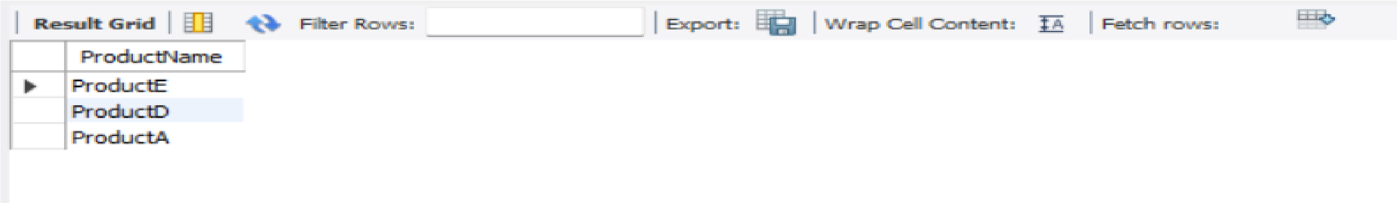
```
SELECT Name
FROM Customers
WHERE NOT EXISTS (
    SELECT 1 FROM Orders
    JOIN OrderDetails USING (OrderID)
    JOIN Products USING (ProductID)
    WHERE Customers.CustomerID = Orders.CustomerID
```



Name
Frank

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

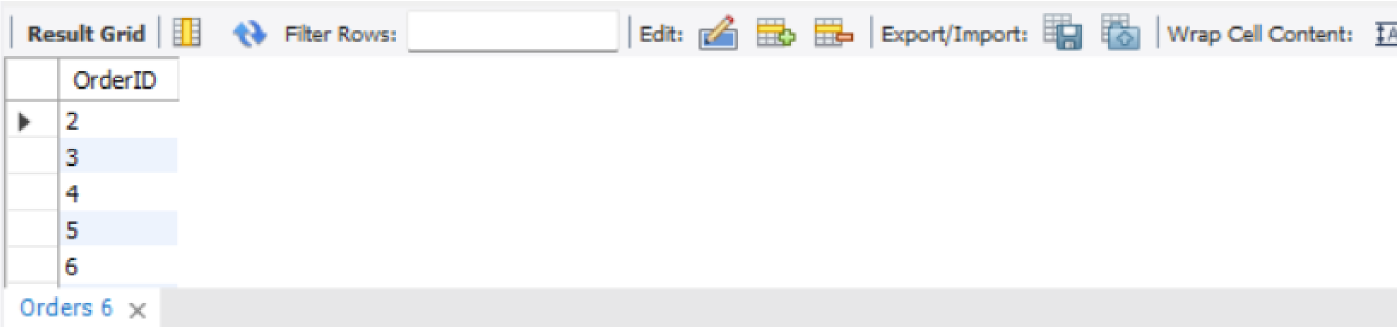
```
SELECT Name
FROM Customers
WHERE NOT EXISTS (
    SELECT 1 FROM Orders
    JOIN OrderDetails USING (OrderID)
JOIN Products USING (ProductID)
WHERE Customers.CustomerID = Orders.CustomerID
 AND Products.Price > 1000 );
```



ProductName
ProductE
ProductD
ProductA

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

```
SELECT OrderID
FROM Orders o
WHERE (SELECT COUNT(*) FROM OrderDetails WHERE OrderID = o.OrderID)
= 1;
```

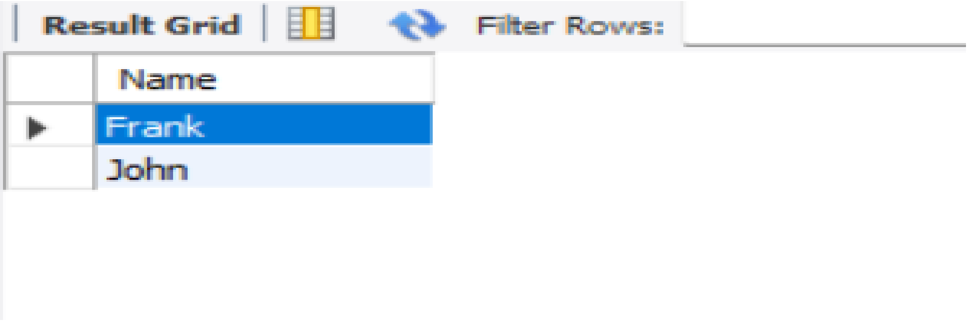


OrderID
2
3
4
5
6

Part B – Correlated & Nested Subqueries (25 marks)

