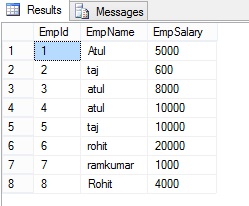
SQL FUNCTIONS

In this article we will learn about some SQL functions Row\_Number() ,Rank(), and Dense\_Rank() and the difference between them.

Creating a table in SQL Server

Here I have an Employe table, the following is the sample data for the Employe Table.



ROW\_NUMBER() Function without Partition By clause

Row\_number plays a very important role in SQL server. Row\_Number function can help to perform more complex ordering of row in the report format that allow the over clause in SQL standard.

**Syntax**

*ROW\_NUMBER () OVER ([PARTITION BY value\_exp, ... [ n ]] ORDER BY\_clause)*

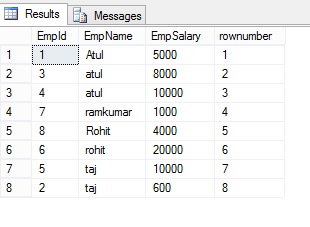
Here we will play with the above Employe table. Let's see how we can use Row\_Number() function.

Over specified the order of the row and Order by sort order for the record. By default order by sort in ascending order.

**Example**

1. **select** \*, ROW\_NUMBER() over(**order** **by** EmpName) **as** rownumber **from** Employe

The following is the output of the above query.



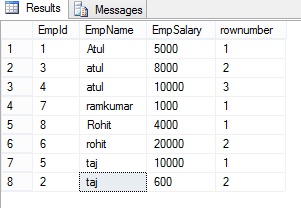
ROW\_NUMBER() Function with Partition By clause

If we want to add row number to each group, and it is reset for every group, let's take a look at the Employe table

**Example**

1. **Select** \*, ROW\_NUMBER() over(partition **by** Empname **order** **by** Empname ) **as** rownumber **from** Employe

The following is the OUTPUT of the above query



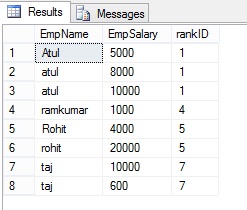
Rank() Function in SQL Server

This function will assign a unique value **to each distinct** Row, but it leaves a group between the groups**.**

**Example**

1. **SELECT** EmpName, EmpSalary ,rank() over(**order** **by** EmpName) **as** rankID **from** Employe

The following is the OUTPUT of the above query.



Gap represents number of occurrence example - **EmpName="atul"**is repeated 3 times and has **rank "1"**, the next rank will be 1+3=4 and same with the next value.

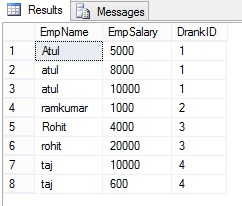
Dense\_Rank() Function in SQL Server

Dense\_Rank() Function is similar to Rank with only difference, this will not leave gaps between groups.

**Example**

1. **SELECT** EmpName ,EmpSalary ,DENSE\_RANK() over(**order** **by** EmpName) **as** DrankID **from** Employe

The following is the OUTPUT of the above query:



**ROW\_NUMBER:** Returns the sequence and unique number for each group based on the fields applied in PARTITION BY clause. If PARTITION BY is not specified, the function treats all rows of the query result set as a single group.  
  
**RANK:** Similar to ROW\_NUMBER function and Returns the rank of each row within the partition of a result set. The rank of a row is one plus the number of ranks that come before the row in question.  
  
**DENSE\_RANK:** Returns the rank of rows within the partition of a result set, without any gaps in the ranking. The rank of a row is one plus the number of distinct ranks that come before the row in question.

**NTILE:** Distributes the rows in an ordered partition into a specified number of groups. The groups are numbered, starting at one. For each row, NTILE returns the number of the group to which the row belongs.

