*Question 1*

*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.*

Answer 1

Use MyGuitarShop

Go

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

DROP PROC spInsertProduct;

GO

CREATE PROC spInsertProduct

@CategoryID int = NULL,

@ProductCode varchar(50) = NULL,

@ProductName varchar(100) = NULL,

@Description varchar(250) = NULL,

@ListPrice money = NULL,

@DisocuntPercent int = NULL,

@DateAdded smalldatetime = NULL

AS

IF @CategoryID IS NULL

THROW 50001, 'Invalid Cateogry ID.', 1;

IF @ProductCode IS NULL

THROW 50001, 'Invalid Product Code.', 1;

IF @ProductName IS NULL

THROW 50001, 'Invalid Product Name.', 1;

IF @Description IS NULL

SET @Description = '';

IF @ListPrice IS NULL

THROW 50001, 'Invalid ListPrice.', 1;

ELSE IF @ListPrice < 0

THROW 50001, 'List Price cannot be a negative number', 1;

IF @DisocuntPercent IS NULL

THROW 50001, 'Invalid Discount Percent', 1;

ELSE IF @DisocuntPercent < 0

THROW 50001, 'Disocunt Percent cannot be a negative number', 1;

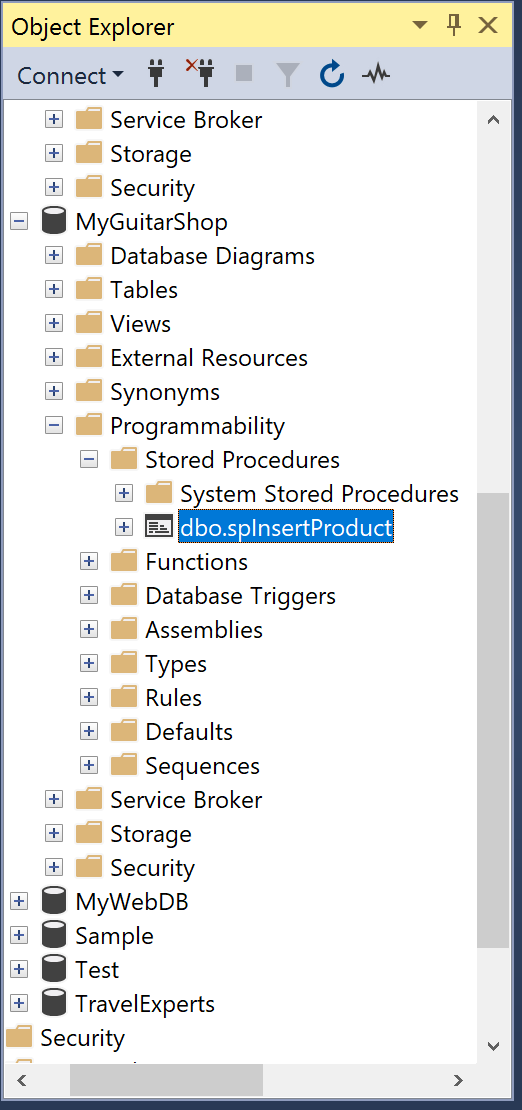
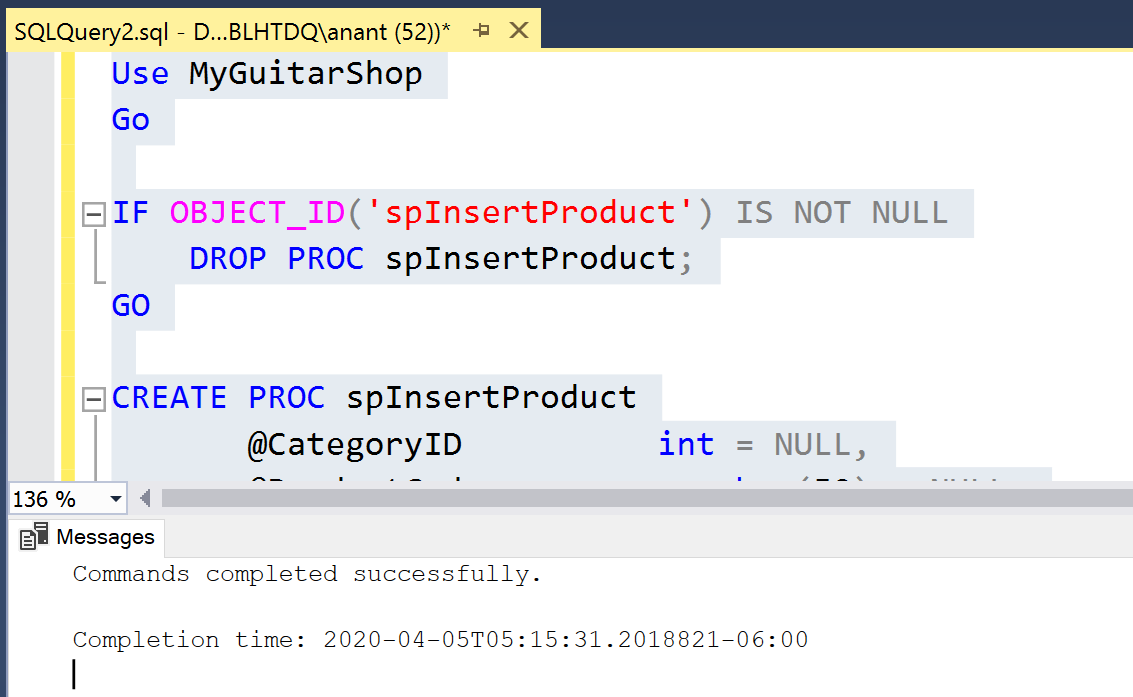
IF @DateAdded IS NULL

