------------DATA DEFINITION LANGUAGE COMMANDS--------------

--To create Table

Create table emp (

EmpID int primary key not null,

Ename varchar(50) not null,

GenderID int,

Age int)

--To View Table

select \* from emp

select \* from GenderID

--To insert Values into Table

insert into emp values (1, 'Anshu', 1, 23)

--To create Table

Create table GenderID (

GenderID int primary key Not Null,

Gender varchar(20)

)

--To insert values into tables

insert into GenderID values (1, 'Male')

insert into GenderID values (2, 'Female')

insert into GenderID values (3, 'Not Availble')

--To add Foreign Key into tables

alter table emp

add constraint emp\_GenderId\_FK FOREIGN KEY (GenderId) references GenderID(GenderID)

--To Change datatype of column

Alter Table emp alter column Ename varchar(50)

--To Add new column into table

Alter Table emp add Salary int

--To drop column

Alter Table emp drop column EmailID

--To Rename column

sp\_rename emp.Ename, Empname, column

--To create Table

Create table vender (

venderID int primary key Not Null,

v\_name varchar(20)

)

--To insert values into tables

insert into vender values (1, 'Karan')

insert into vender values (2, 'Aman')

insert into vender values (3, 'Vimal')

--To View Table

select \* from vender

--To Delete all Records

Truncate table vender

--To Drop/Delete Table

Drop table vender

--To View Structure of Table

sp\_help emp

-----------------DATAMANIPULATINGLANGUAGE----------------

--To insert values into tables

insert into GenderID values (1, 'Male')

insert into GenderID values (2, 'Female')

insert into GenderID values (3, 'Not Availble')

--To View Table

select \* from emp

select \* from GenderID

--To update values into column with condition (where)

update emp

set Salary = 13500

where EmpID = 7

--To Delete values from table with condition (where)

Delete from emp where EmpID = 7

--To view Specific Column

select EmpID, Ename, Age from emp

--To view Specific Column with conditions (where)

select EmpID, Ename, Age from emp where EmpID = 5

--To insert Values into Table

insert into emp values (7, 'Karan', 1, 26, 'ka@ka.com')

--To view columns with different name as you want

select EmpID as EID, Ename as Emp\_name from emp

--To view columns with different name as you want

select EmpID as EID, Ename as Emp\_name from emp

--To view columns with different name as you want with math functions

select EmpID as EID, Ename as Emp\_name, Salary\*12 as annuall\_salary from emp

--To use identity function seed (use for increment int values with spevific given condition

create table vender(

V\_ID int identity(100,1),

V\_name varchar(50)

)

-----------Built in Function in SQL---------------------

--Mathematical Functions

select ABS(-15)

select CEILING(15.000)

select FLOOR(15.000)

select SQUARE(5)

select SQRT(25)

select POWER(3,2)

select SIGN(42)

select SIGN(-42)

select PI()

select LOG(2)

select LOG10(10)

select SIN(30)

select COS(30)

select TAN(30)

--String Functions

select ASCII('Z')

select CHAR(90)

select LEFT('ANSHU', 2)

select RIGHT('ANSHU', 2)

select LEN('ANSHU')

select LOWER('ANSHU')

select UPPER('ANSHU')

select LTRIM(' ANSHU')

select RTRIM('ANSHU ')

select REPLACE('JACK AND JUE', 'j', 'BL')

select REVERSE('ANSHU')

select REPLICATE('ANSHU',4)

select ('ANSHU' + SPACE(10) +'IS')

select SUBSTRING('HELLO', 1, 3)

--DATETIME FUNCTIONS

select GETDATE()

select DAY(GETDATE())

select MONTH(GETDATE())

select YEAR(GETDATE())

select GETUTCDATE()

select DATENAME(DW, GETDATE())

select DATEPART(DD, GETDATE())

select DATEADD(DD, 5, GETDATE())

select DATEDIFF(MM, '2012-12-15', GETDATE())

--CONVERSION FUNCTION

select CAST(1521 as int)

select CONVERT(varchar(10), GETDATE(), 113)

--AGGIGREGATE FUNCTIONS

select SUM(Salary) as total\_salary from emp

select AVG(Salary) as avg\_salary from emp

select MAX(Salary) as max\_salary from emp

select MIN(Salary) as min\_salary from emp

select COUNT(Salary) as no\_of\_salaries from emp

select COUNT(DISTINCT GenderID) from emp

--ARITHMETIC OPERATOR

select 100+250

select 100\*250

select 100%250

select 100-250

select 20/5+20/2

select 30.50+20

--ARITHMETIC OPERATOR WITH CASE EXAMPLE

create table student(

Sid int, Sname varchar(50),

Maths int, Physics int,

Che int, Total int,

Average int, Class varchar(max))

update student

set Total = Maths+Physics+Che

update student

set Average = Total/3

update student

set Class=

CASE

WHEN average>=60 then 'FIRST CLASS'

