Varun Sagar Theegala

3 SQL FUNCTIONS FOR

DATA DEDUPLICATION AND RANKING

FOR

DATA ANALYTICS & REPORTING

In data analytics and business reporting, duplicate records can cause issues in analysis and reporting. Properly ranking records helps in identifying unique entries and determining their relative importance or order.

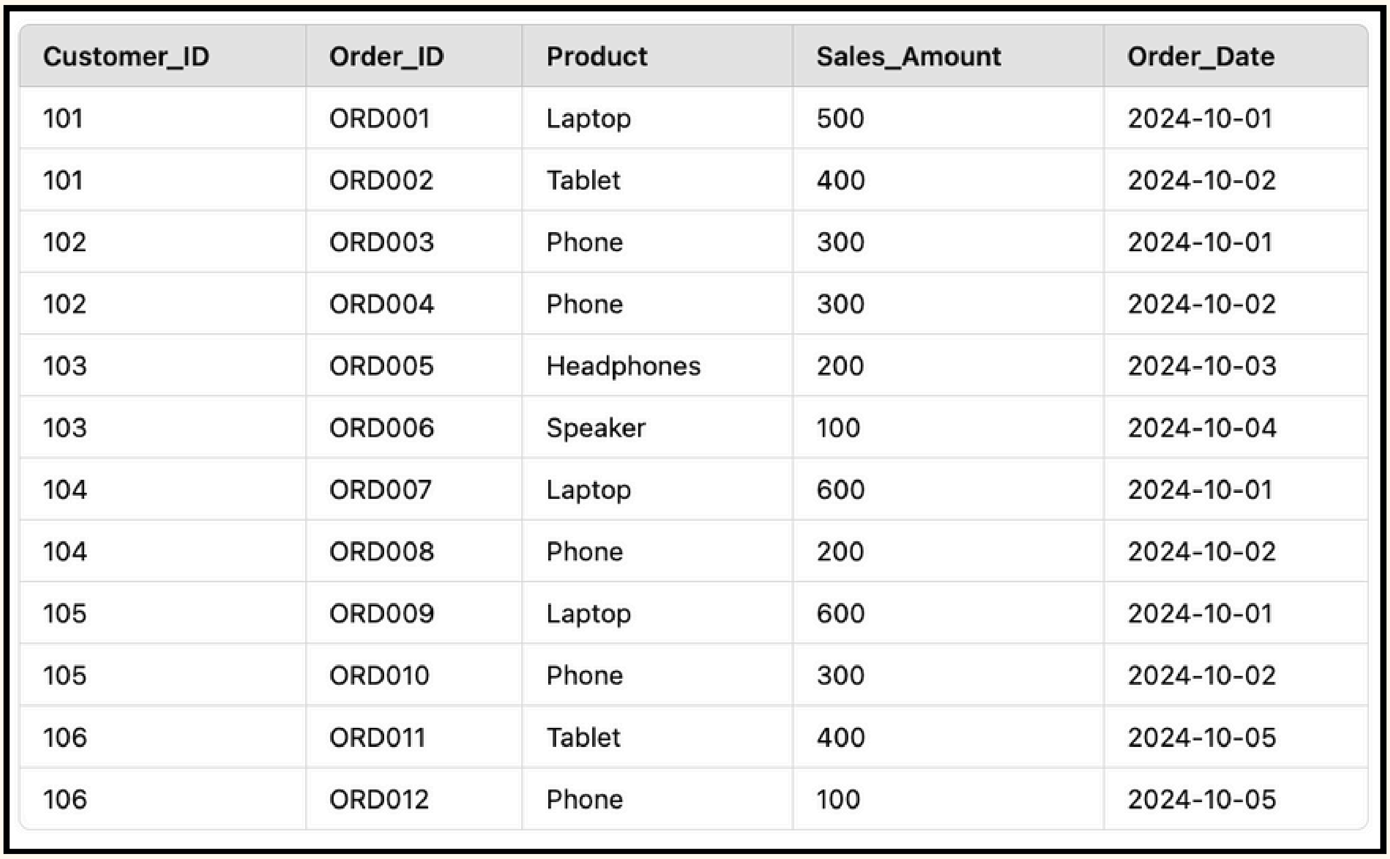
Why it matters for your data?