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

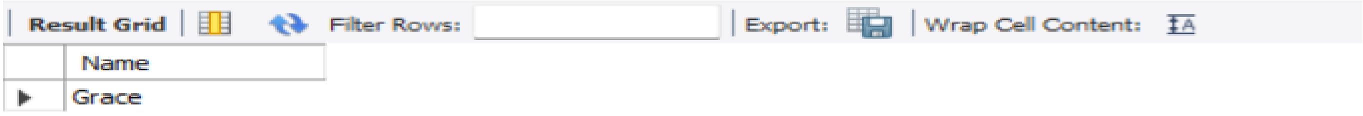
```
SELECT Name
FROM Customers
WHERE CustomerID IN (
   SELECT CustomerID FROM Orders
   WHERE OrderDate IN (
      SELECT OrderDate FROM Orders JOIN Customers USING (CustomerID) WHERE
    Name='John' ));
```



Name
Frank
John

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

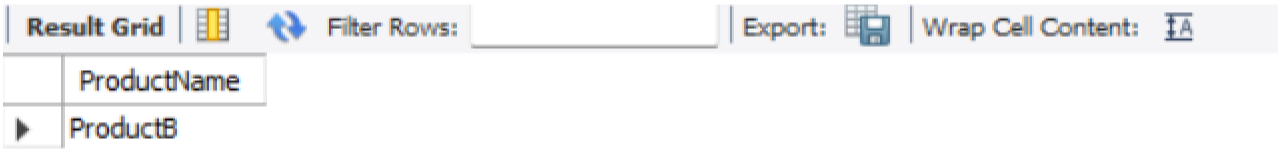
```
SELECT Name
FROM Customers
WHERE CustomerID = (
   SELECT CustomerID FROM Orders ORDER BY OrderDate DESC LIMIT 1);
```



Name
Grace

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

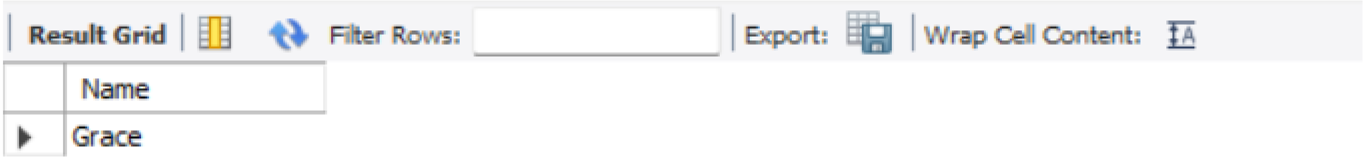
```
SELECT ProductName
FROM Products
WHERE Price = (
   SELECT MIN(Price) FROM Products WHERE Price > (SELECT MIN(Price)
FROM Products));
```



