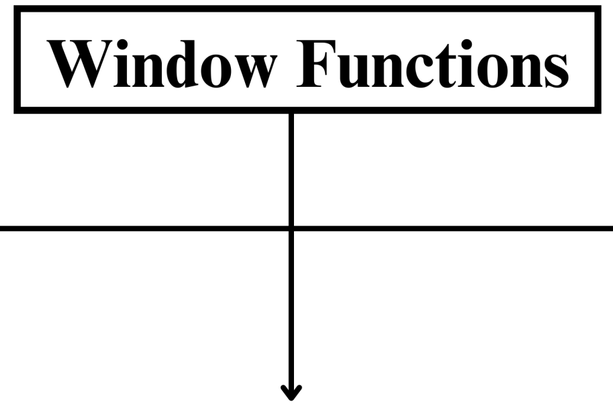
Window Function in SQL

Window function in SQL is known as analytic function which uses values from one or
multiple rows to return a value for each row. This contrasts with aggregate function, which returns a single value for multiple rows

Syntax:

```
window_function() OVER (
    [PARTITION BY partition_expression]
    [ORDER BY sort_expression])
```



Aggregate Function

- AVG( )
- MAX( )
- MIN ( )
- SUM ( )
- COUNT ( )

Ranking Function

- ROW_NUMBER( )
- RANK( )
- DENSE_RANK ( )
- PERCENT_RANK( )
- NTILE( )

Value Function

- LAG( )
- LEAD( )
- FISRT_VALUE( )
- LAST_VALUE( )
- NTH_VALUE( )

Aggregate Function:

```
select *,
avg(Salary) over() as "Average Salary"
from employee ;
```

emp_ID	emp_NAME	DEPT_NAME	SALARY	Average Salary
101	Mohan	Admin	4000	5791.6667
102	Rajkumar	HR	3000	5791.6667
103	Akbar	IT	4000	5791.6667
104	Dorvin	Finance	6500	5791.6667
105	Rohit	HR	3000	5791.6667
106	Rajesh	Finance	5000	5791.6667
107	Preet	HR	7000	5791.6667
108	Maryam	Admin	4000	5791.6667

Above we can see window function is giving average salary for each row

```
select *,
avg(Salary) over(partition by dept_name ) as
"Average_Salary" from employee ;
```

emp_ID	emp_NAME	DEPT_NAME	SALARY	Average_Salary
101	Mohan	Admin	4000	3750.0000
108	Maryam	Admin	4000	3750.0000
113	Gautham	Admin	2000	3750.0000
120	Monica	Admin	5000	3750.0000
104	Dorvin	Finance	6500	5875.0000
106	Rajesh	Finance	5000	5875.0000
116	Satya	Finance	6500	5875.0000
118	Tejaswi	Finance	5500	5875.0000
102	Rajkumar	HR	3000	4583.3333
105	Rohit	HR	3000	4583.3333
107	Preet	HR	7000	4583.3333
114	Manisha	HR	3000	4583.3333
117	Adarsh	HR	3500	4583.3333
119	Cory	HR	8000	4583.3333
103	Akbar	IT	4000	7300.0000
109	Sanjay	IT	6500	7300.0000
110	Vasudha	IT	7000	7300.0000
111	Melinda	IT	8000	7300.0000
112	Komal	IT	10000	7300.0000
115	Chandni	IT	4500	7300.0000
121	Rosalin	IT	6000	7300.0000
122	Ibrahim	IT	8000	7300.0000
123	Vikram	IT	8000	7300.0000

Above we can see window function is giving average salary department wise

Note:
We can use other aggregate function in same way as we used average function here.

Ranking Function:

ROW_NUMBER( ):

It assigns consecutive numbers starting from 1 to all rows in the table.

Syntax:

```
select *,
row_number() over(order by salary) as
'Index' from employee ;
```

emp_ID	emp_NAME	DEPT_NAME	SALARY	Index
113	Gautham	Admin	2000	1
102	Rajkumar	HR	3000	2
105	Rohit	HR	3000	3
114	Manisha	HR	3000	4
117	Adarsh	HR	3500	5
101	Mohan	Admin	4000	6
103	Akbar	IT	4000	7
108	Maryam	Admin	4000	8
115	Chandni	IT	4500	9
106	Rajesh	Finance	5000	10
120	Monica	Admin	5000	11
118	Tejaswi	Finance	5500	12
121	Rosalin	IT	6000	13
104	Dorvin	Finance	6500	14
109	Sanjay	IT	6500	15
116	Satya	Finance	6500	16
107	Preet	HR	7000	17
110	Vasudha	IT	7000	18
111	Melinda	IT	8000	19
119	Cory	HR	8000	20
122	Ibrahim	IT	8000	21
123	Vikram	IT	8000	22
112	Komal	IT	10000	23
124	Dheeraj	IT	11000	24

Ranking Function:

RANK ( ):

The RANK() function in SQL gives a position number to each row in a group. If two or more rows have the same value, they get the same number, and the next row's number skips ahead. For example, if three rows are ranked 2, the next row will be ranked 5, not 3.

