**Windows Functions**

In SQL Server, window functions (also known as analytic functions or windowing functions) are a set of functions that allow you to perform calculations across a specific subset of rows in a result set. These functions operate on a "window" or a group of rows defined by a specified windowing clause.

Unlike regular aggregate functions (e.g., SUM, AVG, COUNT), which provide a single result for a group of rows, window functions return a value for each row in the result set, based on the specified window frame.

Here are some key characteristics of window functions:

1. Calculation Scope: Window functions operate on a specified window or subset of rows within the result set. The window is defined using the ORDER BY clause, PARTITION BY clause, or both.

2. Result Per Row: Unlike aggregate functions that collapse multiple rows into a single result, window functions provide a result for each row in the result set. Each row's result is calculated based on the window defined.

3. Window Frame: The window frame determines the range of rows used for the calculation. It can be defined using various clauses, such as ROWS BETWEEN, RANGE BETWEEN, and UNBOUNDED PRECEDING/FOLLOWING.

4. Sorting: The ORDER BY clause is used to define the order of rows within the window, allowing the window function to work on the data in a specific order.

5. Partitioning: The PARTITION BY clause is used to divide the result set into logical partitions. The window function operates separately on each partition.

6. Function Types: SQL Server provides various window functions, including ranking functions (e.g., ROW\_NUMBER, RANK), aggregate functions (e.g., SUM, AVG) with the OVER clause, lead/lag functions, first\_value/last\_value functions, and more.

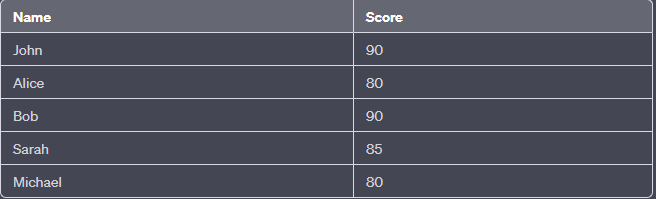
By using window functions, you can perform complex calculations and analysis within your SQL queries without the need for subqueries or temporary tables. Window functions are especially useful when you need to calculate running totals, perform comparisons with neighboring rows, or generate rankings and percentiles within your result set.

Example:

SELECT ROW\_NUMBER() OVER (ORDER BY OrderDate) AS RowNumber, OrderID, OrderDate FROM Orders

SELECT RANK() OVER (ORDER BY OrderDate) AS Rank, OrderID, OrderDate

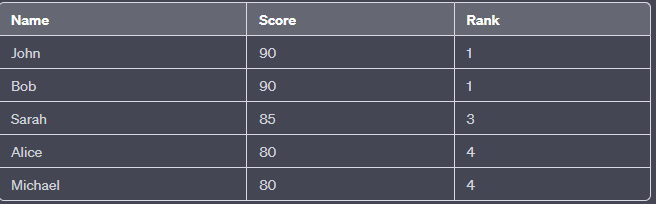
FROM Orders



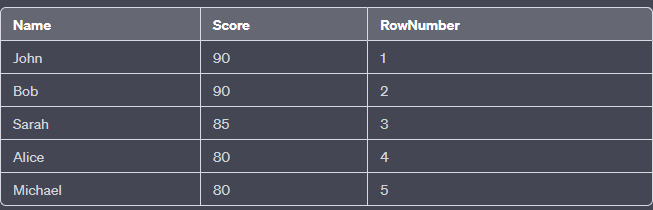
Using RANK:

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

FROM Students



Using ROW\_NUMBER:



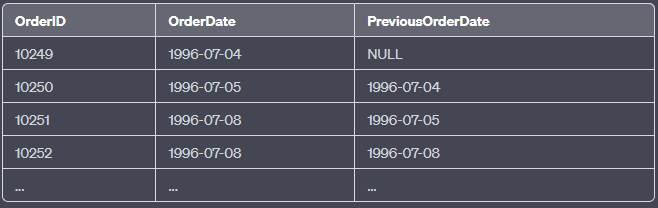
SELECT DENSE\_RANK() OVER (ORDER BY OrderDate) AS DenseRank, OrderID, OrderDate

FROM Orders

SELECT OrderID, OrderDate, LAG(OrderDate) OVER (ORDER BY OrderDate) AS PreviousOrderDate

