P4 IMPLEMENTATION

CLOTHING E-COMMERCE

Based on the final ERD submitted in the P3, we have developed the SQL queries to implement and create various stored procedures, triggers and views.

To sum up the details of the database created, the following table would state the components:

Check Constraints:

Sno.	Table	Primary key	Foreign Key	Check constraint
1.	category	categoryID	-	CHK_categoryID
2.	creditcard	creditCardNoID	customerID	-
3.	customer	customerID	-	CHK_PhoneNo
4.	customerAddress	customerAddressID	customerID	CHK_PostalCode_customerAddress
5.	customerFeedback	FeedbackID	customerID, orderID	CHK_experienceRating, CHK_productRating, CHK_shippingRating
6.	order	orderID	customerID	CHK_orderID
7.	orderDetailsID	orderDetailsID	orderID, productID	CHK_orderDetailsID
8.	payment	paymentID	orderID, creditCardNoID	CHK_paymentID
9.	product	productID	categoryID	CHK_productID
10.	productStock	productStockID	productID	CHK_productStock
11.	shipment	shipmentID	customerAddressID, orderID	CHK_shipmentID
12.	shipper	shipperID	shipmentID	CHK_shipperID
13.	supplier	supplierID	-	CHK_supplierID
14.	supplierAddress	supplierAddressID	supplierID	CHK_supplierAddressID
15.	supplies	supplierID, productID	supplierID, productID	-

Data encryption:

Sno.	Table	Column Encrypted
1.	creditcard	creditCardNo

Non-Clustered Indexes:

S.No	Table	Non-Clustered Indexes
1.	creditcard	prim_key_creditcardID
2.	payment	prim_key_payment
3.	productStock	prim_key_productStock
4.	shipper	prim_key_shipperID

Procedures, views, trigger created:

S.No	Category created	Name
1.	User-Defined Function	CustomerFullName
2.	User-Defined Function	SupplierFullName
3.	User-Defined Function	GetOrderTotal
4.	Procedure	GetCustomerFeedbackWith
5.	Procedure	GetCustomerInformation
6.	Procedure	GetPaymentStatusWith
7.	Procedure	updateProductPrice
8.	Procedure	GetCustomerIDandOrderIDWithExperienceRating
9.	Trigger	CheckProductPriceChanges
10.	View	CustomersAndTheirCreditCards
11.	View	ShowSupplierInfoAndTheirAddresses
12.	View	ViewProductInfoWithUnits

SQL QUERIES

CREATING A DATABASE:

SQL Query:

```
CREATE DATABASE [DMDDP4]

GO

USE DMDDP4

GO
```

CREATING TABLES:

Table : category

```
/* CREATE table category */
CREATE TABLE [dbo].[category] (
  [categoryID] int NOT NULL,
  [categoryName] varchar(30) NOT NULL,
  [categoryDescription] varchar(200) NOT NULL,
  [categoryPicture] varbinary(max),
  CONSTRAINT prim_Key PRIMARY KEY CLUSTERED ([categoryID] ASC),
  )
ON [PRIMARY]

--Add a CHECK FOR Category Table --
ALTER TABLE [dbo].[category] ADD CONSTRAINT CHK_CategoryID CHECK (categoryID > 0 );
GO
```

Table : customer

```
/* CREATE table customer */
CREATE TABLE [dbo].[customer] (
  [customerID] int NOT NULL,
  [customerFirstName] varchar(45) NOT NULL,
  [customerLastName] varchar(45) NOT NULL,
  [customerPhoneNo] varchar(45),
  [customerEmail] varchar(45) NOT NULL

  CONSTRAINT prim_Key_customer PRIMARY KEY CLUSTERED ([customerID] ASC),
  )
ON [PRIMARY]

--Add a CHECK for CustomerPhoneNo in customer Table --
ALTER TABLE [dbo].[customer]  WITH CHECK ADD  CONSTRAINT CHK_PhoneNo
CHECK (customerPhoneNo NOT LIKE '%[^0-9]%')
```

Table: creditcard

```
/* CREATE table creditcard */
CREATE TABLE [dbo].[creditcard] (
  [creditCardNoID] int IDENTITY (1,1),
  [creditCardNo] varchar(100) NOT NULL,
  [customerID] int NOT NULL,
  [SetAsPrimary] varchar(20) NOT NULL,
  [creditCardType] varchar(20),
  [cardExpiry] varchar(20),
  CONSTRAINT prim_key_creditcard PRIMARY KEY NONCLUSTERED ([creditCardNoID])
  )
ON [PRIMARY]

-- Add and CHECK Constraint FOREIGN KEY for CreditCard Table --
ALTER TABLE [dbo].[creditcard] WITH CHECK ADD CONSTRAINT foreign_CustomerID_CreditCard
FOREIGN KEY ([customerID]) REFERENCES [dbo].[customer] ([customerID])
```

Table: customerAddress

```
/* CREATE table customerAddress */
CREATE TABLE [dbo].[customerAddress] (
  [customerAddressID] int NOT NULL,
  [customerID] int NOT NULL,
  [street] varchar(20) NOT NULL,
  [city] varchar(20) NOT NULL,
  [PostalCode] varchar(20) NOT NULL,
  [useAsBillingAddress] varchar(20) NOT NULL

  CONSTRAINT prim_Key_customerAddress PRIMARY KEY CLUSTERED ([customerAddressID] ASC),
  )
ON [PRIMARY]

-- Add a CHECK CONSTRAINT FOREIGN KEYS for CustomerAddress Table --
ALTER TABLE [dbo].[customerAddress]  WITH CHECK ADD CONSTRAINT foreign_customerAddress
FOREIGN KEY ([customerID]) REFERENCES [dbo].[customer] ([customerID])

-- Add CHECK CONSTRAINT for PHONE No in CustomerFeedbackTable --
ALTER TABLE [dbo].[customerAddress]  WITH CHECK ADD CONSTRAINT
CHK_PostalCode_customerAddress
CHECK (PostalCode NOT LIKE '%[^0-9]%')
```

Table: order

```
/* CREATE table order */
CREATE TABLE [dbo].[order] (
  [orderID] int NOT NULL,
  [orderDate] date NOT NULL,
  [customerID] int NOT NULL,
  [orderTotal] varchar(20) NOT NULL,
  [orderTime] time
```

```
  CONSTRAINT prim_Key_order PRIMARY KEY CLUSTERED ([orderID] ASC),
  )
ON [PRIMARY]

-- Add CHECK CONSTRAINT FOREIGN KEYS for order Table--
ALTER TABLE [dbo].[order] WITH CHECK ADD CONSTRAINT foreign_key_customer_order
FOREIGN KEY ([customerID]) REFERENCES [dbo].[customer] ([customerID])


--Add a CHECK for OrderID in order Table --
ALTER TABLE [dbo].[order] WITH CHECK ADD CONSTRAINT CHK_orderID CHECK (orderID > 0);
```

Table: shipment

```
/* CREATE table shipment */
CREATE TABLE [dbo].[shipment] (
  [shipmentID] int NOT NULL,
  [orderID] int NOT NULL,
  [customerAddressID] int NOT NULL,
  [shippingDate] date NOT NULL,
  [shippingMethod] varchar(50) NOT NULL

  CONSTRAINT prim_Key_shipment PRIMARY KEY CLUSTERED ([shipmentID] ASC),
  )
ON [PRIMARY]

-- Add CHECK CONSTRAINT FOREIGN KEY for shipment Table--
ALTER TABLE [dbo].[shipment] WITH CHECK ADD CONSTRAINT foreign_shipment
FOREIGN KEY ([customerAddressID]) REFERENCES [dbo].[customerAddress]
([customerAddressID])

ALTER TABLE [dbo].[shipment] WITH CHECK ADD CONSTRAINT foreign_key_shipment_order
FOREIGN KEY ([orderID]) REFERENCES [dbo].[order] ([orderID])

--Add a CHECK for shipmentID in shipment Table --
ALTER TABLE [dbo].[shipment] WITH CHECK ADD  CONSTRAINT CHK_shipmentID CHECK (shipmentID
> 0 );
```

Table: customerfeedback

```
/* CREATE table customerFeedback */
CREATE TABLE [dbo].[customerFeedback] (
  [FeedbackID] int NOT NULL,
  [customerID] int NOT NULL,
  [orderID] int NOT NULL,
  [productRating] decimal(2,1),
  [shippingRating]  decimal(2,1),
  [experienceRating] decimal(2,1)

  CONSTRAINT prim_Key_customerFeedback PRIMARY KEY NONCLUSTERED ([FeedbackID] ASC),
  )
ON [PRIMARY]

-- Add CHECK CONSTRAINT FOREIGN KEYS for CustomerFeedback Table--
```

```sql
ALTER TABLE [dbo].[customerFeedback]  WITH CHECK ADD CONSTRAINT
foreign_key_customer_customerFeedback
FOREIGN KEY ([customerID]) REFERENCES [dbo].[customer] ([customerID])

ALTER TABLE [dbo].[customerFeedback]  WITH CHECK ADD CONSTRAINT
foreign_key_order_customerFeedback
FOREIGN KEY ([orderID]) REFERENCES [dbo].[order] ([orderID])

-- Add CHECK CONSTRAINT for COLUMN VALUES in CustomerFeedbackTable --
ALTER TABLE [dbo].[customerFeedback] WITH CHECK ADD  CONSTRAINT CHK_productRating
CHECK ([productRating] > 0 AND [productRating] <= 5);

ALTER TABLE [dbo].[customerFeedback] WITH CHECK ADD  CONSTRAINT CHK_shippingRating
CHECK ([shippingRating] > 0 AND [shippingRating] <= 5);

ALTER TABLE [dbo].[customerFeedback] WITH CHECK ADD  CONSTRAINT CHK_experienceRating
CHECK ([experienceRating] > 0 AND [experienceRating] <= 5); ------------------------------------
-----------------------------------------------------------------------------
```

