Fikru Abasori

Nov30,2021

IT FDN 130A

Assignment 07

<https://github.com/fikru2021/DBFoundations-Module07>.

**Intro**

Working on this assignment and reading the web articles, I have learned about different SQL functions. Specifically , UDF ( user defined functions) and when to use UDF , Table-Valued functions , Simple in-line table valued functions, Multi-statement table valued functions. In addition, I have learned the differences between Scalar, Inline, and Multi-Statement Functions.

When you would use a SQL UDF:

Just like other 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. User-defined-functions ( UDF) are usually used because:

1. They allow modular programming: that is, You can create the function once, store it in the database, and call it any number of times in your program. User-defined functions can be modified independently of the program source code.
2. They allow faster execution: Similar to stored procedures, Transact-SQL user-defined functions reduce the compilation cost of Transact-SQL code by caching the plans and reusing them for repeated executions. This means the user-defined function does not need to be reparsed and reoptimized with each use resulting in much faster execution times.
3. They help reduce network traffic: An operation that filters data based on some complex constraint that cannot be expressed in a single scalar expression can be expressed as a function. The function can then be invoked in the WHERE clause to reduce the number of rows sent to the client.

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

User-defined scalar functions return a single data value of the type defined in the RETURNS clause. For an inline scalar function, the returned scalar value is the result of a single statement. For a multistatement scalar function, the function body can contain a series of Transact-SQL statements that return the single value. The return type can be any data type except text, ntext, image, cursor, and timestamp.

<https://docs.microsoft.com/en-us/sql/relational-databases/user-defined-functions/user-defined-functions?view=sql-server-ver15>. ( External site)

-- Question 1 (5% of pts):

-- show a list of Product names and the price of each product.

-- Use a function to format the price as US dollars.

-- Order the result by the product name.

Table

Description automatically generated

Figure 1: Results of Question 1

-- Question 2 (10% of pts):

-- Show a list of Category and Product names, and the price of each product.

-- Use a function to format the price as US dollars.

-- Order the result by the Category and Product.

Table

Description automatically generated

Figure 2: Results of Question 2

-- Question 3 (10% of pts):

-- Use functions to show a list of Product names, each Inventory Date, and the Inventory Count.

-- Format the date like 'January, 2017'.

-- Order the results by the Product and Date.

Table

Description automatically generated

Figure 3: Results of Question 3

-- Question 4 (10% of pts):

-- CREATE A VIEW called vProductInventories

-- Shows a list of Product names, each Inventory Date, and the Inventory Count.

-- Format the date like 'January, 2017'.

-- Order the results by the Product and Date. (Note: The result is the same as the previous question)

-- Check that it works: Select \* From vProductInventories;

Table

Description automatically generated

Figure 4: Results of Question 4

-- Question 5 (10% of pts):

-- CREATE A VIEW called vCategoryInventories.

-- Shows a list of Category names, Inventory Dates, and a TOTAL Inventory Count BY CATEGORY.

-- Format the date like 'January, 2017'.

-- Order the results by the Product and Date.

-- Check that it works: Select \* From vCategoryInventories;

Table

Description automatically generated

Figure 5: Results of Question 5

-- Question 6 (15% of pts):

-- CREATE ANOTHER VIEW called vProductInventoriesWithPreviouMonthCounts.

-- Show a list of Product names, Inventory Dates, Inventory Count, AND the Previous Month Count.

-- Use functions to set any January NULL counts to zero.

-- Order the results by the Product and Date.

-- This new view must use your vProductInventories view.

Table

Description automatically generated

Figure 6: Results of Question 6

-- Question 7 (15% of pts):

-- CREATE a VIEW called vProductInventoriesWithPreviousMonthCountsWithKPIs.

-- Show columns for the Product names, Inventory Dates, Inventory Count, Previous Month Count.

-- The Previous Month Count is a KPI. The result can show only KPIs with a value of either 1, 0, or -1.

-- Display months with increased counts as 1, same counts as 0, and decreased counts as -1.

-- Varify that the results are ordered by the Product and Date.

-- Important: This new view must use your vProductInventoriesWithPreviousMonthCounts view!

-- Check that it works: Select \* From vProductInventoriesWithPreviousMonthCountsWithKPIs;

Table

Description automatically generated

Figure 7: Results of Question 7

-- Question 8 (25% of pts):

-- CREATE a User Defined Function (UDF) called fProductInventoriesWithPreviousMonthCountsWithKPIs.

-- Show columns for the Product names, Inventory Dates, Inventory Count, the Previous Month Count.

-- The Previous Month Count is a KPI. The result can show only KPIs with a value of either 1, 0, or -1.

-- Display months with increased counts as 1, same counts as 0, and decreased counts as -1.

-- The function must use the ProductInventoriesWithPreviousMonthCountsWithKPIs view.

-- Varify that the results are ordered by the Product and Date.

-- Check that it works:

Select \* From fProductInventoriesWithPreviousMonthCountsWithKPIs(1);

Select \* From fProductInventoriesWithPreviousMonthCountsWithKPIs(0);

Select \* From fProductInventoriesWithPreviousMonthCountsWithKPIs(-1);

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Figure 8: Results of Question 8

**Summary**

I have learned about different SQL functions. Specifically , UDF ( user defined functions) and when to use UDF , Table-Valued functions , Simple in-line table valued functions, Multi-statement table valued functions. In addition, I have learned the differences between Scalar, Inline, and Multi-Statement Functions.