Qiaoyi Yang
Nov 29th, 2021
IT Foundations of Database Management
Assignment 07

Assignment 07 – Functions

Introduction

The seventh assignment of this class is about how to uses SQL Functions to retrieve information from a database. I personally think it's the most different assignment so far especially for the User Defined Functions. I have learned combine functions to create better looking results. Also, Logical Functions allow me to look for a condition the evaluates to true or false and then return an appropriate value such as The String function, The Format Function, The Immediate IF function etc... Partitioned or Windowed Functions can be used to group data differently than the standard Group By clause. It's very useful for creating report. Besides the SQL Server's built-in functions, there are customized functions called User Defined Functions. There are two types of functions; functions that return a table of values and functions that return a single value. Scalars Functions return a single (scalar) value as an expression. Inline table valued function refers to a TVF where the function body just contains one line of select statement. The multi-statement table-valued function is slightly more complicated than the other two types of functions because it uses multiple statements to build the table that is returned to the calling statement.

Questions

1. ***Explain when you would use a SQL UDF.***

   UDF is a very powerful tool in databases， I would use it when I need to use one or more values from different tables in a join operation where some type of calculation needs to be done and an aggregation returned.

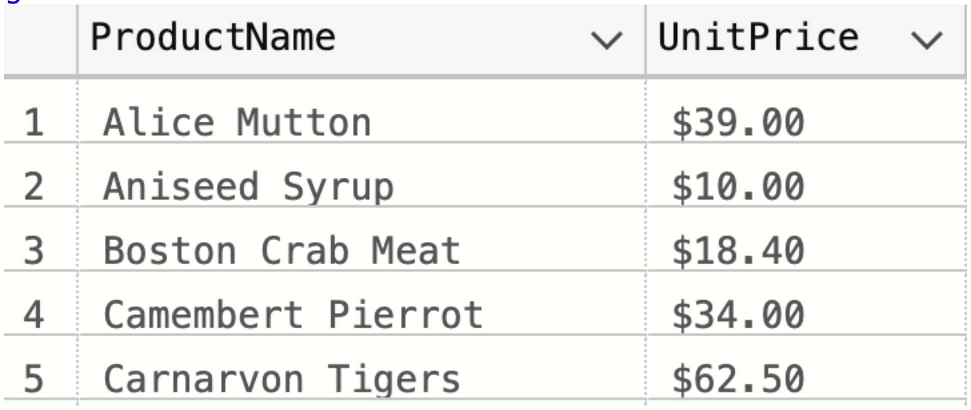
2. ***Explain are the differences between Scalar, Inline, and Multi-Statement Functions.***

   *A **scalar function** returns a single value of the data type referenced in the RETURNS clause of the CREATE FUNCTION statement. The returned data can be of any type except text, ntext, image, cursor, or timestamp.*

   Inline table valued function refers to a TVF where the function body **just contains one line of select statement.** There is **not return variable**. Multi-statement table valued function refers to a TVF where it has a return table variable. Inside the function body, there will be statements populating this table variable.

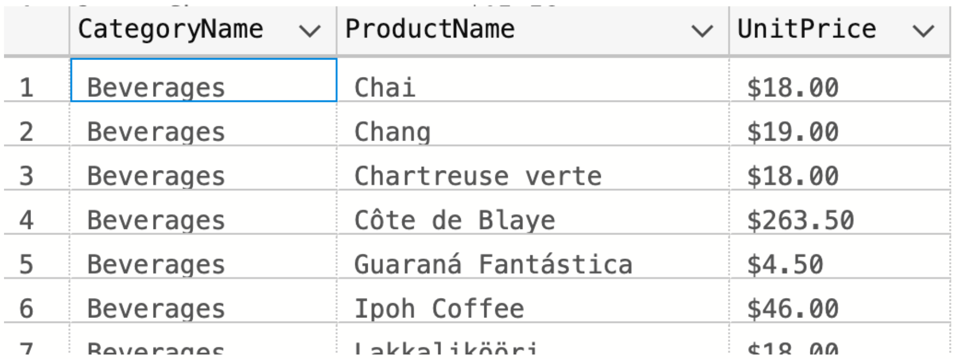
   The multi-statement table-valued function is slightly more complicated than the other two types of functions **because** it uses multiple statements to build the table that is returned to the calling statement. **Unlike the inline table-valued function, a table variable must be explicitly declared and defined.**

```sql
-- 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.

-- <Put Your Code Here> --
Select ProductName,
    Format (UnitPrice, 'c','en-US') as 'UnitPrice'
From Products
Order By ProductName
go
```



