Activity No. 13.1 - Trigger	
Name: Efa, Christian Guevarra, Hans Angelo Mendoza, John Renzo Nicolas, Sean Julian Vinluan, Armando	**Date:** 02/12/2022
Section: CPE21S3	**Instructor:** Dr. Jonathan Vidal Taylar

Objectives:

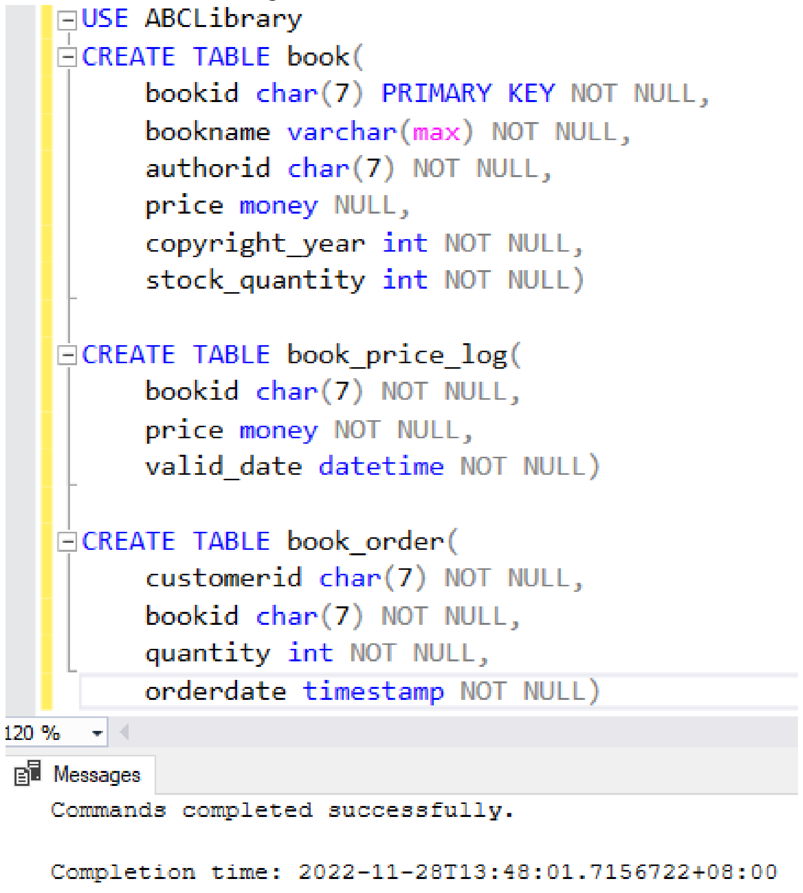
This activity aims to create and implement trigger in databases

Intended Learning Outcomes (ILOs):

The students should be able to:
2.1 Create triggers in database
2.2 Implement and execute triggers.

Output

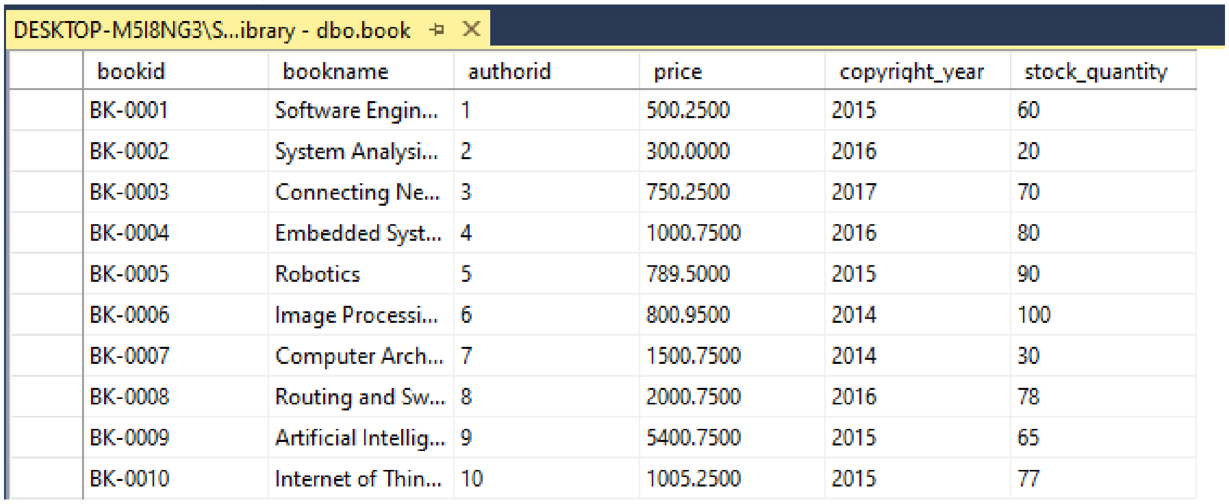
Create ABCLibrary Database
1. Create ABCLibrary Database
2. Create the following tables:



```
USE ABCLibrary
CREATE TABLE book(
    bookid char(7) PRIMARY KEY NOT NULL,
    bookname varchar(max) NOT NULL,
    authorid char(7) NOT NULL,
    price money NULL,
    copyright_year int NOT NULL,
    stock_quantity int NOT NULL)

CREATE TABLE book_price_log(
    bookid char(7) NOT NULL,
    price money NOT NULL,
    valid_date datetime NOT NULL)

CREATE TABLE book_order(
    customerid char(7) NOT NULL,
    bookid char(7) NOT NULL,
    quantity int NOT NULL,
    orderdate timestamp NOT NULL)
```

```
Commands completed successfully.

Completion time: 2022-11-28T13:48:01.7156722+08:00
```

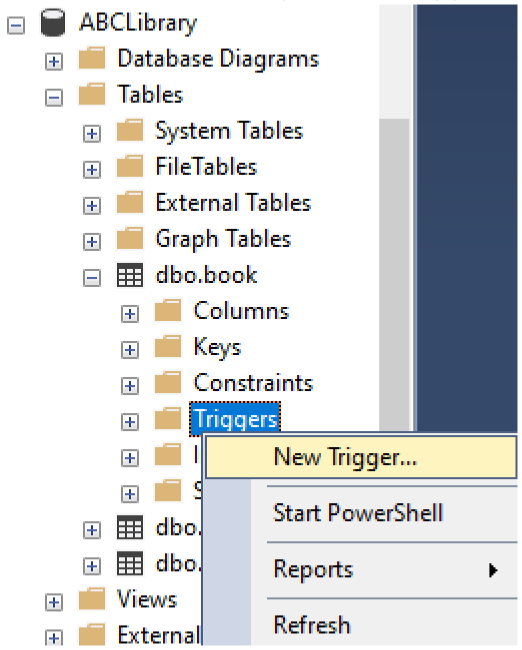
3. Insert the following data in the book table.



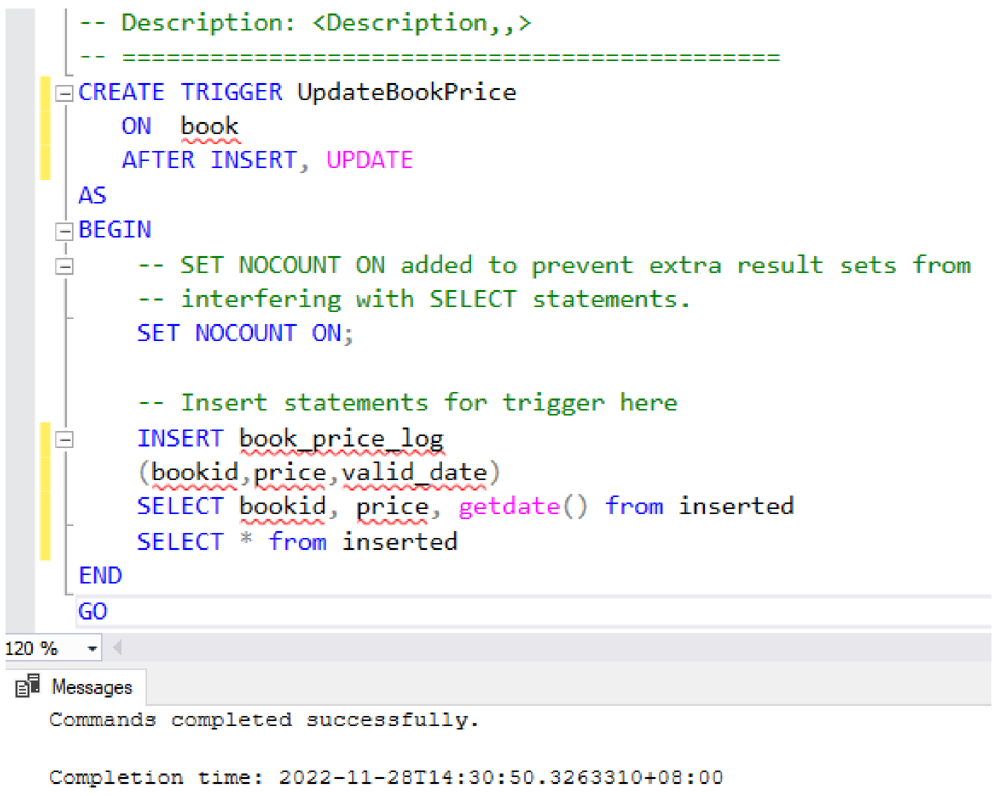
DESKTOP-M5I8NG3\S...ibrary - dbo.book

bookid	bookname	authorid	price	copyright_year	stock_quantity
BK-0001	Software Engin...	1	500.2500	2015	60
BK-0002	System Analysi...	2	300.0000	2016	20
BK-0003	Connecting Ne...	3	750.2500	2017	70
BK-0004	Embedded Syst...	4	1000.7500	2016	80
BK-0005	Robotics	5	789.5000	2015	90
BK-0006	Image Processi...	6	800.9500	2014	100
BK-0007	Computer Arch...	7	1500.7500	2014	30
BK-0008	Routing and Sw...	8	2000.7500	2016	78
BK-0009	Artificial Intellig...	9	5400.7500	2015	65
BK-0010	Internet of Thin...	10	1005.2500	2015	77

Create Data Manipulation Language (DML) Trigger
Example No. 1
Step 1: In **Object Explorer**, connect to an instance of Database Engine and then expand that instance.
Step 2: Expand **Databases**, expand the **ABCLibrary** database, and then expand **Tables**. Expand **book** table. Right-click **Trigger** and choose **New Trigger**.



Step 3: Modify the trigger using the given screenshot. Type your name as author and the creation date.



```
-- Description: <Description,,>
-- =============================================
CREATE TRIGGER UpdateBookPrice
   ON  book
   AFTER INSERT, UPDATE
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for trigger here
    INSERT book_price_log
    (bookid,price,valid_date)
    SELECT bookid, price, getdate() from inserted
    SELECT * from inserted
END
GO
```

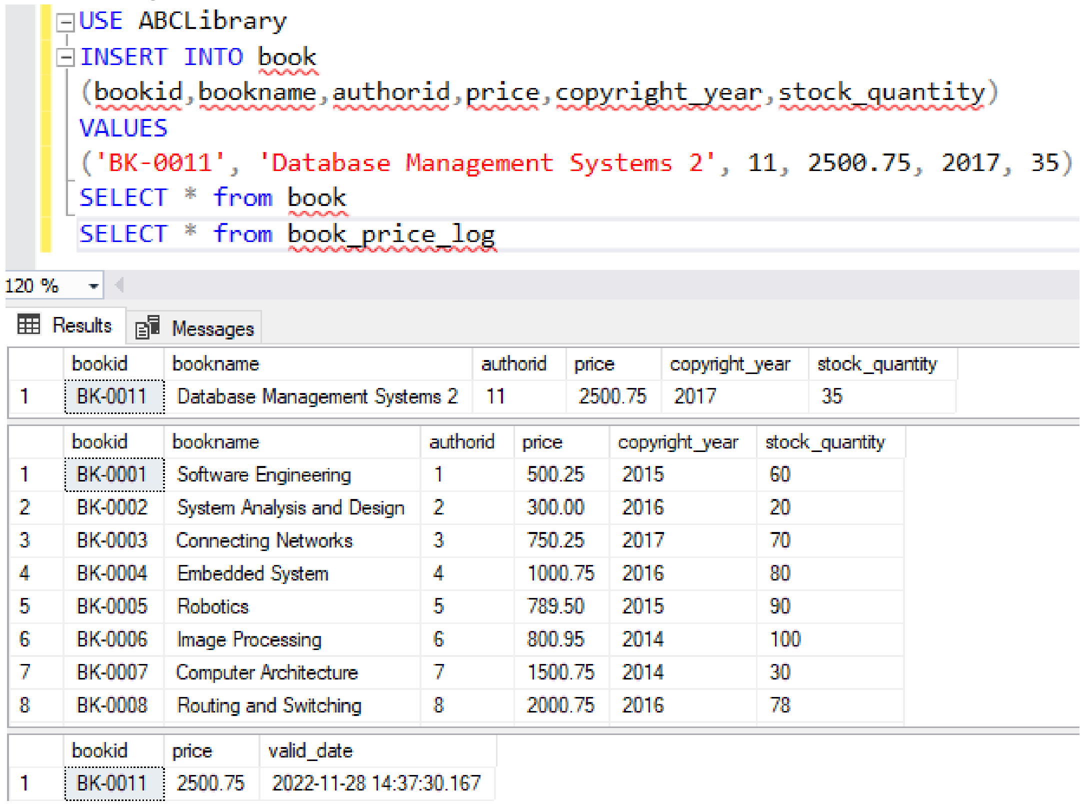
120 %

Messages

```
Commands completed successfully.

Completion time: 2022-11-28T14:30:50.3263310+08:00
```

Step 4: Click execute or press F5 to save the trigger.
Step 5: Test the UpdateBookPriceLog trigger. Create a new query and type the given statement. Click Execute.