Table: product

```sql
/* CREATE table product */
CREATE TABLE [dbo].[product] (
  [productID] int NOT NULL,
  [categoryID] int NOT NULL,
  [productName] varchar(45) NOT NULL,
  [productPrice] int,
  [productColor] varchar(20),
  [productSize] varchar(20),
  [discount] varchar(20),
  [productWeight] varchar(20),
  [productPicture] varbinary(max),
  [productDescription] varchar(200)

  CONSTRAINT prim_Key_product PRIMARY KEY CLUSTERED ([productID] ASC),
  )
ON [PRIMARY]

-- Add CHECK CONSTRAINT FOREIGN KEY for product Table--
ALTER TABLE [dbo].[product] WITH CHECK ADD  CONSTRAINT foreign_key_categoryID
FOREIGN KEY ([categoryID]) REFERENCES [dbo].[category] ([categoryID])

--Add a CHECK for productID in product Table --
ALTER TABLE [dbo].[product] WITH CHECK ADD CONSTRAINT CHK_productID CHECK (productID > 0
);
```

Table: orderdetails

```sql
/* CREATE table orderDetails */
CREATE TABLE [dbo].[orderDetails] (
  [orderDetailsID] int NOT NULL,
  [orderID] int NOT NULL,
  [productID] int NOT NULL,
```

```
  [orderQuantity] varchar(20),
  [fulfillmentStatus] varchar(20)

  CONSTRAINT prim_Key_orderDetails PRIMARY KEY CLUSTERED ([orderDetailsID] ASC),
  )
ON [PRIMARY]

-- Add CHECK CONSTRAINT FOREIGN KEYS for orderDetails Table--
ALTER TABLE [dbo].[orderDetails] WITH CHECK ADD  CONSTRAINT
foreign_key_orderID_orderDetails
FOREIGN KEY ([orderID]) REFERENCES [dbo].[order] ([orderID])

ALTER TABLE [dbo].[orderDetails] WITH CHECK ADD  CONSTRAINT
foreign_key_productID_orderDetails
FOREIGN KEY ([productID]) REFERENCES [dbo].[product] ([productID])

--Add a CHECK for orderDetailsID in orderDetails Table --
ALTER TABLE [dbo].[orderDetails] WITH CHECK ADD CONSTRAINT CHK_orderDetailsID CHECK
(orderDetailsID > 0 ); ----------------------------------------------------------------------------------------
-----------------------------
```

Table: payment

```
/* CREATE table payment */
CREATE TABLE [dbo].[payment] (
  [paymentID] int NOT NULL,
  [orderID] int NOT NULL,
  [paymentMethod] varchar(30) NOT NULL,
  [paymentStatus] varchar(20),
  [paymentDate] date,
  [paymentTime] time,
  [paymentError] varchar(20),
  [creditCardNoID] int NOT NULL

  CONSTRAINT prim_Key_payment PRIMARY KEY NONCLUSTERED ([paymentID] ASC),
  )
ON [PRIMARY]

-- Add CHECK CONSTRAINT FOREIGN KEY for payment Table--
ALTER TABLE [dbo].[payment] WITH CHECK ADD  CONSTRAINT foreign_key_orderID_payment
FOREIGN KEY ([orderID]) REFERENCES [dbo].[order] ([orderID])

ALTER TABLE [dbo].[payment] WITH CHECK ADD CONSTRAINT foreign_key_creditCardNoID_payment
FOREIGN KEY (creditCardNoID) REFERENCES [dbo].[creditCard] ([creditCardNoID])

--Add a CHECK for paymentID in payment Table --
ALTER TABLE [dbo].[payment] WITH CHECK ADD CONSTRAINT CHK_paymentID CHECK (paymentID > 0)
 ----------------------------------------------------------------------------------------------------------------
-
```

Table: ProductStock

```
/* CREATE table productStock */
CREATE TABLE [dbo].[productStock] (
  [productStockID] int NOT NULL,
  [productID] int NOT NULL,
  [unitsInStock] varchar(20),
```

```
  [unitsInOrder] varchar(20)

  CONSTRAINT prim_Key_productStockID PRIMARY KEY NONCLUSTERED ([productStockID]),
  )
ON [PRIMARY]

-- Add CHECK CONSTRAINT FOREIGN KEY for productStock Table--
ALTER TABLE [dbo].[productStock]  WITH CHECK ADD CONSTRAINT
foreign_key_productID_productStock
FOREIGN KEY ([productID]) REFERENCES [dbo].[product] ([productID])

--Add a CHECK for productStockID in productStock Table --
ALTER TABLE [dbo].[productStock] WITH CHECK ADD CONSTRAINT CHK_productStock CHECK
(productStockID > 0) ----------------------------------------------------------------------------------------
--------------------------
```

Table: shipper

```
/* CREATE table shipper */
CREATE TABLE [dbo].[shipper] (
  [shipperID] int NOT NULL,
  [shipmentID] int NOT NULL,
  [shipperName] varchar(45),
  [shipperPhoneNo] varchar(20)

  CONSTRAINT prim_Key_shipperID PRIMARY KEY NONCLUSTERED ([shipperID]),
  )
ON [PRIMARY]

-- Add CHECK CONSTRAINT FOREIGN KEY for shipper Table--
ALTER TABLE [dbo].[shipper] WITH CHECK ADD CONSTRAINT foreign_key_shipmentID_shipper
FOREIGN KEY ([shipmentID]) REFERENCES [dbo].[shipment] ([shipmentID])

--Add a CHECK for shipperID in shipper Table --
ALTER TABLE [dbo].[shipper]  WITH CHECK ADD CONSTRAINT CHK_shipperID CHECK (shipperID >
0)
 ----------------------------------------------------------------------------------------------------------
```

Table: supplier

```
/* CREATE table supplier */
CREATE TABLE [dbo].[supplier] (
  [supplierID] int NOT NULL,
  [supplierFirstName] varchar(20) NOT NULL,
  [supplierLastName] varchar(20),
  [supplierPhoneNo] varchar(20),
  [supplierEmail] varchar(45),
  [supplierURL] varchar(45),
  [supplierDescription] varchar(200),

  CONSTRAINT prim_Key_supplierID PRIMARY KEY CLUSTERED ([supplierID] ASC),
  )
ON [PRIMARY]

--Add a CHECK for supplierID in supplier Table --
ALTER TABLE [dbo].[supplier]  WITH CHECK ADD CONSTRAINT CHK_supplierID CHECK (supplierID
> 0)
```

Table: supplierAddress

```
/* CREATE table supplierAddress */
CREATE TABLE [dbo].[supplierAddress] (
  [supplierAddressID] int NOT NULL,
  [supplierID] int NOT NULL,
  [street] varchar(20) NOT NULL,
  [city] varchar(20),
  [postalCode] varchar(20),

  CONSTRAINT prim_Key_supplierAddressID PRIMARY KEY NONCLUSTERED ([supplierAddressID]),
  )
ON [PRIMARY]

-- Add CHECK CONSTRAINT FOREIGN KEY for supplierAddress Table--
ALTER TABLE [dbo].[supplierAddress] WITH CHECK ADD CONSTRAINT
foreign_key_supplierID_supplierAddress
FOREIGN KEY ([supplierID]) REFERENCES [dbo].[supplier] ([supplierID])

--Add a CHECK for supplieraddressID in supplierAddress Table --
ALTER TABLE [dbo].[supplierAddress] WITH CHECK ADD CONSTRAINT CHK_supplierAddressID CHECK
(supplierAddressID > 0)
```

