-- stored procedures can return max one value thats too integer

-- in case we want to return more than 1 values , we do that by using output

-- prameters

-- parameters : which we are passing to the sp

-- input > Parameters which we are passing , and we are pssing its value

--output > Parameters who will be assigned values insode the procedure

use employee

create proc GetDetails(@id int , @name varchar(20) output,

@dept varchar(20) output)

AS

Begin

Select @name=name , @dept=dept From Employee where id=@id

End

Calling procedure

declare @id int

Declare @name varchar(20)

declare @dept varchar(20)

set @id=1

exec GetDetails 1 , @name output , @dept output

print @name + ' ' + @dept

alter proc UpdateRec(@id int, @name varchar(20), @dept varchar(20))

AS

begin

if(@name is null or @name ='')

update employee set dept=@dept where id=@id

else if (@dept is null or @dept ='')

update employee set name= @name where id=@id

else

update employee set name= @name ,dept=@dept where id=@id

end

declare @name varchar(20)

declare @dept varchar(20)

set @dept='HR'

exec UpdateRec 1 ,@name, @dept

select \* from employee

-- functions : system defined / used defined

-- functions are scalar : which retuns a single value

-- difference in functions and stored procedures

-- functions are always compiled , sp are stored in pre-compiled form

-- function have select statement , sp can have all SQL statements

-- always have to return a value , sp may or may not return a value

-- sp if it retuns , it always returns value thats integer and that too one

-- in case you wnt to returm more than 1 value , we use output para

-- functions only have input para , sp have both input / output para

-- functions can be called in stored procedures

-- SCALAR VALUE

create function AddNumbers(@num1 int , @num2 int)

returns int

AS

Begin

return @num1 + @num2

End

Calling function

select dbo.AddNumbers(1,2)

create function combine\_strings(@firstname varchar(20), @lastname varchar(20))

returns varchar(40)

AS

begin

return @firstname + ' ' + @lastname

end

select dbo.combine\_strings('A','B')

select dbo.combine\_strings(name , dept) from employee

--select name=@name from employee where id=@id

-- Inline table valued function

create function GetAllEmployees()

returns table

as

return (select \* from employee)

select \* from dbo.GetAllEmployees()

create function GetAllEmployeesByDept(@dept varchar(20))

returns table

as

return (select \* from employee where dept=@dept)

select \* from dbo.GetAllEmployeesByDept('HR')

create proc A1

As

Begin

Select len('aaa')

Select \* from dbo.GetAllEmployees()

ENd

exec A1

What is an Error : Unexpected Outcome, unwanted

Error : Syntax Error > Error in syntax in language

Create proc A1

AS

Select \* from employee

Logical Error : Logical Error : We always get output, but that is not correct

Difficult to find out

Run time Errors / Exception : Which might or might not come

Depends upon what we enter at run time

First Name : Ajay

Last Name : aaa

Age :

We cant avoid Exceptions Handle them

Exception Handling

Try Begin Try ------- End Try

Any statement which can cause exeption

Catch Begin Catch ------- End Catch

This is a block which handles exceptions

create proc DivideNumbers(@num1 decimal , @num2 decimal , @res decimal output)

AS

Begin

set @res = @num1/@num2

end

declare @num1 decimal =10

declare @num2 decimal =0

declare @res decimal

exec DivideNumbers @num1 , @num2 , @res output

--print 'The result is ' + @res

print 'The result is ' + Convert(varchar(20), @res)

alter proc DivideNumbers(@num1 decimal , @num2 decimal , @res decimal output)

AS

Begin

begin try

set @res = @num1/@num2

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

end

declare @num1 decimal =10

declare @num2 decimal =0

declare @res decimal

exec DivideNumbers @num1 , @num2 , @res output

--print 'The result is ' + @res

print 'The result is ' + Convert(varchar(20), @res)

--ASCII code > Every character takes 1 byte

--UniCODE > N it sis used for multilinguual applications or data

CREATE TABLE persons

(

person\_id INT

PRIMARY KEY IDENTITY,

first\_name NVARCHAR(100) NOT NULL,

last\_name NVARCHAR(100) NOT NULL

)

CREATE TABLE deals

(

deal\_id INT

PRIMARY KEY IDENTITY,

person\_id INT NOT NULL,

deal\_note NVARCHAR(100),

FOREIGN KEY(person\_id) REFERENCES persons(

person\_id)

)

insert into

persons(first\_name, last\_name)

values

('John','Doe'),

('Jane','Doe');

insert into

deals(person\_id, deal\_note)

values

(1,'Deal for John Doe');

select \* from persons

select \* from deals

delete from persons where person\_id=1

create proc usp\_DisplayError

As

begin

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

exec usp\_DisplayError

alter proc DeletePersons ( @person\_id INT )

AS

BEGIN

BEGIN TRY

-- BEGIN TRANSACTION;

-- delete the person

DELETE FROM persons

WHERE person\_id = @person\_id;

-- if DELETE succeeds, commit the transaction

-- COMMIT TRANSACTION;

END TRY

BEGIN CATCH

exec usp\_DisplayError

end catch

end

exec DeletePersons 2

-- To see text of a procedure

sp\_helptext InsertStudent

sp\_help emp

Triggers

DML Triggers : When you perform Insertion Updation Deletion on tables

DDL Triggers : When you perform some action on database

DML Triggers

After/For

Instead of

create trigger T2

on emp

after insert

As

Begin

print 'Record inserted'

End

select \* from emp

insert into emp values (6,'Ajay',9000)

create trigger T3

on emp

after delete

As

Begin

rollback

print 'Record can not be deleted'

End

delete emp

create trigger T2

on emp

after insert

As

Begin

print 'Record inserted'

End

select \* from emp

insert into emp values (6,'Ajay',9000)

create trigger T3

on emp

after delete

As

Begin

rollback

print 'Record can not be deleted'

End

sp\_settriggerorder T4, FIRST, 'UPDATE'

exec sp\_settriggerorder tr\_au\_upd, FIRST, 'DELETE'

delete emp

select \* from emp

create trigger T4

on emp

for insert,delete,update

As

Begin

rollback

print 'Record can not be added or deleted or modified'

End

delete emp

select \* from sys.triggers

CREATE TRIGGER tr\_au\_upd ON EMp

AFTER UPDATE,INSERT,DELETE

AS

PRINT 'TRIGGER OUTPUT' + CONVERT(VARCHAR(5),@@ROWCOUNT)

+ 'ROW UPDATED'

GO

delete emp