A. Inserting Book Information



```
USE ABCLibrary
INSERT INTO book
(bookid,bookname,authorid,price,copyright_year,stock_quantity)
VALUES
('BK-0011', 'Database Management Systems 2', 11, 2500.75, 2017, 35)
SELECT * from book
SELECT * from book_price_log
```

120 %

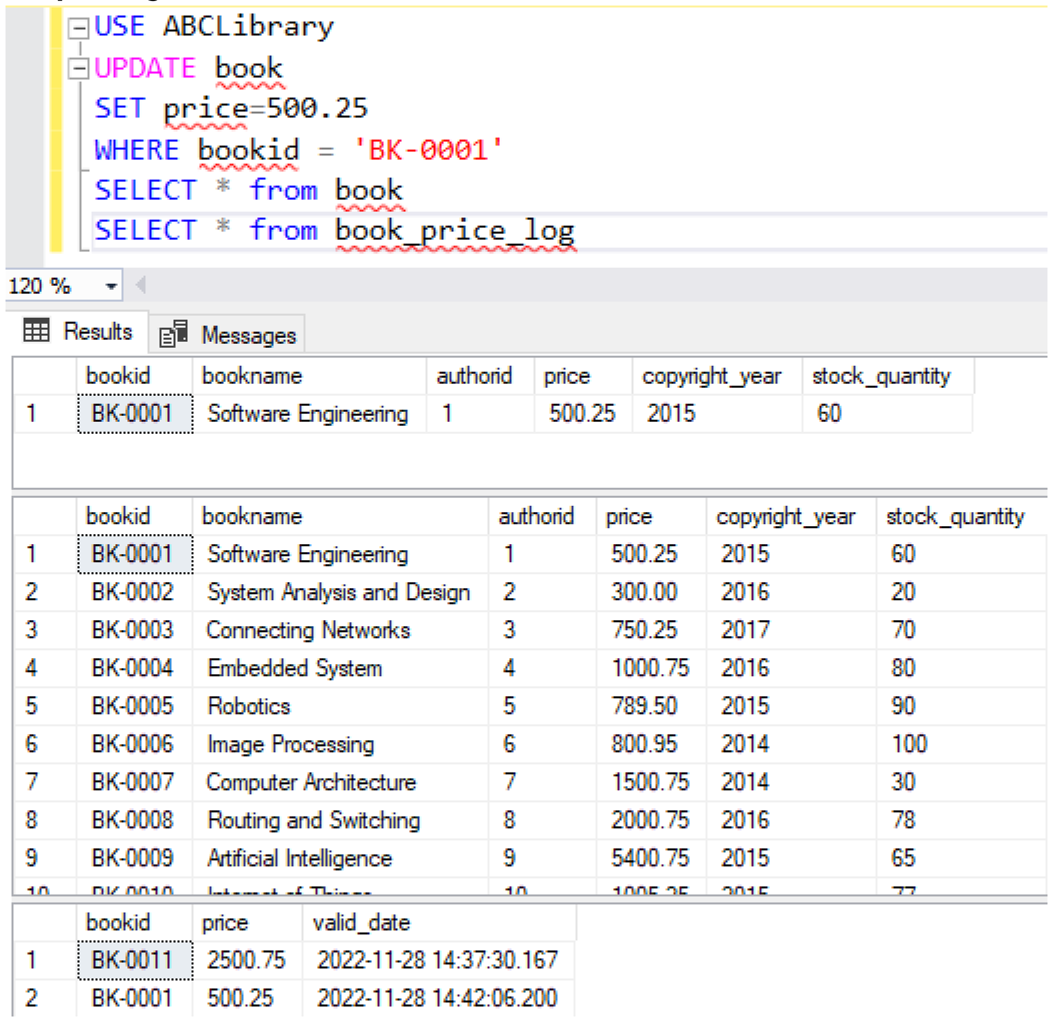
Results | Messages

	bookid	bookname	authorid	price	copyright_year	stock_quantity
1	BK-0011	Database Management Systems 2	11	2500.75	2017	35

	bookid	bookname	authorid	price	copyright_year	stock_quantity
1	BK-0001	Software Engineering	1	500.25	2015	60
2	BK-0002	System Analysis and Design	2	300.00	2016	20
3	BK-0003	Connecting Networks	3	750.25	2017	70
4	BK-0004	Embedded System	4	1000.75	2016	80
5	BK-0005	Robotics	5	789.50	2015	90
6	BK-0006	Image Processing	6	800.95	2014	100
7	BK-0007	Computer Architecture	7	1500.75	2014	30
8	BK-0008	Routing and Switching	8	2000.75	2016	78

	bookid	price	valid_date
1	BK-0011	2500.75	2022-11-28 14:37:30.167

B. Updating Book Information



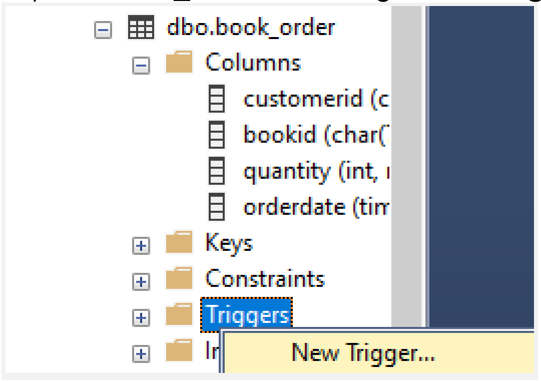
```
USE ABCLibrary
UPDATE book
SET price=500.25
WHERE bookid = 'BK-0001'
SELECT * from book
SELECT * from book_price_log
```

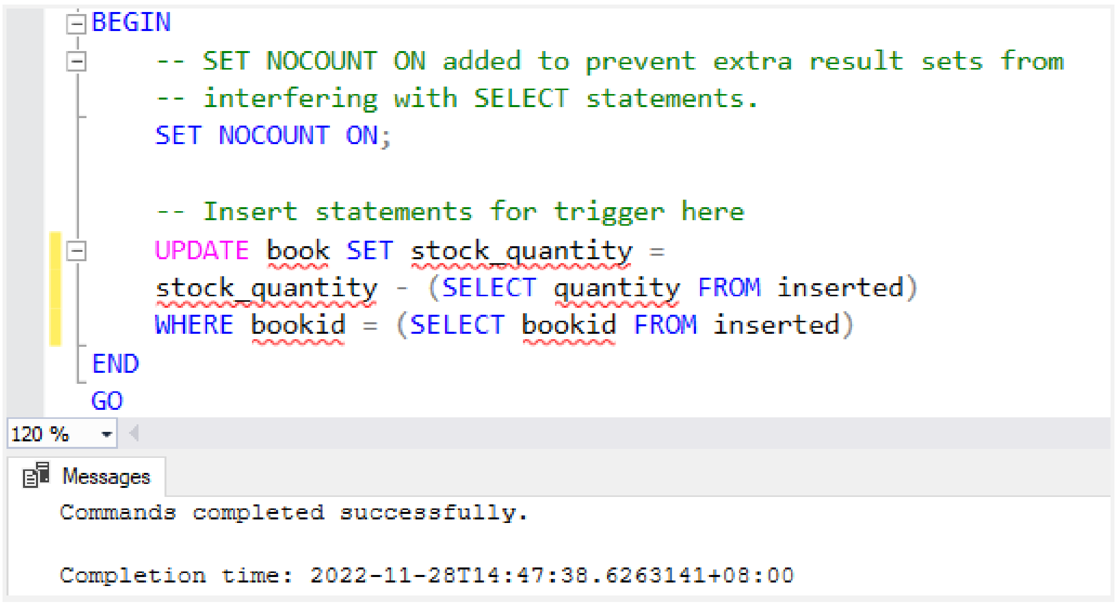
	bookid	bookname	authorid	price	copyright_year	stock_quantity
1	BK-0001	Software Engineering	1	500.25	2015	60

	bookid	bookname	authorid	price	copyright_year	stock_quantity
1	BK-0001	Software Engineering	1	500.25	2015	60
2	BK-0002	System Analysis and Design	2	300.00	2016	20
3	BK-0003	Connecting Networks	3	750.25	2017	70
4	BK-0004	Embedded System	4	1000.75	2016	80
5	BK-0005	Robotics	5	789.50	2015	90
6	BK-0006	Image Processing	6	800.95	2014	100
7	BK-0007	Computer Architecture	7	1500.75	2014	30
8	BK-0008	Routing and Switching	8	2000.75	2016	78
9	BK-0009	Artificial Intelligence	9	5400.75	2015	65

	bookid	price	valid_date
1	BK-0011	2500.75	2022-11-28 14:37:30.167
2	BK-0001	500.25	2022-11-28 14:42:06.200

Example No. 2
Step 1: In **Object Explorer**, connect to an instance of Database Engine and then expand that instance.
Step 2: Expand **Databases**, expand the **ABCLibrary** database, and then expand **Tables**. Expand **book_order** table. Right-click **Trigger** and choose **New Trigger**.

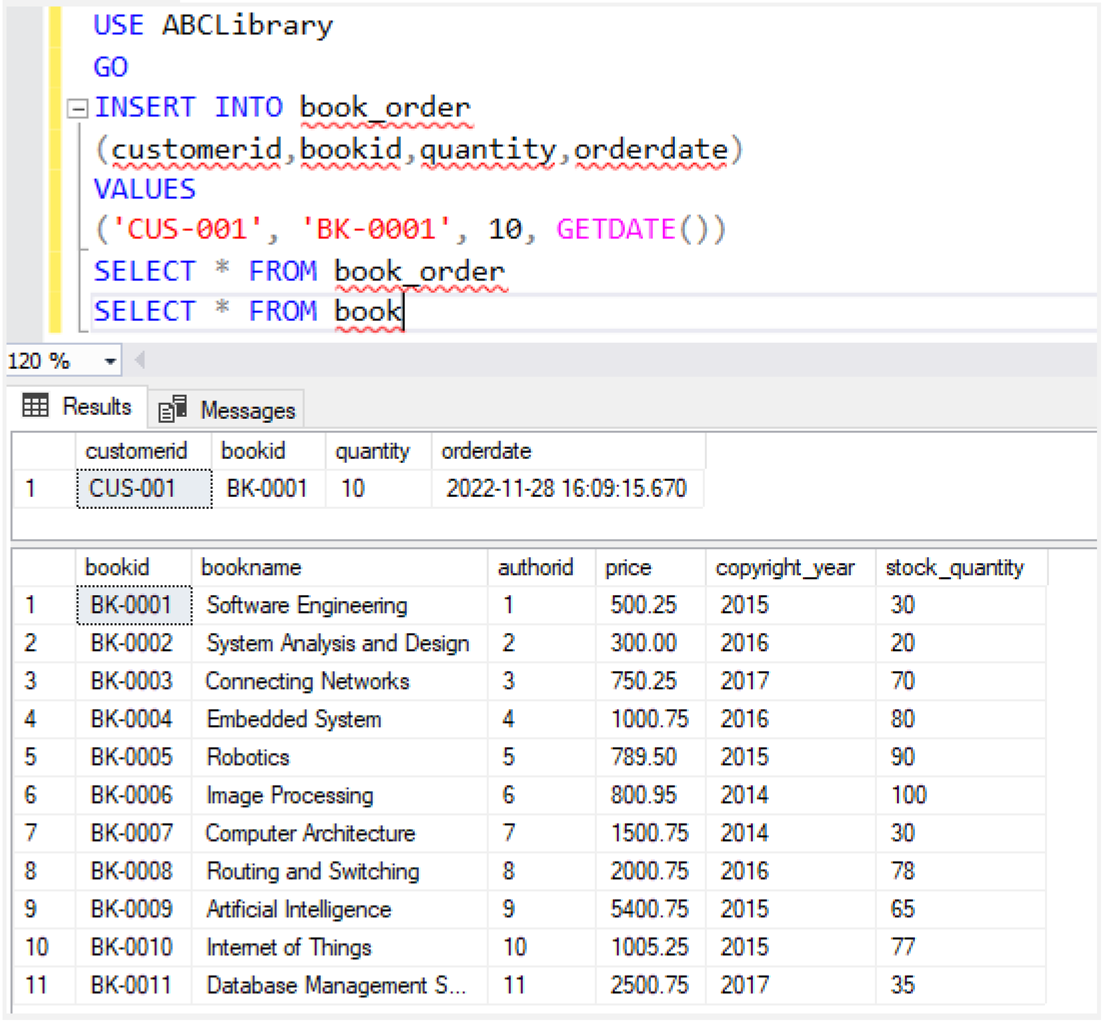




```
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for trigger here
    UPDATE book SET stock_quantity =
    stock_quantity - (SELECT quantity FROM inserted)
    WHERE bookid = (SELECT bookid FROM inserted)
END
GO
```

```
Commands completed successfully.

Completion time: 2022-11-28T14:47:38.6263141+08:00
```

Step 4: Click execute or press F5 to save the trigger.
Step 5: Test the UpdateBookStock trigger. Create a new query and type the given statement. Click Execute.

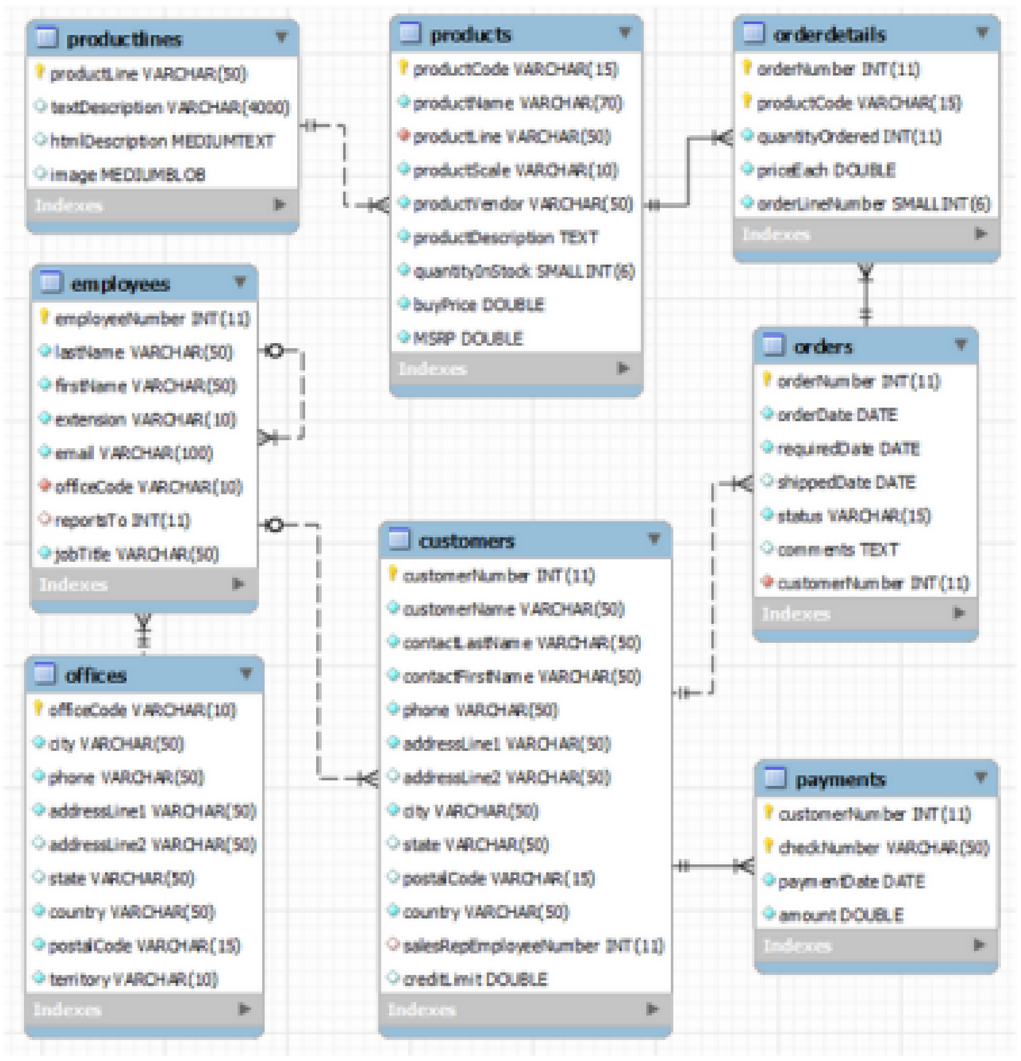


```
USE ABCLibrary
GO
INSERT INTO book_order
(customerid,bookid,quantity,orderdate)
VALUES
('CUS-001', 'BK-0001', 10, GETDATE())
SELECT * FROM book_order
SELECT * FROM book
```

