Creating SQL DML Triggers (Data Manipulation Language)

Login Triggers: Monitoring and Controlling Login Events

\*Open SQL Server Man Studio.

\*New Query Ctrl+N.

\*Create the table CustomersTriggers:

use mike;

go

create table CustomersTriggers(

CustomerID int not null,

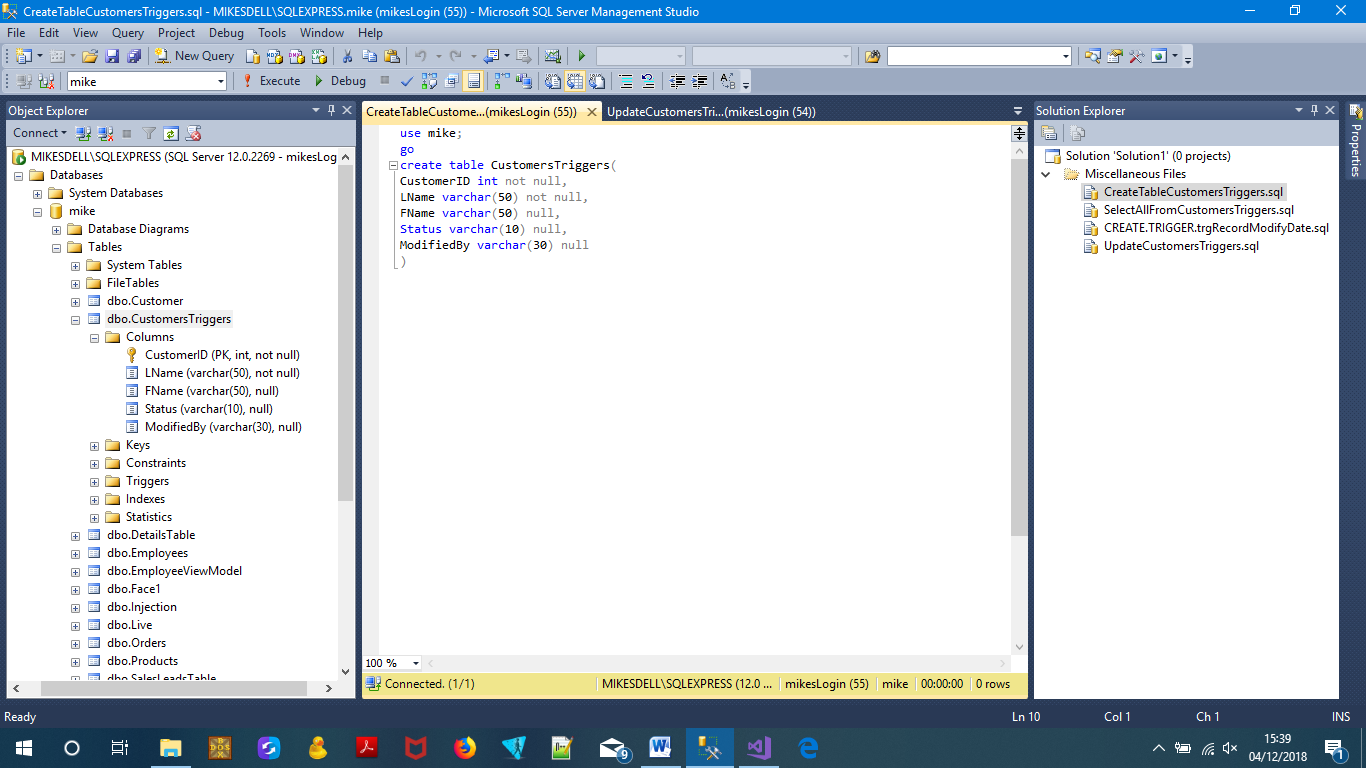
LName varchar(50) not null,

FName varchar(50) null,

Status varchar(10) null,

ModifiedBy varchar(30) null

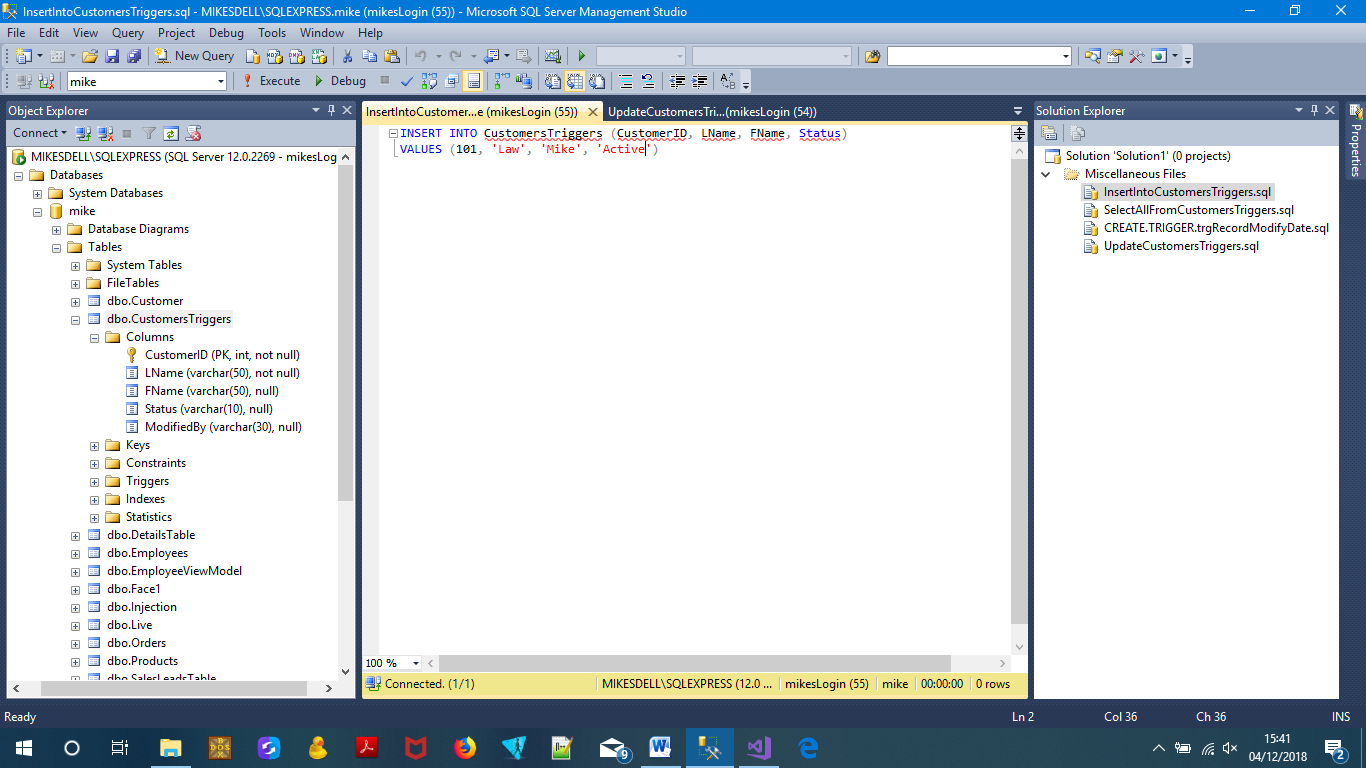
)



\*Add a customer with the following code:

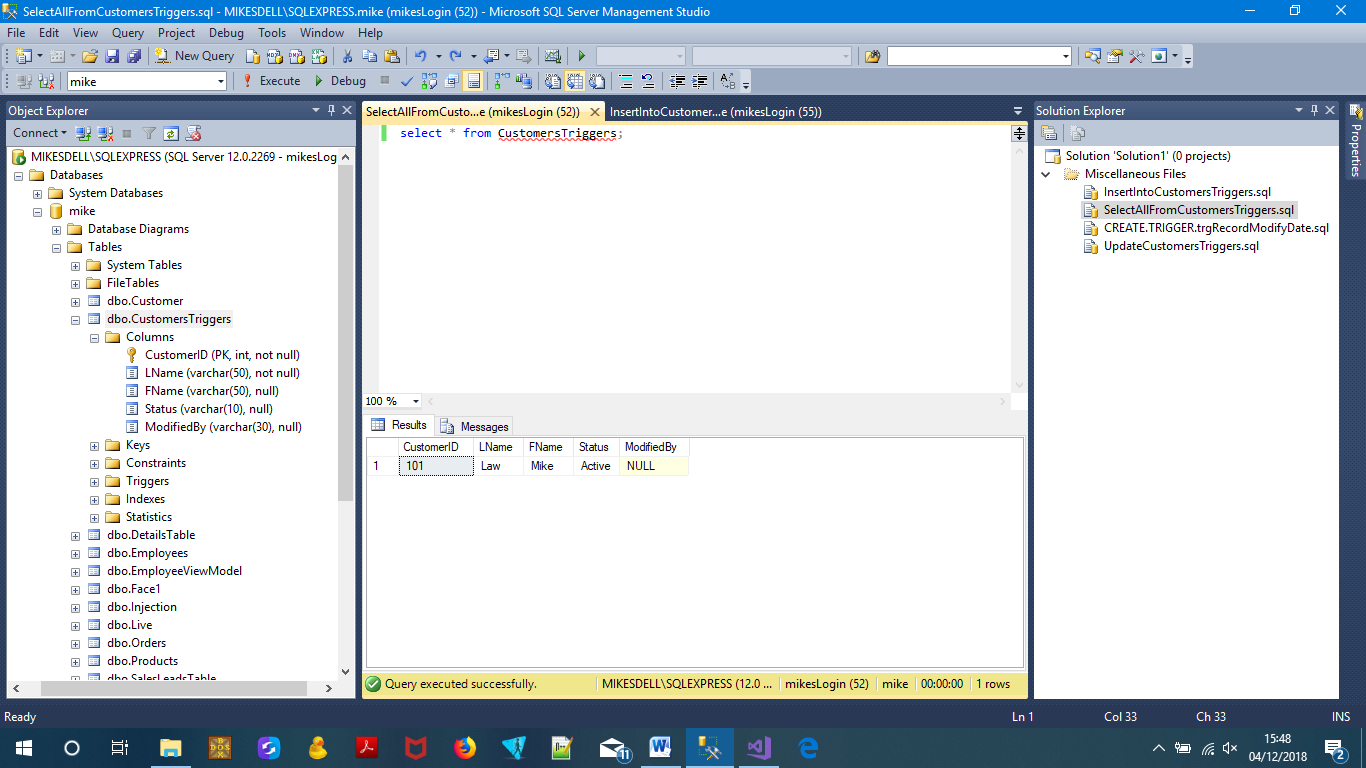
INSERT INTO CustomersTriggers (CustomerID, LName, FName, Status)

VALUES (101, 'Law', 'Mike', 'Active')



\*View your customer data with the following query:

select \* from CustomersTriggers;



You should see the row you added and a blank NULL ModifiedBy column.

\*Add an update trigger on the CustomersTriggers table with the following code:

CREATE TRIGGER trgRecordModifyDate

ON CustomersTriggers

AFTER UPDATE

AS

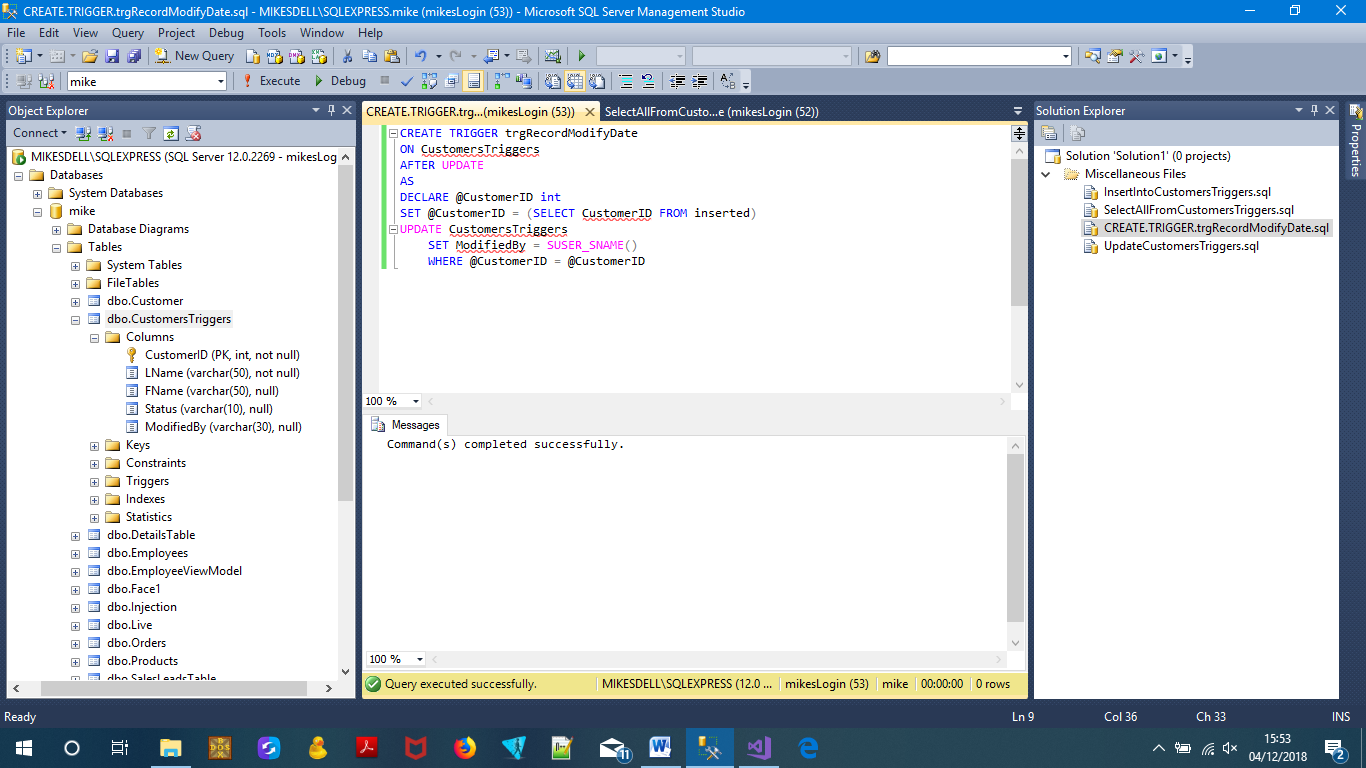
DECLARE @CustomerID int

SET @CustomerID = (SELECT CustomerID FROM inserted)

UPDATE CustomersTriggers

SET ModifiedBy = SUSER\_SNAME()