WHEN average>=50 then 'SECOND CLASS'

WHEN average>=40 then 'THIRD CLASS'

ELSE

'FAIL'

END

Select \* from student

insert into student(Sid,Sname,Maths,Physics,Che) values(1,'Anshu',42,58,72)

insert into student(Sid,Sname,Maths,Physics,Che) values(2,'Aman',40,55,70)

insert into student(Sid,Sname,Maths,Physics,Che) values(3,'Vimal',32,51,60)

insert into student(Sid,Sname,Maths,Physics,Che) values(4,'Karan',42,53,70)

insert into student(Sid,Sname,Maths,Physics,Che) values(5,'Anjali',46,55,52)

insert into student(Sid,Sname,Maths,Physics,Che) values(6,'Archana',49,51,53)

insert into student(Sid,Sname,Maths,Physics,Che) values(7,'Sonu',48,58,72)

--Assignment Operator

Select \* from emp where Ename='Anshu' and Salary=10000

Select \* from emp where Ename='joshitha' or Salary=13000

Select \* from emp where not Ename='Anand'

Select \* from emp where Salary between 12000 and 50000

Update emp set ename='SAI' where EmpID=101 and Salary=25000

SELECT \* FROM emp WHERE Salary>10000

SELECT \* FROM emp WHERE Salary>=9000 AND Salary<=15000

SELECT \* FROM emp WHERE Salary NOT BETWEEN 9000 AND 15000

SELECT \* FROM emp WHERE Ename LIKE 'A%'

SELECT \* FROM emp WHERE Ename LIKE '%u'

SELECT \* FROM emp WHERE Ename LIKE '%n%'

SELECT \* FROM emp WHERE Ename LIKE '--'

SELECT \* FROM emp WHERE Ename LIKE '%R%' AND Salary>9000

--Write a Query to display the employee details whose greater than ram

SELECT \* FROM emp WHERE Ename > 'RAM'

--Query used for finding EmpID whose EmpID starts with 1 and also end with 1

SELECT \* FROM EMP WHERE EmpID LIKE '1%1'

UPDATE emp SET Ename='Kristy' WHERE EmpID=101 OR EmpID=103 OR EmpID=107

UPDATE emp SET Ename='Kristy' WHERE Age IS NULL

UPDATE emp SET Salary=12000 WHERE EmpID=1 AND Ename LIKE 'R%'

UPDATE emp SET EmpID=3 WHERE Ename='-a'

UPDATE emp SET Salary=8500 WHERE EmpID=1 OR EmpID=6

(OR)

UPDATE emp SET Salary=8500 WHERE EmpID IN(1,5)

UPDATE emp SET Salary=8500 WHERE EmpID NOT IN (1,5)

UPDATE emp SET Salary=15000 WHERE Ename='%a' AND EmpID=6

UPDATE emp SET Salary=5500 WHERE EmpID LIKE '%4' AND EmpID LIKE '2%'

UPDATE emp SET Salary=25000 WHERE Salary<10000 AND Ename LIKE '%A%' AND EmpID IN(5)

UPDATE emp SET Salary=10000 WHERE Salary BETWEEN 8500 AND 9000

--To View Table

select \* from emp

select \* from GenderID

--To Create Table

CREATE TABLE EMP\_HYD (EID INT,ENAME VARCHAR(50), SALARY MONEY)

CREATE TABLE EMP\_CHENNAI(EID INT, ENAME VARCHAR(50))

--To insert values

Select \* from EMP\_CHENNAI

insert into EMP\_CHENNAI values(1,'Vinay')

insert into EMP\_CHENNAI values(2,'Arun')

insert into EMP\_CHENNAI values(3,'Sherry')

insert into EMP\_CHENNAI values(4,'Mary')

insert into EMP\_CHENNAI values(5,'Kite')

insert into EMP\_CHENNAI values(6,'Kanchana')

insert into EMP\_CHENNAI values(7,'Sunder')

