National University of Computer and Emerging Sciences

Lab Manual 7

"Stored Procedures"

Database Systems Lab

Fall 2024

Department of Computer Science FAST-NU, Lahore, Pakistan

1. Objective

The purpose of this lab manual is to introduce stored procedures and how to create them and use them.

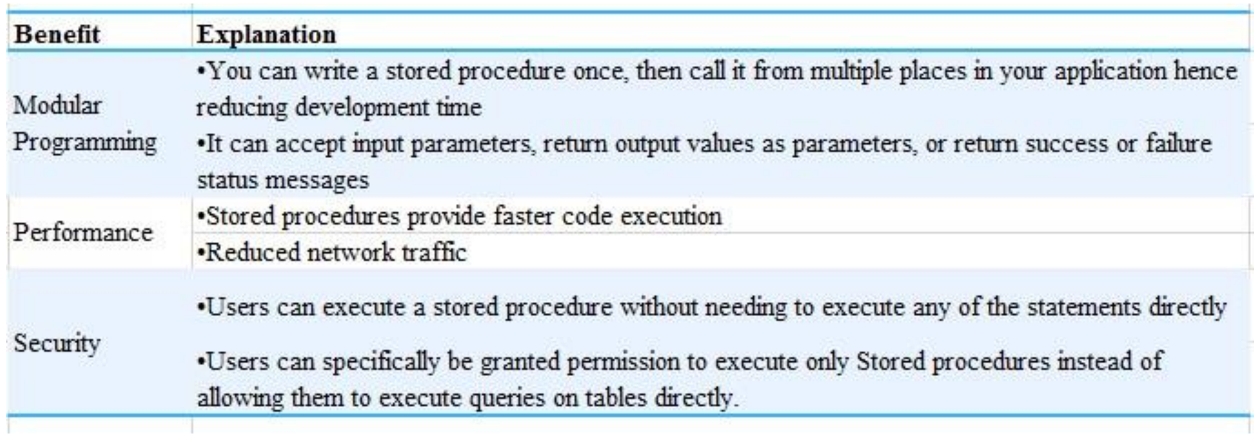
2. Prerequisites

- SQL Server 2014 Database Development.
- Chapter 5 Elmasri

3. Stored Procedures

Stored Procedure in SQL server can be defined as the set of logical group of SQL statements which are grouped to perform a specific task. A stored procedure is a prepared SQL code that you save so that you can reuse the code over and over again.

Benefits of Stored Procedures



Benefit	Explanation
Modular Programming	•You can write a stored procedure once, then call it from multiple places in your application hence reducing development time •It can accept input parameters, return output values as parameters, or return success or failure status messages
Performance	•Stored procedures provide faster code execution •Reduced network traffic
Security	•Users can execute a stored procedure without needing to execute any of the statements directly •Users can specifically be granted permission to execute only Stored procedures instead of allowing them to execute queries on tables directly.

Every time you execute simple SQL statements, syntax checking and compilation are done before execution and data return. However, syntax check and compilation is done while creating a procedure, and not on every execution which makes it faster than simple SQL statements.

