Name: JohnRobert Wilson
Date: November 25, 2023
Course: IT Foundations of Database Management

Assignment 07 - Functions

Introduction

This week we continued from module 6 with functions, another way to store repeatable complex operations within a database for use in reporting. There are quite a few pre-defined functions from Microsoft that allow for manipulation of data within a table. In addition to these we worked with User Defined Function (UDF) where we created our own functions composed of a series of operations.

Explain when you would use a SQL UDF.

Functions often perform the same outcomes as view. However like a view, functions can be built and structured to reduce manual inputs. A view stores a select statement so that it can be used multiple times, where as a UDF also stores a series of operations to generate an outcome. This can be done to generate a report, or run a series of operations per line of a database. In my personal work, I will likely go with UDF's to generate reports for user task success and failure rates since my audience only needs to the report, and it will be easier for me to show them how to call a function than to structure a view with specific clauses. For example all workflows with a task success less than 70%. The user's won't need to worry about structure a where clause for this.

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

Scalar, Inline, and Multi-Statement functions differ on where and what is returned. Scalar functions generate a single value where as a Multi-Statement functions will return multiple values as a table. These functions can be made up of multiple sub-functions, or inline functions. It's important when working with these to know the order and sequence of the nested functions. This was important in questions 6, when we combined IsNull, and Lag functions.

```
Select
    Product
    , [Month, YYYY]
    , [Inventory Count]
    , IsNull(
        LAG([Inventory Count]) OVER (ORDER BY Product, MONTH([Month, YYYY])), 0
        ) AS 'Previous Month Count'
    FROM vProductInventories
    ORDER BY Product, MONTH([Month, YYYY])
;
GO
```

Appendix

Built in functions

GetDate ()
DatePart(mm)(dd)(yyyy) Extract any part of a date that is specified.
GetDate
Sum()
Max()
Min()
Avg()
Count()
Cast() Cast data into a format for example Cast('1' as INT) adds 1 as an integer. Can also do as decimal, and as nVarChar(50)

```
--( Conversions )--
Select Cast('1' as int), Cast('1' as decimal(5,2)), Cast(1 as nVarchar(50));
Select Convert(int,'1'), Convert(decimal(3,2),'1'), Convert(nvarchar(50), 1);
```

Convert()

- Functions will require dbo. And in the select statement of a function it will need the () at the end. This will work with direct inputs… but the sub-query needs to be set as an

separate clause.

```
    );
go
Declare @ID int
Select @ID = CategoryID from Northwind.dbo.Categories Where CategoryName = 'Dariy'
Select * From dbo.fProducts(@ID); -- 12 rows
go
```