WHERE @CustomerID = @CustomerID



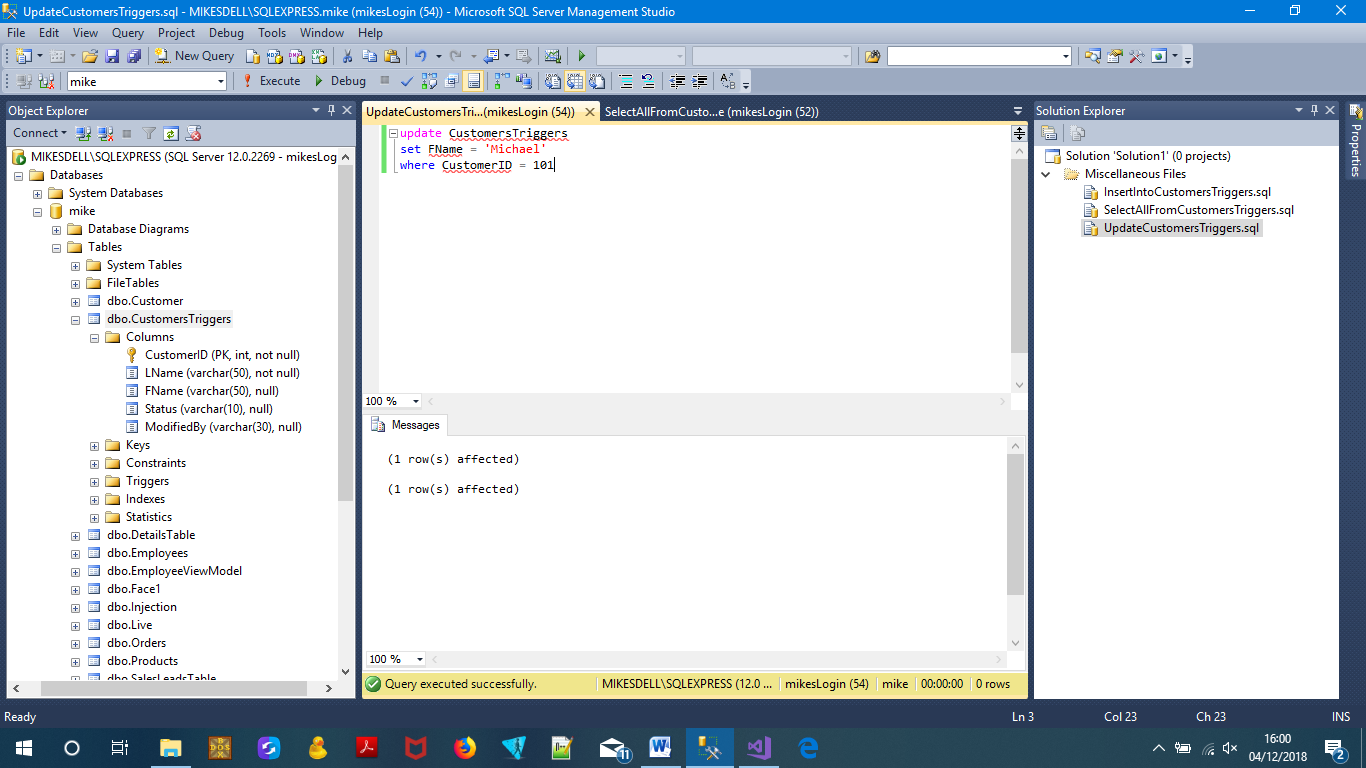
The @CustomerID variable used to capture CustomerID was modified from the last UPDATE statement. Then the suser\_sname( ) function captures the identity of the user that executed the UPDATE statement and stores it in the ModifiedBy column.

\*Modify the customer data with the following query:

update CustomersTriggers

set FName = 'Michael'

