**EXPERIMENT 7**

**Hafiz Ahmad 19L-1316**

# Stored procedures

**Objective**

* Repeat Lab6 questions
* Learn how to create and Execute Stored Procedures with parameters
* Learn how to create and Execute Stored Procedures without parameters

## Lab6 Revision

Please run the following commands in your own database and give the screenshot of the output:

select e.fname, e.lname from works\_on w

inner join employee e on e.ssn=w.essn

inner join project p on w.Pno=p.Pnumber

where p.Pname='ProductX'

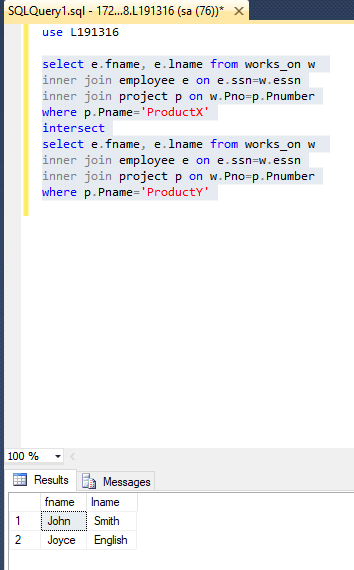
intersect

select e.fname, e.lname from works\_on w

inner join employee e on e.ssn=w.essn

inner join project p on w.Pno=p.Pnumber

where p.Pname='ProductY'

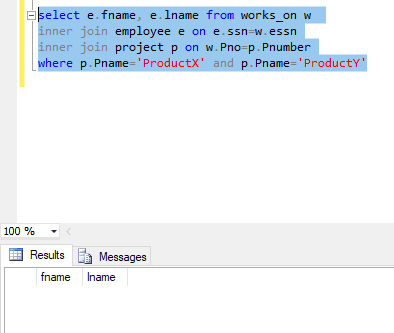


select e.fname, e.lname from works\_on w

inner join employee e on e.ssn=w.essn

inner join project p on w.Pno=p.Pnumber

where p.Pname='ProductX' and p.Pname='ProductY'



Give your comments regarding the outputs you observed in the space below:

Intersect prints working on both project and will not give the outputs which is working on both project as

## Stored Procedures

|  |  |
| --- | --- |
|  |  |

**Overview:**

A stored procedure is nothing more than prepared SQL code that you save so you can reuse the code over and over again.  So if you think about a query that you write over and over again, instead of having to write that query each time you would save it as a stored procedure and then just call the stored procedure to execute the SQL code that you saved as part of the stored procedure.

In addition to running the same SQL code over and over again you also have the ability to pass parameters to the stored procedure, so depending on what the need is the stored procedure can act accordingly based on the parameter values that were passed.

Stored procedures are useful in the following circumstances:

* If a database program is needed by several applications, it can be stored at the server and invoked by any of the application programs. This reduces duplication of effort and improves software modularity.
* Executing a program at the server can reduce data transfer and communication cost between the client and server in certain situations.
* These procedures can enhance the modeling power provided by views by allowing more complex types of derived data to be made available to the database users. Additionally, they can be used to check for complex constraints that are beyond the specification power of assertions and triggers.

**The general form of declaring stored procedures is as follows:**

CREATE PROCEDURE <procedure name> (<parameters>)

<Local declarations>

<Procedure body>;

The parameters and local declarations are optional, and are specified only if needed.

There are various options that can be used to create stored procedures.  In these next few topics we will discuss creating a simple stored procedure to more advanced options that can be used when creating stored procedures.

1. **Stored Procedure (Without Parameters)**

In the example given below we will just select employee names from employee table. The given procedure has no parameter and simply returns the result of query.

To create this procedure that returns employee’s names you would run the following statement:

Create procedure getEmployeeNames

as

select fname, lname from employee

To execute this procedure you would simply run:

EXEC getEmployeeNames

Or simply

getEmployeeNames

1. **Stored Procedure (With Parameters)**

The real power of stored procedures is the ability to pass parameters and have the stored procedure handles the differing requests that are made.

In the example given below we will query the employee table but instead of getting back all records we will limit it employee living in just a particular city.

Here we have one parameter **city** of type varchar defined in procedure. The syntax for defining parameter can be seen from below example.

Create procedure getEmployeeAddress @City varchar(20)

as

select fname, lname,address from employee

where address like '%'+@City+'%'

To call this procedure we would run the following statement:

exec getEmployeeAddress @City='Houston'

If you try to execute the procedure without passing a parameter value you will get an error message such as the following which states the procedure was expecting a parameter which was not provided.

Msg 201, Level 16, State 4, Procedure getEmployeeAddress, Line 0

Procedure or function 'getEmployeeAddress' expects parameter '@City', which was not supplied.

1. **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 the example given below 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.

Alter procedure getEmployeeAddress @City varchar(20)=NULL

as

select fname, lname,address from employee

where address like '%'+@City+'%'

We could change this stored procedure and use the IS NULL 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.

Alter procedure getEmployeeAddress @City varchar(20)=NULL

as

select fname, lname,address from employee

where address like '%'+ISNULL(@City,address)+'%'

1. **Multiple Parameters:**

We can define multiple parameters by listing each parameter and the data type separated by a comma. Following example shows how to make procedure with multiple parameters.

Create procedure getEmployeeDetails @City varchar(20)=NULL, @pSal int=NULL

as

select \* from employee

where address like '%'+ISNULL(@City,address)+'%'

and salary=isnull(@pSal, salary)

To execute the above procedure, you could do any of the following:

Exec getEmployeeDetails

getEmployeeDetails @City='Houston'

Exec getEmployeeDetails @City='Houston', @pSal=55000

1. **Output Parameter:**

So far we have seen how to pass parameters into a stored procedure, but another option is to pass parameter values back out from a stored procedure.  One option for this may be that you call another stored procedure that does not return any data, but returns parameter values to be used by the calling stored procedure.