Table: supplies

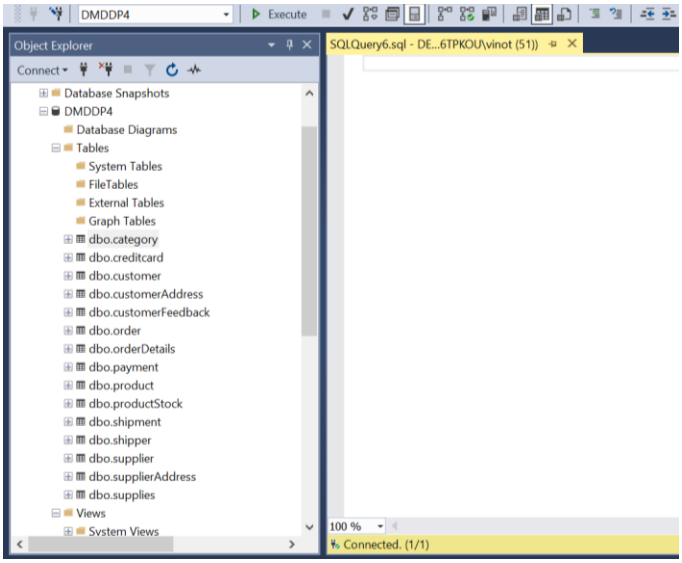
```
/* CREATE table supplies */
CREATE TABLE [dbo].[supplies] (
  [supplierID] int NOT NULL,
  [productID] int NOT NULL

  CONSTRAINT prim_Key_supplies PRIMARY KEY CLUSTERED ([supplierID],[productID]),
  )
ON [PRIMARY]

-- Add CHECK CONSTRAINT FOREIGN KEY for supplies Table--
ALTER TABLE [dbo].[supplies]  WITH CHECK ADD CONSTRAINT foreign_key_supplierID_supplies
FOREIGN KEY ([supplierID]) REFERENCES [dbo].[supplier] ([supplierID])

ALTER TABLE [dbo].[supplies]  WITH CHECK ADD CONSTRAINT foreign_key_productID_supplies
FOREIGN KEY ([productID]) REFERENCES [dbo].[product] ([productID])
```

Result: **Displaying the names of the table in the created database DMDDP4**



INSERTING THE DATA INTO THE CREATED TABLES OF THE DATABASE DMDDP4

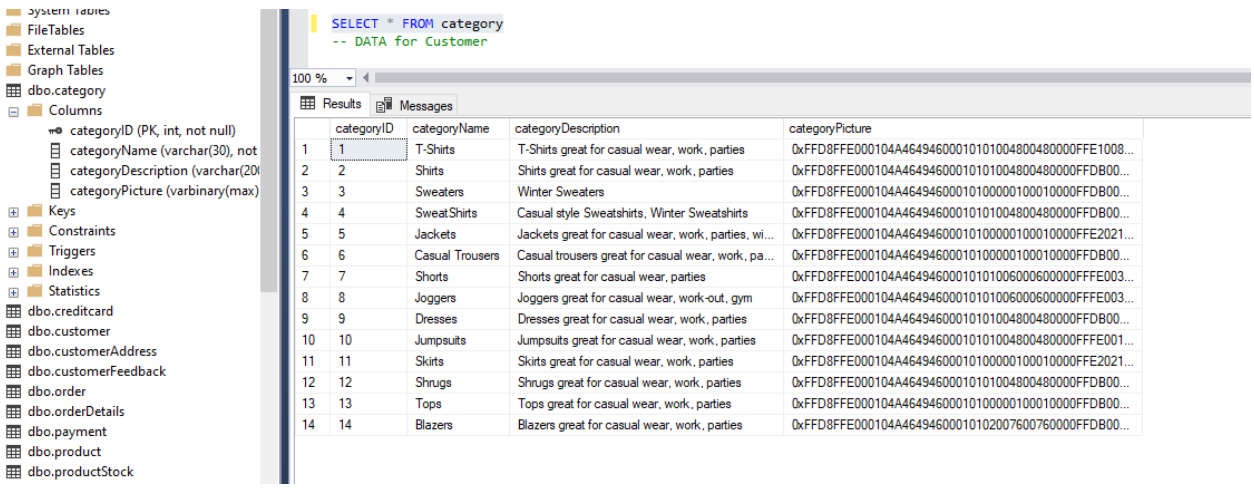
Inserting into category table:

```
-- DATA for category


INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
VALUES (1,'T-Shirts','T-Shirts great for casual wear, work, parties', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\T-shirt.jpg', SINGLE_BLOB)image));
INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
VALUES(2,'Shirts','Shirts great for casual wear, work, parties', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\Shirt.jpg', SINGLE_BLOB)image));
INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
VALUES(3,'Sweaters','Winter Sweaters', (SELECT * FROM OPENROWSET (BULK N'D:Saved
Pictures\Sweater.jpg', SINGLE_BLOB)image));
INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
VALUES(4,'SweatShirts','Casual style Sweatshirts, Winter Sweatshirts', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\Sweat shirt.jpg', SINGLE_BLOB)image));
INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
VALUES(5,'Jackets','Jackets great for casual wear, work, parties, winter', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\Jacket.jpg', SINGLE_BLOB)image));
INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
VALUES(6,'Casual Trousers','Casual trousers great for casual wear, work, parties',
(SELECT * FROM OPENROWSET (BULK N'D:Saved Pictures\Trousers.jpg', SINGLE_BLOB)image));
INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
VALUES(7,'Shorts','Shorts great for casual wear, parties', (SELECT * FROM OPENROWSET
(BULK N'D:Saved Pictures\Shorts.jpg', SINGLE_BLOB)image));
INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
VALUES(8,'Joggers','Joggers great for casual wear, work-out, gym', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\Joggers.jpg', SINGLE_BLOB)image));
INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
```

```
VALUES(9,'Dresses','Dresses great for casual wear, work, parties', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\Dresses.jpg', SINGLE_BLOB)image));
INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
VALUES(10,'Jumpsuits','Jumpsuits great for casual wear, work, parties', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\Jumpsuits.jpg', SINGLE_BLOB)image));
INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
VALUES(11,'Skirts','Skirts great for casual wear, work, parties', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\Skirts.jpg', SINGLE_BLOB)image));
INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
VALUES(12,'Shrugs','Shrugs great for casual wear, work, parties', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\Shrugs.jpg', SINGLE_BLOB)image));
INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
VALUES(13,'Tops','Tops great for casual wear, work, parties', (SELECT * FROM OPENROWSET
(BULK N'D:Saved Pictures\Tops.jpg', SINGLE_BLOB)image));
INSERT INTO category (categoryID, categoryName, categoryDescription, categoryPicture)
VALUES (14,'Blazers','Blazers great for casual wear, work, parties', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\Blazers.jpg', SINGLE_BLOB)image));
```

Displaying category table: The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



	categoryID	categoryName	categoryDescription	categoryPicture
1	1	T-Shirts	T-Shirts great for casual wear, work, parties	0xFFD8FFE000104A4649460001010100480048000 0FFE1008...
2	2	Shirts	Shirts great for casual wear, work, parties	0xFFD8FFE000104A4649460001010100480048000 0FFDB00...
3	3	Sweaters	Winter Sweaters	0xFFD8FFE000104A4649460001010000010001000 0FFDB00...
4	4	SweatShirts	Casual style Sweatshirts, Winter Sweatshirts	0xFFD8FFE000104A4649460001010100480048000 0FFDB00...
5	5	Jackets	Jackets great for casual wear, work, parties, wi...	0xFFD8FFE000104A4649460001010000010001000 0FFE2021...
6	6	Casual Trousers	Casual trousers great for casual wear, work, pa...	0xFFD8FFE000104A4649460001010000010001000 0FFDB00...
7	7	Shorts	Shorts great for casual wear, parties	0xFFD8FFE000104A4649460001010100600060000 0FFFE003...
8	8	Joggers	Joggers great for casual wear, work-out, gym	0xFFD8FFE000104A4649460001010100600060000 0FFFE003...
9	9	Dresses	Dresses great for casual wear, work, parties	0xFFD8FFE000104A4649460001010100480048000 0FFDB00...
10	10	Jumpsuits	Jumpsuits great for casual wear, work, parties	0xFFD8FFE000104A4649460001010100480048000 0FFFE001...
11	11	Skirts	Skirts great for casual wear, work, parties	0xFFD8FFE000104A4649460001010000010001000 0FFE2021...
12	12	Shrugs	Shrugs great for casual wear, work, parties	0xFFD8FFE000104A4649460001010100480048000 0FFDB00...
13	13	Tops	Tops great for casual wear, work, parties	0xFFD8FFE000104A4649460001010000010001000 0FFDB00...
14	14	Blazers	Blazers great for casual wear, work, parties	0xFFD8FFE000104A4649460001010200760076000 0FFDB00...

