SQL Window Functions and CTEs Manual

Topics Covered:

- Introduction to Window Functions
- Understanding the OVER() Clause
- Key Window Functions:
    - LEAD()
    - LAG()
    - RANK()
    - DENSE_RANK()
- The PARTITION BY Clause in Window Functions
- Common Table Expressions (CTEs)

1. Introduction to Window Functions

Window Functions are powerful SQL features that allow for complex calculations across sets of rows that are related to the current row. Unlike aggregate functions, which reduce multiple rows to a single summarised result, window functions perform calculations across rows while preserving the individual row details. This unique capability enables sophisticated data analysis within a single query, significantly enhancing the flexibility and efficiency of database operations. Window functions are particularly useful for running totals, moving averages, and other calculations that need to consider a subset of rows without collapsing them into a single output row.

2. Understanding the OVER() Clause

At the core of window functions is the OVER() clause, which specifies the partitioning and ordering of rows for the function to operate on. The OVER() clause has two main components:

PARTITION BY: This optional keyword divides the data set into partitions or groups, within which the window function operates independently. This is similar to how GROUP BY aggregates data, but with PARTITION BY, each row's original identity is retained, allowing for more granular analysis.
ORDER BY: Also optional, this component specifies the order in which the rows should be processed within each partition or across the entire data set if PARTITION BY is not specified. This ordering is crucial for functions that depend on a specific sequence of rows, such as LEAD() or LAG(), as it determines the relative position of rows used in calculations.

3. Key Window Functions

LEAD() and LAG()

LEAD(): This function is used to access data from a row that follows the current row within the specified partition or result set. It's particularly useful for comparing current row values with those of subsequent rows.

LAG(): In contrast, LAG() accesses data from a row preceding the current row, allowing for backward-looking calculations, such as comparing current row values to previous ones.

RANK() and DENSE_RANK()
RANK(): Assigns a unique rank to each row within a partition based on the specified ordering, with gaps in rank values for tied rows. This means if two rows tie for a position, the next rank will be incremented to reflect the gap.

DENSE_RANK(): Similar to RANK(), but without gaps for ties, ensuring a dense, consecutive ranking. Tied rows receive the same rank, and the following rank is immediately after the last assigned rank, regardless of ties.

4. The PARTITION BY Clause in Window Functions
The PARTITION BY clause enhances the analytical capabilities of window functions by segmenting data into distinct groups for separate analysis. This allows for the application of calculations to each partition, enabling detailed comparisons and insights at a more granular level. By dividing the dataset, analysts can isolate specific subsets for targeted analysis, such as comparing sales performance across different regions or departments.

5. Common Table Expressions (CTEs)
CTEs offer a way to define a temporary result set that can be referenced within an SQL statement. This feature simplifies complex queries by allowing them to be broken down into more manageable parts. CTEs are defined using the WITH keyword followed by the CTE name and the query that defines the CTE's result set. They are especially useful for recursive queries, which refer back to themselves in iterative data processing. By providing a structure for organizing subqueries, CTEs enhance the readability and maintainability of SQL code, making complex queries more understandable.

Reading Material:

https://learnsql.com/blog/window-functions-vs-aggregate-functions/

https://www.youtube.com/watch?v=Ww71knvhQ-s&authuser=1

https://www.youtube.com/watch?v=l_Zn5sdkamM&authuser=1

https://learnsql.com/blog/what-is-common-table-expression/