ProductName
ProductB

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

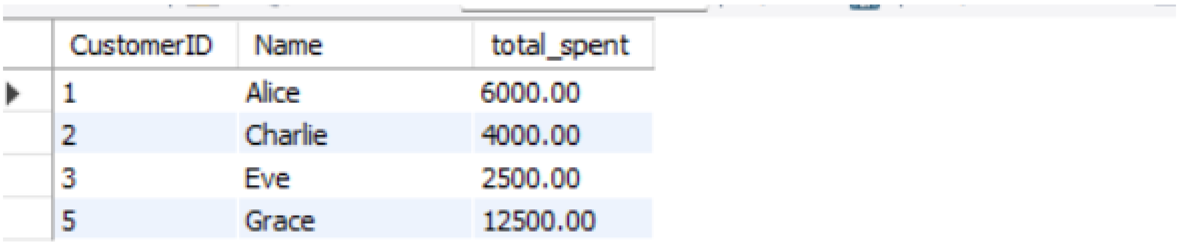
```
SELECT Name
FROM Customers
HAVING SUM((Quantity * Price)) > 2 * (SELECT AVG(total) FROM (
   SELECT SUM(Quantity * Price) AS total FROM Orders
   JOIN OrderDetails USING (OrderID)
   JOIN Products USING (ProductID)
GROUP BY CustomerID) AS T);
```



Name
Grace

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

```
SELECT Name
FROM Customers
HAVING SUM((Quantity * Price)) > ANY (
   SELECT SUM((Quantity * Price)) FROM Customers
   JOIN Orders USING (CustomerID)
   JOIN OrderDetails USING (OrderID)
   JOIN Products USING (ProductID)
   WHERE City='Delhi'
   GROUP BY CustomerID);
```

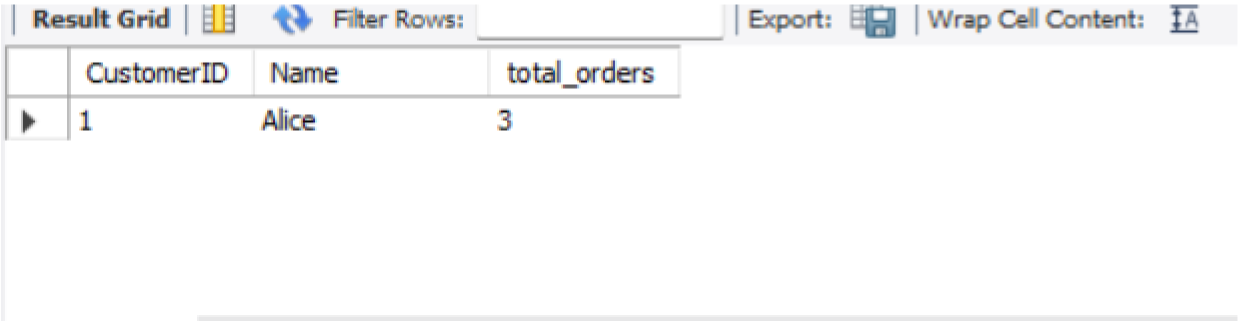


CustomerID	Name	total_spent
1	Alice	6000.00
2	Charlie	4000.00
3	Eve	2500.00
5	Grace	12500.00

Part C – Join + Subquery Mix (30 marks)

11. Use a correlated subquery to find customers who have placed **more orders than the average** number of orders placed by all customers.

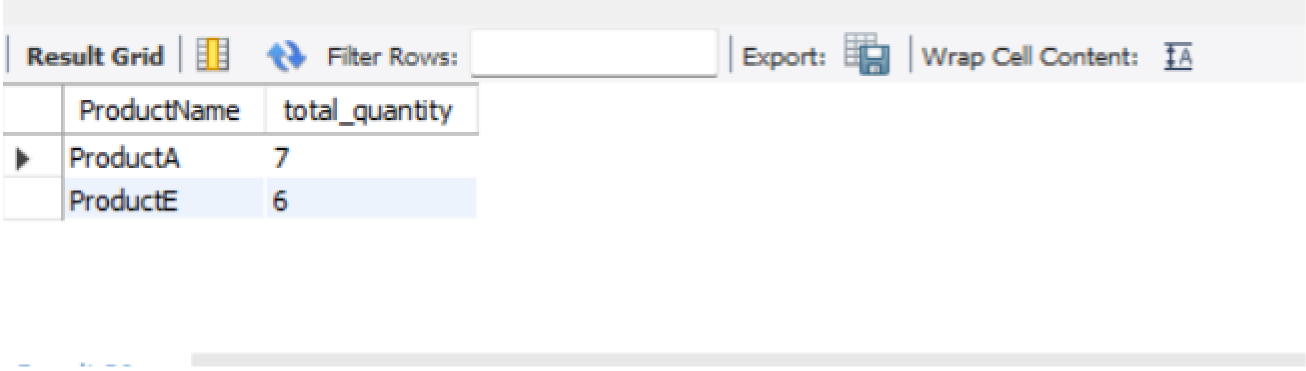
```
SELECT Name
FROM Customers
HAVING COUNT(Orders.OrderID) > (SELECT AVG(cnt) FROM (SELECT
CustomerID, COUNT(OrderID) AS cnt FROM Orders GROUP BY CustomerID) AS
T);
```