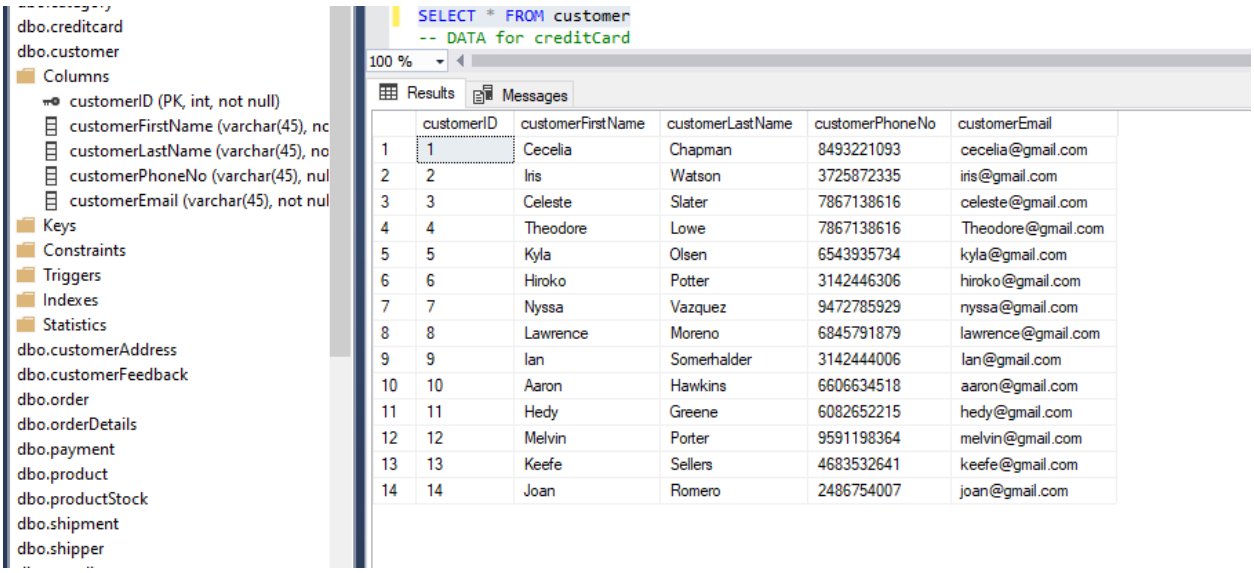
Inserting into Customer table:

```
-- DATA for Customer


INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
VALUES (1,'Cecelia','Chapman','8493221093','cecelia@gmail.com');
INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
VALUES (2,'Iris','Watson','3725872335','iris@gmail.com');
INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
VALUES(3,'Celeste','Slater','7867138616','celeste@gmail.com');
INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
```

```
VALUES(4,'Theodore','Lowe','7867138616','Theodore@gmail.com');
INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
VALUES(5,'Kyla','Olsen','6543935734','kyla@gmail.com');
INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
VALUES(6,'Hiroko','Potter','3142446306','hiroko@gmail.com');
INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
VALUES(7,'Nyssa','Vazquez','9472785929','nyssa@gmail.com');
INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
VALUES(8,'Lawrence','Moreno','6845791879','lawrence@gmail.com');
INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
VALUES(9,'Ian','Somerhalder','3142444006','Ian@gmail.com');
INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
VALUES(10,'Aaron','Hawkins','6606634518','aaron@gmail.com');
INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
VALUES(11,'Hedy','Greene','6082652215','hedy@gmail.com');
INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
VALUES(12,'Melvin','Porter','9591198364','melvin@gmail.com');
INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
VALUES(13,'Keefe','Sellers','4683532641','keefe@gmail.com');
INSERT INTO Customer (customerID, CustomerFirstName, CustomerLastName, CustomerPhoneNo,
CustomerEmail)
VALUES(14,'Joan','Romero','2486754007','joan@gmail.com');
```

Displaying customer table: The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



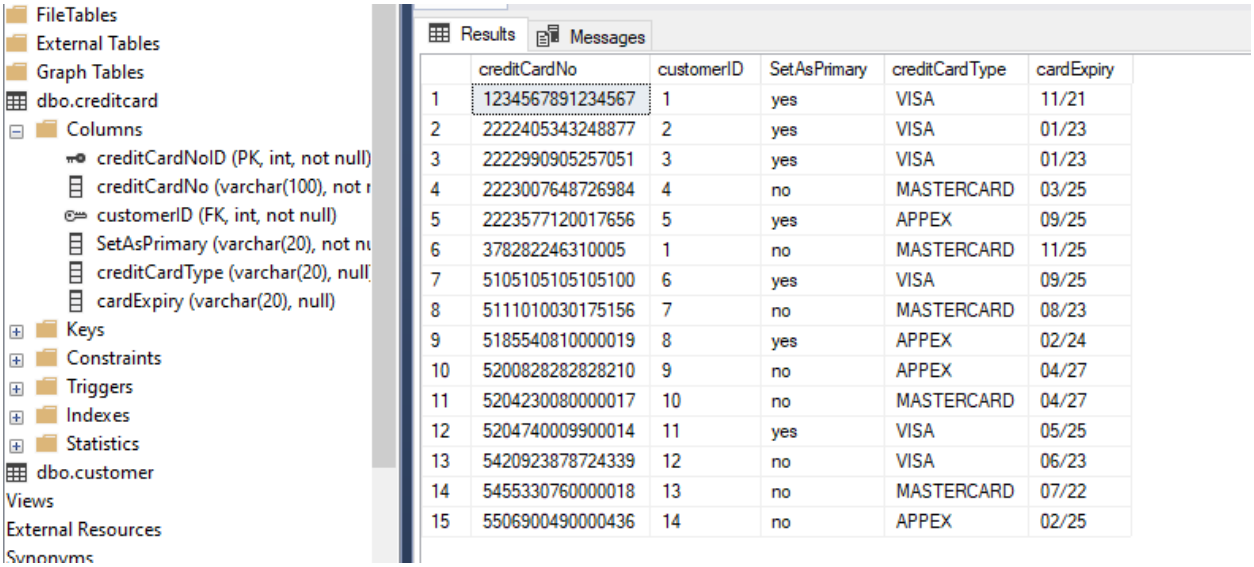
	customerID	customerFirstName	customerLastName	customerPhoneNo	customerEmail
1	1	Cecelia	Chapman	8493221093	cecelia@gmail.com
2	2	Iris	Watson	3725872335	iris@gmail.com
3	3	Celeste	Slater	7867138616	celeste@gmail.com
4	4	Theodore	Lowe	7867138616	Theodore@gmail.com
5	5	Kyla	Olsen	6543935734	kyla@gmail.com
6	6	Hiroko	Potter	3142446306	hiroko@gmail.com
7	7	Nyssa	Vazquez	9472785929	nyssa@gmail.com
8	8	Lawrence	Moreno	6845791879	lawrence@gmail.com
9	9	Ian	Somerhalder	3142444006	Ian@gmail.com
10	10	Aaron	Hawkins	6606634518	aaron@gmail.com
11	11	Hedy	Greene	6082652215	hedy@gmail.com
12	12	Melvin	Porter	9591198364	melvin@gmail.com
13	13	Keefe	Sellers	4683532641	keefe@gmail.com
14	14	Joan	Romero	2486754007	joan@gmail.com

Inserting into CreditCard table:

```
-- DATA for creditCard


INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES ('1234567891234567', 1,'yes','VISA','11/21');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('2222405343248877', 2,'yes','VISA','01/23');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('2222990905257051', 3,'yes','VISA','01/23');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('2223007648726984', 4,'no','MASTERCARD','03/25');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('2223577120017656', 5,'yes','APPEX','09/25');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('378282246310005', 1,'no','MASTERCARD','11/25');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('5105105105105100', 6,'yes','VISA','09/25');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('5111010030175156', 7,'no','MASTERCARD','08/23');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('5185540810000019', 8,'yes','APPEX','02/24');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('5200828282828210', 9,'no','APPEX','04/27');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('5204230080000017', 10,'no','MASTERCARD','04/27');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('5204740009900014', 11,'yes','VISA','05/25');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('5420923878724339', 12,'no','VISA','06/23');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('5455330760000018', 13,'no','MASTERCARD','07/22');
INSERT INTO creditCard (creditCardNo, customerID, setAsPrimary, creditCardType,
cardExpiry)
VALUES('5506900490000436', 14,'no','APPEX','02/25');
```

Displaying CreditCard table: The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



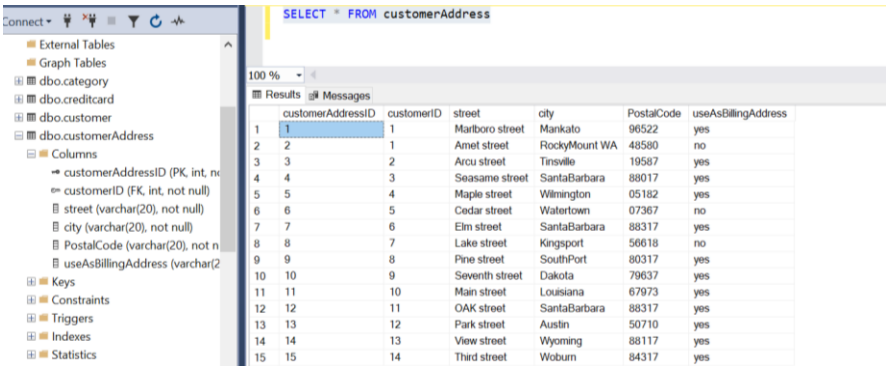
	creditCardNo	customerID	SetAsPrimary	creditCardType	cardExpiry
1	1234567891234567	1	yes	VISA	11/21
2	2222405343248877	2	yes	VISA	01/23
3	2222990905257051	3	yes	VISA	01/23
4	2223007648726984	4	no	MASTERCARD	03/25
5	2223577120017656	5	yes	APPEX	09/25
6	378282246310005	1	no	MASTERCARD	11/25
7	5105105105105100	6	yes	VISA	09/25
8	5111010030175156	7	no	MASTERCARD	08/23
9	5185540810000019	8	yes	APPEX	02/24
10	5200828282828210	9	no	APPEX	04/27
11	5204230080000017	10	no	MASTERCARD	04/27
12	5204740009900014	11	yes	VISA	05/25
13	5420923878724339	12	no	VISA	06/23
14	5455330760000018	13	no	MASTERCARD	07/22
15	5506900490000436	14	no	APPEX	02/25