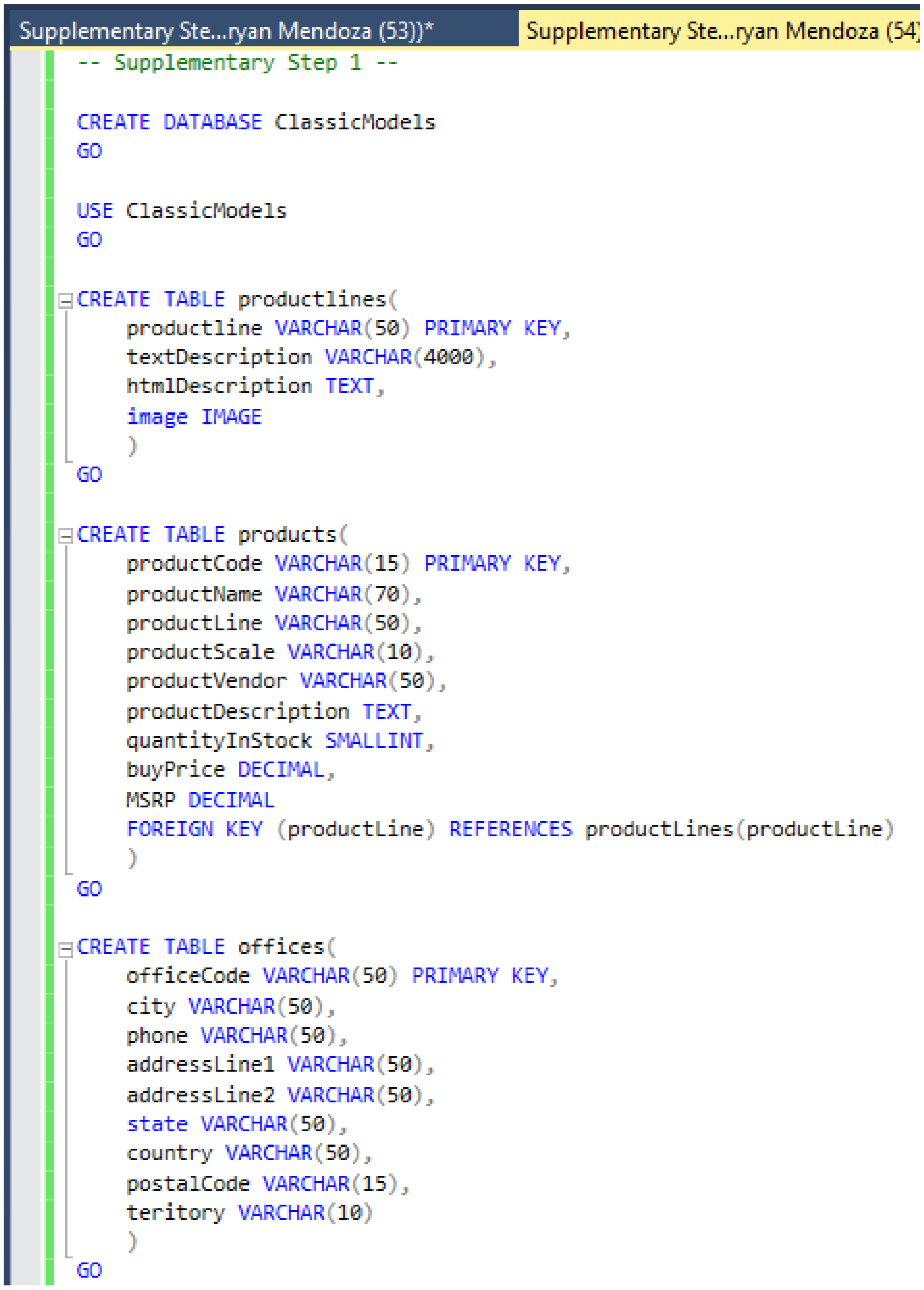
	customerid	bookid	quantity	orderdate
1	CUS-001	BK-0001	10	2022-11-28 16:09:15.670

	bookid	bookname	authorid	price	copyright_year	stock_quantity
1	BK-0001	Software Engineering	1	500.25	2015	30
2	BK-0002	System Analysis and Design	2	300.00	2016	20
3	BK-0003	Connecting Networks	3	750.25	2017	70
4	BK-0004	Embedded System	4	1000.75	2016	80
5	BK-0005	Robotics	5	789.50	2015	90
6	BK-0006	Image Processing	6	800.95	2014	100
7	BK-0007	Computer Architecture	7	1500.75	2014	30
8	BK-0008	Routing and Switching	8	2000.75	2016	78
9	BK-0009	Artificial Intelligence	9	5400.75	2015	65
10	BK-0010	Internet of Things	10	1005.25	2015	77
11	BK-0011	Database Management S...	11	2500.75	2017	35

Supplementary Activity

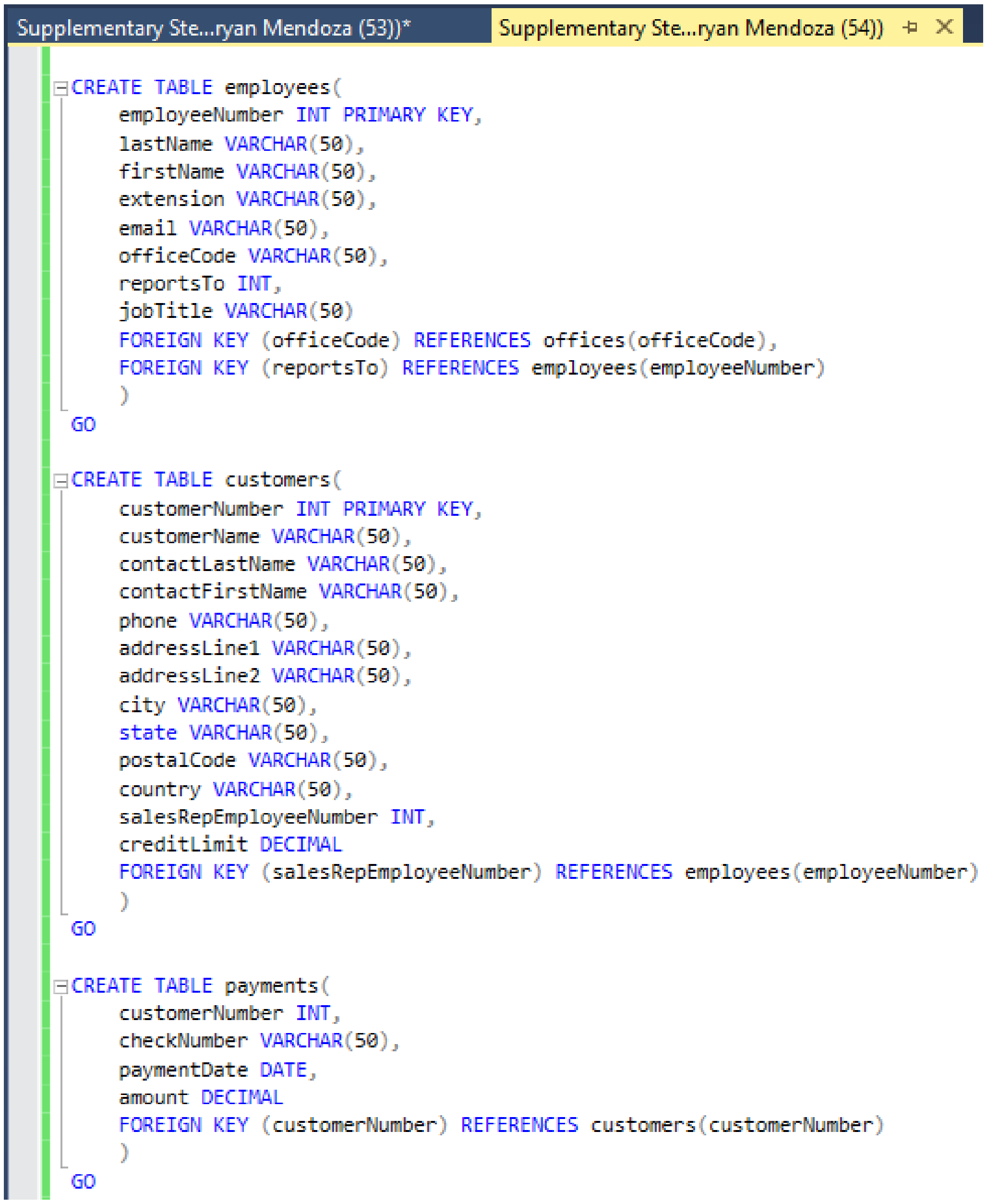


a. Use the given ERD to create the ClassicModels database and tables. Assign the appropriate data type.


Supplementary Ste...ryan Mendoza (53))*
Supplementary Ste...ryan Mendoza (54

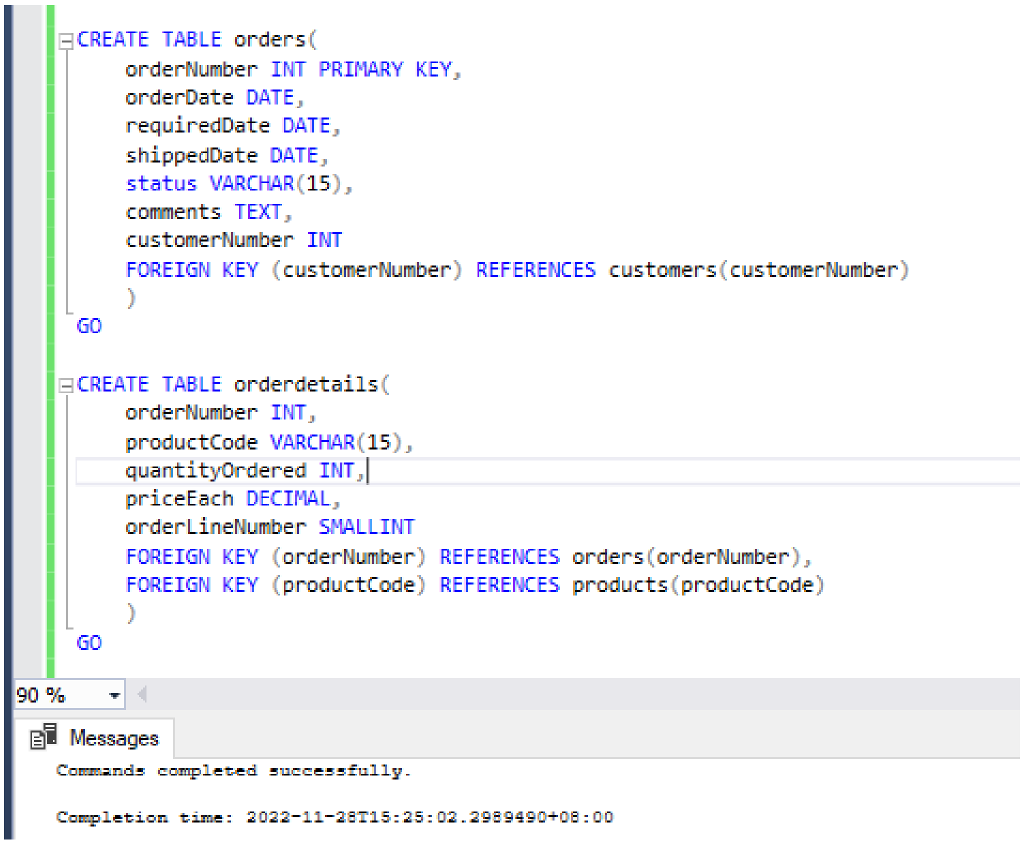
```sql
-- Supplementary Step 1 --

CREATE DATABASE ClassicModels
GO

USE ClassicModels
GO

CREATE TABLE productlines(
    productline VARCHAR(50) PRIMARY KEY,
    textDescription VARCHAR(4000),
    htmlDescription TEXT,
    image IMAGE
    )
GO

CREATE TABLE products(
    productCode VARCHAR(15) PRIMARY KEY,
    productName VARCHAR(70),
    productLine VARCHAR(50),
    productScale VARCHAR(10),
    productVendor VARCHAR(50),
    productDescription TEXT,
    quantityInStock SMALLINT,
    buyPrice DECIMAL,
    MSRP DECIMAL
    FOREIGN KEY (productLine) REFERENCES productLines(productLine)
    )
GO

CREATE TABLE offices(
    officeCode VARCHAR(50) PRIMARY KEY,
    city VARCHAR(50),
    phone VARCHAR(50),
    addressLine1 VARCHAR(50),
    addressLine2 VARCHAR(50),
    state VARCHAR(50),
    country VARCHAR(50),
    postalCode VARCHAR(15),
    teritory VARCHAR(10)
    )
GO
```

Continuation


Supplementary Ste...ryan Mendoza (53))*
Supplementary Ste...ryan Mendoza (54))

```
CREATE TABLE employees(
    employeeNumber INT PRIMARY KEY,
    lastName VARCHAR(50),
    firstName VARCHAR(50),
    extension VARCHAR(50),
    email VARCHAR(50),
    officeCode VARCHAR(50),
    reportsTo INT,
    jobTitle VARCHAR(50)
    FOREIGN KEY (officeCode) REFERENCES offices(officeCode),
    FOREIGN KEY (reportsTo) REFERENCES employees(employeeNumber)
    )
GO

CREATE TABLE customers(
    customerNumber INT PRIMARY KEY,
    customerName VARCHAR(50),
    contactLastName VARCHAR(50),
    contactFirstName VARCHAR(50),
    phone VARCHAR(50),
    addressLine1 VARCHAR(50),
    addressLine2 VARCHAR(50),
    city VARCHAR(50),
    state VARCHAR(50),
    postalCode VARCHAR(50),
    country VARCHAR(50),
    salesRepEmployeeNumber INT,
    creditLimit DECIMAL
    FOREIGN KEY (salesRepEmployeeNumber) REFERENCES employees(employeeNumber)
    )
GO

CREATE TABLE payments(
    customerNumber INT,
    checkNumber VARCHAR(50),
    paymentDate DATE,
    amount DECIMAL
    FOREIGN KEY (customerNumber) REFERENCES customers(customerNumber)
    )
GO
```