- Duplicate records can **distort business insights** and decisions.
- Correct ranking ensures that **data is ordered accurately** for analysis and reporting.
- SQL ranking functions like RANK(), DENSE_RANK(), and ROW_NUMBER() **allow you to deduplicate, sort, and filter data efficiently**.

We'll see how to apply these functions to solve real-world business challenges

SAMPLE DATASET

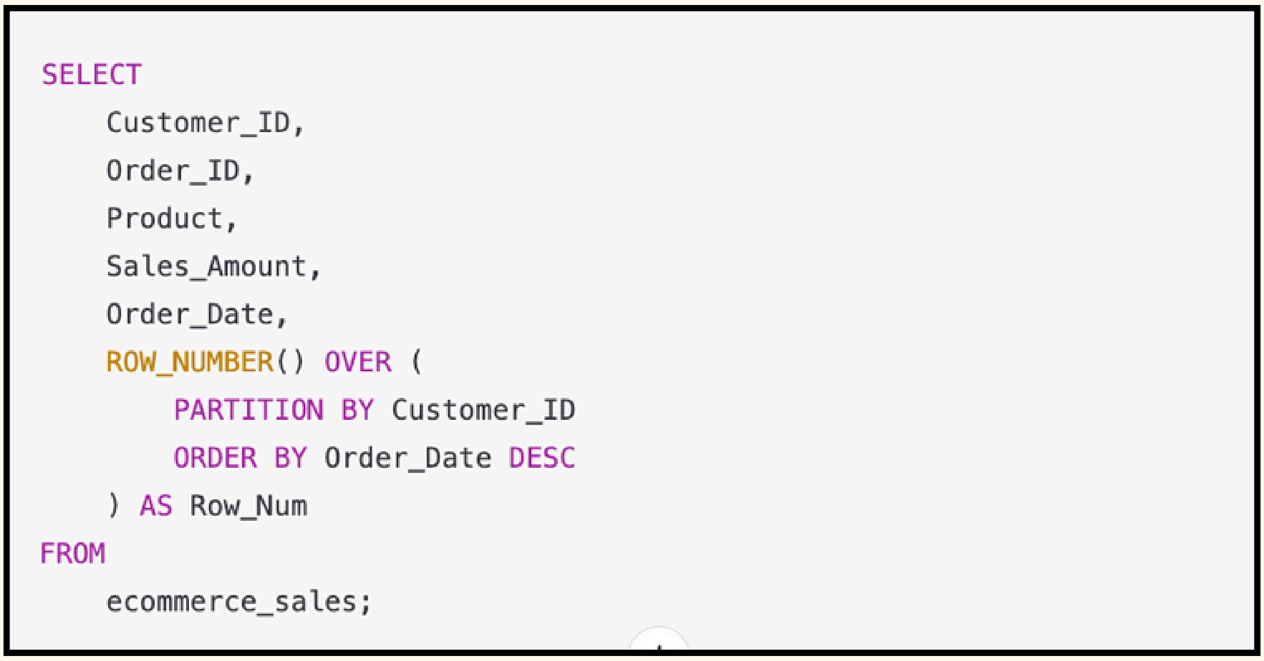
In eCommerce, duplicate entries in customer sales data can inflate revenue, misallocate marketing budgets, or create customer dissatisfaction.



