# MARMARA UNIVERSITY

# FACULTY OF ENGINEERING

Computer Engineering

**CSE 3055 – DATABASE SYSTEMS**

PROJECT STEP#3

150118008 Osman Tunç Poyraz

150119035 Batuhan Baştürk

150120029 Fırat Bakıcı

## Project Description

In this project, we’re going to help the OQ company to help them build their database about their sales to other companies. The database system we’ve chosen for this project is MSSQL.

## Entities & Definitions

- Customer: this entity will keep customer info in detail
- Employee: this entity will keep employee info in detail and be the supertype of Manager and Staff entities
- Manager: this entity will keep the information of managers.
- Staff: this entity will keep the staff info
- Team: this entity is where the manager leads on or where the staff works on.
- Order: this entity will keep the orders
- Product: this entity will keep the information of the products
- Bill: this entity will keep the bill of the orders

- CustomerCompany: this entity will keep the company information that the product is ordered to
- Company: this entity will hold the information about the company where the employee works
- Delivery: this entity holds the information of the delivery with details.

## Scope

The database is for a company that has multiple departments, each with its own team of employees. The company has customer companies that place orders for products, which are then delivered to a specific country and region. The company also tracks billing information for each order.

All the tables that are declared in our design are included.

All product and material lists are provided in this database system as we are desired.

Order named as Ordering due to mssql's keyword reasons.

Added productTravelTime to Delivery Table.

Added totalPurchase to CustomerCompany Table.

Deleted amount from Bill Table.

ProductID added to Bill Table in the diagram.

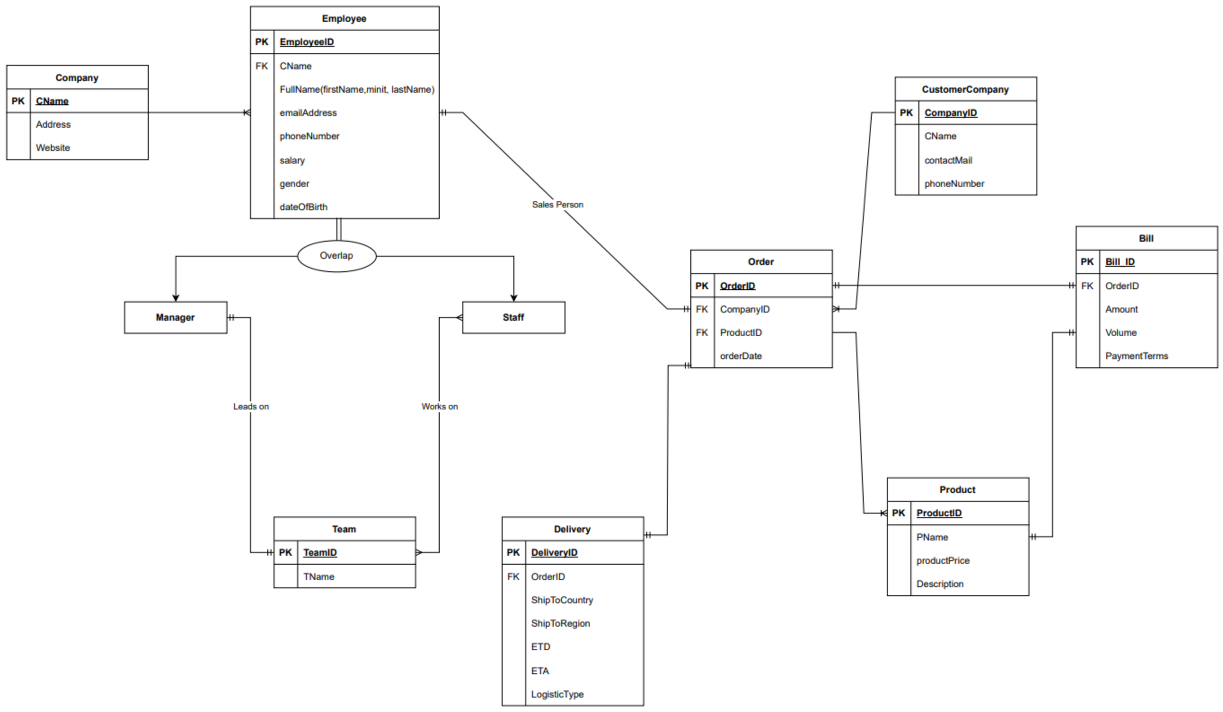
## Data & Requirement Analysis

Our clients ask us to help track its customer’s orders and bill’s essentially. Also we are keeping track of when our product reaches our customer and when it's leaving us. And we also keep track of how our product reaches our customers by air or by sea by truck.We are also keeping the information of our employee’s like who is who and who sold what etc… But the main thing was that we needed to order a list and the order’s bill.

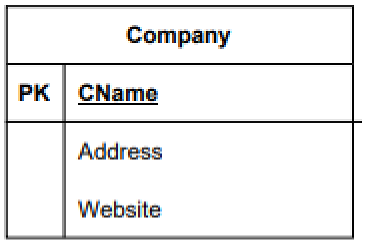
In the Order list there is an employee who made the deal or in charge of it, there is customerID, a ProductID which is getting sold, and the date of the Order.

And In the Bill list there is productID, OrderID that we are billing, the Volume of the product and the payment terms, which is the required amount that customer needs to pay for the deal.

## Database Diagram



## Tables



| Company | |
|---|---|
| PK | CName |
| | Address<br>Website |

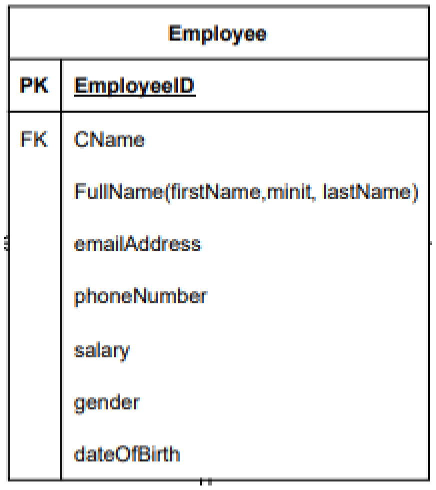
```
CREATE TABLE Company (
CName VARCHAR(255) PRIMARY KEY NOT NULL,
Address VARCHAR(255),
Website VARCHAR(255),
);
```

*Company Table*: This table holds the information about the company where the employee works.

- CName: Company name, consists of 255 characters, shouldn't be null
- Address: Address of company, consists of 255 characters

- Website: Website of company, consists of 255 characters

The CName column is the primary key of the Company table.



```
CREATE TABLE Employee (
EmployeeID INT PRIMARY KEY IDENTITY(1,1),
CName VARCHAR(255) NOT NULL,
FirstName VARCHAR(40) NOT NULL,
LastName VARCHAR(40),
emailAddress VARCHAR(255) NOT NULL UNIQUE,
phoneNumber VARCHAR(255) NOT NULL UNIQUE,
salary int,
gender CHAR(1) NOT NULL CHECK (gender IN ('M', 'F')),
dateOfBirth DATE NOT NULL,
FOREIGN KEY (CName) REFERENCES Company (CName)
);
```

*Employee Table*: This table keeps the employee info in detail and be the supertype of Manager and Staff entities.

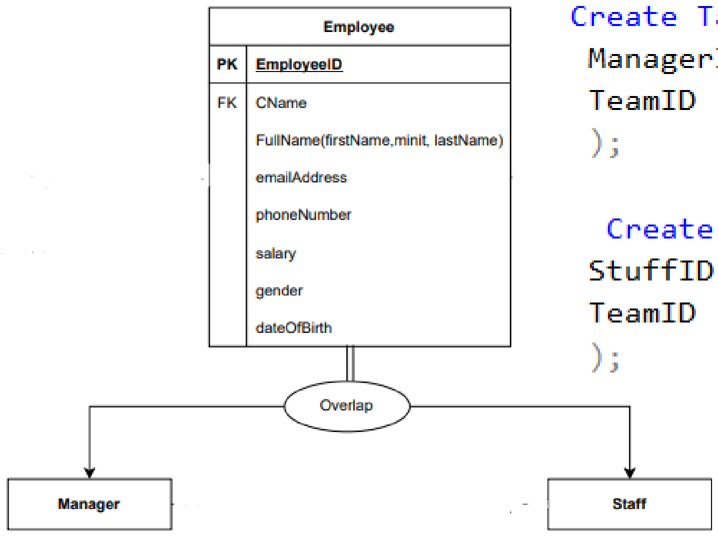
- EmployeeID: Employee's ID number, integer
- FirstName: First Name of the Employee, consists of 40 characters, shouldn't be null
- LastName: Last Name of the Employee, consists of 40 characters
- emailAddress: Email Address of the Employee, consists of 255 characters, shouldn't be null and must be **unique**
- phoneNumber: Phone Number of the Employee, integer, shouldn't be null and must be **unique**
- salary: Salary of the Employee, integer
- gender: Gender of the Employee, 1 character
- dateOfBirth: Date of birth of Employee, consists of 40 characters

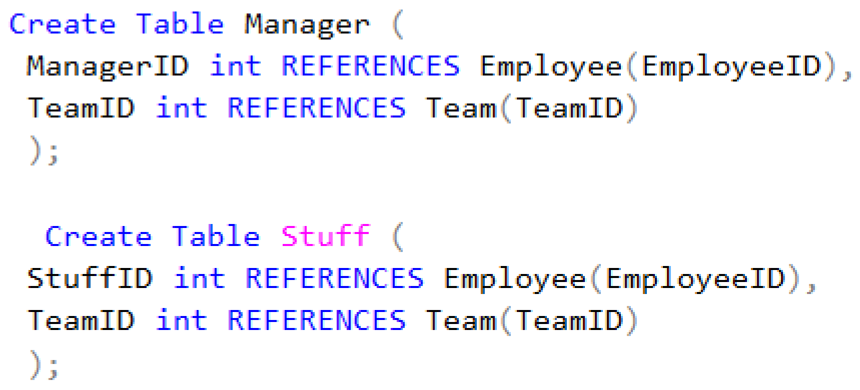
EmployeeID is an identity column.

