

Assignment - 4

Create Stored procedure in Northwind database to insert or update a record in a table

1. Create a stored procedure in the Northwind database that will calculate the average value of Freight for a specified customer. Then, a business rule will be added that will be triggered before every Update and Insert command in the Orders controller, and will use the stored procedure to verify that the Freight does not exceed the average freight. If it does, a message will be displayed and the command will be cancelled.

```
CREATE PROC smInsertUpdateOrders
```

```
@OrderID INT,  
@CustomerID NCHAR(5),  
@EmployeeID INT,  
@OrderDate DATETIME,  
@RequiredDate DATETIME,  
@ShippedDate DATETIME,  
@ShipVia INT,  
@Freight MONEY,  
@ShipName NVARCHAR(40),  
@ShipAddress NVARCHAR(60),  
@ShipCity NVARCHAR(15),
```

```
@ShipRegion NVARCHAR(15),  
@ShipPostalCode NVARCHAR(10),  
@ShipCountry NVARCHAR(15),  
@type NVARCHAR(7)
```

```
AS
```

```
BEGIN
```

```
    DECLARE @AvgFreight MONEY
```

```
    SELECT @AvgFreight = AVG(Freight) FROM Orders
```

```
    WHERE CustomerID = @CustomerID
```

```
    GROUP BY CustomerID
```

```
    IF(@AvgFreight < @Freight)
```

```
        BEGIN
```

```
            RAISERROR('Operation denied due to Average Freight  
Condition.',10,1)
```

```
        END
```

```
    ELSE
```

```
        BEGIN
```

```
            IF(@type = 'INSERT')
```

```
                BEGIN
```

```
                    INSERT INTO Orders(OrderID, CustomerID,  
EmployeeID, OrderDate, RequiredDate, ShippedDate, ShipVia, Freight,
```

```
ShipName, ShipAddress, ShipCity, ShipRegion, ShipPostalCode,  
ShipCountry)
```

```
VALUES (@OrderID, @CustomerID,  
@EmployeeID, @OrderDate, @RequiredDate, @ShippedDate, @ShipVia,  
@Freight, @ShipName, @ShipAddress, @ShipCity, @ShipRegion,  
@ShipPostalCode, @ShipCountry)
```

```
END
```

```
ELSE
```

```
BEGIN
```

```
UPDATE Orders SET
```

```
CustomerID = @CustomerID, EmployeeID =  
@EmployeeID,
```

```
OrderDate = @OrderDate, RequiredDate =  
@RequiredDate, ShippedDate = @ShippedDate,
```

```
ShipVia = @ShipVia, Freight = @Freight,  
ShipName = @ShipName,
```

```
ShipAddress = @ShipAddress, ShipCity =  
@ShipCity, ShipRegion = @ShipRegion,
```

```
ShipPostalCode = @ShipPostalCode,  
ShipCountry = @ShipCountry
```

```
WHERE OrderID = @OrderID
```

```
END
```

```
END
```

```
END
```

```
SET IDENTITY_INSERT Orders ON
```

```
SQLQuery7.sql - PC...Northwind (sa (52)) X SQLQuery3.sql - PC...Northwind (sa (50))

--1.....

CREATE PROC smInsertUpdateOrders
    @OrderID INT,
    @CustomerID NVARCHAR(5),
    @EmployeeID INT,
    @OrderDate DATETIME,
    @RequiredDate DATETIME,
    @ShippedDate DATETIME,
    @ShipVia INT,
    @Freight MONEY,
    @ShipName NVARCHAR(40),
    @ShipAddress NVARCHAR(60),
    @ShipCity NVARCHAR(15),
    @ShipRegion NVARCHAR(15),
    @ShipPostalCode NVARCHAR(10),
    @ShipCountry NVARCHAR(15),
    @type NVARCHAR(7)
AS
BEGIN
    DECLARE @AvgFreight MONEY
    SELECT @AvgFreight = AVG(Freight) FROM Orders
    WHERE CustomerID = @CustomerID
    GROUP BY CustomerID

    IF (@AvgFreight < @Freight)
    BEGIN
        RAISERROR('Operation denied due to Average Freight Condition.',10,1)
    END
    ELSE
    BEGIN
        IF (@type = 'INSERT')
        BEGIN
            INSERT INTO Orders(OrderID, CustomerID, EmployeeID, OrderDate, RequiredDate, ShippedDate, ShipVia, Freight, ShipName, ShipAddress, ShipCity, ShipRegion, ShipPostalCode, ShipCountry)
            VALUES (@OrderID, @CustomerID, @EmployeeID, @OrderDate, @RequiredDate, @ShippedDate, @ShipVia, @Freight, @ShipName, @ShipAddress, @ShipCity, @ShipRegion, @ShipPostalCode, @ShipCountry)
        END
        ELSE
        END
    END
END

100 % - 4
Messages
Commands completed successfully.
```

```
SQLQuery1.sql - PC...Northwind (sa (53)) X

    ShipAddress = @ShipAddress, ShipCity = @ShipCity, ShipRegion = @ShipRegion,
    ShipPostalCode = @ShipPostalCode, ShipCountry = @ShipCountry
    WHERE OrderID = @OrderID
END
END
END

SET IDENTITY_INSERT Orders ON

EXEC smInsertUpdateOrders 10248, 'CHOPS', 4, '1996-07-03 00:00:00.000', '1996-07-15 00:00:00.000', '1996-07-10 00:00:00.000', 1, 30.00, 'Vins et alcools Chevalier', '59 rue c

133 % - 4
Messages
(1 row affected)
```

