Name: Jennifer Grace L. Duldulao
Date: November 23, 2025
Course: Foundations Of Databases & SQL Programming
Github Link: https://github.com/hihellojini/DBFoundations-Module07

Assignment 7 – Functions

Introduction

This assignment explains SQL Functions, particularly on User-Defined Functions or known as UDFs. This will also explain the differences of the scalar, inline and multi-statement functions.

SQL UDF

SQL UDF stands for SQL User-Defined Function. This is a function that can be created in a SQL database to capture repetitive logic into a single callable function that can be called in queries just like the other built-in SQL functions such as min(), avg(), etc. This function is particularly tailored to the specific need of the user. It can also accept parameters and return a result.

Below is an example of a UDF.

```
create function fProductInventoriesWithPreviousMonthCountsWithKPIs(@value1 int)
	returns table
		as
			return select
				[ProductName],
				[InventoryDate],
				[InventoryCount],
				[PreviousMonthCount],
				[CountVsPreviousCountKPI]
			from vProductInventoriesWithPreviousMonthCountsWithKPIs
			where [CountVsPreviousCountKPI] = @value1;
go
```

Scalar, Inline, and Multi-Statement Functions

Scalar functions return a single value and are used for simple calculations or transformations, but can be slow since they run row by row. Inline table-valued functions return a table using a single **select** statement and perform efficiently because SQL can optimize them like a view. Multi-statement table-valued functions also return a table but allow multiple steps and more complex logic, though they may be slower due to using table variables.

Each type has a trade-off between simplicity, flexibility, and performance, so choosing the right function depends on the task.

Table 1 below shows a comparison summary.

Type of Function	*Returns*	*Use*	*Performance*
Scalar	Single value	Simple calculations	Row-by-row execution and may be slow
In-line	Table	Optimized like a view	Very efficient
Multi-statement	Table	Complex logic requiring multiple steps	Slower due to table variable usage

Table 1. Table Comparison of Types of UDF

Summary

SQL UDFs improve code reusability, readability, and consistency. The type of UDF to use depends on whether you need a single value, a table, or complex logic. Understanding their differences will help write better and more efficient SQL code.