Inserting into customerAddress table:

```
-- DATA for customerAddress


INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (1, 1,'Marlboro street','Mankato','96522','yes');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (2, 1,'Amet street','RockyMount WA','48580','no');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (3, 2,'Arcu street','Tinsville','19587','yes');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (4, 3,'Seasame street','SantaBarbara','88017','yes');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (5, 4,'Maple street','Wilmington','05182','yes');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (6, 5,'Cedar street','Watertown','07367','no');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (7, 6,'Elm street','SantaBarbara','88317','yes');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (8, 7,'Lake street','Kingsport','56618','no');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (9, 8,'Pine street','SouthPort','80317','yes');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (10, 9,'Seventh street','Dakota','79637','yes');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (11, 10,'Main street','Louisiana','67973','yes');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (12, 11,'OAK street','SantaBarbara','88317','yes');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (13, 12,'Park street','Austin','50710','yes');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (14, 13,'View street','Wyoming','88117','yes');
INSERT INTO customerAddress (customerAddressID, customerID, street, city, postalCode,
useAsBillingAddress) VALUES (15, 14,'Third street','Woburn','84317','yes');
```

Displaying customerAddress table: The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



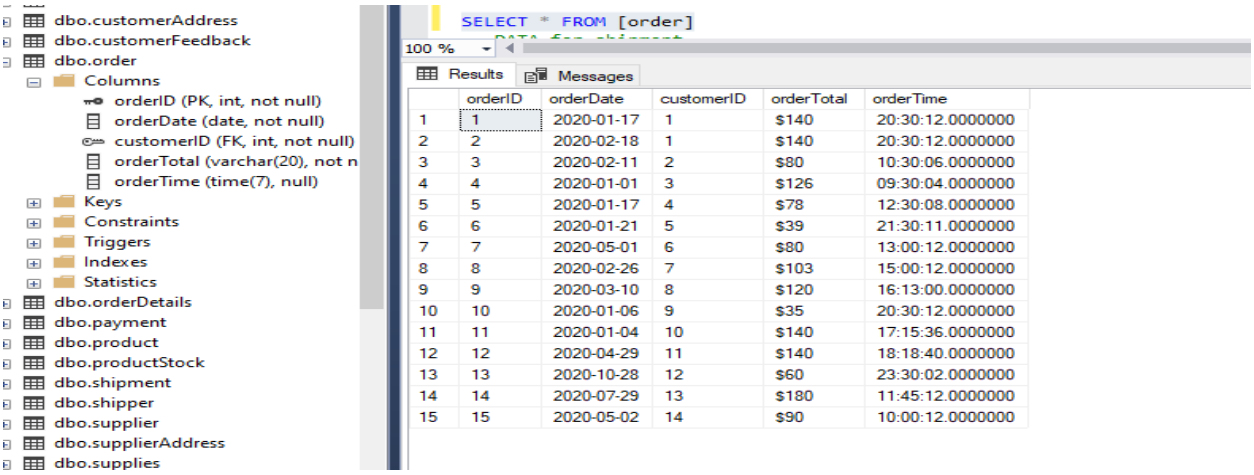
	customerAddressID	customerID	street	city	PostalCode	useAsBillingAddress
1	1	1	Marlboro street	Mankato	96522	yes
2	2	1	Amet street	RockyMount WA	48580	no
3	3	2	Arcu street	Tinsville	19587	yes
4	4	3	Seasame street	SantaBarbara	88017	yes
5	5	4	Maple street	Wilmington	05182	yes
6	6	5	Cedar street	Watertown	07367	no
7	7	6	Elm street	SantaBarbara	88317	yes
8	8	7	Lake street	Kingsport	56618	no
9	9	8	Pine street	SouthPort	80317	yes
10	10	9	Seventh street	Dakota	79637	yes
11	11	10	Main street	Louisiana	67973	yes
12	12	11	OAK street	SantaBarbara	88317	yes
13	13	12	Park street	Austin	50710	yes
14	14	13	View street	Wyoming	88117	yes
15	15	14	Third street	Woburn	84317	yes

Inserting into order:

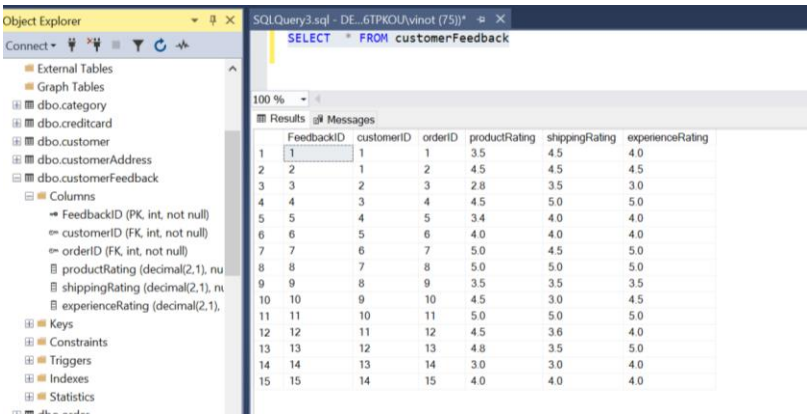
```
-- DATA for Order


INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (1,
1,'01/17/2020','$140','20:30:12');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (2,
1,'02/18/2020','$140','20:30:12');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (3,
2,'02/11/2020','$80','10:30:06');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (4,
3,'01/01/2020','$126','09:30:04');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (5,
4,'01/17/2020','$78','12:30:08');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (6,
5,'01/21/2020','$39','21:30:11');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (7,
6,'05/01/2020','$80','13:00:12');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (8,
7,'02/26/2020','$103','15:00:12');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (9,
8,'03/10/2020','$120','16:13:00');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (10,
9,'01/06/2020','$35','20:30:12');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (11,
10,'01/04/2020','$140','17:15:36');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (12,
11,'04/29/2020','$140','18:18:40');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (13,
12,'10/28/2020','$60','23:30:02');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (14,
13,'07/29/2020','$180','11:45:12');
INSERT INTO [Order] (orderID, customerID, orderDate, orderTotal, orderTime) VALUES (15,
14,'05/02/2020','$90','10:00:12');
```

Displaying order table: The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



	orderID	orderDate	customerID	orderTotal	orderTime
1	1	2020-01-17	1	$140	20:30:12.0000000
2	2	2020-02-18	1	$140	20:30:12.0000000
3	3	2020-02-11	2	$80	10:30:06.0000000
4	4	2020-01-01	3	$126	09:30:04.0000000
5	5	2020-01-17	4	$78	12:30:08.0000000
6	6	2020-01-21	5	$39	21:30:11.0000000
7	7	2020-05-01	6	$80	13:00:12.0000000
8	8	2020-02-26	7	$103	15:00:12.0000000
9	9	2020-03-10	8	$120	16:13:00.0000000
10	10	2020-01-06	9	$35	20:30:12.0000000
11	11	2020-01-04	10	$140	17:15:36.0000000
12	12	2020-04-29	11	$140	18:18:40.0000000
13	13	2020-10-28	12	$60	23:30:02.0000000
14	14	2020-07-29	13	$180	11:45:12.0000000
15	15	2020-05-02	14	$90	10:00:12.0000000



	FeedbackID	customerID	orderID	productRating	shippingRating	experienceRating
1	1	1	1	3.5	4.5	4.0
2	2	1	2	4.5	4.5	4.5
3	3	2	3	2.8	3.5	3.0
4	4	3	4	4.5	5.0	5.0
5	5	4	5	3.4	4.0	4.0
6	6	5	6	4.0	4.0	4.0
7	7	6	7	5.0	4.5	5.0
8	8	7	8	5.0	5.0	5.0
9	9	8	9	3.5	3.5	3.5
10	10	9	10	4.5	3.0	4.5
11	11	10	11	5.0	5.0	5.0
12	12	11	12	4.5	3.6	4.0
13	13	12	13	4.8	3.5	5.0
14	14	13	14	3.0	3.0	4.0
15	15	14	15	4.0	4.0	4.0

