-- Assignment 3 Question 1

--In this lab you will use MyGuitarShop database to create a stored procedure, two functions, and a trigger.

--You will prepare a Word document that contains your Transact-SQL statements and screenshots that prove that

--the programming modules work as intended.

--Requirements

-- Create a stored procedure named spInsertProduct that inserts a row into the Products table.

--This stored procedure should accept five parameters. One parameter for each of these columns: CategoryID,

--ProductCode, ProductName, ListPrice, and DiscountPercent. (Note that ProductID is an identity column.)

--This stored procedure should set the Description column to an empty string, and it should set the DateAdded column

--to the current date. If the value for the ListPrice column is a negative number, the stored procedure should raise

--an error that indicates that this column doesn’t accept negative numbers. Similarly, the procedure should raise an

--error if the value for the DiscountPercent column is a negative number. Code at least two EXEC statements that test

--this procedure, one with correct parameters, and one that causes an error. Run SELECT query that shows the inserted row(s).

--In the Word document include all your statements and screenshots from the execution.

select \* from products

select \* from Categories

USE MyGuitarShop;

GO

IF OBJECT\_ID('spInsertProduct') IS NOT NULL -- this is to drop the procedure if it has been created before

DROP PROC spInsertProduct;

GO

-- declaring the data type to be inserted into the products table

CREATE PROC spInsertProduct

@CategoryID int, @ProductCode varchar(10), --ProductID is auto generated we don't provide value,(identity column)

@ProductName varchar(50), @ListPrice money,

@DiscountPercent int

AS

-- Introducing Data validation statements to ensure that negative number is

-- not inserted into ListPrice and DiscountPercent Colunms respectively

IF NOT EXISTS ( SELECT \* FROM Categories WHERE CategoryID = @CategoryID)

THROW 50001, 'Invalid CategoryID.' , 1;

IF @ListPrice IS NULL OR @ListPrice <= 0

THROW 50001, 'This column does not accept a negative number.', 1;

IF @DiscountPercent < 0

THROW 50001, 'This column does not accept a negative number.', 1;

-- stating the items to be inserted into products table and their conditions

INSERT Products

VALUES (@CategoryID, @ProductCode, @ProductName, '',

@Listprice, @DiscountPercent, GETDATE());

EXEC spInsertProduct

3, 'Chike', 'Michael Smith', 10, 20; -- This was inserted into products as seen below as the ProductID = 22