Syntax:

```
select *,
row_number() over(order by salary)
as 'Index' ,
rank() over(order by salary) as
'Rank'
from employee ;
```

emp_ID	emp_NAME	DEPT_NAME	SALARY	Index	Rank
113	Gautham	Admin	2000	1	1
102	Rajkumar	HR	3000	2	2
105	Rohit	HR	3000	3	2
114	Manisha	HR	3000	4	2
117	Adarsh	HR	3500	5	5
101	Mohan	Admin	4000	6	6
103	Akbar	IT	4000	7	6
108	Maryam	Admin	4000	8	6
115	Chandni	IT	4500	9	9
106	Rajesh	Finance	5000	10	10
120	Monica	Admin	5000	11	10
118	Tejaswi	Finance	5500	12	12
121	Rosalin	IT	6000	13	13
104	Dorvin	Finance	6500	14	14
109	Sanjay	IT	6500	15	14
116	Satya	Finance	6500	16	14
107	Preet	HR	7000	17	17
110	Vasudha	IT	7000	18	17
111	Melinda	IT	8000	19	19
119	Cory	HR	8000	20	19
122	Ibrahim	IT	8000	21	19
123	Vikram	IT	8000	22	19
112	Komal	IT	10000	23	23
124	Dheeraj	IT	11000	24	24

As we can see here rank 2 is assigned to 3 employees so it skips 3,4 & provide next rank as 5

Ranking Function:

DENSE_RANK ( ):

The DENSE_RANK() function in SQL gives a position number to each row in a group. If two or more rows have the same value, they get the same number, but the next row's number is the next in the sequence. For example, if three rows are ranked 2, the next row will be ranked 3.

Syntax:

```
select *,
row_number() over(order by salary  ) as
'Index' ,
rank() over(order by salary) as 'Rank',
dense_rank() over(order by salary) as
'Dense Rank'
 from employee ;
```

emp_ID	emp_NAME	DEPT_NAME	SALARY	Index	Rank	Dense Rank
113	Gautham	Admin	2000	1	1	1
102	Rajkumar	HR	3000	2	2	2
105	Rohit	HR	3000	3	2	2
114	Manisha	HR	3000	4	2	2
117	Adarsh	HR	3500	5	5	3
101	Mohan	Admin	4000	6	6	4
103	Akbar	IT	4000	7	6	4
108	Maryam	Admin	4000	8	6	4
115	Chandni	IT	4500	9	9	5
106	Rajesh	Finance	5000	10	10	6
120	Monica	Admin	5000	11	10	6
118	Tejaswi	Finance	5500	12	12	7
121	Rosalin	IT	6000	13	13	8
104	Dorvin	Finance	6500	14	14	9
109	Sanjay	IT	6500	15	14	9
116	Satya	Finance	6500	16	14	9
107	Preet	HR	7000	17	17	10
110	Vasudha	IT	7000	18	17	10
111	Melinda	IT	8000	19	19	11
119	Cory	HR	8000	20	19	11
122	Ibrahim	IT	8000	21	19	11
123	Vikram	IT	8000	22	19	11
112	Komal	IT	10000	23	23	12
124	Dheeraj	IT	11000	24	24	13

As we can see here rank 2 is assigned to 3 employees so in case of dense_rank it doesn’t skips 3,4 & provide next rank as 3

Value Function:

LAG ( ):

The LAG() function in SQL looks at the value in the previous row and puts it in the current row. It's like looking one step back. For example, if you have a list of numbers, it can show what the number was right before each one. This helps compare values that are next to each other.

```
select * ,
lag(salary) over () as 'Lag Value'
from employee;
```

emp_ID	emp_NAME	DEPT_NAME	SALARY	Lag Value
101	Mohan	Admin	4000	NULL
102	Rajkumar	HR	3000	4000
103	Akbar	IT	4000	3000
104	Dorvin	Finance	6500	4000
105	Rohit	HR	3000	6500
106	Rajesh	Finance	5000	3000
107	Preet	HR	7000	5000
108	Maryam	Admin	4000	7000
109	Sanjay	IT	6500	4000
110	Vasudha	IT	7000	6500
111	Melinda	IT	8000	7000
112	Komal	IT	10000	8000
113	Gautham	Admin	2000	10000
114	Manisha	HR	3000	2000
115	Chandni	IT	4500	3000
116	Satya	Finance	6500	4500
117	Adarsh	HR	3500	6500
118	Tejaswi	Finance	5500	3500
119	Cory	HR	8000	5500
120	Monica	Admin	5000	8000
121	Rosalin	IT	6000	5000
122	Ibrahim	IT	8000	6000
123	Vikram	IT	8000	8000
124	Dheeraj	IT	11000	8000

Value lagged by one row.