Setting up output parameters for a stored procedure is basically the same as setting up input parameters; the only difference is that you use the OUTPUT clause after the parameter name to specify that it should return a value.  The output clause can be specified by either using the keyword "OUTPUT" or just "OUT".

Create procedure getEmployeeCount @City varchar(20)=NULL, @empCount int OUT

as

select @empCount=COUNT(\*)from employee

where address like '%'+ISNULL(@City,address)+'%'

We can also use output instead of out in the procedure.

To call this stored procedure we would execute it as follows.  First we are going to declare a variable, execute the stored procedure and then select the returned valued.

Declare @empCnt int

Exec getEmployeeCount'Houston', @empCnt output

select @empCnt

1. **Deleting a Stored Procedure:**

To drop a single stored procedure you use the DROP PROCEDURE or DROP PROC command as follows.

Drop proc getEmployeeCount

Drop procedure getEmployeeDetails

OR drop procedure lab7.getEmployeeDetails

if the procedure exists within a schema named as Lab7

1. **Modifying an Existing Stored Procedure:**

To change the stored procedure and save the updated code you would use the ALTER PROCEDURE command as follows.

Alter procedure getEmployeeAddress @City varchar(20)=NULL

as

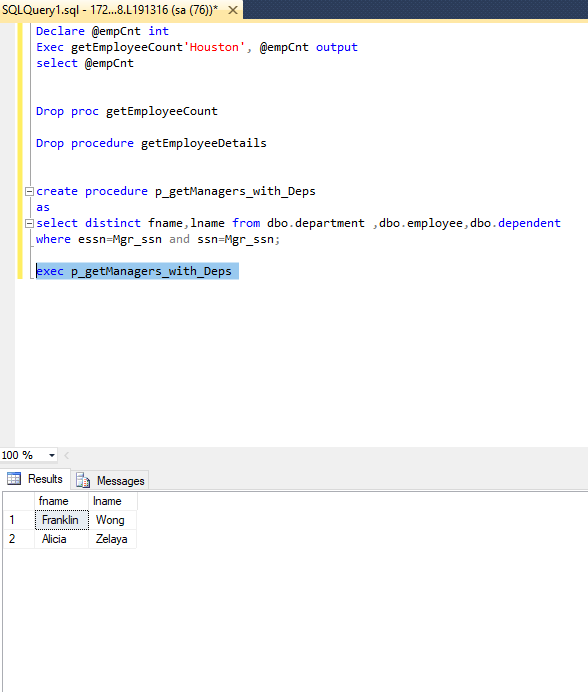
select fname, lname,addressfrom employee

where address like '%'+ISNULL(@City,address)+'%'

**Exercise Questions:**

Using your own database (created in Lab3), write an **SQL statement along with output in the space provided** to answer each of the following queries:

1. Create a store procedure called p\_getManagers\_with\_Deps that outputs the names of managers who have a dependent.



1. Modify the above procedure (question 3) to display the salaries of such employees as well as their names.

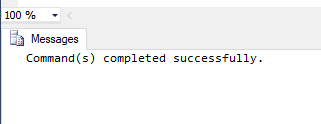
alter procedure p\_getManagers\_with\_Deps

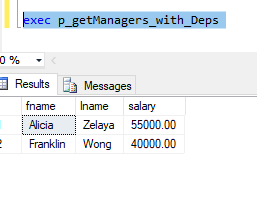
as

select distinct fname, lname, salary

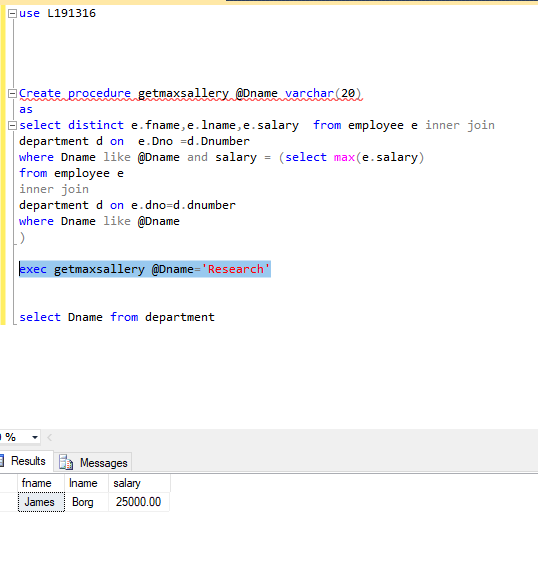
from dbo.department, dbo.employee, dbo.dependent

where essn=Mgr\_ssn and ssn=Mgr\_ssn;

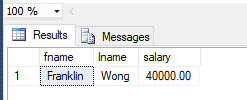




1. Create a stored procedure that returns the maximum salary earner of the department whose name is given as parameter.



exec getmaxsallery @Dname='Administration'



1. Create a stored procedure called “SP\_NewEmployeeDetail” which should insert the following details from Employee table in to a new table “T\_EmployeeDetail” .The table can be manually created with the selection of following attributes.

Employee (SSN, Name, salary, Address)

create table T\_EmployeeDetail(

ssn int not null,

name varchar(20),

salary int,

address varchar(250),

constraint pkssn primary key(ssn)

)

create procedure SP\_NewEmployeeDetail

as insert into T\_EmployeeDetail

select ssn,fname,salary,address from employee

exec SP\_NewEmployeeDetail

select \*from T\_EmployeeDetail

