**Kevin Xie**

**5/20/21**

**ITFDN130**

**Assignment07**

<https://github.com/bgcoolant/DBFoundations-Module07>

**Introduction**

User Defined Function is a UDF is a programming construct that accepts parameters, does actions and returns the result of that action. The result either is a scalar value (single value) or result set (table). UDFs can be used in scripts, Stored Procedures, triggers and other UDFs within a database.

1. Explain when you would use a SQL UDF.

UDF is used when the built-in functions do not meet user’s requirement, or it is convoluted to call out a series of such functions to achieve the purpose. It also offers more flexibility in SQL programming as it allows users to customize functions to their needs. The main benefits of UDF include:

* UDFs support modular programming. Once you create a UDF and store it in a database then you can call it any number of times. You can modify the UDF independent of the source code.
* UDFs reduce the compilation cost of T-SQL code by caching plans and reusing them for repeated execution.
* They can reduce network traffic. If you want to filter data based on some complex constraints then that can be expressed as a UDF. Then you can use this UDF in a WHERE clause to filter data.

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

A Scalar UDF accepts zero or more parameters and return a single value. The return type of a scalar function is any data type except text, ntext, image, cursor and timestamp. Scalar functions can be used in a WHERE clause of the SQL Query. The simplest example is any mathematical calculation. We can create a function that takes in parameters and calculate them into a pre-defined result. For example, the following UDF calculate addition of the two parameters:

Create Function **dbo**.**AddValues**(@Value1 Float, @Value2 Float)

Returns Float

As

Begin

Return(Select @Value1 + @Value2);

End

Go

Select tempdb.**dbo**.**AddValues**(1,2);

go

Table Value Functions are broken down to Inline Functions and Multi-Statement Functions. An Inline Table Valued 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. For example, the following functions returns a table of inventories that are greater than 60.

Create Function dbo.GetQTY (@count INT)

Return Table

As

Return

Select \* From dbo.Inventories Where [Count] > @Count:

Go

Select \* From dbo.GetQTY(60);

Go

A Multi-Statement function contains multiple SQL statements enclosed in BEGIN-END blocks. In the function body you can read data from databases and do some operations. In a Multi-Statement Table valued function the return value is declared as a table variable and includes the full structure of the table to be returned. The RETURN statement is without a value and the declared table variable is returned. The syntax is following:

CREATE FUNCTION MultiStatement\_TableValued\_Function\_Name

(

@param1 DataType,

@param2 DataType,

@paramN DataType

)

RETURNS

@OutputTable TABLE

(

@Column1 DataTypeForColumn1 ,

@Column2 DataTypeForColumn2

)

AS

BEGIN

--FunctionBody

RETURN

END