2. write a SQL query to Create Stored procedure in the Northwind database to retrieve Employee Sales by Country

```
CREATE PROCEDURE spSalesbyCountry
@StartingDate Date,
@EndingDate Date
AS
BEGIN
SELECT Employees.Country, SUM([Order Details].Quantity *
[Order Details].UnitPrice) AS [Total Sale]
FROM Employees
JOIN Orders ON Employees.EmployeeID = Orders.EmployeeID
JOIN [Order Details] ON [Order Details].OrderID =
Orders.OrderID
WHERE Orders.OrderDate BETWEEN @StartingDate AND
@EndingDate
GROUP BY Employees.Country
END
```

```
SQLQuery7.sql - PC...Northwind (sa (52))*  SQLQuery3.sql - PC...Northwind (sa (56))*

--2.....

CREATE PROCEDURE spSalesbyCountry
@StartingDate Date,
@EndingDate Date
AS
BEGIN
SELECT Employees.Country, SUM([Order Details].Quantity * [Order Details].UnitPrice) AS [Total Sale]
FROM Employees
JOIN Orders ON Employees.EmployeeID = Orders.EmployeeID
JOIN [Order Details] ON [Order Details].OrderID = Orders.OrderID
WHERE Orders.OrderDate BETWEEN @StartingDate AND @EndingDate
GROUP BY Employees.Country
END

spSalesbyCountry '1996-06-01', '1996-08-01'
```

161 %

Messages

Commands completed successfully.

```
SQLQuery7.sql - PC...Northwind (sa (52))*  SQLQuery3.sql - PC...Northwind (sa (56))*

--2.....

CREATE PROCEDURE spSalesbyCountry
@StartingDate Date,
@EndingDate Date
AS
BEGIN
SELECT Employees.Country, SUM([Order Details].Quantity * [Order Details].UnitPrice) AS [Total Sale]
FROM Employees
JOIN Orders ON Employees.EmployeeID = Orders.EmployeeID
JOIN [Order Details] ON [Order Details].OrderID = Orders.OrderID
WHERE Orders.OrderDate BETWEEN @StartingDate AND @EndingDate
GROUP BY Employees.Country
END

spSalesbyCountry '1996-06-01', '1996-08-01'
```