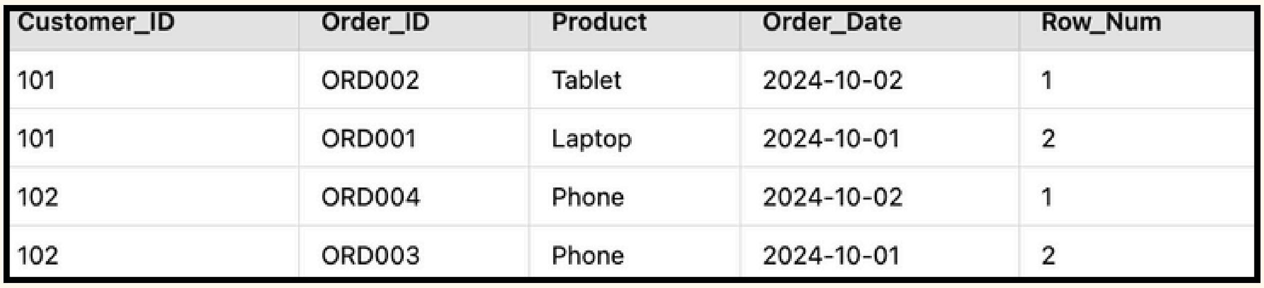
Customer_ID	Order_ID	Product	Sales_Amount	Order_Date
101	ORD001	Laptop	500	2024-10-01
101	ORD002	Tablet	400	2024-10-02
102	ORD003	Phone	300	2024-10-01
102	ORD004	Phone	300	2024-10-02
103	ORD005	Headphones	200	2024-10-03
103	ORD006	Speaker	100	2024-10-04
104	ORD007	Laptop	600	2024-10-01
104	ORD008	Phone	200	2024-10-02
105	ORD009	Laptop	600	2024-10-01
105	ORD010	Phone	300	2024-10-02
106	ORD011	Tablet	400	2024-10-05
106	ORD012	Phone	100	2024-10-05

ROW NUMBER()

Business Case: *Identify the most recent purchase for each customer*



```
SELECT
    Customer_ID,
    Order_ID,
    Product,
    Sales_Amount,
    Order_Date,
    ROW_NUMBER() OVER (
        PARTITION BY Customer_ID
        ORDER BY Order_Date DESC
    ) AS Row_Num
FROM
    ecommerce_sales;
```



Customer_ID	Order_ID	Product	Order_Date	Row_Num
101	ORD002	Tablet	2024-10-02	1
101	ORD001	Laptop	2024-10-01	2
102	ORD004	Phone	2024-10-02	1
102	ORD003	Phone	2024-10-01	2

*Use **ROW_NUMBER** to uniquely rank orders by date for each customer..*

VARUN SAGAR
THEEGALA

NEXT

RANK()

Business Case: *Determine the top-performing products by sales amount per customer.*

```
SELECT Customer_ID, Order_ID, Product, Sales_Amount,
       RANK() OVER (PARTITION BY Customer_ID ORDER BY Sales_Amount DESC) AS Rank
FROM ecommerce_sales;
```

Customer_ID	Order_ID	Product	Sales_Amount	Rank
101	ORD001	Laptop	500	1
101	ORD002	Tablet	400	2
102	ORD003	Phone	300	1
102	ORD004	Phone	300	1

