CREATE PROCEDURE uspGetAddress @City nvarchar(30)

AS

SELECT \*

FROM AdventureWorks.Person.Address

WHERE City = @City

EXEC uspGetAddress @City = 'New York'

CREATE PROCEDURE uspGetAddress @City nvarchar(30)

AS

SELECT \*

FROM AdventureWorks.Person.Address

WHERE City LIKE @City + '%'

GO

In both of the proceeding examples it assumes that a parameter value will always be passed. If you try to execute the procedure without passing a parameter value you will get an error message such as the following:

|  |
| --- |
| Msg 201, Level 16, State 4, Procedure uspGetAddress, Line 0  Procedure or function 'uspGetAddress' expects parameter '@City', which was not supplied. |

**Default Parameter Values**

In most cases it is always a good practice to pass in all parameter values, but sometimes it is not possible.  So in this example we use the NULL option to allow you to not pass in a parameter value.  If we create and run this stored procedure as is it will not return any data, because it is looking for any City values that equal NULL.

|  |
| --- |
| CREATE PROCEDURE uspGetAddress @City nvarchar(30) = NULL  AS  SELECT \*  FROM AdventureWorks.Person.Address  WHERE City = @City  GO |

We could change this stored procedure and use the ISNULL function to get around this.  So if a value is passed it will use the value to narrow the result set and if a value is not passed it will return all records.

|  |
| --- |
| CREATE PROCEDURE uspGetAddress @City nvarchar(30) = NULL  AS  SELECT \*  FROM AdventureWorks.Person.Address  WHERE City = ISNULL(@City,City)  GO |

**Multiple Parameters**

Setting up multiple parameters is very easy to do.  You just need to list each parameter and the data type separated by a comma as shown below.

|  |
| --- |
| CREATE PROCEDURE uspGetAddress @City nvarchar(30) = NULL, @AddressLine1 nvarchar(60) = NULL  AS  SELECT \*  FROM AdventureWorks.Person.Address  WHERE City = ISNULL(@City,City)  AND AddressLine1 LIKE '%' + ISNULL(@AddressLine1 ,AddressLine1) + '%'  GO |

To execute this you could do any of the following:

|  |
| --- |
| EXEC uspGetAddress @City = 'Calgary'  --or  EXEC uspGetAddress @City = 'Calgary', @AddressLine1 = 'A'  --or  EXEC uspGetAddress @AddressLine1 = 'Acardia'  -- etc... |

sp\_help emp

create function emp\_func (@ead varchar(10))

returns @table table

(

eno int,

ename varchar(10),

eadd varchar(10),

esal int,

doj datetime,

status varchar(1))

as

begin

insert @table

select \* from emp

where eadd=@ead

return

end

select \* from emp\_func('vashi')

create function func1(@salary int)

returns int

as

begin

return (@salary \* 12)

end

declare @sal int

set @sal=dbo.func1(6000)

print @sal

create function deptfunc(@dept varchar(10))

returns table

as

return(

select \* from dbo.dept35 where dname=@dept

)

select \* from deptfunc('it')

create function empfunc1(@esal int)

returns @table TABLE

(

ENO VARCHAR(5),

ENAME VARCHAR(10),

EADD VARCHAR(10),

ESAL INT)

AS

BEGIN

INSERT @TABLE SELECT \* FROM EMP35 WHERE ESAL > @ESAL

RETURN

END

SELECT \* FROM EMPFUNC1(5000)

create procedure myproc1 @num varchar(10)

as

begin

if exists(select \* from emp35 where eno=@num)

begin

print 'the employee exists:'

return 0

end

else

return 1

end

create procedure myproc3 @num varchar(10)

as

begin

declare @value int

exec @value=myproc1 'e001';

if(@value=0)

print 'we are in proc3 ::employee exists:'

else

print 'employee does not exist'

end

exec myproc3 'e001'

Calling one procedure from another

create procedure sanketproc4 @salary int

as

begin

if exists(select \* from emp35 where esal >@salary)

begin

return 0

end

else

return 1

end

create procedure sanketproc2 @salary int

as

begin

declare @returnvalue int

exec @returnvalue=sanketproc3 @salary

if (@returnvalue=0)

print 'employee earning more than 6000 exist';

else

print 'no such employee exist'

end;

exec sanketproc2 6000