**Triggers**

5. Let’s assume that we have a table name “Book\_History” table. If a particular book is deleted from the “Book” table, an entry with same book records to “Book\_History” table must take place. Automate this process using trigger.

insert into Books values('c','study related',100,2224568223622,'9/9/2014','C',5,2);

create table Book\_History

(

iBookId int not null,

cTitle char(20) not null,

vDescription char(20),

iPrice int,

iISBN bigint not null unique,

dPublicationDate date,

cImage char(10),

iCategoryId int not null,

iPublisherId int not null

);

alter table Book\_History

add primary key (iBookId);

alter table Book\_History

add foreign key (iCategoryId) references Category(iCategoryId);

alter table Book\_History

add foreign key (iPublisherId) references Publisher(iPublisherId);

drop table Book\_History;

CREATE TRIGGER moveTo

ON Books

FOR DELETE

AS

BEGIN

DECLARE

@bookId INT ,

@title CHAR(20),

@descr CHAR(20),

@price INT,

@ISBN BIGINT,

@pubDate DATE,

@img char,

@catId INT,

@pubId INT

SELECT @bookId=iBookId,

@title=cTitle,

@descr=vDescription,

@price=iPrice,

@ISBN=iISBN,

@pubDate=dPublicationDate,

@img= cImage,

@catId=iCategoryId,

@pubId=iPublisherId

FROM DELETED

INSERT INTO Book\_History

VALUES(@bookId, @title,@descr,@price,@ISBN,@pubDate,@img, @catId, @pubId)

PRINT 'Book added to History table'

END;

select \* from Books;

select \* from Book\_History;

DELETE FROM Books

WHERE iBookId = 10;

select \* from Books;

select \* from Book\_History;

6. The “Book” table got an attribute “Price”. Let’s assume that we have a business requirement where we must ensure that the “Price” should not be less than 1. If any insert or update statement tries to make the “Price” less than 1, the SQL Server must terminate such insert or update statements. Write an appropriate trigger to implement the business requirement.

CREATE TRIGGER priceCheck

ON Books

FOR UPDATE,INSERT

AS

BEGIN

DECLARE @price int

select @price=iPrice

from inserted

IF @price< 1

BEGIN

ROLLBACK

PRINT 'Price is less than 1'

END

END;

INSERT INTO Books values('javascript','study related','0','5555112222','11/12/2000','JAVASCRIPT','5','2');

select \* from Books;

INSERT INTO Books values('javascript','study related','600','5555112222','11/12/2000','JAVASCRIPT','5','2');

select \* from Books;

7. Create a trigger on the table “Order” and add the following functionalities. When a new order is placed, it should check whether the required quantity is available in the “Book” table. If not, it should show appropriate message and the insert statement to “Order” table should be terminated. If the quantity in book table is sufficient, it should deduct the quantity ordered from the quantity in hand in the book table and update the quantity.

alter table Books

add iQuantityy int;

CREATE TRIGGER quantityCheck

ON Orders

FOR INSERT

AS

BEGIN

DECLARE @quantity int,@net int,@orderQuant int,@bid int

select @orderQuant=iQuantity,@bid=iBookId from inserted

select @quantity=iQuantityy from Books where iBookId=@bid

if (@quantity > @orderQuant)

BEGIN

update Books

set iQuantityy = (@quantity-@orderQuant)

where iBookId=@bid

END

else

BEGIN

ROLLBACK

PRINT 'not enough quantity of given book'

END

END;

select \* from Books;

select \* from Orders;

INSERT INTO Orders values('5/05/2016','2','200','pune','13');

INSERT INTO Orders values('5/10/2015','11','100','mumbai','1');