SET @DateAdded = GETDATE();

INSERT Products

VALUES (@CategoryID, @ProductCode, @ProductName, @Description, @ListPrice, @DisocuntPercent, @DateAdded);

Return @@identity



declare

@CategoryID int,

@ProductCode varchar(50),

@ProductName varchar(100),

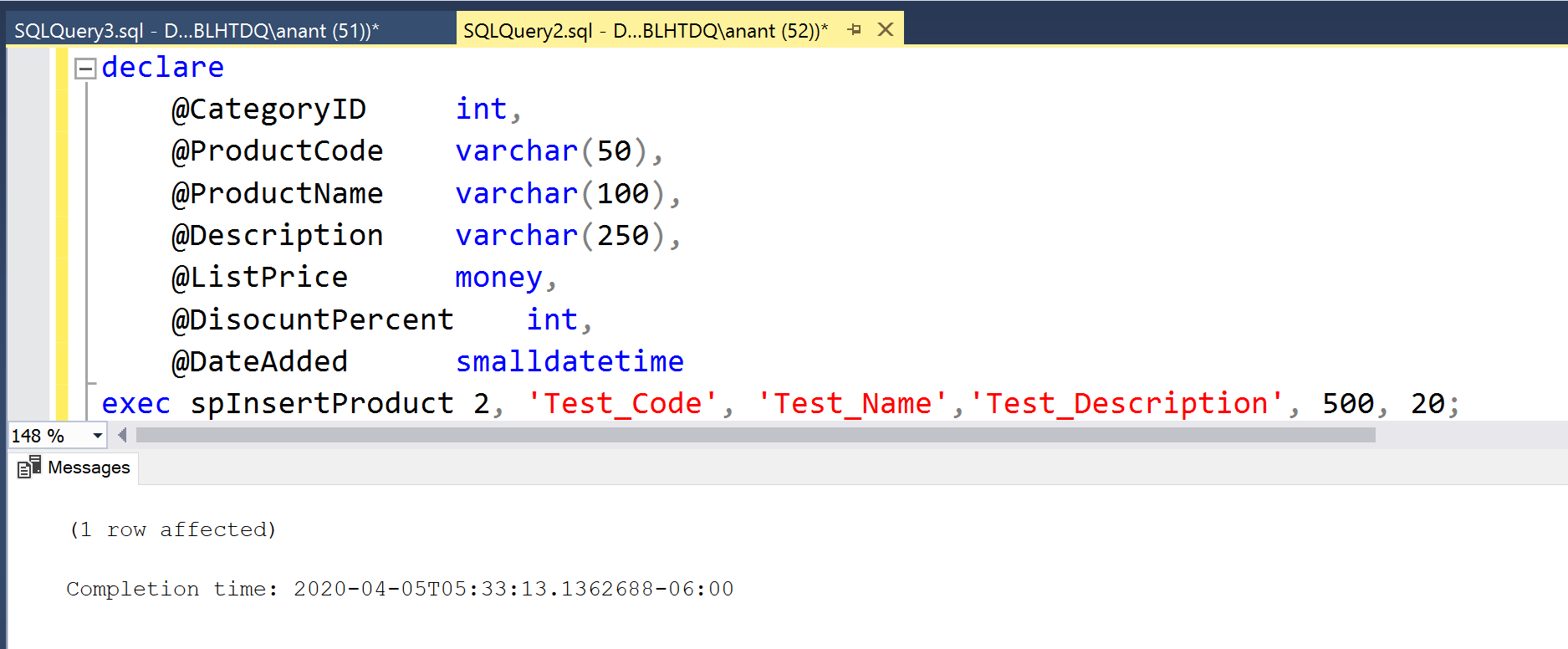
@Description varchar(250),