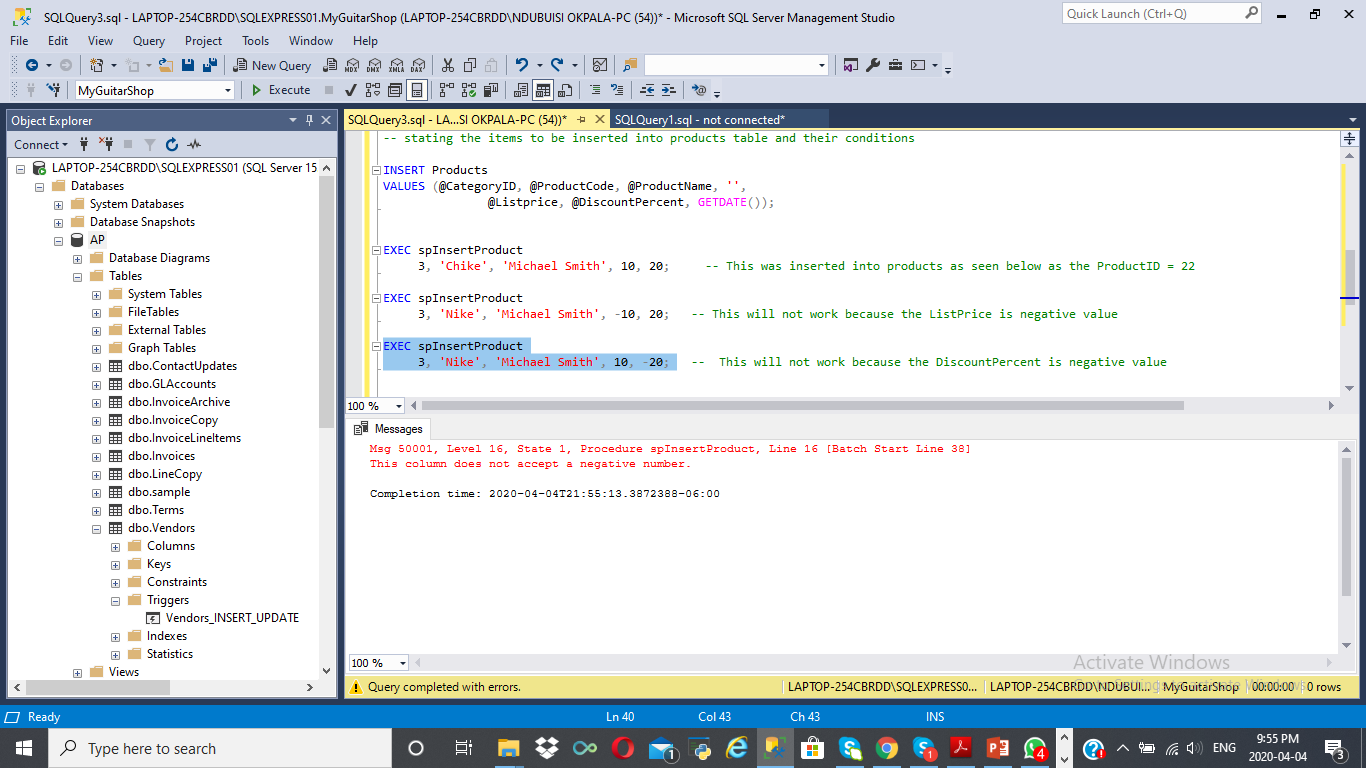
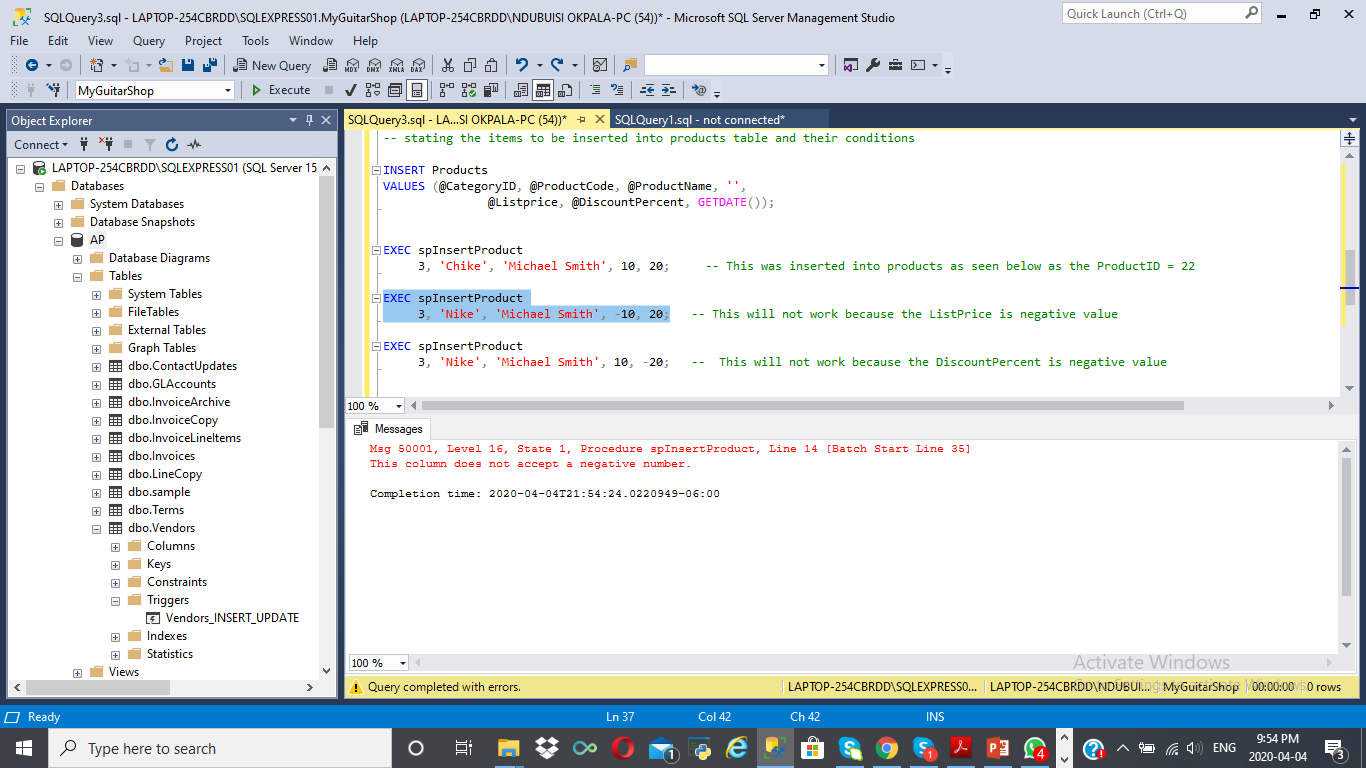