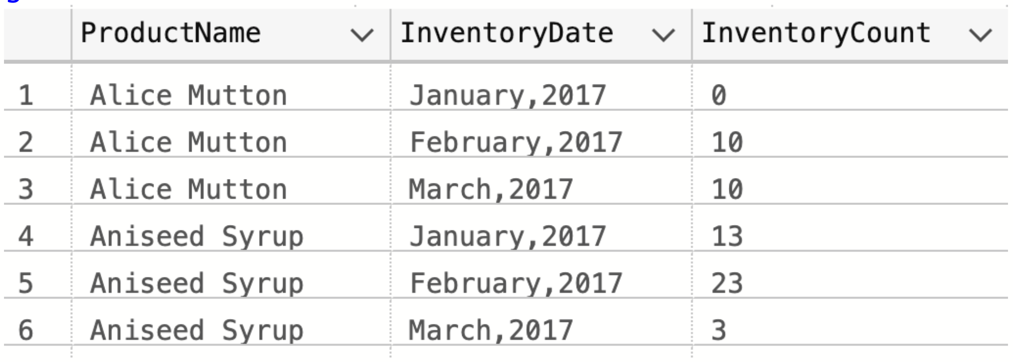
	ProductName	UnitPrice
1	Alice Mutton	$39.00
2	Aniseed Syrup	$10.00
3	Boston Crab Meat	$18.40
4	Camembert Pierrot	$34.00
5	Carnarvon Tigers	$62.50

```sql
-- 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.
-- <Put Your Code Here> --
Select CategoryName, ProductName,
    Format (UnitPrice, 'c','en-US') as 'UnitPrice'
From Categories as c
Join Products as p
On c.CategoryID = p.CategoryID
Order By CategoryName, ProductName
go
```



	CategoryName	ProductName	UnitPrice
1	Beverages	Chai	$18.00
2	Beverages	Chang	$19.00
3	Beverages	Chartreuse verte	$18.00
4	Beverages	Côte de Blaye	$263.50
5	Beverages	Guaraná Fantástica	$4.50
6	Beverages	Ipoh Coffee	$46.00
7	Beverages	Lakkalikööri	$18.00

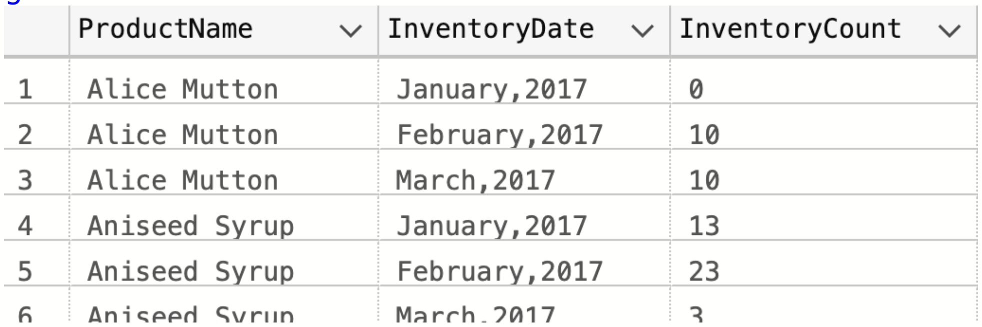
```
-- Question 3 (10% of pts):
-- Use functions to show a list of Product names, each Inventory
Date, and the Inventory Count.
-- Format the date like 'January, 2017'.
-- Order the results by the Product and Date.

-- <Put Your Code Here> --
Select ProductName,
       DATENAME(month,InventoryDate) + ',' +
DATENAME(year,InventoryDate) AS InventoryDate,
       [Count] as InventoryCount
From Products as p
Join Inventories as i
On p.ProductID = i.ProductID
Order By 1,Cast(InventoryDate as Date);
go
```



	ProductName	InventoryDate	InventoryCount
1	Alice Mutton	January,2017	0
2	Alice Mutton	February,2017	10
3	Alice Mutton	March,2017	10
4	Aniseed Syrup	January,2017	13
5	Aniseed Syrup	February,2017	23
6	Aniseed Syrup	March,2017	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'.
```

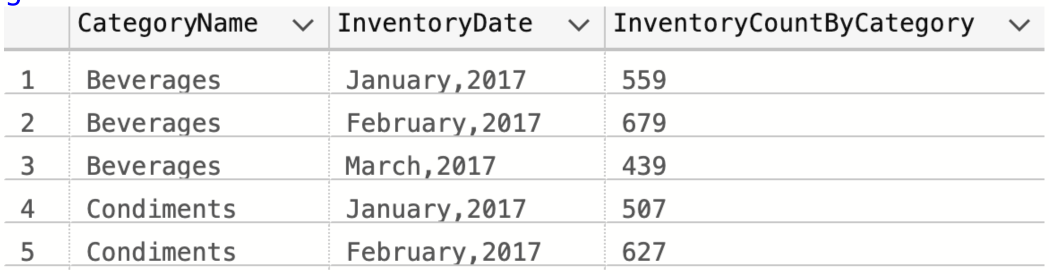
```sql
-- Order the results by the Product and Date.

-- <Put Your Code Here> --
Create View vProductInventories
AS
Select TOP 1000000000
    ProductName,
    DATENAME(month,InventoryDate) + ',' +
DATENAME(year,InventoryDate) as InventoryDate,
    [Count] as InventoryCount
From Products as p
Join Inventories as i
On p.ProductID = i.ProductID
Order By 1,Cast(InventoryDate as Date);
go

-- Check that it works: Select * From vProductInventories;
Select * From vProductInventories
go
```



	ProductName	InventoryDate	InventoryCount
1	Alice Mutton	January,2017	0
2	Alice Mutton	February,2017	10
3	Alice Mutton	March,2017	10
4	Aniseed Syrup	January,2017	13
5	Aniseed Syrup	February,2017	23
6	Aniseed Syrup	March,2017	3

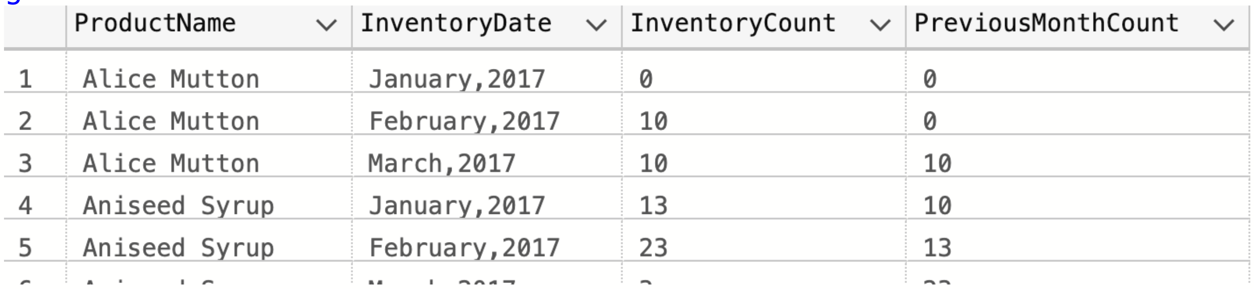
```sql
-- Question 5 (10% of pts):
-- CREATE A VIEW called vCategoryInventories.
-- Shows a list of Category names, Inventory Dates, and a TOTAL
Inventory Count BY CATEGORY
-- Format the date like 'January, 2017'.
-- Order the results by the Product and Date.

