*Function*

Function is a database object in Sql Server. Basically it is a set of sql statements that accepts only input parameters, perform actions and return the result. Function can return only single value or a table. We can’t use function to Insert, Update, Delete records in the database table(s). For more about stored procedure and function refer the article [Difference between Stored Procedure and Function](http://www.dotnet-tricks.com/Tutorial/sqlserver/7EDL150912-Difference-between-Stored-Procedure-and-Function-in-SQL-Server.html)

Types of Function

1. System Defined Function

These functions are defined by Sql Server for different purpose. We have two types of system defined function in Sql Server

* 1. Scalar Function

Scalar functions operates on a single value and returns a single value. Below is the list of some useful Sql Server Scalar functions.

System Scalar Function

Scalar Function

Description

abs(-10.67)

This returns absolute number of the given number means 10.67.

rand(10)

This will generate random number of 10 characters.

round(17.56719,3)

This will round off the given number to 3 places of decimal means 17.567

upper('dotnet')

This will returns upper case of given string means 'DOTNET'

lower('DOTNET')

This will returns lower case of given string means 'dotnet'

ltrim(' dotnet')

This will remove the spaces from left hand side of 'dotnet' string.

convert(int, 15.56)

This will convert the given float value to integer means 15.

* 1. Aggregate Function

Aggregate functions operates on a collection of values and returns a single value. Below is the list of some useful Sql Server Aggregate functions.

System Aggregate Function

Aggregate Function

Description

max()

This returns maximum value from a collection of values.

min()

This returns minimum value from a collection of values.

avg()

This returns average of all values in a collection.

count()

This returns no of counts from a collection of values.

1. User Defined Function

These functions are created by user in system database or in user defined database. We three types of user defined functions.

* 1. Scalar Function

User defined scalar function also returns single value as a result of actions perform by function. We return any datatype value from function.

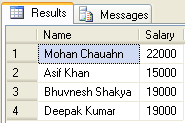
* + 1. ***--Create a table***
    2. **CREATE TABLE Employee**
    3. **(**
    4. **EmpID int PRIMARY KEY,**
    5. **FirstName varchar(50) NULL,**
    6. **LastName varchar(50) NULL,**
    7. **Salary int NULL,**
    8. **Address varchar(100) NULL,**
    9. **)**
    10. ***--Insert Data***
    11. **Insert into Employee(EmpID,FirstName,LastName,Salary,Address) Values(1,'Mohan','Chauahn',22000,'Delhi');**
    12. **Insert into Employee(EmpID,FirstName,LastName,Salary,Address) Values(2,'Asif','Khan',15000,'Delhi');**
    13. **Insert into Employee(EmpID,FirstName,LastName,Salary,Address) Values(3,'Bhuvnesh','Shakya',19000,'Noida');**
    14. **Insert into Employee(EmpID,FirstName,LastName,Salary,Address) Values(4,'Deepak','Kumar',19000,'Noida');**
    15. ***--See created table***
    16. **Select \* from Employee**



* + 1. ***--Create function to get emp full name***
    2. **Create function fnGetEmpFullName**
    3. **(**
    4. **@FirstName varchar(50),**
    5. **@LastName varchar(50)**
    6. **)**
    7. **returns varchar(101)**
    8. **As**
    9. **Begin return (Select @FirstName + ' '+ @LastName);**
    10. **end**

http://www.dotnet-tricks.com/Content/images/sqlserver/success.png

* + 1. ***--Calling the above created function***
    2. **Select dbo.fnGetEmpFullName(FirstName,LastName) as Name, Salary from Employee**



* 1. Inline Table-Valued Function

User defined inline table-valued function returns a table variable as a result of actions perform by function. The value of table variable should be derived from a single SELECT statement.

* + 1. ***--Create function to get employees***
    2. **Create function fnGetEmployee()**
    3. **returns Table**
    4. **As**
    5. **return (Select \* from Employee)**

http://www.dotnet-tricks.com/Content/images/sqlserver/success.png

* + 1. ***--Now call the above created function***
    2. **Select \* from fnGetEmployee()**



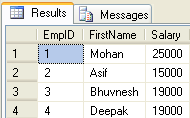
* 1. Multi-Statement Table-Valued Function

User defined multi-statement table-valued function returns a table variable as a result of actions perform by function. In this a table variable must be explicitly declared and defined whose value can be derived from a multiple sql statements.

* + 1. ***--Create function for EmpID,FirstName and Salary of Employee***
    2. **Create function fnGetMulEmployee()**
    3. **returns @Emp Table**
    4. **(**
    5. **EmpID int,**
    6. **FirstName varchar(50),**
    7. **Salary int**
    8. **)**
    9. **As**
    10. **begin**
    11. **Insert into @Emp Select e.EmpID,e.FirstName,e.Salary from Employee e;**
    12. ***--Now update salary of first employee***
    13. **update @Emp set Salary=25000 where EmpID=1;**
    14. ***--It will update only in @Emp table not in Original Employee table***
    15. **return**
    16. **end**

http://www.dotnet-tricks.com/Content/images/sqlserver/success.png

* + 1. ***--Now call the above created function***
    2. **Select \* from fnGetMulEmployee()**



* + 1. ***--Now see the original table. This is not affected by above function update command***
    2. **Select \* from Employee**



Note

1. Unlike Stored Procedure, Function returns only single value.
2. Unlike Stored Procedure, Function accepts only input parameters.
3. Unlike Stored Procedure, Function is not used to Insert, Update, Delete data in database table(s).
4. Like Stored Procedure, Function can be nested up to 32 level.
5. User Defined Function can have upto 1023 input parameters while a Stored Procedure can have upto 2100 input parameters.
6. User Defined Function can't returns XML Data Type.
7. User Defined Function doesn't support Exception handling.
8. User Defined Function can call only Extended Stored Procedure.
9. User Defined Function doesn't support set options like set ROWCOUNT etc.