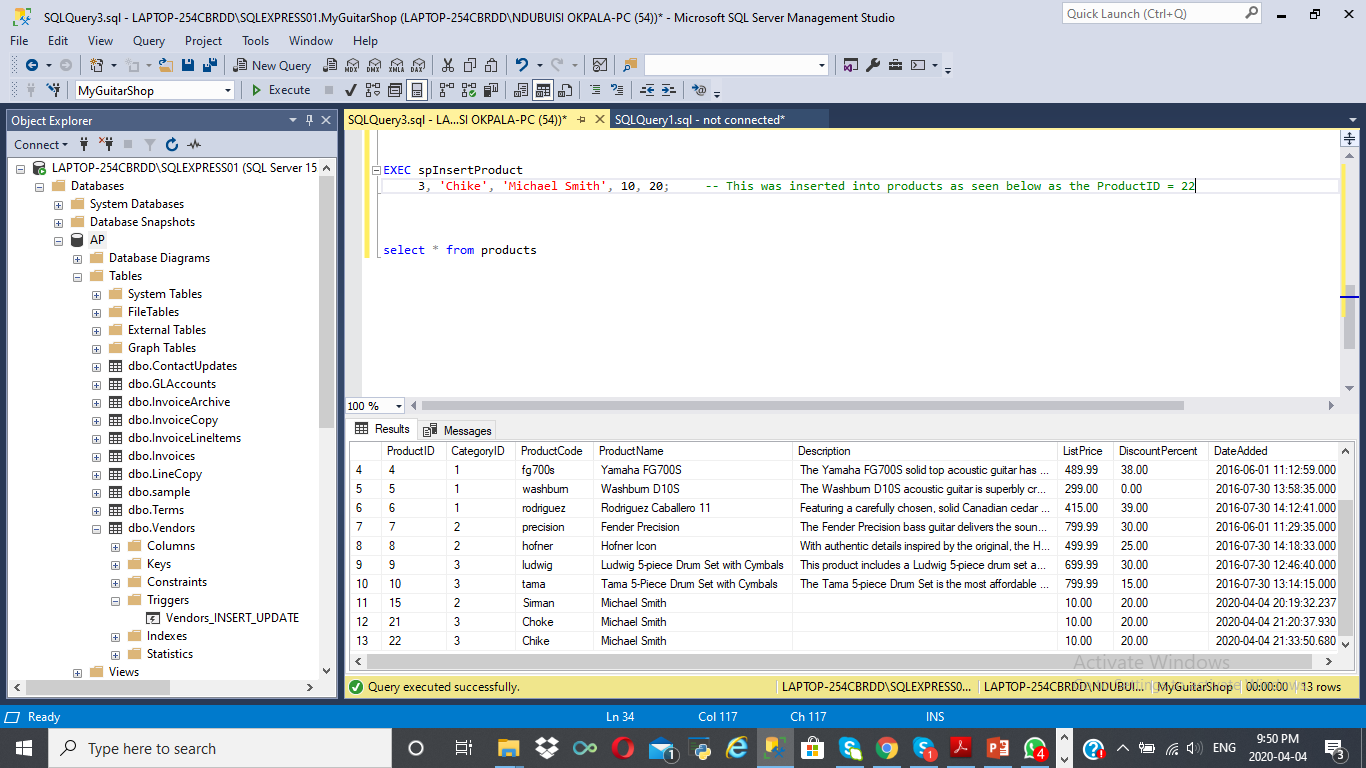
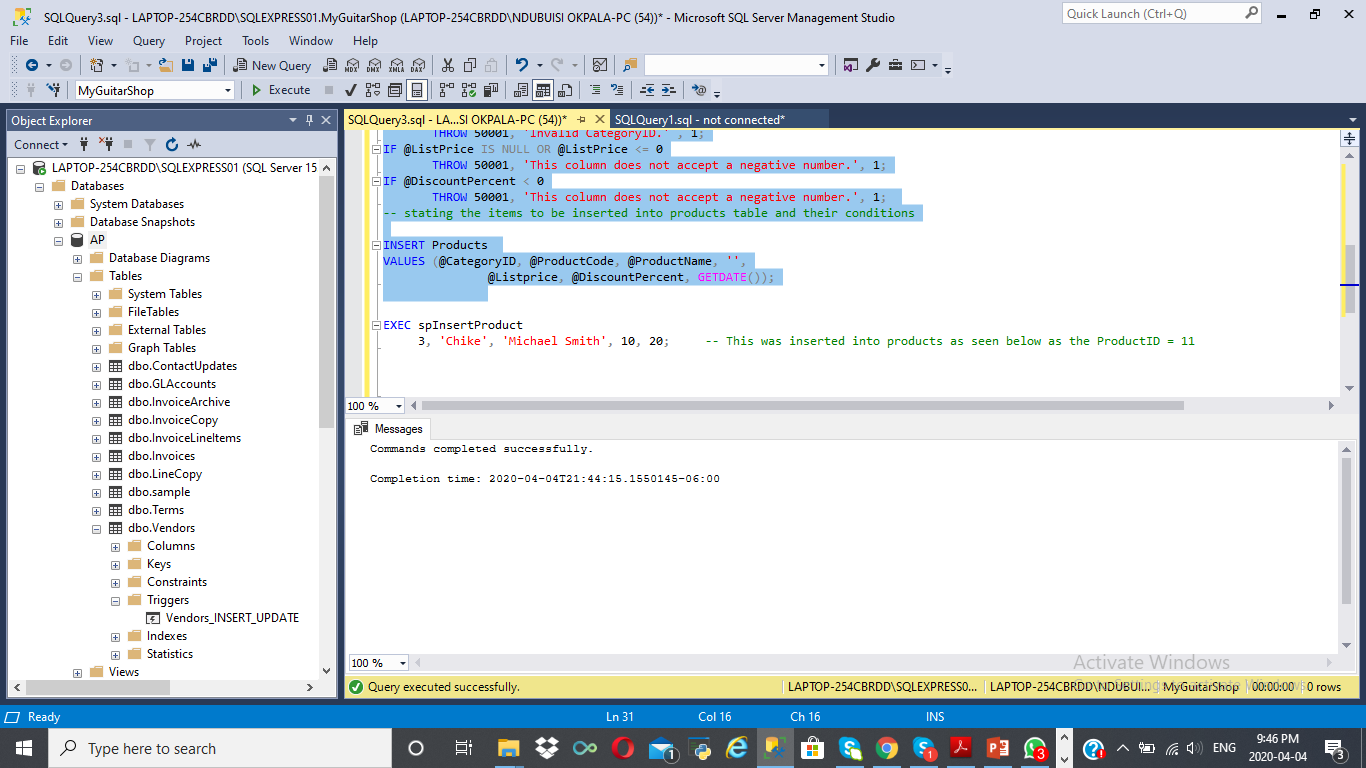
EXEC spInsertProduct

