Sarah Hogan

August 22, 2021

IT FDN 130 – Foundations of Databases & SQL Programming

Assignment 07

GitHub: [SarahEH/DBFoundations-Module07](https://github.com/SarahEH/DBFoundations-Module07)

# **Assignment 07 – Functions**

## Introduction

In this paper, I will discuss the use of Functions in SQL, specifically when to use User Defined Functions, and the differences between scalar functions compared to inline and multi-statement functions.

## When to use a SQL User Defined Function (UDF)

User Defined Functions, known as UDFs, are part of the SQL lexicon. The SQL language comes with many pre-loaded functions that can perform aggregations and basic math, can manipulate formatting of things like dates and currencies, and can perform a variety of text and string functions to find, parse, and trim your data cells as text. However, occasionally a SQL user will need something more unique, more discrete, or more powerful. User Defined Functions can serve this purpose.

UDFs can be used to simplify a repeated set of operations, rather than needing to constantly replicate the code. UDFs can also take on variables that can be passed into the function to affect the outcomes. UDFs can produce a single scalar value like multiplying two numbers together (particularly useful when calculating something like item cost and item quantity to get the full order value). UDFs can also produce a full view statement, as in our homework where we wanted to pass a particular variable into the ‘WHERE’ clause to filter down results that matched the parameters selected by the user.

UDFs cannot encapsulate create & insert statements, so they do have an upper limit, but they can add a whole new dimension to working with SQL depending on what kind of calculations and reporting capability are needed to meet the business requirements.

## Scalar, Inline, Multi-Statement Functions

User Defined Functions have three primary groupings of what they are and what they do.

1. Scalar UDFs will produce a single answer. You can write the internal code to be as simple as adding or subtracting, or as complicated as the mind can imagine, and a scalar UDF will take in as many variables as it needs to and generate an answer. This can be particularly useful when you need to add a calculated column to a SQL table, or to generate a value to be used as a variable elsewhere. Since scalar UDFs generate a very simple, very basic answer/result, they can even be used as check constraints when formatting tables.
2. Inline and Multi-Statement Functions are two kinds of table-valued functions, so I’ll group them together under this hierarchy. Scalar UDFs return single numbers, Inline and Multi-Statement Functions return tables.
   1. Inline UDFs would contain data from a **single SELECT statement**. It’s similar in nature to a view in that it returns a somewhat simple table, but more complex in that you can pass in parameters that would most likely manifest in the WHERE clause. Perhaps you want a function to return the billing for the month? Pass the month-end date as a parameter, and have it populate into the WHERE clause to filter down the date.
   2. Multi-Statement UDFs would contain data from **multiple statements,** as the name suggests. The syntax on Multi-Statement functions will mandate the use of “BEGIN” and “END” to capture all lines of code. This is unique but necessary for Multi-Statement Functions so that all of the commands get incorporated correctly. Multi-Statement functions often incorporate more complex data manipulation, which can be anything from an INSERT, UPDATE, or DELETE statement, table UNIONs, manipulating inputs before passing them into a traditional SELECT, FROM, WHERE, etc.

## Summary

In this paper, I have discussed a short overview of what Functions are and the different applications of them. Scalar functions are the easiest to write, inline functions are most similar to a view statement, but every once and a while the business will require the use of a multi-statement function. At the moment, I cannot think of a way to use multi-statement functions in my day to day work, but I can already see cases where both scalar and inline functions have the opportunity to improve the work we do!