# [User-defined functions](https://docs.microsoft.com/en-us/sql/relational-databases/user-defined-functions/user-defined-functions)

Like functions in programming languages, SQL Server user-defined functions are routines that accept parameters, perform an action, such as a complex calculation, and return the result of that action as a value. The return value can either be a single scalar value or a result set.

## User-defined functions

Why use them?

* **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.

Note

Transact-SQL user-defined functions in queries can only be executed on a single thread (serial execution plan).

## Types of functions

**Scalar Function**  
User-defined scalar functions return a single data value of the type defined in the RETURNS clause. For an inline scalar function, there is no function body; the scalar value is the result of a single statement. For a multistatement scalar function, the function body, defined in a BEGIN...END block, contains a series of Transact-SQL statements that return the single value. The return type can be any data type except **text**, **ntext**, **image**, **cursor**, and **timestamp**.

**Table-Valued Functions**  
User-defined table-valued functions return a **table** data type. For an inline table-valued function, there is no function body; the table is the result set of a single SELECT statement.

**System Functions**  
SQL Server provides many system functions that you can use to perform a variety of operations. They cannot be modified.

## Deterministic and Nondeterministic Functions

Deterministic functions always return the same result any time they are called with a specific set of input values and given the same state of the database. Nondeterministic functions may return different results each time they are called with a specific set of input values even if the database state that they access remains the same. For example, the function AVG always returns the same result given the qualifications stated above, but the GETDATE function, which returns the current datetime value, always returns a different result.

## Create User-defined Functions

### 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 cannot 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.
* 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

### Scalar Functions

CREATE FUNCTION <SF\_Function\_Name>(@ProductID int)

RETURNS int

AS BEGIN

DECLARE @ret int;

…………….

Return @ret

END

### Table-Valued Functions

#### Inline Table Valued function:

CREATE FUNCTION <ITVF\_Function\_Name>(@storeid int)

RETURNS TABLE

AS

RETURN

(

< SELECT Statement >

);

### SELECT \* FROM <Function\_Name> (602);

#### Multi statement Table-Valued Function:

CREATE FUNCTION <MSTVF\_Function\_Name> (@InEmpID INTEGER)

RETURNS @retFindReports TABLE

(

EmployeeID int primary key NOT NULL,

FirstName nvarchar(255) NOT NULL,

LastName nvarchar(255) NOT NULL,

JobTitle nvarchar(50) NOT NULL,

RecursionLevel int NOT NULL

)

--Returns a result set that lists all the employees who report to the

--specific employee directly or indirectly.\*/

AS BEGIN

INSERT @retFindReports

SELECT EmployeeID, FirstName, LastName, JobTitle, RecursionLevel

FROM EMP\_cte

RETURN

END

SELECT EmployeeID, FirstName, LastName, JobTitle, RecursionLevel

FROM dbo.ufn\_FindReports(1);

## Create CLR Functions

You can create a database object inside an instance of SQL Server that is programmed in an assembly created in the Microsoft .NET Framework common language runtime (CLR). Database objects that can leverage the rich programming model provided by the common language runtime include aggregate functions, functions, stored procedures, triggers, and types.

Creating a CLR function in SQL Server involves the following steps:

* Define the function as a static method of a class in a language supported by the .NET Framework. Then compile the class to build an assembly in the .NET Framework by using the appropriate language compiler.
* Register the assembly in SQL Server by using the CREATE ASSEMBLY statement.
* Create the function that references the registered assembly by using the [CREATE FUNCTION](https://docs.microsoft.com/en-us/sql/t-sql/statements/create-function-transact-sql) statement.

## Create User-defined Aggregates

You can create a database object inside SQL Server that is programmed in a CLR assembly. Database objects that can leverage the rich programming model provided by the CLR include triggers, stored procedures, functions, aggregate functions, and types.

Like the built-in aggregate functions provided in Transact-SQL, user-defined aggregate functions perform a calculation on a set of values and return a single value.

Creating a user-defined aggregate function in SQL Server involves the following steps:

* Define the user-defined aggregate function as a class in a Microsoft .NET Framework-supported language. Compile this class to build a CLR assembly using the appropriate language compiler.
* Register the assembly in SQL Server using the CREATE ASSEMBLY statement..
* Create the user-defined aggregate that references the registered assembly using the CREATE AGGREGATE statement.