Ginevra Brannigan
3/12/2025
IT Foundations of Database Management
GitHub Link: https://github.com/GBrannigan13/DBFoundations

Assignment 07 –Functions

Introduction:

Session 7 evaluated the various functions, how they are used and the importance of them when pulling data. Below is a review of UDFs, Scalar, Inline and multi-statement functions.

Explain when you would use a SQL UDF

The use of User-Defined Functions (UDFs) are particularly useful when there are complex calculations, for example, calculating custom pricing, data transformation such as converting between different units or measurements. UDFs also ensure consistent application of business logic across applications, reduction of code duplication in multiple queries or stored procedures and is key for data validation and standardizing input across multiple applications.

UDFs are best used when calculations are complex, the logic needs to be reused across multiple queries and business rules need to be centralized. It is best to avoid using UDFs when a simple built-in function could be used, the logic needs frequent updates and performance is critical.

Explain are the differences between Scalar, Inline, and Multi-Statement Functions

Scalar Functions returns a single value, such as a string, number or date. They can be used in the SELECT statements and must have the BEGIN/END block in the logic. Scalar Functions are generally slower in performance and the database state cannot be modified. This function is best used for simple calculations.

Inline Table-Valued Functions returns a table based on a single SELECT. The RETURNS TABLE clause indicates that the function will return a table, but the function body consists solely of a SELECT statement. and does not need the BEGIN/END block. This Function is the best performance of all UDFs and can be treated as a parameterized view. Due to its performance it is best to use the Inline Function over others.

Multi-Statement Table-Valued Functions RETURNS @OutputTable TABLE clause defines the structure of the table the function returns. The function body is executed using BEGIN/END blocks and can include multiple SQL statements such as INSERT, UPDATE, SELECT, and other operations. Compared to the other functions it is slower; however, it is more flexible. This function is best only used when necessary for complex logic.

Summary:

Functions are great to consider for their performance impact in high-volume scenarios, however, while they provide excellent encapsulation and reusability, they can impact performance if not designed carefully, especially in large datasets.