3, 'Nike', 'Michael Smith', -10, 20; -- This will not work because the ListPrice is negative value

EXEC spInsertProduct

3, 'Nike', 'Michael Smith', 10, -20; -- This will not work because the DiscountPercent is negative value

select \* from products



--Assignment 3 Question 2

--Create two functions:

--a. The first function is named fnDiscountPrice that calculates the discount price of an item in the OrderItems

--table (discount amount subtracted from item price). To do that, this function should accept one parameter for the item ID,

--and it should return the value of the discount price for that item.

--b. The second function is named fnItemTotal and calculates the total amount of an item in the OrderItems

--table (discount price multiplied by quantity). To do that, this function should accept one parameter for the item ID,

--it should use the fnDiscountPrice function that you just created, and it should return the value of the total for that item.

--Call both functions, and include in the Word document all Transact-SQL statements and execution screenshots

select \* from OrderItems;

IF OBJECT\_ID('fnDiscountPrice') IS NOT NULL

DROP FUNCTION fnDiscountPrice;

GO

CREATE FUNCTION fnDiscountPrice

(@ItemID int)

RETURNS money

BEGIN

RETURN (SELECT SUM(ItemPrice - DiscountAmount)

FROM OrderItems

WHERE @ItemID = ItemID);

END;

--- to validate the function 'fnDiscountPrice' created above, we use the below query statement

print dbo.fnDiscountPrice

print 'Discount Price :' + format(dbo.fnDiscountPrice(2), 'c'); -- 303.79

IF OBJECT\_ID('fnItemTotal') IS NOT NULL

DROP FUNCTION fnItemTotal;

GO

CREATE FUNCTION fnItemTotal

(@ItemID int)

RETURNS money

BEGIN

RETURN (SELECT sum((ItemPrice - dbo.fnDiscountPrice(ItemID)) \* Quantity)