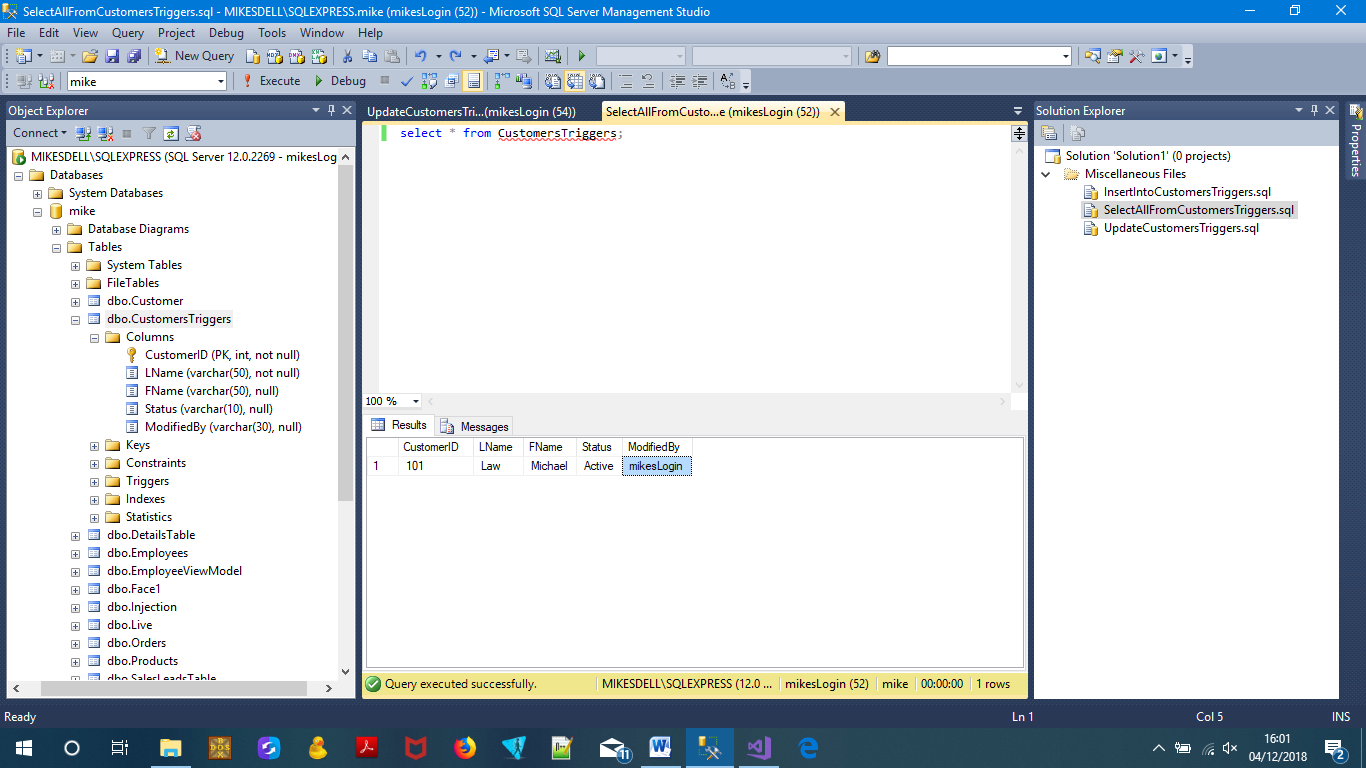
where CustomerID = 101



This causes the Update Trigger to fire and add data to the ModifiedBy column called “mikesLogin”, and you should see that the ModifiedBy column is no longer NULL.

Now view your new customer data with the following query:

select \* from CustomersTriggers;



