1) create stored procedure spInsertProduct

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

DROP PROC spInsertProduct;

GO

CREATE PROC spInsertProduct

@CategoryID int, @ProductCode varchar(10),

@ProductName varchar(255), @ListPrice money,

@DiscountPercent money

AS

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

THROW 50001, 'Not a valid CategoryID!', 1;

IF @ListPrice < 0

THROW 50001, 'ListPrice should not be a negative number.', 1;

If @DiscountPercent < 0

THROW 50001, 'DiscountPercent should not be a negative number.', 1;

ELSE

INSERT Products

VALUES (@CategoryID, @ProductCode,

@ProductName,'', @ListPrice,

@DiscountPercent, GETDATE());

RETURN @@IDENTITY;

--execute procedure with negative discount percent

BEGIN TRY

EXEC spInsertProduct

3,'userProduct','userPrdtName',500,-6;

END TRY

BEGIN CATCH

PRINT 'An error occurred.';

PRINT 'Message: '+ ERROR\_MESSAGE();

IF ERROR\_NUMBER() >= 50000

PRINT 'This is a custom error message.';

END CATCH;

--execute procedure with negative list price

BEGIN TRY

EXEC spInsertProduct

3,'user\_product','user\_pdt\_name',-500,10;

END TRY

BEGIN CATCH

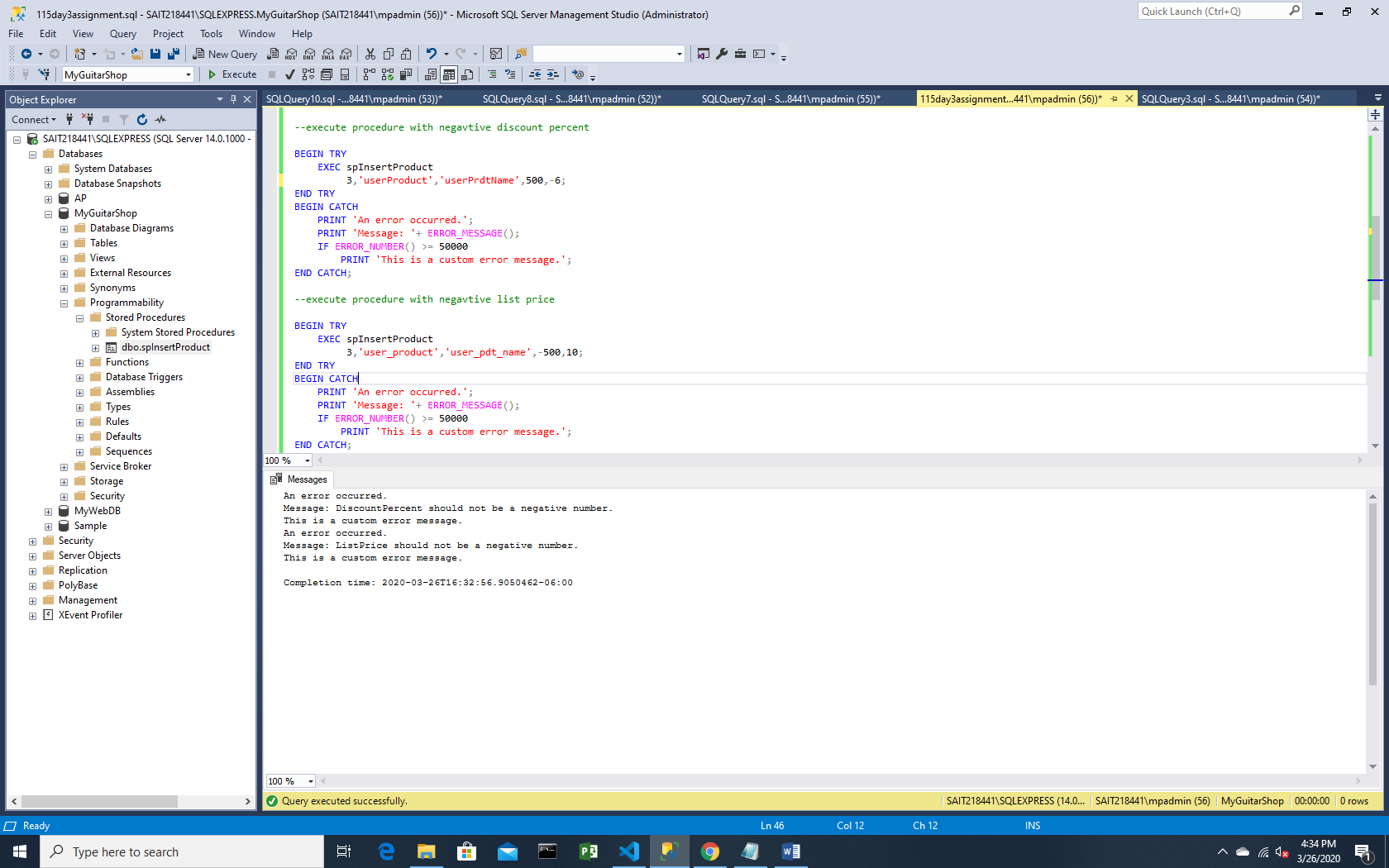
PRINT 'An error occurred.';

PRINT 'Message: '+ ERROR\_MESSAGE();

IF ERROR\_NUMBER() >= 50000

PRINT 'This is a custom error message.';

END CATCH;



--execute with correct data

BEGIN TRY

EXEC spInsertProduct

3,'product123','user\_pdt\_name',500,10;

