Maria Leal
November 30, 2022
FDN 130 A
Assignment 07
https://github.com/MlSQL130/DBFoundations-Module07.git

Functions

Introduction

Functions are named expressions that can return a single value or multiple values. They can be used multiple times and only need to be written once. They reduce repetitive tasks. There are two types of SQL functions, built in or user defined functions(UDFs). This paper will focus on User Defined Functions.

SQL User Defined Functions

When none of the built in functions work, you repeatedly find yourself writing the same query, or created a complex query, then its's time to consider creating a user defined function. There are two types of UDFs; scalar and table valued functions.

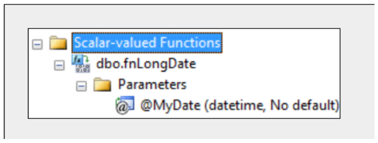
Scalar Functions

Scalar functions accept any number of parameters and returns a single value each time you execute it. Figure 1 provides an example of the syntax required to create a function.

```
CREATE FUNCTION fnLongDate—you name your function
(
@MyDate AS DATETIME--After listing the parameter you tell it what kind of data to return
)
RETURNS VARCHAR(MAX) –you tell it to return a string of text
AS
BEGIN
--Our function code goes here!
DATENAME(DW,@MyDate) + ' ' +
DATENAME(D,@MyDate) + ' ' +
DATENAME(M,@MyDate) + ' ' +
DATENAME(YY,@MyDate)
END
```

Figure 1. User-Defined Function(https://www.wiseowl.co.uk/blog/s344/sql- functions.htm, 2013)

You refresh the database and check that the new function is in the database scalar function folder, seen Figure 2.



Execute a query using your new scalar function.

```
SELECT
FilmName
,dbo.fnLongDate(FilmReleaseDate)
FROM
tblFilm
```

Figure 2. Scalar-valued function folder example and function execution

Table Value Functions

Table value functions(TVF) are used to return a table and more than one value. Views can return tables as well, but they don’t accept parameters. There are two types of TVFs; in-line and multi-statement.

In-line functions are made up of one select statement, returns a single set of rows. The syntax for an in-line function would look like the example in Figure 3.

```
CREATE FUNCTION fnFilmsByDuration(
@duration int –parameter name and datatype
)
RETURNS TABLE
AS
RETURN
SELECT—using a single select statement
FilmId,
FilmName,
FilmRunTimeMinutes
FROM
tblFilm
WHERE
FilmRunTimeMinutes >= @duration

SELECT * from dbo.fnFilmsByDuration(190)—execute  function
```

Figure3. In-line function example (https://www.wiseowl.co.uk/blog/s347/in-line.htm, 2013)

Multi-Statement functions are used to build some unique code outside the context of a Select statement. The idea is to build a table from multiple sources using unique logic, or duplicating the functionality of an in-line function with more complex logic.

```
CREATE FUNCTION fnName(
-- can have 0, 1, 2 or more parameters
@param1 datatype,
@param2 datatype, ...
)
-- define output table  format to return—different from in-line function you define table and identify columns to return-
RETURNS @TableName TABLE (
Column1 datatype,
Column2 datatype,
...
Columnn datatype,
)
AS
BEGIN—
-- typically insert rows into this table
-- eventually, return the results
RETURN
 END
```

Figure 4. Multi-Statement Syntax (https://www.wiseowl.co.uk/blog/s347/multi-statement.htm, 2013)

Scalar, Inline, Multi-Statement Functions

The differences between these types of user-defined functions are the following

Scalar Function	In-Line Function	Multi-Statement Functions
Returns single value	Returns a table object	Returns a table variable
Syntax has Begin and End	No Begin and End required	Begin and End in syntax
No Select Statement	Single Select Statement	Multiple Select Statements
Don’t define table variables	Don’t define table variables	Need to define table variables

Summary

Functions can help reduce repetitive tasks, allow parameters and return tables. But then you may ask why not a stored procedure or a view. Consider the use case scenario and then determine what will work best or start a conversation with your peers. Have fun in the process.