Name: Marcus Bradlee

Date: November 26, 2025

Course: IT FDN 130

GitHub: DBFoundations

Assignment 7 – Functions

Introduction

Assignment 7 introduced us to using SQL functions. There are two basic categories of functions: SQL's built-in functions and user defined functions (UDFs). In this assignment we utilized both types of functions, using the built-in functions in our select statements and building a UDF for the retrieval of data from our views.

User Defined Functions

In addition to SQL's built in functions, you are also able to create your own custom functions, also known as User Defined Functions (UDFs). There are two basic kinds of functions:

- Functions that return a table of values (Inline / Multi-statement)
- Functions that return a single value (Scalar)

When you need to perform a specific task in your database such as a complex operation, a calculation, or some type of data manipulation, it is best to write a UDF. This is especially true if this specific task might be repetitive, as UDFs make repetitive tasks much more efficient as you reduce redundancy.

Inline / Multi-statement Functions

An inline function is a function which returns a table using a single select statement. For example:

```
Create Function fProductList(@IDValue int)
       Returns Table As
       Return Select
              p.ProductID
              ,p.ProductName
              ,p.UnitPrice
       From vProducts p
       Where p.ProductID = @IDValue
Go
```

The example above demonstrates an inline UDF which is comprised of a single select statement and returns a table of the ProductID, ProductName, and UnitPrice of products whose ProductID matches the function parameter passed in by the user.

A multi-statement function also returns a table, however it is comprised of multiple SQL statements such as Insert, Update, and Delete, not just Select.

Scalar Functions

A scalar function is a function which returns only a single value, such as an integer or a string. The use of parameters is typically considered much more useful in scalar functions as opposed to inline functions. For example:

```
Create Function dbo.fSum(@Value1 float, @Value2 float)
	Returns float As
	Begin
		Return(Select @Value1 + @Value2)
	End
Go

Select dbo.fSum(1, 2)
```

The example above demonstrates a scalar UDF which takes two float values as parameters and returns the sum. In this case, given the values 1 and 2, the return value would be 3.

Summary

Built-in SQL functions allow us to perform specific tasks to create more complex queries, and user defined functions add an even further amount of flexibility and complexity. UDFs help to encapsulate SQL statements which reduce redundancy and increase reusability. UDFs are a great tool to utilize when designing a clean, efficient, and secure relational database.