--Set Operator

--Union

Select Ename from EMP\_HYD

Union

Select Ename from EMP\_CHENNAI

--Union all

Select Ename from EMP\_HYD

Union all

Select Ename from EMP\_CHENNAI

--Intersect

Select Ename from EMP\_HYD

Intersect

Select Ename from EMP\_CHENNAI

--Except

Select Ename from EMP\_HYD

Except

Select Ename from EMP\_CHENNAI

Select Ename from EMP\_CHENNAI

Except

Select Ename from EMP\_HYD

--Clauses in SQL

--Where

SELECT \* FROM emp WHERE Salary=10000

--Orderby

SELECT \* FROM emp ORDER BY EmpID

SELECT \* FROM emp ORDER BY EmpID desc

--Top 3

UPDATE TOP 3 emp SET ENAME = 'Ayush'

DELETE TOP 3 FROM emp

--Count, Sum, Having

SELECT COUNT(\*) as total\_records FROM emp

SELECT Age, COUNT=COUNT(\*) FROM emp GROUP BY GenderID

SELECT Age, TOTAL\_SALARY=SUM(Salary) FROM EMP GROUP BY GenderID

SELECT Age,COUNT=COUNT(\*) FROM EMP GROUP BY EmpID HAVING COUNT (\*) >3

--Synonyms

Create synonym syn\_emp for employee

Drop synonym syn\_emp

--Syntax to Creating a table from an existing table

Select \* into New\_Emp from emp

Select EmpID, Ename into Test\_Emp from emp

Select \* into Dummy\_Emp from emp where 1=2

--To copying the values from another table

Insert into Dummy\_Emp select \* from emp

----------Constraint in SQL-------------

--Unique Key Constraint

create table emp1(EID int unique,ENAME varchar(50) unique,SALARY money)

--Not Null Contraint

create table emp2(EID int not null, ENAME varchar(50) not null, SALARY money)

--Check Constrait (it will allow specific condition what you have mentioned in ur statement)

create table emp3(eno int,ename varchar(50),age int check(age between 20 and 30))

--Primary Key Constraint

create table emp4(EID int primary key,ENAME varchar(50),SALARY money)

--Foreign Key Constraint

create table Department(Deptno int primary key,DNAME varchar(50),LOCATION varchar(max))

insert into Department values(10,'Sales','Chennai')

insert into Department values(20,'Production','Mumbai')

insert into Department values(30,'Finance','Delhi')

insert into Department values(40,'Research','Hyderabad')

create table Employee1(EID int,ENAME varchar(50),SALARY money,Deptno int foreign key references Department(Deptno))

insert into Employee1 values(101,'Sai',35000,10)

insert into Employee1 values(102,'Pavan',45000,20)

insert into Employee1 values(103,'Kamal',74000,30)

insert into Employee1 values(104,'Ravi',58000,40)

--Foreign Key Rule: On delete cascade

create table Emp5(EID int,ENAME varchar(50),SALARY money,

Deptno int foreign key references Department(Deptno)on delete cascade on update cascade)

--Making a Relationship between Three Tables

--Case 1

Create table CUSTOMER (CID Int primary key,CNAME Varchar(20),MAILID Varchar(40))

Insert customer values(1,'a','a@gmail.com'),

(2,'b','b@gmail.com'),

(3,'c','c@gmail.com')

--Case 2

Create table PRODUCTS (PCODE Int primary key,PNAME varchar(50),PRICE money)

Insert products values(10,'C', 500),

(20,'C++', 1000),

(30,'.NET',35000),

(40,'SQL', 1800)

Create table ORDERS (ORID Int primary key, ORDATE date, QUANTITY int,

CID Int foreign key references CUSTOMER(cid)on update cascade on delete cascade,

PCODE Int foreign key references PRODUCTS(pcode) on update cascade on delete cascade)

Update CUSTOMER

set cid=100 where cid=1

Delete from CUSTOMER

where cid=100

Update PRODUCTS

set ORID=21

where ORID=20

Delete from PRODUCTS

where ORID=30

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

--Adding Constraint on an existing table

--Adding Primary Key on an existing table:

ALTER TABLE EMPLOYEE1 ALTER COLUMN EID INT NOT NULL

ALTER TABLE EMPLOYEE1 ADD CONSTRAINT X PRIMARY KEY(EID)

--Adding Unique Key on an existing table:

ALTER TABLE EMPLOYEE1 ADD CONSTRAINT X UNIQUE(ENAME)

--Adding Check Key on an existing table:

ALTER TABLE EMPLOYEE1 ADD CONSTRAINT X CHECK(SALARY>8000)

--Adding Foreign Key on an existing table:

ALTER TABLE DEPT ADD CONSTRAINT Y FOREIGN KEY (EID) REFERENCES EMPLOYEE(EID) ON UPDATE CASCADE ON DELETE CASCADE

--Dropping Constraint from an existing table:

ALTER TABLE EMPLOYEE DROP CONSTRAINT X

------------Joins in SQL-------------------

--EQUIJOIN

SELECT \* FROM Employee1, Department WHERE (Employee1.EID = Department.Deptno)

SELECT E.EmpID,E.ENAME, E.Salary, D.Deptno, D.DNAME FROM EMP E, Department D WHERE E.EmpID=D.Deptno

--INNER JOIN

SELECT \* FROM emp E

INNER JOIN Department D

E.EmpID=D.Deptno

--OUTER JOIN

SELECT \* FROM EMP

LEFT OUTER JOIN DEPT ON

EMP.EID=DEPT.DNO

SELECT \* FROM EMP

RIGHT OUTER JOIN DEPT ON

EMP.EID=DEPT.DNO

SELECT \* FROM EMP

FULL OUTER JOIN DEPT ON

EMP.EID=DEPT.DNO

--NON EQUIJOIN

SELECT \* FROM EMP, SALGRADE

WHERE (SALARY >LOWSAL) AND (SALARY < HIGHSAL)

--SELF JOIN

SELECT E.EID,E.ENAME,E.SALARY, M.MID, M.ENAMEMANAGERSNAME, M.SALARY FROM EMP E, EMP M

WHERE E.EID=M.MID

--CROSS JOIN

SELECT \* FROM EMP,DEPT

SELECT \* FROM EMP CROSS JOIN DEPT

--NATURAL JOIN

SELECT EID, ENAME, SALARY, DNO, DNAME FROM EMP E, DEPT D

WHERE E.EID=D.DNO

--Making To Joins Three Tables

CREATE TABLE STUDENTS (

SID Int, SNAME varchar(20),

SMBNO char(10),CID Int)

INSERT STUDENTS VALUES (1,'aa','7894561233',10),

(2,'bb','9874563211',20),

(3,'cc','8749653215',30),

(4,'dd','7788996655',40)

CREATE TABLE COURSES (

CID int,CNAME Varchar (20),

CFEE decimal(6,2))

INSERT COURSES VALUES(10,'c',500),

(20,'c++',1000),

(50,'sql',1800),

(60,'.net',3500),

(70,'sap',8000)

CREATE TABLE REGISTER(

SNO int,REGDATE date,

CID Int)

INSERT REGISTER VALUES

(100,'2014/10/20',10),

(101,'2014/10/21',80),

(102,'2014/10/22',90)

Select \* from COURSES c

inner join students s on c.cid=s.cid

inner join register r on s.cid=r.cid

--------------TRANSACTION CONTROLL LANGUAGE------

--BEGIN TRANSACTION: Begin Transaction commandis used to start the transaction.

--Begin Transaction with name isused to add nested transactions.

--COMMIT: Commit command is used to end the transaction and save the data permanent part of the database (or)

--it is used to make the transaction is permanent so we cannot undoor recall the records.

--Syntax:

-- BeginTransaction

-- <WriteStatements>

-- Commit

BEGIN TRANSACTION

INSERT INTO emp VALUES(8,'Anmol',1,26,'anm@anm.com', 6500)

INSERT INTO emp VALUES(9,'sujatha',2,27,'s@s.com',25000)

COMMIT

--ROLLBACK: Rollback command is used to undo the transactions and gets back to the initial state where transaction started

--Syntax:

-- BeginTransaction

-- Rollback

BEGIN TRANSACTION

DELETE FROM emp WHERE EmpID=8

DELETE FROM emp WHERE EmpID=9

BEGIN TRANSACTION

ROLLBACK

--SAVEPOINT: Savepoint is used for dividing (or) breaking a transaction into multiple units.

--So that user will have a chance of rollbacking a transaction upto alocation.

--Syntax:

-- Begin Transaction

-- Save transaction <transaction name>

-- <Write Statements>

BEGIN TRANSACTION

UPDATE emp SET SALARY=99000 WHERE EmpID=1

UPDATE emp SET SALARY=88000 WHERE EmpID=2

SAVE TRANSACTION S1

UPDATE emp SET SALARY=77000 WHERE EmpID=3

UPDATE emp SET SALARY=66000 WHERE EmpID=4

SAVE TRANSACTION S2

UPDATE emp SET SALARY=55000 WHERE EmpID=5

UPDATE emp SET SALARY=44000 WHERE EmpID=6

--CASE 1:

BEGIN

TRANSACTION ROLLBACK

--CASE 1:

BEGIN

TRANSACTION ROLLBACK S1

--CASE 1:

BEGIN

TRANSACTION ROLLBACK S2

---------------Sub Query--------------------

--Syntax: select \* from <TableName> where (condition)(select \* from……..(Select \* from…..(select \* from……..)))