CName is a foreign key that references Company table's CName.

EmployeeID attribute is the primary key of the Employee table.

the gender column is checked that it should be either 'M' or 'F'.





```
Create Table Manager (
 ManagerID int REFERENCES Employee(EmployeeID),
 TeamID int REFERENCES Team(TeamID)
 );

  Create Table Stuff (
 StuffID int REFERENCES Employee(EmployeeID),
 TeamID int REFERENCES Team(TeamID)
 );
```

*Manager Table:* This table keeps the information of managers. It is also a subtype of the Employee table.

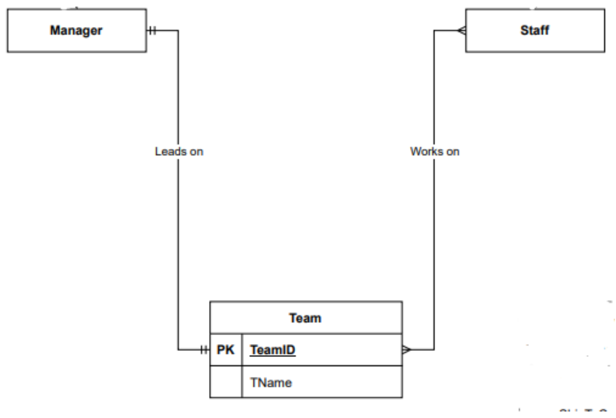
ManagerID is a foreign key that references the Employee table's EmployeeID.

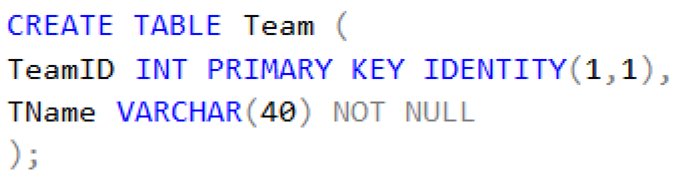
TeamID is a foreign key that references Team table's (screenshot below) TeamID.

*Staff Table*: This table keeps the staff information. It is also a subtype of the Employee table.

StuffID is a foreign key that references the Employee table's EmployeeID.

TeamID is a foreign key that references Team table's (screenshot below) TeamID.





```
CREATE TABLE Team (
TeamID INT PRIMARY KEY IDENTITY(1,1),
TName VARCHAR(40) NOT NULL
);
```

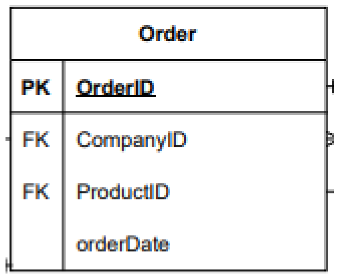
*Team Table:* This table is where the manager leads on or where the staff works on.

- TeamID: Team's ID number, integer
- TeamName: Name of the team, consists of 40 characters, shouldn't be null.

TeamID is an identity column.

TeamID attribute is the primary key of the Team table.

```
CREATE TABLE Ordering (
OrderID INT PRIMARY KEY IDENTITY(1,1),
CompanyID INT NOT NULL,
ProductID INT NOT NULL,
orderDate DATE NOT NULL DEFAULT GETDATE(),
FOREIGN KEY (CompanyID) REFERENCES CustomerCompany (CompanyID),
FOREIGN KEY (ProductID) REFERENCES Product (ProductID)
);
```



| Order | |
|---|---|
| PK | OrderID |
| FK | CompanyID |
| FK | ProductID |
| | orderDate |

*Ordering Table*: This table keeps the track of orders.

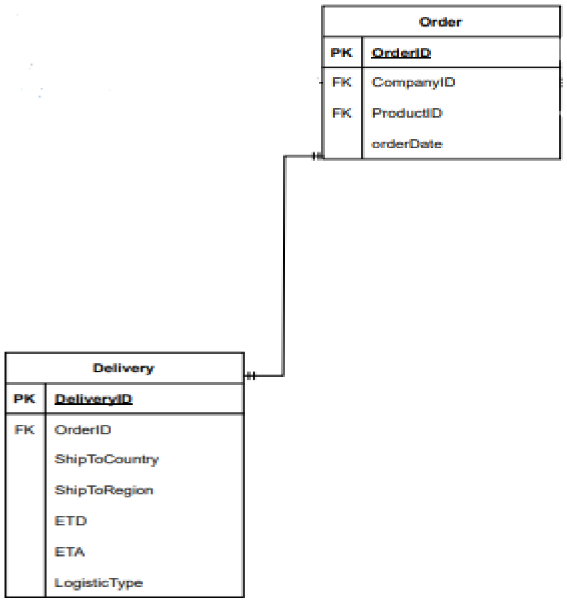
- OrderID: Order's ID number, integer
- orderdate: Date of the order, date

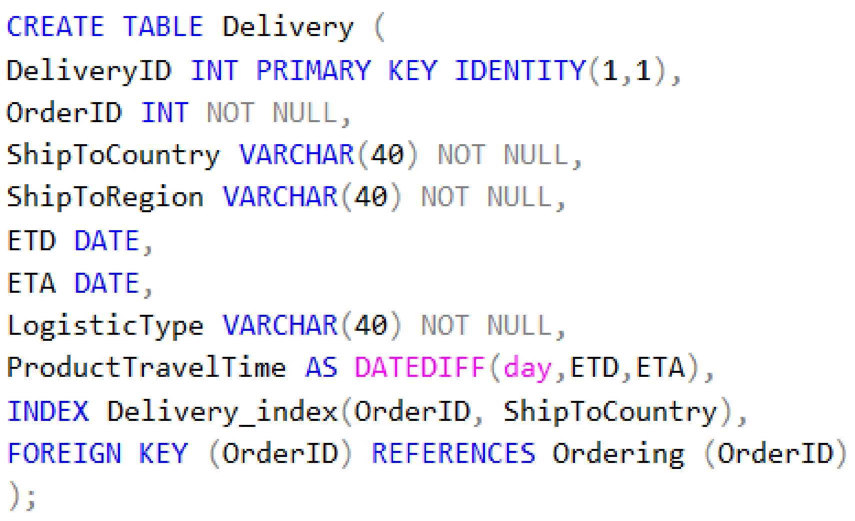
CompanyID is a foreign key that references CustomerCompany table's CompanyID and shouldn't be null.

ProductID is a foreign key that references Product table's ProductID and shouldn't be null.

orderDate is a default assigned attribute via GETDATE() function.

The OrderID attribute is the primary key of the Ordering table.





```
CREATE TABLE Delivery (
DeliveryID INT PRIMARY KEY IDENTITY(1,1),
OrderID INT NOT NULL,
ShipToCountry VARCHAR(40) NOT NULL,
ShipToRegion VARCHAR(40) NOT NULL,
ETD DATE,
ETA DATE,
LogisticType VARCHAR(40) NOT NULL,
ProductTravelTime AS DATEDIFF(day,ETD,ETA),
INDEX Delivery_index(OrderID, ShipToCountry),
FOREIGN KEY (OrderID) REFERENCES Ordering (OrderID)
);
```

*Delivery Table*: This table holds the information of the delivery with details of where to ship, estimated time values, logistic type etc.

- DeliveryID: ID of the delivery, integer
- ShipToCountry: Country where the delivery goes to, consists of 40 characters, shouldn't be null.
- ShipToRegion: Region where the delivery goes to, consists of 40 characters, shouldn't be null.
- ETD: Estimated time of departure of the delivery, date
- ETA: Estimated time of arrival of the delivery, date
- LogisticType: Type of logistic where the delivery shipped with, consists of 40 characters and shouldn't be null.

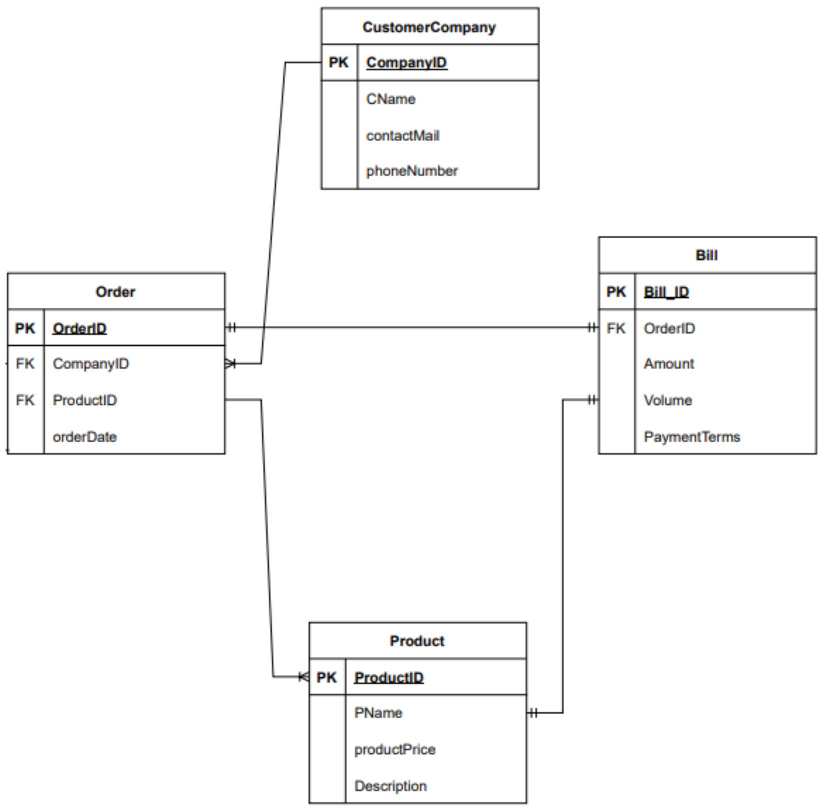
OrderID is a foreign key that references an Ordering table's OrderID.

