 # Window Functions

Window functions apply to aggregate and ranking functions over a particular window (set of rows).

```
select * from sales_sample_data
```

121 %

Results | Messages

	cust_id	region	cust_name	month	sales
1	1	North	Aman	2024-01-01	1000
2	2	South	Rahul	2024-01-01	1500
3	3	North	Aman	2024-02-01	1200
4	4	South	Rahul	2024-02-01	1300
5	5	North	Aman	2024-03-01	1100
6	6	South	Rahul	2024-03-01	1400
7	7	North	Aditya	2024-04-01	1300
8	8	South	Sahil	2024-04-01	1400

1) **Row Number** - Assign a unique number to rows.
   Scenario 1: You want to assign a unique rank to each sales record based on the month.

```
select *,
row_number() over(order by month) as row_num
from sales_sample_data
```

121 %

Results | Messages

	cust_id	region	cust_name	month	sales	row_num
1	1	North	Aman	2024-01-01	1000	1
2	2	South	Rahul	2024-01-01	1500	2
3	3	North	Aman	2024-02-01	1200	3
4	4	South	Rahul	2024-02-01	1300	4
5	5	North	Aman	2024-03-01	1100	5
6	6	South	Rahul	2024-03-01	1400	6
7	7	North	Aditya	2024-04-01	1300	7
8	8	South	Sahil	2024-04-01	1400	8

suraj67.sk@gmail.com

Scenario 2: You want to assign a unique rank to each monthly sale.

```
select *,
row_number() over(partition by month order by sales desc) as row_num
from sales_sample_data
```

121 %

Results | Messages

	cust_id	region	cust_name	month	sales	row_num
1	2	South	Rahul	2024-01-01	1500	1
2	1	North	Aman	2024-01-01	1000	2
3	4	South	Rahul	2024-02-01	1300	1
4	3	North	Aman	2024-02-01	1200	2
5	6	South	Rahul	2024-03-01	1400	1
6	5	North	Aman	2024-03-01	1100	2
7	8	South	Sahil	2024-04-01	1400	1
8	7	North	Aditya	2024-04-01	1300	2

2) **Rank()** - Assign a rank to rows, with gaps for ties.
Scenario 1: rank each sales record based on the month.

```
select *,
rank() over(order by sales desc) as sales_rank
from sales_sample_data
```

121 %

Results | Messages

	cust_id	region	cust_name	month	sales	sales_rank
1	2	South	Rahul	2024-01-01	1500	1
2	6	South	Rahul	2024-03-01	1400	2
3	8	South	Sahil	2024-04-01	1400	2
4	7	North	Aditya	2024-04-01	1300	4
5	4	South	Rahul	2024-02-01	1300	4
6	3	North	Aman	2024-02-01	1200	6
7	5	North	Aman	2024-03-01	1100	7
8	1	North	Aman	2024-01-01	1000	8

suraj67.sk@gmail.com

Scenario 2: Rank customers based on their sales per month.

```
select *,
rank() over(partition by cust_name order by sales desc) as sales_rank
from sales_sample_data
```

121 %

Results | Messages

	cust_id	region	cust_name	month	sales	sales_rank
1	7	North	Aditya	2024-04-01	1300	1
2	3	North	Aman	2024-02-01	1200	1
3	5	North	Aman	2024-03-01	1100	2
4	1	North	Aman	2024-01-01	1000	3
5	2	South	Rahul	2024-01-01	1500	1
6	6	South	Rahul	2024-03-01	1400	2
7	4	South	Rahul	2024-02-01	1300	3
8	8	South	Sahil	2024-04-01	1400	1

3) **DENSE_RANK()** – Assign a rank to rows, without gaps for ties.

```
select *,
dense_rank() over(order by sales desc) as sales_rank
from sales_sample_data
```

121 %

Results | Messages

	cust_id	region	cust_name	month	sales	sales_rank
1	2	South	Rahul	2024-01-01	1500	1
2	6	South	Rahul	2024-03-01	1400	2
3	8	South	Sahil	2024-04-01	1400	2
4	7	North	Aditya	2024-04-01	1300	3
5	4	South	Rahul	2024-02-01	1300	3
6	3	North	Aman	2024-02-01	1200	4
7	5	North	Aman	2024-03-01	1100	5
8	1	North	Aman	2024-01-01	1000	6

suraj67.sk@gmail.com

4) **LAG()** – Access the previous row's data.
Scenario 1: Compare the current month's sales with the previous month's sales.

```
select *,
lag(sales,1) over(order by sales desc) as prev_sales
from sales_sample_data
```

