create database Temp

use Temp

-- Trainer & Batch

create table Trainer (trainerid int, trainername varchar(20), experience

int , qualification varchar(20))

-- Be default all columns are nullbale > allows null

alter table Trainer add constraint pk primary key (trainerid)

alter table Trainer alter column trainerid int not null

alter table Trainer add constraint pk primary key (trainername)

-- Batch Table

create table batch (batchid int primary key, batchname varchar(20),

trainerid int constraint fk foreign key references trainer(trainerid) )

-- To add identity to batchid column

alter table batch add id\_new int identity

alter table batch drop column batchid

alter table batch drop constraint PK\_\_batch\_\_78CDDB4B03317E3D

-- Any rule i m creating

-- on which column????

-- foreign key ()

alter table batch drop constraint fk1

alter table batch add constraint fk1 foreign key(trainerid) references trainer(trainerid)

on delete cascade

exec sp\_rename 'batch.id\_new', 'batchid' ,'COLUMN'

select \* from batch

insert into trainer values('Ajay', 12, 'BTech')

insert into trainer values('Deepak', 20, 'MTech')

insert into trainer values('Sagar', 20, 'MTech')

insert into trainer values('Lalit', 20, 'MTech')

Select \* from trainer

insert into batch values('B001',1)

insert into batch values('B002',1)

insert into batch values('B003',2)

insert into batch values('B002',6)

insert into batch values('B003',7)

select \* from batch

Select b.batchname, t.trainername

from batch b

inner join trainer t

on b.trainerid = t.trainerid

delete from trainer where trainerid=1

Select \* from trainer

Select \* from batch

alter table batch with nocheck add constraint fk1 foreign key(trainerid) references trainer(trainerid)

on delete set null

delete from trainer where trainerid=2

Select \* from trainer

Select \* from batch

EmpId name managerid

1 ajay 3

2 deepak 8

3 jatin 8

4 deepak 6

5 manoj 7

6 faran 3

7 pradeep 1

8 jagriti 1

-- unary relation : 1

-- binary relation : 2

-- unary relationship / recursive relationship

-- self join

-- when a table is linked to itself

create table employee (id int primary key , name varchar(20), address varchar(20),

managerid int)

insert into employee values (1,'Ajay','Delhi',6),

(2,'Deepak','Calcutta',6),

(3,'Vijay','NDelhi',8),

(4,'Pradeep','Delhi',1),

(5,'Om','Delhi',8),

(6,'Deepak Kumar','Bombay',2),

(7,'Pooja','NDelhi',6),

(8,'Gagan','Delhi',6)

select \* from employee

-- Display Employee Names and their Manager Names

Select a.name As "Employee Name" , b.name As "Manager Name"

from employee a join employee b

on a.managerid = b.id

no check option

create table emp (id int primary key , name varchar(20) not null , salary int , dept varchar(20))

insert into emp values (1,'Ajay', 12000, 'HR'),

(2,'Ajay', 22000, 'Sales'),

(3,'Ajay', 17000, 'IT'),

(4,'Ajay', 2100, 'Mktng')

select \* from emp

alter table emp with nocheck add constraint ck check (salary between 20000 and 30000)

-- with check and with no check clauses can not be sued witj primary key and

-- unique constraints

-- constraints

-- primary key (cannot use )

-- foreign key

-- default

-- check

-- unique (cannot use )

-- not null

Normalization : Big tables into smaller tables

Data should not be duplicated

* Create a composite / super key (primary Key)

create table sp (pcode int

, scode int, qty int

constraint pk1 primary key(pcode, scode))

-- constraints at end of all the columns, table level constraints

insert into sp values(1,1,90)

insert into sp values(1,2,90)

insert into sp values(1,3,90)

insert into sp values(2,1,90)

insert into sp values(2,2,90)

insert into sp values(2,3,90)

pl sql

sql with procedural features of programming languages

if else

loops

function

procedures

declarations

executable statements

exception Handling

declare @name varchar(20)

set @name='Ajay'

print @name

Go

Resuablity : Use it again and again

Stored procedures: A block of SQL Statements.

Advantages of using Stored Procedures

1. Reusability
2. When we write any statement, all statements are sent to server separately

Insert …

Select ….

Update ---

Insert ----

Procedures are sent as a block to server together

Statements are compiled every time whenever we execute them

Procedures are stored in pre-compiled form

create procedure usp\_Display

AS

Begin --{

declare @name varchar(20)

set @name='Ajay'

print @name

end --}

-- execute

exec usp\_Display

create procedure GetEmployees

AS

begin

select \* from employee

end

exec GetEmployees

create procedure InsertEmployee

AS

Begin

insert into employee values(10,'Lalit','Jaipur',2)

End

exec InsertEmployee

exec GetEmployees

create procedure UpdateEmployee

AS

Begin

Update employee set managerid=1 where id=3

End

create procedure DeleteEmployee

AS

Begin

Delete employee where id=3

End

execute UpdateEmployee

execute DeleteEmployee

execute GetEmployees

void add()

{

Print 2+5

}

void add(int x, int y)

{

Print x+y

}

Input Parameters

-- input parameters : Parameters that we are passing to a procedure

alter proc InsertEmployee (@id int , @name varchar(20), @address varchar(20),

@managerid int)

AS

Begin

insert into employee values(@id, @name,@address,@managerid)

End

exec InsertEmployee 11,'Deepak','N Delhi',2

declare @id int

declare @name varchar(20)

declare @address varchar(20)

declare @managerid int

set @id=12

set @name='Ajay'

set @address='Delhi'

set @managerid=1

exec InsertEmployee @id, @name, @address, @managerid

int add(int x, int y)

{

Print x+y

Return x+y

}

drop proc InsertEmployee

create proc InsertEmployee (@id int , @name varchar(20), @address varchar(20),

@managerid int)

AS

Begin

if(exists(Select \* from employee where id=@id))

print 'Record already exists'

else

begin

insert into employee values(@id, @name,@address,@managerid)

print 'Record inserted'

end

End

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

Procedure retuning values

drop proc InsertEmployee

alter proc InsertEmployee (@id int , @name varchar(20), @address varchar(20),

@managerid int)

AS

Begin

if(exists(Select \* from employee where id=@id))

return 0

else

begin

insert into employee values(@id, @name,@address,@managerid)

return 1

end

End

* Calling part

declare @res int

declare @id int

declare @name varchar(20)

declare @address varchar(20)

declare @managerid int

set @id=14

set @name='Ajay'

set @address='Delhi'

set @managerid=1

exec @res=InsertEmployee @id, @name, @address, @managerid

if @res=0

print 'Record already exist'

else

print 'Record inserted'

update Employee

alter procedure UpdateEmployee(@id int , @name varchar(20))

AS

Begin

if(exists(Select \* from employee where id=@id))

begin

Update employee set name= @name where id=@id

return 1

end

else

return 0

End

* Calling part

declare @res int

declare @id int

declare @name varchar(20)

set @id=14

set @name='Ajay'

exec @res=UpdateEmployee @id, @name

print @res

if @res='1'

print 'Record upadted'

else

print 'Record not exist'