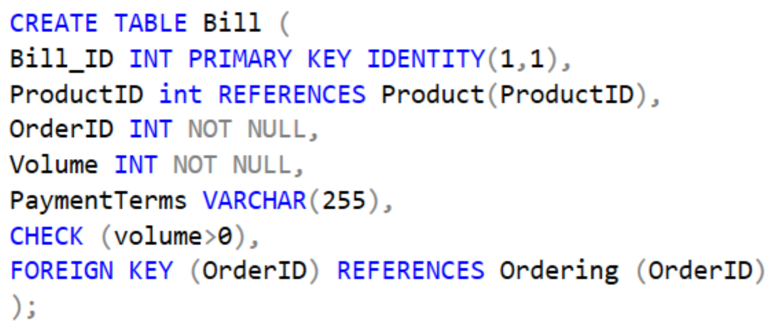
ProductTravelTime is a computed column via DATEDIFF() function with ETD, ETA parameters.

DeliveryID is the primary key of the Delivery table.

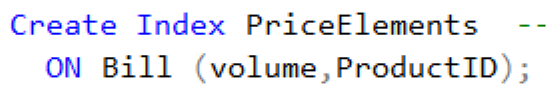
DeliveryID is an identity column.

There is also an Index we've created named Delivery_index that retrieves orderID and shipToCountry columns from the database more quickly.





```
CREATE TABLE Bill (
Bill_ID INT PRIMARY KEY IDENTITY(1,1),
ProductID int REFERENCES Product(ProductID),
OrderID INT NOT NULL,
Volume INT NOT NULL,
PaymentTerms VARCHAR(255),
CHECK (volume>0),
FOREIGN KEY (OrderID) REFERENCES Ordering (OrderID)
);
```



```
 Create Index PriceElements  --
   ON Bill (volume,ProductID);
```

*Bill Table*: This table keeps the bill of the orders

- Bill_ID: ID of the bill, integer
- Volume: Volume of the bill, integer, shouldn’t be null.
- PaymentTerms: Payment terms of the bill, consists of 255 characters.

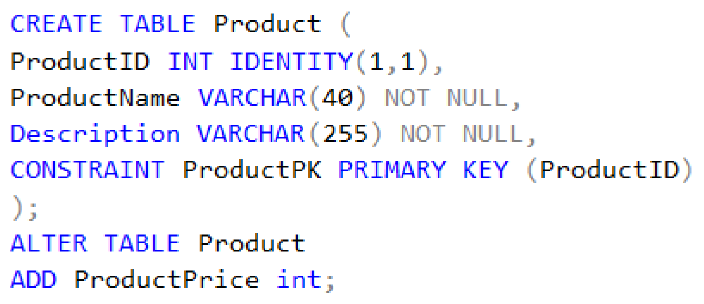
OrderID is a foreign key that references an Ordering table’s OrderID.

ProductID is a foreign key that references Product table’s ProductID.

BillID is the primary key of the Bill table.

BillID is an identity column.

There is a constraint that checks the volume column of the Bill table. Volume should be always greater than 0 when the table is created.
There is also an Index we’ve created named PriceElements that retrieves volume and productID columns from the database more quickly.



```sql
CREATE TABLE Product (
ProductID INT IDENTITY(1,1),
ProductName VARCHAR(40) NOT NULL,
Description VARCHAR(255) NOT NULL,
CONSTRAINT ProductPK PRIMARY KEY (ProductID)
);
ALTER TABLE Product
ADD ProductPrice int;
```

*Product Table:* This table keeps the information of the products with detail.

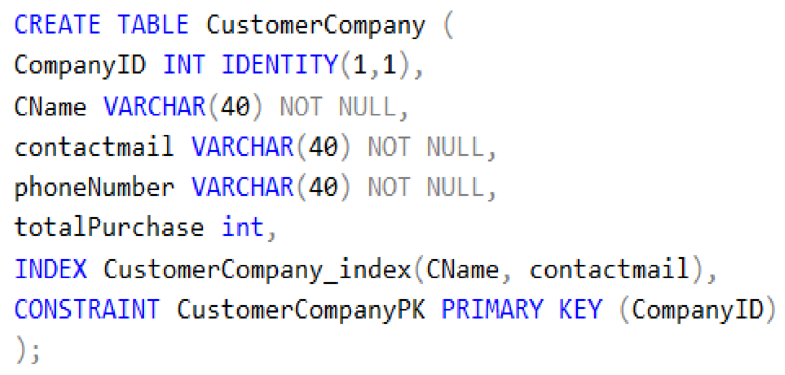
- ProductID: ID of the product, integer
- Product Name: Name of the product, consists of 40 characters and shouldn’t be null.

- Description: Description about product, consists of 255 characters, and shouldn’t be null.

ProductID is the primary key of the Product table.

ProductID is an identity column.

After creating the Product table, using ALTER, we added a ProductPrice column which is an integer.



```
CREATE TABLE CustomerCompany (
CompanyID INT IDENTITY(1,1),
CName VARCHAR(40) NOT NULL,
contactmail VARCHAR(40) NOT NULL,
phoneNumber VARCHAR(40) NOT NULL,
totalPurchase int,
INDEX CustomerCompany_index(CName, contactmail),
CONSTRAINT CustomerCompanyPK PRIMARY KEY (CompanyID)
);
```

*CustomerCompany Table:* This table keeps the company information that the product is ordered to.

- CompanyID: ID of the company, integer
- CName: Name of the company, consists of 40 characters, shouldn’t be null.
- contactMail: Mail of the company, consists of 40 characters, shouldn’t be null.
- phoneNumber: Phone number of the company, integer, shouldn’t be null.
- totalPurchase: Total purchase amount of the customer company, integer

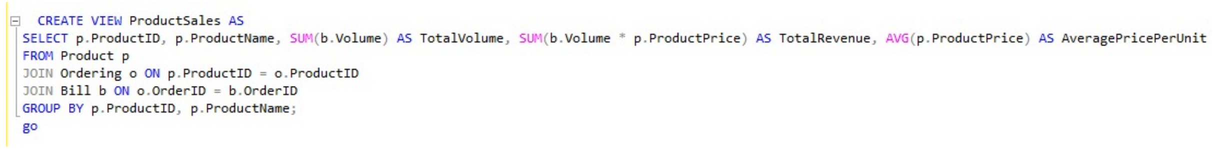
There is also an Index we've created named CustomerCompany_index that retrieves CName and contactMail columns from the database more quickly.

CompanyID is the primary key of the CustomerCompany table.

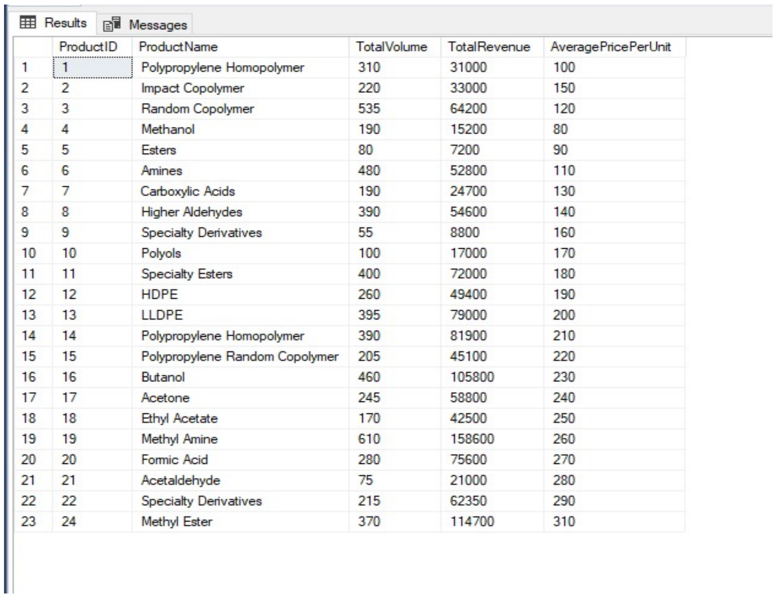
CompanyID is an identity column.

## VIEWS

**1-)Product Sales View**: This view could display the total sales (in terms of both volume and revenue) for each product, as well as the average price per unit. It could be created using the following SQL:



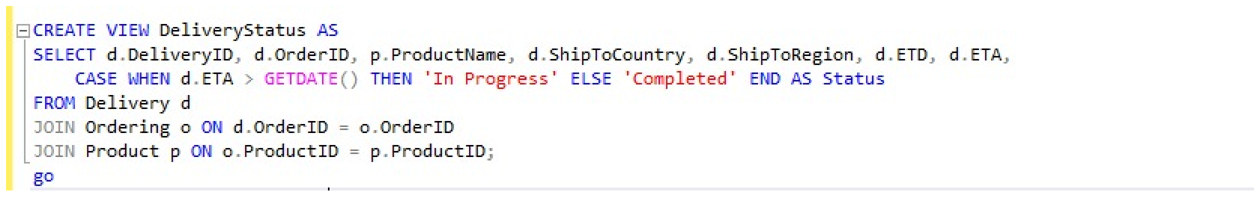
```sql
 CREATE VIEW ProductSales AS
SELECT p.ProductID, p.ProductName, SUM(b.Volume) AS TotalVolume, SUM(b.Volume * p.ProductPrice) AS TotalRevenue, AVG(p.ProductPrice) AS AveragePricePerUnit
FROM Product p
JOIN Ordering o ON p.ProductID = o.ProductID
JOIN Bill b ON o.OrderID = b.OrderID
GROUP BY p.ProductID, p.ProductName;
go
```



