Ranking Functions

RANK(), DENSE\_RANK(), and ROW\_NUMBER(). These functions assign ranks or row numbers to rows in a result set based on the specified ordering.

**1. RANK()**

* **Description**: Assigns a rank to rows, with ties receiving the same rank. Gaps exist in the ranking sequence after ties.
* **Syntax**:

SQL:

RANK() OVER (PARTITION BY column1, column2, ... ORDER BY column\_name [ASC|DESC])

* **Use Case**: Competitions or scenarios where ties are expected and gaps in ranking are acceptable.

**Example:**

SQL:

SELECT Name, Salary, RANK() OVER (ORDER BY Salary DESC) AS Rank

FROM Employees;

| **Name** | **Salary** | **Rank** |
| --- | --- | --- |
| John | 7000 | 1 |
| Alice | 7000 | 1 |
| Bob | 6000 | 3 |
| Charlie | 5000 | 4 |

**2. DENSE\_RANK()**

* **Description**: Similar to RANK(), but without gaps in ranking. Ties receive the same rank, but the next rank follows immediately without skipping numbers.
* **Syntax**:

SQL:

DENSE\_RANK() OVER (PARTITION BY column1, column2, ... ORDER BY column\_name [ASC|DESC])

* **Use Case**: When continuous ranking without gaps is needed, such as in leaderboards or reports.

**Example:**

**SQL:**

SELECT Name, Salary, DENSE\_RANK() OVER (ORDER BY Salary DESC) AS Rank

FROM Employees

| **Name** | **Salary** | **Rank** |
| --- | --- | --- |
| John | 7000 | 1 |
| Alice | 7000 | 1 |
| Bob | 6000 | 2 |
| Charlie | 5000 | 3 |

**3. ROW\_NUMBER()**

* **Description**: Assigns a **unique number** to each row based on the order defined in the ORDER BY clause. There are **no ties**; each row gets a distinct number, even if values are identical.
* **Syntax**:

SQL:

ROW\_NUMBER() OVER (PARTITION BY column1, column2, ... ORDER BY column\_name [ASC|DESC])

* **Use Case**: When you need to assign a unique sequential number to each row, such as for pagination or deduplication of rows.

**Example:**

SQL:

SELECT Name, Salary, ROW\_NUMBER() OVER (ORDER BY Salary DESC) AS RowNum

FROM Employees;

| **Name** | **Salary** | **RowNum** |
| --- | --- | --- |
| John | 7000 | 1 |
| Alice | 7000 | 2 |
| Bob | 6000 | 3 |
| Charlie | 5000 | 4 |

**Comparison Table of SQL: Ranking Functions**

| **Feature** | **RANK()** | **DENSE\_RANK()** | **ROW\_NUMBER()** |
| --- | --- | --- | --- |
| **Handles Ties** | Yes, assigns same rank | Yes, assigns same rank | No ties, each row is unique |
| **Gaps After Ties** | Yes, skips ranks | No, continuous ranking | No gaps, but unique numbering |
| **Ranking Order** | Defined by ORDER BY | Defined by ORDER BY | Defined by ORDER BY |
| **Partitioning** | Yes, with PARTITION BY | Yes, with PARTITION BY | Yes, with PARTITION BY |
| **Common Use Case** | Competitions | Leaderboards | Pagination, Deduplication |