CustomerID	Name	total_orders
1	Alice	3

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

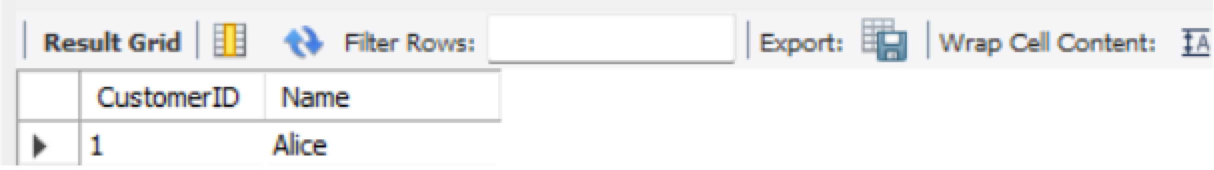
```
SELECT ProductName

FROM Products

HAVING SUM(Quantity) > (SELECT AVG(sum_quantity) FROM (SELECT
ProductID, SUM(Quantity) AS sum_quantity FROM OrderDetails GROUP BY
ProductID) AS T);
```



ProductName	total_quantity
ProductA	7
ProductE	6

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

```
SELECT Name

FROM Customers

WHERE EXISTS (

   SELECT 1 FROM Orders o

   JOIN OrderDetails od USING (OrderID)

   WHERE o.CustomerID = Customers.CustomerID

    AND od.ProductID NOT IN (

    SELECT ProductID FROM OrderDetails GROUP BY ProductID HAVING
COUNT(DISTINCT OrderID) > 1 ));
```