Continuation

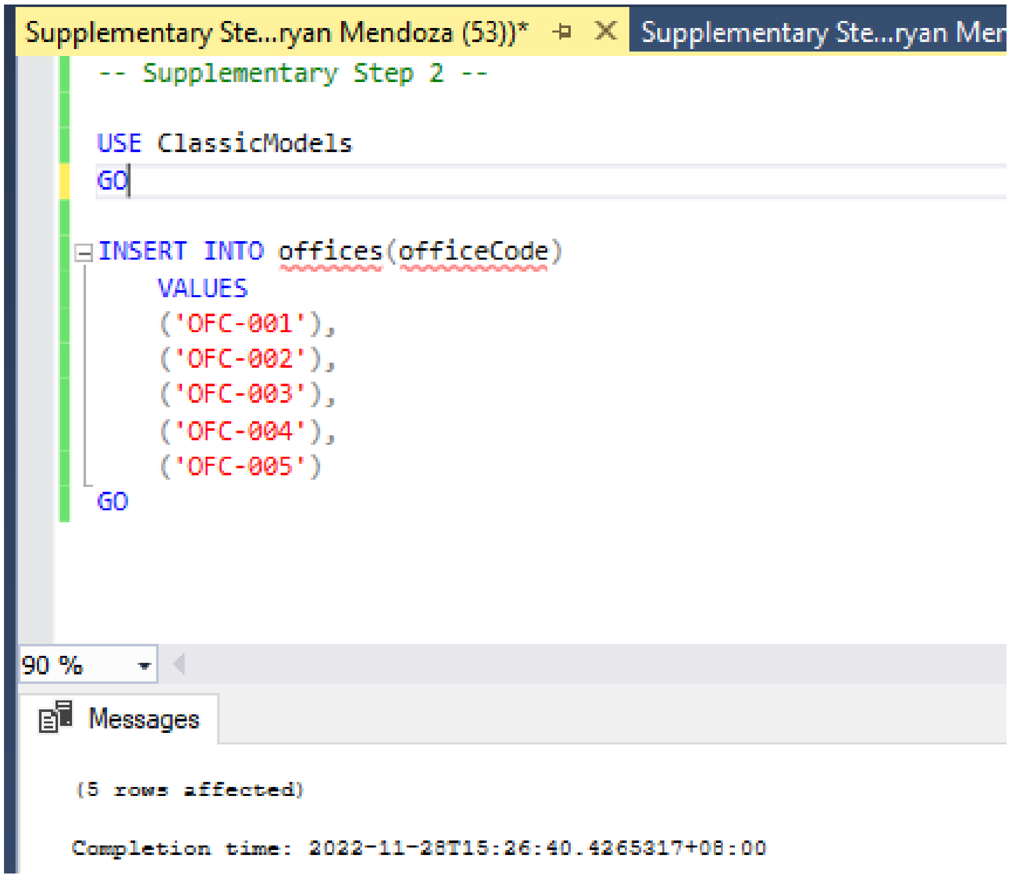


```
CREATE TABLE orders(
    orderNumber INT PRIMARY KEY,
    orderDate DATE,
    requiredDate DATE,
    shippedDate DATE,
    status VARCHAR(15),
    comments TEXT,
    customerNumber INT
    FOREIGN KEY (customerNumber) REFERENCES customers(customerNumber)
    )
GO

CREATE TABLE orderdetails(
    orderNumber INT,
    productCode VARCHAR(15),
    quantityOrdered INT,
    priceEach DECIMAL,
    orderLineNumber SMALLINT
    FOREIGN KEY (orderNumber) REFERENCES orders(orderNumber),
    FOREIGN KEY (productCode) REFERENCES products(productCode)
    )
GO
```

```
Commands completed successfully.

Completion time: 2022-11-28T15:25:02.2989490+08:00
```

Observation:
In order to construct the database, we used our understanding based from the previous discussions in order to follow the relationships present with each entity.

b. Insert five (5) Office records where the Office code starts with OFC-001.


Supplementary Ste...ryan Mendoza (53))*
Supplementary Ste...ryan Mer
90 %
Messages

```
-- Supplementary Step 2 --

USE ClassicModels
GO

INSERT INTO offices(officeCode)
    VALUES
    ('OFC-001'),
    ('OFC-002'),
    ('OFC-003'),
    ('OFC-004'),
    ('OFC-005')
GO
```

```
(5 rows affected)

Completion time: 2022-11-28T15:26:40.4265317+08:00
```

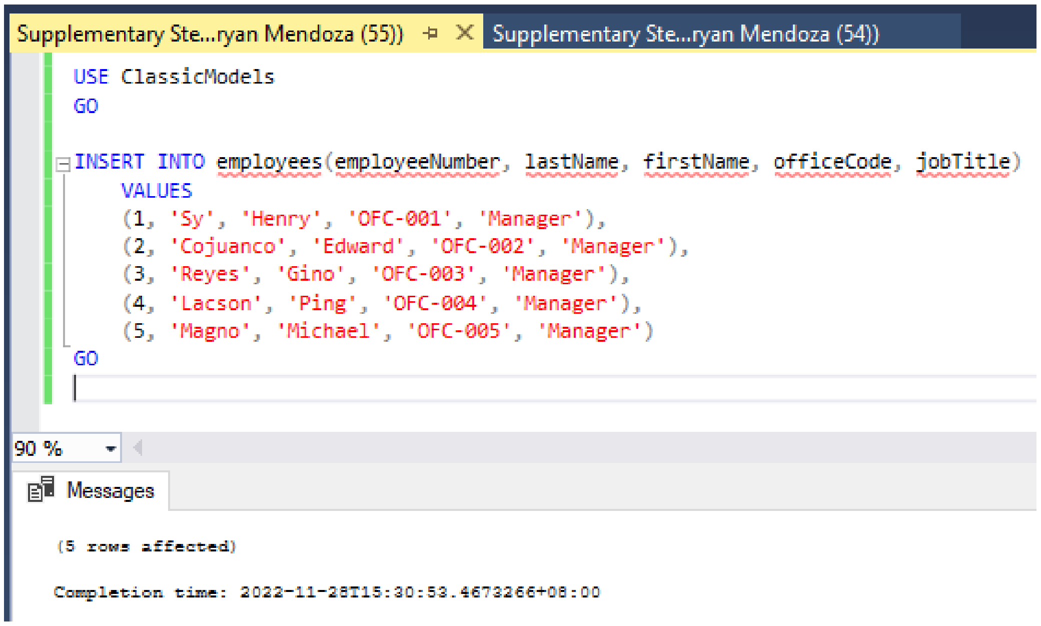
Observation:
To initialize, we used a normal query in order to insert entries to the offices table. The other columns were left empty as the only required column for this procedure is the officeCode column only.

c. Insert five(5) employee records (5) with managerial position. Use Manager as Job Title

Given:

Employee Number	Lastname	Firstname	Office Code
1	Sy	Henry	OFC-001
2	Cojuangco	Edward	OFC-002
3	Reyes	Gino	OFC-003
4	Lacson	Ping	OFC-004
5	Magno	Michael	OFC-005

Insertion to the Database:


Supplementary Ste...ryan Mendoza (55))
Supplementary Ste...ryan Mendoza (54))
90 %
Messages

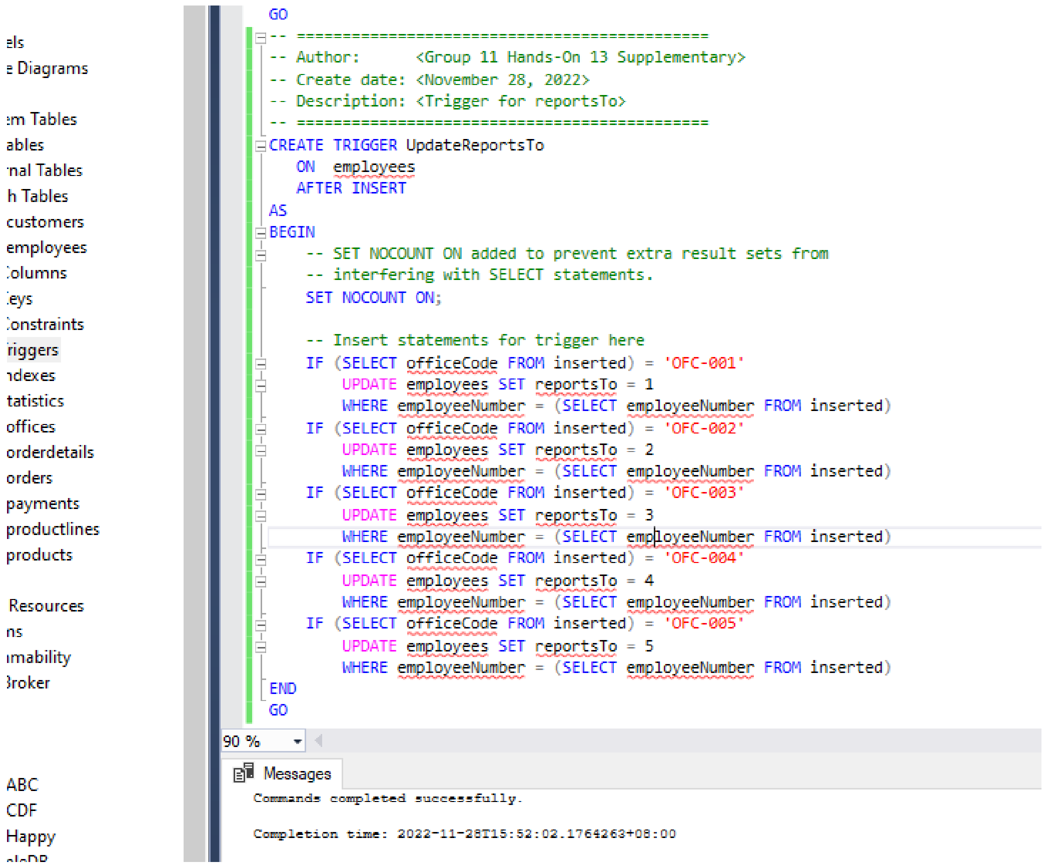
```
USE ClassicModels
GO

INSERT INTO employees(employeeNumber, lastName, firstName, officeCode, jobTitle)
    VALUES
    (1, 'Sy', 'Henry', 'OFC-001', 'Manager'),
    (2, 'Cojuanco', 'Edward', 'OFC-002', 'Manager'),
    (3, 'Reyes', 'Gino', 'OFC-003', 'Manager'),
    (4, 'Lacson', 'Ping', 'OFC-004', 'Manager'),
    (5, 'Magno', 'Michael', 'OFC-005', 'Manager')
GO
```

```
(5 rows affected)

Completion time: 2022-11-28T15:30:53.4673266+08:00
```

Observation:
Similarly, we inserted entries on the employees table by using the given 5 entries from the laboratory procedure. We used a normal query in order to insert them.

d. Create a trigger that automatically updates the reportsTo field of the employee depending on the OFFICE CODE. Insert 2 records for each office code.

Creation of Trigger



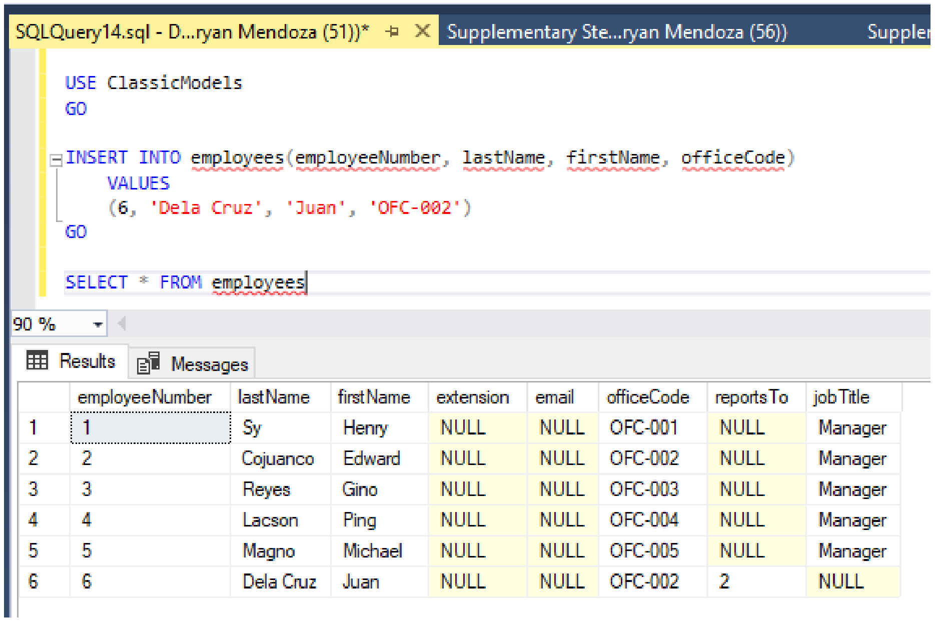
```
GO
-- =============================================
-- Author:      <Group 11 Hands-On 13 Supplementary>
-- Create date: <November 28, 2022>
-- Description: <Trigger for reportsTo>
-- =============================================
CREATE TRIGGER UpdateReportsTo
   ON  employees
   AFTER INSERT
AS
BEGIN
   -- SET NOCOUNT ON added to prevent extra result sets from
   -- interfering with SELECT statements.
   SET NOCOUNT ON;

   -- Insert statements for trigger here
   IF (SELECT officeCode FROM inserted) = 'OFC-001'
      UPDATE employees SET reportsTo = 1
      WHERE employeeNumber = (SELECT employeeNumber FROM inserted)
   IF (SELECT officeCode FROM inserted) = 'OFC-002'
      UPDATE employees SET reportsTo = 2
      WHERE employeeNumber = (SELECT employeeNumber FROM inserted)
   IF (SELECT officeCode FROM inserted) = 'OFC-003'
      UPDATE employees SET reportsTo = 3
      WHERE employeeNumber = (SELECT employeeNumber FROM inserted)
   IF (SELECT officeCode FROM inserted) = 'OFC-004'
      UPDATE employees SET reportsTo = 4
      WHERE employeeNumber = (SELECT employeeNumber FROM inserted)
   IF (SELECT officeCode FROM inserted) = 'OFC-005'
      UPDATE employees SET reportsTo = 5
      WHERE employeeNumber = (SELECT employeeNumber FROM inserted)
END
GO
```

```
Commands completed successfully.

Completion time: 2022-11-28T15:52:02.1764263+08:00
```

Observation:
We created a trigger on the employees table in order to automatically update the reportsTo field of an entry depending on their assigned officeCode. We used an If-Else Statement in order to determine which condition matches the officeCode of an entry.