Example

select \* from EmployeSalary

O/P

empid project salary

121 p1 120000.00

122 p1 320000.00

123 p2 220000.00

124 p2 420000.00

125 p3 920000.00

126 p3 190000.00

127 p3 12333.00

128 p3 NULL

129 p1 120000.00

SELECT \*

,ROW\_NUMBER() OVER (ORDER BY project) AS "Row Number"

,RANK() OVER (ORDER BY project) AS Rank

,DENSE\_RANK() OVER (ORDER BY project) AS "Dense Rank"

,NTILE(4) OVER (ORDER BY project) AS Quartile

from EmployeSalary

O/P

**empid project salary Row Number Rank Dense Rank Quartile**

121 p1 120000.00 1 1 1 1

122 p1 320000.00 2 1 1 1

129 p1 120000.00 3 1 1 1

123 p2 220000.00 4 4 2 2

124 p2 420000.00 5 4 2 2

125 p3 920000.00 6 6 3 3

126 p3 190000.00 7 6 3 3

127 p3 12333.00 8 6 3 4

128 p3 NULL 9 6 3 4

SELECT \*

,ROW\_NUMBER() OVER (ORDER BY salary) AS "Row Number"

,RANK() OVER (ORDER BY salary) AS Rank

,DENSE\_RANK() OVER (ORDER BY salary) AS "Dense Rank"

,NTILE(4) OVER (ORDER BY salary) AS Quartile

from EmployeSalary

empid project salary Row Number Rank Dense Rank Quartile

128 p3 NULL 1 1 1 1

127 p3 12333.00 2 2 2 1

129 p1 120000.00 3 3 3 1

121 p1 120000.00 4 3 3 2

126 p3 190000.00 5 5 4 2

123 p2 220000.00 6 6 5 3

122 p1 320000.00 7 7 6 3

124 p2 420000.00 8 8 7 4

125 p3 920000.00 9 9 8 4

SELECT \*

,ROW\_NUMBER() OVER (ORDER BY salary desc) AS "Row Number"

,RANK() OVER (ORDER BY salary desc) AS Rank

,DENSE\_RANK() OVER (ORDER BY salary desc) AS "Dense Rank"

,NTILE(4) OVER (ORDER BY salary desc) AS Quartile

from EmployeSalary

empid project salary Row Number Rank Dense Rank Quartile

125 p3 920000.00 1 1 1 1

124 p2 420000.00 2 2 2 1

122 p1 320000.00 3 3 3 1

123 p2 220000.00 4 4 4 2

126 p3 190000.00 5 5 5 2

121 p1 120000.00 6 6 6 3

129 p1 120000.00 7 6 6 3

127 p3 12333.00 8 8 7 4

128 p3 NULL 9 9 8 4 4

Example from project valueline

select top 15 vlid, RecordType,value, DENSE\_RANK() over(order by vlid)as "dense rank"

,RANK()over (order by vlid)as "rank"

,ROW\_NUMBER() over (order by vlid) as"rowno"

from Financial\_Calculateds\_TANGO

vlid RecordType value dense rank rank rowno

0 AnnualVLEST 12.36 % 1 1 1

0 AnnualVLPROJ 9.88 % 1 1 2

69 AnnualVLActual 69.57 % 2 3 3

138 AnnualVLActual 29.28 % 3 4 4

138 AnnualVLEST 61.11 % 3 4 5

138 AnnualVLEST 43.70 % 3 4 6

138 AnnualVLPROJ 45.33 % 3 4 7

219 AnnualVLActual 20.45 % 4 8 8

219 AnnualVLActual 16.47 % 4 8 9

219 AnnualVLActual 28.19 % 4 8 10

219 AnnualVLEST 26.67 % 4 8 11

219 AnnualVLEST 27.41 % 4 8 12

219 AnnualVLPROJ 24.53 % 4 8 13

230 AnnualVLActual 68.18 % 5 14 14

230 AnnualVLActual 105.26 % 5 14 15

From dense rank that is observed there is 5 vlid total from 15 rows