

# Simple Function

Sunday, May 28, 2017 4:31 PM

The screenshot shows a SQL script in a window titled 'SimpleFunction.sql...ls (LABS\delip (55))'. The script contains the following T-SQL code:

```
USE Schools;
GO

Create Function fnCube(@num float)
Returns float
As
Begin
    Return @num * @num * @num
End
Go

Select dbo.fnCube(3)

Drop Function fnCube;
GO
```

Below the script editor, the 'Results' tab is active, displaying the output of the 'Select' statement. The output is a single row with the value 27.

	(No column name)
1	27

# Scaler Function

Saturday, May 27, 2017 11:06 AM

The screenshot displays the SQL Server Enterprise Manager interface. The main window shows a SQL script for creating two scalar functions. The script is as follows:

```
USE Northwind;
GO

IF OBJECT_ID (N'dbo.fnTotalOnOrder', N'FN') IS NOT NULL
    DROP FUNCTION fnTotalOnOrder;
GO

IF OBJECT_ID (N'dbo.fnTotalInStock') IS NOT NULL
    DROP FUNCTION fnTotalInStock;
GO

CREATE FUNCTION dbo.fnTotalOnOrder(@CategoryID int)
RETURNS int
AS
BEGIN
    DECLARE @retTotal int;
    SELECT @retTotal = SUM(UnitPrice * UnitsOnOrder)
    FROM Products
    WHERE (CategoryID = @CategoryID)
    IF (@retTotal IS NULL)
        SET @retTotal = 0;
    RETURN @retTotal;
END;
GO

CREATE FUNCTION dbo.fnTotalInStock(@CategoryID int)
RETURNS int
AS
BEGIN
    DECLARE @retTotal int;
    SELECT @retTotal = SUM(UnitPrice * UnitsInStock)
    FROM Products
    WHERE (CategoryID = @CategoryID)
    IF (@retTotal IS NULL)
        SET @retTotal = 0;
    RETURN @retTotal;
END;
GO
```

The right-hand pane shows the 'Programmability' folder expanded, with 'Functions' selected. Under 'Functions', the following functions are listed:

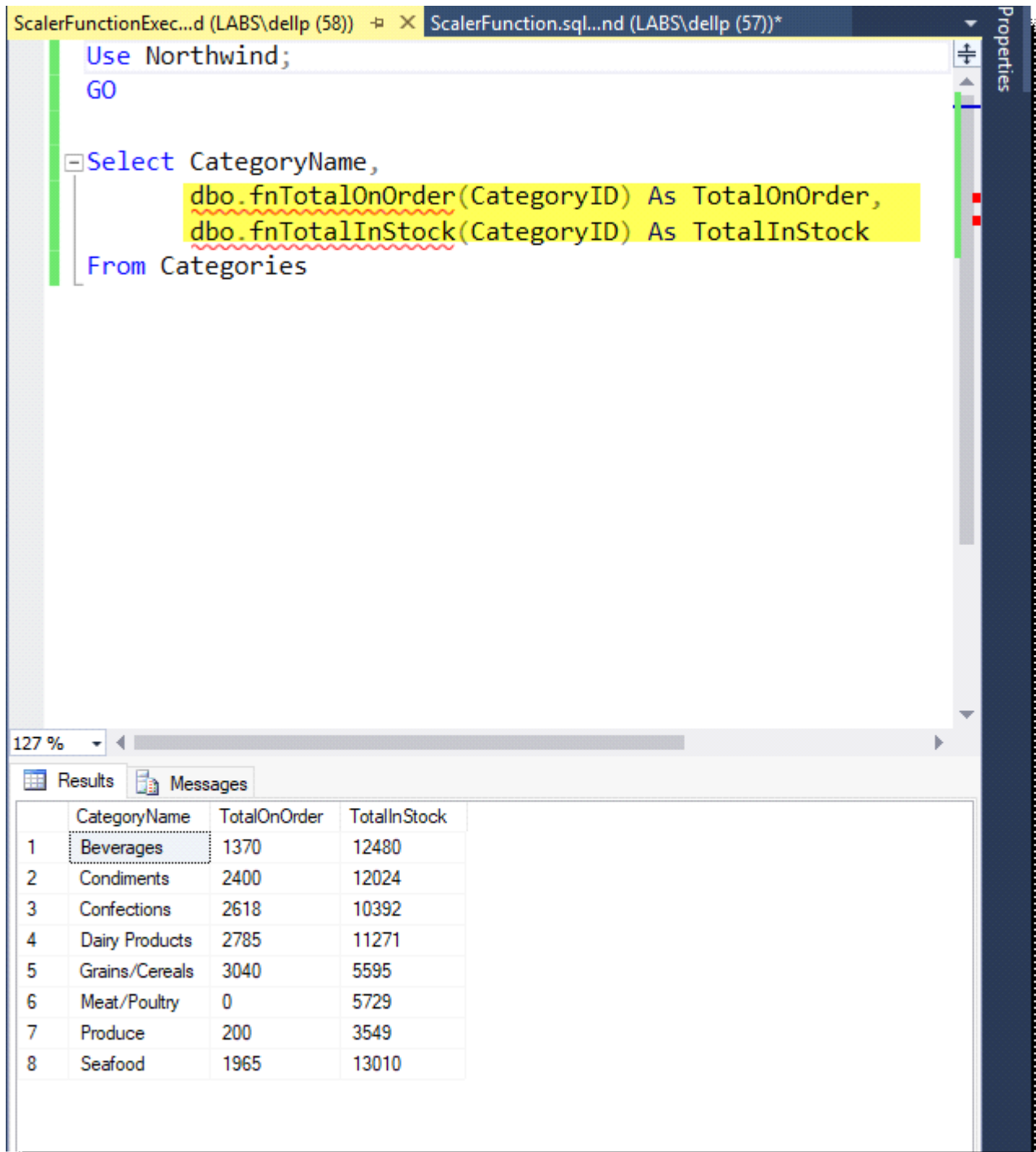
- Table-valued Functions
- Scalar-valued Functions
  - dbo.fnTotalInStock
  - dbo.fnTotalOnOrder
- Aggregate Functions
- System Functions
- Database Triggers
- Assemblies

The bottom status bar indicates a zoom level of 115%.



# Execute Function

Saturday, May 27, 2017 9:18 PM



The screenshot shows a SQL Server Enterprise Manager window with two tabs: "ScalerFunctionExec...d (LABS\delip (58))" and "ScalerFunction.sql...nd (LABS\delip (57))\*". The active tab displays a T-SQL query in the query editor. The query is as follows:

```
Use Northwind;  
GO  
  
Select CategoryName,  
       dbo.fnTotalOnOrder(CategoryID) As TotalOnOrder,  
       dbo.fnTotalInStock(CategoryID) As TotalInStock  
From Categories
```

The query results are displayed in the "Results" pane at the bottom of the window. The results are presented in a table with four columns: "CategoryName", "TotalOnOrder", and "TotalInStock". The table contains eight rows of data, corresponding to the categories in the Northwind database. The "CategoryName" column is highlighted in the first row.

	CategoryName	TotalOnOrder	TotalInStock
1	Beverages	1370	12480
2	Condiments	2400	12024
3	Confections	2618	10392
4	Dairy Products	2785	11271
5	Grains/Cereals	3040	5595
6	Meat/Poultry	0	5729
7	Produce	200	3549
8	Seafood	1965	13010