161 %

Results Messages

	Country	Total Sale
1	UK	10382.40
2	USA	22283.70

3. write a SQL query to Create Stored procedure in the Northwind database to retrieve Sales by Year

--for a specific year

ALTER PROCEDURE Sales_For_Specified_Year

@Year INT

AS

BEGIN

SELECT YEAR(Orders.ShippedDate) AS [YEAR],

SUM([Order Details].Quantity * [Order Details].UnitPrice) AS [Total Sale]

FROM Orders

INNER JOIN [Order Details] ON [Order Details].OrderID =
Orders.OrderID

WHERE YEAR(ShippedDate) = @Year

GROUP BY YEAR(ShippedDate)

END

Sales_For_Specified_Year 1997

--for all years

CREATE PROCEDURE Sales_by_Year

AS

BEGIN

SELECT YEAR(Orders.ShippedDate) AS [YEAR],

```
SUM([Order Details].Quantity * [Order Details].UnitPrice) AS [Total Sale]
```

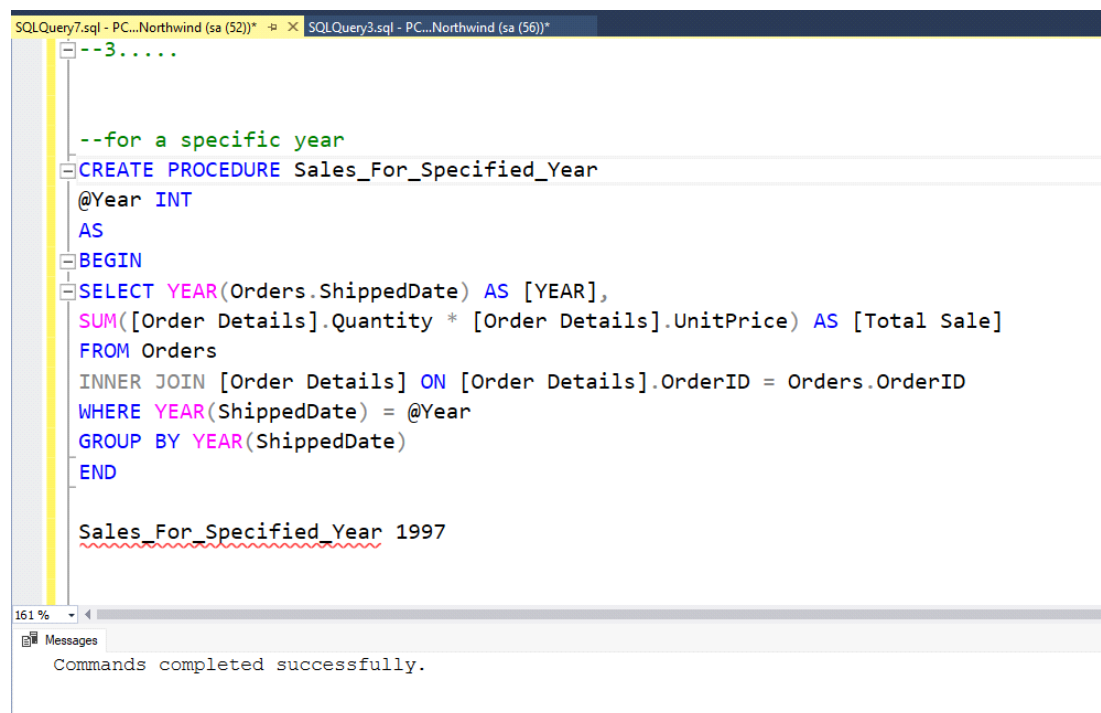
```
FROM Orders
```

```
INNER JOIN [Order Details] ON [Order Details].OrderID =  
Orders.OrderID
```

```
GROUP BY YEAR(ShippedDate)
```

```
END
```