-- <Put Your Code Here> --
Create View vCategoryInventories
AS
```

```
Select TOP 1000000000
    c.CategoryName,
    DATENAME(month,i.InventoryDate) + ',' +
DATENAME(year,i.InventoryDate) as InventoryDate,
    SUM(i.[Count]) as InventoryCountByCategory
From Categories as c
Join Products as p
On c.CategoryID = p.CategoryID
Join Inventories as i
On p.ProductID = i.ProductID
Group By CategoryName, InventoryDate
Order By 1,Cast(InventoryDate as Date);
go
-- Check that it works: Select * From vCategoryInventories;
Select * From vCategoryInventories
go
```



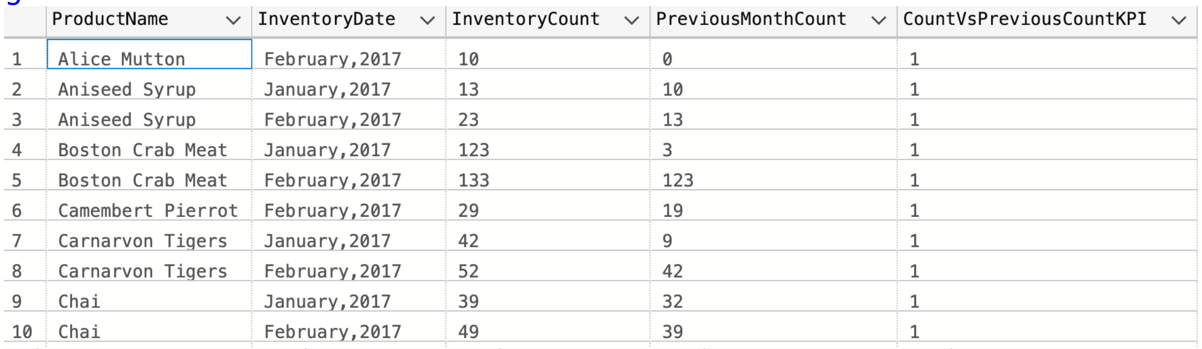
	CategoryName	InventoryDate	InventoryCountByCategory
1	Beverages	January,2017	559
2	Beverages	February,2017	679
3	Beverages	March,2017	439
4	Condiments	January,2017	507
5	Condiments	February,2017	627

```
-- Question 6 (10% 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.

-- <Put Your Code Here> --
Create View vProductInventoriesWithPreviouMonthCounts
AS
Select TOP 1000000000
        ProductName,
        InventoryDate,
        InventoryCount,
        IsNull(Lag(InventoryCount) Over (Order by ProductName,
Month(InventoryDate)),0) as PreviousMonthCount
From vProductInventories as vci
Order By 1,Cast(InventoryDate as Date);
go
-- Check that it works: Select * From
vProductInventoriesWithPreviousMonthCounts;
Select * From vProductInventoriesWithPreviouMonthCounts
go
```

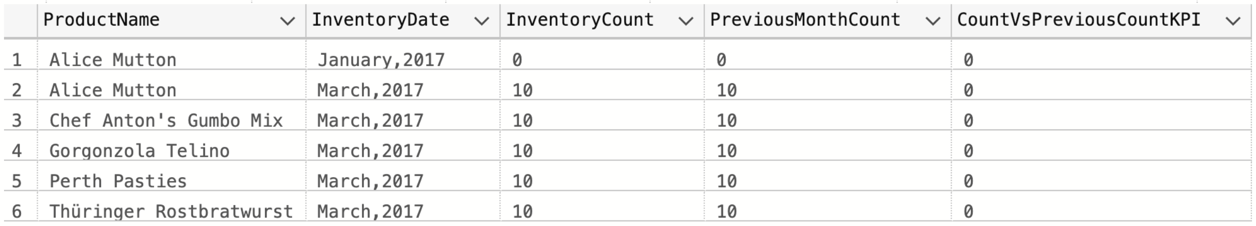


	ProductName	InventoryDate	InventoryCount	PreviousMonthCount
1	Alice Mutton	January,2017	0	0
2	Alice Mutton	February,2017	10	0
3	Alice Mutton	March,2017	10	10
4	Aniseed Syrup	January,2017	13	10
5	Aniseed Syrup	February,2017	23	13

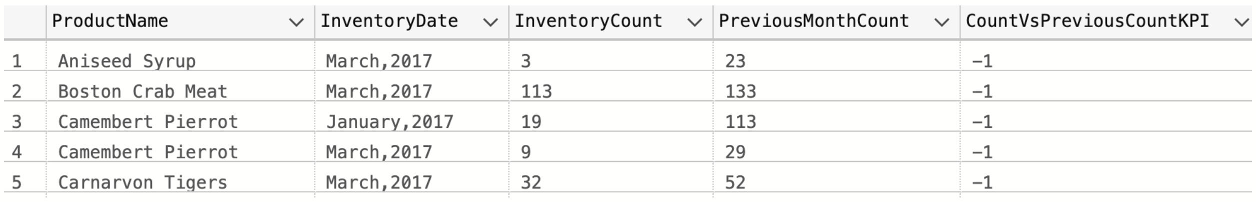
```
-- Question 7 (15% of pts):
-- CREATE a VIEW called
vProductInventoriesWithPreviousMonthCountsWithKPIs.
-- Show columns for the Product names, Inventory Dates,
Inventory Count, Previous Month Count.
-- The Previous Month Count is a KPI. The result can show only
KPIs with a value of either 1, 0, or -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.