END TRY

BEGIN CATCH

PRINT 'An error occurred.';

PRINT 'Message: '+ ERROR\_MESSAGE();

IF ERROR\_NUMBER() >= 50000

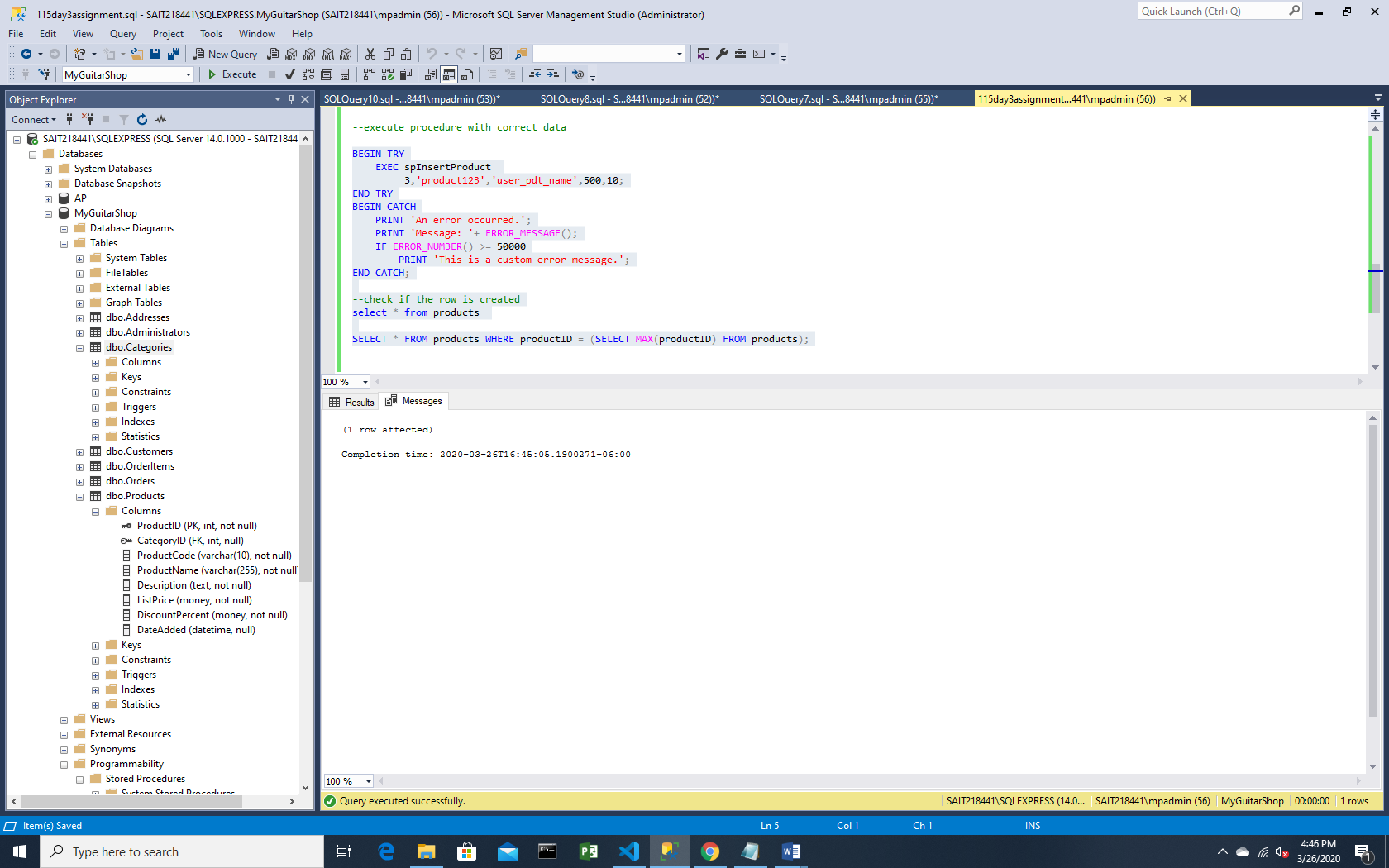
PRINT 'This is a custom error message.';

