Select \* from employee

how many number of employees are working in the depart of 'hetesh'

Select count(eid) from Employee where DeptId=(Select deptid from employee where ename ='hetesh')

declare @avgsal int

select @avgsal= AVG(salary) from Employee

Set @avgsal = @avgsal+ 1000

PRINT @avgsal

GO

IF ELSE

Syn : IF <condition>

BEGIN

stmts

stmst

END

ELSE

BEGIN

stmts

stmts

END

EX:

DECLARE @avgsal INT

SELECT @avgsal = AVG(Salary) FROM Employee

IF @avgsal >=5000

BEGIN

PRINT 'NO NEED TO INCREASE SALARIES...'

END

ELSE

BEGIN

PRINT 'YES - NEED TO INCREASE SLARIES ...'

END

GO

CASE

---------------------

Same like the switch case in programing lang

CASE input\_expression

WHEN when\_expression THEN result\_expression

WHEN when\_expression THEN result\_expression

...

ELSE else\_result\_expression

END

EX:

SELECT Eid, Ename,

CASE Gender

WHEN 'M' THEN 'Male'

WHEN 'F' THEN 'Female'

WHEN 'T' THEN 'Tran'

ELSE 'NA'

END 'Gender'

FROM Employee

WHILE

--------------------------

Used to repeat a set of statements unitl the condition evaluates false

Syn:

WHILE condition

BEGIN

sqlstmts

sqlstmts

END

Ex:

declare @avgsal int

select @avgsal = AVG(salary) from Employee

while @avgsal < 5000

begin

update employee set salary = salary +1

select @avgsal = AVG(salary) from Employee

end

go

select \* from Employee

Stored procedures

-----------------------------

\* is a set of staments which will execute when we invoke that SP

\* it will be compiled only one time and get saved at DB level

\* When ever we invoke the SP , the already compiled instance at DB will get executed

\* Can work on DDL , DML stmts

\* Can accept parameters also

CREATE PROCEDURE prcName (param1 datatype, param2 datatype,..)

AS

BEGIN

stmts;

stmts;

END

EX:

CREATE PROCEDURE sp\_GetAllEmployees

AS

BEGIN

SELECT \* FROM Employee

END

exec sp\_GetAllEmployees

\* Procedure with params

CREATE PROCEDURE sp\_GetEmployeeByDeptId (@deptId int)

AS

BEGIN

SELECT \* FROM Employee WHERE DeptId = @deptId

END

EXEC sp\_GetEmployeeByDeptId 30

EXEC sp\_GetEmployeeByDeptId 10

EXEC sp\_GetEmployeeByDeptId 5

INSERT/UPDATE

Ex: With multiple parameters

CREATE PROCEDURE sp\_AddOrUpdateEmp (@eid int, @ename varchar(100),@salary int, @mngrid int , @deptid int, @gender varchar(5))

AS

BEGIN

IF EXISTS(SELECT \* FROM Employee WHERE Eid = @eid)

BEGIN

UPDATE Employee SET Ename=@ename, Salary = @salary, MngrId = @mngrid , DeptId = @deptid , Gender = @gender

WHERE Eid = @eid

END

ELSE

BEGIN

INSERT INTO Employee(Eid,Ename,Salary,MngrId,DeptId,Gender) VALUES(@eid,@ename,@salary,@mngrid, @deptid,@gender)

END

END

EXEC sp\_AddOrUpdateEmp 2002, 'Agarwal',3000,1004,20,'M'

EXEC sp\_AddOrUpdateEmp 2003, 'Rakesh',3500,1006,20,'F'

EXEC sp\_GetAllEmployees

EXEC sp\_AddOrUpdateEmp 1014, 'Ayesha',3200,1006,20,'F'

CREATE PROCEDURE sp\_DelEmpById (@id int)

AS

BEGIN

DELETE FROM Employee WHERE Eid = @id

END

Exec sp\_DelEmpById 1014

USER DEFINED FUNCTIONS

-------------------------------------------

\* You can create functions in situations when you need to implement a programming logic that does not involve any

permanent changes to the database objects outside the function

\* Can not work on DDL Staments

\* Need to be executed with in a select statement

CREATE FUNCTION FuncName (Param1 type,Parm2 type,.. )

RETURNS Type

AS

BEGIN

SQLStmts,

…

RETURN value;

END

Select \* from employee

CREATE FUNCTION fn\_GetAvgSalaryByDeptId (@deptid int)

RETURNS INT

AS

BEGIN

DECLARE @avgsal INT

SELECT @avgsal = AVG(Salary) FROM Employee WHERE DeptId = @deptId;

RETURN @avgsal;

END

\* A user defined function can be invoked in a select statement . must be preceded with schema ie dbo

SELECT dbo.fn\_GetAvgSalaryByDeptId(70)