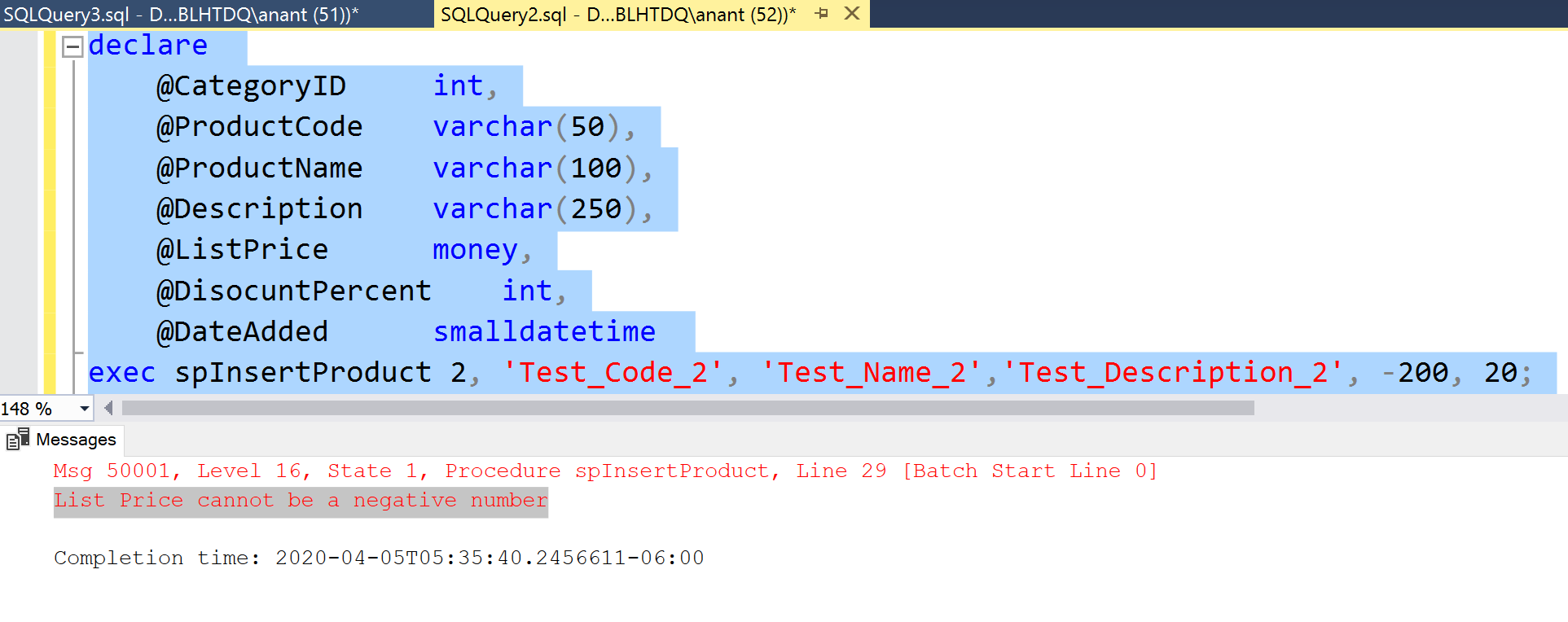
@ListPrice money,

@DisocuntPercent int,

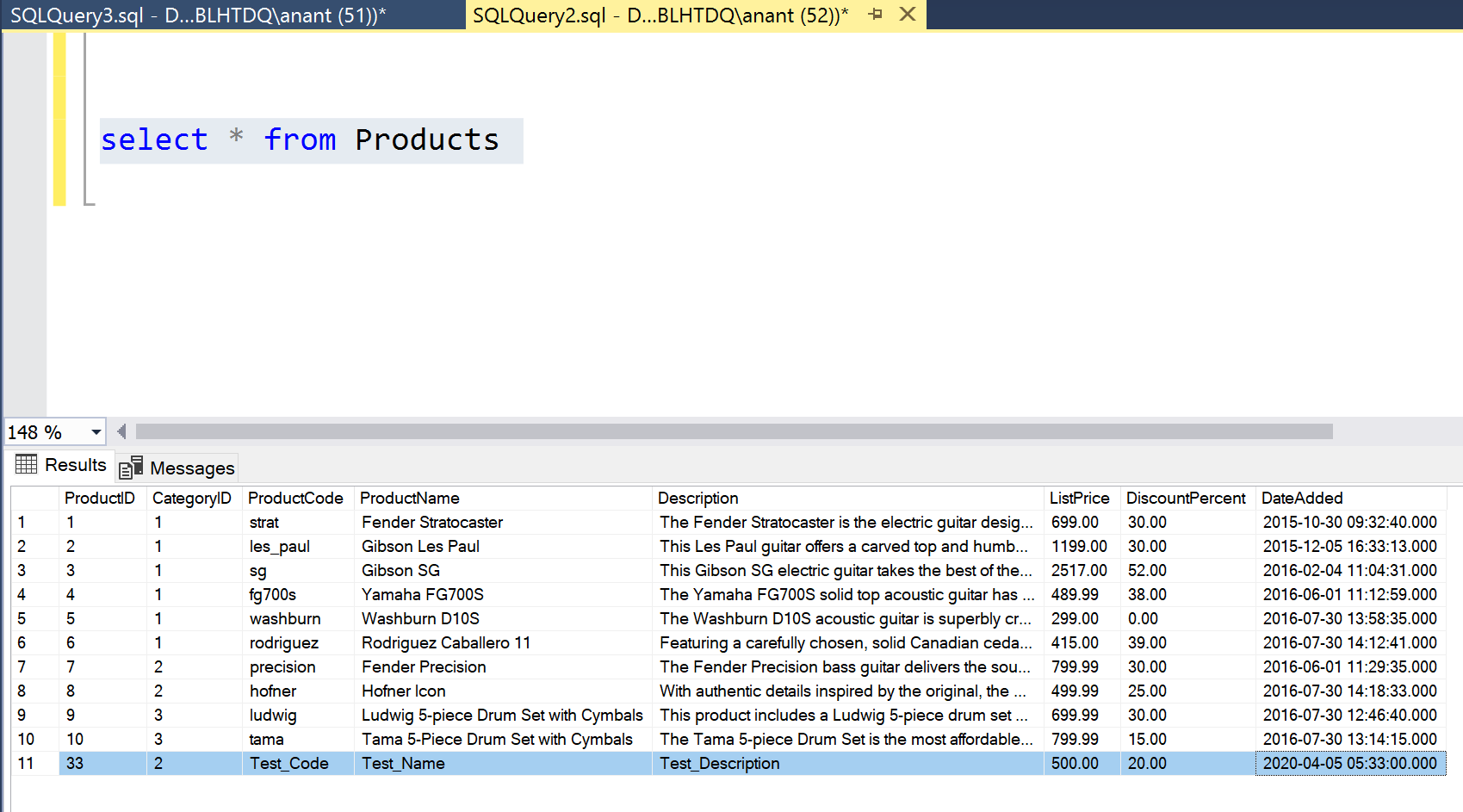
@DateAdded smalldatetime

exec spInsertProduct 2, 'Test\_Code', 'Test\_Name','Test\_Description', 500, 20

**Successful execution**

**\*\*\*\*\*Error with list price = -200**

**Select Statement**



*Question 2(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.*