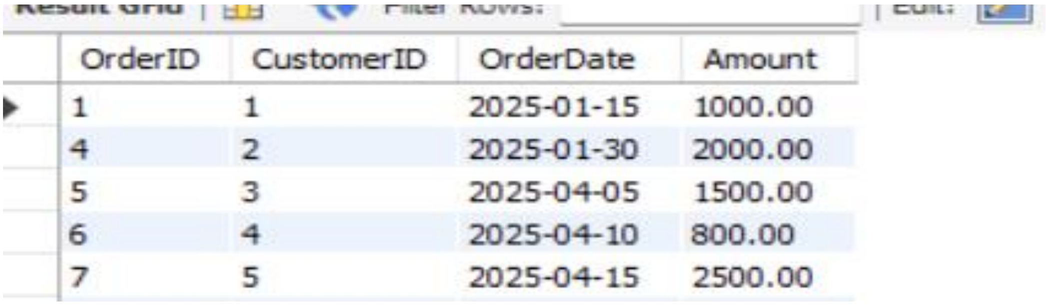


CustomerID	Name
1	Alice

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

```
SELECT o.*
```

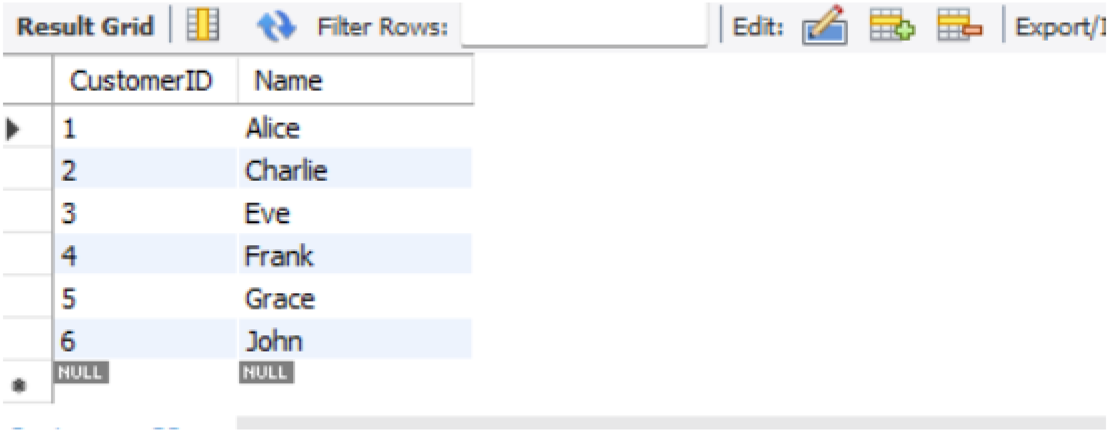
```
FROM Orders o
WHERE o.Amount = (
  SELECT MAX(Amount) FROM Orders WHERE CustomerID = o.CustomerID);
```



OrderID	CustomerID	OrderDate	Amount
1	1	2025-01-15	1000.00
4	2	2025-01-30	2000.00
5	3	2025-04-05	1500.00
6	4	2025-04-10	800.00
7	5	2025-04-15	2500.00

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

```
SELECT Name
FROM Customers
WHERE NOT EXISTS (
  SELECT 1 FROM Orders o
  JOIN OrderDetails od USING (OrderID)
  WHERE o.CustomerID = Customers.CustomerID
   AND od.Quantity > 5);
```

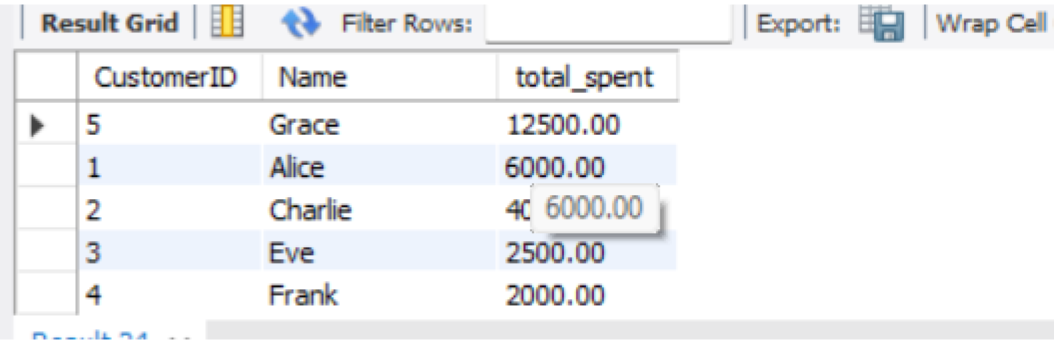


CustomerID	Name
1	Alice
2	Charlie
3	Eve
4	Frank
5	Grace
6	John
NULL	NULL

Part D – Joins & Set Operations (25 marks)

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

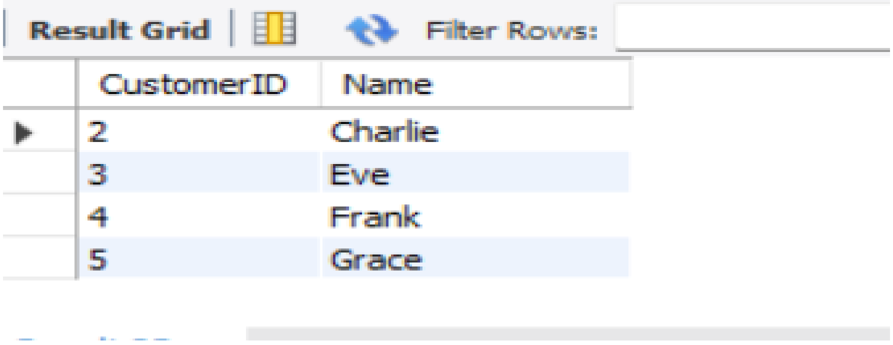
```
SELECT Name
FROM Customers
ORDER BY SUM((Quantity * Price)) DESC LIMIT 5;
```