\*Add an INSTEAD OF Trigger to prevent the deletion of customers with the following code:

CREATE TRIGGER trgNoDelete

ON CustomersTriggers

INSTEAD OF DELETE

AS

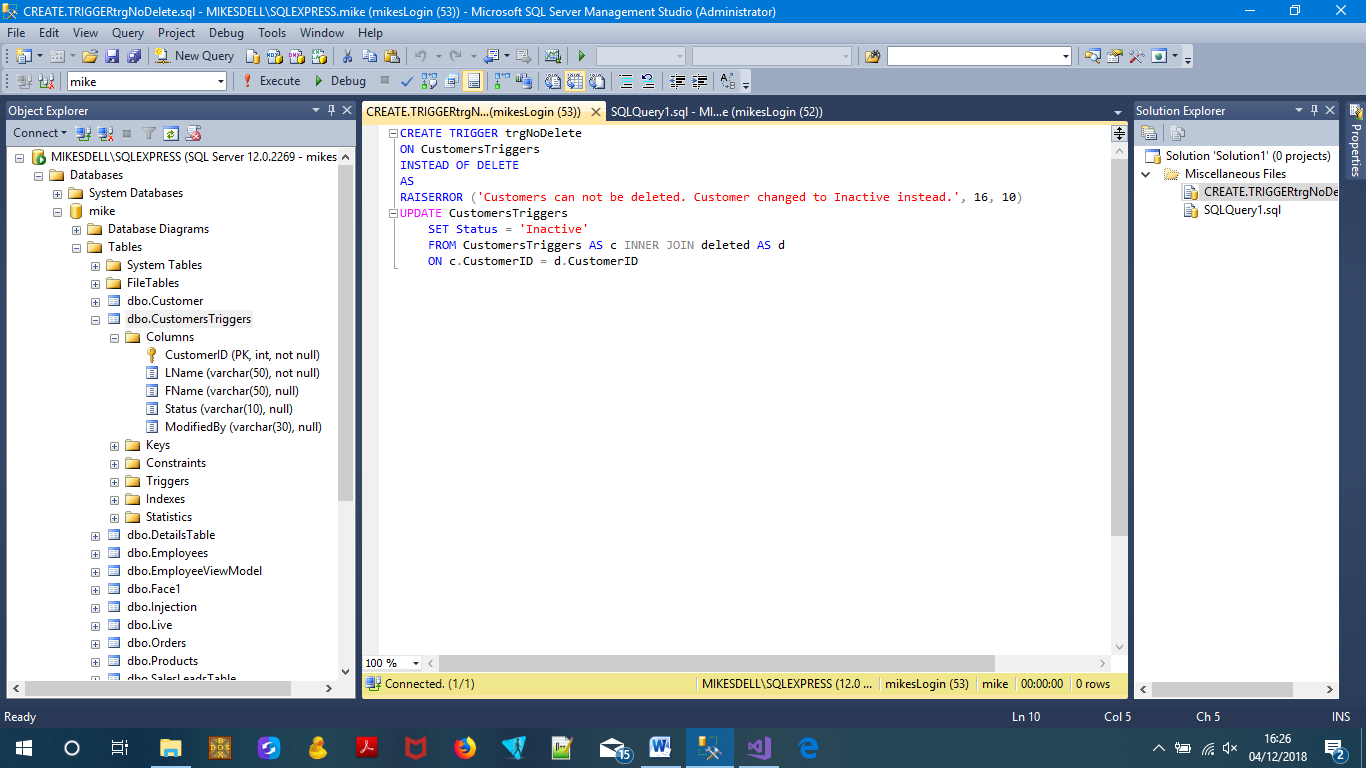
RAISERROR ('Customers can not be deleted. Customer changed to Inactive instead.', 16, 10)