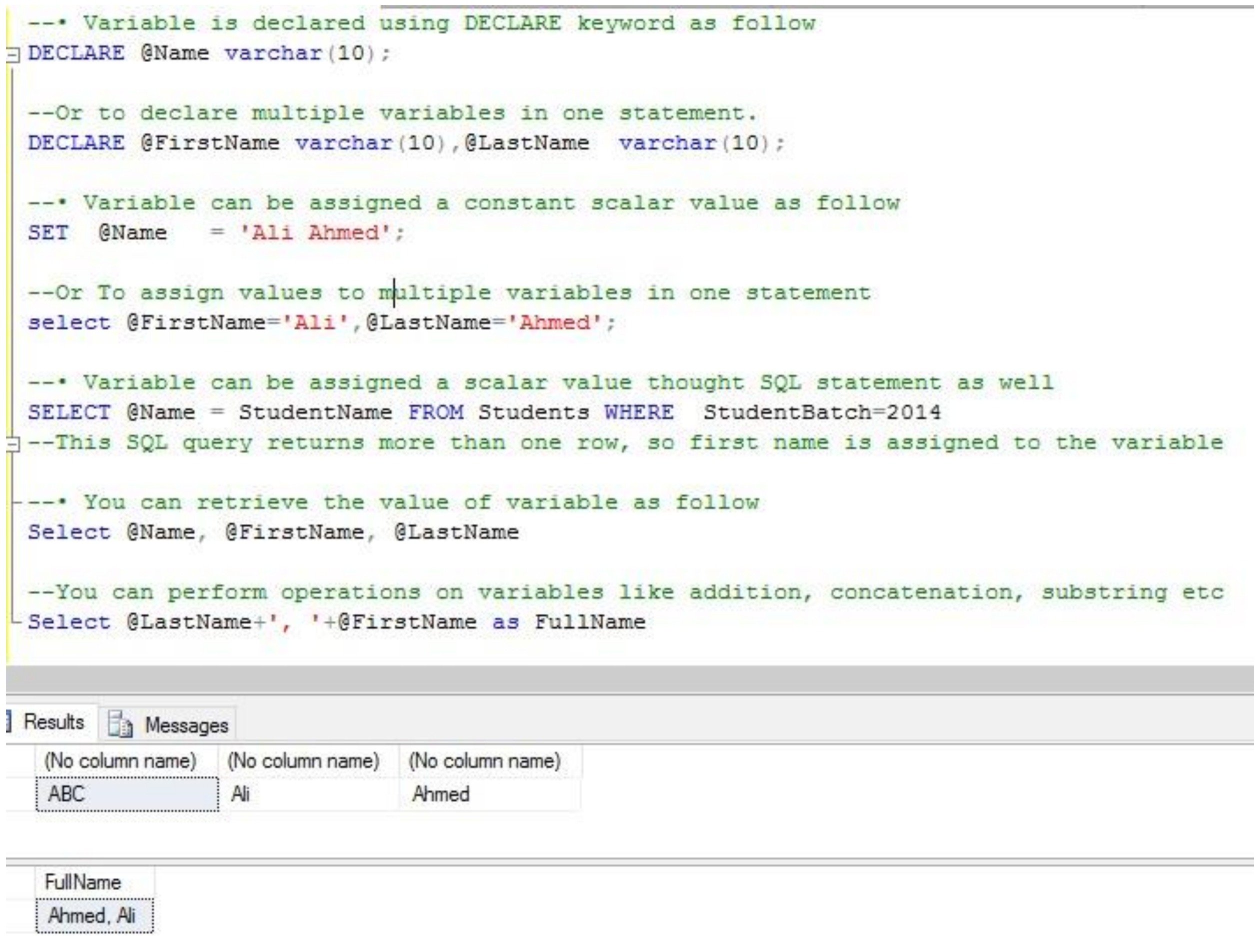
Variables.

Before we start with stored procedures, we should get to know the variables. Like in any other programing language SQL also provides scalar variables, which are very useful when creating stored procedures.

- Variable in SQL start with @ symbol
- Variable is declared using DECLARE keyword as follow o
  DECLARE @variableName datatype;
  Or to declare multiple variables in one statement. *o*
  DECLARE @variable1Name Datatype,@variable2Name datatype;
- Variable can be assigned a constant scalar value as follow *o SET @ variableName = value;*
  Or To assign values to multiple variables in one statement *o*
  select @ variable1Name = value, @variable2Name =value;
- Variable can be assigned a scalar value thought SQL statement as well *o*
  SELECT @vairableName = columnName FROM Table WHERE <condition> If SQL query returns more than one row, 1st value will be assigned to variable
- You can retrieve the value of variable as follow *o* *Select @variableName*

- You can perform operations on variables like addition, concatenation, substring etc.

TRY IT



```
--• Variable is declared using DECLARE keyword as follow
DECLARE @Name varchar(10);

--Or to declare multiple variables in one statement.
DECLARE @FirstName varchar(10),@LastName  varchar(10);

--• Variable can be assigned a constant scalar value as follow
SET  @Name   = 'Ali Ahmed';

--Or To assign values to multiple variables in one statement
select @FirstName='Ali',@LastName='Ahmed';

--• Variable can be assigned a scalar value thought SQL statement as well
SELECT @Name = StudentName FROM Students WHERE  StudentBatch=2014
--This SQL query returns more than one row, so first name is assigned to the variable

--• You can retrieve the value of variable as follow
Select @Name, @FirstName, @LastName

--You can perform operations on variables like addition, concatenation, substring etc
Select @LastName+', '+@FirstName as FullName
```

Results | Messages

(No column name)	(No column name)	(No column name)
ABC	Ali	Ahmed

FullName
Ahmed, Ali

NOTE: USE AND DECLARE VARIABLE IN SAME BATCH OF STATEMENTS, IF DECLARE STATEMENT IS NOT IN SAME BATCH, YOU WILL GET ERROR WHILE USING A VARIABLE.