121 %

Results | Messages

	cust_id	region	cust_name	month	sales	prev_sales
1	2	South	Rahul	2024-01-01	1500	NULL
2	6	South	Rahul	2024-03-01	1400	1500
3	8	South	Sahil	2024-04-01	1400	1400
4	7	North	Aditya	2024-04-01	1300	1400
5	4	South	Rahul	2024-02-01	1300	1300
6	3	North	Aman	2024-02-01	1200	1300
7	5	North	Aman	2024-03-01	1100	1200
8	1	North	Aman	2024-01-01	1000	1100

Scenario 2: Compare the Customer-wise current month's sales with the previous month's sales.

```
select *,
lag(sales,1) over(partition by cust_name order by sales desc) as prev_sales,
sales - cast(lag(sales,1) over(partition by cust_name order by sales desc) as float) as sales_diff
from sales_sample_data
```

121 %

Results | Messages

	cust_id	region	cust_name	month	sales	prev_sales	sales_diff
1	7	North	Aditya	2024-04-01	1300	NULL	NULL
2	3	North	Aman	2024-02-01	1200	NULL	NULL
3	5	North	Aman	2024-03-01	1100	1200	-100
4	1	North	Aman	2024-01-01	1000	1100	-100
5	2	South	Rahul	2024-01-01	1500	NULL	NULL
6	6	South	Rahul	2024-03-01	1400	1500	-100
7	4	South	Rahul	2024-02-01	1300	1400	-100
8	8	South	Sahil	2024-04-01	1400	NULL	NULL

suraj67.sk@gmail.com

5) **LEAD()** – Access the next row's data.
Scenario: Compare the current month's sales to the next month's sales to track changes.

```
select *,
lead(sales,1) over(order by sales desc) as next_month_sales
from sales_sample_data
```

121 %

Results | Messages

	cust_id	region	cust_name	month	sales	next_month_sales
1	2	South	Rahul	2024-01-01	1500	1400
2	6	South	Rahul	2024-03-01	1400	1400
3	8	South	Sahil	2024-04-01	1400	1300
4	7	North	Aditya	2024-04-01	1300	1300
5	4	South	Rahul	2024-02-01	1300	1200
6	3	North	Aman	2024-02-01	1200	1100
7	5	North	Aman	2024-03-01	1100	1000
8	1	North	Aman	2024-01-01	1000	NULL

6) **SUM() OVER()** – Calculate a running total or cumulative sum.
Scenario: You want to calculate the cumulative sales by each Customer.

```
select *,
sum(cast(sales as float)) over(partition by cust_name order by sales desc) as cumulative_sales
from sales_sample_data
```

121 %

Results | Messages

	cust_id	region	cust_name	month	sales	cumulative_sales
1	7	North	Aditya	2024-04-01	1300	1300
2	3	North	Aman	2024-02-01	1200	1200
3	5	North	Aman	2024-03-01	1100	2300
4	1	North	Aman	2024-01-01	1000	3300
5	2	South	Rahul	2024-01-01	1500	1500
6	6	South	Rahul	2024-03-01	1400	2900
7	4	South	Rahul	2024-02-01	1300	4200
8	8	South	Sahil	2024-04-01	1400	1400

suraj67.sk@gmail.com

7) **AVG() OVER()** – Calculate an Avg sum.
Scenario: You want to calculate the average sales by each Customer.

```
select *,
avg(cast(sales as float)) over(partition by cust_name order by sales desc) as avg_sales
from sales_sample_data
```

121 %

Results | Messages

	cust_id	region	cust_name	month	sales	avg_sales
1	7	North	Aditya	2024-04-01	1300	1300
2	3	North	Aman	2024-02-01	1200	1200
3	5	North	Aman	2024-03-01	1100	1150
4	1	North	Aman	2024-01-01	1000	1100
5	2	South	Rahul	2024-01-01	1500	1500
6	6	South	Rahul	2024-03-01	1400	1450
7	4	South	Rahul	2024-02-01	1300	1400
8	8	South	Sahil	2024-04-01	1400	1400

8) **NTILE()** – Divide rows into a specified number of buckets.
Scenario: Split sales into 3 performance groups based on sales.

```
select *,
ntile(3) over(order by sales desc) as performance_groups
from sales_sample_data
```

121 %

Results | Messages

	cust_id	region	cust_name	month	sales	performance_groups
1	2	South	Rahul	2024-01-01	1500	1
2	6	South	Rahul	2024-03-01	1400	1
3	8	South	Sahil	2024-04-01	1400	1
4	7	North	Aditya	2024-04-01	1300	2
5	4	South	Rahul	2024-02-01	1300	2
6	3	North	Aman	2024-02-01	1200	2
7	5	North	Aman	2024-03-01	1100	3
8	1	North	Aman	2024-01-01	1000	3

suraj67.sk@gmail.com