--Nested Subqueries:

--To find the details of employee who is earning the highest salary

select\* from employee

where salary =(select MAX(salary) from employee)

--To find the details of employee who is earning second highest salary

select\* from employee

where Salary=(select MAX(salary) from employee

where Salary<(select MAX(salary) from employee))

--To find the details of employee who is earning third highest salary.

select\* from employee

where Salary=(select MAX(salary) from employee

where Salary<(select MAX(salary) from employee

where Salary<(select MAX(salary) from employee)))

--To display employee details who are working in branch\_id 1.

select \* from employee

where emp\_id in (select emp\_id FROM employee

where branch\_id=1)

--To display employee details whoare working in JAVA or HR department.

select\* from employee1

where EID IN (select EID FROM employee

where DNAME='.NET' OR DNAME='HR')

--Co-related Queries

--Syntax:

-- SELECT \* FROM Employee Emp1 WHERE (N-1)=or N>(

-- SELECT COUNT(DISTINCT(Emp2.Salary)) FROM Employee Emp2 WHERE Emp2.Salary>Emp1.Salary)

--To display top 2 salaries list from employee table.

Select\*from emp E

where 2>(select count(salary) from emp M

where M.salary>E.salary)

-- OR

Select \* from emp E

where 2>(select count(Distinct salary) from emp M

where M.salary>E.salary)

--To display least 2 salaries list from employee table.

Select \* from emp E

where 2>(select count(salary) from emp M

where M.salary<E.salary)

-- OR

Select \* from emp E

where 2>(select count(Distinct salary) from emp M

where M.salary<E.salary)

--To display “Nth” salary from employee table

Select \* from emp E

where 0=(select count(salary) from emp M

where M.salary>E.salary)

--To get the details of the department in which employee are working

Select \* from department D Exists (select \* from employee E where E.deptno=D.deptno)

--To get the details of the department in which employee are not working.

Select\* from department D not Exists (select\* from employee E where E.deptno=D.deptno)

--Syntax to Find Any Position Record From A table

SELECT \* FROM(SELECT\* ,ROW\_NUMBER() OVER (ORDER BY

EmpID) AS RowNum FROM emp)emp

WHERE RowNum=5

--Syntax to Delete Duplicate Records from a table contains more than two same duplicate values:

(select \* ,ROW\_NUMBER() over (partition by sid,sname,fee order by sid,sname,fee)rownum from student)

delete from duplicates where rownum>1

--------------------Index in SQL--------------------

--Non-Clustered Indexes

--Syntax:

-- CreateIndex<IndexName>on<TableName>(ColumnName)

Create index demoindex on EMP (EmpID)

--Complex view

--Syntax:

-- create view <viewname> as select \* from <tablename>

create view simpleview as select \* from emp

--------------------T/SQLProgramming----------------------

--Declaring Variables In T/SQL Program

--Syntax:

-- Declare @<var> [as] <datatype>[size]

declare @eid int

declare @ename varchar(50)

--Assigning Values to variables

--Syntax:

-- Set@ <var>=<value>

Set @Eid=101

Set @ename='SAI'

--Printing Values of Variables

--Syntax:

-- Print @<var>

Print @Eid

--Structure of T/SQL Program

--Declare

-- @<var1> [datatype][size]

-- Set @<var> = <values>

-- <Statements>

-- Print @<var>.

--Write a T/SQL program to input two values and inter change the variable values

declare

@a int,

@b int,

@c int

set @a=10

set @b=20

set @c=@a

set @a=@b

set @b=@c

print @a

print @b

--Write a T/SQL program to input studentid,name,marks and find the total marks of a student.

declare

@stdno int,

@stdname varchar(50),

@m1 int,

@m2 int,

@m3 int,

@tm int

set @stdno=101

set @stdname='SAI'

set @m1=75;

set @m2=85

set @m3=65;

set @tm=@m1+@m2+@m3

print @stdno

print @stdname

print @tm

--Write a T/SQL programer to perform arithematic operation

declare

@a int,

@b int,

@c int,

@d int,

@e int,

@f int;

set @a=10

set @b=12;

set @c=@a+@b

set @d=@a-@b

set @e=@a\*@b

set @f=@a/@b

print @c

print @d

print @e

print @f

--If-Else Statement

/\*Syntax:

If (condition)