CREATE Stored Procedure

Following is the syntax to create stored procedure: Input and output parameter a uses as required.

```
CREATE PROCEDURE [procedureName]
@input_param1 datatype,
@input_param2 datatype,
@output_param1 datatype OUTPUT,
@output _param2 datatype OUTPUT
AS
BEGIN

                    (SQL Queries)

END
```

go

How to execute Stored Procedure

declare @my_output_param1 int,
@my_output_param2 varchar(10) --these are the variables in which output variables of procedure will return values

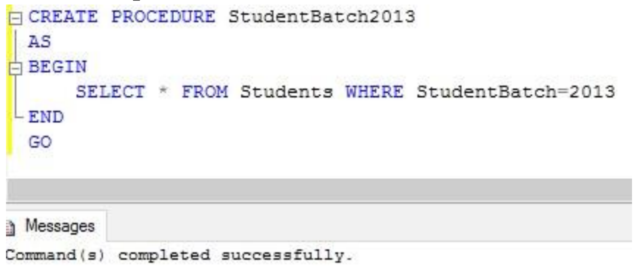
Exec dbo.procedure_name
@input_param1=value,
@input_param2 =value,
@output_param1=@my_output_param1 OUTPUT , @output_param2 =@my_output_param2 OUTPUT

select @my_output_param1 ,@my_output_param2 – you will then have to use select statements to retrieve data from parameters

Stored Procedures without I/O parameters

TRY IT:

Create this procedure to obtain all the students of batch 2013

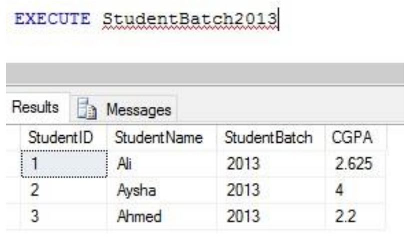


```
CREATE PROCEDURE StudentBatch2013
AS
BEGIN
    SELECT * FROM Students WHERE StudentBatch=2013
END
GO
```

Messages

```
Command(s) completed successfully.
```

Now execute this procedure



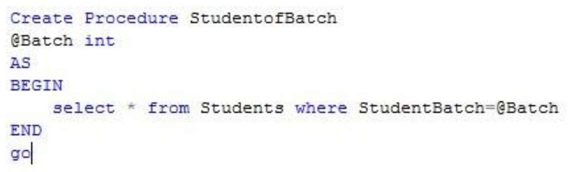
```
EXECUTE StudentBatch2013
```

Results | Messages

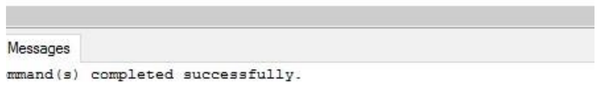
StudentID	StudentName	StudentBatch	CGPA
1	Ali	2013	2.625
2	Aysha	2013	4
3	Ahmed	2013	2.2

Stored procedure with input parameters

TRY IT
Create a SP which takes batchNo as input and returns all students of that batch.

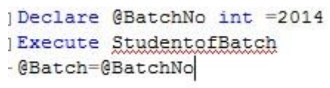


```
Create Procedure StudentofBatch
@Batch int
AS
BEGIN
    select * from Students where StudentBatch=@Batch
END
go
```

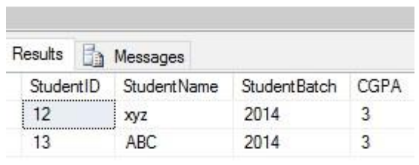


```
Messages
mmand(s) completed successfully.
```

Now execute it



```
Declare @BatchNo int =2014
Execute StudentofBatch
@Batch=@BatchNo
```



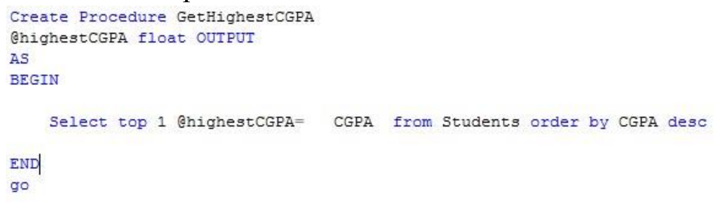
Results | Messages

StudentID	StudentName	StudentBatch	CGPA
12	xyz	2014	3
13	ABC	2014	3

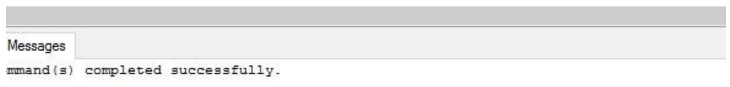
Store Procedures with output parameters

TRY IT:

Create a stored procedure that will return max CGPA in an output parameter

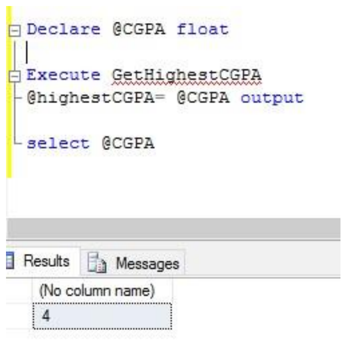


```
Create Procedure GetHighestCGPA
@highestCGPA float OUTPUT
AS
BEGIN

    Select top 1 @highestCGPA=   CGPA  from Students order by CGPA desc

END
go
```



```
Messages
mmand(s) completed successfully.
```

Execute it



```
Declare @CGPA float

Execute GetHighestCGPA
@highestCGPA= @CGPA output

select @CGPA
```

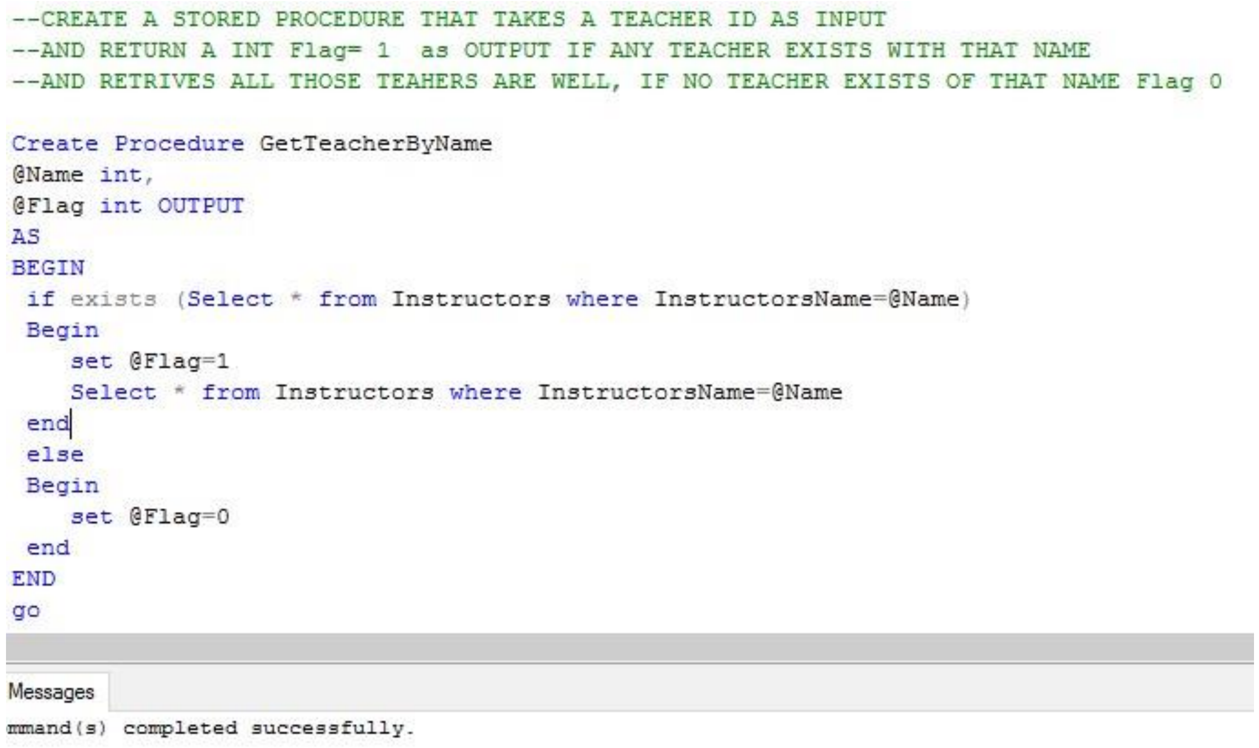
Results | Messages

```
(No column name)
4
```

QUESTION: WRITE A SP TO GET AVERAGE CGPA.