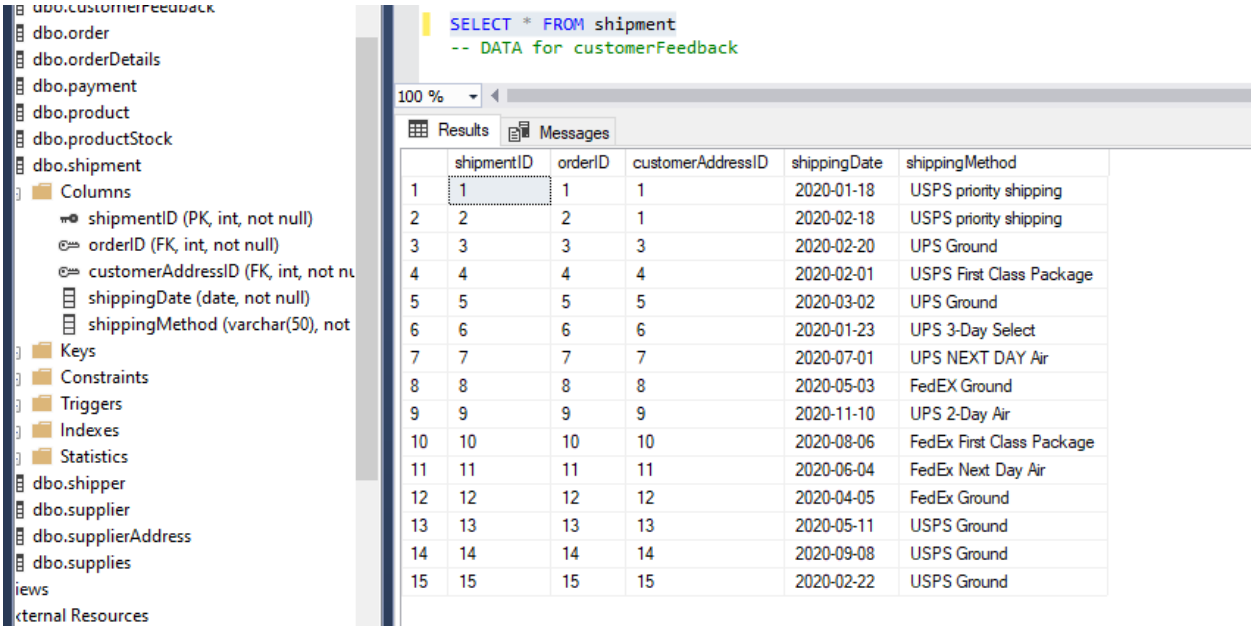
Inserting into shipment table:

```
-- DATA for shipment


INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (1, 1, 1,'01/18/2020','USPS priority shipping');
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (2, 2, 1,'02/18/2020','USPS priority shipping');
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (3, 3, 3,'02/20/2020','UPS Ground');
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (4, 4, 4,'02/01/2020','USPS First Class Package');
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (5, 5, 5,'03/02/2020','UPS Ground');
```

```
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (6, 6, 6,'01/23/2020','UPS 3-Day Select');
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (7, 7, 7,'07/01/2020','UPS NEXT DAY Air');
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (8, 8, 8,'05/03/2020','FedEX Ground');
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (9, 9, 9,'11/10/2020','UPS 2-Day Air');
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (10, 10, 10,'08/06/2020','FedEx First Class Package');
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (11, 11, 11,'06/04/2020','FedEx Next Day Air');
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (12, 12, 12,'04/05/2020','FedEx Ground');
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (13, 13, 13,'05/11/2020','USPS Ground');
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (14, 14, 14,'09/08/2020','USPS Ground');
INSERT INTO shipment (shipmentID, orderID, customerAddressID, shippingDAte,
shippingMethod)
VALUES (15, 15, 15,'02/22/2020','USPS Ground');
```

<u>Displaying shipment table:</u> The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



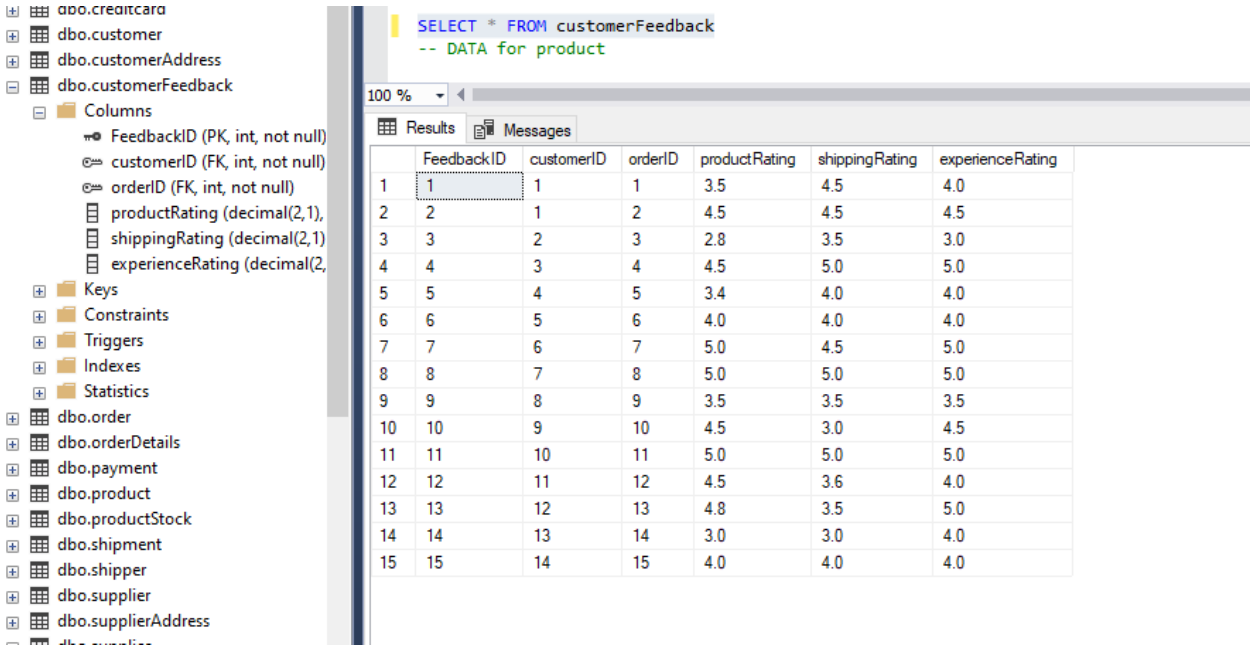
	shipmentID	orderID	customerAddressID	shippingDate	shippingMethod
1	1	1	1	2020-01-18	USPS priority shipping
2	2	2	1	2020-02-18	USPS priority shipping
3	3	3	3	2020-02-20	UPS Ground
4	4	4	4	2020-02-01	USPS First Class Package
5	5	5	5	2020-03-02	UPS Ground
6	6	6	6	2020-01-23	UPS 3-Day Select
7	7	7	7	2020-07-01	UPS NEXT DAY Air
8	8	8	8	2020-05-03	FedEX Ground
9	9	9	9	2020-11-10	UPS 2-Day Air
10	10	10	10	2020-08-06	FedEx First Class Package
11	11	11	11	2020-06-04	FedEx Next Day Air
12	12	12	12	2020-04-05	FedEx Ground
13	13	13	13	2020-05-11	USPS Ground
14	14	14	14	2020-09-08	USPS Ground
15	15	15	15	2020-02-22	USPS Ground

Inserting into customerFeedback table:

```
-- DATA for customerFeedback


INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (1, 1, 1, 3.5, 4.5, 4.0);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (2, 2, 1, 4.5, 4.5, 4.5);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (3, 3, 2, 2.8, 3.5, 3.0);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (4, 4, 3, 4.5, 5.0, 5.0);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (5, 5, 4, 3.4, 4.0, 4.0);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (6, 6, 5, 4.0, 4.0, 4.0);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (7, 7, 6, 5.0, 4.5, 5.0);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (8, 8, 7, 5.0, 5.0, 5.0);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (9, 9, 8, 3.5, 3.5, 3.5);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (10, 10, 9, 4.5, 3.0, 4.5);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (11, 11, 10, 5.0, 5.0, 5.0);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (12, 12, 11, 4.5, 3.6, 4.0);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (13, 13, 12, 4.8, 3.5, 5.0);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (14, 14, 13, 3.0, 3.0, 4.0);
INSERT INTO customerFeedback (feedbackID, orderID, customerID, productRating,
shippingRating, experienceRating) VALUES (15, 15, 14, 4.0, 4.0, 4.0);
```

Displaying customerFeedback table: The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



	FeedbackID	customerID	orderID	productRating	shippingRating	experienceRating
1	1	1	1	3.5	4.5	4.0
2	2	1	2	4.5	4.5	4.5
3	3	2	3	2.8	3.5	3.0
4	4	3	4	4.5	5.0	5.0
5	5	4	5	3.4	4.0	4.0
6	6	5	6	4.0	4.0	4.0
7	7	6	7	5.0	4.5	5.0
8	8	7	8	5.0	5.0	5.0
9	9	8	9	3.5	3.5	3.5
10	10	9	10	4.5	3.0	4.5
11	11	10	11	5.0	5.0	5.0
12	12	11	12	4.5	3.6	4.0
13	13	12	13	4.8	3.5	5.0
14	14	13	14	3.0	3.0	4.0
15	15	14	15	4.0	4.0	4.0