Answer 2(a)

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

DROP FUNCTION fnDiscountPrice;

GO

CREATE FUNCTION fnDiscountPrice

(@ItemID int)

RETURNS money

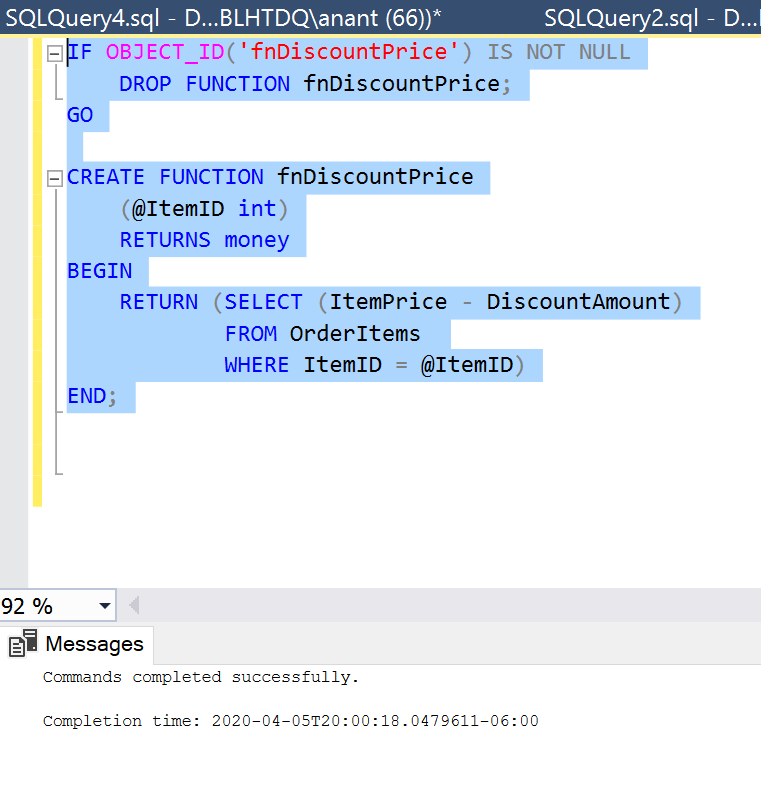
BEGIN

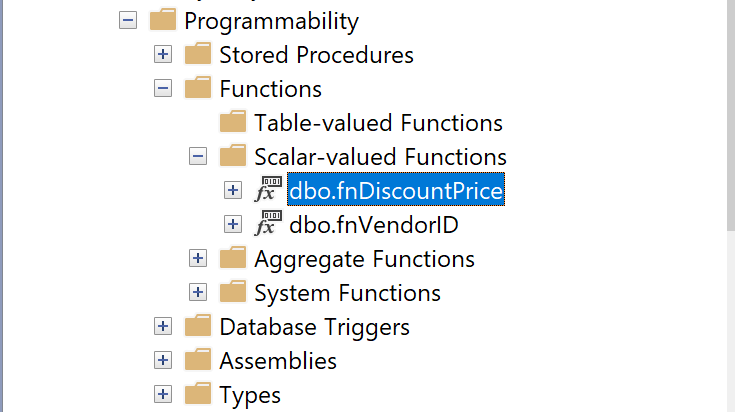
RETURN (SELECT (ItemPrice - DiscountAmount)

