Functions

# Introduction

This week’s assignment introduced the class to functions that can not just perform calculations on a set of values and return a single value but also on whole tables and return unique tables. These functions can among other things select, partition, window and report. Examples of some built-in functions for SQL include aggregate functions (max, min, count, sum, avg), conversion functions (cast, convert, string, concat, IsNumeric, IsDate, Date/Time Functions), partition functions (over-partition) and ranking functions (rank, lag, lead). A user can define their own functions that are different from the inherent SQL functions so as to broaden their function capabilities. In addendum to last week’s assignment’s use of a Github repository, this assignment had the class create a Github webpage to display this knowledge document. The formatting was done with the use of the programming language, **Markdown**, an easy-to-read and easy-to-write syntax for formatting plain text. The language allows text to be formatted for titles, subtitles, captions, **bolding**, *italics* and much more. In addition, images and code can be uploaded and added. All in all, it allows for another format to share one’s findings and work.

# SQL UDF

A **SQL User Defined Function (UDF)** is custom built function by a programmer that can either return a table of values or a single value. Scalar functions return a single (scalar) value as an expression. A scalar function will ask for one or more parameters. It then will use a combination of other functions as well as select statements and such to produce that single scalar value. Table functions can do the same but then intake the parameter and apply it to a whole table, row by row. UDFs come in handy when checking constraints as one can not reference a column in another table otherwise. This allows the script to better connect different tables within a database while producing a refined table output.

# Scalar, Inline and Multi-statement Functions

A **scalar function** returns a single value as a result of actions performed by a function. It can accept 0 to many input parameters and still return a single value. A scalar user-defined function returns one of the scalar data types (int, char, varchar) but the text, string, ntext and other data types are not supported. An **inline function** returns a table variable as a result of actions performed by the function. An inline table-valued function returns a variable of data type table whose value is derived from a single SELECT statement. Since the return value is derived from the SELECT statement, there is no BEGIN/END block needed in the CREATE FUNCTION statement. There is also no need to specify the table variable name because the structure of the returned value is generated from the columns that compose the select statement. No duplicate column names are allowed and all derived columns must have an associated alias. A **Multi-statement function** returns a table variable as a result of actions performed by a function . It can have one or more than one T-SQL statement. Within the create function command you must define the table structure that is being returned. After creating this type of user-defined function, we can use it in the FROM clause of a T-SQL command unlike the behavior found when using a stored procedure which can also return record sets.

# Summary

The assignment was a success in introducing functions. The questions in the sql script file built on each other one by one, starting with learning how to use the format function. It then progressed to incorporating functions for selected columns while accounting for joins and referencing views. Finally, a user defined function referenced a view that used performance tracking functions before directing a function table output dependent on one column’s integer. These lessons stacked up nicely and I look forward to applying them to whole stored procedures.