Sales_by_Year



The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'SQLQuery7.sql - PC...Northwind (sa (52))' and 'SQLQuery3.sql - PC...Northwind (sa (56))'. The active tab displays a SQL script. The script includes a comment '--3.....', another comment '--for a specific year', and a 'CREATE PROCEDURE Sales_For_Specified_Year' statement. The procedure takes an '@Year INT' parameter and contains a 'SELECT' query that calculates the total sales for a specific year by joining 'Orders' and 'Order Details' tables. The 'SELECT' query uses 'YEAR(Orders.ShippedDate)' as the group by field and 'SUM([Order Details].Quantity * [Order Details].UnitPrice)' as the total sales calculation. The procedure is then executed with the year 1997. The bottom of the window shows a 'Messages' pane with the text 'Commands completed successfully.'

```
--3.....  
  
--for a specific year  
CREATE PROCEDURE Sales_For_Specified_Year  
@Year INT  
AS  
BEGIN  
SELECT YEAR(Orders.ShippedDate) AS [YEAR],  
SUM([Order Details].Quantity * [Order Details].UnitPrice) AS [Total Sale]  
FROM Orders  
INNER JOIN [Order Details] ON [Order Details].OrderID = Orders.OrderID  
WHERE YEAR(ShippedDate) = @Year  
GROUP BY YEAR(ShippedDate)  
END  
  
Sales_For_Specified_Year 1997
```

161 %
Messages
Commands completed successfully.


```
--3.....
```

```
--for a specific year
```

```
CREATE PROCEDURE Sales_For_Specified_Year
```

```
@Year INT
```

```
AS
```

```
BEGIN
```

```
SELECT YEAR(Orders.ShippedDate) AS [YEAR],
```

```
SUM([Order Details].Quantity * [Order Details].UnitPrice) AS [Total Sale]
```

```
FROM Orders
```

```
INNER JOIN [Order Details] ON [Order Details].OrderID = Orders.OrderID
```

```
WHERE YEAR(ShippedDate) = @Year
```

```
GROUP BY YEAR(ShippedDate)
```

```
END
```

```
Sales_For_Specified_Year 1997
```

161 %

Results Messages

	YEAR	Total Sale
1	1997	649038.81

4. write a SQL query to Create Stored procedure in the Northwind database to retrieve Sales By Category

--for specified categories

```
CREATE PROCEDURE Sales_for_Specified_Category
@categoryid INT
AS
BEGIN
SELECT Categories.CategoryID,Categories.CategoryName,
SUM([Order Details].Quantity * [Order Details].UnitPrice) AS [Total
Sale]
FROM Products
INNER JOIN [Order Details] ON [Order Details].ProductID =
Products.ProductID
INNER JOIN Categories ON Categories.CategoryID = Products.CategoryID
WHERE Categories.CategoryID = @categoryid
GROUP BY Categories.CategoryID,Categories.CategoryName
END
```

Sales_for_Specified_Category 1

--for all categories

```
CREATE PROCEDURE Sales_by_Category
AS
BEGIN
SELECT Categories.CategoryID,Categories.CategoryName,
```

```
SUM([Order Details].Quantity * [Order Details].UnitPrice) AS [Total Sale]
```

```
FROM Products
```