Calling the Trigger: Auto Updating reportsTo given an officeCode

Test Value 1



SQLQuery14.sql - D...ryan Mendoza (51))* | Supplementary Ste...ryan Mendoza (56)) | Suppler

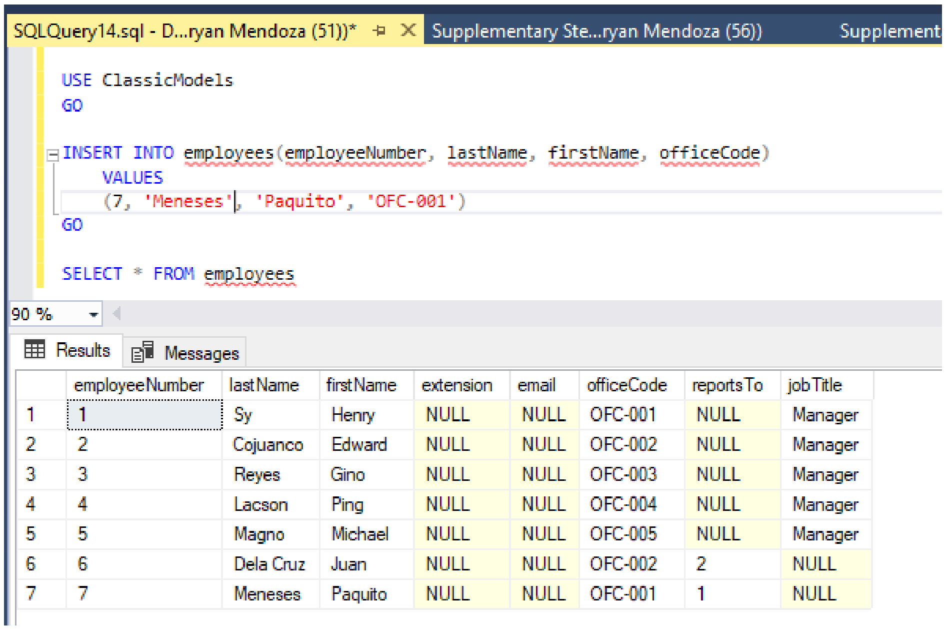
```
USE ClassicModels
GO

INSERT INTO employees(employeeNumber, lastName, firstName, officeCode)
    VALUES
    (6, 'Dela Cruz', 'Juan', 'OFC-002')
GO

SELECT * FROM employees
```

90 %

Results | Messages

	employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1	1	Sy	Henry	NULL	NULL	OFC-001	NULL	Manager
2	2	Cojuanco	Edward	NULL	NULL	OFC-002	NULL	Manager
3	3	Reyes	Gino	NULL	NULL	OFC-003	NULL	Manager
4	4	Lacson	Ping	NULL	NULL	OFC-004	NULL	Manager
5	5	Magno	Michael	NULL	NULL	OFC-005	NULL	Manager
6	6	Dela Cruz	Juan	NULL	NULL	OFC-002	2	NULL

Test Value 2



SQLQuery14.sql - D...ryan Mendoza (51))* | Supplementary Ste...ryan Mendoza (56)) | Supplement

```
USE ClassicModels
GO

INSERT INTO employees(employeeNumber, lastName, firstName, officeCode)
    VALUES
    (7, 'Meneses', 'Paquito', 'OFC-001')
GO

SELECT * FROM employees
```

90 %

Results | Messages

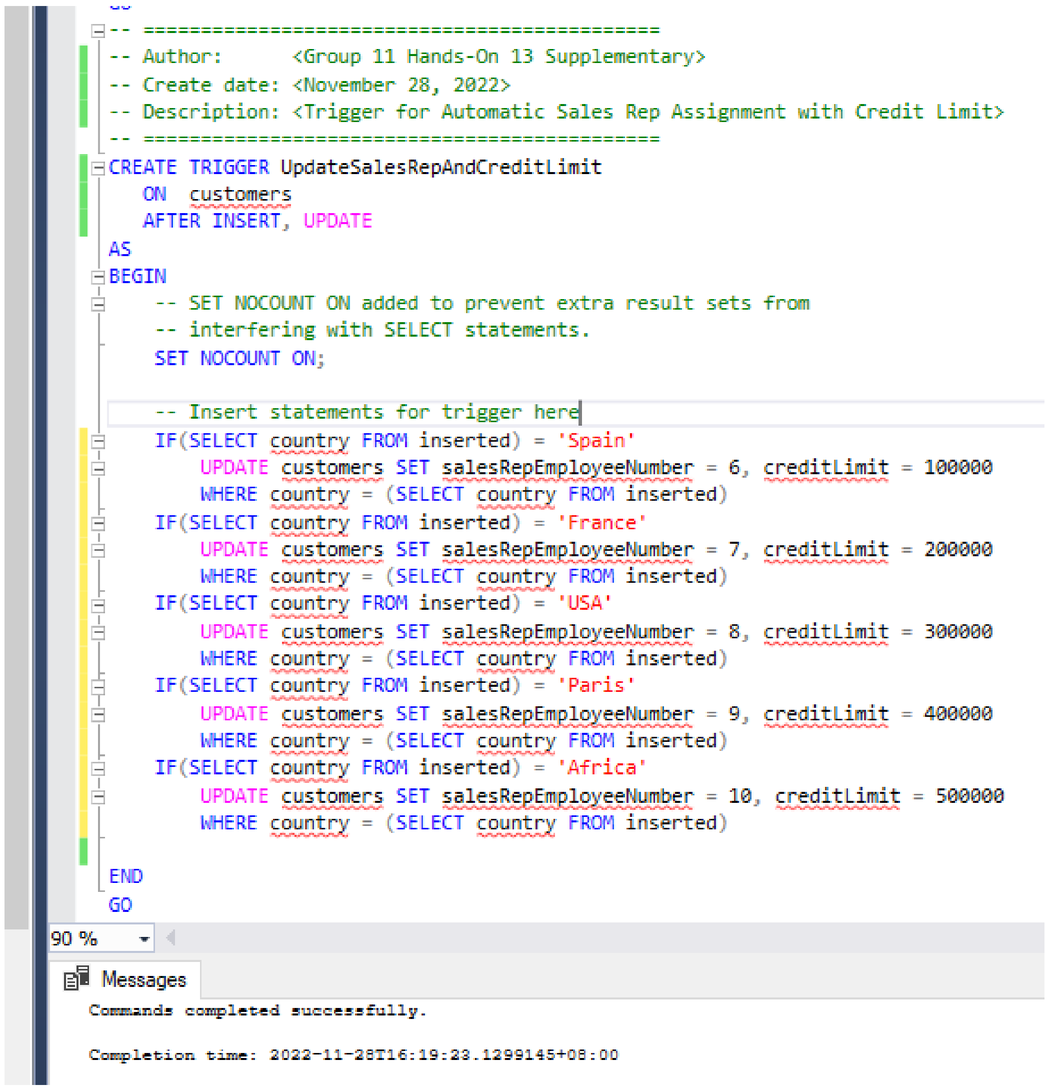
	employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1	1	Sy	Henry	NULL	NULL	OFC-001	NULL	Manager
2	2	Cojuanco	Edward	NULL	NULL	OFC-002	NULL	Manager
3	3	Reyes	Gino	NULL	NULL	OFC-003	NULL	Manager
4	4	Lacson	Ping	NULL	NULL	OFC-004	NULL	Manager
5	5	Magno	Michael	NULL	NULL	OFC-005	NULL	Manager
6	6	Dela Cruz	Juan	NULL	NULL	OFC-002	2	NULL
7	7	Meneses	Paquito	NULL	NULL	OFC-001	1	NULL

Observation:

Since we have an if else statement, the multiple insertion was not applicable, and as we can observe the trigger was successful since it updates the newly inserted employees' reportsTo column based on their office code.

5. Create a trigger that automatically updates the Sales Rep Employee and Credit Limit field of the customer table depending on the country. Insert 2 records for each customer per country. Update the last 2 records to change the country to Spain and France respectively.

Creation of the trigger



```
-- =============================================
-- Author:      <Group 11 Hands-On 13 Supplementary>
-- Create date: <November 28, 2022>
-- Description: <Trigger for Automatic Sales Rep Assignment with Credit Limit>
-- =============================================
CREATE TRIGGER UpdateSalesRepAndCreditLimit
   ON  customers
   AFTER INSERT, UPDATE
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for trigger here
    IF(SELECT country FROM inserted) = 'Spain'
        UPDATE customers SET salesRepEmployeeNumber = 6, creditLimit = 100000
        WHERE country = (SELECT country FROM inserted)
    IF(SELECT country FROM inserted) = 'France'
        UPDATE customers SET salesRepEmployeeNumber = 7, creditLimit = 200000
        WHERE country = (SELECT country FROM inserted)
    IF(SELECT country FROM inserted) = 'USA'
        UPDATE customers SET salesRepEmployeeNumber = 8, creditLimit = 300000
        WHERE country = (SELECT country FROM inserted)
    IF(SELECT country FROM inserted) = 'Paris'
        UPDATE customers SET salesRepEmployeeNumber = 9, creditLimit = 400000
        WHERE country = (SELECT country FROM inserted)
    IF(SELECT country FROM inserted) = 'Africa'
        UPDATE customers SET salesRepEmployeeNumber = 10, creditLimit = 500000
        WHERE country = (SELECT country FROM inserted)

END
GO
```

90 %

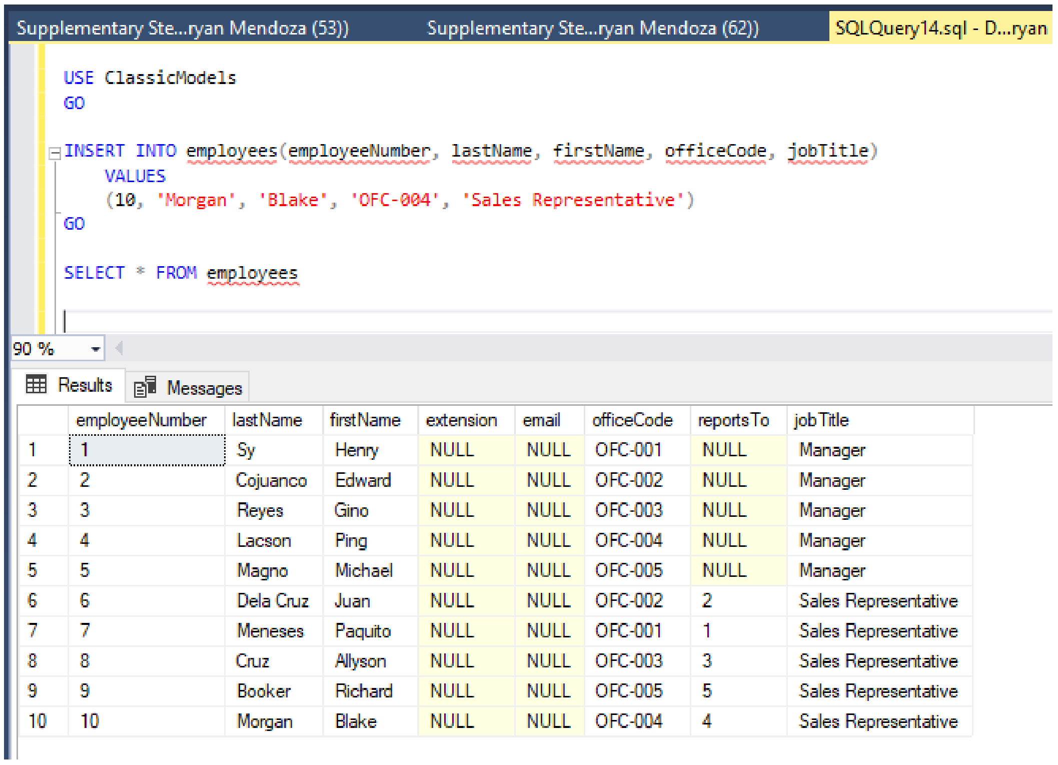
Messages

```
Commands completed successfully.

Completion time: 2022-11-28T16:19:23.1299145+08:00
```

Given:

Country	Employee Number	Credit Limit
Spain	6	100000
France	7	200000
USA	8	300000
Paris	9	400000
Africa	10	500000

Adding More Sales Rep using previous trigger to perform the procedure

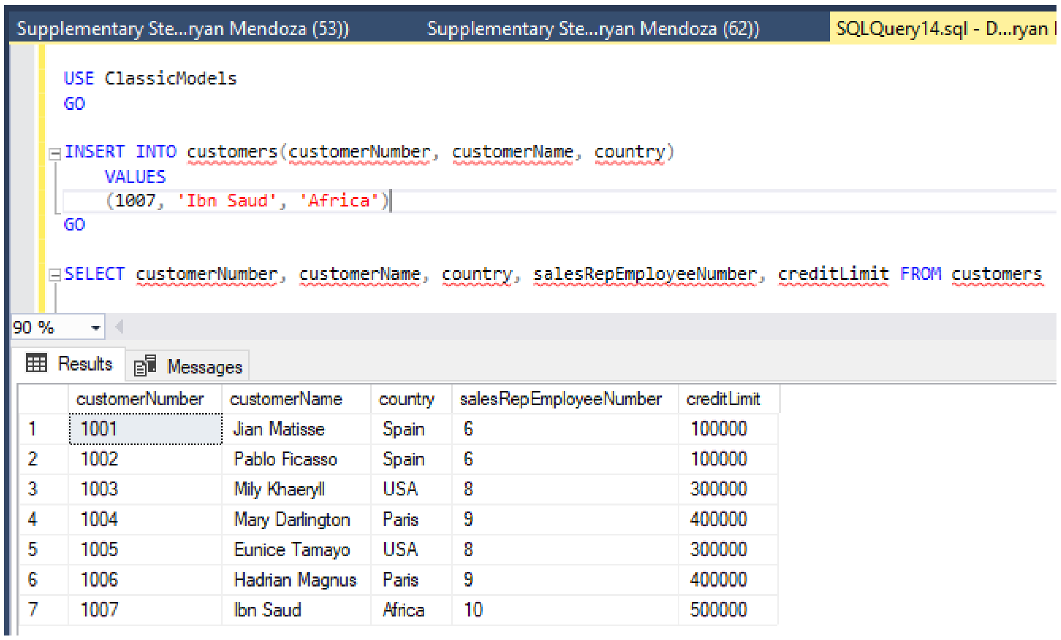

Supplementary Ste...ryan Mendoza (53))
Supplementary Ste...ryan Mendoza (62))
SQLQuery14.sql - D...ryan
90 %
Results
Messages

```
USE ClassicModels
GO

INSERT INTO employees(employeeNumber, lastName, firstName, officeCode, jobTitle)
    VALUES
    (10, 'Morgan', 'Blake', 'OFC-004', 'Sales Representative')
GO

SELECT * FROM employees
```

	employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1	1	Sy	Henry	NULL	NULL	OFC-001	NULL	Manager
2	2	Cojuanco	Edward	NULL	NULL	OFC-002	NULL	Manager
3	3	Reyes	Gino	NULL	NULL	OFC-003	NULL	Manager
4	4	Lacson	Ping	NULL	NULL	OFC-004	NULL	Manager
5	5	Magno	Michael	NULL	NULL	OFC-005	NULL	Manager
6	6	Dela Cruz	Juan	NULL	NULL	OFC-002	2	Sales Representative
7	7	Meneses	Paquito	NULL	NULL	OFC-001	1	Sales Representative
8	8	Cruz	Allyson	NULL	NULL	OFC-003	3	Sales Representative
9	9	Booker	Richard	NULL	NULL	OFC-005	5	Sales Representative
10	10	Morgan	Blake	NULL	NULL	OFC-004	4	Sales Representative

Observation:
We used the trigger created on the letter d procedure in order to add more employees on the employees table, these employees will be used on this new trigger for procedure 5.