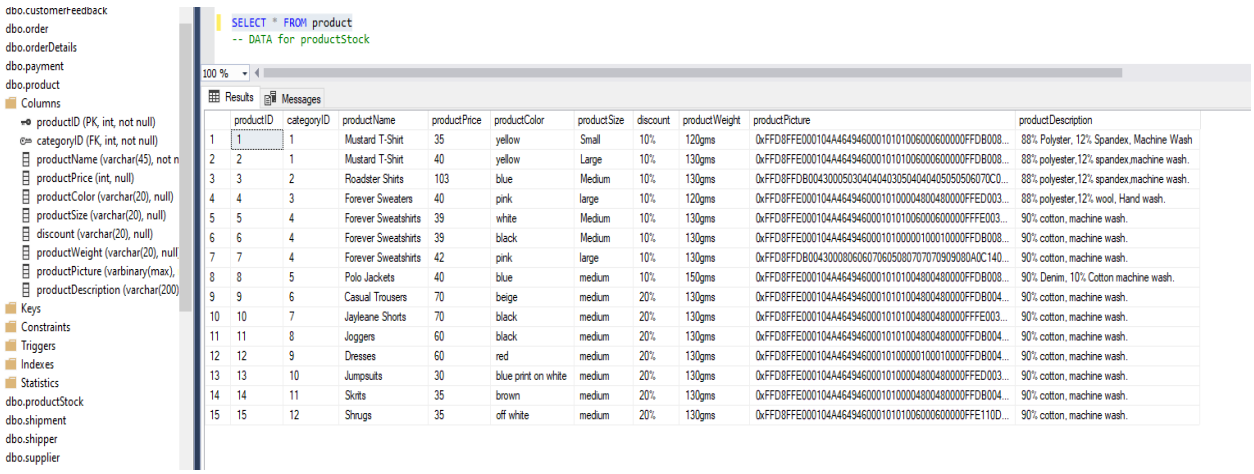
Inserting into Product table:

```
-- DATA for product


INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (1,'Mustard T-Shirt', 1, 35,'yellow','Small','10%','120gms', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\MustardT.jpg', SINGLE_BLOB)image),'88% Polyster, 12%
Spandex, Machine Wash');
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (2,'Mustard T-Shirt', 1, 40,'yellow','Large','10%','130gms', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\MustardT.jpg', SINGLE_BLOB)image),'88% polyester,12%
spandex,machine wash.');
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (3,'Roadster Shirts', 2, 103,'blue','Medium','10%','130gms', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\RoadsterShirt.jpg', SINGLE_BLOB)image),'88%
polyester,12% spandex,machine wash.');
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (4,'Forever Sweaters', 3, 40,'pink','large','10%','120gms', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\ForeverSweater.jpg', SINGLE_BLOB)image),'88%
polyester,12% wool, Hand wash.');
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (5,'Forever Sweatshirts', 4, 39,'white','Medium','10%','130gms', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\ForeverSweatshirtWhite.jpg', SINGLE_BLOB)image),'90%
cotton, machine wash.');
```

```sql
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (6,'Forever Sweatshirts', 4, 39,'black','Medium','10%','130gms', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\ForeverSweatshirtBlack.jpg', SINGLE_BLOB)image),'90%
cotton, machine wash.');
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (7,'Forever Sweatshirts', 4, 42,'pink','large','10%','130gms', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\ForeverSweatshirtPink.jpg', SINGLE_BLOB)image),'90%
cotton, machine wash.');
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (8,'Polo Jackets', 5, 40,'blue','medium','10%','150gms', (SELECT * FROM OPENROWSET
(BULK N'D:Saved Pictures\PoloJacketBlue.jpg', SINGLE_BLOB)image),'90% Denim, 10% Cotton
machine wash.');
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (9,'Casual Trousers', 6, 70,'beige','medium','20%','130gms', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\CasualTrousersBeige.jpg', SINGLE_BLOB)image),'90%
cotton, machine wash.');
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (10,'Jayleane Shorts', 7, 70,'black','medium','20%','130gms', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\BlackJShorts.jpg', SINGLE_BLOB)image),'90% cotton,
machine wash.');
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (11,'Joggers', 8, 60,'black','medium','20%','130gms', (SELECT * FROM OPENROWSET
(BULK N'D:Saved Pictures\BlackJoggers.jpg', SINGLE_BLOB)image),'90% cotton, machine
wash.');
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (12,'Dresses', 9, 60,'red','medium','20%','130gms', (SELECT * FROM OPENROWSET
(BULK N'D:Saved Pictures\RedDress.jpg', SINGLE_BLOB)image),'90% cotton, machine wash.');
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (13,'Jumpsuits', 10, 30,'blue print on white','medium','20%','130gms', (SELECT *
FROM OPENROWSET (BULK N'D:Saved Pictures\BlueonWhiteJumpsuit.jpg',
SINGLE_BLOB)image),'90% cotton, machine wash.');
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (14,'Skrits', 11, 35,'brown','medium','20%','130gms', (SELECT * FROM OPENROWSET
(BULK N'D:Saved Pictures\BrownSkirt.jpg', SINGLE_BLOB)image),'90% cotton, machine
wash.');
INSERT INTO product (productID, productName, categoryID, productPrice, productColor,
productSize, discount, productWeight, productPicture, productDescription)
VALUES (15,'Shrugs', 12, 35,'off white','medium','20%','130gms', (SELECT * FROM
OPENROWSET (BULK N'D:Saved Pictures\OffWhiteShrugs.jpg', SINGLE_BLOB)image),'90% cotton,
machine wash.');
```

Displaying product table: The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



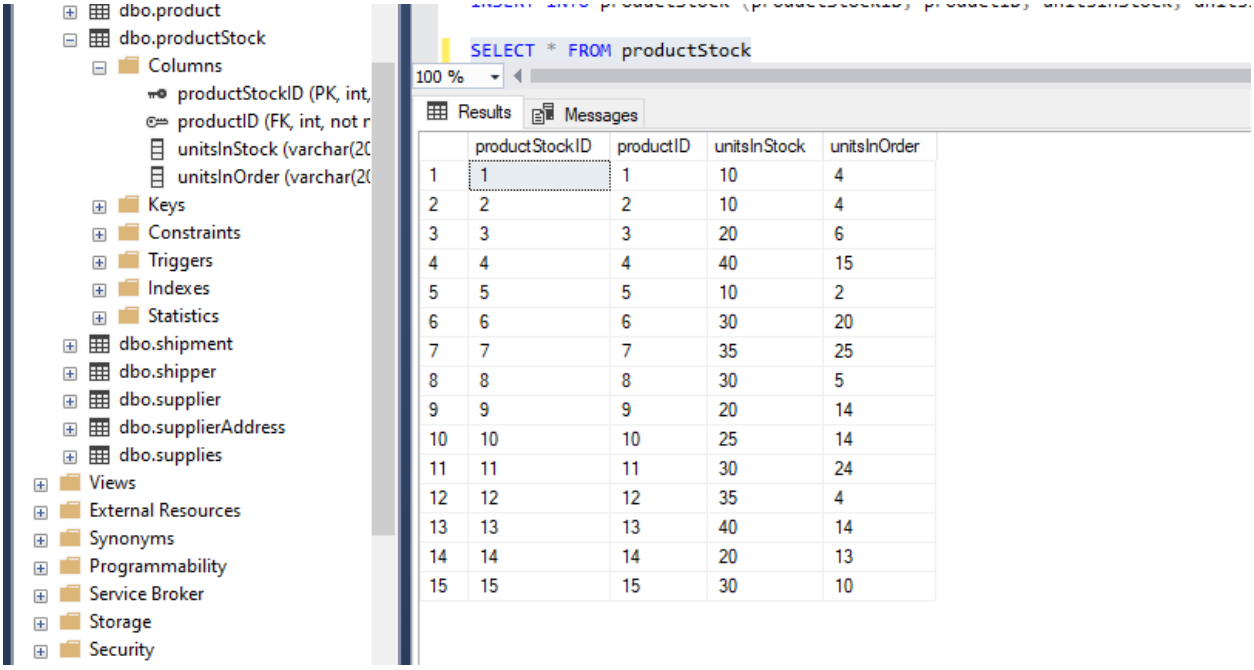
	productID	categoryID	productName	productPrice	productColor	productSize	discount	productWeight	productPicture	productDescription
1	1	1	Mustard T-Shirt	35	yellow	Small	10%	120gms	0xFFD8FFE000104A46494600010101006000600000FFDB008...	88% Polyster, 12% Spandex, Machine Wash
2	2	1	Mustard T-Shirt	40	yellow	Large	10%	130gms	0xFFD8FFE000104A46494600010101006000600000FFDB008...	88% polyester,12% spandex,machine wash.
3	3	2	Roadster Shirts	103	blue	Medium	10%	130gms	0xFFD8FFDB00430005030404040403050404040505050607 0C0...	88% polyester,12% spandex,machine wash.
4	4	3	Forever Sweaters	40	pink	large	10%	120gms	0xFFD8FFE000104A46494600010100004800480000FFED003...	88% polyester,12% wool, Hand wash.
5	5	4	Forever Sweatshirts	39	white	Medium	10%	130gms	0xFFD8FFE000104A46494600010101006000600000FFFE003...	90% cotton, machine wash.
6	6	4	Forever Sweatshirts	39	black	Medium	10%	130gms	0xFFD8FFE000104A46494600010100000100010000FFDB008...	90% cotton, machine wash.
7	7	4	Forever Sweatshirts	42	pink	large	10%	130gms	0xFFD8FFDB00430008060607060508070707090908 0AC140...	90% cotton, machine wash.
8	8	5	Polo Jackets	40	blue	medium	10%	150gms	0xFFD8FFE000104A46494600010101004800480000FFDB008...	90% Denim, 10% Cotton machine wash.
9	9	6	Casual Trousers	70	beige	medium	20%	130gms	0xFFD8FFE000104A46494600010101004800480000FFDB004...	90% cotton, machine wash.
10	10	7	Jayleane Shorts	70	black	medium	20%	130gms	0xFFD8FFE000104A46494600010101004800480000FFFE003...	90% cotton, machine wash.
11	11	8	Joggers	60	black	medium	20%	130gms	0xFFD8FFE000104A46494600010101004800480000FFDB004...	90% cotton, machine wash.
12	12	9	Dresses	60	red	medium	20%	130gms	0xFFD8FFE000104A46494600010100000100010000FFDB004...	90% cotton, machine wash.
13	13	10	Jumpsuits	30	blue print on white	medium	20%	130gms	0xFFD8FFE000104A46494600010100004800480000FFED003...	90% cotton, machine wash.
14	14	11	Skrits	35	brown	medium	20%	130gms	0xFFD8FFE000104A46494600010100004800480000FFDB004...	90% cotton, machine wash.
15	15	12	Shrugs	35	off white	medium	20%	130gms	0xFFD8FFE000104A46494600010101006000600000FFE110D...	90% cotton, machine wash.