FROM OrderItems

WHERE ItemID = @ItemID)

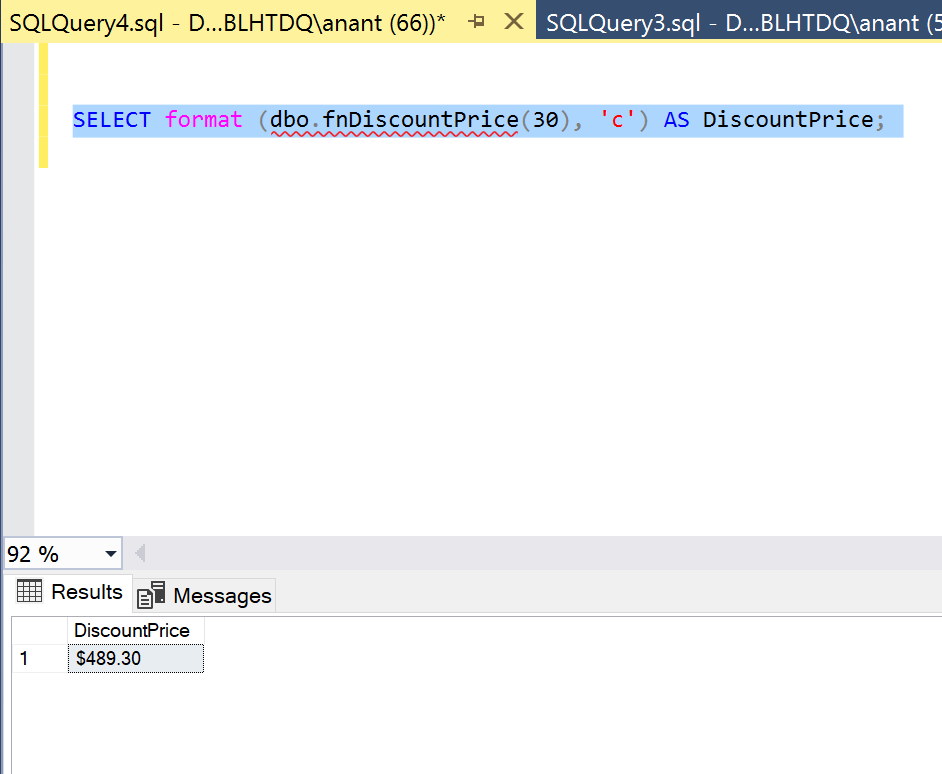
END;





For testing we passed the ItemID = 30

SELECT format (dbo.fnDiscountPrice(30), 'c') AS DiscountPrice;



*Question 2(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.*

Answer 2(b)

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

DROP FUNCTION fnItemTotal;

GO

CREATE FUNCTION fnItemTotal

(@ItemID int)