Value Function:

LEAD ( ):

The LEAD() function in SQL looks at the value in the next row and puts it in the current row. It's like looking one step ahead. For example, if you have a list of numbers, it can show what the number will be right after each one. This helps compare values that are next to each other.

```
select * ,
lead(salary) over () as 'Lead Value'
from employee;
```

emp_ID	emp_NAME	DEPT_NAME	SALARY	Lead Value
101	Mohan	Admin	4000	3000
102	Rajkumar	HR	3000	4000
103	Akbar	IT	4000	6500
104	Dorvin	Finance	6500	3000
105	Rohit	HR	3000	5000
106	Rajesh	Finance	5000	7000
107	Preet	HR	7000	4000
108	Maryam	Admin	4000	6500
109	Sanjay	IT	6500	7000
110	Vasudha	IT	7000	8000
111	Melinda	IT	8000	10000
112	Komal	IT	10000	2000
113	Gautham	Admin	2000	3000
114	Manisha	HR	3000	4500
115	Chandni	IT	4500	6500
116	Satya	Finance	6500	3500
117	Adarsh	HR	3500	5500
118	Tejaswi	Finance	5500	8000
119	Cory	HR	8000	5000
120	Monica	Admin	5000	6000
121	Rosalin	IT	6000	8000
122	Ibrahim	IT	8000	8000
123	Vikram	IT	8000	11000
124	Dheeraj	IT	11000	NULL

Value leaded by one row.

Value Function:

FIRST_VALUE ( ):

The FIRST_VALUE() function in SQL is like looking at a line of people and figuring out who was at the very front of the line. It tells you the first person in the line based on some order we set up, like who came first in time or alphabetically.

```
select * ,
first_value(emp_NAME) over (order by
emp_NAME asc) as 'First Value'
from employee;
```

emp_ID	emp_NAME	DEPT_NAME	SALARY	First Value
117	Adarsh	HR	3500	Adarsh
103	Akbar	IT	4000	Adarsh
115	Chandni	IT	4500	Adarsh
119	Cory	HR	8000	Adarsh
124	Dheeraj	IT	11000	Adarsh
104	Dorvin	Finance	6500	Adarsh
113	Gautham	Admin	2000	Adarsh
122	Ibrahim	IT	8000	Adarsh
112	Komal	IT	10000	Adarsh
114	Manisha	HR	3000	Adarsh
108	Maryam	Admin	4000	Adarsh
111	Melinda	IT	8000	Adarsh
101	Mohan	Admin	4000	Adarsh
120	Monica	Admin	5000	Adarsh
107	Preet	HR	7000	Adarsh
106	Rajesh	Finance	5000	Adarsh
102	Rajkumar	HR	3000	Adarsh
105	Rohit	HR	3000	Adarsh
121	Rosalin	IT	6000	Adarsh
109	Sanjay	IT	6500	Adarsh
116	Satya	Finance	6500	Adarsh
118	Tejaswi	Finance	5500	Adarsh
110	Vasudha	IT	7000	Adarsh
123	Vikram	IT	8000	Adarsh

Value Function:

LAST_VALUE ( ):

LAST_VALUE() in SQL is like looking at a line of people and figuring out who was at the very end of the line. It tells you the last person in the line based on some order we set up, like who came last in time or alphabetically.

```
select * ,
last_value(emp_NAME) over ( ) as 'Last
Value'
from employee;
```

emp_ID	emp_NAME	DEPT_NAME	SALARY	First Value
101	Mohan	Admin	4000	Dheeraj
102	Rajkumar	HR	3000	Dheeraj
103	Akbar	IT	4000	Dheeraj
104	Dorvin	Finance	6500	Dheeraj
105	Rohit	HR	3000	Dheeraj
106	Rajesh	Finance	5000	Dheeraj
107	Preet	HR	7000	Dheeraj
108	Maryam	Admin	4000	Dheeraj
109	Sanjay	IT	6500	Dheeraj
110	Vasudha	IT	7000	Dheeraj
111	Melinda	IT	8000	Dheeraj
112	Komal	IT	10000	Dheeraj
113	Gautham	Admin	2000	Dheeraj
114	Manisha	HR	3000	Dheeraj
115	Chandni	IT	4500	Dheeraj
116	Satya	Finance	6500	Dheeraj
117	Adarsh	HR	3500	Dheeraj
118	Tejaswi	Finance	5500	Dheeraj
119	Cory	HR	8000	Dheeraj
120	Monica	Admin	5000	Dheeraj
121	Rosalin	IT	6000	Dheeraj
122	Ibrahim	IT	8000	Dheeraj
123	Vikram	IT	8000	Dheeraj
124	Dheeraj	IT	11000	Dheeraj