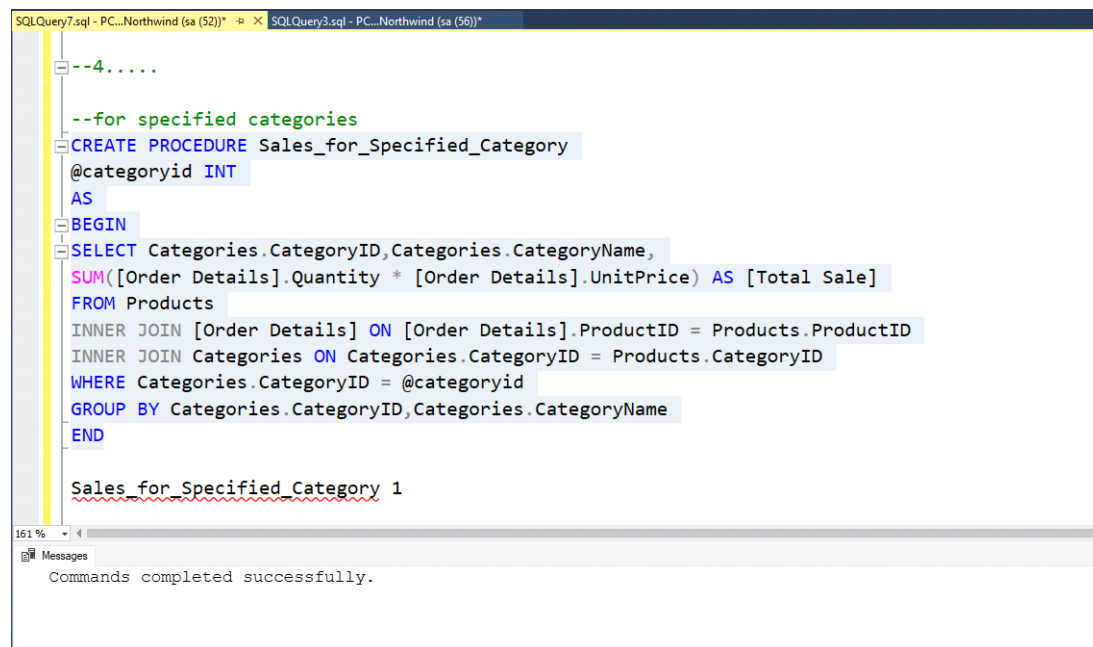
```
INNER JOIN [Order Details] ON [Order Details].ProductID =  
Products.ProductID
```

```
INNER JOIN Categories ON Categories.CategoryID = Products.CategoryID
```

```
GROUP BY Categories.CategoryID,Categories.CategoryName
```

```
END
```

Sales_by_Category



The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'SQLQuery7.sql - PC...Northwind (sa (52))' and 'SQLQuery3.sql - PC...Northwind (sa (56))'. The active tab displays a SQL script. The script starts with a comment '--4.....' and another comment '--for specified categories'. It then defines a stored procedure 'Sales_for_Specified_Category' with a parameter '@categoryid INT'. The procedure body begins with 'BEGIN', followed by a 'SELECT' statement that retrieves 'Categories.CategoryID' and 'Categories.CategoryName'. The 'SELECT' statement includes a subquery that calculates the total sales for a specific category. The subquery uses 'SUM([Order Details].Quantity * [Order Details].UnitPrice) AS [Total Sale]' and joins 'Products' and 'Order Details' tables. The 'WHERE' clause filters for the specified category. The 'GROUP BY' clause groups the results by 'Categories.CategoryID' and 'Categories.CategoryName'. The procedure ends with 'END'. Below the script, the command 'Sales_for_Specified_Category 1' is executed. The 'Messages' pane at the bottom shows the message 'Commands completed successfully.'

```
--4.....  
  
--for specified categories  
CREATE PROCEDURE Sales_for_Specified_Category  
@categoryid INT  
AS  
BEGIN  
SELECT Categories.CategoryID,Categories.CategoryName,  
SUM([Order Details].Quantity * [Order Details].UnitPrice) AS [Total Sale]  
FROM Products  
INNER JOIN [Order Details] ON [Order Details].ProductID = Products.ProductID  
INNER JOIN Categories ON Categories.CategoryID = Products.CategoryID  
WHERE Categories.CategoryID = @categoryid  
GROUP BY Categories.CategoryID,Categories.CategoryName  
END  
  
Sales_for_Specified_Category 1
```