Trigger Execution: Sales Rep Trigger



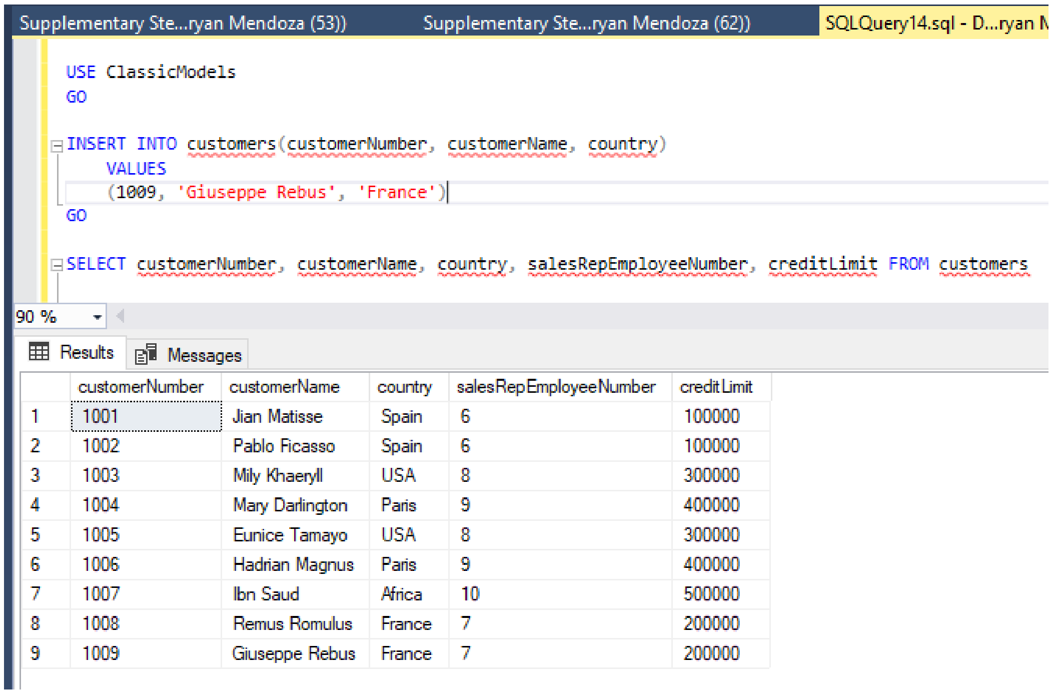
Supplementary Ste...ryan Mendoza (53)) Supplementary Ste...ryan Mendoza (62)) SQLQuery14.sql - D...ryan

```
USE ClassicModels
GO

INSERT INTO customers(customerNumber, customerName, country)
    VALUES
    (1007, 'Ibn Saud', 'Africa')
GO

SELECT customerNumber, customerName, country, salesRepEmployeeNumber, creditLimit FROM customers
```

90 %

Results Messages

	customerNumber	customerName	country	salesRepEmployeeNumber	creditLimit
1	1001	Jian Matisse	Spain	6	100000
2	1002	Pablo Ficasso	Spain	6	100000
3	1003	Mily Khaeryll	USA	8	300000
4	1004	Mary Darlington	Paris	9	400000
5	1005	Eunice Tamayo	USA	8	300000
6	1006	Hadrian Magnus	Paris	9	400000
7	1007	Ibn Saud	Africa	10	500000



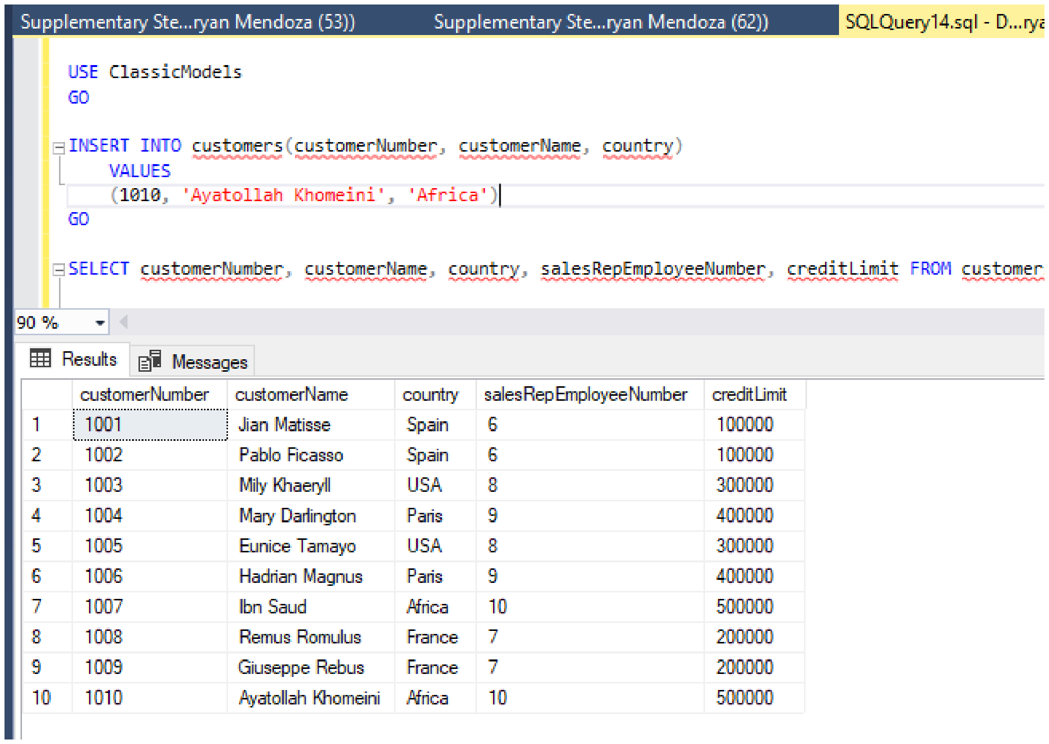
Supplementary Ste...ryan Mendoza (53)) Supplementary Ste...ryan Mendoza (62)) SQLQuery14.sql - D...ryan M

```
USE ClassicModels
GO

INSERT INTO customers(customerNumber, customerName, country)
    VALUES
    (1009, 'Giuseppe Rebus', 'France')
GO

SELECT customerNumber, customerName, country, salesRepEmployeeNumber, creditLimit FROM customers
```

90 %

Results Messages

	customerNumber	customerName	country	salesRepEmployeeNumber	creditLimit
1	1001	Jian Matisse	Spain	6	100000
2	1002	Pablo Ficasso	Spain	6	100000
3	1003	Mily Khaeryll	USA	8	300000
4	1004	Mary Darlington	Paris	9	400000
5	1005	Eunice Tamayo	USA	8	300000
6	1006	Hadrian Magnus	Paris	9	400000
7	1007	Ibn Saud	Africa	10	500000
8	1008	Remus Romulus	France	7	200000
9	1009	Giuseppe Rebus	France	7	200000


Supplementary Ste...ryan Mendoza (53))
Supplementary Ste...ryan Mendoza (62))
SQLQuery14.sql - D...rya
90 %
Results
Messages

```
USE ClassicModels
GO

INSERT INTO customers(customerNumber, customerName, country)
    VALUES
    (1010, 'Ayatollah Khomeini', 'Africa')
GO

SELECT customerNumber, customerName, country, salesRepEmployeeNumber, creditLimit FROM customer
```

	customerNumber	customerName	country	salesRepEmployeeNumber	creditLimit
1	1001	Jian Matisse	Spain	6	100000
2	1002	Pablo Ficasso	Spain	6	100000
3	1003	Mily Khaeryll	USA	8	300000
4	1004	Mary Darlington	Paris	9	400000
5	1005	Eunice Tamayo	USA	8	300000
6	1006	Hadrian Magnus	Paris	9	400000
7	1007	Ibn Saud	Africa	10	500000
8	1008	Remus Romulus	France	7	200000
9	1009	Giuseppe Rebus	France	7	200000
10	1010	Ayatollah Khomeini	Africa	10	500000

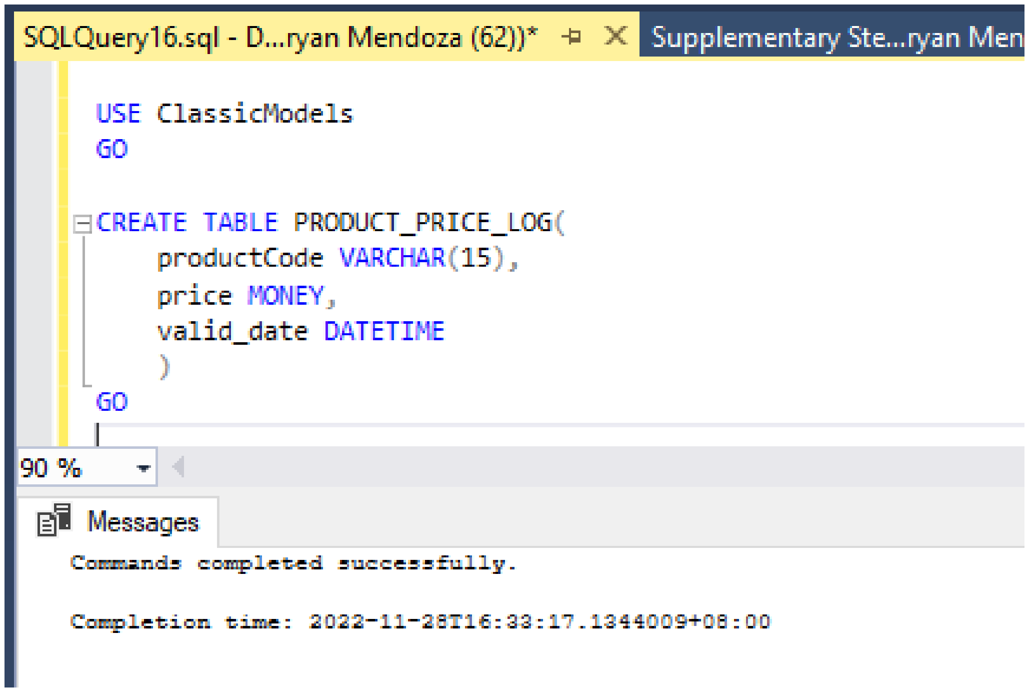
Observation:

Using the given table, we used these data to assign each employee from 6 to 10, as a Sales Representative for each region. As we can observe on the output, we added 10 customers with their name and location. We can see that a sales Representative is automatically assigned to each customer based on their country.

6. Create a table PRODUCT_PRICE_LOG.

productCode varchar(15)
price money
valid_date datetime

Creation of Table



SQLQuery16.sql - D...ryan Mendoza (62))*
Supplementary Ste...ryan Men
90 %
Messages

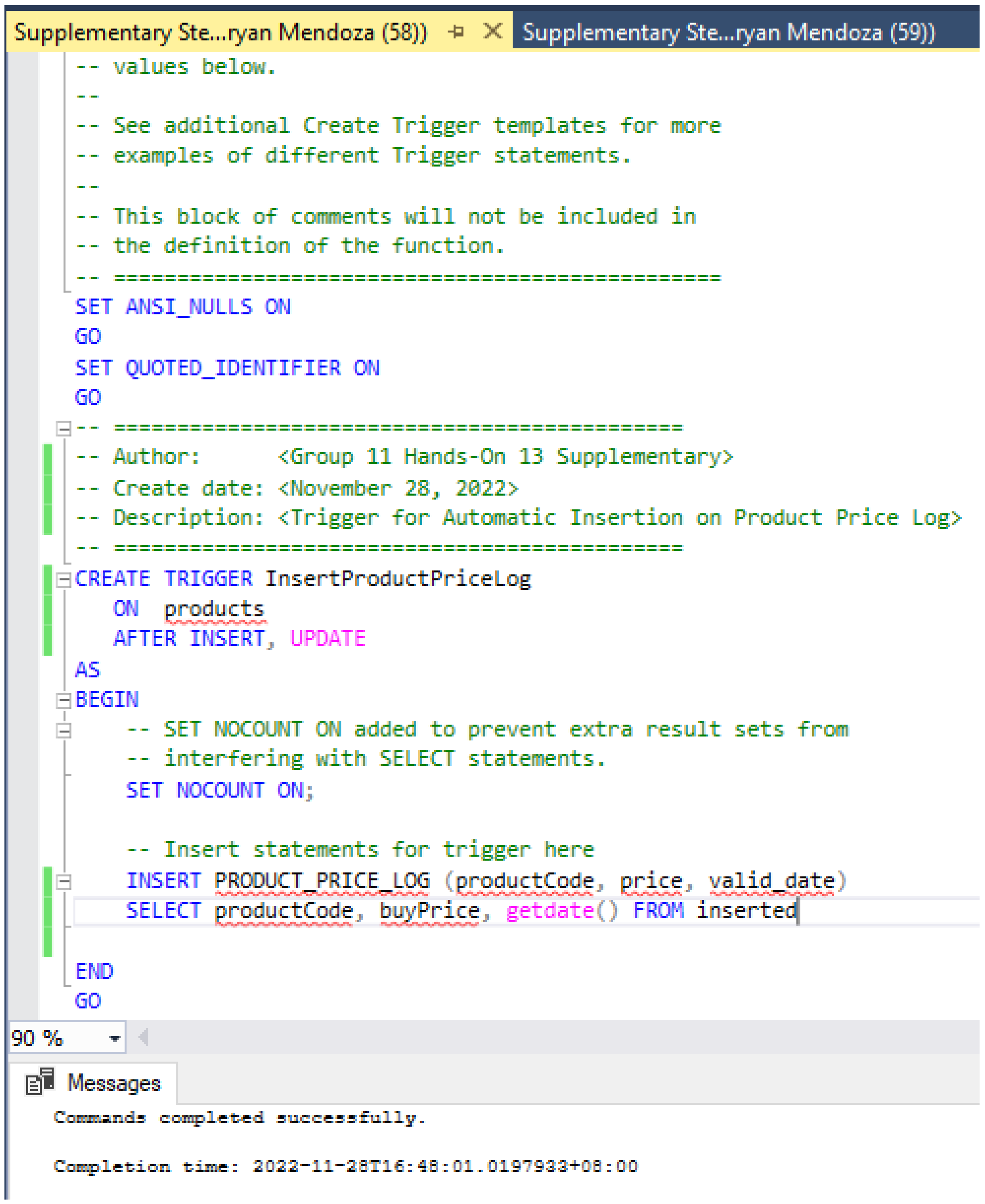
```
USE ClassicModels
GO

CREATE TABLE PRODUCT_PRICE_LOG(
    productCode VARCHAR(15),
    price MONEY,
    valid_date DATETIME
    )
GO
```

```
Commands completed successfully.

Completion time: 2022-11-28T16:33:17.1344009+08:00
```

Observation:
Using a normal query, we have created a table in order to perform the next procedures.

