

/\*

PROG 140      Module 7 Assignment #8  
POINTS 50      DUE DATE : Consult module

**Angelique Refahiyat**

**06/02/2018**

Develop a SQL statement for each task listed. This exercise uses the Northwind database.

Please type your SQL statement under each task below. Ensure your statement works by testing it prior to turning it in

When writing answers for your select statements please keep in mind that there is more than one way to write a query.

Please do your best to write a query that not only returns the information the questions asks for but to the best of your ability,

in the best way possible to suit their needs. For example, consider the best way to sort, to name columns, etc. These are things you must think about on the job!

Turn In:

For this exercise you will submit one WORD documents (instead of a .sql file) in which you have copied and pasted

your entire work from your .sql file including all assignment questions and your SQL queries.

The document should contain your name at the top, assignment title, the date and any difficulties encountered.

Submit your file to the instructor using the Canvas Assignment tool.

\*/

-- Tasks:

/\*

1.

10 pts

Write a simple trigger that only prints a message stating that a row in the customers table has been updated and that fires only upon the execution of an update statement on the customers table.

Include a statement that will fire your trigger and also a statement to drop your trigger.

```

*/
-- write your SQL statements here:

use Northwind
go
CREATE TRIGGER UpdateAlert on [dbo].[Customers]
FOR UPDATE
AS
DECLARE @AlertNotice VARCHAR(50)

IF UPDATE([CompanyName])
BEGIN
SET @AlertNotice = 'Updated Company Name'
END
IF UPDATE([ContactTitle])
BEGIN
SET @AlertNotice = 'Updated Contact title'
END
--test
--SELECT * from[dbo].[Customers]
--where [CustomerID] = 'VICTE'
UPDATE [dbo].[Customers]
SET [CompanyName] ='stockingggg victualities'
WHERE [CustomerID] = 'VICTE'
SELECT [CustomerID],[CompanyName],[ContactTitle], @AlertNotice as notification FROM INSERTED
DROP TRIGGER UpdateAlert
GO
/*

```

2.

10 pts

Create a trigger that will run instead of the Update statement on the Suppliers table. It will fire when an Update statement is executed against the Suppliers table and will an error message saying: "Updating information in this table is not allowed"

Include a statement that will fire your trigger and also a statement to drop your trigger.

(Hints: There is such a thing as an "instead of trigger". See the demo file for an example! You can use RaisError for the error message.)

```

*/
-- write your SQL statements here:
USE Northwind
GO
CREATE TRIGGER UpdateProhibition on [dbo].[Suppliers]
INSTEAD OF UPDATE, DELETE
AS
BEGIN
RAISERROR ( 'Updating information in this table is not allowed',18,0)
END

UPDATE [dbo].[Suppliers]
SET [CompanyName]='lyngbisylt'
WHERE [SupplierID]= 21
SELECT [SupplierID],[CompanyName],[ContactTitle]
FROM INSERTED

```

/\*

3.

15 pts

Northwind management is concerned that someone is updating and deleting products without following proper procedures. You are asked to provide a log of all future UPDATES and DELETES performed on the products table. Unfortunately

the products table does not store this information. You decide to create a log table and a trigger that will load this log table with the productid,

the date and the user of the deleted or updated product.

\*\*\*\*\*

Note: Here's how to find the user!! There is an example in the demo file in the TRIGGER trgCheckSalaryCap however it is in the part I did not present on the video since

I had to end the video and thought that 4 videos would be too much for everyone. Its actually pretty easy.

There is a user function that retrieves the current user.

It's called "user". The user can be obtained like this:

```

Declare @User varchar(20)
Select @User = System_User

```

\*\*\*\*\*

Include a statement that will fire your trigger and also a statement to drop your trigger.

```
*/
-- write your SQL statements here:
DECLARE @sys_usr varchar(100);
SET @sys_usr = SYSTEM_USER;
SELECT 'The current system user is: ' + @sys_usr;
GO
ALTER TABLE [dbo].[Order Details] DROP CONSTRAINT FK_Order_Details_Products
GO
CREATE TABLE login_Details
(
    [ProductID] int NOT NULL,
        [userName] varchar(100) NOT NULL,
        [ProductName] [nvarchar](40) NOT NULL,
        dateChanged date
    )
    --Alter TABLE login_Details ADD FOREIGN KEY ([ProductID]) REFERENCES
[dbo].[Products]([ProductID])
GO
insert into [dbo].[Products]([ProductName])
values ('XYZCorpId');

GO
CREATE TRIGGER trgAfter_delete_track
ON [dbo].[Products]
After DELETE,Update
AS

--DECLARE @sys_usr char(30)
--SET @sys_usr = SYSTEM_USER
--DECLARE @OldProduct nvarchar(40)
--DECLARE @User char(30)
--DECLARE @prID int
BEGIN
set nocount on
```