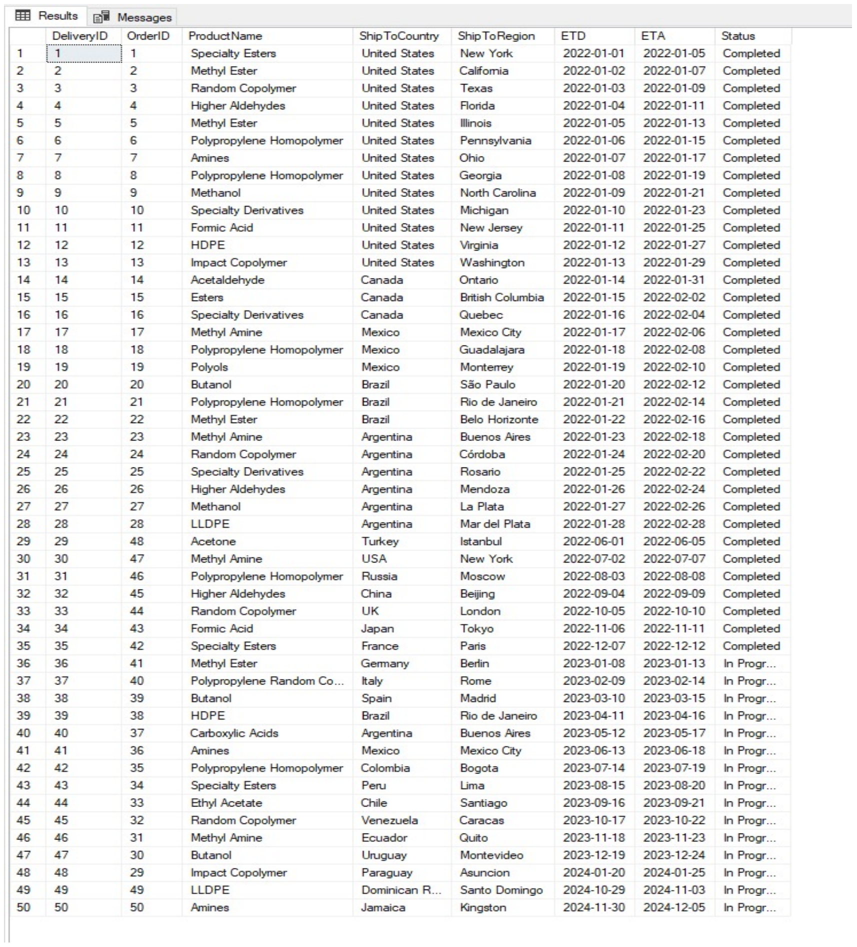
Results | Messages

| | ProductID | ProductName | TotalVolume | TotalRevenue | AveragePricePerUnit |
|---|---|---|---|---|---|
| 1 | 1 | Polypropylene Homopolymer | 310 | 31000 | 100 |
| 2 | 2 | Impact Copolymer | 220 | 33000 | 150 |
| 3 | 3 | Random Copolymer | 535 | 64200 | 120 |
| 4 | 4 | Methanol | 190 | 15200 | 80 |
| 5 | 5 | Esters | 80 | 7200 | 90 |
| 6 | 6 | Amines | 480 | 52800 | 110 |
| 7 | 7 | Carboxylic Acids | 190 | 24700 | 130 |
| 8 | 8 | Higher Aldehydes | 390 | 54600 | 140 |
| 9 | 9 | Specialty Derivatives | 55 | 8800 | 160 |
| 10 | 10 | Polyols | 100 | 17000 | 170 |
| 11 | 11 | Specialty Esters | 400 | 72000 | 180 |
| 12 | 12 | HDPE | 260 | 49400 | 190 |
| 13 | 13 | LLDPE | 395 | 79000 | 200 |
| 14 | 14 | Polypropylene Homopolymer | 390 | 81900 | 210 |
| 15 | 15 | Polypropylene Random Copolymer | 205 | 45100 | 220 |
| 16 | 16 | Butanol | 460 | 105800 | 230 |
| 17 | 17 | Acetone | 245 | 58800 | 240 |
| 18 | 18 | Ethyl Acetate | 170 | 42500 | 250 |
| 19 | 19 | Methyl Amine | 610 | 158600 | 260 |
| 20 | 20 | Formic Acid | 280 | 75600 | 270 |
| 21 | 21 | Acetaldehyde | 75 | 21000 | 280 |
| 22 | 22 | Specialty Derivatives | 215 | 62350 | 290 |
| 23 | 24 | Methyl Ester | 370 | 114700 | 310 |

**2-)Delivery Status View**: This view could display the delivery ID, order ID, product name, shipping country, shipping region, estimated time of departure, estimated time of arrival, and the delivery status (either "in progress" or "completed") for each delivery.



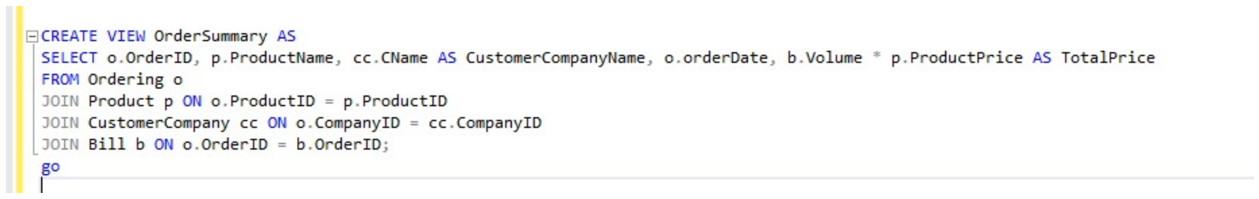
```
CREATE VIEW DeliveryStatus AS
SELECT d.DeliveryID, d.OrderID, p.ProductName, d.ShipToCountry, d.ShipToRegion, d.ETD, d.ETA,
    CASE WHEN d.ETA > GETDATE() THEN 'In Progress' ELSE 'Completed' END AS Status
FROM Delivery d
JOIN Ordering o ON d.OrderID = o.OrderID
JOIN Product p ON o.ProductID = p.ProductID;
go
```



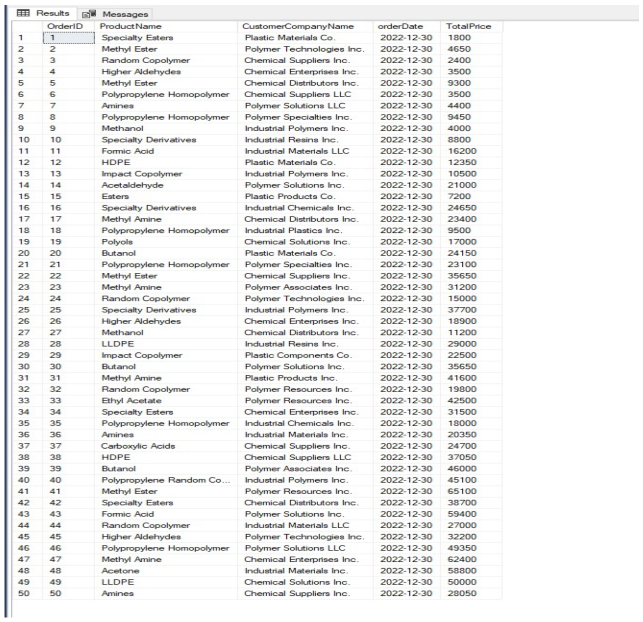
Results | Messages

| | DeliveryID | OrderID | ProductName | ShipToCountry | ShipToRegion | ETD | ETA | Status |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | Specialty Esters | United States | New York | 2022-01-01 | 2022-01-05 | Completed |
| 2 | 2 | 2 | Methyl Ester | United States | California | 2022-01-02 | 2022-01-07 | Completed |
| 3 | 3 | 3 | Random Copolymer | United States | Texas | 2022-01-03 | 2022-01-09 | Completed |
| 4 | 4 | 4 | Higher Aldehydes | United States | Florida | 2022-01-04 | 2022-01-11 | Completed |
| 5 | 5 | 5 | Methyl Ester | United States | Illinois | 2022-01-05 | 2022-01-13 | Completed |
| 6 | 6 | 6 | Polypropylene Homopolymer | United States | Pennsylvania | 2022-01-06 | 2022-01-15 | Completed |
| 7 | 7 | 7 | Amines | United States | Ohio | 2022-01-07 | 2022-01-17 | Completed |
| 8 | 8 | 8 | Polypropylene Homopolymer | United States | Georgia | 2022-01-08 | 2022-01-19 | Completed |
| 9 | 9 | 9 | Methanol | United States | North Carolina | 2022-01-09 | 2022-01-21 | Completed |
| 10 | 10 | 10 | Specialty Derivatives | United States | Michigan | 2022-01-10 | 2022-01-23 | Completed |
| 11 | 11 | 11 | Formic Acid | United States | New Jersey | 2022-01-11 | 2022-01-25 | Completed |
| 12 | 12 | 12 | HDPE | United States | Virginia | 2022-01-12 | 2022-01-27 | Completed |
| 13 | 13 | 13 | Impact Copolymer | United States | Washington | 2022-01-13 | 2022-01-29 | Completed |
| 14 | 14 | 14 | Acetaldehyde | Canada | Ontario | 2022-01-14 | 2022-01-31 | Completed |
| 15 | 15 | 15 | Esters | Canada | British Columbia | 2022-01-15 | 2022-02-02 | Completed |
| 16 | 16 | 16 | Specialty Derivatives | Canada | Quebec | 2022-01-16 | 2022-02-04 | Completed |
| 17 | 17 | 17 | Methyl Amine | Mexico | Mexico City | 2022-01-17 | 2022-02-06 | Completed |
| 18 | 18 | 18 | Polypropylene Homopolymer | Mexico | Guadalajara | 2022-01-18 | 2022-02-08 | Completed |
| 19 | 19 | 19 | Polyols | Mexico | Monterrey | 2022-01-19 | 2022-02-10 | Completed |
| 20 | 20 | 20 | Butanol | Brazil | São Paulo | 2022-01-20 | 2022-02-12 | Completed |
| 21 | 21 | 21 | Polypropylene Homopolymer | Brazil | Rio de Janeiro | 2022-01-21 | 2022-02-14 | Completed |
| 22 | 22 | 22 | Methyl Ester | Brazil | Belo Horizonte | 2022-01-22 | 2022-02-16 | Completed |
| 23 | 23 | 23 | Methyl Amine | Argentina | Buenos Aires | 2022-01-23 | 2022-02-18 | Completed |
| 24 | 24 | 24 | Random Copolymer | Argentina | Córdoba | 2022-01-24 | 2022-02-20 | Completed |
| 25 | 25 | 25 | Specialty Derivatives | Argentina | Rosario | 2022-01-25 | 2022-02-22 | Completed |
| 26 | 26 | 26 | Higher Aldehydes | Argentina | Mendoza | 2022-01-26 | 2022-02-24 | Completed |
| 27 | 27 | 27 | Methanol | Argentina | La Plata | 2022-01-27 | 2022-02-26 | Completed |
| 28 | 28 | 28 | LLDPE | Argentina | Mar del Plata | 2022-01-28 | 2022-02-28 | Completed |
| 29 | 29 | 48 | Acetone | Turkey | Istanbul | 2022-06-01 | 2022-06-05 | Completed |
| 30 | 30 | 47 | Methyl Amine | USA | New York | 2022-07-02 | 2022-07-07 | Completed |
| 31 | 31 | 46 | Polypropylene Homopolymer | Russia | Moscow | 2022-08-03 | 2022-08-08 | Completed |
| 32 | 32 | 45 | Higher Aldehydes | China | Beijing | 2022-09-04 | 2022-09-09 | Completed |
| 33 | 33 | 44 | Random Copolymer | UK | London | 2022-10-05 | 2022-10-10 | Completed |
| 34 | 34 | 43 | Formic Acid | Japan | Tokyo | 2022-11-06 | 2022-11-11 | Completed |
| 35 | 35 | 42 | Specialty Esters | France | Paris | 2022-12-07 | 2022-12-12 | Completed |
| 36 | 36 | 41 | Methyl Ester | Germany | Berlin | 2023-01-08 | 2023-01-13 | In Progr... |
| 37 | 37 | 40 | Polypropylene Random Co... | Italy | Rome | 2023-02-09 | 2023-02-14 | In Progr... |
| 38 | 38 | 39 | Butanol | Spain | Madrid | 2023-03-10 | 2023-03-15 | In Progr... |
| 39 | 39 | 38 | HDPE | Brazil | Rio de Janeiro | 2023-04-11 | 2023-04-16 | In Progr... |
| 40 | 40 | 37 | Carboxylic Acids | Argentina | Buenos Aires | 2023-05-12 | 2023-05-17 | In Progr... |
| 41 | 41 | 36 | Amines | Mexico | Mexico City | 2023-06-13 | 2023-06-18 | In Progr... |
| 42 | 42 | 35 | Polypropylene Homopolymer | Colombia | Bogota | 2023-07-14 | 2023-07-19 | In Progr... |
| 43 | 43 | 34 | Specialty Esters | Peru | Lima | 2023-08-15 | 2023-08-20 | In Progr... |
| 44 | 44 | 33 | Ethyl Acetate | Chile | Santiago | 2023-09-16 | 2023-09-21 | In Progr... |
| 45 | 45 | 32 | Random Copolymer | Venezuela | Caracas | 2023-10-17 | 2023-10-22 | In Progr... |
| 46 | 46 | 31 | Methyl Amine | Ecuador | Quito | 2023-11-18 | 2023-11-23 | In Progr... |
| 47 | 47 | 30 | Butanol | Uruguay | Montevideo | 2023-12-19 | 2023-12-24 | In Progr... |
| 48 | 48 | 29 | Impact Copolymer | Paraguay | Asuncion | 2024-01-20 | 2024-01-25 | In Progr... |
| 49 | 49 | 49 | LLDPE | Dominican R... | Santo Domingo | 2024-10-29 | 2024-11-03 | In Progr... |
| 50 | 50 | 50 | Amines | Jamaica | Kingston | 2024-11-30 | 2024-12-05 | In Progr... |