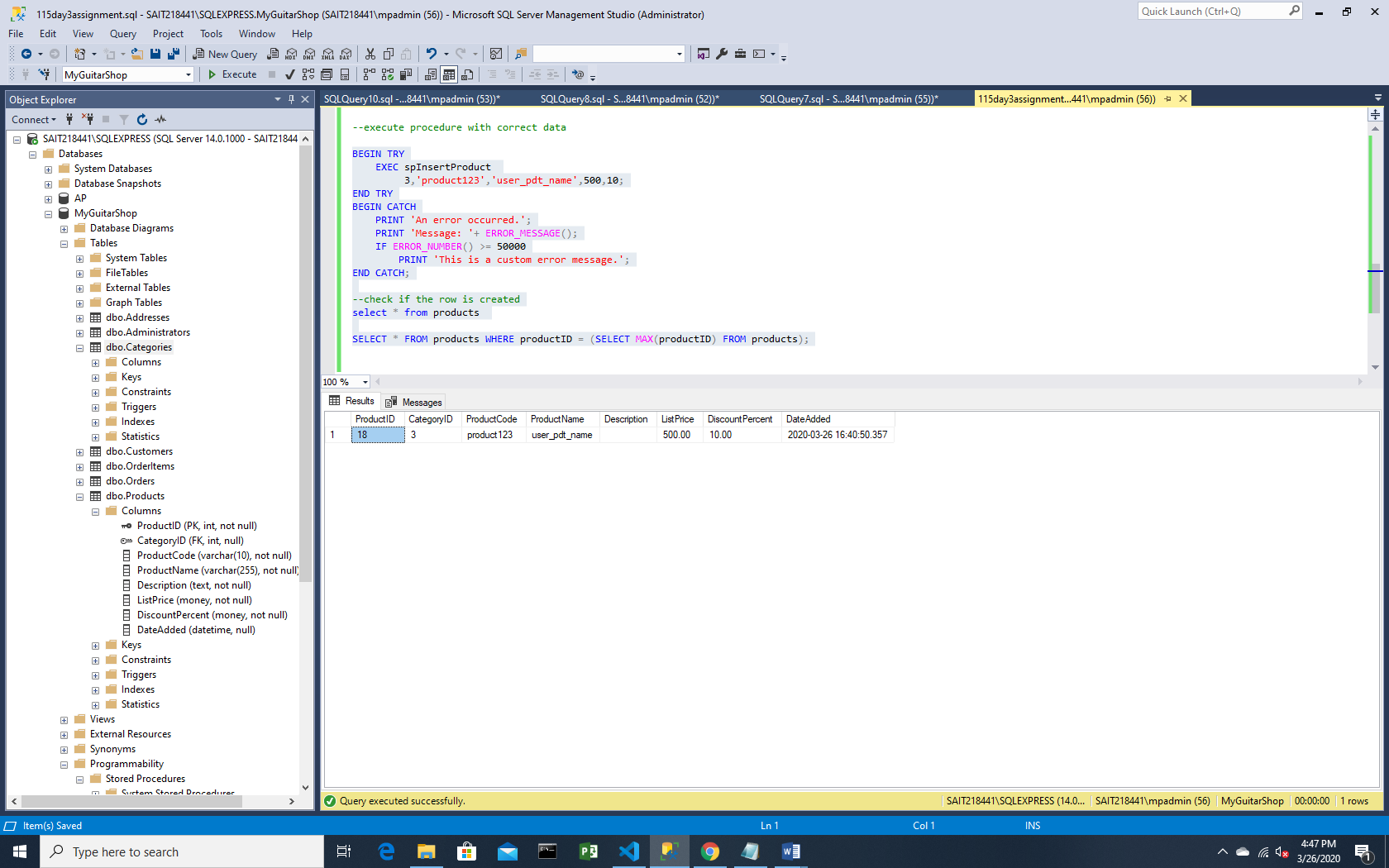
END CATCH;

--check if the row is created

select \* from products

SELECT \* FROM products WHERE productID = (SELECT MAX(productID) FROM products);





2)create two functions

--- create function to calculate the discount price of an item in orderitems table (discount amount - item price)

-- function should have one input parameter for itemid

-- and return discount price

create function fnDiscountPrice

(@ItemID int)

returns money

begin

return(select (ItemPrice - DiscountAmount)

from OrderItems

where ItemID = @ItemID);

end

--create another function to calculate the total amount of an item in orderitems table (discount price \* Quantity)

--fn should have one input parameter ItemID and should

--return total amount for that item

create function fnItemTotal

(@ItemID int)

