Laura Truong

May 21, 2021

Introduction to SQL

Assignment07

<https://github.com/laurat-tech/DBFoundations-Module07>

Functions

**Introduction**

In Assignment 7 of IT FDN 130 A with Professor Randall Root, we learn about how to use SQL functions to retrieve information from a database. We cover aggregate functions, selecting with common functions, partitioned or windowed functions, using functions for reporting, user defined functions, and using UDF’s for check constraints. We will then answer a series of questions and publish our work to GitHub. By the end of this lesson, we will be able to explain when you would use a SQL UDF and explain the differences between scalar, inline and multi-statement functions.

***Explain when you would use a SQL UDF.***

UDF’s are custom functions with a return value of a single scalar value or a result set. UDF’s are used for the following reasons:

* They allow modular programming- once they are created and stored in a database, they can be called at any time or modified independently of the program source code.
* They execute faster- They reduce the compilation cost of t-sql code by caching plans and reusing them for repeated execution. UDF’s do not need to be reparsed with each use so they execute much faster.
* They reduce network traffic- If an end-user decides that they want to filter data based on complex constraints, they will invoke it in a ‘Where’ clause to reduce the number of rows sent to the client.

***Explain the differences between Scalar, Inline, and Multi-Statement Functions.***

Scalar functions accept zero or more parameters and allow you to bring back a single value and help simplify your code. Scalar functions will return all data types except for text, ntext, image, cursor, and timestamp. They can be used in a ‘Where’ clause of the SQL query. You can create a scalar function using the statement below:

CREATE FUNCTION [schema\_name.]function\_name (parameter\_list)

RETURNS data\_type AS

BEGIN

statements

RETURN value

END

Note that “The schema name is optional. If you don’t explicitly specify it, SQL Server uses dbo by default” ([SQL Server Scalar Functions By Practical Examples (sqlservertutorial.net)](https://www.sqlservertutorial.net/sql-server-user-defined-functions/sql-server-scalar-functions/)).

Table value functions accept zero or more parameters and return a table variable. They are broken down into inline functions and multi-statement functions. An in-line table function contains a single statement that must be a ‘Select’ statement. The result of the query becomes the return value of the function. There is no need for a Begin-End block in an inline function. Inline table functions can be created with the following syntax:

CREATE FUNCTION function-name (Parameters)

RETURNS return-type

AS

RETURN

Multi-statement functions contain multiple SQL statements that are surrounded by Begin-End blocks. The return value is declared a table variable and includes the full structure of the table to be returned. The ‘Return’ statement does not have a value and the declared table variable is returned. A multi-statement table function can be created using the following syntax from [User Defined Functions in SQL Server (c-sharpcorner.com)](https://www.c-sharpcorner.com/UploadFile/3194c4/user-defined-functions-in-sql-server/):

CREATE FUNCTION function-name (Parameters)

RETURNS @TableName TABLE

(Column\_1 datatype,

.

.

Column\_n datatype

)

AS

BEGIN

Statement 1

Statement 2

.

.

Statement n

RETURN

END

**Summary**

In Assignment 7 of IT FDN 130 A with Professor Randall Root, we learn about how to use SQL functions to retrieve information from a database. We cover aggregate functions, selecting with common functions, partitioned or windowed functions, using functions for reporting, user defined functions, and using UDF’s for check constraints. We eventually created some SQL queries/views and published our work to GitHub. We conclude the assignment by explaining when you would use a SQL UDF and explaining the differences between scalar, inline and multi-statement functions. Understanding a UDF and its different types can save a developer a lot of time by making it easy for them to save and access redundant code and reduce network traffic.