Name: mGrilli
Date: 11/30/2022
Course Name: IT FDN 130
GitHub URL: https://github.com/genevievelim/DBFoundations-Module07.git

Assignment 07
FUNCTIONS

Summary

This week we expanded our knowledge by adding SQL functions to our ever-growing toolkit. These functions come in a variety of formats and have several different use cases. These functions include Scalar, Inline, Multi-statement, and User Defined Functions (UDF). Functions are a slick way to allow users to execute very lengthy and complex code over and over without knowing how things are working in the background.

Topic 1 — When to use a UDF

Before we jump into what a UDF is, lets take a look at how one is made.

1
```
CREATE VIEW
    vProductInventoriesWithPreviousMonthCountsWithKPIs
AS
SELECT
    vProducts.ProductName,[InventoryDate] = DateName(Month, vInventories.InventoryDate) + ' , ' +
                DateName(Year, vInventories.InventoryDate),
    [InventoryCount] = vInventories.Count, [Previous Month Count] = ISNULL( LAG(vInventories.Count,1)
        OVER (ORDER BY vProducts.ProductName,  InventoryDate), 0) ,
    [Count vs Previous Count KPI] =

CASE
    WHEN
        vInventories.Count > ISNULL( LAG(vInventories.Count,1) OVER (ORDER BY vProducts.ProductName,
                        InventoryDate), 0 )  THEN 1
     WHEN
        vInventories.Count = ISNULL( LAG(vInventories.Count,1) OVER (ORDER BY vProducts.ProductName,
                        InventoryDate), 0 )  THEN 0
    WHEN
        vInventories.Count < ISNULL( LAG(vInventories.Count,1) OVER (ORDER BY vProducts.ProductName,
                        InventoryDate), 0 )  THEN -1
END
FROM
    vProducts
JOIN
    vInventories
ON
    vProducts.ProductID = vInventories.ProductID
GO
```

2
```
CREATE FUNCTION
    fProductInventoriesWithPreviousMonthCountsWithKPIs(@value As INT)
RETURNS
    Table
AS
RETURN
    SELECT
        *
    FROM
        vProductInventoriesWithPreviousMonthCountsWithKPIs
    WHERE
        [Count vs Previous Count KPI] = @value
GO
```

3
```
SELECT * From fProductInventoriesWithPreviousMonthCountsWithKPIs(1);
```

1. *We create a complex queery as a view*
2. *We create a function with parameters that calls our view*
3. *We call our function giiving it an agurment*

User defined function is a database object that is used to save a lengthy or complicated query so that it can be ran repeatedly using simple commands. A function can return a single value back to the caller, or it can return a result set. Through the use of parameters, functions can also be made dynamic.

Topic 2

Explain are the Differences between Scalar, Inline, and Multi-Statement Functions

A **Scalar Function** is a function that returns a single value back to the caller. A Scalar Valued Function can be used anywhere a single value is expected. You can pass in one or more parameters to a Scalar Valued Function and do work based on those parameters.

```
CREATE FUNCTION
    functionName ( <optional parameter list> )
RETURNS
    <data type of return value>
AS
BEGIN
    <function body>
RETURN
    <value to return>
END
```

A **Inline Table Function** returns a result set to the caller by way of an inner SELECT statement that is part of the function definition. You can query an Inline Table Valued function similar to how you would query a table. They accept input parameters which allow them to return a different result set depending on the parameters passed in.

```
CREATE FUNCTION
    <schema>.function_name( <optional_
    parameters> )
RETURNS
    Table
AS
RETURN
<SELECT statement>
```

A **Multi Statement Function** returns the result set of a table variable that is created, defined, and populated within the definition of the function. The advantage of a Multi Statement Table Valued function is you have complete control over the definition of the table variable, including naming your columns, as well as setting up appropriate data types, constraints, indexes, etc.

```
CREATE FUNCTION
    <schema>.function_name( <optional
    parameters> )
RETURNS
    @table_variable_name TABLE
    (
        <table variable definition>
    )
AS
BEGIN
     <function body to populate @table_
     variable_name>
RETURN
END
```

Conclusion

SQL Functions are great tools that allow for a more robust, yet more convenient user experience for the people we develop for. In SQL there a 3 main types at our deposal, they include the Scalar, the inline, and the Multi–statement functions. These functions allow us to keep out code DRY and easy to use, all the while giving users to have more control of the data they get back.