As a Data Analyst, As you should be proficient in advance concepts of SQL . Here are a few advanced SQL methods:  
  
1. Common Table Expressions (CTEs) - CTEs allow you to create temporary result sets that can be referenced within a SELECT, INSERT, UPDATE, or DELETE statement. They are especially useful for breaking down complex queries into simpler, more readable components.  
  
2. Window Functions - These functions perform calculations across a set of table rows that are related to the current row. Unlike aggregate functions, window functions do not cause rows to become grouped into a single output row, allowing you to retain individual row information while performing calculations.  
  
3. Subqueries and Correlated Subqueries - Subqueries can be used to provide data that will be used in the main query as a condition to further restrict the data to be retrieved. Correlated subqueries, which reference columns from the outer query, can dynamically adapt their results based on the current row being processed.  
  
4. Pivot and Unpivot Operations - These operations enable you to transform rows into columns and vice versa, which is particularly useful for generating reports or reformatting data for analysis.  
  
5. Recursive Queries - Using recursive CTEs, you can perform hierarchical or recursive data analysis, such as traversing a tree structure or generating a series of numbers.  
  
6. Analytic Functions - These include ranking functions (e.g., RANK, DENSE\_RANK), NTILE, and LEAD/LAG, which provide sophisticated ways to analyze data within partitions of the data set.  
  
  
7. Advanced Joins - Mastering various types of joins (e.g., INNER JOIN, LEFT JOIN, CROSS JOIN) and understanding their implications on query performance and result set structure is crucial for combining data from multiple tables effectively.  
  
8. Indexing Strategies - Understanding how to create and use indexes efficiently can drastically improve the performance of your SQL queries, especially in large databases.  
  
9. Query Optimization - Techniques such as analyzing query execution plans, avoiding unnecessary columns in SELECT statements, and leveraging database-specific optimization features can help in writing more efficient and faster SQL queries.  
  
10. Dynamic SQL - This involves building SQL queries dynamically at runtime, which can be useful for applications that need to adapt queries based on user input or other dynamic factors.