# Table Value Function

Sunday, May 28, 2017 10:05 AM

The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows the SQL script for creating and testing a table-valued function named `dbo.fnCustomersOrdersDateRange`. The script includes a `USE Northwind` statement, a `GO` command, the `CREATE FUNCTION` definition, and a `SELECT` statement to test the function with date ranges '19970101' and '19970131'. The bottom pane shows the results of the query, displaying a table with columns `Company Name`, `OrderDate`, and `TotalSale`.

```
USE Northwind
GO

CREATE FUNCTION dbo.fnCustomersOrdersDateRange
(
    @StartDate as Date,
    @EndDate as Date
)
RETURNS TABLE
AS
RETURN
(
    SELECT c.CompanyName, o.OrderDate,
           (od.UnitPrice*od.Quantity) As TotalSale
    FROM Customers As c
    Inner Join Orders As o on c.CustomerID = o.CustomerID
    Inner Join [Order Details] As od on o.OrderID = od.OrderID
    WHERE o.OrderDate Between @StartDate AND @EndDate);
GO

SELECT * FROM dbo.fnCustomersOrdersDateRange('19970101','19970131');
GO

Drop Function dbo.fnCustomersOrdersDateRange;
GO
```

	Company Name	OrderDate	TotalSale
1	Eastern Connection	1997-01-01 00:00:00.000	2079.00
2	Eastern Connection	1997-01-01 00:00:00.000	504.00
3	Eastern Connection	1997-01-01 00:00:00.000	480.00
4	Rattlesnake Canyon Grocery	1997-01-01 00:00:00.000	372.60
5	Rattlesnake Canyon Grocery	1997-01-01 00:00:00.000	2128.00
6	Rattlesnake Canyon Grocery	1997-01-01 00:00:00.000	336.00
7	Rattlesnake Canyon Grocery	1997-01-01 00:00:00.000	1032.00
8	Ernst Handel	1997-01-02 00:00:00.000	432.00
9	Ernst Handel	1997-01-02 00:00:00.000	2281.50
10	Ernst Handel	1997-01-03 00:00:00.000	291.90
11	Ernst Handel	1997-01-03 00:00:00.000	714.00
12	Magazzini Alimentari Riuniti	1997-01-03 00:00:00.000	747.00
13	Magazzini Alimentari Riuniti	1997-01-03 00:00:00.000	448.00
14	Magazzini Alimentari Riuniti	1997-01-03 00:00:00.000	480.00
15	LINO-Delicateses	1997-01-06 00:00:00.000	400.00
16	Queen Cozinha	1997-01-07 00:00:00.000	144.00
17	Queen Cozinha	1997-01-07 00:00:00.000	240.00

# Table Multi-Statement

Sunday, May 28, 2017 10:11 AM

```
TableMultiStatemen...d (LABS\delip (62))* X TableValuedFunctio...s (LABS\delip (60)) TableValuedFunctio...s (LABS\delip (58))
4  --Get Last Orders By Date
5  -----
6  CREATE FUNCTION dbo.fnGetLastOrders()
7  RETURNS @CustomerOrder TABLE
8  (
9      SaleOrderID INT NOT NULL,
10     CustomerID varchar(5) NOT NULL,
11     ProductName varchar(50) NULL,
12     OrderDate DATETIME NOT NULL,
13     Quantity INT NOT NULL,
14     Price smallMoney NOT NULL
15 )
16 AS
17 BEGIN
18     INSERT @CustomerOrder
19     SELECT o.OrderID, o.CustomerID, p.ProductName, o.OrderDate, od.Quantity, od.UnitPrice
20     FROM Orders o
21         INNER JOIN [Order Details] od
22             ON o.OrderID = od.OrderID
23         INNER JOIN Products p
24             ON od.ProductID = p.ProductID
25     WHERE o.OrderDate = (
26         Select Max(md.OrderDate)
27         FROM Orders As md
28     )
29     RETURN
30 END
31 GO
32
33 Select * From dbo.fnGetLastOrders();
34 Drop Function dbo.fnGetLastOrders;
```

