In SQL, **`LEAD` and `LAG`** are window functions used to access data from rows that are either ahead (LEAD) or behind (LAG) the current row within the result set. These functions are particularly useful for comparing data between adjacent rows or fetching values from rows at different positions within the result set. Here's how they work:

**1. LEAD Function:**

The `LEAD` function retrieves data from the next row in the result set. You can specify the number of rows to look ahead. The basic syntax is:

```
LEAD(column_name, number_of_rows, default_value) OVER (ORDER BY ordering_column)
```

- `column_name`: The column from which you want to retrieve data.
- `number_of_rows`: The number of rows ahead to look for data.
- `default_value`: An optional default value to return if there are no more rows to look ahead.
- `ORDER BY ordering_column`: Defines the order of rows within the result set.

Example:

```
SELECT employee_name, salary, LEAD(salary, 1, 0) OVER (ORDER BY hire_date) AS next_salary
FROM employees;
```

This query retrieves the current employee's salary and the salary of the next employee (ordered by hire date). If there is no next employee, it will default to 0.

**2. LAG Function:**

The `LAG` function retrieves data from the previous row in the result set. It's similar to `LEAD` but looks behind the current row. The syntax is the same as `LEAD`.

Example:

```
SELECT employee_name, salary, LAG(salary, 1, 0) OVER (ORDER BY hire_date) AS previous_salary
FROM employees;
```

This query retrieves the current employee's salary and the salary of the previous employee (ordered by hire date). If there is no previous employee, it will default to 0.

`LEAD` and `LAG` functions are frequently used in analytical queries where you need to compare data across rows in a result set, such as calculating the difference between consecutive values, finding trends, or identifying gaps in a sequence. They are especially valuable when working with time-series data or any data with an inherent order.