Window Function in SQL:

1. Performs calculations across a set of table rows related to the current row.
2. Does not collapse rows like aggregate functions; retains all rows.
3. Uses `OVER()` clause to define the window (set of rows).
4. Can include `PARTITION BY` to group rows for calculation.
5. Can include `ORDER BY` to define row order within the partition.
6. Useful for ranking, running totals, percentiles, and comparisons.
7. Improves performance by avoiding subqueries or joins for row-wise operations.
8. Supported in most SQL databases like MySQL, PostgreSQL, SQL Server, and Oracle.

ROW_NUMBER()

Syntax: `ROW_NUMBER() OVER(PARTITION BY col ORDER BY col)`

Eg:

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

RANK()

Syntax: `RANK() OVER(PARTITION BY col ORDER BY col)`

Eg:

```
SELECT name, salary, RANK() OVER(ORDER BY salary DESC) AS sal_rank
FROM employees;
```

DENSE_RANK()

Syntax: `DENSE_RANK() OVER(ORDER BY col)`

Example:

```
SELECT name, salary, DENSE_RANK() OVER(ORDER BY salary DESC) AS sal_rank
FROM employees;
```

LAG(col, offset, default)

Syntax: `LAG(col, offset, default) OVER(PARTITION BY col ORDER BY col)`

Eg:

```
SELECT name, month, salary, LAG(salary, 1, 0) OVER(PARTITION BY name ORDER BY month) AS prev_salary
FROM salaries;
```

LEAD(col, offset, default)

Syntax: `LEAD(col, offset, default) OVER(PARTITION BY col ORDER BY col)`

Example:

SELECT name, month, salary, LEAD(salary, 1) OVER(PARTITION BY name ORDER BY month) AS next_salary
FROM salaries;

FIRST_VALUE(col)

Syntax: `FIRST_VALUE(col) OVER(PARTITION BY col ORDER BY col)`

Eg:

SELECT name, department, salary, FIRST_VALUE(salary) OVER(PARTITION BY department ORDER BY salary DESC) AS highest_salary
FROM employees;

LAST_VALUE(col)

Syntax: `LAST_VALUE(col) OVER(PARTITION BY col ORDER BY col ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)`

Eg:

SELECT name, department, salary, LAST_VALUE(salary) OVER(PARTITION BY department ORDER BY salary DESC ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS lowest_salary
FROM employees;

SUM(col)

Syntax: `SUM(col) OVER(PARTITION BY col ORDER BY col ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)`

Eg:

SELECT name, month, salary, SUM(salary) OVER(PARTITION BY name ORDER BY month) AS running_total
FROM salaries;

AVG(col)

Syntax: `AVG(col) OVER(PARTITION BY col ORDER BY col)`

Eg:

SELECT department, salary, AVG(salary) OVER(PARTITION BY department) AS dept_avg
FROM employees;