WINDOW FUNCTIONS IN SQL

Window functions in SQL are powerful tools used across various industries to analyze and manage data while keeping the original rows intact. Let's break down the three categories of window functions you mentioned and how each is applied in real-world industries.

1. Use of Aggregate Functions as Window Functions

These functions are applied to a set of rows and return a single value for each row while retaining all individual rows in the result set. Common aggregate functions include `SUM()`, `AVG()`, `MIN()`, `MAX()`, and `COUNT()`.

Industry-Specific Use Cases:

- Retail:

- Use Case: Cumulative Sales by Day
  - Function: `SUM() OVER (PARTITION BY store_id ORDER BY sale_date)`
  - Retailers track cumulative sales by day for each store to understand sales trends.
  - Example:

```
SELECT store_id, sale_date, sale_amount,
SUM(sale_amount) OVER (PARTITION BY store_id ORDER BY sale_date) AS running_total
FROM sales;
```

- Finance:

- Use Case: Average Loan Amount per Customer
  - Function: `AVG() OVER (PARTITION BY customer_id)`
  - Financial institutions may calculate the average loan amount for each customer.
  - Example:

```
SELECT customer_id, loan_amount,
AVG(loan_amount) OVER (PARTITION BY customer_id) AS avg_loan
FROM loans;
```

- Healthcare:

- Use Case: Maximum Patient Age for a Condition
  - Function: `MAX() OVER (PARTITION BY condition_id)`
  - Hospitals may track the maximum age of patients diagnosed with a specific condition.
  - Example:

```
SELECT condition_id, patient_age,
MAX(patient_age) OVER (PARTITION BY condition_id) AS oldest_patient
FROM patient_data;
```

- Telecom:

- Use Case: Count of Calls per Customer

- Function: `COUNT() OVER (PARTITION BY customer_id)`

- Telecom companies may use this to track how many calls each customer made.

- Example:

```
SELECT customer_id, call_duration,
COUNT(*) OVER (PARTITION BY customer_id) AS call_count
FROM call_logs;
```

- Manufacturing:

- Use Case: Minimum Production Output Per Machine

- Function: `MIN() OVER (PARTITION BY machine_id)`

- Factories can monitor the lowest production output of each machine.

- Example:

```
SELECT machine_id, production_output,
MIN(production_output) OVER (PARTITION BY machine_id) AS min_output
FROM production;
```

2. Use of Ranking Window Functions

Ranking functions allow you to assign a rank or position to each row within a partition of a result set.

Industry-Specific Use Cases:

- Retail:

- Use Case: Rank Products by Sales

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

- Stores rank products based on sales volume to identify best-sellers.

- Example:

```
SELECT product_id, sale_amount,
RANK() OVER (ORDER BY sale_amount DESC) AS product_rank
FROM product_sales;
```

- Finance:

- Use Case: Ranking Customers by Loan Size

- Function: `ROW_NUMBER() OVER (ORDER BY loan_amount DESC)`

- Banks may assign a unique rank to customers based on loan size.

- Example:

```
SELECT customer_id, loan_amount,
ROW_NUMBER() OVER (ORDER BY loan_amount DESC) AS loan_rank
FROM loans;
```

- Education:

- Use Case: Ranking Students by Scores
- Function: `DENSE_RANK() OVER (PARTITION BY course_id ORDER BY grade DESC)`
- Schools rank students based on grades within a specific course.
- Example:

```
SELECT student_id, course_id, grade,
DENSE_RANK() OVER (PARTITION BY course_id ORDER BY grade DESC) AS student_rank
FROM student_grades;
```

- Telecommunications:

- Use Case: Divide Customers into Percentiles Based on Usage
- Function: `NTILE() OVER (ORDER BY data_usage)`
- Telecom providers can divide customers into quartiles based on their data usage.
- Example:

```
SELECT customer_id, data_usage,
NTILE(4) OVER (ORDER BY data_usage) AS usage_quartile
FROM customer_data;
```

- Logistics:

- Use Case: Calculate Cumulative Distribution of Delivery Times
- Function: `CUME_DIST() OVER (ORDER BY delivery_time)`
- Logistics companies track the cumulative distribution of delivery times.
- Example:

```
SELECT order_id, delivery_time,
CUME_DIST() OVER (ORDER BY delivery_time) AS delivery_percentile
FROM delivery_data;
```

3. Use of Analytical Window Functions

Analytical window functions (`LEAD()`, `LAG()`, `FIRST_VALUE()`, `LAST_VALUE()`) provide access to data from subsequent, preceding, or specific rows relative to the current row.

Industry-Specific Use Cases:

- Finance:

- Use Case: Compare Current Loan with Previous Loan

- Function: `LAG()`

- Banks compare a customer's current loan amount with their previous one.

- Example:

```
SELECT customer_id, loan_date, loan_amount,

LAG(loan_amount) OVER (PARTITION BY customer_id ORDER BY loan_date) AS previous_loan

FROM loans;
```

- Healthcare:

- Use Case: Identify the First Diagnosis for Each Patient

- Function: `FIRST_VALUE() OVER (PARTITION BY patient_id ORDER BY diagnosis_date)`

- Healthcare providers track the first diagnosis received by each patient.

- Example:

```
SELECT patient_id, diagnosis_date, diagnosis,

FIRST_VALUE(diagnosis) OVER (PARTITION BY patient_id ORDER BY diagnosis_date) AS first_diagnosis

FROM patient_diagnoses;
```

- Telecom:

- Use Case: Track Data Usage Changes

- Function: `LEAD()`

- Telecom companies can monitor changes in data usage from one month to the next.

- Example:

```
SELECT customer_id, usage_month, data_usage,

LEAD(data_usage) OVER (PARTITION BY customer_id ORDER BY usage_month) AS next_month_usage

FROM usage_data;
```

- Retail:

- Use Case: Track First and Last Sale of Each Product

- Function: `FIRST_VALUE()`, `LAST_VALUE()`

- Retailers track the first and last sale date of a product to identify product lifecycle.

- Example:

```
SELECT product_id, sale_date,

FIRST_VALUE(sale_date) OVER (PARTITION BY product_id ORDER BY sale_date) AS first_sale,

LAST_VALUE(sale_date) OVER (PARTITION BY product_id ORDER BY sale_date) AS last_sale

FROM sales;
```

- Manufacturing:

- Use Case: Monitor Production Trends

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

- Manufacturers compare production outputs for consecutive days to identify fluctuations.

- Example:

```
SELECT machine_id, production_date, production_output,

LAG(production_output) OVER (PARTITION BY machine_id ORDER BY production_date) AS previous_output,

LEAD(production_output) OVER (PARTITION BY machine_id ORDER BY production_date) AS next_output

FROM production_data;
```

Summary:

Window functions provide granular insights by maintaining row-level detail while performing complex calculations. Across industries, they are used for:

1. Monitoring trends (e.g., sales, loan amounts, production output)
2. Ranking individuals (e.g., customers, students, products)
3. Comparing current data with past or future data (e.g., using `LAG()`, `LEAD()`)
4. Calculating running totals, averages, and percentiles to provide actionable insights without losing individual row details.

They are versatile tools that help data analysts and scientists in industries ranging from finance to healthcare gain deeper insights into datasets without summarizing or collapsing rows, making them indispensable in day-to-day operations.