returns money

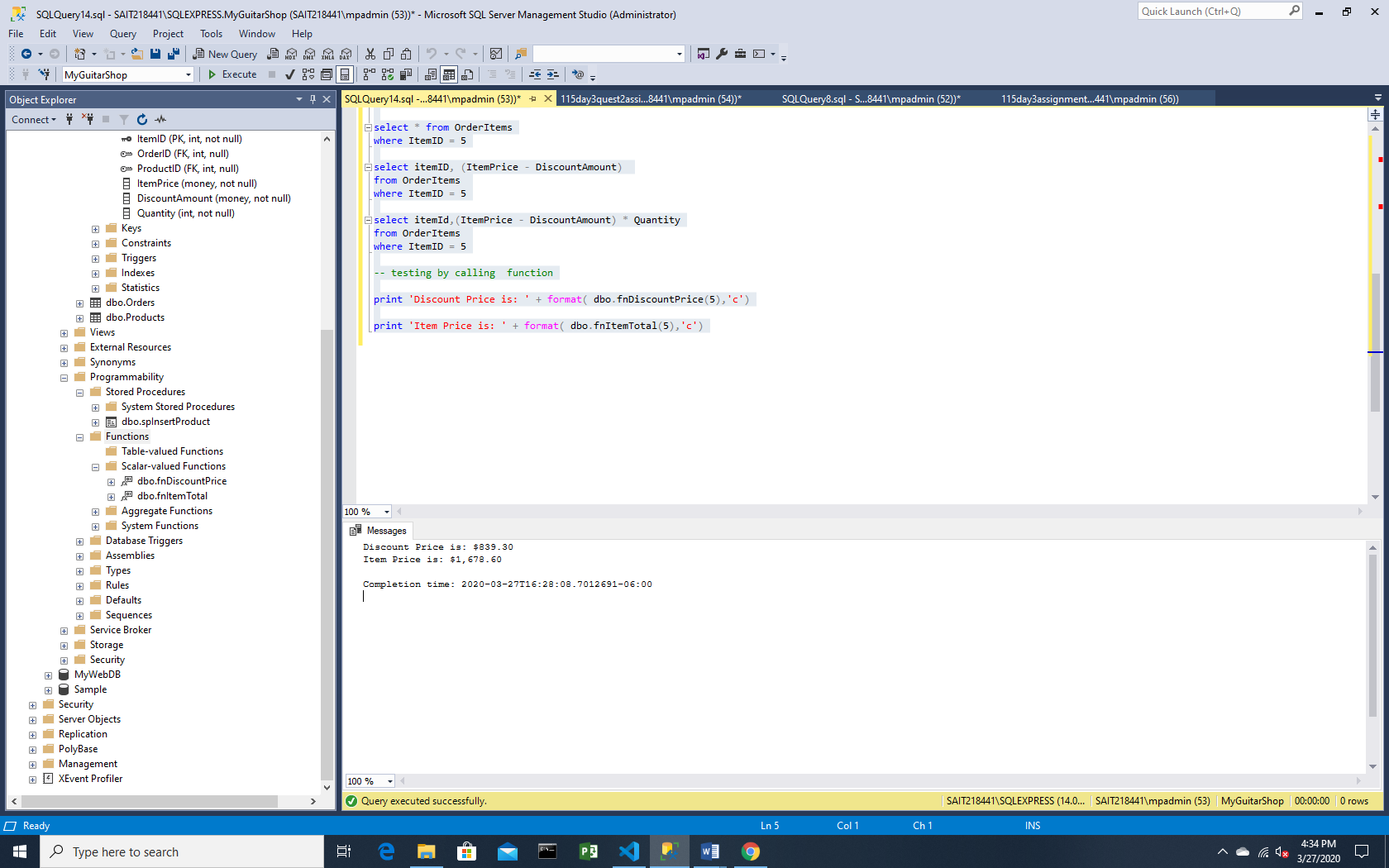
begin

return(select dbo.fnDiscountPrice(@ItemID) \* Quantity

from OrderItems

where ItemID = @ItemID);

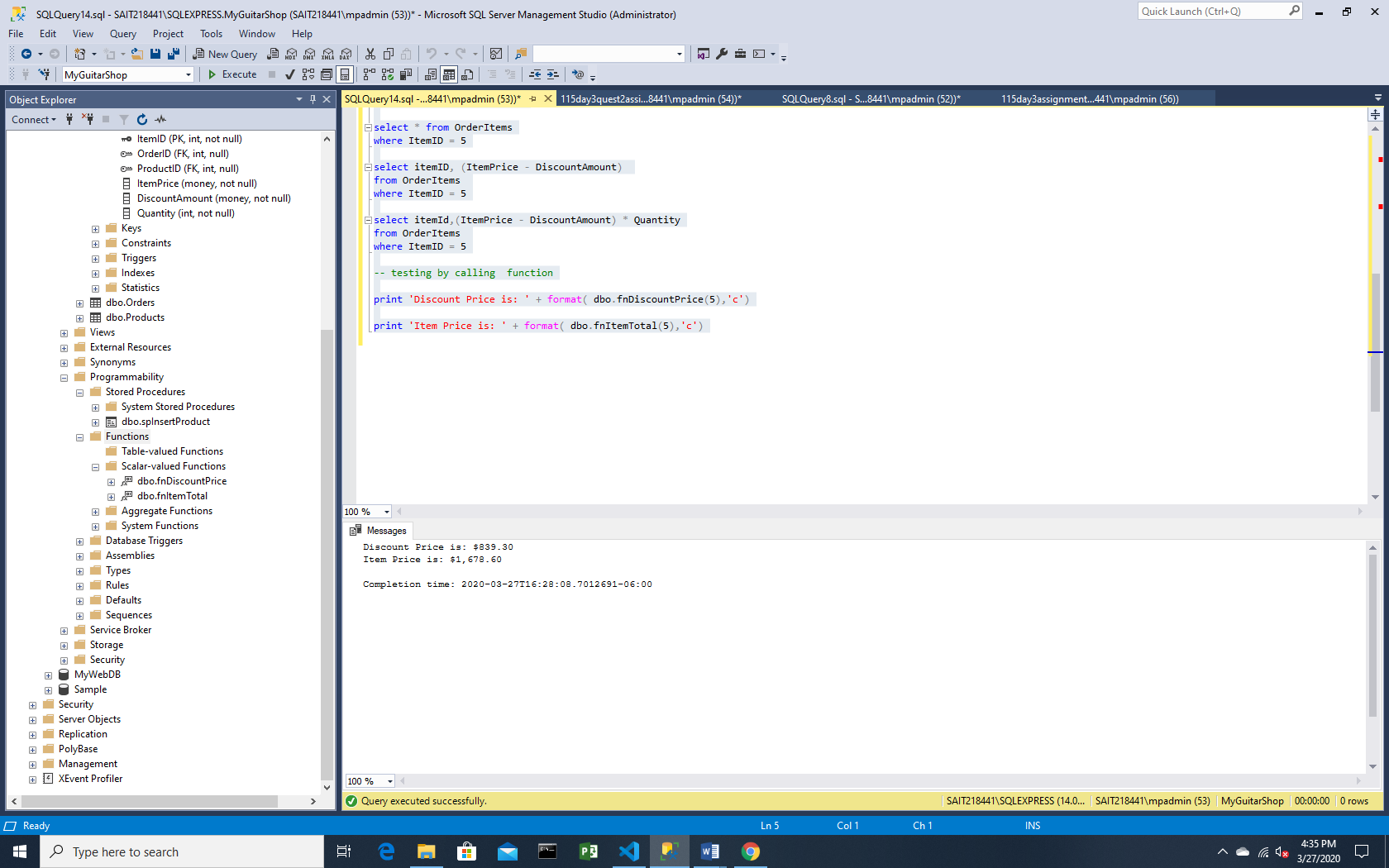
end



-- testing by calling function

print 'Discount Price is: ' + format(dbo.fnDiscountPrice(5),'c')

print 'Item Price is: ' + format(dbo.fnItemTotal(5),'c')



