User Defined Functions

In comparison to Stored Procedures, SQL provides another way to add functionality with SQL functions. We have already seen system defined functions such as AVG(), MAX(), and SUM() become extremely useful. Creating a function with you own custom defined programmability is useful way to reuse functionality.

SQL User Defined functions can take in parameters and return an output.

The output can be in the form of a scalar value (one value) or a table.

We can see the functions we create under the “Programmability” tab in Management Studio.

SQL Management Studio



Once created, functions will reside on the server to be use by anyone with access to the function.

A simplified syntax to create a Scalar function is

CREATE FUNCTION [ schema\_name. ] function\_name

( [ { @parameter\_name [ AS ][ type\_schema\_name. ] parameter\_data\_type

    [ ,...n ]

  ]

)

RETURNS return\_data\_type

    [ AS ]

    BEGIN

        function\_body

        RETURN scalar\_expression

    END

[ ; ]

For this example, we will create our own scalar function. Suppose we have a table called **EmployeeTransfers**.

This table has the following columns: EmpId, EmpBasePay, EmpTransferState, EmpFiliingStatus.

Use the Script below to setup the data:

-- Begin Script --

-- Drop uncomment if needed

--drop table EmployeeTransfers

--GO

create table EmployeeTransfers

(

 EmpIdSSN char(9),

 EmpBasePay decimal(16,4),

 EmpTransferState char(2),

 EmpFederalFilingStatus char(1)

)

GO

insert into EmployeeTransfers

values

('555342121',60000.00, 'CA', 'S'),

('552342120',70000.00, 'NY', 'M'),

('552342120',87000.00, 'AZ', 'M'),

('456772332',55000.00, 'CT', 'S'),

('332223445',33000.00, 'NJ', 'M'),

('550667788',54000.00, 'DC', 'M')

GO

-- End Script --

This table has a list of employees which are transferring to a new state and their base pay before the transfer. We would like to create a function to calculate their new salary in the new state.

Suppose that we have another table called **StateTransferRates** which shows the percentage of change for the particular state. Use the script below to setup this table.

--- BEGIN SCRIPT ---

-- Drop uncomment if needed

--drop table StateTransferRates

--go

create table StateTransferRates

(

 State char(2),

 Rate decimal(16,4)

)

GO

insert into StateTransferRates

values

('CA',5.5),

('NY', 10.5),

('AZ',3.2),

('CT',2.2),

('DC',4.0),

('NJ',3.9)

GO

--- END SCRIPT ------

Now, we would like to create a scalar function called **CalculateTransferPay** which takes in 2 parameters: 1) the base pay and 2) the state. The function will calculate and return the new base pay for that state.

So for example, employee 555342121 is transferring to CA, so his new wage rate should be calculated as:

$60,000 \* 5.5% = $3,300

So his new base rate would be = $63,300. Our new function should reflect that.

Our new function will be create as such:

--- BEGIN SCRIPT ---

drop function dbo.CalculateTransferPay

go

create function dbo.CalculateTransferPay

(

 @basePay decimal(16,4), --These are the parameters we will pass

 @state char(2)

) returns decimal(16,4) --This is the scalar which would be returned

as

begin

 declare @newbasePay decimal(16,4)

 declare @rate decimal(16,4)

 --grab the rate from our stateTransferRates table

 select @rate = rate from StateTransferRates

 where state = @state

 --calculate the new base pay and return that

 set @newbasePay =@basePay + (@basePay \* (@rate / 100))

 return @newbasePay

end

GO

--- END SCRIPT ------

Once you create the function, you can find it under SQL Management Studio.



This function is now available to be executed within a query. For example, we can query all the employee in our EmployeeTransfers table and see what their new base pay will be for the state that they are transferring to.

--Now we can utilize our function to return a new base pay value

select

EmpIdSSN,

EmpTransferState,

EmpBasePay,

dbo.CalculateTransferPay(EmpBasePay,EmpTransferState) as newbasePay

from EmployeeTransfers



Notice the following about our query.

1. We had to explicitly call the function with the schema prefix (dbo).
2. We passed the parameters (EmpBasePay, EmpTransferState) which were the values for each row from our query, respectively.
3. The output of the function, defined as a decimal was given a label of “newbasePay” in our result set.

Some Limitations of functions are:

1. You can’t call a Stored Procedure from a function.
2. You can’t set transactions in a function.
3. Cannot use a Try-Catch block.

For further reading about creating user defined functions, please visit the MSDN page at:

<https://msdn.microsoft.com/en-us/library/ms191320.aspx>

**Assignment (5 points Extra Credit)**

1. Given the tables above, create your own function to return the Employee’s STATE filing status based on their existing FEDERAL filing status. Different states have different codes for filing status. The logic is as follows:

If State = MS and

 Federal Filing: Single (S) then Code is A

 Or Federal Filing: Married(M) then Code is M

If State = NJ and

 Federal Filing: Single (S) then Code is B

 Or Federal Filing: Married(M) then Code is A

If State = AZ and

Federal Filing: Single(S) then Code is A

 Or Federal Filing: Married(M) then Code is B

If State = CT and

 Federal Filing: Single(S) then Code is F

Or Federal Filing: Married(M) then Code is M

If State = DC and

 Federal Filing: Single(S) then Code is S

 Or Federal Filing: Married(M) then Code is Y

All Other states

Federal Filing: Single (S) then Code is S

 Or Federal Filing: Married (M) then Code is M

This function will be called GetStateFilingStatus() and it will take 2 parameters, the federal filing status( char(1) ) and the state ( char(2) ) respectively.

You should create this function so that it returns one of the Codes, which is of data type char(1).

So for example, if perform a query such that

select dbo.GetStateFilingStatus('M','NJ') as StateFilingStatus

then based on the mapping logic above, this would return back the state filing status of ‘A’



Create both the function and then query the EmployeeTransfers table to use the function with the EmpFederalFilingStatus and EmpTransferState columns.