RETURNS money

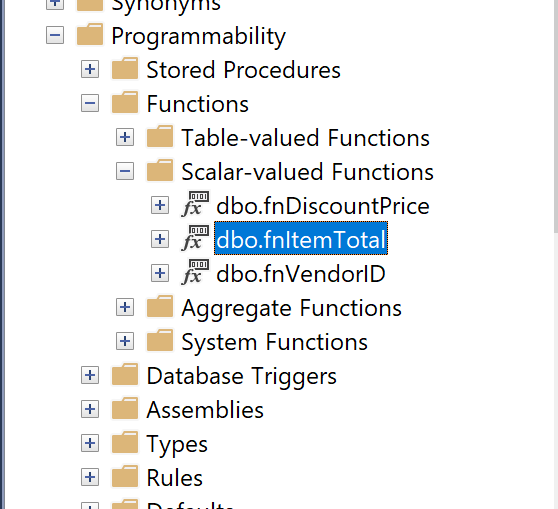
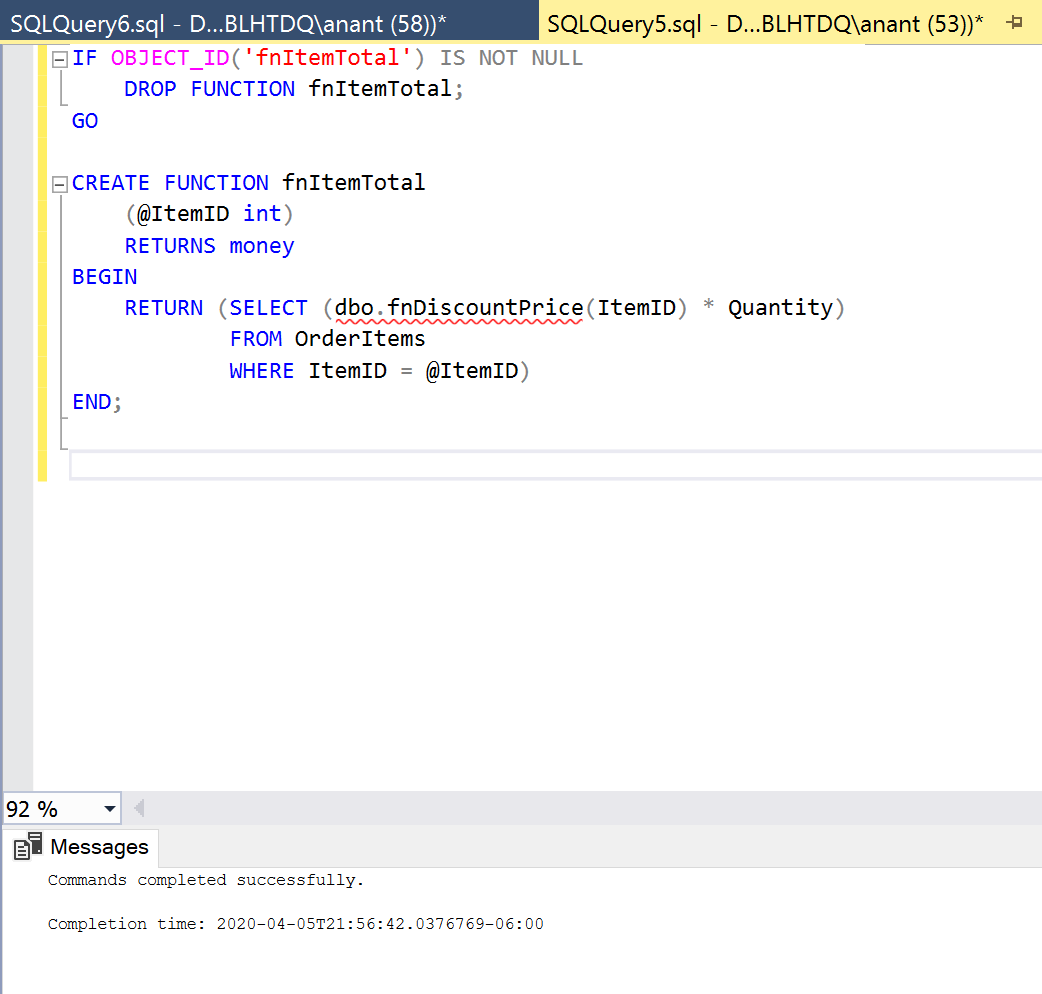
BEGIN

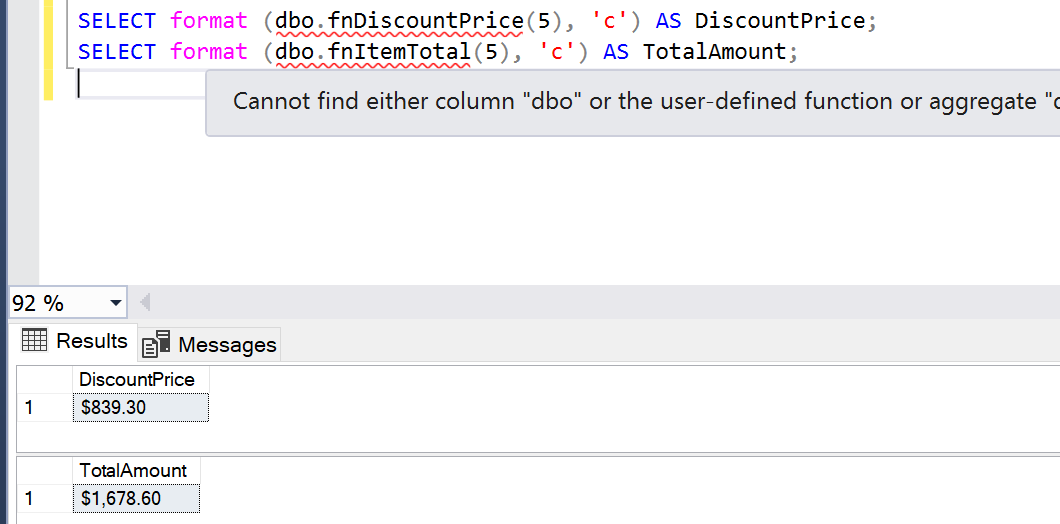
RETURN (SELECT (dbo.fnDiscountPrice(ItemID) \* Quantity)

FROM OrderItems

WHERE ItemID = @ItemID)

END;

For testing we passed the ItemID = 5 as its Quantity = 2

SELECT format (dbo.fnItemTotal(5), 'c')

AS TotalAmount;

*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.*

Answer 3

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

DROP TRIGGER Products\_UPDATE;

