

Final Project.

Write and test seven (7) triggers for seven (7) separate tables to implement the business rules.

- At least two (2) of the triggers must be for delete.

1<sup>st</sup> Delete Trigger

The screenshot shows a SQL Server Enterprise Manager interface. The top pane displays a SQL query window titled 'SQLQuery2.sql - O...CST\23307960 (68))'. The query defines a trigger named 'Orders\_delete' on the 'orders' table. The trigger is an 'after delete' trigger. It declares a variable '@orderid' of type 'int'. It then selects the '@orderid' from the 'deleted' table and inserts it into the 'orderlogs' table with the status 'Deleted'. Finally, it deletes rows from the 'orders' table where the 'orderid' is '3' or '88'. Below the query, there are two result grids. The first grid shows the 'orders' table with columns: orderid, customer\_customerid, employee\_employeeid, orderdate, and Product\_ProductID. The second grid shows the 'orderlogs' table with columns: Orderid and status.

```
SQLQuery2.sql - O...CST\23307960 (68)) SQLQuery1.sql - O...CST\23307960 (68))

-- Create Trigger Orders_delete
-- on orders
-- after delete
-- as
-- begin
--   Declare @orderid int
--   select @orderid = deleted.orderid
--   from deleted
--   insert into orderlogs
--   values(@orderid, 'Deleted')
-- END

delete from orders where orderid = '3';
delete from orders where orderid = '88';

select * from orders;
select * From orderlogs;
```

100 %

Results Messages

|   | orderid | customer_customerid | employee_employeeid | orderdate  | Product_ProductID |
|---|---------|---------------------|---------------------|------------|-------------------|
| 1 | 50      | 1                   | 20                  | 2020-09-02 | 30                |
| 2 | 51      | 2                   | 21                  | 2020-01-02 | 31                |
| 3 | 52      | 3                   | 22                  | 2020-01-06 | 32                |
| 4 | 53      | 4                   | 23                  | 2020-07-06 | 33                |
| 5 | 54      | 5                   | 24                  | 2020-07-06 | 34                |
| 6 | 99      | 777                 | 889                 | NULL       | NULL              |

|   | Orderid | status  |
|---|---------|---------|
| 1 | 3       | Deleted |
| 2 | 88      | ADDED   |
| 3 | 88      | ADDED   |
| 4 | 88      | Deleted |

## 2<sup>nd</sup> Delete Trigger.

```
SQLQuery1.sql - O...CST\23307960 (68))* ➦ ✕  
  
Create Trigger Employeeinfotriggger  
on  
employeeinformation  
after  
delete  
as  
Begin  
Declare @Id int  
select @Id = deleted.Id  
from DELETED  
  
Insert into employeeinformationlog  
values(@Id, 'DELETED')  
END  
  
Delete from employeeinformation where Id = '2'  
Delete from employeeinformation where Id = '3'  
  
Select * from employeeinformation
```

100 %

Messages

(1 row affected)  
(1 row affected)  
(1 row affected)  
(1 row affected)

Completion time: 2021-05-15T08:05:22.8053341-04:00

```
SQLQuery1.sql - O...CST\23307960 (68))* ➦ ✕  
  
values(@Id, 'DELETED')  
END  
  
Delete from employeeinformation where Id = '2'  
Delete from employeeinformation where Id = '3'  
  
Select * from employeeinformation  
Select * from employeeinformationlog
```

100 %

Results Messages

|   | ID | Name  | location        |
|---|----|-------|-----------------|
| 1 | 1  | Max   | Junction street |
| 2 | 4  | james | Southern blvd   |
| 3 | 5  | kylee | Junction blvd   |
| 4 | 6  | nancy | canal street    |

|   | id | status  |
|---|----|---------|
| 1 | 2  | DELETED |
| 2 | 3  | DELETED |

- At least two (2) of the triggers must be for insert.

1<sup>st</sup> Insert

```
SQLQuery1.sql - O...CST\23307960 (69))*
Create Trigger CustomeInsert
on
Customer
after Insert
AS
Begin
Declare @customerid int
Select @customerid = INSERTED.customerid
from INSERTED

Print ' New value has been added to the Table Customer'
END

Insert into customer(customerid, name, address, phone)
Values('99','abigel','bronxave','83939939')
```

100 %

Messages

New value has been added to the Table Customer

(1 row affected)

Completion time: 2021-05-12T20:55:49.3730830-04:00

100 %

|   | customerid | name       | address         | phone     |
|---|------------|------------|-----------------|-----------|
| 1 | 1          | jasonblain | woodhaveblvd    | 637898298 |
| 2 | 3          | sarahlue   | woodsideave     | 47899389  |
| 3 | 5          | george     | grandcentralave | 898094489 |
| 4 | 9          | fung       | florida ave     | 84984939  |
| 5 | 98         | abigel     | bronxave        | 83939939  |
| 6 | 99         | abigel     | bronxave        | 83939939  |

## 2nd Insert

SQLQuery2.sql - O...CST\23307960 (68))    SQLQuery1.sql - O...CST\23307960 (69))\*    X

```
CREATE TRIGGER OrderInsert
ON
  Orders
AFTER INSERT
AS
BEGIN
  DECLARE @orderid int
  SELECT @orderid = INSERTED.orderid
  FROM INSERTED
  INSERT INTO orderlogs
  VALUES(@orderid, 'ADDED')
END

insert into orders (orderid,customer_customerid,employee_employeeid) values('88','544','443')
insert into orders (orderid,customer_customerid,employee_employeeid) values('03','494','889')

select * from orderlogs
```

100 %

Results    Messages

|   | orderid | status |
|---|---------|--------|
| 1 | 99      | ADDED  |
| 2 | 88      | ADDED  |
| 3 | 88      | ADDED  |
| 4 | 3       | ADDED  |

- At least two (2) of the triggers must be for update.

## 1<sup>st</sup> Update

S23307960

Execute

SQLQuery1.sql - O...CST\23307960 (68))\*

```

Create Trigger
order_update
on Orders
after
update
AS
Begin
Declare @orderid int
Declare @Action varchar(50)
Select @orderid = Inserted.orderid
from Inserted

IF Update(orderid)
Set @Action = 'ID Updated'

Insert into orderlogs
values(@orderid, @Action)
END
  
```

100 %

Messages

Commands completed successfully.

Completion time: 2021-05-14T12:25:36.6409549-04:00

100 %

SQLQuery1.sql - O...CST\23307960 (68))\*

```

Update orders set orderid = '100' where customer_customerid = '5';

select * From orders;
select * from orderlogs;
  
```

100 %

Results

|   | orderid | customer_customerid | employee_employeeid | orderdate  | Product_ProductID |
|---|---------|---------------------|---------------------|------------|-------------------|
| 1 | 91      | 1                   | 20                  | 2020-09-02 | 30                |
| 2 | 80      | 2                   | 21                  | 2020-01-02 | 31                |
| 3 | 80      | 3                   | 22                  | 2020-01-06 | 32                |
| 4 | 80      | 4                   | 23                  | 2020-07-06 | 33                |
| 5 | 100     | 5                   | 24                  | 2020-07-06 | 34                |
| 6 | 80      | 777                 | 889                 | NULL       | NULL              |

Messages

|   | Orderid | status     |
|---|---------|------------|
| 1 | 3       | Deleted    |
| 2 | 88      | ADDED      |
| 3 | 88      | ADDED      |
| 4 | 88      | Deleted    |
| 5 | 80      | ID Updated |
| 6 | 80      | ID Updated |
| 7 | 91      | ID Updated |
| 8 | 100     | ID Updated |

2nd Update

SQLQuery1.sql - O...CST\23307960 (68))\*

```
CREATE TRIGGER Store_trigger
on
store
after
Update
as
Begin
Declare @storeid int
Declare @Action varchar(50)
select @storeid = Inserted.storeid
from Inserted
IF Update(storeid)
set @Action = 'ID updated'
IF Update(location)
set @Action = 'Location updated'
IF Update(Name)
Set @Action = 'Name updated'
Insert into StoreLogs
Values(@storeid,@Action)

END
```

100 %

Messages

Commands completed successfully.

Completion time: 2021-05-14T13:57:24.7358945-04:00

```
Update store set location = 'richmond' where Name = 'iut';
```

```
Update store set Name = 'jerry' where location = 'rowaway';
```

```
Select * from store;
```

```
select * from StoreLogs;
```

100 %

Results

Messages

|   | storeid | location | Name  |
|---|---------|----------|-------|
| 1 | 10      | nasvill  | xyz   |
| 2 | 11      | highroad | abc   |
| 3 | 12      | highland | xzy   |
| 4 | 13      | rowaway  | jerry |
| 5 | 14      | richmond | iut   |
| 6 | 15      | hilltop  | cvb   |

|   | storeid | status           |
|---|---------|------------------|
| 1 | 14      | Location updated |
| 2 | 13      | Name updated     |
| 3 | 14      | Location updated |
| 4 | 13      | Name updated     |

- At least one (1) of the triggers must be for delete/insert/update.

SQLQuery1.sql - O...CST\23307960 (68))\*

```
Create Trigger newTrigger
ON
Orders
After insert, update, delete
as
Begin
    Select * from inserted
    Select * from Deleted
End

Update orders
set orderid = '88'
where customer_customerid = '67';
```

100 %

Results Messages

|   | orderid | customer_customerid | employee_employeeid | orderdate  | Product_ProductID |
|---|---------|---------------------|---------------------|------------|-------------------|
| 1 | 88      | 67                  | 66                  | 2012-02-01 | NULL              |

|   | orderid | customer_customerid | employee_employeeid | orderdate  | Product_ProductID |
|---|---------|---------------------|---------------------|------------|-------------------|
| 1 | 91      | 67                  | 66                  | 2012-02-01 | NULL              |



Data security: (at least three fields) client's sensitive data/information such as email/phone number/ credit cards etc. are expected to be secure and mask when displaying.

Execute

SQLQuery1.sql - O...CST\23307960 (70))\*

```
-- Create Table CustomerPaymentInformation(  
    CustomerID int PRIMARY KEY,  
    FirstName Varchar(20) Masked With (FUNCTION = 'partial(1,"XXXX",0)') NULL,  
    LastName Varchar(20),  
    PaymentNumber varchar(12) MASKED WITH (FUNCTION = 'default()')NULL,  
    PaymentType varchar(12) Masked with(Function = 'partial(1,"XXXXX",0)')Null);  
  
-- Insert CustomerPaymentInformation(CustomerID,FirstName,LastName,PaymentNumber,PaymentType)  
Values('11','JOSEPH','COREY','986754675678','CREDIT'),  
('12','JUDYE','VANCE','378787879878','DEBIT'),  
('13','ANDRE','STEPHEN','989809765436','DEBIT'),  
('45','JOHNY','BARBRA','839876545673','CREDIT');  
  
Create user TestCustomer WITHOUT LOGIN;  
GRANT SELECT ON CustomerPaymentInformation TO TestCustomer;  
EXECUTE AS user = 'TestCustomer';  
  
select * from CustomerPaymentInformation
```

100 %

Results Messages

|   | CustomerID | FirstName | LastName | PaymentNumber | PaymentType |
|---|------------|-----------|----------|---------------|-------------|
| 1 | 11         | JXXXX     | COREY    | xxxx          | CXXXXX      |
| 2 | 12         | JXXXX     | VANCE    | xxxx          | DXXXXX      |
| 3 | 13         | AXXXX     | STEPHEN  | xxxx          | DXXXXX      |
| 4 | 45         | JXXXX     | BARBRA   | xxxx          | CXXXXX      |

Data retention policy: data is kept indefinitely. *There are numerous ways to resolved this issue, find one and implement.*

--> In Order to resolve the Problem of Data Retention Policy we need to maintain the backups of our data because the data keeps on piling up. We need to auto clean up and delete backups older than a certain period like a Month. With the Maintenance plan we can deletes backups data, and we can specify the amount of time the data should be deleted in.

Back Up Database - S23307960

Select a page

- General
- Media Options
- Backup Options

Script ? Help

Backup set

Name: Database Backup

Description:

Backup set will expire:

☒ After: 90 days

☐ On: 5/15/2021

Compression

Set backup compression: Use the default server setting

Encryption

☐ Encrypt backup

Algorithm: AES 128

Certificate or Asymmetric key:

Encryption is available only when Back up to a new media set is selected in Media Options.

Connection

Server: ORDB1\cst4714\_Prod

Connection: CST\_23307960

[View connection properties](#)

Progress

Ready

OK Cancel

- ✎ Database
- ✎ Database Audit Specification
- ✎ Database DDL Trigger
- ✎ Database Maintenance
- ✎ Database Options
- ✎ Database Performance
- ✎ Database Replica State
- ✎ Database Role
- ✎ Database Security
- ✎ Default
- ✎ Endpoint
- ✎ Filegroup