/-- Triggers

In SQL Server there are 3 Types of Triggers.

1. DML Triggers :- Insert, Update and Delete

2. DDL Triggers

3. Logon Triggers

--\*\* DML Triggers \*\*--

-- DML Triggers are fired automatically in response to DML events.

--DML Triggers can be again classified into 2 types:-

--1. After triggers (For trigger)

--2. Instead of triggers

\*\* After Triggers, fires after the triggering action. The Insert, Update and Delete statements,

causes an after trigger to fire after the respective statements complete execution.

\*\* Instead of triggers, fires instead of the triggering action. The Insert, Update and Delete

statements, causes an Instead of rigger to fire Instead of the respective statement execution.

\*\* Triggers can be considered as special kind of stored procedures. It will executes automatically and responds to a triggering action.

select \* from tblEmployee

create table tblEmployeeAudit

(

Id int primary key identity (1,1),

AuditData varchar (300)

)

--For Insert Action

create trigger tr\_tblEmployee\_ForInsert

on tblEmployee

for insert

as

begin

Declare @Id int

select @Id = Id from inserted

insert into tblEmployeeAudit

values ('New Employee with Id = ' +cast (@Id as varchar (5))+ ' is added at '+cast(Getdate() as varchar (20)))

End

insert into tblEmployee values

(11,'Omkar','Male',3)

select \* from tblEmployeeAudit

-- For Delete Action

create trigger tr\_tblEmployee\_ForDelete

on tblEmployee

for delete

as

begin

declare @Id int

select @Id = Id from deleted

insert into tblEmployeeAudit

values ('An existing employee with Id = '+cast (@Id as varchar(5))+' is deleted at ' +cast (Getdate() as varchar(20)))

End

delete from tblEmployee where id=11

--Duplicate table for temporary basis

Drop table #TempTable

select \* into #TempTable from tblEmployee

--For After Update Action

alter trigger tr\_tblEmployee\_ForUpdate

on tblEmployee

for update

as

begin

Declare @ID int

Declare @OldName varchar(20),@NewName varchar (20)

Declare @OldGender varchar (20), @NewGender varchar(20)

Declare @OldDeptId int, @NewDeptId int

Declare @AuditString varchar (500)

select \* into #TempTable from inserted

while (Exists (select Id from #TempTable))

Begin

set @AuditString=''

select Top 1 @Id =Id, @NewName=Name,

@NewGender=Gender, @NewDeptId=DepartmentId

from #TempTable

select @OldName=Name, @OldGender=Gender,

@OldDeptId=DepartmentId from deleted where Id=@ID

set @AuditString = 'Employee with Id = '+ cast (@ID as varchar(5))+ ' changed'

if (@OldName<> @NewName)

set @AuditString= @AuditString + ' Name from '+@OldName+' to '+@NewName

if (@OldGender<>@NewGender)

set @AuditString=@AuditString +' Gender from '+ @OldGender+' to '+@NewGender

if (@OldDeptId<>@NewDeptId)

set @AuditString=@AuditString +' Department Id from '+cast(@OldDeptId as varchar(2))+' to '+cast(@NewDeptId as varchar(2))

Insert into tblEmployeeAudit values(@AuditString)

Delete from #TempTable where Id=@ID

End

End

update tblEmployee set Name='Ram',Gender='Male' where id =2

update tblEmployee set Name= 'Urvi', Gender='Female',DepartmentId=3 where Id=2

select \* from tblEmployee

select \* from tblEmployeeAudit

\*Instead of Insert :-

-- Instead of trigger

create table tblDepartment

(

DeptId int,

DeptName varchar (30),

)

insert into tblDepartment values

(4,'Admin')

select \* from tblEmployee1

select \* from tblDepartment

create view vWEmployeeDetails

as

select ID, Name, Gender, DeptName

from tblEmployee1

join tblDepartment

on tblDepartment.DeptId=tblEmployee1.DepartmentID

select \* from vWEmployeeDetails

--Insert valid Dept Name otherwise error will thorwn

insert into vWEmployeeDetails values(8,'Bunty','Male','Mech') --ERROR

--Inserting valid Details

insert into vWEmployeeDetails values(9,'Narendra','Male','IT')

create trigger tr\_vWEmployeeDetails\_InsteadofInsert

on vWEmployeeDetails

Instead of Insert

as

begin

Declare @DeptId int

--Check if there is valid Department Id

--For the given Department Name

select @DeptId=DeptId from tblDepartment

join inserted

on inserted.DeptName=tblDepartment.DeptName

--If Department Id is Null Throw an Error

--And Stop Processing

If(@DeptId is null)

Begin

Raiserror('Invalid Department Name, statement terminated',16,1)

return

End

--Finally insert into tblEmployee table

Insert into tblEmployee1 (Id, Name, Gender,DepartmentId)

select Id, Name, Gender, @DeptId from inserted

End

\*\*Instead of Update :-

--Instead of Update Trigger

create trigger tr\_vWEmployeeDetails\_InsteadofUpdate

on vWEmployeeDetails

instead of update

as

Begin

--if EmployeeId is updated

if(update(Id))

Begin

Raiserror('ID cannot be Changed',16,1)

return

End

--If DeptName is Updated

if(update(DeptName))

Begin

Declare @DeptId int

select @DeptId =DeptId from

tblDepartment

join inserted

on tblDepartment.DeptName=inserted.DeptName

if (@DeptId is Null)

Begin

Raiserror ('Invalid Department Name',16,1)

return

End

Update tblEmployee1 set DepartmentId=@DeptId

from inserted

join tblEmployee1

on tblEmployee1.Id=inserted.Id

End

--If Gender is updated

if(update (Gender))

Begin

update tblEmployee1 set Gender=inserted.Gender

from inserted

join tblEmployee1

on tblEmployee1.Id=inserted.Id

End

--If Name is Updated

if (update(Name))

Begin

update tblEmployee1 set Name=inserted.Name

from inserted

join tblEmployee1

on tblEmployee1.Id=inserted.Id

End

End

update vWEmployeeDetails set Name='Johny' where id =1

update vWEmployeeDetails set Name='Jimmy' ,Gender='Female',DeptName='IT' where id=1

select \* from tblDepartment

select \* from tblEmployee1

select \* from vWEmployeeDetails

\*\* Instead of Delete :-

--Instead of Delete Triggers

alter trigger tr\_vWEmployeeDetails\_InsteadofDelete

on vWEmployeeDetails

instead of delete

as

begin

--Joins

Delete tblEmployee1

from tblEmployee1

join deleted

on tblEmployee1.Id=deleted.Id

--Or

--subquery

--Delete from tblEmployee1

--where Id in (select Id from deleted)

End

delete from vWEmployeeDetails where id in (1,2)