FROM Orders

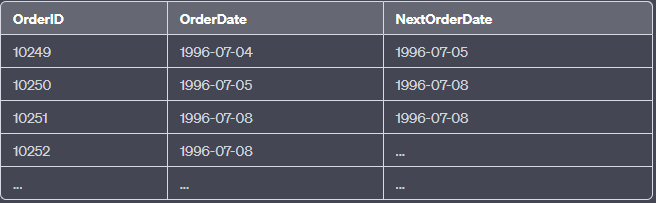
The LAG function allows you to retrieve the value of a column from the previous row, based on a specified ordering. It is often used to calculate the difference or change between consecutive rows, track historical values, or perform other calculations that involve accessing the value from a preceding row.



SELECT OrderID, OrderDate, LEAD(OrderDate) OVER (ORDER BY OrderDate) AS NextOrderDate

FROM Orders

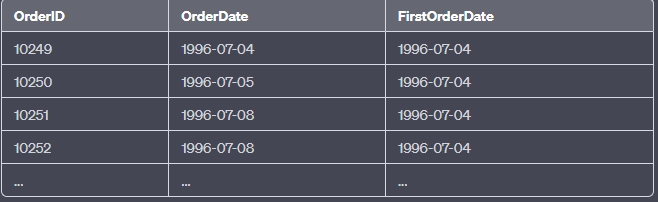
The LEAD function enables you to retrieve the value of a column from the next row, based on a specified ordering. It is often used to calculate the difference or change between consecutive rows, track future values, or perform other calculations that involve accessing the value from a following row.



SELECT OrderID, OrderDate, FIRST\_VALUE(OrderDate) OVER (ORDER BY OrderDate) AS FirstOrderDate

FROM Orders

The FIRST\_VALUE function retrieves the value from the first row in the result set, based on the specified ordering. In this case, since the ordering is done by the OrderDate, the first value of the OrderDate is repeated for all rows.



SELECT OrderID, OrderDate, LAST\_VALUE(OrderDate) OVER (ORDER BY OrderDate ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS LastOrderDate

FROM Orders

TASKS:

1. Rank the employees based on their sales amount:
2. Determine the month-wise sales growth rate for each product:
3. Rank the customers based on their total order count in each country:
4. Distribute customers into deciles(10) based on their order count
5. Divide employees into three groups based on their sales performance

| **Name** | **Description** |
| --- | --- |
| [CUME\_DIST](https://www.sqlservertutorial.net/sql-server-window-functions/sql-server-cume_dist-function/) | Calculate the cumulative distribution of a value in a set of values |
| [DENSE\_RANK](https://www.sqlservertutorial.net/sql-server-window-functions/sql-server-dense_rank-function/) | Assign a rank value to each row within a partition of a result, with no gaps in rank values. |
| [FIRST\_VALUE](https://www.sqlservertutorial.net/sql-server-window-functions/sql-server-first_value-function/) | Get the value of the first row in an ordered partition of a result set. |
| [LAG](https://www.sqlservertutorial.net/sql-server-window-functions/sql-server-lag-function/) | Provide access to a row at a given physical offset that comes before the current row. |
| [LAST\_VALUE](https://www.sqlservertutorial.net/sql-server-window-functions/sql-server-last_value-function/) | Get the value of the last row in an ordered partition of a result set. |
| [LEAD](https://www.sqlservertutorial.net/sql-server-window-functions/sql-server-lead-function/) | Provide access to a row at a given physical offset that follows the current row. |
| [NTILE](https://www.sqlservertutorial.net/sql-server-window-functions/sql-server-ntile-function/) | Distribute rows of an ordered partition into a number of groups or buckets |
| [PERCENT\_RANK](https://www.sqlservertutorial.net/sql-server-window-functions/sql-server-percent_rank-function/) | Calculate the percent rank of a value in a set of values. |
| [RANK](https://www.sqlservertutorial.net/sql-server-window-functions/sql-server-rank-function/) | Assign a rank value to each row within a partition of a result set |
| [ROW\_NUMBER](https://www.sqlservertutorial.net/sql-server-window-functions/sql-server-row_number-function/) | Assign a unique sequential integer to rows within a partition of a result set, the first row starts from 1. |
|  |  |