	CustomerID	Name	total_spent
▶	5	Grace	12500.00
	1	Alice	6000.00
	2	Charlie	40 6000.00
	3	Eve	2500.00
	4	Frank	2000.00

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

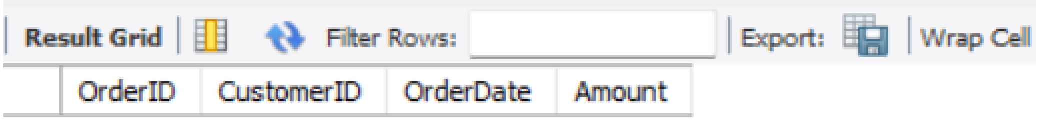
```
SELECT Name

FROM Customers

HAVING COUNT(DISTINCT ProductID) = 1;
```



	CustomerID	Name
▶	2	Charlie
	3	Eve
	4	Frank
	5	Grace

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

```
SELECT o.*

FROM Orders o

WHERE o.Amount NOT IN (SELECT Amount FROM Orders ORDER BY Amount
DESC LIMIT 10);
```



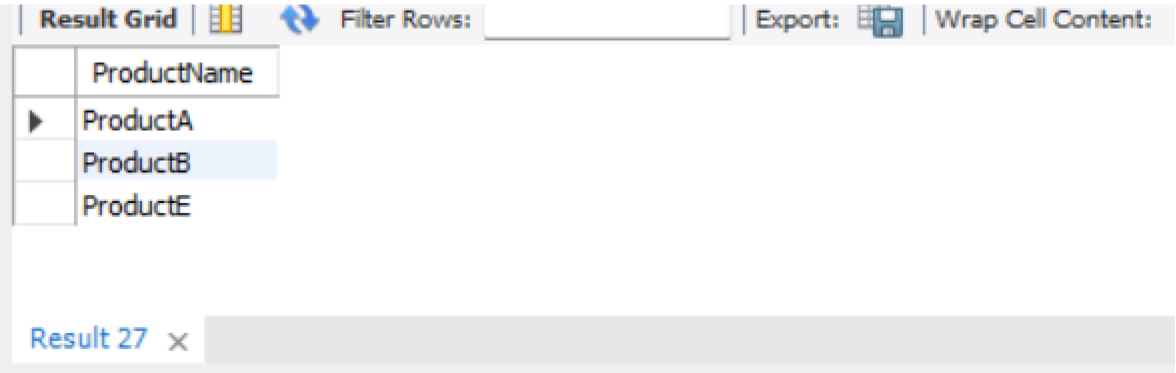
	OrderID	CustomerID	OrderDate	Amount

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

```
SELECT Name

FROM Customers

WHERE EXISTS (

   SELECT 1 FROM Orders o

   WHERE o.CustomerID = Customers.CustomerID

    AND o.OrderDate >= CURDATE() - INTERVAL 7 DAY)
AND NOT EXISTS (
    SELECT 1 FROM Orders o
    WHERE o.CustomerID = Customers.CustomerID
      AND o.OrderDate < CURDATE()
      AND o.OrderDate >= CURDATE() - INTERVAL 30 DAY);
```

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

```
SELECT ProductName
FROM Products
HAVING COUNT(DISTINCT OrderID) = (
    SELECT MAX(cnt) FROM (SELECT ProductID, COUNT(DISTINCT OrderID) AS cnt FROM OrderDetails GROUP BY ProductID) AS T;
```



ProductName
ProductA
ProductB
ProductE