{

Statements

}

Else

{

Statements

} \*/

--Write T/SQL program to find big number from two variables

declare @a int, @b int

set @a=30;

set @b=20;

if (@a>@b)

print 'a is big'

else if(@a=@b)

print'Both are equal'

else

print 'B is big'

--To find positive and negative

declare

@a int

set @a=10

if(@a>0)

print 'a is positive'

else

if(@a=0)

print 'a is neutral'

else

print 'a is negative'

--To find the number is even or odd

declare

@a int

set @a=4

if((@a%2)=0)

print 'a is even'

else

print 'a is odd'

--If there are multiple statements being enclosed between each block then we can put the munder Begin and End Statements

DECLARE @WEEK INT

SET @WEEK=DATEPART(DW, GETDATE())

IF @WEEK=1

PRINT 'SUNDAY' ELSE IF @WEEK=2

PRINT 'MONDAY' ELSE IF @WEEK=3

PRINT 'TUESDAY' ELSE IF @WEEK=4

PRINT 'WEDNESDAY' ELSE IF @WEEK=5

PRINT 'THURSDAY' ELSE IF @WEEK=6

PRINT 'FRIDAY' ELSE IF @WEEK=7

PRINT 'SATURDAY'

--The case function what we have discussed under the System Functions canal so be used here as following

DECLARE @WEEK INT

SET @WEEK=DATEPART(DW, GETDATE())

SELECT CASE @WEEK

WHEN 1 THEN 'SUNDAY'

WHEN 2 THEN 'MONDAY'

WHEN 3 THEN 'TUESDAY'

WHEN 4 THEN 'WEDNESDAY'

WHEN 5 THEN 'THURSDAY'

WHEN 6 THEN 'FRIDAY'

WHEN 7 THEN 'SATURDAY'

END

--This can be written in the second style of the CASE Statement also that has been discussed in the SQL

DECLARE @WEEK INT

SET @WEEK=DATEPART(DW, GETDATE())SELECT CASE

WHEN @WEEK=1 THEN'SUNDAY'

WHEN @WEEK=2 THEN 'MONDAY' WHEN @WEEK=3 THEN 'TUESDAY'

WHEN @WEEK=4 THEN'WEDNESDAY' WHEN @WEEK=5 THEN 'THURSDAY'WHEN @WEEK=6 THEN 'FRIDAY' WHEN @WEEK=7 THEN 'SATURDAY'

END

--While Loop

/\* WHILE Boolean expression

[BEGIN]

<sql\_statement|statement\_block>

[BREAK]

<sql\_statement|statement\_block>

[CONTINUE]

<sql\_statement|statement\_block>

[END] \*/

DECLARE @X INT SET @X=0

WHILE @X<10 BEGIN

SET @X=@X+1

PRINT @X

END

DECLARE @X INT SET @X=0

WHILE @X<10 BEGIN

SET @X=@X+1IF @X=6 BREAK

PRINT @X

END

DECLARE @X INT SET @X=0

WHILE @X<10 BEGIN

SET @X=@X+1

IF @X=6 CONTINUE

PRINT @X

END

--Assinging values from columns into variables

--SELECT @<var> = <col\_name> [,……n] FROM <table\_name> [CONDITIONS]

declare @ename nvarchar(50)

SELECT @ENAME=ENAME FROM EMP WHERE EmpID =6

--A simple TSQL program which takes the Empno and prints the Name and Salary

DECLARE

@EmpID INT,

@ENAME VARCHAR(50),

@SAL MONEY

SET @EmpID=1

SELECT @ENAME=ENAME, @SAL=salary FROM EMP WHERE @EmpID=@EmpID

PRINT upper(@ENAME) +' EARNS '+'$'+CAST(@SAL AS VARCHAR)

/\* A Program which takes the Empnoand increments the Salary of the person on the following criteria:

If Job is President increment with 10%

If Job is Manager increment with 8%

If Job is Analyst increment with 6%

If Job is anything other increment with5% \*/

DECLARE @EmpID INT,

@JOB VARCHAR(50)

SET @EmpID=7

SELECT @JOB=JOB FROM EMP WHERE EmpID=@EmpID

IF @JOB='PRESIDENT'

UPDATE EMP SET salary = salary +salary\* 0.1 WHERE EmpID= @EmpID

ELSE IF @JOB='MANAGER'

UPDATE EMP SET salary = salary +salary\* 0.10WHERE EmpID=@EmpID

ELSE IF @JOB='ANALYST'

