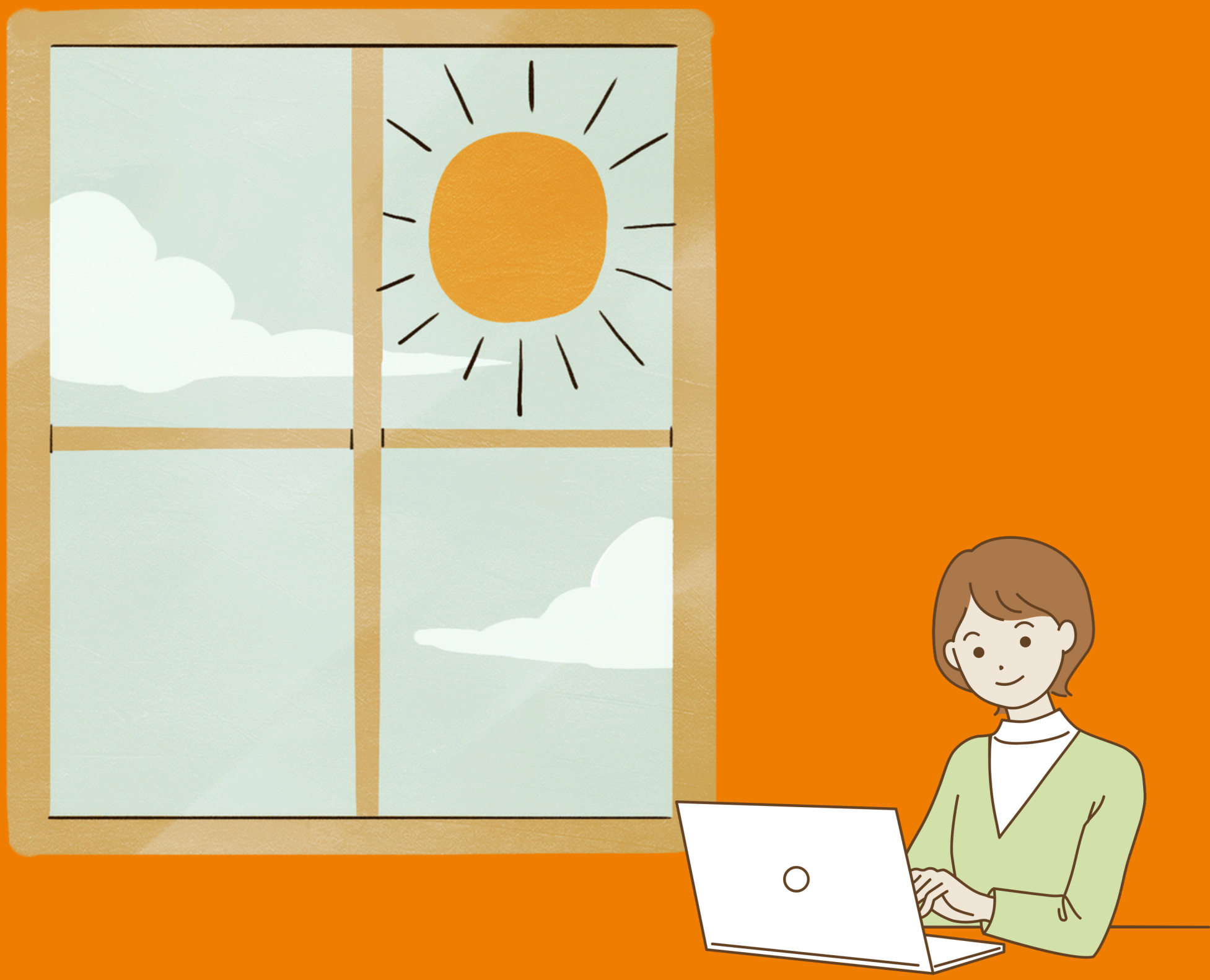
WINDOW FUNCTIONS FOR ADVANCED ANALYSIS IN SQL



WHAT ARE WINDOW FUNCTIONS?