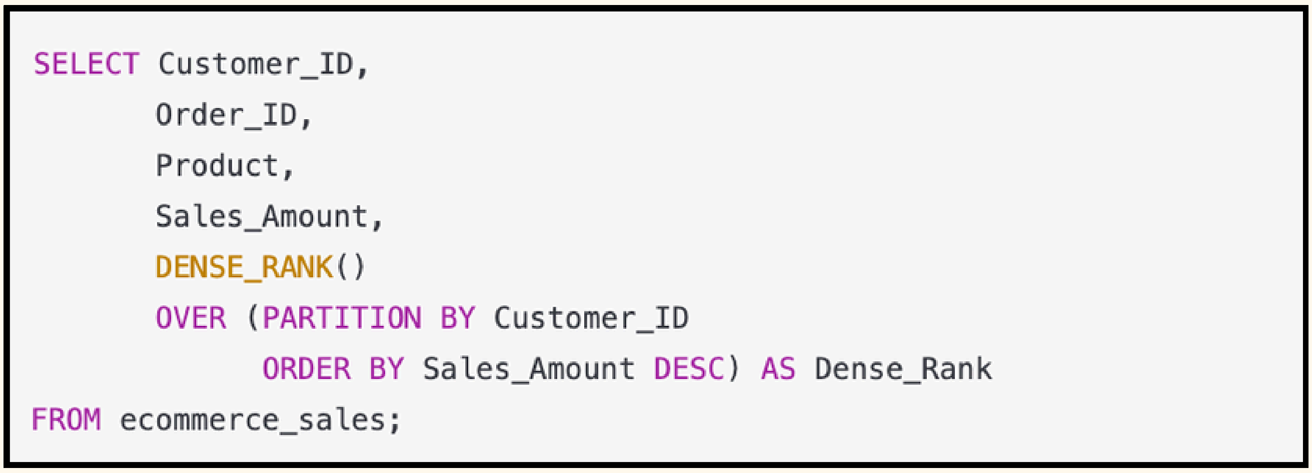
*Use **RANK** to account for ties in sales values while ranking.*

VARUN SAGAR
THEEGALA

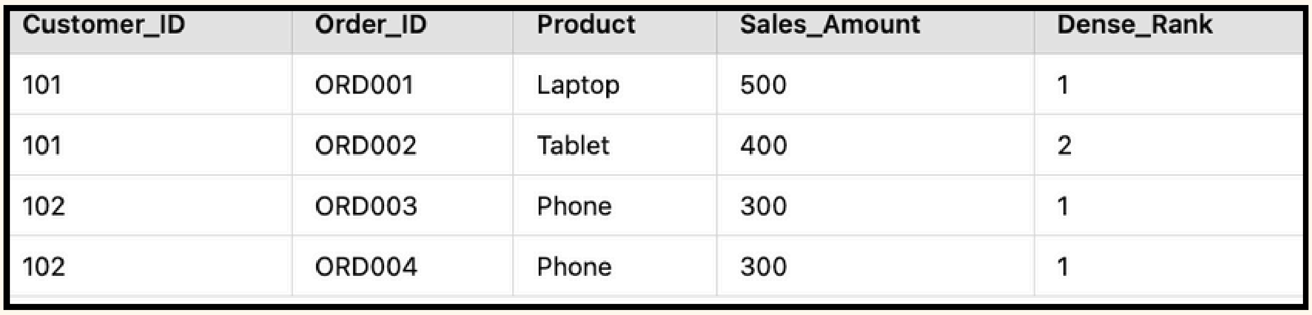
NEXT

DENSE RANK()

***Business Case**: Categorize sales performance tiers for each customer.*



```sql
SELECT Customer_ID,
       Order_ID,
       Product,
       Sales_Amount,
       DENSE_RANK()
       OVER (PARTITION BY Customer_ID
             ORDER BY Sales_Amount DESC) AS Dense_Rank
FROM ecommerce_sales;
```



Customer_ID	Order_ID	Product	Sales_Amount	Dense_Rank
101	ORD001	Laptop	500	1
101	ORD002	Tablet	400	2
102	ORD003	Phone	300	1
102	ORD004	Phone	300	1

*Use **DENSE_RANK** to group sales into tiers without gaps in ranks.*

VARUN SAGAR
THEEGALA

NEXT

REMEMBER

1. Use **ROW_NUMBER** to pinpoint and remove duplicates precisely.
2. Use **RANK** to skip gaps but keep duplicate entries ranked equally.
3. Use **DENSE_RANK** for continuous numbering without gaps, even for ties.

Always validate deduplication logic to align with business needs.

Data deduplication ensures accurate reporting and avoids costly mistakes in decision-making.

Varun Sagar
Theegala

SHARE THIS

If you think **your network** would find this **valuable**

FOLLOW ME

I help you **GROW & SUSTAIN** as a **Data Analyst**