--testing by select query

select \* from OrderItems

where ItemID = 5

select ItemID, (ItemPrice - DiscountAmount)

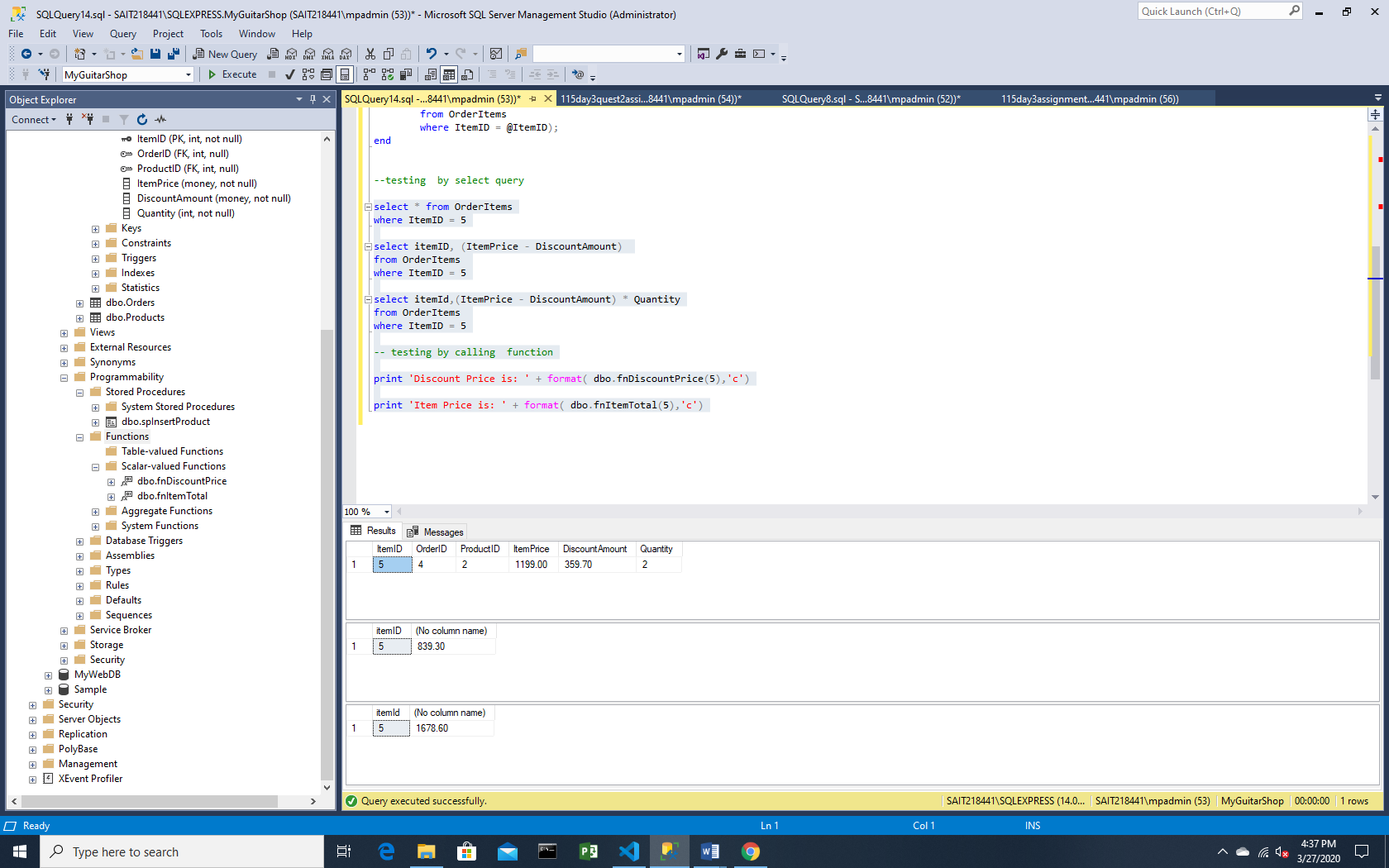
from OrderItems

where ItemID = 5

select ItemID,(ItemPrice - DiscountAmount) \* Quantity

from OrderItems

where ItemID = 5



3) create a trigger named Products\_Update

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

DROP TRIGGER Products\_Update;

GO

create TRIGGER Products\_Update

ON Products

AFTER UPDATE

AS

DECLARE @DiscountPercent money

IF EXISTS --Test whether DiscountPercent was changed

(SELECT \*

FROM Deleted JOIN Products

ON Deleted.Productid = Products.ProductID

WHERE Deleted.DiscountPercent <> Products.DiscountPercent)

BEGIN

select @DiscountPercent = DiscountPercent

from

Products

where ProductID IN (SELECT ProductID FROM Inserted);

IF @DiscountPercent < 0 or @DiscountPercent > 100

THROW 50027, 'Discount Percent should be between 0 and 100.', 1;