UPDATE EMP SET salary = salary +salary\* 0.06 WHERE EmpID=@EmpID

ELSE

UPDATE EMP SET salary = salary +salary\* 0.05 WHERE EmpID=@EmpID

select \* from emp

update emp set job='PRESIDENT' where EmpID=1

update emp set job='MANAGER' where EmpID=2

update emp set job='ANALYST' where EmpID=3

update emp set job='Receptionist' where EmpID=4

update emp set job='ANALYST' where EmpID=5

update emp set job='PRESIDENT' where EmpID=6

update emp set job='Account Assistant' where EmpID=7

update emp set job='ANALYST' where EmpID=8

update emp set job='MANAGER' where EmpID=9

-------------------Stored Procedures/Procedure-------------------

/\* Syntax:

Create Procedures <ProceduresName>

As

Begin

<Statements>

End \*/

/\* How to Call a Stored Procedures/Procedure

Syntax:

Exec <Procedurename> \*/

create procedure Test1

as

begin

print 'WELCOME TO STORED PROCEDURES'

end

--Passing Parameters to Procedures

/\* Syntax:

Create Procedures<ProceduresName>

(Passing parameters)

As

Begin

<Statements>

End \*/

--Write a program to add the two values with Parameters Procedure

create procedure test2

(@a int,@b int)

as

begin

declare @c int

set @c=@a+@b

print'Additionoftwo variablesare:- '+cast(@c as varchar)

end

--Write a program to perform arithmetic operations of two values with Parameters Procedure

create procedure test3

(@a int,@b int)

as

begin

declare @x int,@y int,@z int,@s int

set @x=@a+@b

set @y=@a-@b

set @z=@a\*@b

set @s=@a/@b;

print'Add of two variable sare:- '+cast(@x as varchar)

print'Sub of two variables are:- '+cast(@y as varchar)

print'Mul of two variables are:- '+cast(@z as varchar)

print'Div of two variables are:- '+cast(@s as varchar)

end

--create a procedure to display employee details to the user

create procedure spselect

as

begin

select \* from emp

end

--create a procedure to accept employee ID and delete the record from employee table

create procedure spdel

@eid int

as

begin

delete from emp

where EmpID = @eid

end

--create a procedure to accept employee

create procedure spupdate

@eid int,@ename varchar(max),@salary money,@address char(30)

as

begin

update Employee set EmpName=@ename,Salary=@salary,Address=@address where EmpID=@eid

end

--create a procedure to add records inemployee table

create procedure spinst

@eid int,@ename varchar(50),@salary money,@address varchar(50)

as

begin

insert into Employee values(@eid,@ename,@salary,@address)

end

--create a procedure to insert records in two tables

create procedure spinserttwotables

@eid int,@ename varchar(50),@salary money,@Address char(40),@Deptno int,@Dname char(30),@Loc char(20)

as

begin

insert into Employee values(@eid,@ename,@salary,@Address,@Deptno)

insert into Dept values(@Deptno,@Dname,@Loc)

end

--A Procedure with Default Values

CREATE PROCEDURE PROC3(@X INT= 100, @Y INT)

AS

BEGIN

DECLARE @Z INT

SET @Z=@X+@Y

PRINT 'The SUM of the 2 Numbers is: '+ CAST(@Z AS VARCHAR)

END

--A Procedure which takes the Empno and prints the Net Salary of the Employee

CREATE PROCEDURE Net\_Sal(@Empno int)

As

Begin

Declare @VSal money,@NSal money,@VPF money,@VPT money

EXEC Deductions @Empno,@VPF OUTPUT,@VPT OUTPUT SELECT @NSal=salary FROM Emp WHERE EmpID=@Empno

SET @NSal= @VSal-@VPF-@VPT

Print 'Net Salary of the Employee is:' +Cast(@NSal as Varchar)

End

--How To Drop Stored Procedure

--Drop Procedure <ProcedureName>

------------------Stored Functions/Functions--------------

--Scalar-ValuedFuction: In this case we can return a attribute data type as an output from the function

/\* Syntax:

Create Function <Function Name>

(@parameter <Data Type>[size])

Returns<return attribute datatype>

As

Begin

<Function Body>

Return <return attribute name>

End \*/

--Create a function to return the cube of the given value.

create function fcube(@x int)returns int

as

begin

return @x\*@x\*@x

end

Output: select dbo.fcube(3)

--Create a function that takes an employee id and returns the salary of that employee.

CREATE FUNCTION fsal (@eid int)

returns money

as

begin

declare @sal money

select @sal=Salary from emp where empid=@eid

return @sal

end

Output: select dbo.fsal(1)

--Table-Valued Fuction: In this case we can return a table as an output from the function.

/\* Syntax:

Create Function <FunctionName>( @parameter <DataType> [size])

Returns <Table>

As

Return <return select statement> \*/

create function ft1( @add varchar(50))

returns table

as

return (select\* from emp where address= @add)

Output: select \* from ft1('Delhi')

create function saidata (@deptno int)

returns table

as

return (select e.eid, e.ename,e.salary,d.deptno,d.dname,d.location from emp e

inner join dept d on e.deptno=d.deptno where e.deptno=@deptno)

Output: Select \* from saidata(10)

--How T oDrop Functions

-- Drop Function <FunctionName>

-----------------------TRIGGERS----------------------------------

--DML Triggers: DML triggers execute when the user tries to modify or change data through data manipulation language events.

--Those are Inserting, Update and Delete statements on the table

/\* Syntax:

Create Trigger <TriggerName>

on <tableName>

For [Insert, Update, Delete]

AS

Begin

<Statements>

End \*/

--trigger that will convert the dname and location int ouppercase when the user insert in lowercase

create trigger per\_trg

on person

after insert

as

begin

declare @pid int,@pname varchar(50),@loc varchar(50)

select @pid=pid,@pname=pname,@loc=loc from inserted

update person set pname=upper(@pname),loc=upper(@loc)

where pid=@pid

end

--Create a trigger to restric DML operations on the table

create trigger nnn

on person

for insert, update, delete

as

begin

print'DML OPeration are Not Allowed'

rollback transaction

end

--A Trigger that will restrict the operations to be performed before 9 A.M and after 5 P.M

CREATE TRIGGER EMP\_TRG

ON EMP

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

DECLARE @DT INT

SET @DT=DATENAME(HH, GETDATE())

IF @DT NOT BETWEEN 9 AND 16

BEGIN

ROLLBACK

RAISERROR('CANNOT PERFORM DMLOPERATIONSNOW', 15,1)END

END

--A program which will restrict the Delete operation if the Job of the person is Manager

Create TRIGGER EMP\_DELETE\_TRG

ON EMP AFTER DELETE

AS

BEGIN

DECLARE @JOB VARCHAR(50)SELECT @JOB = job FROM DELETED

IF @JOB='MANAGER'

BEGIN ROLLBACK

RAISERROR('CANNOT DELETE MANAGER FROM THE TABLE', 15, 1)

End

END

--A Trigger which will restrict to update the Salary of the Employee if the New Salary is less than the Old Salary.

CREATE TRIGGER EMP\_UPDATE\_TRG

ON EMP

AFTER UPDATE

AS

BEGIN

DECLARE @OLDSAL MONEY

DECLARE @NEWSAL MONEY

SELECT @OLDSAL=SAL FROM DELETED SELECT @NEWSAL=Salary FROM INSERTED

IF @OLDSAL> @NEWSAL

BEGIN ROLLBACK

RAISERROR('NEWSAL CANNOT BELESS THANOLD SAL',15,1)

END

END

--Dropping DML Triggers

--Syntax: Drop <Trigger> <TriggerName>

--DDLTriggers:DDL triggers fire in response to a data definition languagee vent like

--create, Alter, drop etc.

--A DDL triggers is aspecial type of procedure that executes in response to a server scoped or database scoped events.

/\* Syntax:

Create Trigger<TriggerName>

on database

after <Eventtype>

As

Begin

<Statements>

End \*/

--Write a trigger which restricts dropping of a table from the database.

create trigger restdrop

on database

after drop\_table

as

begin

rollback

raiserror('Cannot drop table under this database',1,1)

end

--Write a trigger which restricts Creating of a table from the database

create trigger restcret

on database

after create\_table

as

begin

rollback

raiserror('Cannotcreatetableunderthisdatabase',1,1)

end

--Write a trigger which restricts Alter of a table from the Database

create trigger restalt

on database

after Alter\_table

as

begin

rollback

raiserror('CannotAltertableunderthisdatabase',1,1)

end

--Dropping DDL Triggers

--Syntax: Drop <Trigger> <TriggerName> on Database

-----------------------Magic Tables-----------------

--Inserted Magic Table

Create TRIGGER Trigger\_ForInsertmagic

ON emp

FOR INSERT

AS

Begin

SELECT \* FROM INSERTED

End

--Deleted Magic Table

Create TRIGGER Trigger\_Fordeletemagic

ON emp

FOR DELETE

AS

Begin

SELECT \* FROM Deleted

End

--Update the Record in Table

Create TRIGGER Trigger\_ForInsertdeletemagic

ON Employee

FOR UPDATE

AS

Begin

SELECT \* FROM INSERTED

SELECT \* FROM DELETED

End