**Name:** Guillermo Dominguez

**Date:** Nov 28th, 2022

**Course:** DB Foundation (SQL)

**Github:** https://github.com/guillermo-dominguez/DBFoundations-Module07

# Assignment 7- Functions

# Introduction

In this write-up, I will explain SQL User Defined Functions (UDF) as well as when to use them, and I will explain the differences between Scalar, Inline, and Multi-Statement Functions.

# SQL UDFs When to Use Them

SQL UDFs are functions that are defined by users when built-in functions cannot accomplish the task at hand. UDFs can return single values (scalar), or return a table. In the assignment this week we created a function that returns a table:

CREATE Function fProductInventoriesWithPreviousMonthCountsWithKPIs (@KPI int)

RETURNS TABLE

AS

RETURN(

SELECT \*

FROM vProductInventoriesWithPreviousMonthCountsWithKPIs

WHERE CountVsPreviousCountKPI = @KPI);

Go

UDFs can also be used for check constraints to reference columns from other tables.

# Differences between Scalar, Inline, and Multi-Statement Functions

Scalar functions are functions that only return one value. Scalar functions require that the user includes the schema name, and users can include several parameters in them. An example from Module 7 notes is:

Create Function **dbo**.**MultiplyValues**(@Value1 Float, @Value2 Float)

Returns Float

As

Begin

Return(Select @Value1 \* @Value2);

End

go

-- Calling the function

Select Tempdb.**dbo**.**MultiplyValues**(4, 5);

Go

**(Module 7 notes, 2022)**

Inline (or table) functions are functions that return tables such as the one in the previous section. Parameters are not particularly useful in these but help define what the function will return. These functions are more similar to views.

Multi-Statement functions also return a table, but the table is more complicated to build as columns have to be defined, and these tables include several select statements.

-- Complex Table Value (Tabluar) Functions with Multiple Statements

go

Create Function dbo.fArithmeticValuesWithFormat(@Value1 Float, @Value2 Float, @FormatAs char(1))

Returns @MyResults Table

( [Sum] sql\_variant

, [Difference] sql\_variant

, [Product] sql\_variant

, [Quotient] sql\_variant

)

As

Begin --< Must use Begin and End with Complex table value functions

If @FormatAs = 'f'

Insert Into @MyResults

Select Cast(@Value1 + @Value2 as Float)

,Cast(@Value1 - @Value2 as Float)

,Cast(@Value1 \* @Value2 as Float)

,Cast(@Value1 / @Value2 as Float)

Else If @FormatAs = 'i'

Insert Into @MyResults

Select Cast(@Value1 + @Value2 as int)

,Cast(@Value1 - @Value2 as int)

,Cast(@Value1 \* @Value2 as int)

,Cast(@Value1 / @Value2 as int)

Else

Insert Into @MyResults

Select Cast(@Value1 + @Value2 as varchar(100))

,Cast(@Value1 - @Value2 as varchar(100))

,Cast(@Value1 \* @Value2 as varchar(100))

,Cast(@Value1 / @Value2 as varchar(100))

Return

End

go

-- Calling the function

Select \* FROM dbo.fArithmeticValuesWithFormat(10, 3, 'f');

Select \* FROM dbo.fArithmeticValuesWithFormat(10, 3, 'i');

Select \* FROM dbo.fArithmeticValuesWithFormat(10, 3, null);

go

**(Module 7 notes, 2022)**

# Summary

In this assignment, I explained SQL UDFs and when to use them, as well as the differences between Scalar, Inline, and Multi-statement Functions.