Stop Writing Complex Queries. Master These Instead.

What Are They?

- Process data without collapsing rows
- Perform calculations across related rows
- Maintain original row-level detail
- Use an OVER() clause to define the window
- Essential for rankings, running totals, and comparisons
- A must-have tool for any data professional

ROW_NUMBER()

- Assigns a unique sequential number to each row
- Often used for ranking or pagination
- Syntax: ROW_NUMBER() OVER(ORDER BY column)
- Rows with identical values get different numbers
- Example: Rank customers by sign-up date
- Crucial for removing duplicate records

RANK() & DENSE_RANK()

- Both assign a rank based on ordered values
- RANK() leaves gaps after ties (1,2,2,4)
- DENSE_RANK() does not leave gaps (1,2,2,3)
- Perfect for leaderboards and sales rankings
- Use when multiple rows share the same value
- Example: Rank products by monthly sales

SUM() Over Windows

- Calculates a running total
- Can be cumulative or within a partition
- Syntax: SUM(column) OVER(ORDER BY date)
- Define frame with ROWS or RANGE
- Example: Track cumulative revenue month-over-month
- Powerful for trend analysis

LAG() & LEAD()

- Access data from previous or next rows
- LAG() looks back, LEAD() looks forward
- Compare current value to past or future
- Essential for calculating period-over-period change
- Example: Compare this month's sales to last month
- Specify offset to look further back/ahead

NTILE()

- Divides rows into roughly equal buckets
- Useful for segmentation and analysis
- Syntax: NTILE(4) creates quartiles
- Often used for cohort analysis
- Example: Segment customers into percentiles by spend
- Helps identify top-performing groups

FOLLOW FOR MORE

Like this post? Repost!

Share your thoughts in the comments!