Windows Functions in SQL

By Person Date

📌 What are Window Functions?

- A **Window Function** performs a calculation **across a set of rows** that are somehow related to the current row.
- Unlike GROUP BY, window functions **do not collapse rows**.
- Syntax uses OVER (…) clause → defines the "window" of rows.

🔹 Common Window Functions in Oracle SQL

1. **Ranking Functions**
   - ROW_NUMBER()
   - RANK()
   - DENSE_RANK()
   - NTILE()
2. **Aggregate Functions as Window Functions**
   - SUM(), AVG(), COUNT(), MIN(), MAX() with OVER()
3. **Analytic Functions**
   - LEAD() and LAG() → look ahead/behind
   - FIRST_VALUE() and LAST_VALUE()

🏢 Domain Example: Sales Data

Imagine a **Retail Sales** table:

```
CREATE TABLE sales (
  sale_id   NUMBER,
  region    VARCHAR2(20),
  product   VARCHAR2(20),
  amount    NUMBER
);
INSERT INTO sales VALUES (1, 'North', 'Laptop', 50000);
INSERT INTO sales VALUES (2, 'North', 'Mobile', 30000);
INSERT INTO sales VALUES (3, 'North', 'Tablet', 20000);
INSERT INTO sales VALUES (4, 'South', 'Laptop', 40000);
INSERT INTO sales VALUES (5, 'South', 'Mobile', 25000);
INSERT INTO sales VALUES (6, 'South', 'Tablet', 15000);
```

🔹 Examples of Window Functions

1. ROW_NUMBER()

👉 Assigns a unique sequence number within each region.

```
SELECT region, product, amount,
```

```
    ROW_NUMBER() OVER (PARTITION BY region ORDER BY amount DESC) AS row_num
FROM sales;
```

✅ Output:

REGION	PRODUCT	AMOUNT	ROW_NUM
North	Laptop	50000	1
North	Mobile	30000	2
North	Tablet	20000	3
South	Laptop	40000	1
South	Mobile	25000	2
South	Tablet	15000	3

2. RANK() vs DENSE_RANK()

👉 Ranking based on sales within region.

```
SELECT region, product, amount,
    RANK() OVER (PARTITION BY region ORDER BY amount DESC) AS rank,
    DENSE_RANK() OVER (PARTITION BY region ORDER BY amount DESC) AS dense_rank
FROM sales;
```

- RANK() skips numbers if there are ties.
- DENSE_RANK() does not skip.

3. SUM() as Window Function

👉 Cumulative sales per region.

```
SELECT region, product, amount,
    SUM(amount) OVER (PARTITION BY region ORDER BY amount DESC) AS running_total
FROM sales;
```

✅ Output (North region):

- Laptop → 50000
    - Mobile → 80000
    - Tablet → 100000

4. LEAD() and LAG()

👉 Compare sales with next/previous product in the same region.

```
SELECT region, product, amount,
    LAG(amount, 1) OVER (PARTITION BY region ORDER BY amount DESC) AS prev_amount,
    LEAD(amount, 1) OVER (PARTITION BY region ORDER BY amount DESC) AS next_amount
FROM sales;
```

✅ Output (North region):

- Laptop (50000, prev=null, next=30000)
- Mobile (30000, prev=50000, next=20000)
- Tablet (20000, prev=30000, next=null)

5. NTILE(3)

👉 Divide products in each region into 3 performance buckets.

```
SELECT region, product, amount,
    NTILE(3) OVER (PARTITION BY region ORDER BY amount DESC) AS performance_bucket
FROM sales;
```

✅ Output (North region):

- Laptop → Bucket 1 (top performers)
- Mobile → Bucket 2
- Tablet → Bucket 3 (low performers)

🎯 Why Use Window Functions?

- In **Retail/E-commerce** → Top 5 selling products per region.
- In **Banking** → Customer transaction history (previous/next transaction).
- In **HR** → Employee salary rankings.
- In **Telecom** → Usage trends per subscriber.

✅ Key Takeaways

1. Window functions work with OVER() clause.
2. They **do not group rows**, they return **row + extra calculation**.
3. Use PARTITION BY (like GROUP BY per group).
4. Use ORDER BY to control ranking or running totals.