# Category Function

Tuesday, June 20, 2017 9:06 PM

The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'CrossApply.sql - (L...nd (LABS\delip (55))\*)' and 'CategoryFunction.s...d (LABS\delip (66))\*'. The script in the second tab is as follows:

```
1 Use Northwind;
2 Go
3
4 Create Function dbo.fnGetCategoryID(@Name VarChar(50))
5     Returns Int
6 As
7     Begin
8         Return (Select CategoryID
9                 From Categories
10                Where CategoryName = @Name)
11     End
12 Go
13
14 Select CategoryID, ProductName
15 From Products
16 Where CategoryID = dbo.fnGetCategoryID('Beverages');
17 Go
18
19 Drop Function dbo.fnGetCategoryID;
20 Go
```

Below the script, the 'Results' tab is active, displaying a table with 12 rows of data. The first row is highlighted. The table has two columns: 'CategoryID' and 'ProductName'.

	CategoryID	ProductName
1	1	Chai
2	1	Chang
3	1	Guaraná Fantástica
4	1	Sasquatch Ale
5	1	Steeleye Stout
6	1	Côte de Blaye
7	1	Chartreuse verte
8	1	Ipoh Coffee
9	1	Laughing Lumberjack Lager
10	1	Outback Lager
11	1	Rhönbräu Klosterbier
12	1	Lakkalikööri



# Cross Apply

Tuesday, June 20, 2017 9:07 PM

SimpleFunction.sql...ls (LABS\delip (55)) CrossApply.sql - (l...nd (LABS\delip (55))\*

```
1 Use Northwind;
2 Go
3 --This may be needed
4 -- Alter database Northwind
5 -- Set Compatibility_level = 90
6
7 CREATE FUNCTION dbo.fn_TopProductsBySupplier
8 (@SupplierID int)
9 RETURNS TABLE
10 AS
11 RETURN
12     SELECT TOP (3) ProductID, ProductName, UnitPrice
13     FROM Products
14     WHERE SupplierID = @SupplierID
15     ORDER BY UnitPrice DESC;
16 Go
17
18 --Test the function
19 SELECT * FROM dbo.fn_TopProductsBySupplier(2);
20
21 -- Test with CROSS APPLY with Function
22 SELECT s.SupplierID, s.CompanyName, p.ProductName, p.UnitPrice
23 FROM Suppliers AS s
24     CROSS APPLY dbo.fn_TopProductsBySupplier(s.SupplierID) p
25 ORDER BY S.SupplierID ASC, p.UnitPrice DESC;
26
27 Drop Function fn_TopProductsBySupplier;
28
29
```

102 %


Results Messages

	ProductID	ProductName	UnitPrice
1	4	Chef Anton's Cajun Seasoning	22.00
2	5	Chef Anton's Gumbo Mix	21.35
3	65	Louisiana Fiery Hot Pepper Sauce	21.05

	SupplierID	CompanyName	ProductName	UnitPrice
1	1	Exotic Liquids	Chang	19.00
2	1	Exotic Liquids	Chai	18.00
3	1	Exotic Liquids	Aniseed Syrup	10.00
4	2	New Orleans Cajun Delights	Chef Anton's Cajun Seasoning	22.00
5	2	New Orleans Cajun Delights	Chef Anton's Gumbo Mix	21.35
6	2	New Orleans Caiun Delights	Louisiana Fiery Hot Pepper Sauce	21.05



4	2	New Orleans Cajun Delights	Chef Anton's Cajun Seasoning	22.00
5	2	New Orleans Cajun Delights	Chef Anton's Gumbo Mix	21.35
6	2	New Orleans Cajun Delights	Louisiana Fiery Hot Pepper Sauce	21.05
7	3	Grandma Kelly's Homestead	Northwoods Cranberry Sauce	40.00
8	3	Grandma Kelly's Homestead	Uncle Bob's Organic Dried Pears	30.00

 Query executed successfully. | (local)\sqlexpress01 (13.0 ... | LABS\dellp (55) | Northwind | 00:00:00 | 74 rows