SQL INTERVIEW PREPAPRATION PART 4.1

WINDOWS FUNCTIONS QUESTIONS:

1. Find the Second Highest Salary

Scenario:
You have an Employees table with columns EmpID, EmpName, and Salary.
Question:
Write a query to find the second highest salary in the table using window functions.
Solution:

```
WITH RankedSalary AS (
   SELECT Salary,
        DENSE_RANK() OVER (ORDER BY Salary DESC) AS Salary_Rank
   FROM Employees
)
SELECT Salary
FROM RankedSalary
WHERE Salary_Rank = 2;
```

2. Running Total

Scenario:
You have a Sales table with columns SaleID, Product, Amount, and SaleDate.

Question:
Write a query to calculate the running total of sales amount ordered by SaleDate.
Solution:

```
SELECT SaleDate,
        Amount,
        SUM(Amount) OVER (ORDER BY SaleDate) AS Sales_Running_Total
FROM Sales;
```

3. Find Top N Percent of Salaries

Scenario:
You have an Employees table.
Question:
Write a query to find the top 10% of salaries in the table using window functions.
Solution:
Query using NTILE():

```
WITH RankedSalary AS(
        SELECT EmpID,
```

```
            EmpName,
            Salary,
            NTILE(10) OVER( ORDER BY Salary DESC) AS Salary_Tile
        FROM Employees
)
SELECT EmpID, EmpName, Salary
FROM RankedSalary
WHERE Salary_Tile = 1;
```

Query using PERCENT_RANK():

```
WITH RankedSalary AS(
        SELECT EmpID,
            EmpName,
            Salary,
            PERCENT_RANK() OVER( ORDER BY Salary DESC) AS Percent_Rank
        FROM Employees
)
SELECT EmpI, EmpName, Salary
FROM RankedSalary
WHERE Percentile_Rank <= 0.10;
```

Explanation:

1. **PERCENT_RANK()**:
    - This window function calculates the relative rank of each row as a percentage of the total rows, ranging from 0 to 1.
    - The ORDER BY Salary DESC ranks the rows based on descending salaries, with the highest salary having the smallest rank value.
2. **WITH RankedSalaries**:
    - A common table expression (CTE) is used to calculate the percentage rank of each employee's salary.
3. **WHERE Percent_Rank <= 0.10**:
    - Filters out the rows that fall in the top 10% of salaries.

4. Average Salary by Department

Scenario:
You have an Employees table with columns EmpID, EmpName, Salary, and DepartmentID.
Question:

Write a query to calculate each employee's salary compared to their department's average salary.

Solution:

```
WITH AverageSalary AS(
	SELECT
		DepartmentID,
		AVG(Salary) OVER(PARTITION BY DepartmentID) AS Average_Salary
	FROM Employees
)
SELECT
	e.EmpID,
	e.EmpName,
	e.DepartmentID,
	e.Salary,
	a.Average_Salary
FROM Employees e
LEFT JOIN AverageSalary a
ON e.DepartmentID = a.DepartmentID;
```

5. Rank Products by Sales

Scenario:

You have a Sales table with columns ProductID, ProductName, and SaleAmount.

Question:

Write a query to rank products by their sales amount within each category.

Solution:

```
SELECT ProductID,
	ProductName,
	SaleAmount,
	DENSE_RANK() OVER(PARTITION BY Category ORDER BY SaleAmount DESC) AS
	Product_Rank
FROM Sales;
```

6. Lag and Lead

Scenario:

You have a Stocks table with columns StockID, StockPrice, and StockDate.

Question:

Write a query to calculate the price difference between the current day and the previous day for each stock.

Solution:

```
SELECT
  StockID,
  StockDate,
```

```
    StockPrice,
    LAG(StockPrice) OVER (PARTITION BY StockID ORDER BY StockDate) AS Prev_Price,
    StockPrice - LAG(StockPrice) OVER (PARTITION BY StockID ORDER BY StockDate) AS
Price_Difference
FROM Stocks;
```

7. Nth Highest Value

Scenario:
You have a Scores table with columns StudentID, Subject, and Score.
Question:
Write a query to find the 3rd highest score in each subject.
Solution:

```
WITH RankedScores AS(
        SELECT StudentID,
                Subject,
                Score,
                RANK() OVER( PARTITION BY Subject ORDER BY Score DESC) AS Score_Rank
        FROM Scores
)
SELECT StudentID,
        Subject,
        Score
FROM RankedScores
WHERE Score_Rank = 3;
```

8. Detect Consecutive Attendance

Scenario:
You have an Attendance table with columns EmpID, Date, and Status (Present/Absent).
Question:
Write a query to find employees who were absent for three or more consecutive days.
Solution:

```
WITH AttendanceWithLag AS(
        SELECT EmpID,
                Date,
                Status,
                LAG(Date) OVER (PARTITION BY EmpID ORDER BY Date) AS Prev_Date
FROM Attendance
),
ConsecutiveAbsences AS(
        SELECT EmpID,
                Date,
                Status,
```

```
        CASE
            WHEN Status = 'Absent' AND DATEDIFF(Date, Prev_Date) = 1 THEN 1
            ELSE 0
        END AS Is_Consecutive
    FROM AttendanceWithLag
),
AbsenceGroups AS(
    SELECT EmpID,
        Date,
        Status,
        SUM(Is_Consecutive) OVER (PARTITION BY EmpID ORDER BY Date ROWS
        BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS Absence_Streak
    FROM ConsecutiveAbsences
)
SELECT DISTINCT EmpID
FROM AbsenceGroups
WHERE AbsenceStreak >= 3;
```

9. Find Customers with Consecutive Purchases

Scenario:
You have an Orders table with columns OrderID, CustomerID, and OrderDate.
Question:
Write a query to find customers who placed orders on consecutive days.
Solution:

```
WITH OrdersWithLag AS (
    SELECT OrderID,
        CustomerID,
        OrderDate,
        LAG(OrderDate) OVER (PARTITION BY CustomerID ORDER BY OrderDate) AS
        Prev_Date
    FROM Orders
)
SELECT DISTINCT CustomerID
FROM OrdersWithLag
WHERE DATEDIFF(OrderDate, Prev_Date) = 1;
```

10. Percent Rank

Scenario:
You have a Scores table with columns StudentID and Score.
Question:
Write a query to calculate the percentile rank of each student's score.
Solution:

```
SELECT StudentID,
        Score,
        PERCENT_RANK() OVER( ORDERBY Score DESC) AS Rank_Percent
FROM Scores;
```