161 %
Messages
Commands completed successfully.

```
--4.....
```

```
--for specified categories
```

```
CREATE PROCEDURE Sales_for_Specified_Category
```

```
@categoryid INT
```

```
AS
```

```
BEGIN
```

```
SELECT Categories.CategoryID,Categories.CategoryName,
```

```
SUM([Order Details].Quantity * [Order Details].UnitPrice) AS [Total Sale]
```

```
FROM Products
```

```
INNER JOIN [Order Details] ON [Order Details].ProductID = Products.ProductID
```

```
INNER JOIN Categories ON Categories.CategoryID = Products.CategoryID
```

```
WHERE Categories.CategoryID = @categoryid
```

```
GROUP BY Categories.CategoryID,Categories.CategoryName
```

```
END
```

```
Sales_for_Specified_Category 1
```

161 %

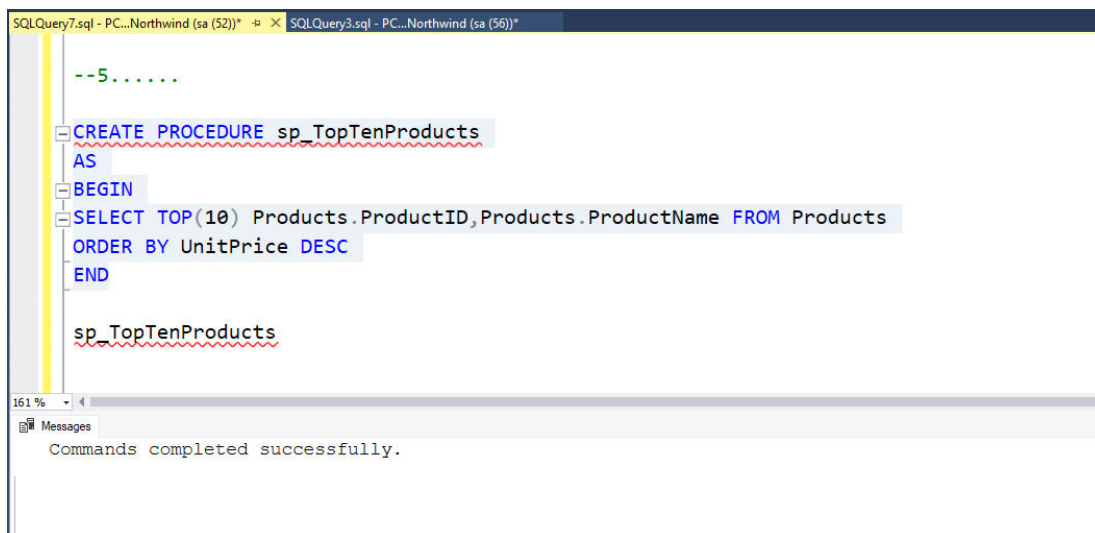
Results Messages

	CategoryID	CategoryName	Total Sale
1	1	Beverages	287576.95

5. write a SQL query to Create Stored procedure in the Northwind database to retrieve Ten Most Expensive Products

```
CREATE PROCEDURE sp_TopTenProducts
AS
BEGIN
SELECT TOP(10) Products.ProductID,Products.ProductName FROM
Products
ORDER BY UnitPrice DESC
END
```

sp_TopTenProducts



SQLQuery7.sql - PC...Northwind (sa (52))* SQLQuery3.sql - PC...Northwind (sa (56))*

--5.....

CREATE PROCEDURE sp_TopTenProducts
AS
BEGIN
SELECT TOP(10) Products.ProductID,Products.ProductName FROM Products
ORDER BY UnitPrice DESC
END

sp_TopTenProducts

161 %

Results Messages

	ProductID	ProductName
1	38	Côte de Blaye
2	29	Thüringer Rostbratwurst
3	9	Mishi Kobe Niku
4	20	Sir Rodney's Marmalade
5	18	Camarvon Tigers
6	59	Raclette Courdavault
7	51	Manjimup Dried Apples
8	62	Tarte au sucre
9	43	Ipoh Coffee
10	28	Rössle Sauerkraut