### Limitations and Restrictions

* User-defined functions cannot be used to perform actions that modify the database state.
* User-defined functions cannot contain an OUTPUT INTO clause that has a table as its target.
* User-defined functions can not return multiple result sets. Use a stored procedure if you need to return multiple result sets.
* Error handling is restricted in a user-defined function. A UDF does not support TRY…CATCH, @ERROR or RAISERROR.
* User-defined functions cannot call a stored procedure, but can call an extended stored procedure.
* User-defined functions cannot make use of dynamic SQL or temp tables. Table variables are allowed.
* SET statements are not allowed in a user-defined function.
* The FOR XML clause is not allowed
* User-defined functions can be nested; that is, one user-defined function can call another. The nesting level is incremented when the called function starts execution, and decremented when the called function finishes execution. User-defined functions can be nested up to 32 levels. Exceeding the maximum levels of nesting causes the whole calling function chain to fail. Any reference to managed code from a Transact-SQL user-defined function counts as one level against the 32-level nesting limit. Methods invoked from within managed code do not count against this limit.
* The following Service Broker statements cannot be included in the definition of a Transact-SQL user-defined function:
  + BEGIN DIALOG CONVERSATION
  + END CONVERSATION
  + GET CONVERSATION GROUP
  + MOVE CONVERSATION
  + RECEIVE
  + SEND

### Security

#### Permissions

Requires CREATE FUNCTION permission in the database and ALTER permission on the schema in which the function is being created. If the function specifies a user-defined type, requires EXECUTE permission on the type.

Use User Defined Function in C# code

create function TCupom (@cupom int)

returns float

as

begin

declare @Tcu float;

select @Tcu = sum (total) from alteraca2 where pedido = @cupom

if (@tcu is null)

set @tcu = 0;

return @tcu;

end

SqlCommand Totalf = new SqlCommand("SELECT dbo.Tcupom(@code)", conex1);

And remove the CommandType, this isn't a Stored Procedure, its a User Defined Function.

In all:

public void TotalCupom(int cupom)

{

float SAIDA;

SqlDataAdapter da2 = new SqlDataAdapter();

if (conex1.State == ConnectionState.Closed)

{ conex1.Open();}

SqlCommand Totalf = new SqlCommand("SELECT dbo.Tcupom(@code)", conex1);

SqlParameter code1 = new SqlParameter("@code", SqlDbType.Int);

code1.Value = cupom;

SAIDA = Totalf.ExecuteScalar();

return SAIDA;

}

## What are the benefits of User-Defined Functions?

The benefits to SQL Server User-Defined functions are numerous. First, we can use these functions in so many different places when compared to the SQL Server stored procedure. The ability for a function to act like a table (for Inline table and Multi-statement table functions) gives developers the ability to break out complex logic into shorter and shorter code blocks. This will generally give the additional benefit of making the code less complex and easier to write and maintain. In the case of a Scalar User-Defined Function, the ability to use this function anywhere you can use a scalar of the same data type is also a very powerful thing. Combining these advantages with the ability to pass parameters into these database objects makes the SQL Server User-Defined function a very powerful tool.

- See more at: <http://www.sqlteam.com/article/user-defined-functions#sthash.24oMwTax.dpuf>

The benefits of using user-defined functions in SQL Server are:

* They allow modular programming.

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.

* 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.

CLR functions offer significant performance advantage over Transact-SQL functions for computational tasks, string manipulation, and business logic. Transact-SQL functions are better suited for data-access intensive logic.

* They can 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 invoked in the WHERE clause to reduce the number or rows sent to the client.

#### What are the benefits of User-Defined Functions?

a. Can be used in a number of places without restrictions as compared to stored procedures.

b. Code can be made less complex and easier to write.

c. Parameters can be passed to the function.

d. They can be used to create joins and also be sued in a select, where or case statement.

e. Simpler to invoke.

#### What is user-defined function? Explain its types i.e. scalar and Inline table value user-defined function.

User defined functions are created and defined by the user. They are created as per users needs. They may or may not accept parameters. They can be used to create joins and simple to invoke as compared to stored procedures

**Types:  
Scalar user defined**: returns values as one of the scalar data types. Text, timestamp, image data types are not supported. It may or may not accept parameters.

**Inline table value user defined** : it returns a table data type. These functions can pass parameters to the sql’s SELECT command. They are similar to views except, they can accept parameters.

Difference bet. Sp and function

Basic Difference

1. Function must return a value but in Stored Procedure it is optional( Procedure can return zero or n values).
2. Functions can have only input parameters for it whereas Procedures can have input/output parameters .
3. Function takes one input parameter it is mandatory but Stored Procedure may take o to n input parameters..
4. Functions can be called from Procedure whereas Procedures cannot be called from Function.

Advance Difference

1. Procedure allows SELECT as well as DML(INSERT/UPDATE/DELETE) statement in it whereas Function allows only SELECT statement in it.
2. Procedures can not be utilized in a SELECT statement whereas Function can be embedded in a SELECT statement.
3. Stored Procedures cannot be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section whereas Function can be.
4. Functions that return tables can be treated as another rowset. This can be used in JOINs with other tables.
5. Inline Function can be though of as views that take parameters and can be used in JOINs and other Rowset operations.
6. Exception can be handled by try-catch block in a Procedure whereas try-catch block cannot be used in a Function.
7. We can go for Transaction Management(COMMIT / ROLLBACK) in Procedure whereas we can't go in Function.

The difference between SP and UDF is listed below:

