W08 Paper: Case Study Working with Subqueries

You have had a profitable eighth week at your new company. You mastered how to use subqueries. You also have discovered that some subqueries are non-correlated, and some are correlated. Your manager is curious how you are able to explain the difference between:

A scalar (or one column and one row) subquery versus a multiple row subquery.

A query managing a single column versus a multicolumn subquery.

A non-correlated and correlated subquery.

A fabricated derived table in the FROM clause versus a common table expression in a WITH clause.

You should return and report with a 3–5 paragraph report that clearly explains these subquery differences with one or two real world examples.

Report:

During this week, I have delved into the use of subqueries in SQL, particularly distinguishing between uncorrelated and correlated subqueries. The main difference between these is that uncorrelated subqueries operate independently of the main query and can run on their own. In contrast, correlated subqueries depend on the outer query, referencing its columns for execution.

In the case of scalar subqueries, which return a single row and column, they differ from multi-row subqueries in that the former are typically used for equality conditions within the main query. For example, a subquery returning the highest customer ID can be used to directly retrieve specific details of that customer. On the other hand, multi-row subqueries are used to filter data in a WHERE clause, using operators such as IN or NOT IN, to specify a range of acceptable values.

Also, about the difference between single-column queries and multi-column subqueries, the former are limited to simple and straightforward operations on a single attribute, multi-column subqueries allow more complex comparisons and are especially useful in JOIN operations, where it is necessary to correlate several columns between different tables.

Finally, I have compared derived tables manufactured in the FROM clause with common table expressions in a WITH clause. While derived tables are useful for simplifying complex queries into a single SQL command, common table expressions offer greater flexibility and clarity, especially in long or nested queries, by allowing recursive references and multiple references to the same defined subquery.