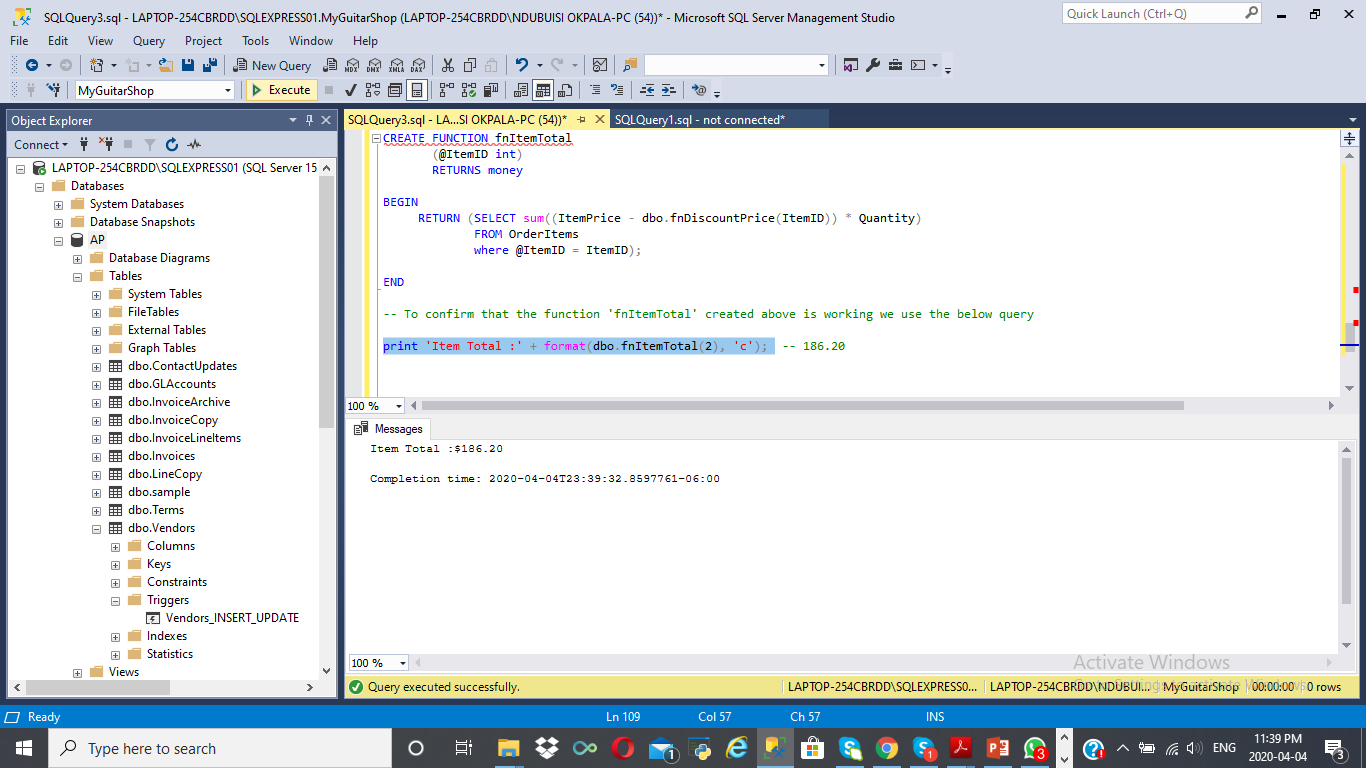
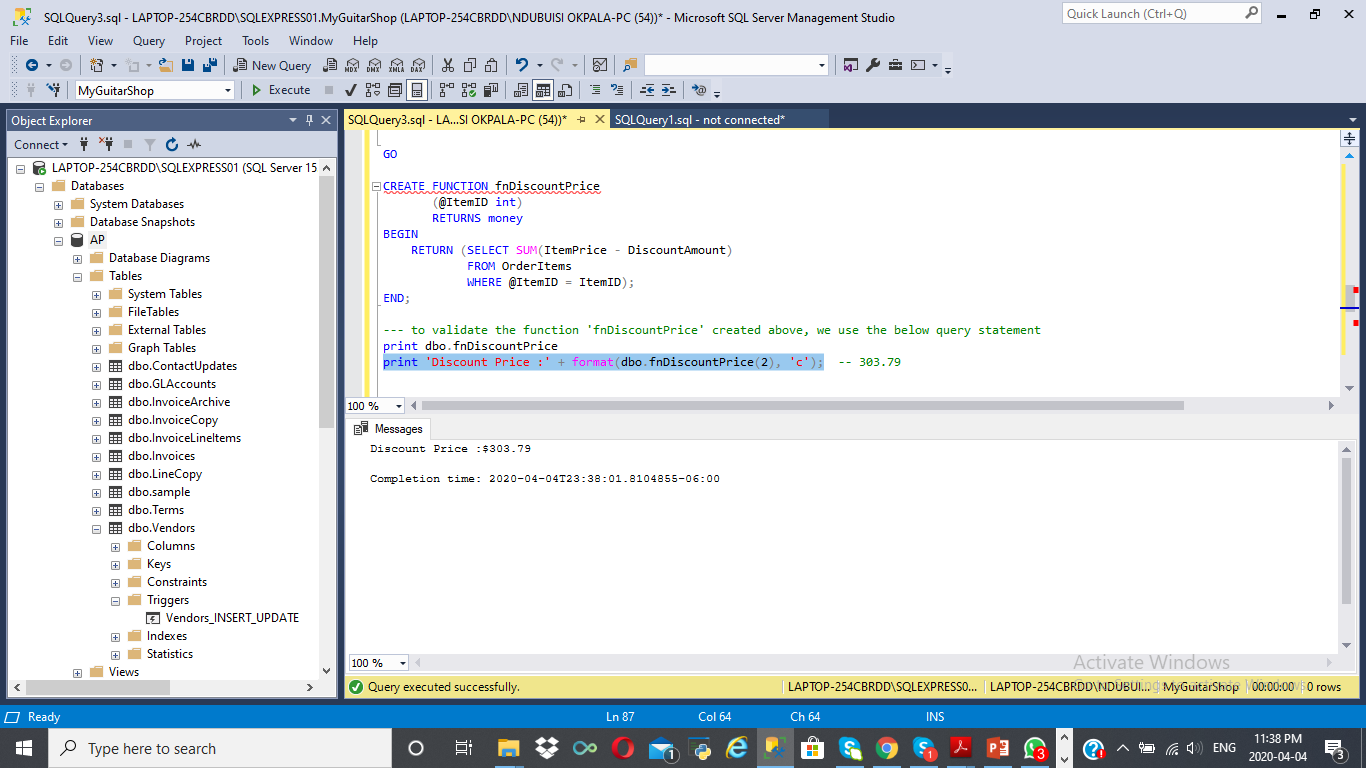
FROM OrderItems

where @ItemID = ItemID);

END

-- To confirm that the function 'fnItemTotal' created above is working we use the below query

print 'Item Total :' + format(dbo.fnItemTotal(2), 'c'); -- 186.20



--ASSIGNMENT 3 QUESTION 3

--Create a trigger named Products\_UPDATE that checks the new value for the DiscountPercent column of the Products table.

--This trigger should raise an appropriate error if the discount percent is greater than 100 or less than 0.

--If the new discount percent is between 0 and 1, this trigger should modify the new discount percent by multiplying it by 100.

--That way, a discount percent of .2 becomes 20. Test this trigger with an appropriate UPDATE statement.

-- this is use to confirm if the trigger has been created before, if exits, we need to drop it before creating a new one

IF OBJECT\_ID ('Products\_UPDATE') IS NOT NULL

DROP TRIGGER Products\_UPDATE;

GO

-- This is to create the trigger named 'Product\_UPDATE

CREATE TRIGGER Products\_UPDATE

ON Products

AFTER UPDATE

AS

-- we need to declare the colunm we want to the trigger to affect and the select statement

DECLARE

@DiscountPercent money;

SELECT @DiscountPercent = DiscountPercent from inserted;

-- we declear the trigger conditions with IF statements

IF @DiscountPercent > 100

THROW 50001, 'This column accepts values between 1 -100.', 1;

IF @DiscountPercent < 0

THROW 50001, 'This column accepts values between 1 -100.', 1;

IF @DiscountPercent < 1

BEGIN

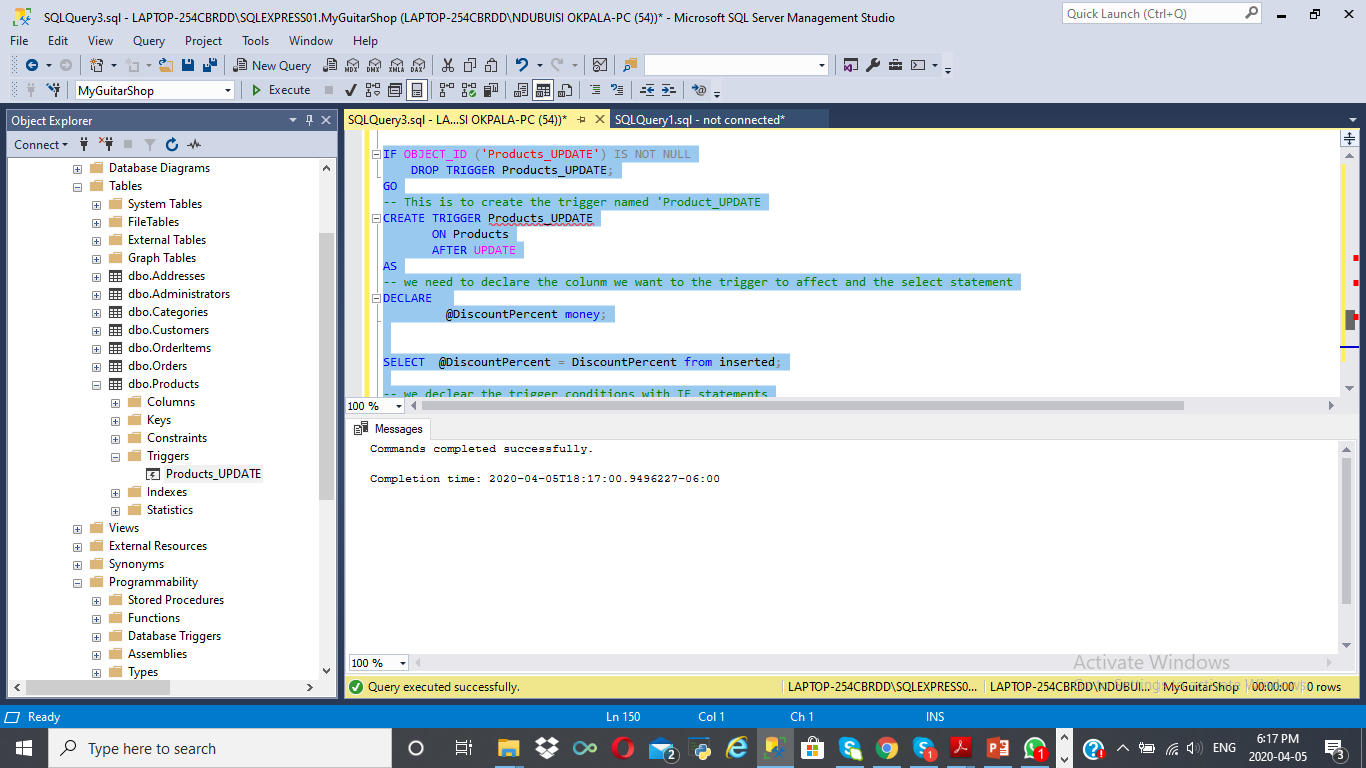
UPDATE Products

SET DiscountPercent = (SELECT DiscountPercent

FROM Inserted) \* 100

WHERE ProductID IN(SELECT ProductID FROM Inserted);

