Window Function in SQL

1. What is a window function in SQL, and how does it differ from an aggregate function?

Answer:

- **Window Function**: Performs a calculation across a set of rows that are related to the current row, without collapsing the result into a single row.
- **Aggregate Function**: Aggregates multiple rows into a single output (e.g., `SUM()`, `AVG()`).

Example:

```
SELECT name, salary, AVG(salary) OVER () AS avg_salary
FROM employees;
```

This calculates the average salary without collapsing rows.

---

2. Explain the use of `ROW_NUMBER()` in SQL and provide an example of when you would use it.

Answer:

- `ROW_NUMBER()` assigns a unique sequential integer to rows within a partition, starting from 1 for each partition.

Example: Find the top 3 highest-paid employees in each department.

```
SELECT name, department, salary,
       ROW_NUMBER() OVER (PARTITION BY department ORDER BY salary
DESC) AS rank
FROM employees
WHERE rank <= 3;
```

---

3. How do `RANK()` and `DENSE_RANK()` differ in SQL? When would you use each one?

Answer:

Window Function in SQL

- **RANK()**: Assigns a rank with gaps in ranking for ties.
- **DENSE_RANK()**: Assigns rank without gaps.

Example:

```
RANK Example
SELECT name, salary, RANK() OVER (ORDER BY salary DESC) AS salary_rank
FROM employees;

 DENSE_RANK Example
SELECT name, salary, DENSE_RANK() OVER (ORDER BY salary DESC) AS
salary_rank FROM employees;
```

---

4. Can you explain the purpose of `PARTITION BY` in a window function, and how it affects the result?

Answer:

- `PARTITION BY` divides the result set into partitions and applies the window function within each partition.

Example:

```
SELECT name, department, salary,
       RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS
department_rank
FROM employees;
```

This ranks employees within each department separately.

---

5. What is the difference between `LAG()` and `LEAD()` functions? How would you use them in a real-world scenario?

Answer:

- `LAG()`: Returns the value of the preceding row.
- `LEAD()`: Returns the value of the following row.

Window Function in SQL

Example: Comparing each month's sales with the previous month.

```
SELECT month, sales,
       LAG(sales, 1) OVER (ORDER BY month) AS previous_month_sales
FROM sales_data;
```

---

6. How can you calculate a running total (cumulative sum) using a window function in SQL?

Answer: You can use `SUM()` with `OVER()` clause.

Example:

```
SELECT name, salary,
       SUM(salary) OVER (ORDER BY salary) AS running_total
FROM employees;
```

---

7. Write a query to calculate the moving average of sales over the last 3 months for each product.

Answer:

```
SELECT product_id, month, sales,
       AVG(sales) OVER (PARTITION BY product_id ORDER BY month ROWS 2
PRECEDING) AS moving_avg
FROM sales_data;
```

This calculates a 3-month moving average (current month and 2 preceding months).

---

8. How would you calculate the percentage of total for each row using a window function?

Answer:

```
SELECT name, salary,
```

Window Function in SQL

```
       salary * 100.0 / SUM(salary) OVER () AS percent_of_total
FROM employees;
```

---

9. Explain the use of `NTILE()` function in SQL and provide an example where it can be applied.

Answer:

- `NTILE()` divides rows into a specified number of roughly equal-sized buckets.

Example: Divide employees into 4 salary quartiles.

```
SELECT name, salary, NTILE(4) OVER (ORDER BY salary) AS
salary_quartile
FROM employees;
```

---

10. What is the difference between `OVER()` with `PARTITION BY` and `OVER()` with `ORDER BY`?

Answer:

- **`PARTITION BY`**: Groups rows into partitions where the window function is applied independently.
- **`ORDER BY`**: Orders rows within a partition to apply the window function.

Example:

```
SELECT name, salary, SUM(salary) OVER (PARTITION BY department ORDER
BY salary) AS running_total
FROM employees;
```

---

11. How would you calculate the first and last value in a partitioned dataset using window functions?

Answer: You can use `FIRST_VALUE()` and `LAST_VALUE()` functions.

Window Function in SQL

Example:

```
SELECT name, salary,
       FIRST_VALUE(salary) OVER (PARTITION BY department ORDER BY
salary) AS first_salary,
       LAST_VALUE(salary) OVER (PARTITION BY department ORDER BY
salary) AS last_salary
FROM employees;
```

---

12. How can window functions help in calculating year-over-year growth in a time series dataset?

Answer: You can use `LAG()` to compare the current year's sales with the previous year.

Example:

```
SELECT year, sales,
       (sales - LAG(sales, 1) OVER (ORDER BY year)) / LAG(sales, 1)
OVER (ORDER BY year) * 100 AS yoy_growth
FROM sales_data;
```

---

13. Given a table of employee salaries, how would you rank employees by salary within each department using window functions?

Answer:

sql
Copy code

```
SELECT name, department, salary,
       RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS
department_rank
FROM employees;
```

Window Function in SQL

14. What are the performance considerations when using window functions in large datasets, and how can you optimize them?

Answer:

- **Performance Considerations**:
  - Window functions can be computationally expensive on large datasets.
  - Sorting large partitions can be costly.
- **Optimization**:
  - Ensure indexes are on partitioning and ordering columns.
  - Minimize the number of rows in partitions.
  - Use `ROWS` instead of `RANGE` when possible for performance gains.

---

15. How can you use window functions to identify duplicate rows or records based on specific criteria?

Answer: You can use `ROW_NUMBER()` to identify duplicates.

Example:

```
WITH CTE AS (
    SELECT name, salary, ROW_NUMBER() OVER (PARTITION BY name, salary
ORDER BY name) AS row_num
    FROM employees
)
SELECT * FROM CTE WHERE row_num > 1;
```

This identifies duplicate rows based on the `name` and `salary` columns.