ElSE IF @DiscountPercent < 1

UPDATE Products

SET DiscountPercent =@DiscountPercent \* 100

WHERE ProductID IN (SELECT ProductID FROM Inserted);

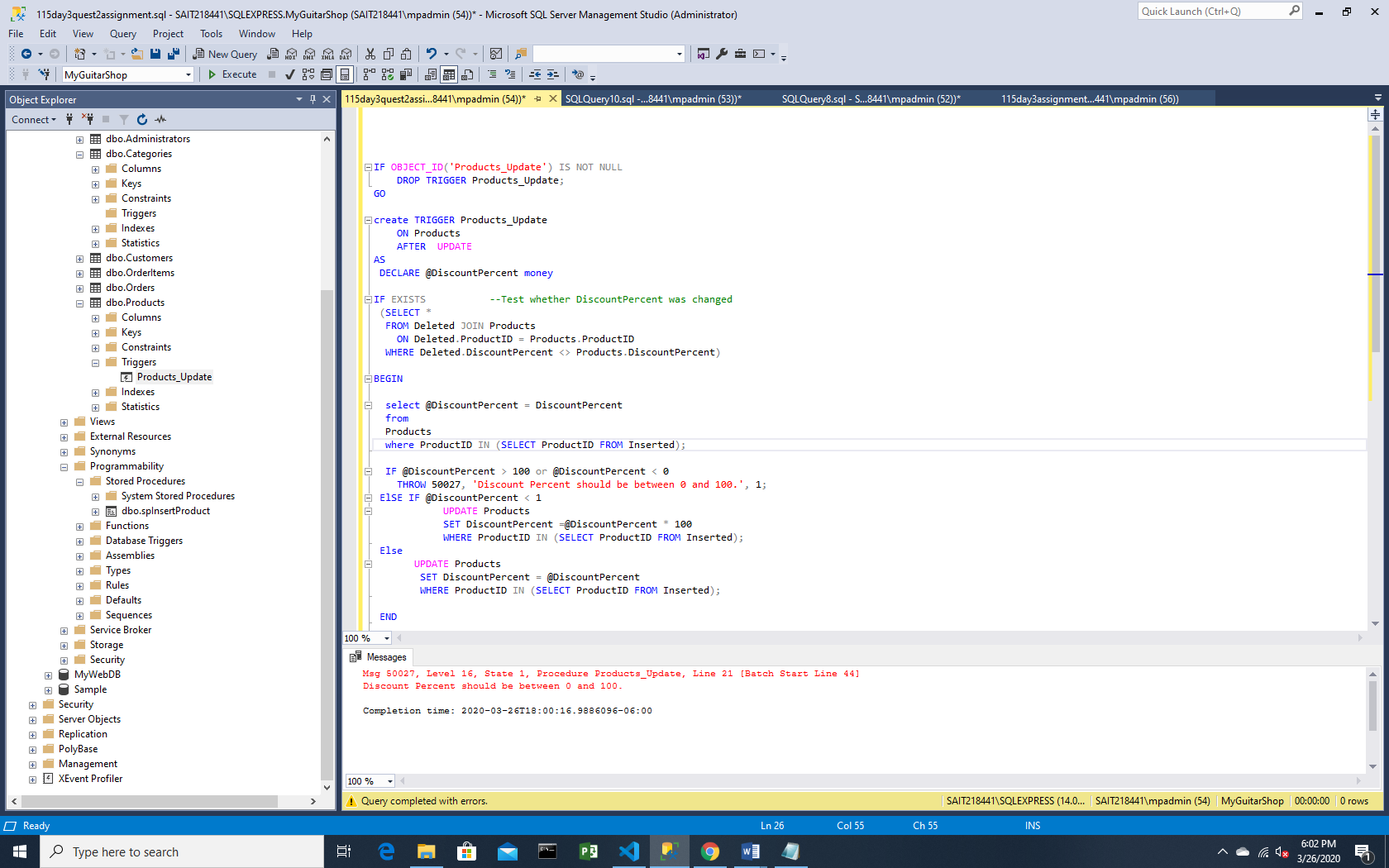
Else

UPDATE Products

SET DiscountPercent = @DiscountPercent

WHERE ProductID IN (SELECT ProductID FROM Inserted);