**3-)Order Summary View**: This view could display the order ID, product name, customer company name, order date, and total price for each order.



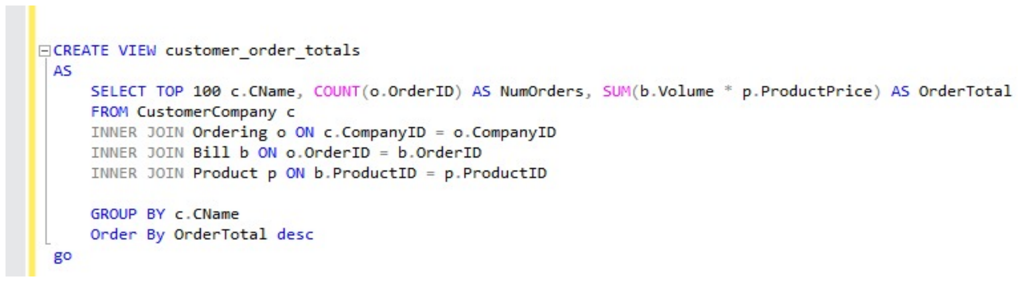
```sql
CREATE VIEW OrderSummary AS
SELECT o.OrderID, p.ProductName, cc.CName AS CustomerCompanyName, o.orderDate, b.Volume * p.ProductPrice AS TotalPrice
FROM Ordering o
JOIN Product p ON o.ProductID = p.ProductID
JOIN CustomerCompany cc ON o.CompanyID = cc.CompanyID
JOIN Bill b ON o.OrderID = b.OrderID;
go
```



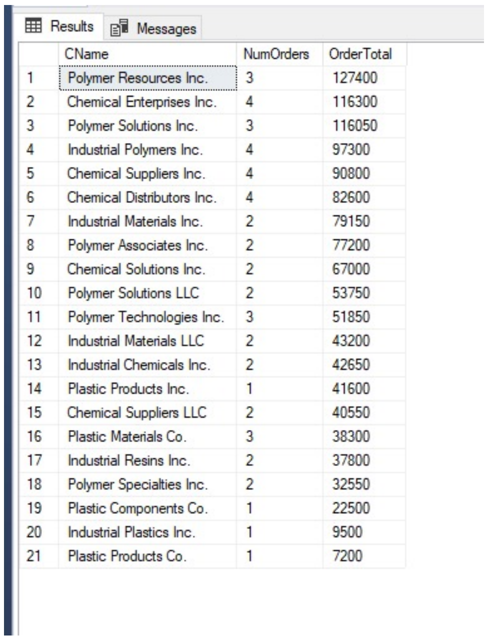
Results | Messages

| | OrderID | ProductName | CustomerCompanyName | orderDate | TotalPrice |
|---|---|---|---|---|---|
| 1 | 1 | Specialty Esters | Plastic Materials Co. | 2022-12-30 | 1800 |
| 2 | 2 | Methyl Ester | Polymer Technologies Inc. | 2022-12-30 | 4650 |
| 3 | 3 | Random Copolymer | Chemical Suppliers Inc. | 2022-12-30 | 2400 |
| 4 | 4 | Higher Aldehydes | Chemical Enterprises Inc. | 2022-12-30 | 3500 |
| 5 | 5 | Methyl Ester | Chemical Distributors Inc. | 2022-12-30 | 9300 |
| 6 | 6 | Polypropylene Homopolymer | Chemical Suppliers LLC | 2022-12-30 | 3500 |
| 7 | 7 | Amines | Polymer Solutions LLC | 2022-12-30 | 4400 |
| 8 | 8 | Polypropylene Homopolymer | Polymer Specialties Inc. | 2022-12-30 | 9450 |
| 9 | 9 | Methanol | Industrial Polymers Inc. | 2022-12-30 | 4000 |
| 10 | 10 | Specialty Derivatives | Industrial Resins Inc. | 2022-12-30 | 8800 |
| 11 | 11 | Formic Acid | Industrial Materials LLC | 2022-12-30 | 16200 |
| 12 | 12 | HDPE | Plastic Materials Co. | 2022-12-30 | 12350 |
| 13 | 13 | Impact Copolymer | Industrial Polymers Inc. | 2022-12-30 | 10500 |
| 14 | 14 | Acetaldehyde | Polymer Solutions Inc. | 2022-12-30 | 21000 |
| 15 | 15 | Esters | Plastic Products Co. | 2022-12-30 | 7200 |
| 16 | 16 | Specialty Derivatives | Industrial Chemicals Inc. | 2022-12-30 | 24650 |
| 17 | 17 | Methyl Amine | Chemical Distributors Inc. | 2022-12-30 | 23400 |
| 18 | 18 | Polypropylene Homopolymer | Industrial Plastics Inc. | 2022-12-30 | 9500 |
| 19 | 19 | Polyols | Chemical Solutions Inc. | 2022-12-30 | 17000 |
| 20 | 20 | Butanol | Plastic Materials Co. | 2022-12-30 | 24150 |
| 21 | 21 | Polypropylene Homopolymer | Polymer Specialties Inc. | 2022-12-30 | 23100 |
| 22 | 22 | Methyl Ester | Chemical Suppliers Inc. | 2022-12-30 | 35650 |
| 23 | 23 | Methyl Amine | Polymer Associates Inc. | 2022-12-30 | 31200 |
| 24 | 24 | Random Copolymer | Polymer Technologies Inc. | 2022-12-30 | 15000 |
| 25 | 25 | Specialty Derivatives | Industrial Polymers Inc. | 2022-12-30 | 37700 |
| 26 | 26 | Higher Aldehydes | Chemical Enterprises Inc. | 2022-12-30 | 18900 |
| 27 | 27 | Methanol | Chemical Distributors Inc. | 2022-12-30 | 11200 |
| 28 | 28 | LLDPE | Industrial Resins Inc. | 2022-12-30 | 29000 |
| 29 | 29 | Impact Copolymer | Plastic Components Co. | 2022-12-30 | 22500 |
| 30 | 30 | Butanol | Polymer Solutions Inc. | 2022-12-30 | 35650 |
| 31 | 31 | Methyl Amine | Plastic Products Inc. | 2022-12-30 | 41600 |
| 32 | 32 | Random Copolymer | Polymer Resources Inc. | 2022-12-30 | 19800 |
| 33 | 33 | Ethyl Acetate | Polymer Resources Inc. | 2022-12-30 | 42500 |
| 34 | 34 | Specialty Esters | Chemical Enterprises Inc. | 2022-12-30 | 31500 |
| 35 | 35 | Polypropylene Homopolymer | Industrial Chemicals Inc. | 2022-12-30 | 18000 |
| 36 | 36 | Amines | Industrial Materials Inc. | 2022-12-30 | 20350 |
| 37 | 37 | Carboxylic Acids | Chemical Suppliers Inc. | 2022-12-30 | 24700 |
| 38 | 38 | HDPE | Chemical Suppliers LLC | 2022-12-30 | 37050 |
| 39 | 39 | Butanol | Polymer Associates Inc. | 2022-12-30 | 46000 |
| 40 | 40 | Polypropylene Random Co... | Industrial Polymers Inc. | 2022-12-30 | 45100 |
| 41 | 41 | Methyl Ester | Polymer Resources Inc. | 2022-12-30 | 65100 |
| 42 | 42 | Specialty Esters | Chemical Distributors Inc. | 2022-12-30 | 38700 |
| 43 | 43 | Formic Acid | Polymer Solutions Inc. | 2022-12-30 | 59400 |
| 44 | 44 | Random Copolymer | Industrial Materials LLC | 2022-12-30 | 27000 |
| 45 | 45 | Higher Aldehydes | Polymer Technologies Inc. | 2022-12-30 | 32200 |
| 46 | 46 | Polypropylene Homopolymer | Polymer Solutions LLC | 2022-12-30 | 49350 |
| 47 | 47 | Methyl Amine | Chemical Enterprises Inc. | 2022-12-30 | 62400 |
| 48 | 48 | Acetone | Industrial Materials Inc. | 2022-12-30 | 58800 |
| 49 | 49 | LLDPE | Chemical Solutions Inc. | 2022-12-30 | 50000 |
| 50 | 50 | Amines | Chemical Suppliers Inc. | 2022-12-30 | 28050 |

**4-)**A view that displays the number of orders placed by each customer company, along with the total value of those orders:



```sql
CREATE VIEW customer_order_totals
AS
    SELECT TOP 100 c.CName, COUNT(o.OrderID) AS NumOrders, SUM(b.Volume * p.ProductPrice) AS OrderTotal
    FROM CustomerCompany c
    INNER JOIN Ordering o ON c.CompanyID = o.CompanyID
    INNER JOIN Bill b ON o.OrderID = b.OrderID
    INNER JOIN Product p ON b.ProductID = p.ProductID

    GROUP BY c.CName
    Order By OrderTotal desc
go
```

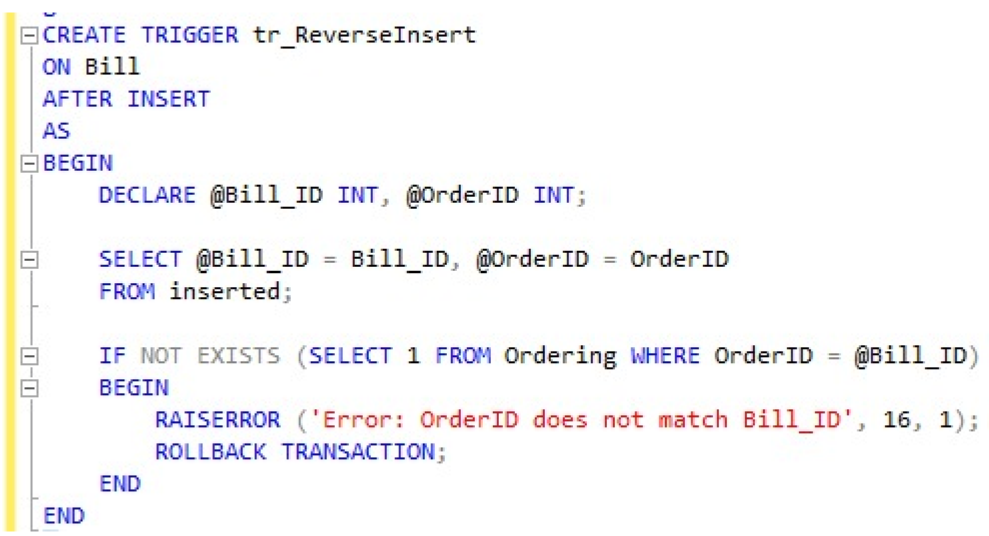


Results | Messages

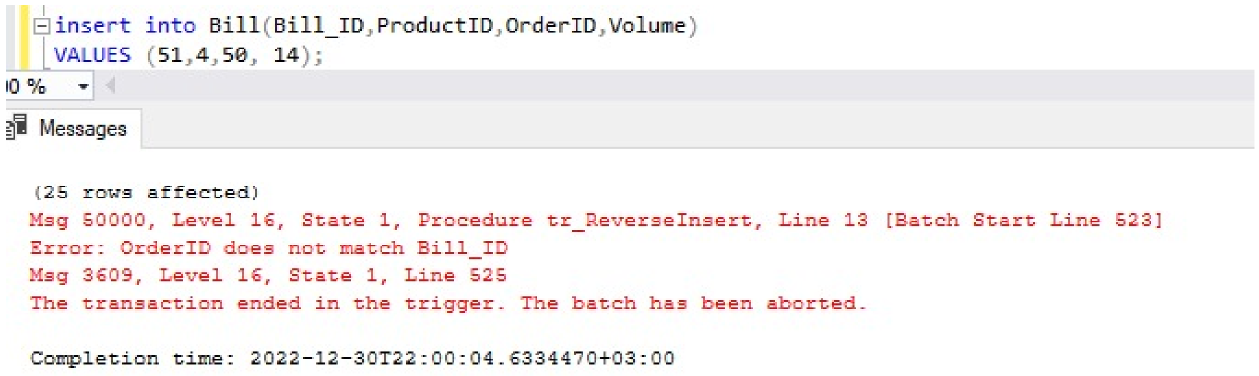
| | CName | NumOrders | OrderTotal |
|---|---|---|---|
| 1 | Polymer Resources Inc. | 3 | 127400 |
| 2 | Chemical Enterprises Inc. | 4 | 116300 |
| 3 | Polymer Solutions Inc. | 3 | 116050 |
| 4 | Industrial Polymers Inc. | 4 | 97300 |
| 5 | Chemical Suppliers Inc. | 4 | 90800 |
| 6 | Chemical Distributors Inc. | 4 | 82600 |
| 7 | Industrial Materials Inc. | 2 | 79150 |
| 8 | Polymer Associates Inc. | 2 | 77200 |
| 9 | Chemical Solutions Inc. | 2 | 67000 |
| 10 | Polymer Solutions LLC | 2 | 53750 |
| 11 | Polymer Technologies Inc. | 3 | 51850 |
| 12 | Industrial Materials LLC | 2 | 43200 |
| 13 | Industrial Chemicals Inc. | 2 | 42650 |
| 14 | Plastic Products Inc. | 1 | 41600 |
| 15 | Chemical Suppliers LLC | 2 | 40550 |
| 16 | Plastic Materials Co. | 3 | 38300 |
| 17 | Industrial Resins Inc. | 2 | 37800 |
| 18 | Polymer Specialties Inc. | 2 | 32550 |
| 19 | Plastic Components Co. | 1 | 22500 |
| 20 | Industrial Plastics Inc. | 1 | 9500 |
| 21 | Plastic Products Co. | 1 | 7200 |

## TRIGGERS

**1-)tr_ReverseInsert**: This trigger helps us check to see if there is an order to bill that we are making. If we don't have an order we can't have a bill. If no one buys anything there is no receipt to be given. The trigger checks if the bill_id and OrderID match or not. If it doesn't match it will reverse the transaction which means there is no order for a receipt to be given.



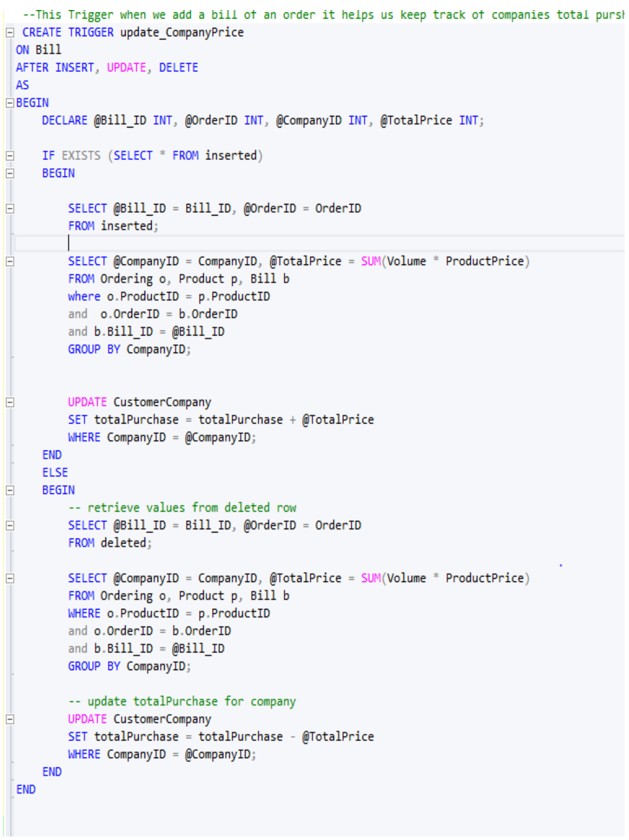
```sql
CREATE TRIGGER tr_ReverseInsert
ON Bill
AFTER INSERT
AS
BEGIN
    DECLARE @Bill_ID INT, @OrderID INT;

    SELECT @Bill_ID = Bill_ID, @OrderID = OrderID
    FROM inserted;

    IF NOT EXISTS (SELECT 1 FROM Ordering WHERE OrderID = @Bill_ID)
    BEGIN
        RAISERROR ('Error: OrderID does not match Bill_ID', 16, 1);
        ROLLBACK TRANSACTION;
    END
END
```



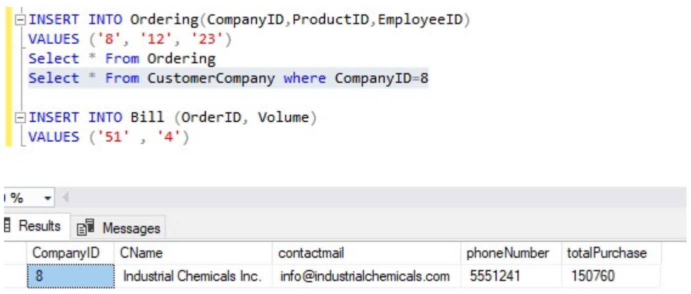
```sql
insert into Bill(Bill_ID,ProductID,OrderID,Volume)
VALUES (51,4,50, 14);
```

```
(25 rows affected)
Msg 50000, Level 16, State 1, Procedure tr_ReverseInsert, Line 13 [Batch Start Line 523]
Error: OrderID does not match Bill_ID
Msg 3609, Level 16, State 1, Line 525
The transaction ended in the trigger. The batch has been aborted.

Completion time: 2022-12-30T22:00:04.6334470+03:00
```

**2-)update_CompanyPrice**: Here we tried to update the customers purchase history, how much total they bought, after every insert,delete or update that happened in the bill table.

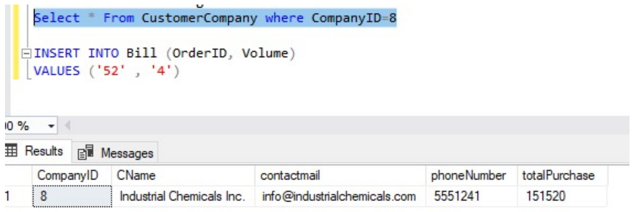


```sql
 --This Trigger when we add a bill of an order it helps us keep track of companies total pursh
 CREATE TRIGGER update_CompanyPrice
ON Bill
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    DECLARE @Bill_ID INT, @OrderID INT, @CompanyID INT, @TotalPrice INT;

    IF EXISTS (SELECT * FROM inserted)
    BEGIN

        SELECT @Bill_ID = Bill_ID, @OrderID = OrderID
        FROM inserted;

        SELECT @CompanyID = CompanyID, @TotalPrice = SUM(Volume * ProductPrice)
        FROM Ordering o, Product p, Bill b
        where o.ProductID = p.ProductID
        and  o.OrderID = b.OrderID
        and b.Bill_ID = @Bill_ID
        GROUP BY CompanyID;


        UPDATE CustomerCompany
        SET totalPurchase = totalPurchase + @TotalPrice
        WHERE CompanyID = @CompanyID;
    END
    ELSE
    BEGIN
        -- retrieve values from deleted row
        SELECT @Bill_ID = Bill_ID, @OrderID = OrderID
        FROM deleted;

        SELECT @CompanyID = CompanyID, @TotalPrice = SUM(Volume * ProductPrice)
        FROM Ordering o, Product p, Bill b
        WHERE o.ProductID = p.ProductID
        and o.OrderID = b.OrderID
        and b.Bill_ID = @Bill_ID
        GROUP BY CompanyID;

        -- update totalPurchase for company
        UPDATE CustomerCompany
        SET totalPurchase = totalPurchase - @TotalPrice
        WHERE CompanyID = @CompanyID;
    END
END
```



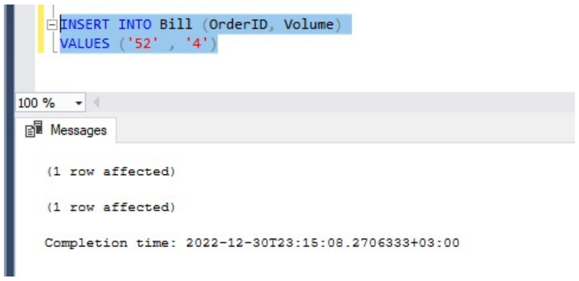
```sql
INSERT INTO Ordering(CompanyID,ProductID,EmployeeID)
VALUES ('8', '12', '23')
Select * From Ordering
Select * From CustomerCompany where CompanyID=8

INSERT INTO Bill (OrderID, Volume)
VALUES ('51' , '4')
```

```
   CompanyID  CName                     contactmail                     phoneNumber  totalPurchase
   8          Industrial Chemicals Inc. info@industrialchemicals.com    5551241      150760
```



```sql
Select * From CustomerCompany where CompanyID=8

INSERT INTO Bill (OrderID, Volume)
VALUES ('52' , '4')
```

```
   CompanyID  CName                     contactmail                     phoneNumber  totalPurchase
1  8          Industrial Chemicals Inc. info@industrialchemicals.com    5551241      151520
```

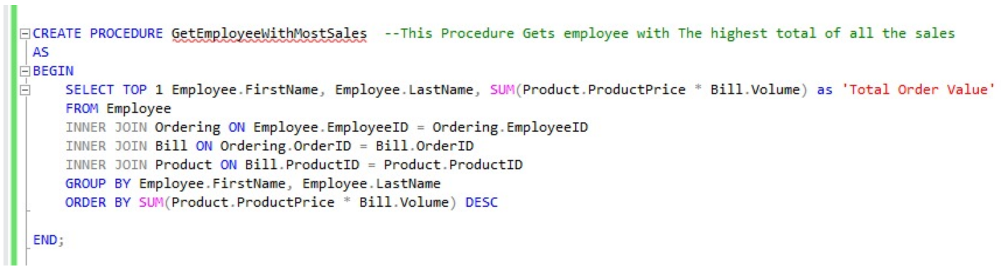


```sql
INSERT INTO Bill (OrderID, Volume)
VALUES ('52' , '4')
```

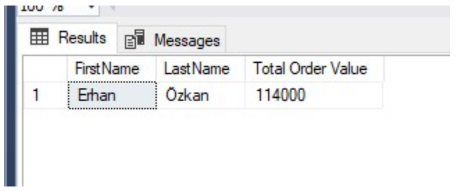
```
(1 row affected)

(1 row affected)

Completion time: 2022-12-30T23:15:08.2706333+03:00
```

## PROCEDURES

**1-)GetEmployeeWithMostSales**: This procedure gets employee with the highest total of all the sales



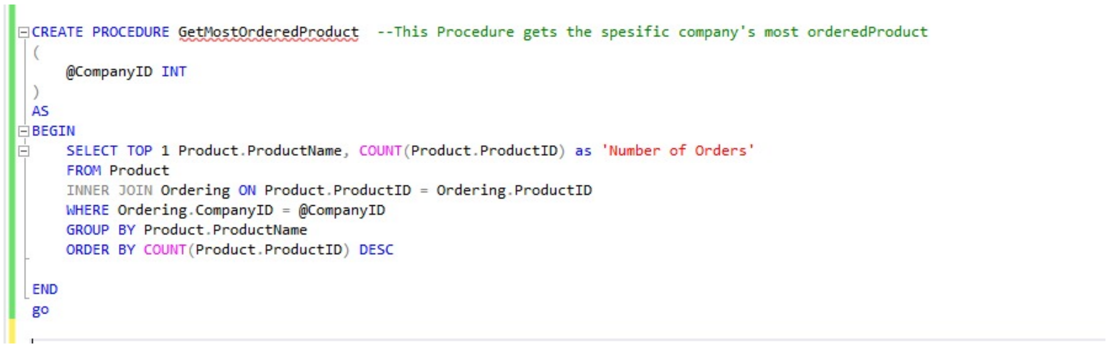
```
CREATE PROCEDURE GetEmployeeWithMostSales  --This Procedure Gets employee with The highest total of all the sales
AS
BEGIN
    SELECT TOP 1 Employee.FirstName, Employee.LastName, SUM(Product.ProductPrice * Bill.Volume) as 'Total Order Value'
    FROM Employee
    INNER JOIN Ordering ON Employee.EmployeeID = Ordering.EmployeeID
    INNER JOIN Bill ON Ordering.OrderID = Bill.OrderID
    INNER JOIN Product ON Bill.ProductID = Product.ProductID
    GROUP BY Employee.FirstName, Employee.LastName
    ORDER BY SUM(Product.ProductPrice * Bill.Volume) DESC

END;
```



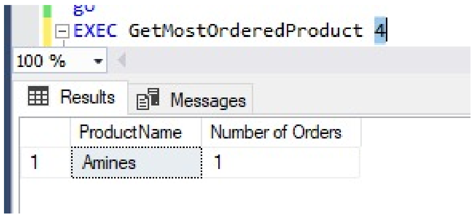
Results | Messages

| | FirstName | LastName | Total Order Value |
|---|---|---|---|
| 1 | Erhan | Özkan | 114000 |

**2-)GetMostOrderedProduct**: This procedure gets the specific company's most ordered product.



```
CREATE PROCEDURE GetMostOrderedProduct  --This Procedure gets the spesific company's most orderedProduct
(
    @CompanyID INT
)
AS
BEGIN
    SELECT TOP 1 Product.ProductName, COUNT(Product.ProductID) as 'Number of Orders'
    FROM Product
    INNER JOIN Ordering ON Product.ProductID = Ordering.ProductID
    WHERE Ordering.CompanyID = @CompanyID
    GROUP BY Product.ProductName
    ORDER BY COUNT(Product.ProductID) DESC

END
go
```



