Introduction to Stored Procedures

What is a Stored Procedure?

* Stored Procedure In SQL server can be defined as the set of logically group of SQL statement which are grouped to perform a specific task. There are many benefits of using a stored procedure

* A stored procedure is nothing more than a prepared SQL code that you save so that you can reuse the code over and over again.

Benefits of using 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.

Stored Procedure vs. SQL Statement

SQL Statement

First Time
- Check syntax
- Compile
- Execute
- Return data

Second Time
- Check syntax
- Compile
- Execute
- Return data

Stored Procedure

Creating
- Check syntax
- Compile

First Time
- *Execute*
- *Return data*

Second Time
- *Execute*
- *Return data*

Syntax for creating/executing Stored Procedure

Creating Stored Procedure:

```
CREATE PROCEDURE dbo.procedure_name
AS
BEGIN
        (SQL Query)
END
```

Executing Stored Procedure:

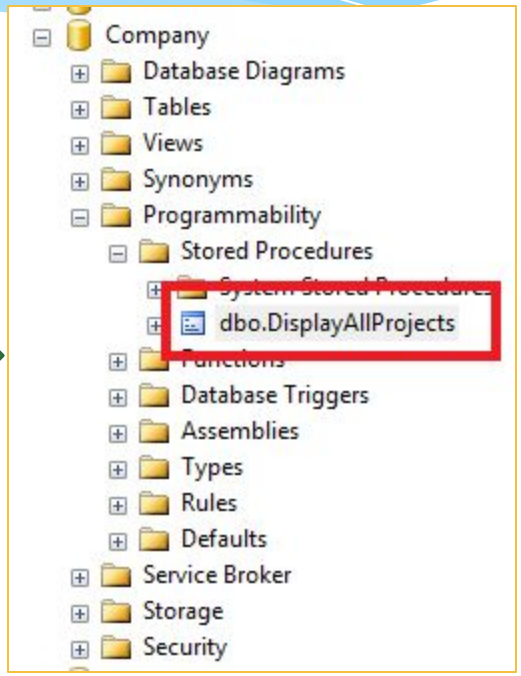
```
EXECUTE dbo.procedure_name
```

CREATE a Stored Procedure

SQLQuery5.sql - P...RESS.RTG (sa (53)) | SQLQuery4.sql - P...ompany (sa (55))*

```
1
2  CREATE PROCEDURE  dbo.DisplayAllProjects
3  AS
4  BEGIN
5  SELECT * from Project
6  END
```

Successfully



Variables in Stored Procedures

* **Declare a variable:**

  DECLARE @limit money money;

  DECLARE @firstName varchar(10),@hi_range varchar(10);

* **Assign a value into a variable:**

  SET @limit money = 0;

  select @firstName = 'ABC', @hi_range ='XYZ';

->Cannot assign value to 2 variables using SET

* **Assign a value into a variable in SQL statement:**

  SELECT @price = price FROM titles WHERE title_id = 'PC2091'

Types of Stored Procedures

Type	Explanation
Stored Procedure (without parameters)	The created procedure has no parameter and simply returns the result of query.
Stored Procedure (with parameters)	The real power of stored procedures is its ability to accept parameters and return results based upon those parameters.

Stored Procedures (with out Parameters)

Creation:

```
CREATE PROCEDURE dbo.DisplayAllProject
AS
BEGIN
        SELECT * FROM Projects
END
```

Execution:

```
EXECUTE DisplayAllProject
```

```
1
2   execute DisplayAllProjects
```

Results | Messages

	ProjectId	PName	StartDate	EndDate	Status	DeptNo
1	1	TK	20090101	20091010	Complete	10
2	2	DS	20090201	NULL	Progress	10
3	3	NF	20100101	NULL	Pending	40
4	4	AS	20080102	20090502	Complete	30
5	5	PD	20081101	NULL	Progress	20
6	6	BT	20100101	NULL	Pending	20
7	7	OP	20090505	20100201	Complete	30

Syntax for creating/executing Stored Procedure with parameters

Creating Stored Procedure:

```
CREATE PROCEDURE dbo.procedure_name
@input_param1 varchar(10),
@input_param2 int,
@output_param1  int OUTPUT,
@output _param2 datetime OUTPUT

AS
BEGIN
            (SQL Query)
END
```

Executing Stored Procedure:

```
declare @my_output_param1 int,@my_output_param2 varchar(10)
Exec dbo.procedure_name
@input_param1=Name',
@input_param2 =10,
@output_param1=@my_output_param1  OUTPUT  ,
@output_param2 =@my_output_param2 OUTPUT

select @my_output_param1 ,@my_output_param2
```

Stored Procedures (with Input Parameters)

Creation:

```
CREATE PROCEDURE dbo.DisplaySelectedProject
@ProjectName varchar(50),
@Status varchar (10)
AS
BEGIN
        SELECT * FROM Project
        where PName = @ProjectName
        and @Status = @Status
END
```

Execution:

```
EXECUTE
dbo.DisplayAllProject @ProjectName='TK',
@Status='Complete'
```

Results | Messages

	ProjectId	PName	StartDate	EndDate	Status	DeptNo
1	1	TK	20090101	20091010	Complete	10

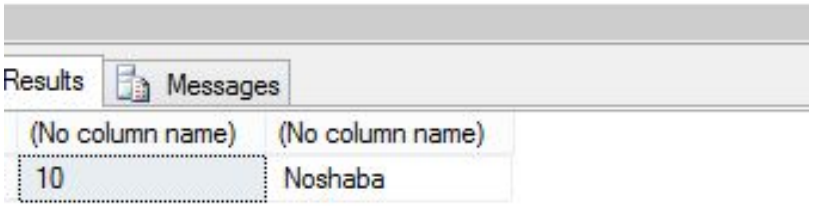
Stored Procedures (with OUTPUT Parameters)

Creation:

```
create procedure test_proc
@input_param1 varchar(10),
@input_param2 int,
@output_param1  int OUTPUT,
@output_param2 varchar(10) OUTPUT
as

begin

set @output_param1=@input_param2
set @output_param2=@input_param1


end
```

Execution:

```
declare @my_output_param1 int,
@my_output_param2 varchar(10)

Exec test_proc
@input_param1='Noshaba',
@input_param2 =10,
@output_param1=@my_output_param1  OUTPUT,
@output_param2 =@my_output_param2 OUTPUT

select @my_output_param1 ,
@my_output_param2
```



Stored Procedures (Default value of parameter)

Creation:

```
CREATE PROCEDURE ProcedureName
    -- Add the parameters for the stored procedure here
    @Param1 INT = 0
    @Param2 INT = 0 OUTPUT
AS
BEGIN
    -- Insert statements for procedure here
    SELECT @Param2 = ColumnName
    From Table_Name
    WHERE ColumnName = @Param1
END
GO
```

Execution:

```
Declare @myoutput int

Exec ProcedureName
@param2= @myoutput

select @myoutput
```

Stored Procedures (IF else Conditions)

```
CREATE PROCEDURE stp_GetBoats
    @BoatName VARCHAR(50)
AS
BEGIN

    IF @BoatName = ''
    BEGIN
        SELECT * FROM BOAT
    END
    ELSE
    BEGIN
        SELECT *
        FROM BOAT
        WHERE BoatName = @BoatName
    END

END
GO
```