END



--testing for discount pirce greater than 100

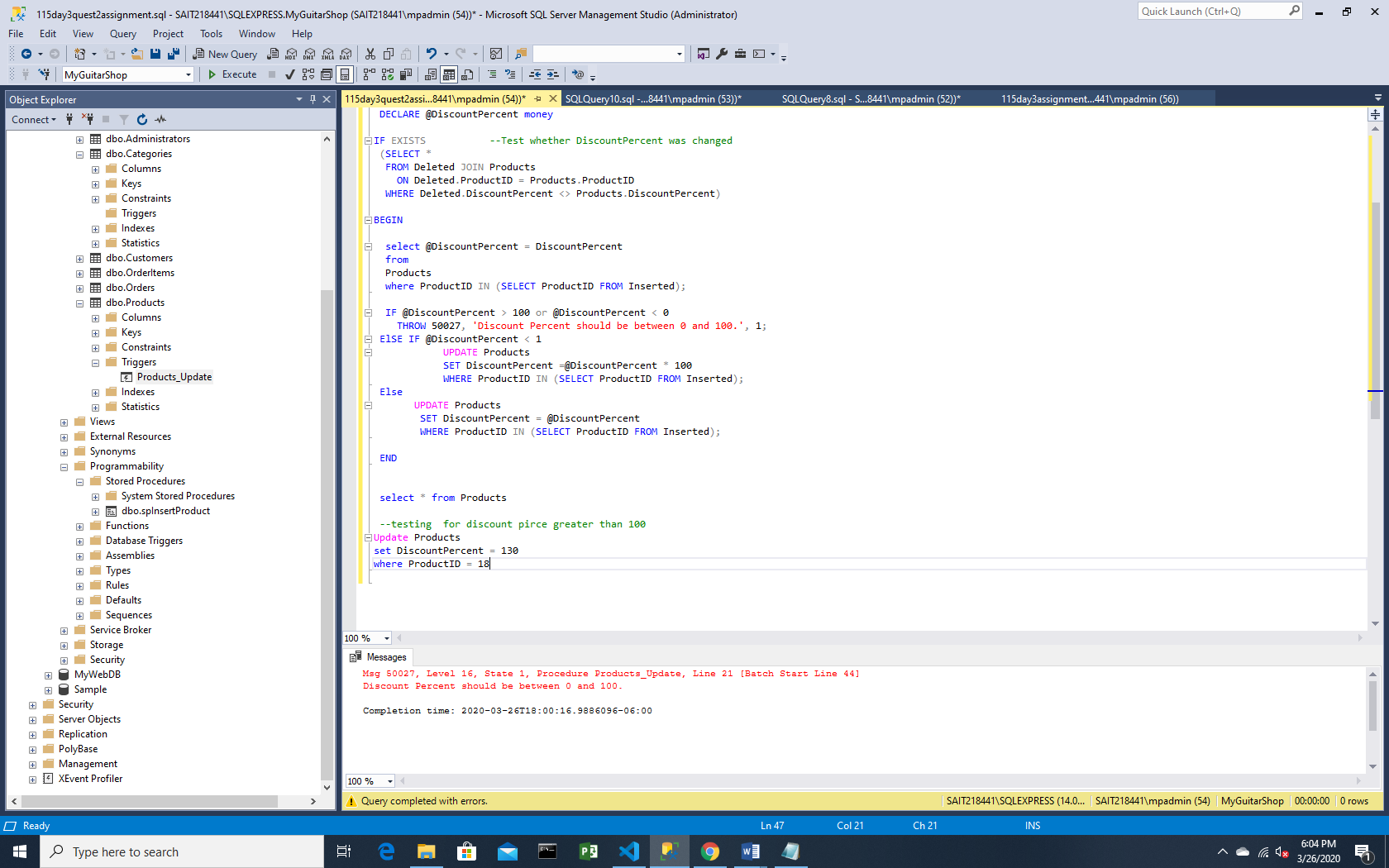
Update Products

set DiscountPercent = 130

where ProductID = 18

select \* from products

where ProductID =18



--testing for discount pirce less than 0

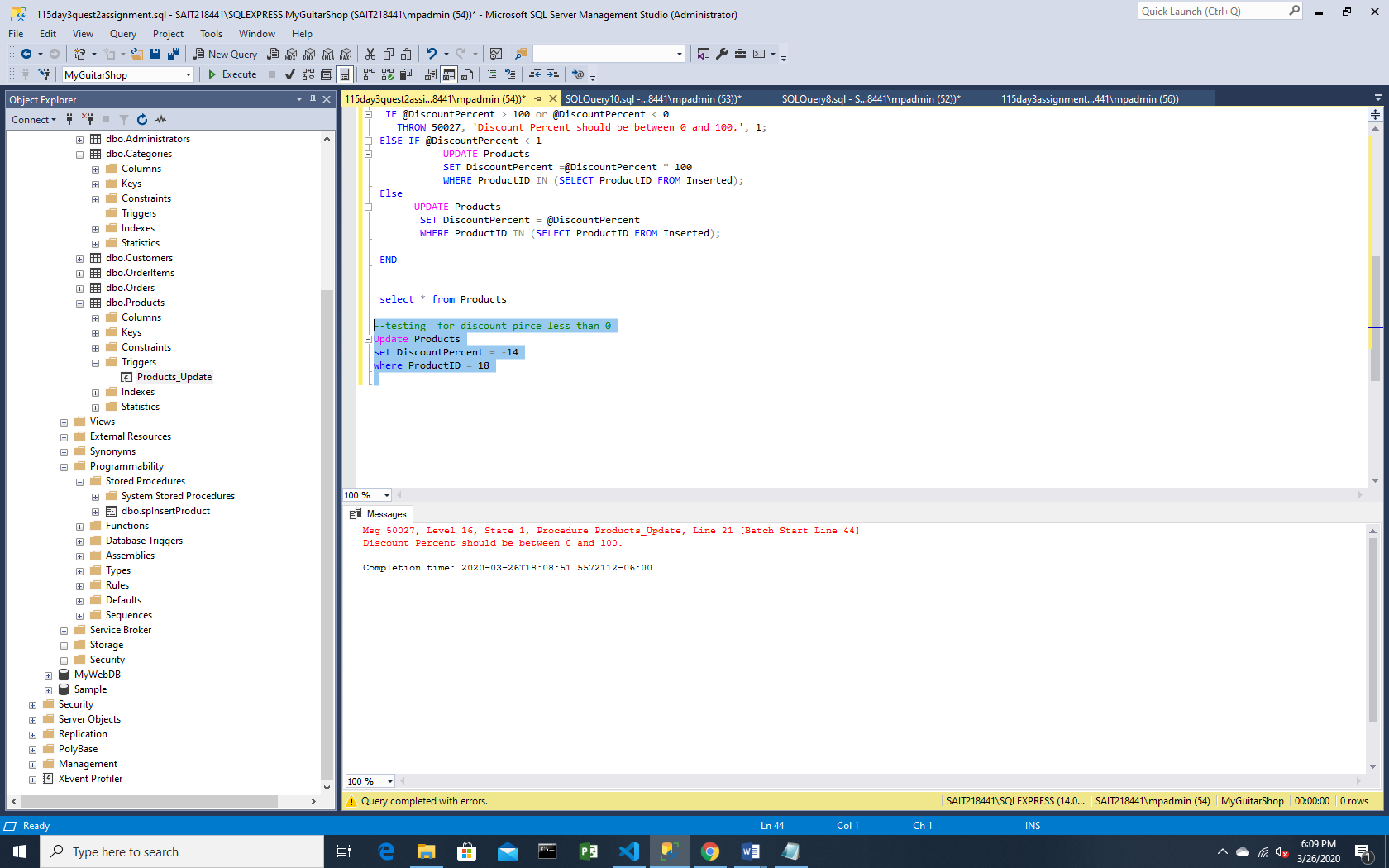
Update Products

set DiscountPercent = -14

where ProductID = 18

select \* from products