END



--this is to view if the update query below was inserted succefully

select \* from products

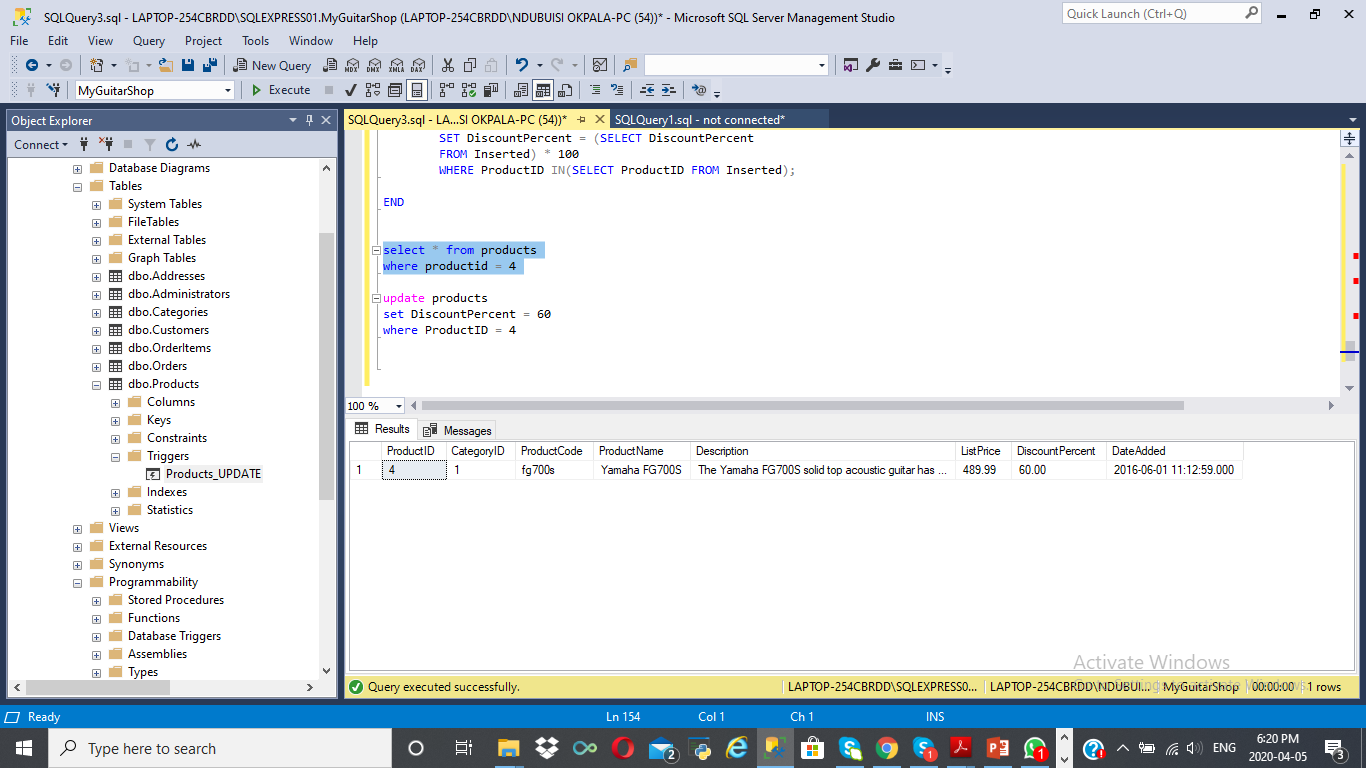
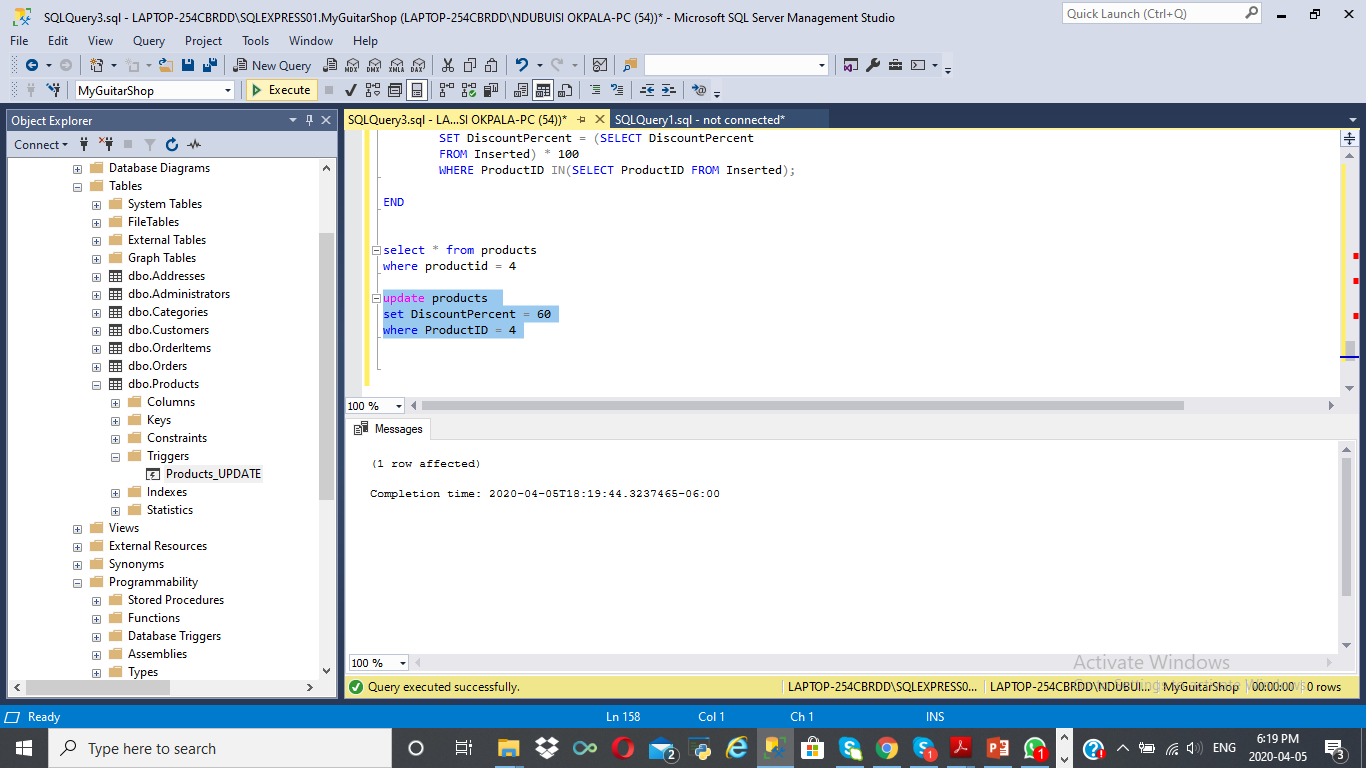
where productid = 4

