Ranking functions are typically used for data analysis and reporting to determine the relative position of data points within a dataset. These functions can be very useful when you need to identify top performers, segment data, or analyze distributions. The choice of which ranking function to use depends on whether you want gaps in the ranks and the specific requirements of your analysis.

Rank functions in SQL are used to assign a rank or position to each row in a result set based on the values in one or more columns. These ranking functions help you identify the relative position of rows, which can be useful for tasks like finding top performers, ranking items, or identifying percentiles. The main ranking functions are `RANK()`, `DENSE_RANK()`, `ROW_NUMBER()`, and `NTILE()`.

1. RANK():

- The `RANK()` function assigns a unique rank to each row within the result set, with the same rank given to rows with identical values.

- If multiple rows have the same values and receive the same rank, the next rank will be skipped. This is known as "ranking with gaps."

Example:

```
SELECT employee_name, salary, RANK() OVER (ORDER BY salary DESC) AS salary_rank
FROM employees;
```

2. DENSE_RANK():

- The `DENSE_RANK()` function assigns a unique rank to each row within the result set, just like `RANK()`.

- However, unlike `RANK()`, if multiple rows have the same values, they will receive the same rank, and there are no gaps in the ranking.

Example:

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

3. ROW_NUMBER():

- The `ROW_NUMBER()` function assigns a unique rank to each row within the result set, similar to `RANK()`.

- Like `RANK()`, if multiple rows have the same values, they will receive different ranks. This function is used for assigning a unique identifier to each row.

Example:

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

4. NTILE():

- The `NTILE()` function divides the result set into a specified number of roughly equal parts (buckets or quantiles) and assigns a group or tile number to each row.

- It is often used for creating percentiles or quartiles by specifying the number of groups. For example, `NTILE(4)` would divide the data into quartiles.

Example:

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