Window functions allow you to perform calculations across a set of table rows that are related to the current row, without grouping the rows together.

Key Points:

- Operates over a "window" of rows related to the current row.
- Unlike aggregate functions, they do not reduce the number of rows returned.

WHY USE WINDOW FUNCTIONS?

Window functions offer advanced analytical capabilities:

- Running totals and cumulative sums.
- Moving averages.
- Ranking data with ROW_NUMBER(), RANK(), and DENSE_RANK().
- Calculating percentiles and ratios.

SYNTAX OF WINDOW FUNCTIONS

Basic syntax for window functions in SQL:

```
SELECT column_name,
       window_function() OVER
(PARTITION BY column ORDER BY
column)
FROM table_name;
```

- PARTITION BY: Divides the result set into partitions.
- ORDER BY: Orders rows within each partition.

ROW_NUMBER() FUNCTION

The ROW_NUMBER() function assigns a unique sequential integer to rows within a partition.

```
SELECT employee_id, department_id,
ROW_NUMBER() OVER (PARTITION
BY department_id ORDER BY salary
DESC) AS rank
FROM employees;
```

Ranks employees in each department based on their salary, with the highest salary receiving rank 1.

RANK() AND DENSE_RANK() FUNCTIONS

- RANK(): Assigns ranks with gaps if there are ties.
- DENSE_RANK(): Assigns ranks without gaps for ties.

```
SELECT employee_id, salary,
RANK() OVER (ORDER BY salary
DESC) AS rank,
DENSE_RANK() OVER (ORDER BY
salary DESC) AS dense_rank
FROM employees;
```

NTILE() FUNCTION

The NTILE() function divides the result set into a specified number of groups, assigning each row a group number.

```
SELECT employee_id, salary,
NTILE(4) OVER (ORDER BY salary
DESC) AS quartile
FROM employees;
```

This divides employees into four salary quartiles.

SWIPE

LEAD() AND LAG() FUNCTIONS

The NTILE() function divides the result set into a specified number of groups, assigning each row a group number. **SELECT employee_id, salary, NTILE(4) OVER (ORDER BY salary DESC) AS quartile FROM employees;** This divides employees into four salary quartiles.

LEAD() AND LAG() FUNCTIONS

- LEAD(): Accesses data from subsequent rows.
- LAG(): Accesses data from previous rows.

```
SELECT employee_id, hire_date,
LAG(hire_date, 1) OVER (ORDER BY
hire_date) AS previous_hire_date,
LEAD(hire_date, 1) OVER (ORDER BY
hire_date) AS next_hire_date
FROM employees;
```

CUMULATIVE SUM() EXAMPLE

The SUM() window function calculates a cumulative total. **SELECT employee_id, salary, SUM(salary) OVER (ORDER BY hire_date) AS cumulative_salary FROM employees;**

MOVING AVERAGE EXAMPLE

Use window functions to compute moving averages.

```
SELECT order_id, order_amount,
AVG(order_amount) OVER (ORDER BY
order_date ROWS BETWEEN 2
PRECEDING AND CURRENT ROW) AS
moving_avg
FROM orders;
```

Calculates a 3-period moving average of order amounts.

PRACTICAL APPLICATIONS OF WINDOW FUNCTIONS

- Analyzing employee rankings within departments. Tracking sales trends
- using moving averages. Calculating running totals for financial reports.
- Segmenting data for percentile calculations.
-

SWIPE

WHEN TO USE WINDOW FUNCTIONS?

- Use window functions when:
- You need row-level calculations while keeping the full data set.
- You want to perform advanced analytics like ranking, cumulative totals, or time-based trends.
- You want to avoid losing rows as with GROUP BY.

Now that you've explored window functions, start using them for advanced analytics and make your SQL queries more powerful!