-- <Put Your Code Here> --
Create View vProductInventoriesWithPreviousMonthCountsWithKPIs
AS
Select TOP 1000000000
        ProductName,
        InventoryDate,
        InventoryCount,
        PreviousMonthCount,
        CountVsPreviousCountKPI = Case
            When InventoryCount > PreviousMonthCount then 1
            When InventoryCount = PreviousMonthCount then 0
            When InventoryCount < PreviousMonthCount then -1
            End
From vProductInventoriesWithPreviouMonthCounts
Order By 1,Cast(InventoryDate as Date);
go
-- Important: This new view must use your
vProductInventoriesWithPreviousMonthCounts view!
-- Check that it works: Select * From
vProductInventoriesWithPreviousMonthCountsWithKPIs;
go
Select * From vProductInventoriesWithPreviousMonthCountsWithKPIs

-- 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.

-- <Put Your Code Here> --
Create Function
fProductInventoriesWithPreviousMonthCountsWithKPIs (@KPIValue
INT)
Returns Table
AS
Return Select
        ProductName,
        InventoryDate,
        InventoryCount,
        PreviousMonthCount,
        CountVsPreviousCountKPI
From vProductInventoriesWithPreviousMonthCountsWithKPIs
Where CountVsPreviousCountKPI = @KPIValue

go

--Check that it works:
Select * From
fProductInventoriesWithPreviousMonthCountsWithKPIs(1);
Select * From
fProductInventoriesWithPreviousMonthCountsWithKPIs(0);
Select * From
fProductInventoriesWithPreviousMonthCountsWithKPIs(-1);

go
```



	ProductName	InventoryDate	InventoryCount	PreviousMonthCount	CountVsPreviousCountKPI
1	Alice Mutton	February,2017	10	0	1
2	Aniseed Syrup	January,2017	13	10	1
3	Aniseed Syrup	February,2017	23	13	1
4	Boston Crab Meat	January,2017	123	3	1
5	Boston Crab Meat	February,2017	133	123	1
6	Camembert Pierrot	February,2017	29	19	1
7	Carnarvon Tigers	January,2017	42	9	1
8	Carnarvon Tigers	February,2017	52	42	1
9	Chai	January,2017	39	32	1
10	Chai	February,2017	49	39	1



	ProductName	InventoryDate	InventoryCount	PreviousMonthCount	CountVsPreviousCountKPI
1	Alice Mutton	January,2017	0	0	0
2	Alice Mutton	March,2017	10	10	0
3	Chef Anton's Gumbo Mix	March,2017	10	10	0
4	Gorgonzola Telino	March,2017	10	10	0
5	Perth Pasties	March,2017	10	10	0
6	Thüringer Rostbratwurst	March,2017	10	10	0



	ProductName	InventoryDate	InventoryCount	PreviousMonthCount	CountVsPreviousCountKPI
1	Aniseed Syrup	March,2017	3	23	-1
2	Boston Crab Meat	March,2017	113	133	-1
3	Camembert Pierrot	January,2017	19	113	-1
4	Camembert Pierrot	March,2017	9	29	-1
5	Carnarvon Tigers	March,2017	32	52	-1

```
/***************************************************************
************************/
```

Summary

This week's coding assignment is challenging. I learned a lot about functions in SQL. It’s a powerful tool can be using create customize reports with detail. I enjoy the learning for this week.