-- this is used to validate one of the trigger conditions above

update products

set DiscountPercent = 60

where ProductID = 4



--this is to view if the update query below wasnt inserted succefully becuase it didnt meet the condition of the trigger

select \* from products

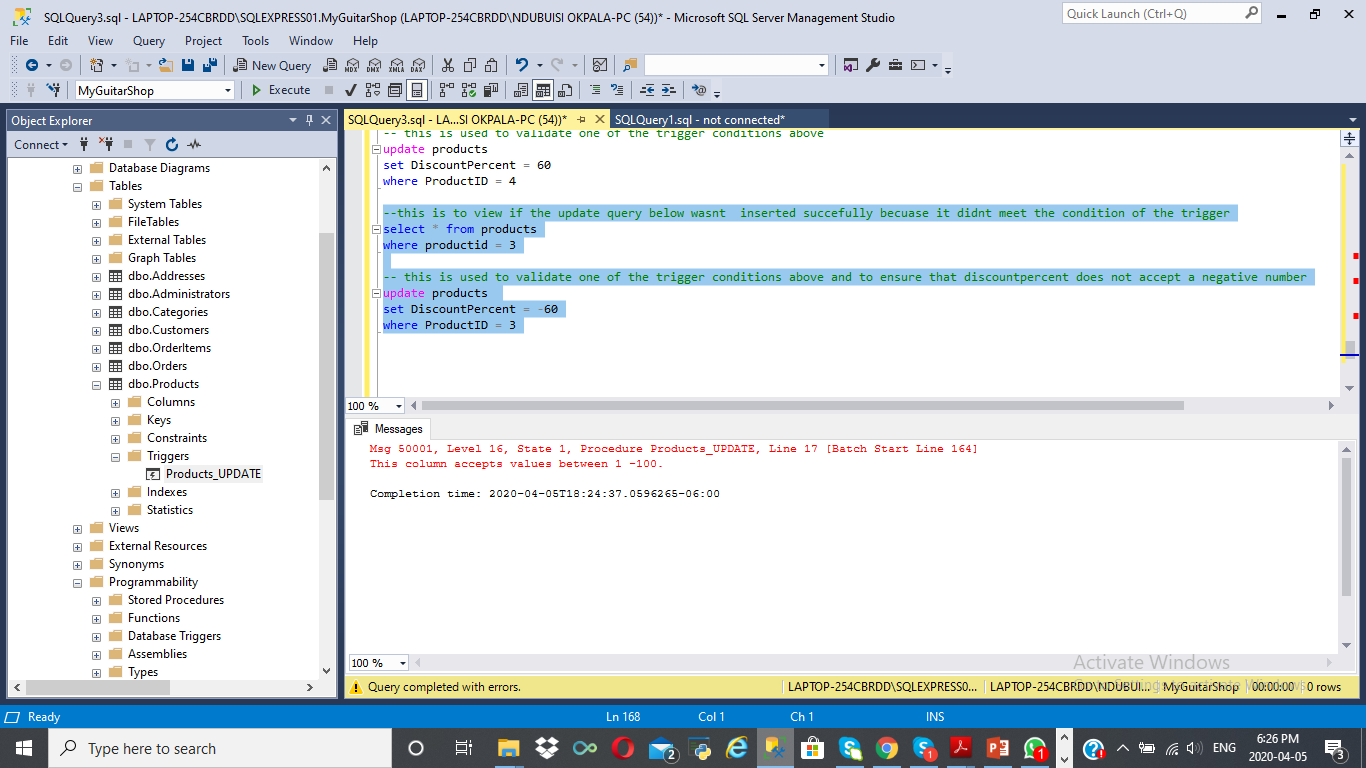
where productid = 3

-- this is used to validate one of the trigger conditions above and to ensure that discountpercent does not accept a negative number

update products

set DiscountPercent = -60

where ProductID = 3



-this is to view if the update query below wasnt inserted succefully becuase it didnt meet the condition of the trigger

select \* from products

where productid = 2

-- this is used to validate one of the trigger conditions above and to ensure that discountpercent does not accept a number above 100

update products

set DiscountPercent = 101

where ProductID = 2