7. Create a trigger that updates the table PRODUCT_PRICE_LOG after inserting and updating the products record.

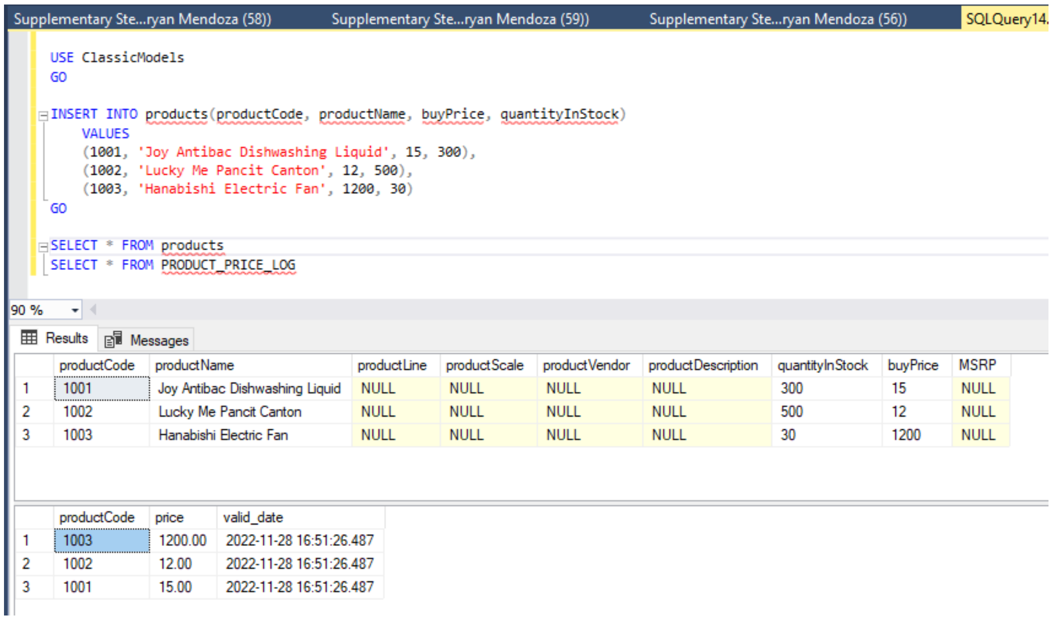
Creation of Trigger


Supplementary Ste...ryan Mendoza (58))
Supplementary Ste...ryan Mendoza (59))
90 %
Messages

```
-- values below.
--
-- See additional Create Trigger templates for more
-- examples of different Trigger statements.
--
-- This block of comments will not be included in
-- the definition of the function.
-- ================================================
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:      <Group 11 Hands-On 13 Supplementary>
-- Create date: <November 28, 2022>
-- Description: <Trigger for Automatic Insertion on Product Price Log>
-- =============================================
CREATE TRIGGER InsertProductPriceLog
   ON  products
   AFTER INSERT, UPDATE
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for trigger here
    INSERT PRODUCT_PRICE_LOG (productCode, price, valid_date)
    SELECT productCode, buyPrice, getdate() FROM inserted

END
GO
```

```
Commands completed successfully.

Completion time: 2022-11-28T16:48:01.0197933+08:00
```

Trigger Execution:


Supplementary Ste...ryan Mendoza (58)) Supplementary Ste...ryan Mendoza (59)) Supplementary Ste...ryan Mendoza (56)) SQLQuery14.

```
USE ClassicModels
GO

INSERT INTO products(productCode, productName, buyPrice, quantityInStock)
    VALUES
    (1001, 'Joy Antibac Dishwashing Liquid', 15, 300),
    (1002, 'Lucky Me Pancit Canton', 12, 500),
    (1003, 'Hanabishi Electric Fan', 1200, 30)
GO

SELECT * FROM products
SELECT * FROM PRODUCT_PRICE_LOG
```

90 %

Results | Messages

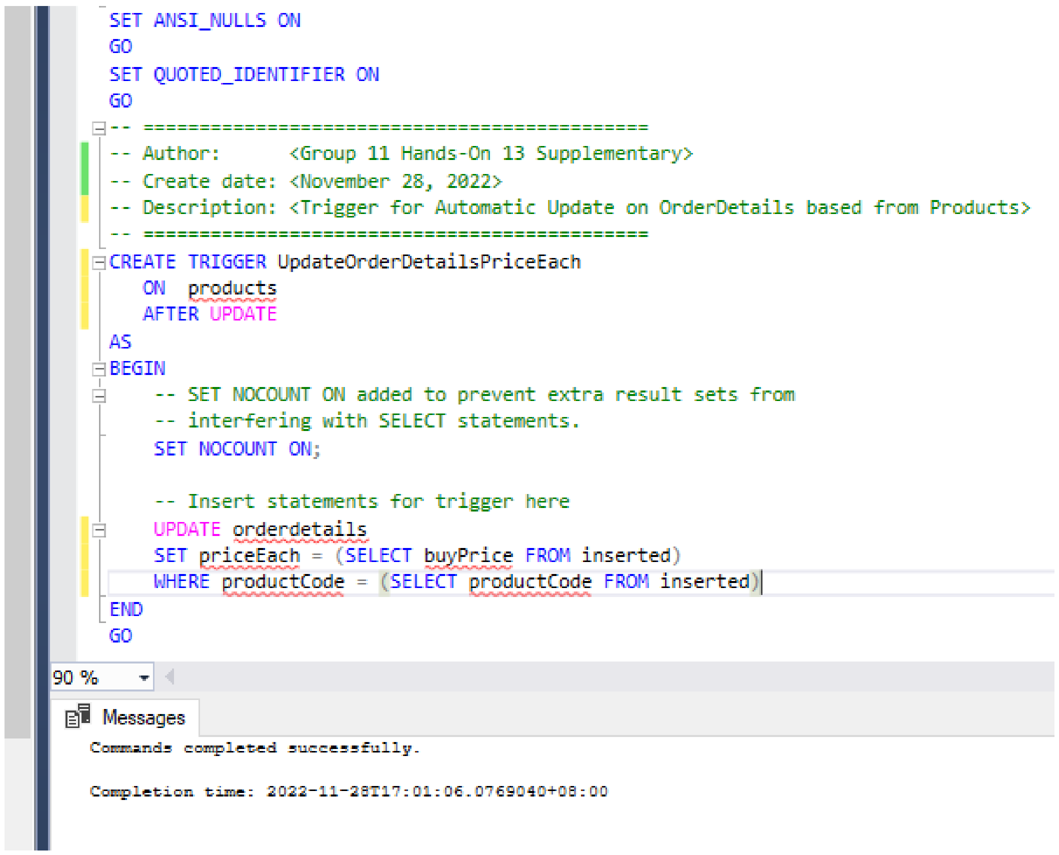
	productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP
1	1001	Joy Antibac Dishwashing Liquid	NULL	NULL	NULL	NULL	300	15	NULL
2	1002	Lucky Me Pancit Canton	NULL	NULL	NULL	NULL	500	12	NULL
3	1003	Hanabishi Electric Fan	NULL	NULL	NULL	NULL	30	1200	NULL

	productCode	price	valid_date
1	1003	1200.00	2022-11-28 16:51:26.487
2	1002	12.00	2022-11-28 16:51:26.487
3	1001	15.00	2022-11-28 16:51:26.487

Observation:
By setting the trigger to activate when there is an insertion or updates on products table, we are able to implement it such that it will insert entries to the product_price_log. As we can observe on the demonstration, every time there are changes on the products table that are about insertion or update, the product_price_log also changes.

8. Create a trigger that automatically updates the priceEach of the orderdetails record depending on the price on the Product table.

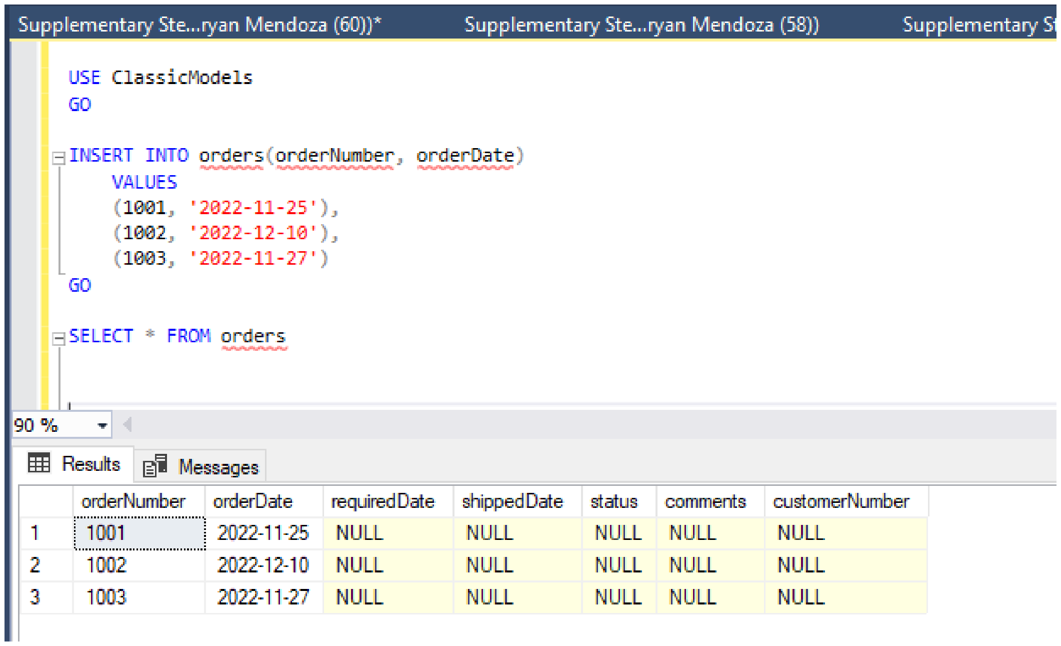
Creation of the trigger



```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:      <Group 11 Hands-On 13 Supplementary>
-- Create date: <November 28, 2022>
-- Description: <Trigger for Automatic Update on OrderDetails based from Products>
-- =============================================
CREATE TRIGGER UpdateOrderDetailsPriceEach
   ON  products
   AFTER UPDATE
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for trigger here
    UPDATE orderdetails
    SET priceEach = (SELECT buyPrice FROM inserted)
    WHERE productCode = (SELECT productCode FROM inserted)
END
GO
```

```
Commands completed successfully.

Completion time: 2022-11-28T17:01:06.0769040+08:00
```

Adding Sample Orders to perform the procedure


Supplementary Ste...ryan Mendoza (60))*   Supplementary Ste...ryan Mendoza (58))   Supplementary S

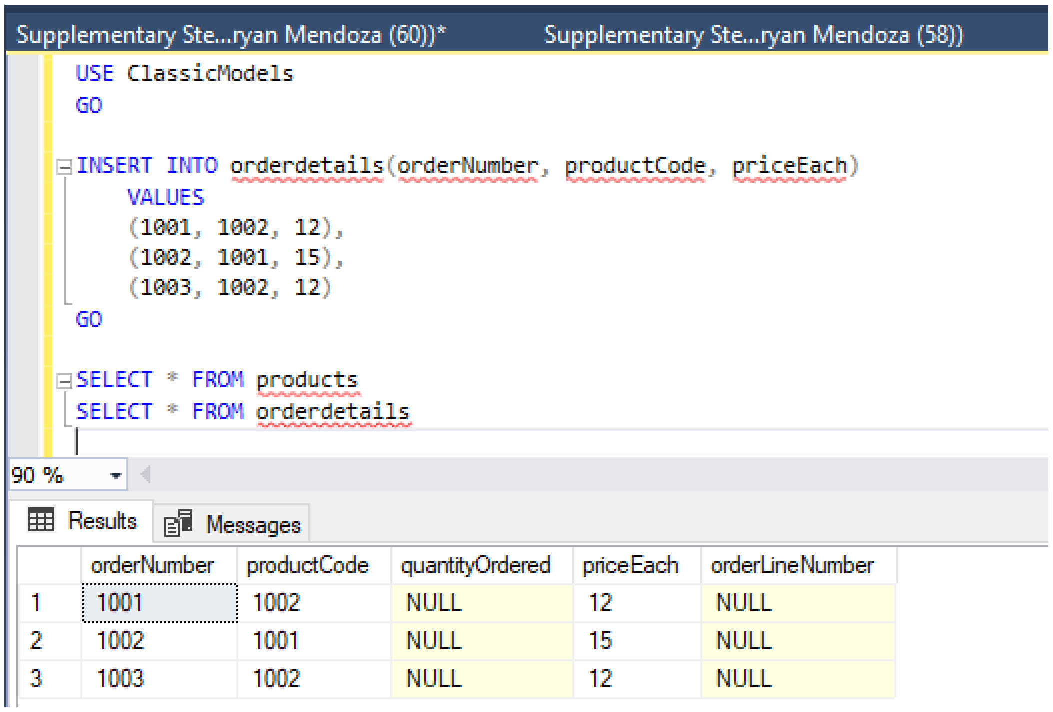
```
USE ClassicModels
GO

INSERT INTO orders(orderNumber, orderDate)
    VALUES
    (1001, '2022-11-25'),
    (1002, '2022-12-10'),
    (1003, '2022-11-27')
GO

SELECT * FROM orders
```

90 %

Results   Messages

	orderNumber	orderDate	requiredDate	shippedDate	status	comments	customerNumber
1	1001	2022-11-25	NULL	NULL	NULL	NULL	NULL
2	1002	2022-12-10	NULL	NULL	NULL	NULL	NULL
3	1003	2022-11-27	NULL	NULL	NULL	NULL	NULL

Adding Sample order details to perform the procedure


Supplementary Ste...ryan Mendoza (60))*   Supplementary Ste...ryan Mendoza (58))

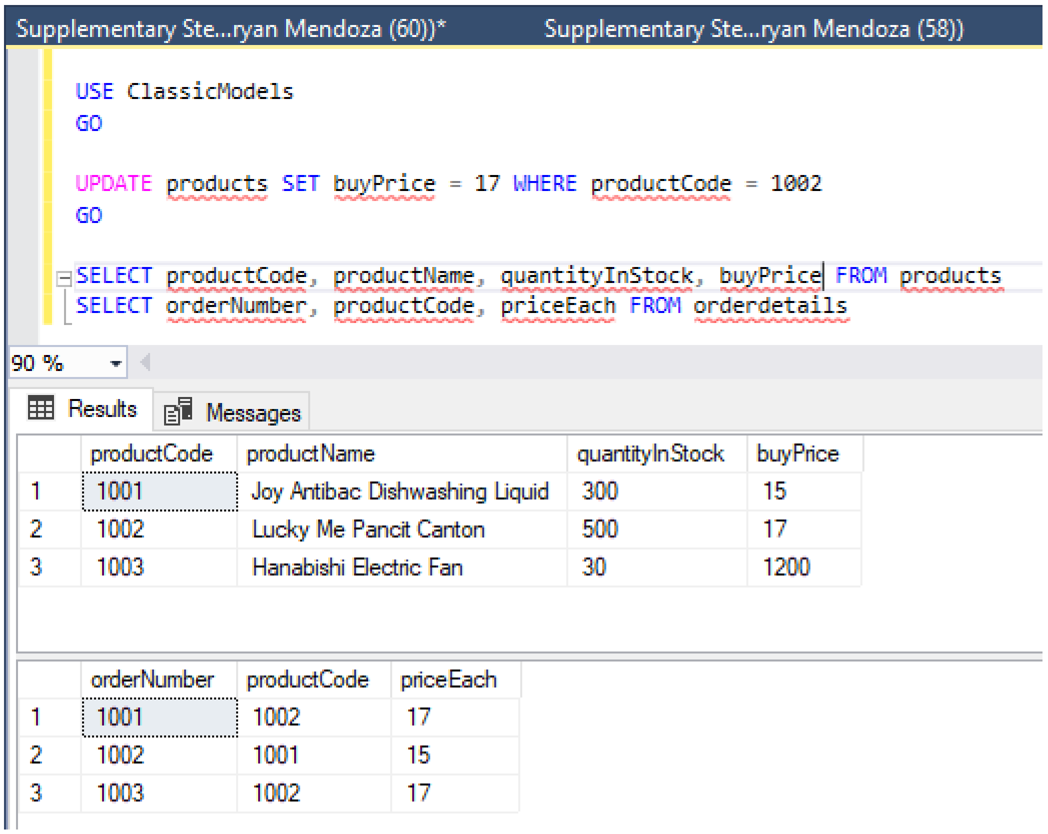
```
USE ClassicModels
GO

INSERT INTO orderdetails(orderNumber, productCode, priceEach)
    VALUES
    (1001, 1002, 12),
    (1002, 1001, 15),
    (1003, 1002, 12)
GO

SELECT * FROM products
SELECT * FROM orderdetails
```

