SQL Interview Questions and Queries

# Query 1

## SQL Query:

SELECT C.CustomerID, C.Name, SUM(O.Amount) AS TOT\_PURCHASE\_AMOUNT  
 FROM Customers AS C  
 JOIN Orders O ON C.CustomerID = O.CustomerID  
 GROUP BY C.CustomerID, C.Name  
 ORDER BY TOT\_PURCHASE\_AMOUNT DESC  
 LIMIT 5;

## Explanation:

This query finds the top 5 customers with the highest total purchase amount.   
 It joins the Customers and Orders tables on CustomerID, groups the results by CustomerID and Name,   
 calculates the total purchase amount using SUM, and sorts the results in descending order of total purchase amount.   
 The LIMIT 5 clause restricts the output to the top 5 customers.

# Query 2

## SQL Query:

SELECT DISTINCT Salary  
 FROM Employees  
 ORDER BY Salary DESC  
 LIMIT 1 OFFSET n-1;

## Explanation:

This query finds the nth highest salary from the Employees table.   
 It selects distinct salary values, orders them in descending order,   
 and uses LIMIT 1 OFFSET n-1 to skip the first n-1 rows and return the nth row.

# Query 3

## SQL Query:

SELECT ProductID, SUM(Quantity) AS TOT\_QTY\_SOLD, DATE\_TRUNC('month', SaleDate) AS MONTH  
 FROM Sales  
 GROUP BY ProductID, MONTH  
 ORDER BY ProductID, MONTH;

## Explanation:

This query finds the total quantity sold for each product per month.   
 It groups the results by ProductID and the month of SaleDate,   
 calculates the total quantity sold using SUM, and orders the results by ProductID and month.

# Query 4

## SQL Query:

SELECT EmployeeID, Name  
 FROM Employees  
 GROUP BY EmployeeID, Name  
 HAVING COUNT(DISTINCT ManagerID) > 1;

## Explanation:

This query finds all employees who have more than one manager.   
 It groups the results by EmployeeID and Name, and uses HAVING COUNT(DISTINCT ManagerID) > 1 to filter the groups with more than one distinct ManagerID.

# Query 5

## SQL Query:

SELECT ProductID, SUM(Quantity) AS SALES\_QTY  
 FROM OrderDetails  
 GROUP BY ProductID  
 ORDER BY SALES\_QTY DESC  
 LIMIT 3;

## Explanation:

This query finds the top 3 products with the highest sales quantity.   
 It groups the results by ProductID, calculates the sales quantity using SUM,   
 orders the results in descending order of sales quantity, and limits the output to the top 3 products.

# Query 6

## SQL Query:

SELECT CustomerID, MAX(OrderDate) AS SECOND\_RECENT\_ORDER\_DATE  
 FROM (  
 SELECT CustomerID, OrderDate,  
 ROW\_NUMBER() OVER (PARTITION BY CustomerID ORDER BY OrderDate DESC) AS ORDERRANK  
 FROM Orders  
 ) AS RANKEDORDERS  
 WHERE ORDERRANK = 2  
 GROUP BY CustomerID;

## Explanation:

This query finds the second most recent order date for each customer.   
 It uses a subquery to rank the orders for each customer by OrderDate in descending order,   
 then selects the second highest rank (ORDERRANK = 2) for each customer.

# Query 7

## SQL Query:

SELECT CustomerID, OrderDate,  
 DATEDIFF(day, LAG(OrderDate) OVER (PARTITION BY CustomerID ORDER BY OrderDate)) AS DaysSinceLastOrder  
 FROM Orders;

## Explanation:

This query calculates the difference in days between successive orders for each customer.   
 It uses the LAG function to get the previous order date for each customer and calculates the difference using DATEDIFF.

# Query 8

## SQL Query:

SELECT c.CustomerID, c.Name  
 FROM Customers c  
 LEFT JOIN Orders o ON c.CustomerID = o.CustomerID  
 WHERE o.CustomerID IS NULL;

## Explanation:

This query finds customers who have not placed any orders.   
 It performs a LEFT JOIN between the Customers and Orders tables and filters the results where the CustomerID in Orders is NULL.