GO

CREATE TRIGGER Products\_UPDATE

ON Products

AFTER INSERT,UPDATE

AS

DECLARE @DiscountPercent float

SELECT @DiscountPercent = DiscountPercent FROM Inserted

IF @DiscountPercent < 0 or @DiscountPercent > 100

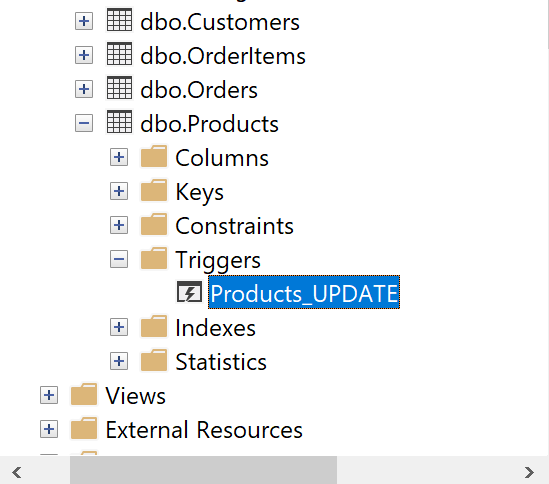
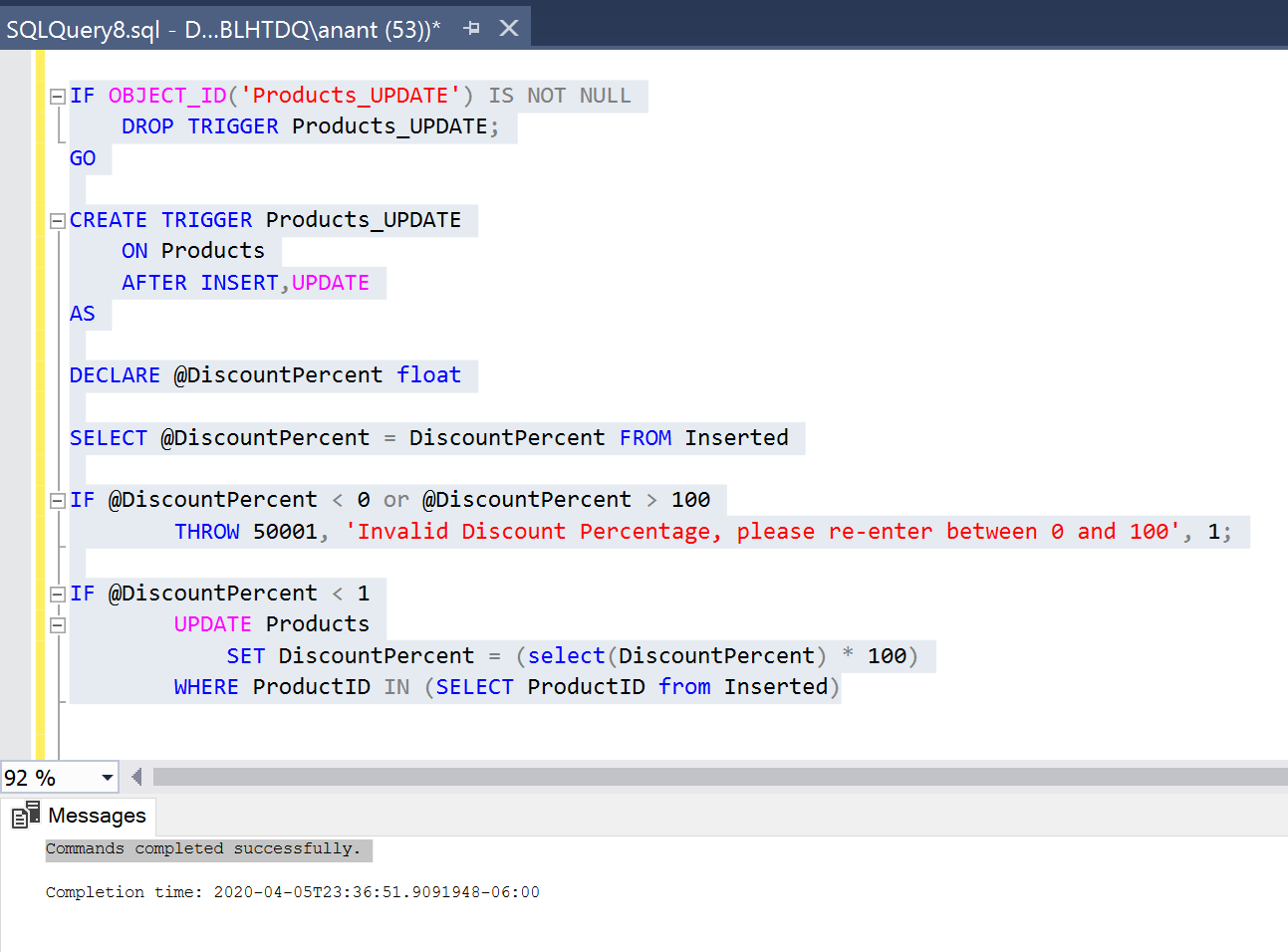
THROW 50001, 'Invalid Discount Percentage, please re-enter between 0 and 100', 1;