where ProductID =18



--testing for discount price between 0 and 1

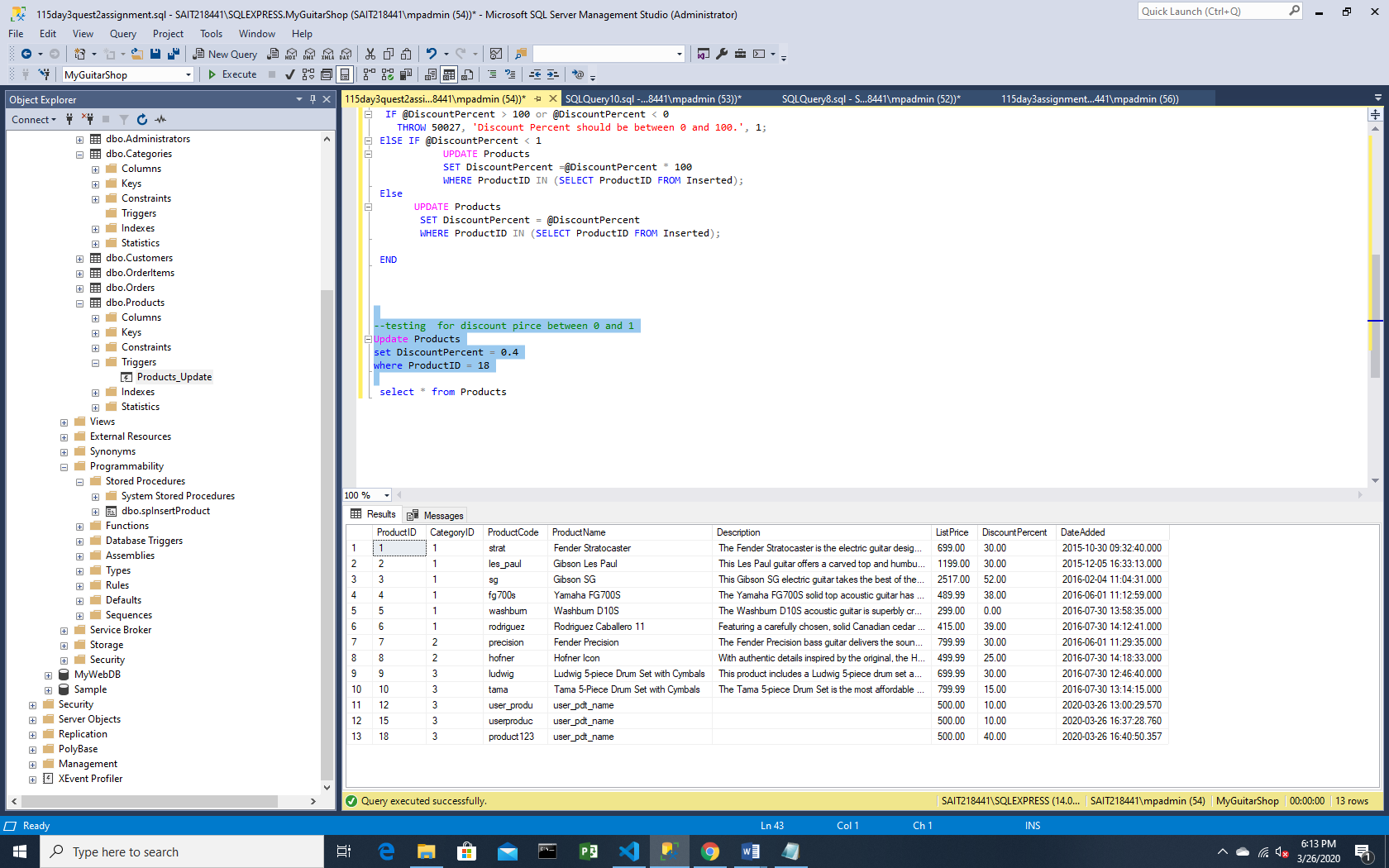
Update Products

set DiscountPercent = 0.4

where ProductID = 18

select \* from products

where ProductID =18



--testing for discount price between 0 and 100

Update Products

set DiscountPercent = 15

where ProductID = 18

select \* from products

where ProductID =18

