01

Window Functions in SQL

02

What Are Window Functions?

A window function in SQL performs calculations across a set of rows related to the current row, but unlike aggregate functions, it doesn’t group the rows into a single result.

Instead, it adds a calculated value to eachrow while keeping all the original rows intact.

03

Syntax of Window Functions

```
FUNCTION_NAME(expression) OVER (
  PARTITION BY column
  ORDER BY column
)
```

- **FUNCTION_NAME:** SUM, AVG, RANK, etc.
- **Expression:** Column/Calculation
- **OVER:** Specifies a window function
- **PARTITION BY:** Divides data into groups.
- **ORDER BY:** Determines calculation order within each group.

04

Types of Window Functions

- **Ranking:** ROW_NUMBER, RANK, DENSE_RANK, NTILE
- **Value:** LAG, LEAD
- **Aggregate:** SUM, AVG, MIN, MAX

05

Types of Ranking Functions

- ROW_NUMBER: Assigns unique numbers.
- RANK: Skips ranks for ties.
- DENSE_RANK: No skipped ranks.
- NTILE: Divides rows into buckets.

Types of Value Functions

- LAG():- Gets the value of a column from previous row.
- LEAD():- Gets the value of a column from next row.
- Both functions are flexible and help in analyzing trends, identifying gaps, or comparing sequential data.

07

Types of Aggregate Functions

- SUM: Running totals.
- AVG: Averages per group.
- MIN/MAX: Find minimum or maximum per group.

Example for ROW_NUMBER

Code Example:

```
SELECT
    ROW_NUMBER() OVER (ORDER BY sales DESC) AS row_number,
    salesperson,
    sales
FROM sales_data;
```

Result Table:

Row_Number	Salesperson	Sales
1	Alice	500
2	Bob	400
3	Charlie	300

Example for RANK

Code Example:

```
SELECT
    RANK() OVER (ORDER BY sales DESC) AS rank,
    salesperson,
    sales
FROM sales_data;
```

Result Table (Ranks Tied Sales)

Rank	Salesperson	Sales
1	Alice	500
2	Bob	400
2	Charlie	400

Example for DENSE_RANK

Code Example:

```
SELECT
    DENSE_RANK() OVER (ORDER BY sales DESC) AS dense_rank,
    salesperson,
    sales
FROM sales_data;
```

Result Table (Dense Ranks with No Skipped Ranks)

Dense_Rank	Salesperson	Sales
1	Alice	500
2	Bob	400
2	Charlie	400
3	David	300

Value Functions (LAG & LEAD)

Code Example:

```
SELECT
    salesperson,
    sales,
    LAG(sales) OVER (ORDER BY sales DESC) AS prev_sales,
    LEAD(sales) OVER (ORDER BY sales DESC) AS next_sales
FROM sales_data;
```

Result Table:

Salesperson	Sales	Prev_Sales	Next_Sales
Alice	500	NULL	400
Bob	400	500	300
Charlie	300	400	NULL

Value Functions (LAG & LEAD)

Code Example:

```
SELECT
    salesperson,
    sales,
    LAG(sales) OVER (ORDER BY sales DESC) AS prev_sales,
    LEAD(sales) OVER (ORDER BY sales DESC) AS next_sales
FROM sales_data;
```

Result Table:

Salesperson	Sales	Prev_Sales	Next_Sales
Alice	500	NULL	400
Bob	400	500	300
Charlie	300	400	NULL

Aggregate Functions - SUM

Code Example:

```
SELECT
    salesperson,
    sales,
    SUM(sales) OVER (ORDER BY sales ASC) AS running_total
FROM sales_data;
```

Result Table:

Salesperson	Sales	Running_Total
Charlie	300	300
Bob	400	700
Alice	500	1200

Aggregate Functions - AVG

Code Example:

```
SELECT
    department,
    employee_id,
    salary,
    AVG(salary) OVER (PARTITION BY department) AS avg_salary
FROM employees;
```

Result Table:

Department	Employee_ID	Salary	Avg_Salary
HR	101	8000	7500
HR	102	7000	7500
IT	103	12000	11500
IT	104	11000	11500

Aggregate Functions - MIN

Code Example:

```
SELECT
    department,
    employee_id,
    salary,
    MIN(salary) OVER (PARTITION BY department) AS min_salary
FROM employees;
```

Result Table:

Department	Employee_ID	Salary	Min_Salary
HR	101	8000	7000
HR	102	7000	7000
IT	103	12000	11000
IT	104	11000	11000

Aggregate Functions - MAX

Code Example:

```
SELECT
    department,
    employee_id,
    salary,
    MAX(salary) OVER (PARTITION BY department) AS max_salary
FROM employees;
```

Result Table:

Department	Employee_ID	Salary	Max_Salary
HR	101	8000	8000
HR	102	7000	8000
IT	103	12000	12000
IT	104	11000	12000

When to Use Window Functions

- Trend Analysis: Compare values over time
- Rankings: Assign ranks within groups..
- Cumulative Totals: Running totals or averages.
- Sequential Comparisons: Find differences between rows.

Conclusion

- Window functions bring power, flexibility, and clarity to your SQL queries.
- Ideal for reporting, dashboards, and analytical use-cases.
- Start using them to level up your data skills!

FOLLOW ME FOR MORE

Bobbili Jagadeesh
https://www.linkedin.com/in/bobbili-jagadeesh