6. write a SQL query to Create Stored procedure in the Northwind database to insert Customer Order Details

```
CREATE PROCEDURE spInsertOrderDetails
(
    @OrderID INT,
    @ProductID INT ,
    @UnitPrice MONEY,
    @Quantity SMALLINT,
    @Discount REAL
)
AS
BEGIN
    INSERT INTO [Order
Details](OrderID,ProductID,UnitPrice,Quantity,Discount)
    VALUES
    (@OrderID,@ProductID,@UnitPrice,@Quantity,@Discount)

END
```

```
SQLQuery7.sql - PC...Northwind (sa (52))*  SQLQuery3.sql - PC...Northwind (sa (56))*
--6.....

CREATE PROCEDURE spInsertOrderDetails
(
    @OrderID INT,
    @ProductID INT,
    @UnitPrice MONEY,
    @Quantity SMALLINT,
    @Discount REAL
)
AS
BEGIN
    INSERT INTO [Order Details](OrderID,ProductID,UnitPrice,Quantity,Discount)
    VALUES (@OrderID,@ProductID,@UnitPrice,@Quantity,@Discount)
END

spInsertOrderDetails 10249,1,21.0,50,0
```

161 %
Messages
Commands completed successfully.

```
SQLQuery7.sql - PC...Northwind (sa (52))*  SQLQuery3.sql - PC...Northwind (sa (56))*

--for all categories
alter PROCEDURE Sales_by_Category
AS
BEGIN
    SELECT Categories.CategoryID,Categories.CategoryName,
    SUM([Order Details].Quantity * [Order Details].UnitPrice) AS [Total Sale]
    FROM Products
    INNER JOIN [Order Details] ON [Order Details].ProductID = Products.ProductID
    INNER JOIN Categories ON Categories.CategoryID = Products.CategoryID
    GROUP BY Categories.CategoryID,Categories.CategoryName
END

Sales_by_Category
```

161 %
Results Messages

	CategoryID	CategoryName	Total Sale
1	1	Beverages	287576.95
2	2	Condiments	113694.75
3	3	Confections	177099.10
4	4	Dairy Products	251330.50
5	5	Grains/Cereals	100726.80
6	6	Meat/Poultry	178188.80
7	7	Produce	105268.60
8	8	Seafood	141623.09

7. write a SQL query to Create Stored procedure in the Northwind database to update Customer Order Details

```
CREATE PROCEDURE spUpdateOrderDetails
(
    @OrderID INT,
    @ProductID INT ,
    @UnitPrice MONEY,
    @Quantity SMALLINT,
    @Discount REAL
)
AS
BEGIN
    UPDATE [Order Details]
    SET UnitPrice = @UnitPrice, Quantity = @Quantity, Discount =
    @Discount
    WHERE OrderID = @OrderID AND ProductID = @ProductID
END
```

```
spUpdateOrderDetails 10249,1,100,13,0.20
```

```
SQLQuery7.sql - PC...Northwind (sa (52))*  SQLQuery3.sql - PC...Northwind (sa (56))*

CREATE PROCEDURE spUpdateOrderDetails
(
    @OrderID INT,
    @ProductID INT,
    @UnitPrice MONEY,
    @Quantity SMALLINT,
    @Discount REAL
)
AS
BEGIN
    UPDATE [Order Details]
    SET UnitPrice = @UnitPrice, Quantity = @Quantity, Discount = @Discount
    WHERE OrderID = @OrderID AND ProductID = @ProductID
END

161 %
Messages
Commands completed successfully.
```

```
SQLQuery7.sql - PC...Northwind (sa (52))*  SQLQuery3.sql - PC...Northwind (sa (56))*

CREATE PROCEDURE spUpdateOrderDetails
(
    @OrderID INT,
    @ProductID INT,
    @UnitPrice MONEY,
    @Quantity SMALLINT,
    @Discount REAL
)
AS
BEGIN
    UPDATE [Order Details]
    SET UnitPrice = @UnitPrice, Quantity = @Quantity, Discount = @Discount
    WHERE OrderID = @OrderID AND ProductID = @ProductID
END

spUpdateOrderDetails 10249,1,100,13,0.20

161 %
Messages
(1 row affected)
```