



Explain each window function of sql with example



Window functions in SQL are functions that perform calculations across a set of table rows that are somehow related to the current row. They differ from aggregate functions in that they do not group the results into a single output row per group. Instead, they return a value for each row in the result set.

Here's an overview of common window functions, including examples:

### 1. `ROW\_NUMBER()`

**Description:** Assigns a unique number to each row, starting at 1.

**Example:**

You've hit the Free plan limit for GPT-4o.

Responses will use our basic model until your limit resets tomorrow after 2:27 AM.

Get Plus



Get citation

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

### 2. `RANK()`

**Description:** Assigns a rank to each row within a partition, with gaps in the ranking values if there are ties.

**Example:**

```
sql Copy code

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

### 3. `DENSE\_RANK()`

**Description:** Similar to `RANK()`, but without gaps in the ranking values.

**Example:**

```
sql Copy code

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

### 4. `NTILE(n)`

**Description:** Divides the result set into `n` approximately equal groups and assigns a number to each row indicating its group.

**Example:**

```
sql Copy code

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

```
employees;
```

## 5. `LAG()``

**Description:** Provides access to a row at a given physical offset prior to the current row.

**Example:**

```
sql
SELECT
    employee_id,
    salary,
    LAG(salary, 1) OVER (ORDER BY employee_id) AS prev_salary
FROM
    employees;
```

## 6. `LEAD()``

**Description:** Provides access to a row at a given physical offset after the current row.

**Example:**

```
sql
SELECT
    employee_id,
    salary,
    LEAD(salary, 1) OVER (ORDER BY employee_id) AS next_salary
FROM
    employees;
```

## 7. `FIRST\_VALUE()``

**Description:** Returns the first value in an ordered set of values.

**Example:**

```
sql
SELECT
    employee_id,
    salary,
    FIRST_VALUE(salary) OVER (ORDER BY hire_date) AS first_salary
FROM
    employees;
```

## 8. `LAST\_VALUE()``

**Description:** Returns the last value in an ordered set of values.

**Example:**

```
sql
SELECT
    employee_id,
    salary,
    LAST_VALUE(salary) OVER (ORDER BY hire_date ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS last_salary
FROM
    employees;
```

## 9. `SUM()``

**Description:** Calculates the sum of a set of values within a window frame.

**Example:**

```
sql
SELECT
    employee_id,
    department_id,
    salary,
    SUM(salary) OVER (PARTITION BY department_id ORDER BY salary ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS dept_total
FROM
    employees;
```

```
FROM
employees;
```

## 10. `AVG()`

**Description:** Calculates the average of a set of values within a window frame.

**Example:**

```
sql Copy code

SELECT
  employee_id,
  department_id,
  salary,
  AVG(salary) OVER (PARTITION BY department_id ORDER BY salary ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS avg_salary
FROM
  employees;
```

## 11. `MIN()`



**Description:** Returns the minimum value in a set of values within a window frame.

**Example:**

```
sql Copy code

SELECT
  employee_id,
  department_id,
  salary,
  MIN(salary) OVER (PARTITION BY department_id ORDER BY salary ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS min_salary
FROM
  employees;
```

## 12. `MAX()`

**Description:** Returns the maximum value in a set of values within a window frame.

**Example:**

```
sql Copy code

SELECT
  employee_id,
  department_id,
  salary,
  MAX(salary) OVER (PARTITION BY department_id ORDER BY salary ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS max_salary
FROM
  employees;
```

## Conclusion

These window functions allow for complex calculations and analyses over subsets of your data, providing a powerful toolset for SQL users to gain deeper insights from their data without needing to perform subqueries or joins.

< 2/2 > 🔊 📄 🗨️ ⚙️

create database tables for above queries and insert some dummy records for it



Sure, I'll create the `employees` table and insert some dummy records. Here are the steps:

### 1. Create the `employees` table:

```
sql Copy code

CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
  salary INT,
  department_id INT);
```

```
last_name VARCHAR(50),
department_id INT,
salary DECIMAL(10, 2),
hire_date DATE
);
```

## 2. Insert some dummy records into the `employees` table:

```
sql Copy code

INSERT INTO employees (employee_id, first_name, last_name, department_id, salary, hire_date)
VALUES
(1, 'John', 'Doe', 1, 60000, '2018-01-15'),
(2, 'Jane', 'Smith', 1, 75000, '2019-03-23'),
(3, 'Jim', 'Brown', 1, 50000, '2017-05-12'),
(4, 'Jake', 'White', 2, 80000, '2016-11-03'),
(5, 'Jill', 'Green', 2, 90000, '2020-06-17'),
(6, 'Jerry', 'Black', 2, 85000, '2015-02-28'),
(7, 'Janet', 'Blue', 3, 72000, '2018-07-19'),
(8, 'James', 'Gray', 3, 65000, '2017-10-01'),
(9, 'Julie', 'Red', 3, 70000, '2021-08-30'),
(10, 'Jack', 'Pink', 4, 56000, '2019-12-05');
```

## Full Script

```
sql Copy code

-- Create the employees table
CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
  department_id INT,
  salary DECIMAL(10, 2),
  hire_date DATE
);

-- Insert dummy records
INSERT INTO employees (employee_id, first_name, last_name, department_id, salary, hire_date)
VALUES
(1, 'John', 'Doe', 1, 60000, '2018-01-15'),
(2, 'Jane', 'Smith', 1, 75000, '2019-03-23'),
(3, 'Jim', 'Brown', 1, 50000, '2017-05-12'),
(4, 'Jake', 'White', 2, 80000, '2016-11-03'),
(5, 'Jill', 'Green', 2, 90000, '2020-06-17'),
(6, 'Jerry', 'Black', 2, 85000, '2015-02-28'),
(7, 'Janet', 'Blue', 3, 72000, '2018-07-19'),
(8, 'James', 'Gray', 3, 65000, '2017-10-01'),
(9, 'Julie', 'Red', 3, 70000, '2021-08-30'),
(10, 'Jack', 'Pink', 4, 56000, '2019-12-05');
```

This script creates the `employees` table and inserts 10 dummy records into it. You can now use this table and data to run the window function queries provided earlier.

🔊 📄 🔄 🗑️ ⚙️

Is this conversation helpful so far?

👍 🗑️ ✕

Can't change a selection

🌟 Upgrade plan  
Get GPT-4, DALL-E, and more

🔒 Message ChatGPT



ChatGPT can make mistakes. Check important info.

?