IF @DiscountPercent < 1

UPDATE Products

SET DiscountPercent = (select(DiscountPercent) \* 100)

WHERE ProductID IN (SELECT ProductID from Inserted)

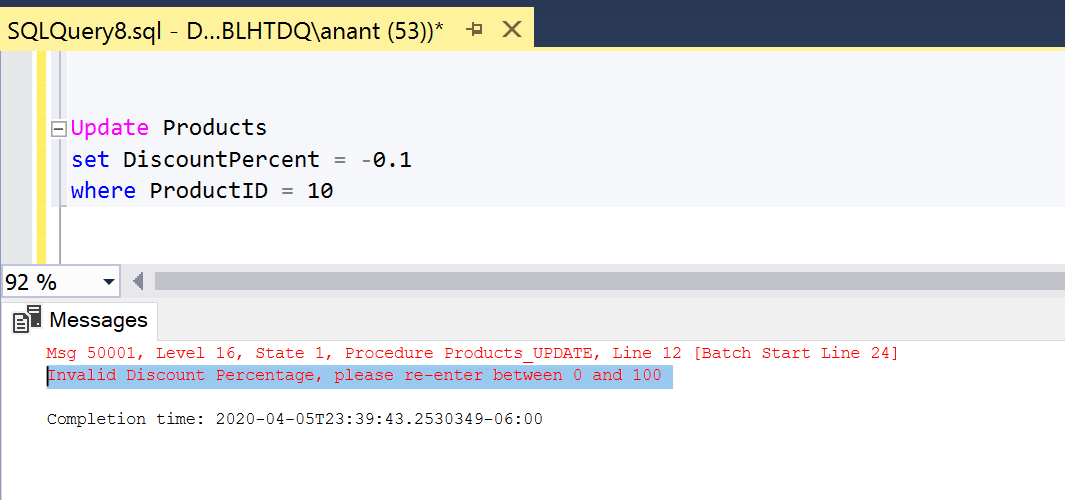
For testing I updated Discount Percent from ProductID = 10

1. When Discount percent < 0

Update Products

set DiscountPercent = -0.1

where ProductID = 10

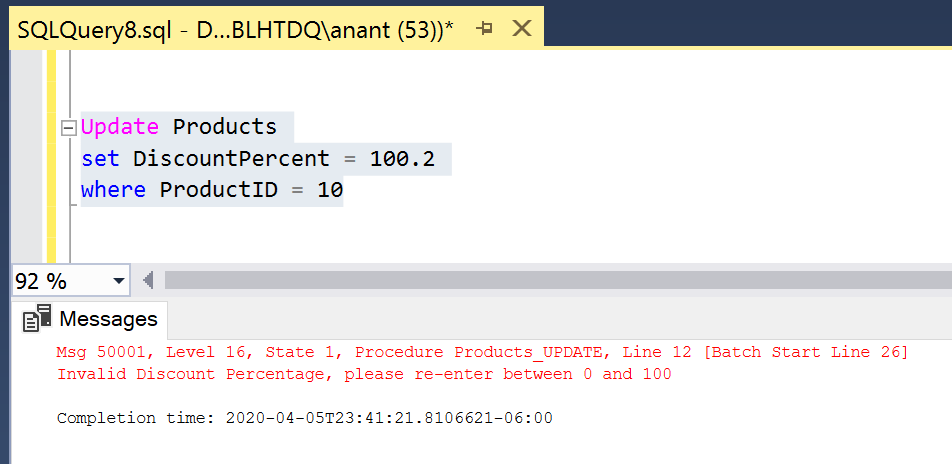


1. When Discount percent > 100

Update Products

set DiscountPercent = 100.2

where ProductID = 10

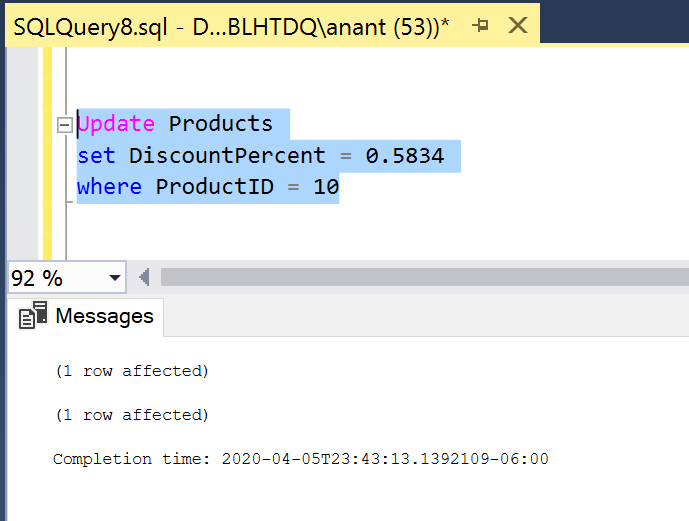


1. When Discount percent is a decimal i.e. less than 1 and greater than 0

Update Products

set DiscountPercent = 0.5834

where ProductID = 10



Products table updated:

