Here's a comprehensive explanation of various Window Functions (Analytic Functions) in SQL:

1. ROW\_NUMBER():

ROW\_NUMBER() assigns a unique sequential number to each row within the result set. It is typically used to generate a unique identifier for each row.

Example:

```sql

SELECT ROW\_NUMBER() OVER (ORDER BY salary DESC) AS row\_num, employee\_name, salary

FROM employees;

```

2. RANK():

RANK() assigns a unique rank to each row within the result set based on the ORDER BY clause. Rows with the same values receive the same rank, and the subsequent rank is skipped accordingly.

Example:

```sql

SELECT RANK() OVER (ORDER BY sales\_amount DESC) AS sales\_rank, product\_name, sales\_amount

FROM products;

```

3. DENSE\_RANK():

DENSE\_RANK() is similar to RANK(), but it does not leave gaps for duplicate values. Rows with the same values receive the same rank, and the subsequent rank is not skipped.

Example:

```sql

SELECT DENSE\_RANK() OVER (ORDER BY customer\_points DESC) AS rank, customer\_name, customer\_points

FROM customers;

```

4. NTILE(n):

NTILE(n) divides the result set into 'n' equally-sized buckets or partitions, and assigns a bucket number to each row based on its position within those partitions.

Example:

```sql

SELECT NTILE(4) OVER (ORDER BY transaction\_date) AS quartile, transaction\_id, transaction\_date

FROM transactions;

```

5. SUM(), AVG(), COUNT(), MIN(), MAX():

These are aggregate functions, but when used as window functions, they calculate the aggregate over the window range rather than the entire result set.

Example:

```sql

SELECT product\_name, sales\_amount,

SUM(sales\_amount) OVER (PARTITION BY product\_category) AS category\_total\_sales,

AVG(sales\_amount) OVER () AS overall\_avg\_sales

FROM sales;

```

6. LEAD() and LAG():

LEAD() allows you to access the value from the next row within the window, while LAG() allows you to access the value from the previous row within the window.

Example:

```sql

SELECT order\_date, order\_amount,

LEAD(order\_amount) OVER (ORDER BY order\_date) AS next\_order\_amount,

LAG(order\_amount) OVER (ORDER BY order\_date) AS prev\_order\_amount

FROM orders;

```

7. FIRST\_VALUE() and LAST\_VALUE():

FIRST\_VALUE() allows you to access the first value in the window, while LAST\_VALUE() allows you to access the last value in the window.

Example:

```sql

SELECT product\_name, sales\_amount,

FIRST\_VALUE(sales\_amount) OVER (PARTITION BY product\_category) AS first\_sale,

LAST\_VALUE(sales\_amount) OVER (PARTITION BY product\_category) AS last\_sale

FROM sales;

```

These are some of the most commonly used window functions in SQL. Window functions are powerful tools that can help you perform complex calculations and analyses across multiple rows in a result set without the need for grouping the data. Understanding how to use them effectively can significantly enhance your SQL skills and enable you to solve a wide range of analytical problems.