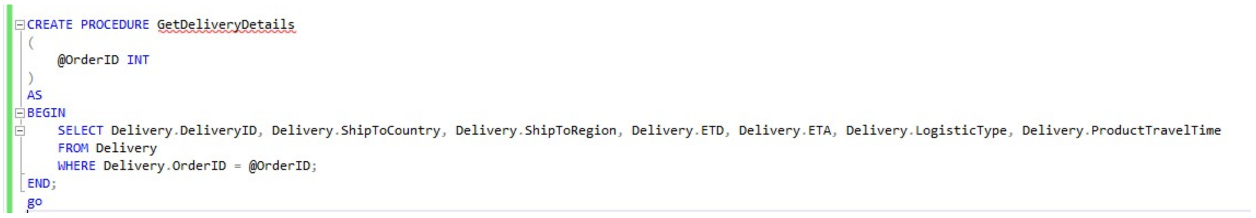
```
EXEC GetMostOrderedProduct 4
```

100 %

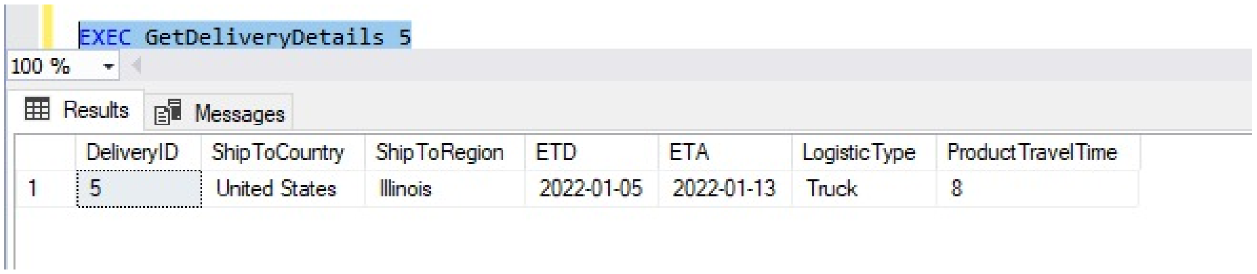
Results | Messages

| | ProductName | Number of Orders |
|---|---|---|
| 1 | Amines | 1 |

**3-)GetDeliveryDetails**: This procedure gets the specific delivery’s details.



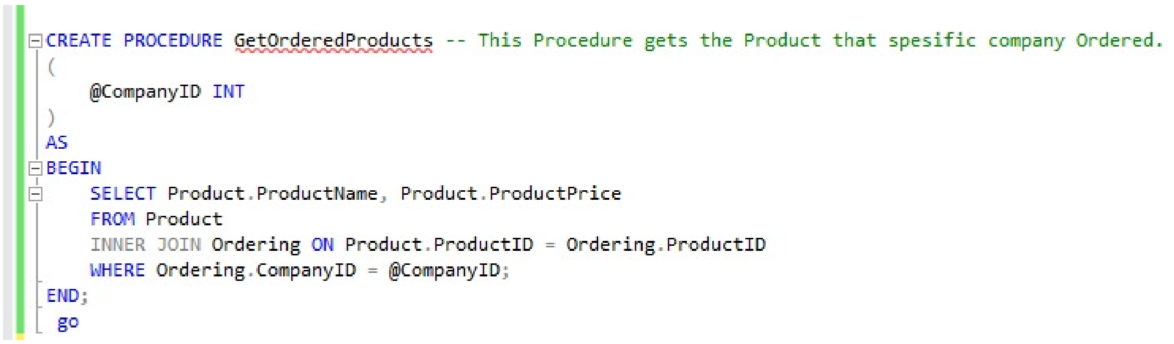
```sql
CREATE PROCEDURE GetDeliveryDetails
(
    @OrderID INT
)
AS
BEGIN
    SELECT Delivery.DeliveryID, Delivery.ShipToCountry, Delivery.ShipToRegion, Delivery.ETD, Delivery.ETA, Delivery.LogisticType, Delivery.ProductTravelTime
    FROM Delivery
    WHERE Delivery.OrderID = @OrderID;
END;
go
```



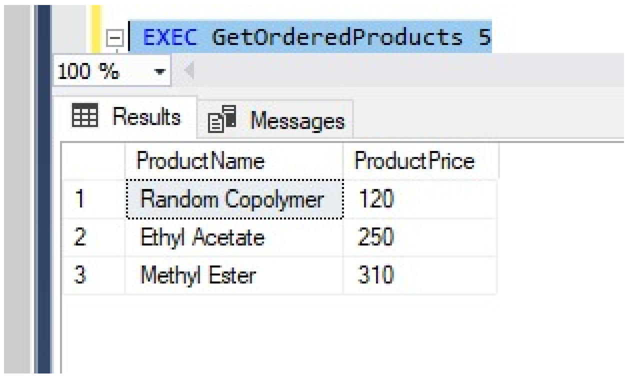
```sql
EXEC GetDeliveryDetails 5
```

| | DeliveryID | ShipToCountry | ShipToRegion | ETD | ETA | LogisticType | ProductTravelTime |
|---|---|---|---|---|---|---|---|
| 1 | 5 | United States | Illinois | 2022-01-05 | 2022-01-13 | Truck | 8 |

**4-)GetOrderedProducts**: This procedure gets the product that a specific company ordered.



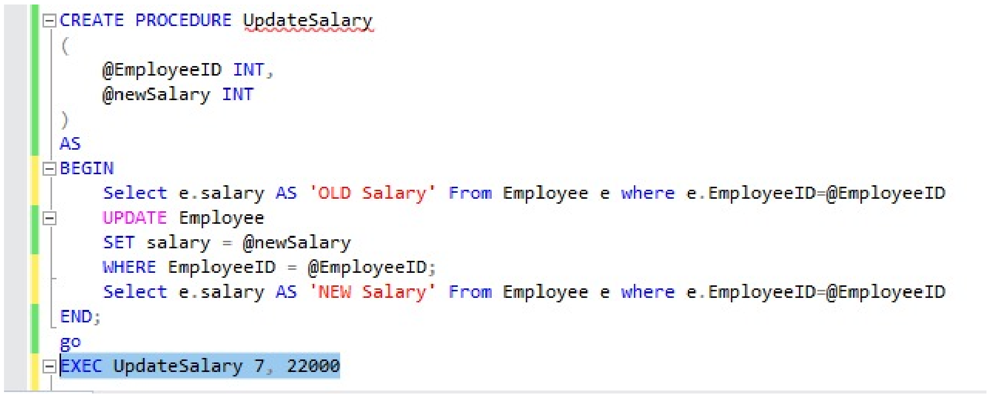
```sql
CREATE PROCEDURE GetOrderedProducts -- This Procedure gets the Product that spesific company Ordered.
(
    @CompanyID INT
)
AS
BEGIN
    SELECT Product.ProductName, Product.ProductPrice
    FROM Product
    INNER JOIN Ordering ON Product.ProductID = Ordering.ProductID
    WHERE Ordering.CompanyID = @CompanyID;
END;
 go
```



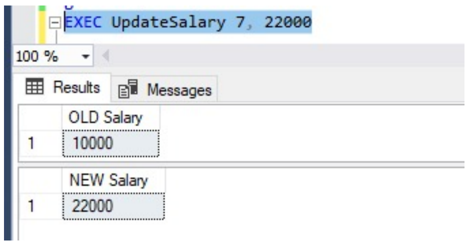
```sql
EXEC GetOrderedProducts 5
```

| | ProductName | ProductPrice |
|---|---|---|
| 1 | Random Copolymer | 120 |
| 2 | Ethyl Acetate | 250 |
| 3 | Methyl Ester | 310 |

**5-)UpdateSalary**: Procedure for updating salary while also showing an old one.



```sql
CREATE PROCEDURE UpdateSalary
(
    @EmployeeID INT,
    @newSalary INT
)
AS
BEGIN
    Select e.salary AS 'OLD Salary' From Employee e where e.EmployeeID=@EmployeeID
    UPDATE Employee
    SET salary = @newSalary
    WHERE EmployeeID = @EmployeeID;
    Select e.salary AS 'NEW Salary' From Employee e where e.EmployeeID=@EmployeeID
END;
go
EXEC UpdateSalary 7, 22000
```



```sql
EXEC UpdateSalary 7, 22000
```

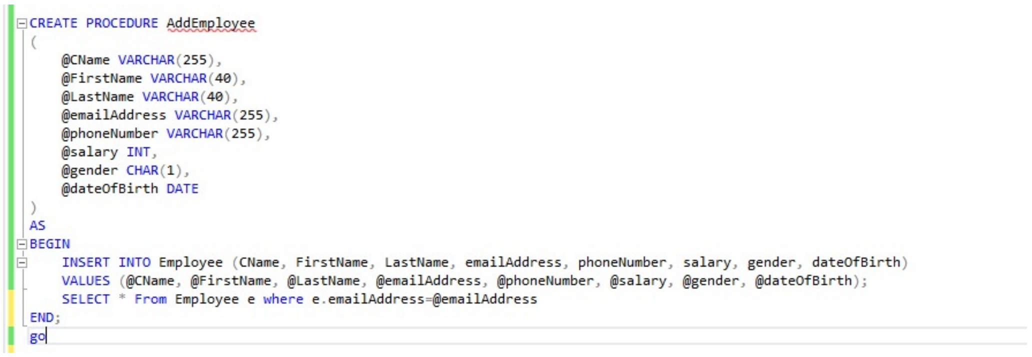
100 %

Results | Messages

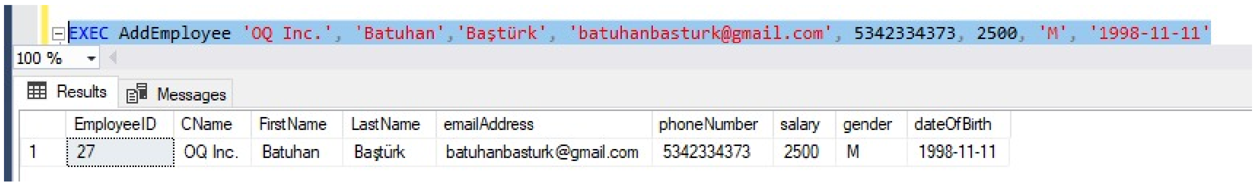
| | OLD Salary |
|---|---|
| 1 | 10000 |

| | NEW Salary |
|---|---|
| 1 | 22000 |

**6-)AddEmployee**: Procedure for inserting employees.



```sql
CREATE PROCEDURE AddEmployee
(
    @CName VARCHAR(255),
    @FirstName VARCHAR(40),
    @LastName VARCHAR(40),
    @emailAddress VARCHAR(255),
    @phoneNumber VARCHAR(255),
    @salary INT,
    @gender CHAR(1),
    @dateOfBirth DATE
)
AS
BEGIN
    INSERT INTO Employee (CName, FirstName, LastName, emailAddress, phoneNumber, salary, gender, dateOfBirth)
    VALUES (@CName, @FirstName, @LastName, @emailAddress, @phoneNumber, @salary, @gender, @dateOfBirth);
    SELECT * From Employee e where e.emailAddress=@emailAddress
END;
go
```



```sql
EXEC AddEmployee 'OQ Inc.', 'Batuhan','Baştürk', 'batuhanbasturk@gmail.com', 5342334373, 2500, 'M', '1998-11-11'
```

100 %

Results | Messages

| | EmployeeID | CName | FirstName | LastName | emailAddress | phoneNumber | salary | gender | dateOfBirth |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 27 | OQ Inc. | Batuhan | Baştürk | batuhanbasturk@gmail.com | 5342334373 | 2500 | M | 1998-11-11 |