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.

Creation of SQL Stored Procedure

Standardize on a Prefix

- usp_
- sp
- usp
- etc..

Before you create a stored procedure you need to know what your end result is, whether you are selecting data, inserting data, etc..

For Select command

```
CREATE PROCEDURE dbo.uspGetAddress

AS

SELECT * FROM Person.Address

GO
```

Calling

- EXEC dbo.uspGetAddress
- EXEC uspGetAddress
- uspGetAddress

When creating a stored procedure you can either use CREATE PROCEDURE or CREATE PROC. After the stored procedure name you need to use the keyword "AS" and then the rest is just the regular SQL code that you would normally execute.

One Parameter

```
CREATE PROCEDURE dbo.uspGetAddress @City nvarchar(30)

AS

SELECT *

FROM Person.Address

WHERE City = @City

GO
```

Calling

- EXEC dbo.uspGetAddress @City = 'New York'EXEC dbo.uspGetAddress

Default Parameter

CREATE PROCEDURE dbo.uspGetAddress @City nvarchar(30) = NULL

AS

SELECT *

FROM Person.Address

WHERE City = @City

GO

CREATE PROCEDURE dbo.uspGetAddress @City nvarchar(30) = NULL, @AddressLine1 nvarchar(60) = NULL

AS

SELECT *

FROM Person.Address

WHERE City = ISNULL(@City,City)

AND AddressLine1 LIKE '%' + ISNULL(@AddressLine1 ,AddressLine1) + '%'

GO

Calling

- EXEC dbo.uspGetAddress @City = 'Calgary'
- EXEC dbo.uspGetAddress @City = 'Calgary', @AddressLine1 = 'A'
- EXEC dbo.uspGetAddress @AddressLine1 = 'Acardia

Output parameter

CREATE PROCEDURE dbo.uspGetAddressCount @City nvarchar(30), @AddressCount int OUTPUT

AS

SELECT @AddressCount = count(*)

FROM AdventureWorks.Person.Address

WHERE City = @City

CREATE PROCEDURE dbo.uspGetAddressCount @City nvarchar(30), @AddressCount int OUT

AS

SELECT @AddressCount = count(*)

FROM AdventureWorks.Person.Address

WHERE City = @City

Calling

- DECLARE @AddressCount int
  EXEC dbo.uspGetAddressCount @City = 'Calgary', @AddressCount = @AddressCount OUTPUT
  SELECT @AddressCount

- DECLARE @AddressCount int
  EXEC dbo.uspGetAddressCount 'Calgary', @AddressCount OUTPUT
  SELECT @AddressCount

Try...Catch paradigm it is basically two blocks of code with your stored procedures that lets you execute some code, this is the Try section and if there are errors they are handled in the Catch section

```
CREATE PROCEDURE dbo.uspTryCatchTest

AS

BEGIN TRY

   SELECT 1/0

END TRY

BEGIN CATCH

   SELECT ERROR_NUMBER() AS ErrorNumber

    ,ERROR_SEVERITY() AS ErrorSeverity

    ,ERROR_STATE() AS ErrorState

    ,ERROR_PROCEDURE() AS ErrorProcedure

    ,ERROR_LINE() AS ErrorLine

    ,ERROR_MESSAGE() AS ErrorMessage;

END CATCH
```

ADD Error In another table

```
ALTER procedure [dbo].[usp_trnActivation]

As

--BEGIN TRANSACTION

BEGIN TRY

DECLARE @year INT,@mm INT,@day INT

SET @year = year(GETDATE())

SET @mm = MONTH(GETDATE())

SET @day = DAY(GETDATE()-1)

select @year,@mm,@day

delete from [trnActivation] where dd = @year and mm = @mm and yy = @day

 SELECT  *    from [172.19.4.125].[imei_final].dbo.[activation] where dd = @year and mm = @mm and yy = @day order by
anum

--COMMIT TRANSACTION

PRINT 'DONE'

END TRY

Begin Catch

            --Rollback transaction

             DECLARE

      @ErrorMessage VARCHAR(4000),   @ErrorSeverity INT,  @ErrorState INT;

  SELECT   @ErrorMessage = ERROR_MESSAGE(),  @ErrorSeverity = ERROR_SEVERITY(),

      @ErrorState = ERROR_STATE();

            Declare @ExecptionCode varchar(20)

            Select @ExecptionCode = count(ExecptionCode) from Execption

INSERT INTO Execption ([ExecptionCode],[RaisedBy] ,[Name]
,[Type],[ErrorMessage],[ErrorSeverity],[ErrorState],[CreatedDate])

   VALUES

 (@ExecptionCode,'StoreProcedure','usp_trnActivation','DB',  @ErrorMessage,@ErrorSeverity, @ErrorState,getdate())

End catch
```

Modifying Stored Procedure

```
ALTER PROCEDURE dbo.uspGetAddress @City nvarchar(30)

AS

SELECT *
```

FROM Person.Address

WHERE City LIKE @City + '%'

GO

DROP Stored Procedure

- DROP PROCEDURE dbo.uspGetAddress
- DROP PROC dbo.uspGetAddress
- DROP PROCEDURE dbo.uspGetAddress, dbo.uspInsertAddress, dbo.uspDeleteAddress