Luis Valderrama

May 21, 2021

IT FDN 130 A

Module 07

https://github.com/lvalderr/DBFoundations-Module07

SQL Functions

Intro

During the course of the Seventh module, I learned the basic concepts and practical uses of User Defined Functions (UDF) available in the MS SQL RDMS. This document addresses the uses, similarities and differences between Scalar, Inline and Multistatement Functions. Additionally, the document references examples used in the database created for this assignment, *Assignment07DB_LuisValderrama, (2021) (SQL Script).*

User Defined Functions (UDF)

"Like functions in programming languages, ***SQL Server user-defined functions are routines that accept parameters, perform an action, such as a complex calculation, and return the result of that action as a value.*** *The return value can either be a single scalar value or a result set.",* *https://docs.microsoft.com/en-us/sql/relational-databases/user-defined-functions/user-defined-functions?view=sql-server-ver15, (2016) (external).* **Data Analysts use User-Defined Functions (UDF) to:**

- Apply **modular programming**. Functions are created, stored in the database, and used indefinitely. UDF can be modified independently of the program source code.
- **Enhance execution times**. Functions reduce cost of Transact-SQL code by catching the plans and reusing them for repeated executions. UDFs are not reoptimized with each use, resulting in efficient execution times.
- **Reduce network traffic**. An operation that filters data based on complex constraints that may not be expressed in a single scalar expression is expressed as function. The function is then called in the WHERE clause in order to limit the number of rows displayed.

Differences of Scalar, Inline and Multistatement Functions

- Scalar Functions return a single data value of the type defined in the RETURNS clause.
- For an Inline Function, the return scalar value is the result of a single statement, while Multistatement Functions body can contain a series of Transact-SQL statements that return a single value.
- Inline Functions cannot have BEGIN and END blocks while, Multistatement Scalar Function can have BEGIN and END blocks
- Inline Functions are better for performance than Multistatement Functions. Therefore, it is preferred to use Inline Functions whenever possible.

Before we examine the Function example created for this assignment, first I want to present the View example used in the *Assignment07DB_LuisValderrama, (2021) (SQL Script)* as shown below on (figure 1.1). This View includes a newly created, column named *CountVsPreviousCountKPI*. The new column includes a Function that compares inventory count from prior month to the current month, and will display '1' if the inventory counts increased, a '0' for no change, and '-1' if the inventory count decreased from the prior month. This is called the Key Performance Indicator or KPI.

```
GO
CREATE VIEW vProductInventoriesWithPreviousMonthCountsWithKPIs
AS
  SELECT TOP 1000000
    ProductName
  , InventoryDate
  , InventoryCount
  , PreviousMonthCount
  , [CountVsPreviousCountKPI] =
  CASE
          WHEN InventoryCount > PreviousMonthCount THEN  1
          WHEN InventoryCount = PreviousMonthCount THEN  0
          WHEN InventoryCount < PreviousMonthCount THEN -1
  END
FROM vProductInventoriesWithPreviousMonthCounts
ORDER BY ProductName;
GO
```

FIGURE 1.1: Example of a View with Function to highlight inventory count variances.

Now, the example presented below on (figure 1.2) displays the structure of the Function created to return the information based on the previously created View table as shown above on (figure 1.1). The Function uses RETURNS TABLE in order to display the data in the form of a table, followed by the attributes to display. This Function also includes an ORDER BY clause with a Function to display the KPI which is based on the inventory count variance.

```
GO
CREATE FUNCTION dbo.fProductInventoriesWithPreviousMonthCountsWithKPIs(@KPIValue INT)
RETURNS TABLE
AS
  RETURN
    SELECT TOP 1000000
          ProductName
        , InventoryDate
        , InventoryCount
        , PreviousMonthCount
        , CountVsPreviousCountKPI
    FROM vProductInventoriesWithPreviousMonthCountsWithKPIs AS v1
    WHERE CountVsPreviousCountKPI = @KPIValue
ORDER BY YEAR(CAST(v1.InventoryDate AS DATE));
GO
```

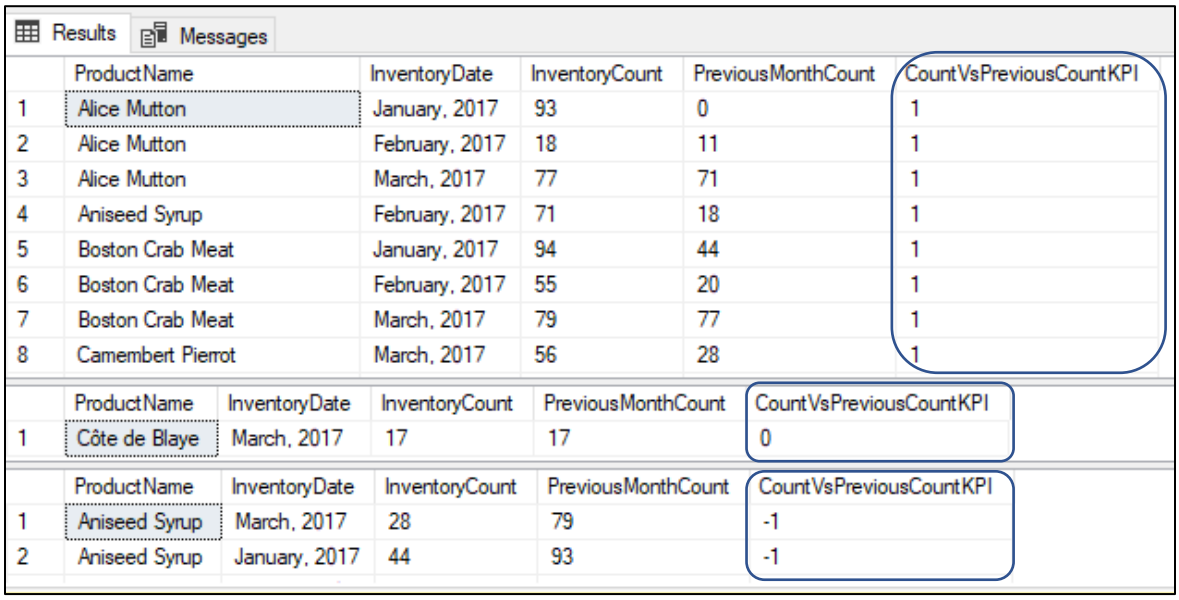
FIGURE 1.2: Example of a Function.

The inventory variance information is displayed in three separate groups when given the corresponding command (based on the KPI) in the SELECT statement. For example, (1), (0), (-1). See (figure 1.3)

```
Select * from fProductInventoriesWithPreviousMonthCountsWithKPIs(1);
Select * From fProductInventoriesWithPreviousMonthCountsWithKPIs(0);
Select * From fProductInventoriesWithPreviousMonthCountsWithKPIs(-1);
```

FIGURE 1.3: Select Statement with (1), (0), (-1).

To further illustrate the example, (figure 1.4) displays the information contained in the previously created View table by way of a Function with the capability to group the data by KPI which is based in inventory count variances from the prior month.



Results | Messages

	ProductName	InventoryDate	InventoryCount	PreviousMonthCount	CountVsPreviousCountKPI
1	Alice Mutton	January, 2017	93	0	1
2	Alice Mutton	February, 2017	18	11	1
3	Alice Mutton	March, 2017	77	71	1
4	Aniseed Syrup	February, 2017	71	18	1
5	Boston Crab Meat	January, 2017	94	44	1
6	Boston Crab Meat	February, 2017	55	20	1
7	Boston Crab Meat	March, 2017	79	77	1
8	Camembert Pierrot	March, 2017	56	28	1

	ProductName	InventoryDate	InventoryCount	PreviousMonthCount	CountVsPreviousCountKPI
1	Côte de Blaye	March, 2017	17	17	0

	ProductName	InventoryDate	InventoryCount	PreviousMonthCount	CountVsPreviousCountKPI
1	Aniseed Syrup	March, 2017	28	79	-1
2	Aniseed Syrup	January, 2017	44	93	-1

FIGURE 1.4: Results from the Function created to group the data by KPI based on inventory count variance from prior month.

Summary

To recap, the seventh module taught me the basic concepts and practical uses of Functions available in the MS SQL RDMS and differences between Scalar, Inline and Multistatement User Defined Functions. The database created for this assignment, Assignment07DB_LuisValderrama, applies Functionalities learned throughout prior modules and the practical application of SQL Functions.