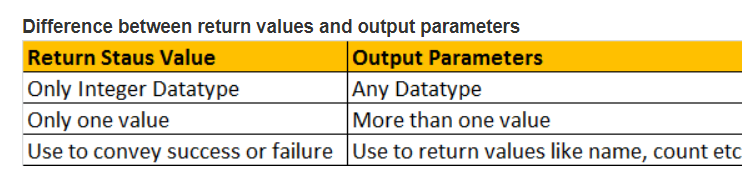
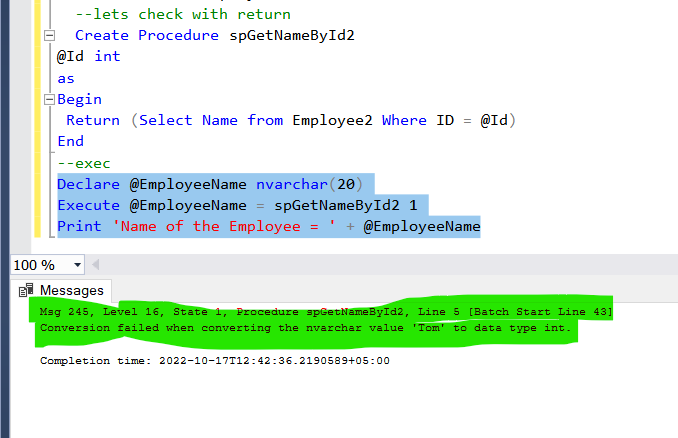
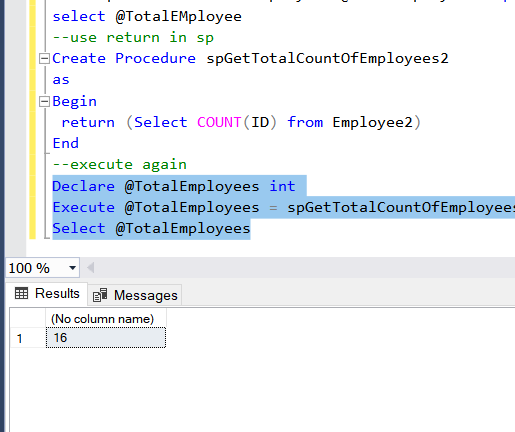
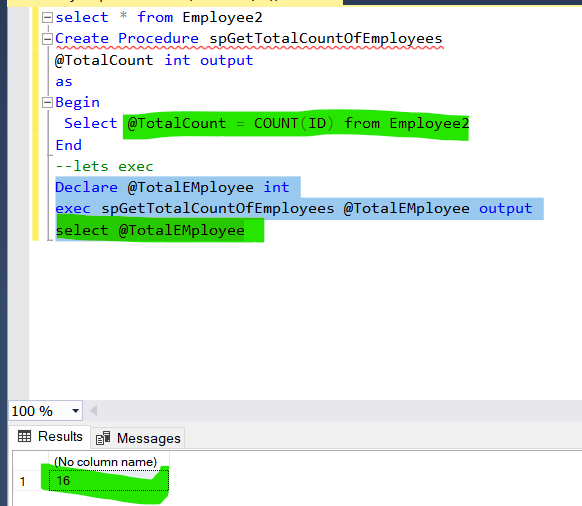
**SQL Server Advanced 7**



**Advantages of SQL SP Proc :**

**The following advantages of using Stored Procedures over adhoc queries (inline SQL)**  
**1. Execution plan retention and reusability** - Stored Procedures are compiled and their execution plan is cached and used again, when the same SP is executed again. Although adhoc queries also create and reuse plan, the plan is reused only when the query is textual match and the datatypes are matching with the previous call. Any change in the datatype or you have an extra space in the query then, a new plan is created.  
  
**2. Reduces network traffic** - You only need to send, EXECUTE SP\_Name statement, over the network, instead of the entire batch of adhoc SQL code.  
  
**3. Code reusability and better maintainability** - A stored procedure can be reused with multiple applications. If the logic has to change, we only have one place to change, where as if it is inline sql, and if you have to use it in multiple applications, we end up with multiple copies of this inline sql. If the logic has to change, we have to change at all the places, which makes it harder maintaining inline sql.  
  
**4. Better Security** - A database user can be granted access to an SP and prevent them from executing direct "select" statements against a table.  This is fine grain access control which will help control what data a user has access to.  
  
**5. Avoids SQL Injection attack** - SP's prevent sql injection attack. [Please watch this video on SQL Injection Attack, for more information.](http://csharp-video-tutorials.blogspot.com/2012/06/sql-injection-attack.html)

select \* from Employee2

Create Procedure spGetTotalCountOfEmployees

@TotalCount int output

as

Begin

Select @TotalCount = COUNT(ID) from Employee2

End

--lets exec

Declare @TotalEMployee int

exec spGetTotalCountOfEmployees @TotalEMployee output

select @TotalEMployee

--use return in sp

Create Procedure spGetTotalCountOfEmployees2

as

Begin

return (Select COUNT(ID) from Employee2)

End

--execute again

Declare @TotalEmployees int

Execute @TotalEmployees = spGetTotalCountOfEmployees2

Select @TotalEmployees

--lets without return

create proc spGetNameById1

@id int,

@name nvarchar(20) output

as

begin

select Name=@name from Employee2 where ID=@id

end

-- exec sp

declare @EmployeeName nvarchar(20)

exec spGetNameById1 3,@EmployeeName output

print 'EmployeeName = ' + @EmployeeName

--lets check table

select \* from Employee2

--lets check with return

Create Procedure spGetNameById2

@Id int

as

Begin

Return (Select Name from Employee2 Where ID = @Id)

End

--exec

Declare @EmployeeName nvarchar(20)

Execute @EmployeeName = spGetNameById2 1

Print 'Name of the Employee = ' + @EmployeeName

--So, using return values, we can only return integers, and that too,

--only one integer. It is not possible, to return

--more than one value using return values, where as output parameters, can return any

--datatype and an sp can have more than one output parameters.

--I always prefer, using output parameters, over RETURN values.

--In general, RETURN values are used to indicate success or failure of

--stored procedure, especially when we are dealing with nested stored procedures.

--Return a value of 0, indicates success, and any nonzero value indicates failure.

------------------------------Advantages of SQL SP Proc-------------- :-------------

--The following advantages of using Stored Procedures over adhoc queries (inline SQL)

--1. Execution plan retention and reusability --- Stored Procedures are compiled and their execution plan is

--cached and used again, when the same SP is executed again. Although adhoc queries also

--create and reuse plan, the plan is reused only when the query is textual match and the

--datatypes are matching with the previous call. Any change in the datatype or

--you have an extra space in the query then, a new plan is created.

--2. Reduces network traffic --- You only need to send, EXECUTE SP\_Name statement,

--over the network, instead of the entire batch of adhoc SQL code.

--3. Code reusability and better maintainability---- A stored procedure can be reused with

--multiple applications. If the logic has to change, we only have one place to change,

--where as if it is inline sql, and if you have to use it in multiple applications,

--we end up with multiple copies of this inline sql. If the logic has to change,

--we have to change at all the places, which makes it harder maintaining inline sql.

--4. Better Security ------ A database user can be granted access to an SP

--and prevent them from executing direct "select" statements against a table.

--This is fine grain access control which will

--help control what data a user has access to.