IF-ELSE conditions

Like in any programing language IF—ELSE in SQL provide ability to conditionally execute a code.
TRY THIS

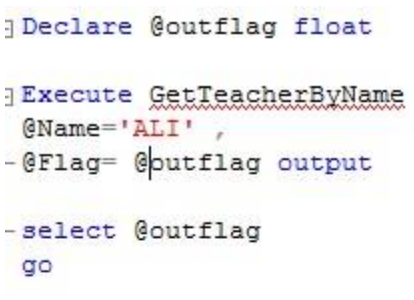


```
--CREATE A STORED PROCEDURE THAT TAKES A TEACHER ID AS INPUT
--AND RETURN A INT Flag= 1  as OUTPUT IF ANY TEACHER EXISTS WITH THAT NAME
--AND RETRIVES ALL THOSE TEAHERS ARE WELL, IF NO TEACHER EXISTS OF THAT NAME Flag 0

Create Procedure GetTeacherByName
@Name int,
@Flag int OUTPUT
AS
BEGIN
 if exists (Select * from Instructors where InstructorsName=@Name)
 Begin
     set @Flag=1
     Select * from Instructors where InstructorsName=@Name
 end
 else
 Begin
     set @Flag=0
 end
END
go
```

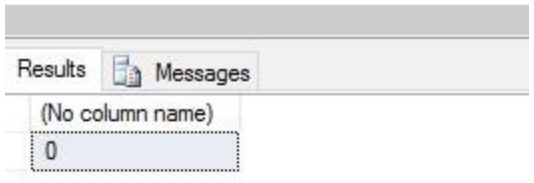
Messages

```
mmand(s) completed successfully.
```

Execute it

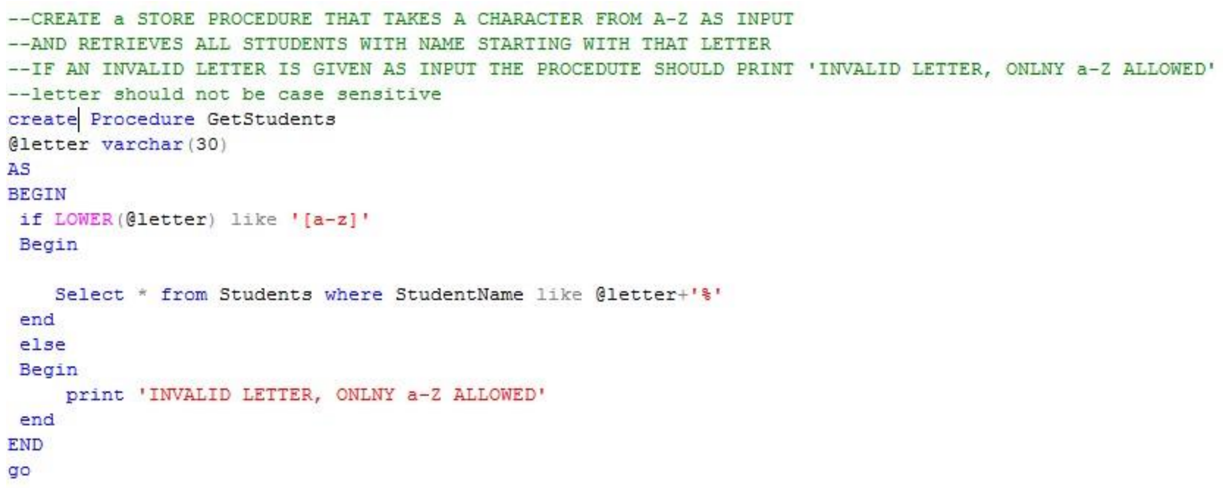


```
Declare @outflag float

Execute GetTeacherByName
@Name='ALI' ,
@Flag= @outflag output

select @outflag
go
```



```
(No column name)
0
```

TRY ANOTHER



```
--CREATE a STORE PROCEDURE THAT TAKES A CHARACTER FROM A-Z AS INPUT
--AND RETRIEVES ALL STTUDENTS WITH NAME STARTING WITH THAT LETTER
--IF AN INVALID LETTER IS GIVEN AS INPUT THE PROCEDUTE SHOULD PRINT 'INVALID LETTER, ONLNY a-Z ALLOWED'
--letter should not be case sensitive
create Procedure GetStudents
@letter varchar(30)
AS
BEGIN
 if LOWER(@letter) like '[a-z]'
 Begin

    Select * from Students where StudentName like @letter+'%'
 end
 else
 Begin
     print 'INVALID LETTER, ONLNY a-Z ALLOWED'
 end
END
go
```

TRY EXECUTING THESE

execute GetStudents @letter= 'B' execute

GetStudents @letter= '1'

Self-exploration

- What are default values? How can you set default values of parameters of Stored Procedures?
- How can you alter your procedure?