90 %

Results   Messages

	orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber
1	1001	1002	NULL	12	NULL
2	1002	1001	NULL	15	NULL
3	1003	1002	NULL	12	NULL

Trigger Execution:


Supplementary Ste...ryan Mendoza (60))*
Supplementary Ste...ryan Mendoza (58))
90 %
Results
Messages

```sql
USE ClassicModels
GO

UPDATE products SET buyPrice = 17 WHERE productCode = 1002
GO

SELECT productCode, productName, quantityInStock, buyPrice FROM products
SELECT orderNumber, productCode, priceEach FROM orderdetails
```

	productCode	productName	quantityInStock	buyPrice
1	1001	Joy Antibac Dishwashing Liquid	300	15
2	1002	Lucky Me Pancit Canton	500	17
3	1003	Hanabishi Electric Fan	30	1200

	orderNumber	productCode	priceEach
1	1001	1002	17
2	1002	1001	15
3	1003	1002	17

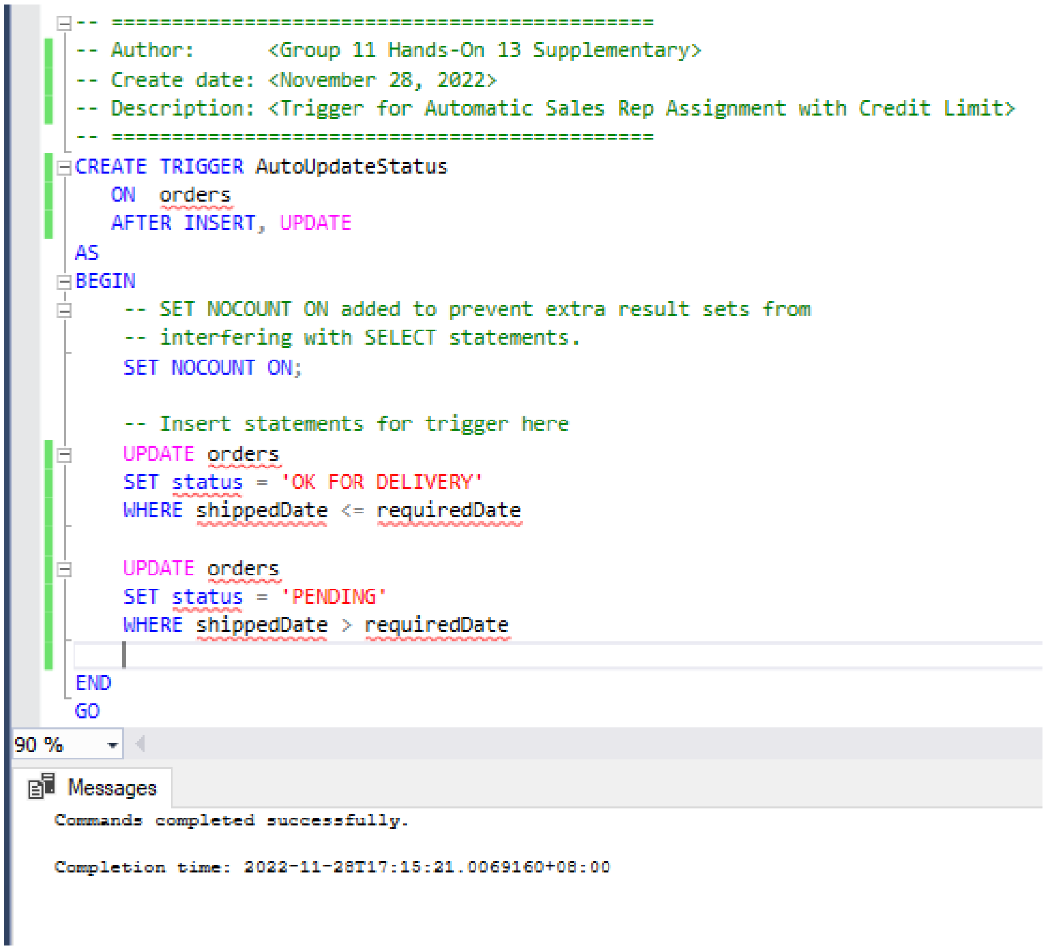
Observation:
We initially added entries on the orders and orderdetails table in order to present output for this procedure.

After the preliminary preparations, we created a trigger that will activate when updates on the products table, this trigger will update the priceEach column of the order details table whenever there are updates on the buyPrice column of the products table.

9. Create a trigger that automatically updates the status depending on the following condition:

- If the shipped date is less than or equal on the required date , the status is OK FOR DELIVERY
- If the shipped is greater than the required date , the status is PENDING.

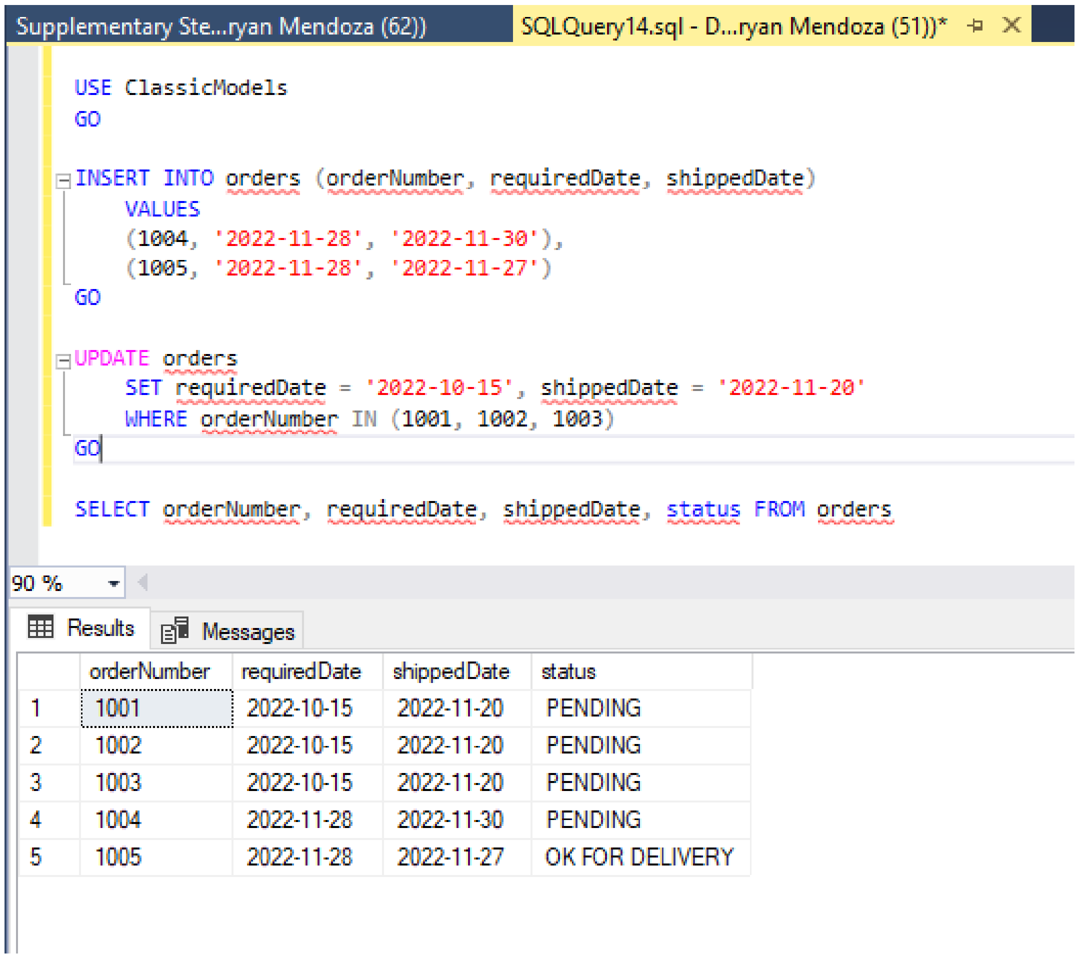
Creation of the Trigger:



```
-- =============================================
-- Author:      <Group 11 Hands-On 13 Supplementary>
-- Create date: <November 28, 2022>
-- Description: <Trigger for Automatic Sales Rep Assignment with Credit Limit>
-- =============================================
CREATE TRIGGER AutoUpdateStatus
   ON  orders
   AFTER INSERT, UPDATE
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for trigger here
    UPDATE orders
    SET status = 'OK FOR DELIVERY'
    WHERE shippedDate <= requiredDate

    UPDATE orders
    SET status = 'PENDING'
    WHERE shippedDate > requiredDate

END
GO
```

```
Commands completed successfully.

Completion time: 2022-11-28T17:15:21.0069160+08:00
```

Trigger Execution:



Supplementary Ste...ryan Mendoza (62)) | SQLQuery14.sql - D...ryan Mendoza (51))*

```
USE ClassicModels
GO

INSERT INTO orders (orderNumber, requiredDate, shippedDate)
    VALUES
    (1004, '2022-11-28', '2022-11-30'),
    (1005, '2022-11-28', '2022-11-27')
GO

UPDATE orders
    SET requiredDate = '2022-10-15', shippedDate = '2022-11-20'
    WHERE orderNumber IN (1001, 1002, 1003)
GO

SELECT orderNumber, requiredDate, shippedDate, status FROM orders
```

90 %

Results | Messages

	orderNumber	requiredDate	shippedDate	status
1	1001	2022-10-15	2022-11-20	PENDING
2	1002	2022-10-15	2022-11-20	PENDING
3	1003	2022-10-15	2022-11-20	PENDING
4	1004	2022-11-28	2022-11-30	PENDING
5	1005	2022-11-28	2022-11-27	OK FOR DELIVERY

Observation:
We have created a trigger that will activate whenever there are insertions or updates on the orders table. This trigger will update the values of the status column of the orders table depending on the requiredDate and shippedDate.

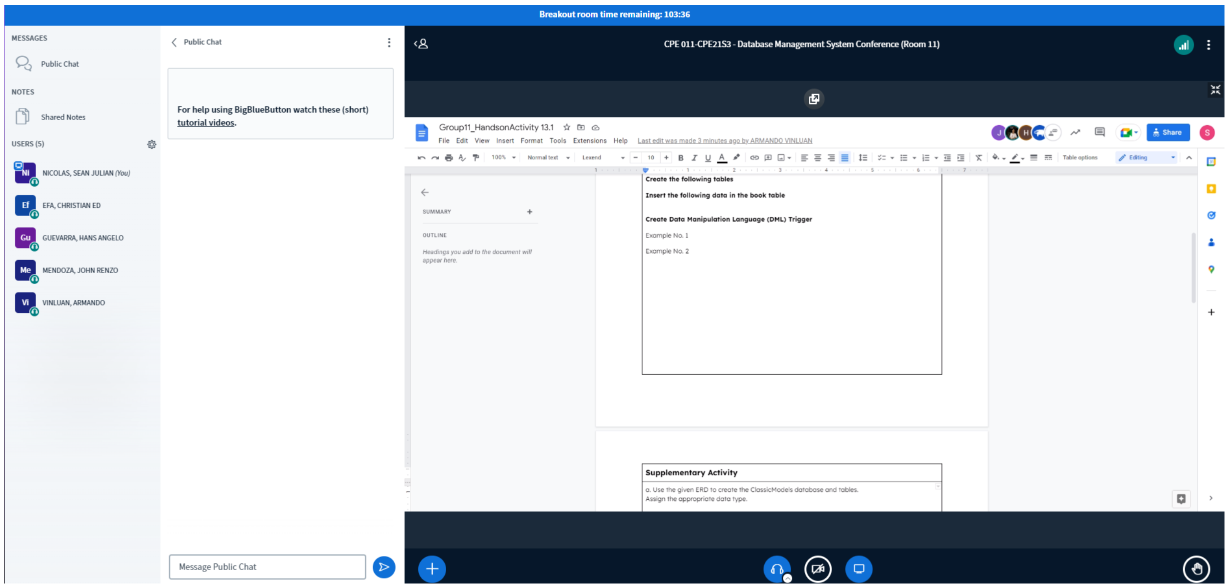
As we can observe on the demonstration, the trigger determines which status would be applicable on each order entry.

Conclusion

Triggers on the Microsoft SQL Server Management Studio, are used in order to automatically modify other tables based on the changes of the other tables within the same database.

This trigger function activates depending on the SQL programmer, it could be when insertion, deletion, modification or even combination of it. The changes it does also depend on the SQL programmer, just like we have observed on this laboratory activity.

Proof of Collaboration


Breakout room time remaining: 103:36
MESSAGES
Public Chat
NOTES
Shared Notes
USERS (5)
Public Chat
Message Public Chat
CPE 011-CPE21S3 - Database Management System Conference (Room 11)
Group11_HandsonActivity 13.1
File Edit View Insert Format Tools Extensions Help
Share
Table options
Editing
SUMMARY
OUTLINE
Headings you add to the document will appear here.

NICOLAS, SEAN JULIAN *(You)*
EFA, CHRISTIAN ED
GUEVARRA, HANS ANGELO
MENDOZA, JOHN RENZO
VINLUAN, ARMANDO

For help using BigBlueButton watch these (short) tutorial videos.

Last edit was made 3 minutes ago by ARMANDO VINLUAN

Create the following tables

Insert the following data in the book table

Create Data Manipulation Language (DML) Trigger

Example No. 1

Example No. 2

Supplementary Activity

a. Use the given ERD to create the ClassicModels database and tables.
Assign the appropriate data type.

Honor Pledge

"I accept responsibility for my role in ensuring the integrity of the work submitted by the group in which I participated."