Part - 4

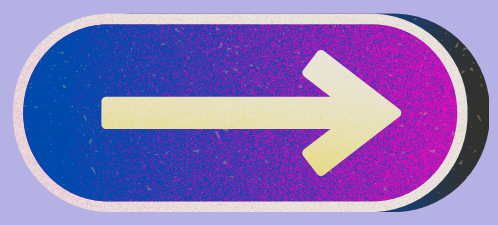
Data Analysis Interview Q & A !!!

Krishan kumar

@Krishan kumar

What are the differences between RANK, ROW_NUMBER, and DENSE_RANK in SQL?

In SQL, RANK, ROW_NUMBER, and DENSE_RANK are functions that help assign numbers to rows within a result set, but each works slightly differently.



1. RANK:

The RANK function assigns a unique rank to each distinct row within a partition of a result set. It creates gaps in the ranking sequence for rows with ties.

Syntax:

```
RANK() OVER (PARTITION BY partition_column ORDER BY order_column)
```

2. ROW_NUMBER:

- **The ROW_NUMBER function assigns a unique, sequential integer to each row within a partition, starting at 1 for the first row.**

Syntax:

```
ROW_NUMBER() OVER (PARTITION BY partition_column ORDER BY order_column)
```

3. DENSE_RANK:

➡ **The DENSE_RANK function is similar to RANK but does not create gaps in the sequence for ties.**

➡ **Syntax:**

```
DENSE_RANK() OVER (PARTITION BY partition_column ORDER BY order_column)
```

What are the Difference among them ?

Differences:

Handling Ties:

- **RANK:** **Creates gaps in the ranking sequence when rows tie. For example, if two rows share rank 1, the next rank will be 3.**

- **ROW_NUMBER:** **Does not handle ties; each row receives a unique number regardless of value.**

- **DENSE_RANK:** **No gaps in the ranking sequence for ties; if two rows are tied at rank 1, the next rank will be 2.**

I want example To understand these concepts

Example:

Consider a table Sales with columns SalesPerson and TotalSales. Here's how each function would rank the rows:

```
SELECT SalesPerson, TotalSales,
       RANK() OVER (ORDER BY TotalSales DESC) AS Rank,
       ROW_NUMBER() OVER (ORDER BY TotalSales DESC) AS RowNumber,
       DENSE_RANK() OVER (ORDER BY TotalSales DESC) AS DenseRank
FROM Sales;
```

Output:

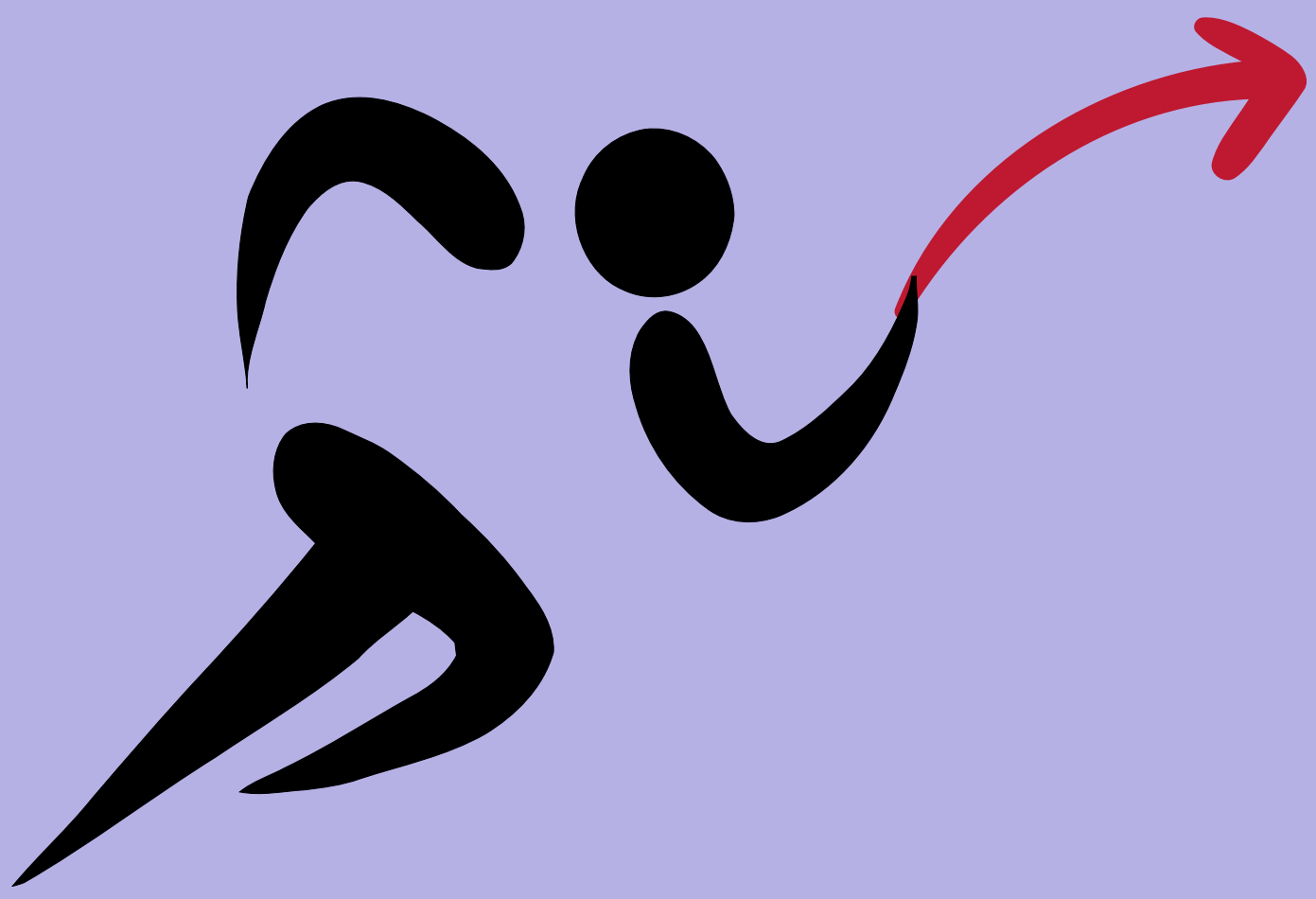
SalesPerson	TotalSales	Rank	RowNumber	DenseRank
krishna	1000	1	1	1
manish	950	2	2	2
rohan	950	2	3	2
karan	900	4	4	3

Use Cases:

- **RANK:** **Ideal for scenarios where relative standing matters, like competitions with gaps for ties.**

- **ROW_NUMBER:** **Best for generating unique row identifiers in each partition, regardless of the row values.**

- **DENSE_RANK:** **Useful when you need consecutive ranks without gaps, like prioritizing tasks in a business workflow.**



Swipe right for the jackpot of the day 😉

wanna see some

Counter Questions

1. Can you use these ranking functions without a partition?

➥ **Yes, if the PARTITION BY clause is not specified,**

The functions apply to the entire result set without partitioning.

2. When would ROW_NUMBER be more suitable than RANK or DENSE_RANK?

➡ **Use ROW_NUMBER when you need a unique identifier for each row,**

particularly in scenarios where ties are irrelevant, and you require sequential numbering.

3. Can these ranking functions improve performance?

➥ **While they don’t directly improve performance, these functions help in organizing and analyzing data efficiently,**

which can lead to optimized queries in certain use cases

4. Is DENSE_RANK faster than RANK?

➡ **Performance differences between DENSE_RANK and RANK are usually minimal,**

As both rely on similar partitioning and sorting operations. The choice depends more on the output structure needed.

5. Can you combine multiple ranking functions in a single query?

➡ **Yes, you can use RANK, ROW_NUMBER, and DENSE_RANK in the same query,**

often useful to compare their outputs for better insight.

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar