WINDOW FUNCTIONS

Window functions in SQL are used to perform calculations across a set of rows that are related to the current row, but without collapsing the results into a single group, as aggregate functions do. Window functions enable complex calculations like running totals, ranking, and percentiles while maintaining the individual rows in the result. Here's how window functions are applied in different industries:

1. Retail

- Use Case: Running Total of Sales

- Function: `SUM()` over a partitioned window

- Example: A retailer might want to calculate a running total of sales for each store over time. Using a window function, they can partition the sales data by store and order the data by date to compute the cumulative sales.

```
SELECT store_id, sale_date, sale_amount,

SUM(sale_amount) OVER (PARTITION BY store_id ORDER BY sale_date)

AS running_total

FROM sales;
```

2. Finance

- Use Case: Ranking Customers by Loan Amount

- Function: `RANK()` or `DENSE_RANK()`

- Example: A financial institution might want to rank customers based on their loan amounts, possibly to prioritize larger loans for processing or to analyze high-value customers.

```
SELECT customer_id, loan_amount,

RANK() OVER (ORDER BY loan_amount DESC) AS loan_rank

FROM loans;
```

3. Healthcare

- Use Case: Analyzing Patient Readmission Rates

- Function: `ROW_NUMBER()`

- Example: A hospital might want to assign a sequential number to each patient admission to track how many times a patient was readmitted within a certain period.

```
SELECT patient_id, admission_date,

ROW_NUMBER() OVER (PARTITION BY patient_id ORDER BY admission_date) AS
admission_number

FROM admissions;
```

4. Telecommunications

- Use Case: Finding the Longest Call

- Function: `MAX()` over a window partitioned by customer

- Example: A telecom provider may want to find the longest call duration for each customer to analyze call behavior patterns.

```
SELECT customer_id, call_duration,

MAX(call_duration) OVER (PARTITION BY customer_id) AS longest_call

FROM call_logs;
```

5. Logistics

- Use Case: Tracking Shipments Over Time

- Function: `LEAD()` or `LAG()`

- Example: A logistics company might use window functions to track the time between two shipments for each customer, calculating the difference between consecutive shipment dates.

```
SELECT customer_id, shipment_date,

LEAD(shipment_date, 1) OVER (PARTITION BY customer_id ORDER BY shipment_date)
AS next_shipment,

LAG(shipment_date, 1) OVER (PARTITION BY customer_id ORDER BY shipment_date)
AS previous_shipment

FROM shipments;
```

6. Education

- Use Case: Ranking Students Based on Grades

- Function: `DENSE_RANK()`

- Example: In an educational institution, students can be ranked within each course based on their grades using window functions. This helps in determining the top-performing students.

```
SELECT student_id, course_id, grade,

DENSE_RANK() OVER (PARTITION BY course_id ORDER BY grade DESC) AS rank

FROM grades;
```

7. Real Estate

- Use Case: Tracking Changes in Property Prices

- Function: `LAG()` and `LEAD()`

- Example: A real estate agency could use window functions to compare current property prices with past and future prices, helping them analyze market trends.

```
SELECT property_id, price,
LAG(price) OVER (ORDER BY sale_date) AS previous_price,
LEAD(price) OVER (ORDER BY sale_date) AS next_price
FROM property_sales;
```

8. Manufacturing

- Use Case: Calculating Running Average Production

- Function: `AVG()` over a partitioned window

- Example: In a manufacturing plant, window functions can be used to calculate a running average of production output for each machine to monitor performance trends over time.

```
SELECT machine_id, production_date, output,
AVG(output) OVER (PARTITION BY machine_id ORDER BY production_date ROWS BETWEEN 7 PRECEDING AND CURRENT ROW) AS avg_output
FROM production_data;
```

9. Human Resources

- Use Case: Tracking Employee Salary Changes

- Function: `LAG()` for salary tracking

- Example: HR departments may use window functions to track salary changes for employees over time, showing the current salary, previous salary, and calculating the percentage increase.

```
SELECT employee_id, salary,
LAG(salary) OVER (PARTITION BY employee_id ORDER BY salary_date) AS previous_salary
FROM employee_salaries;
```

Common Window Functions:

1. `ROW_NUMBER()`: Assigns a unique number to rows within a partition.
2. `RANK()`: Provides a ranking of rows within a partition, with gaps if ties exist.
3. `DENSE_RANK()`: Similar to `RANK()` but without gaps between rank values.
4. `LEAD()`: Accesses data from subsequent rows in a result set.
5. `LAG()`: Accesses data from preceding rows in a result set.
6. `SUM()` / `AVG()` / `COUNT()`: Performs aggregate calculations over a window of rows.

Window functions provide insights while maintaining detailed data, making them valuable in industries where trends, rankings, and comparisons are essential.