Marci DeLeon
November 28, 2021
IT FDN 130: Foundations of Databases and SQL Programming
Module 7
https://github.com/Marci-student/DBFoundations

Functional Functions

Introduction

Functions are much of what makes database analysis possible. What's the point of having quantifiable data if you can't count, sum, or average it? How can you tell how many activities were done on-time, or even better, what percentage of the activities were done on-time? Functions make calculations and analyses possible.

In this essay, I will explain what a User-Defined Function (UDF) is, as well as the differences between various types of functions.

What Is a UDF and Why Is It Useful?

Many functions are hard-coded into SQL. Functions such as SUM(), CONCAT(), or DATEDIFF() have been used by legions of coders in a multitude of ways, but do not need the user to explain how the computer is to carry out those functions.

A user-defined function (UDF), on the other hand, must be created and defined by the coder. This allows for questions to be answered that one coder wants to know the answer to. For example, FMD-145 requires that all food firms receive a copy of their Establishment Inspection Report within 45 calendar days of final determination of classification[1]. In order to find how many of these letters were within timeframe, I might create a user-defined function using the CASE() function that would output the results of early/on-time communications versus late communications. In addition to being useful in the initial query that I made it for, having this code be a function would mean that it would be very easy for me to copy and use in other queries, or to provide to others to use in their own queries.

Additionally, UDFs (along with other functions) can be easily used in a SELECT statement, even in the WHERE or HAVING clauses, and the output can become a new column in the table or view.[2] All of these are powerful reasons to use functions, and especially to create functions for answering one's own questions.

[1] "FMD-145: Release of the Establishment Inspection Report." DIR-000067, version 3, L. Holmquist. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwipnufv2rz0AhVpqVsKHTh8CRsQFnoECAUQAQ&url=https%3A%2F%2Fwww.fda.gov%2Fmedia%2F83055%2Fdownload&usg=AOvVaw3D1BCHnlUSJEuhP4d8g7aw. Accessed 11/28/21. Outside reference.
[2] "SQL Server: Advantages and Disadvantages of User-Defined Functions." Baya Pavliashvili, informIT. https://www.informit.com/articles/article.aspx?p=31724. Accessed 11/28/21. Outside reference.

Scalar, Inline, and Multi-Statement Functions

All three of these are different types of functions in SQL, and they provide different types of outputs. Here, I will explain what they are and the differences between them.

Scalar Functions

A scalar function takes one or more parameters and returns a single value. This function may be as simple as “@input + 2” or as complex as meteorological projections.

Inline Functions

While a scalar function provides one answer, an inline function provides a table of values. It is somewhat like a view in that it will output a table with different columns. It can only have one select statement, unlike a multi-statement function. But, like all functions, it can accept parameters and perform actions.

Multi-Statement Functions

As implied above, a multi-statement (table-valued) function is the same as an inline function, save that it can have multiple select statements. This can be very useful if you’re using values that are found in different tables (or views).

Summary

Functions enable analysis – a database is a mere collection of data without them. It is through functions that an analyst can use the data to actually come to conclusions. There are many types of functions. They can be hard-coded or user defined. They can output a single value (as in a scalar function), or a full table (as in an inline or multi-statement function).