UPDATE CustomersTriggers

SET Status = 'Inactive'

FROM CustomersTriggers AS c INNER JOIN deleted AS d

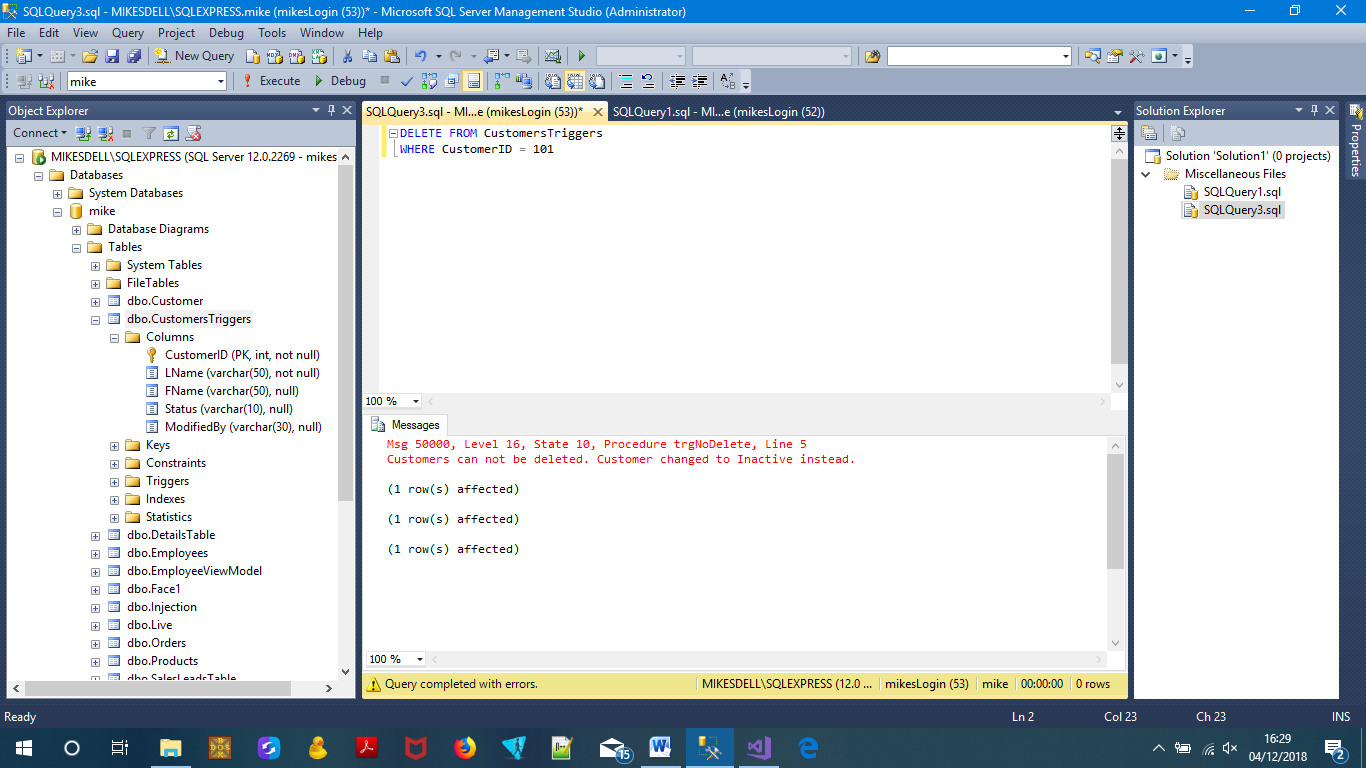
ON c.CustomerID = d.CustomerID



\*Try to delete a customer with the following code:

DELETE FROM CustomersTriggers

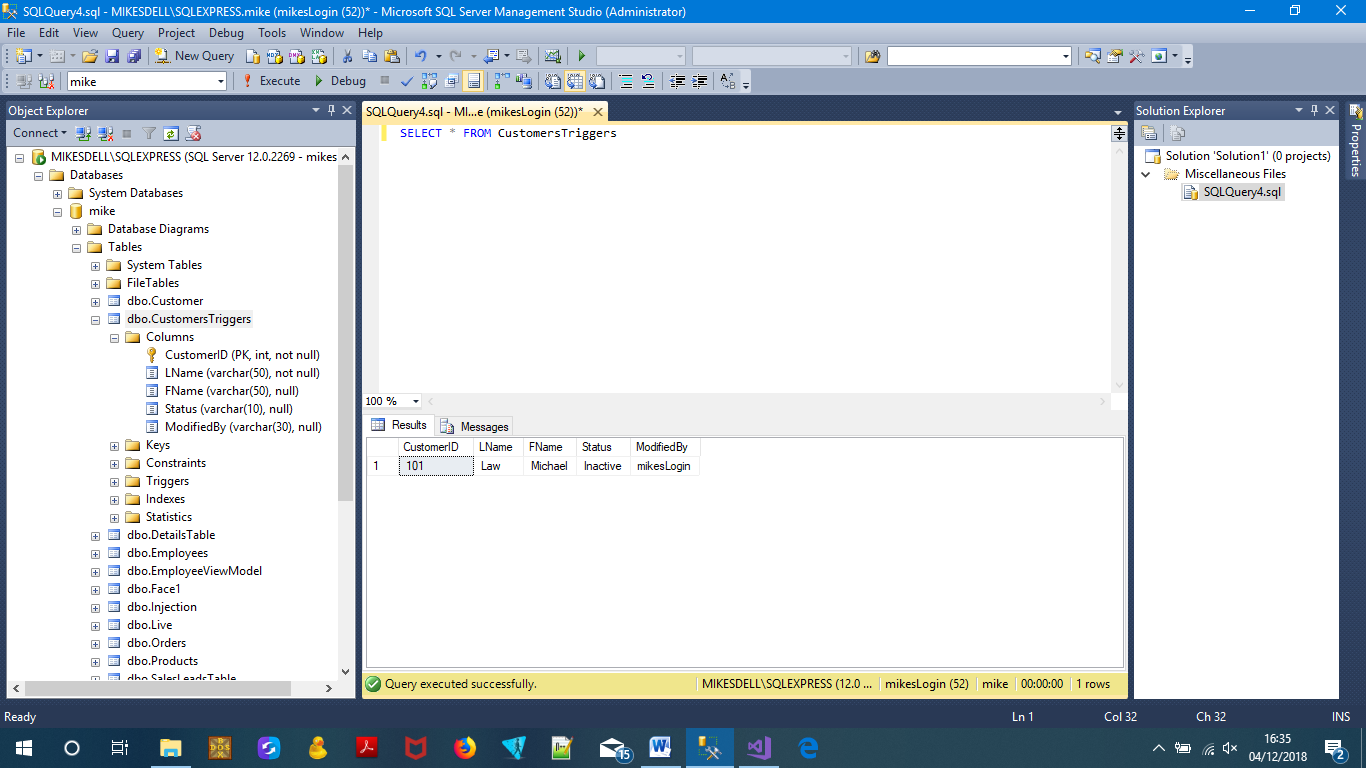
WHERE CustomerID = 101



Instead of deleting the customer, the error message is displayed.

\*View your new customer data with the query:

SELECT \* FROM CustomersTriggers



Notice that the status has been changed to Inactive.