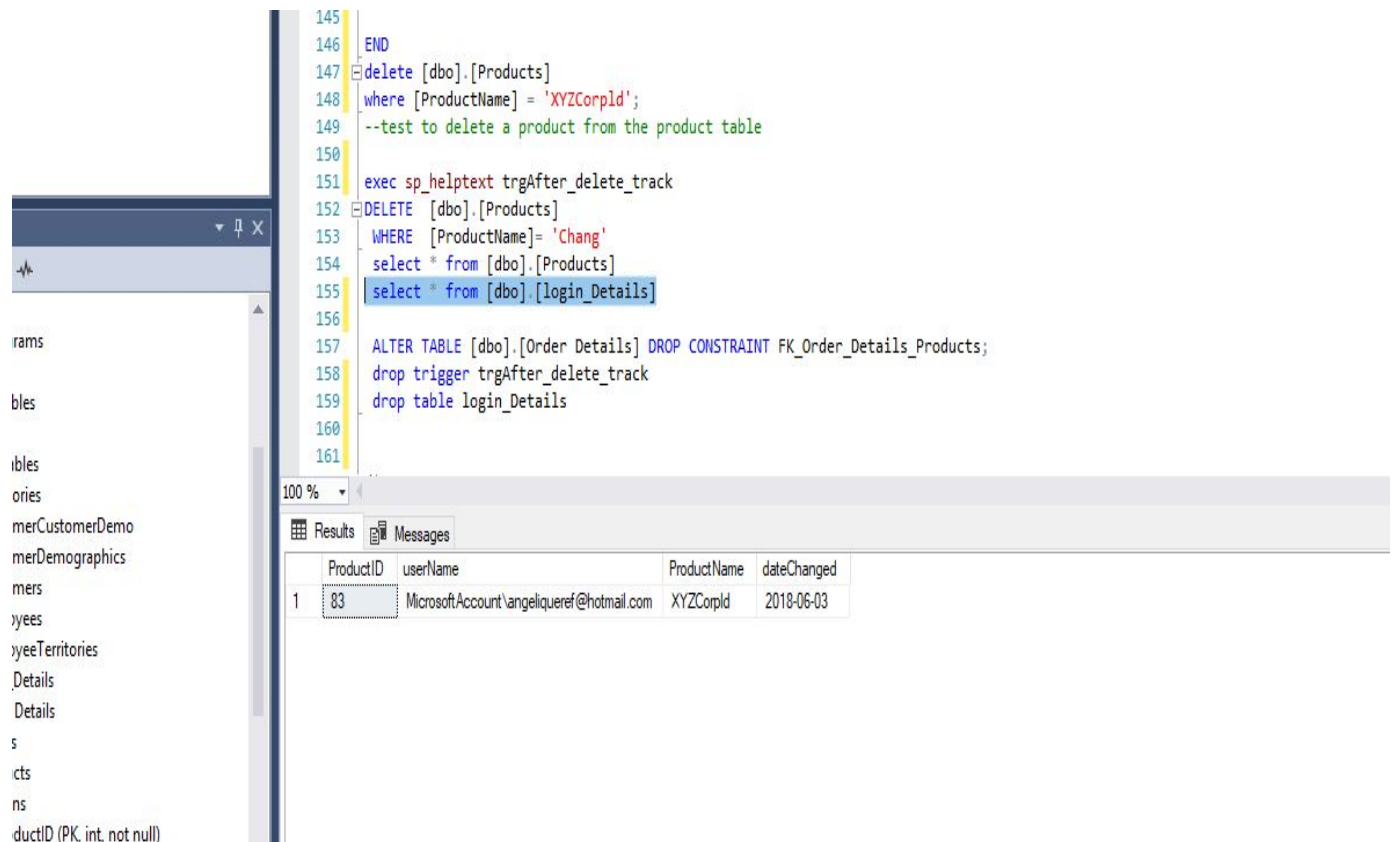
```

INSERT INTO [dbo].[login_Details]
SELECT ProductID,SYSTEM_USER,[ProductName],CURRENT_TIMESTAMP from DELETED d
set nocount off
END
DELETE [dbo].[Products]
WHERE [ProductName] = 'XYZCorpld';
--test to delete a product from the product table

EXEC sp_helptext trgAfter_delete_track
DELETE [dbo].[Products]
WHERE [ProductName]='Chang'
SELECT * FROM [dbo].[Products]
SELECT * FROM [dbo].[login_Details]

DROP TRIGGER trgAfter_delete_track
DROP TABLE login_Details

```



The screenshot shows the SQL Server Enterprise Manager interface. On the left is the 'Server Enterprise' tree view. The central pane displays a SQL script with line numbers 145 to 161. The script includes a DELETE statement for the 'Products' table, a trigger help text execution, another DELETE statement, and a SELECT statement. The bottom pane shows the 'Results' tab with a single row of data from the 'login\_Details' table.

**SQL Script:**

```

145 END
146
147 delete [dbo].[Products]
148 where [ProductName] = 'XYZCorpld';
149 --test to delete a product from the product table
150
151 exec sp_helptext trgAfter_delete_track
152 DELETE [dbo].[Products]
153 WHERE [ProductName]='Chang'
154 select * from [dbo].[Products]
155 select * from [dbo].[login_Details]
156
157 ALTER TABLE [dbo].[Order Details] DROP CONSTRAINT FK_Order_Details_Products;
158 drop trigger trgAfter_delete_track
159 drop table login_Details
160
161

```

**Results:**

ProductID	userName	ProductName	dateChanged
83	MicrosoftAccount\angeliqueref@hotmail.com	XYZCorpld	2018-06-03

/\*

4.

15 pts

In our demo file Cursors.sql there is a stored procedure called "Rebuild\_All\_Indexes" near the end of that file.

Use

this stored procedure as a model to create a stored procedure called "uspListAllUserTables" that will print the names of all tables in the sys.Objects table that are user tables (ie., type = 'U'). (Hint: you will not need to return a table object - just using the print statement within the sproc will print out the tables)

Include a statement to execute your stored procedure and be sure to test it to ensure it works.

\*/

-- write your SQL statements here:

Go

Create PROCEDURE uspPrintTables

AS

DECLARE Alltables cursor

FOR

SELECT NAME FROM sys.Objects WHERE type = 'U'

OPEN Alltables

DECLARE @table\_name SYSNAME

```
FETCH next FROM Alltables INTO @table_name
WHILE @@fetch_status = 0
BEGIN
    PRINT @table_name
    FETCH next FROM Alltables INTO @table_name
END
CLOSE Alltables
DEALLOCATE Alltables
EXEC uspPrintTables

DROP PROCEDURE uspPrintTables
```