Inserting into productStock table:

```
-- DATA for productStock


INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(1, 1,'10','4');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(2, 2,'10','4');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(3, 3,'20','6');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(4, 4,'40','15');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(5, 5,'10','2');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(6, 6,'30','20');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(7, 7,'35','25');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(8, 8,'30','5');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(9, 9,'20','14');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(10, 10,'25','14');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(11, 11,'30','24');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(12, 12,'35','4');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(13, 13,'40','14');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(14, 14,'20','13');
INSERT INTO productStock (productStockID, productID, unitsInStock, unitsInOrder) VALUES
(15, 15,'30','10');
```

Displaying productStock table: The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



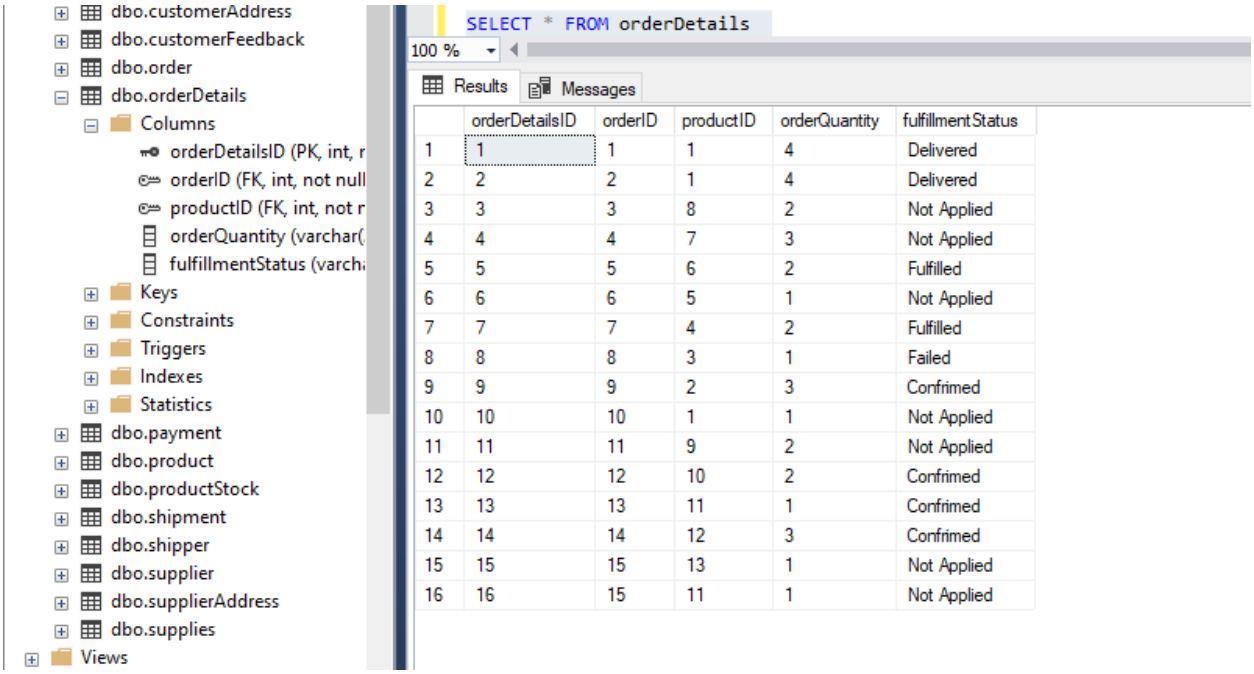
Inserting into orderDetails table:

```
-- DATA for orderDetails


INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (1,1,1,'4','Delivered');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (2,2,1,'4','Delivered');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (3,3,8,'2','Not Applied');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (4,4,7,'3','Not Applied');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (5,5,6,'2','Fulfilled');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (6,6,5,'1','Not Applied');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (7,7,4,'2','Fulfilled');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (8,8,3,'1','Failed');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (9,9,2,'3','Confrimed');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (10,10,1,'1','Not Applied');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (11,11,9,'2','Not Applied');
```

```sql
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (12,12,10,'2','Confrimed');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (13,13,11,'1','Confrimed');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (14,14,12,'3','Confrimed');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (15,15,13,'1','Not Applied');
INSERT INTO orderDetails (orderDetailsID, orderID, productID, orderQuantity,
fulfillmentStatus) VALUES (16,15,11,'1','Not Applied');
```

Displaying orderDetails table: The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



	orderDetailsID	orderID	productID	orderQuantity	fulfillmentStatus
1	1	1	1	4	Delivered
2	2	2	1	4	Delivered
3	3	3	8	2	Not Applied
4	4	4	7	3	Not Applied
5	5	5	6	2	Fulfilled
6	6	6	5	1	Not Applied
7	7	7	4	2	Fulfilled
8	8	8	3	1	Failed
9	9	9	2	3	Confrimed
10	10	10	1	1	Not Applied
11	11	11	9	2	Not Applied
12	12	12	10	2	Confrimed
13	13	13	11	1	Confrimed
14	14	14	12	3	Confrimed
15	15	15	13	1	Not Applied
16	16	15	11	1	Not Applied

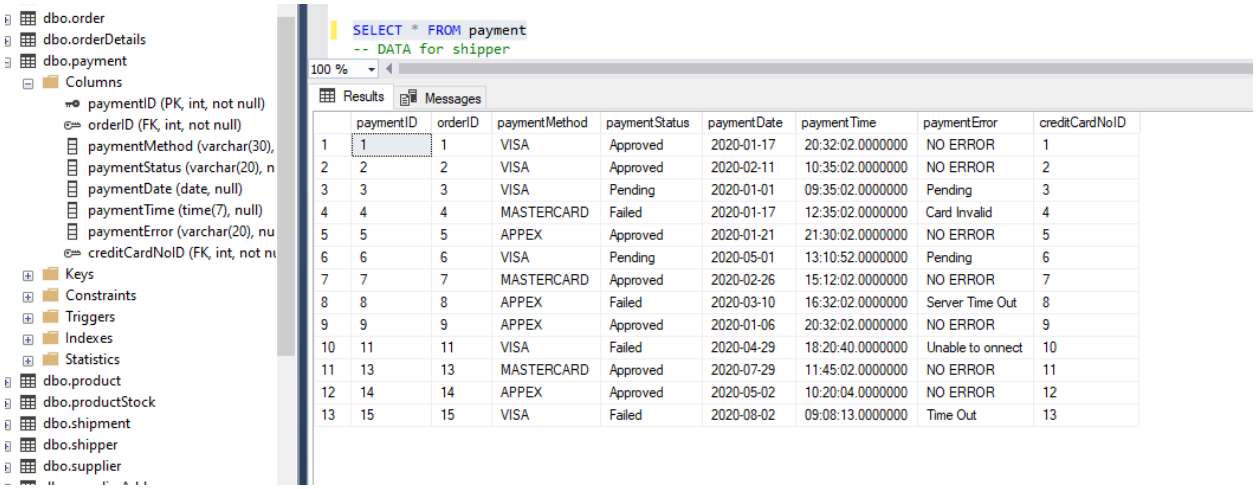
Inserting into payment table:

```sql
-- DATA for payment


INSERT INTO payment (paymentID, orderID, paymentMethod, paymentStatus, paymentDate,
paymentTime, paymentError, creditCardNoID) VALUES (1,
1,'VISA','Approved','01/17/2020','20:32:02','NO ERROR', 1);
INSERT INTO payment (paymentID, orderID, paymentMethod, paymentStatus, paymentDate,
paymentTime, paymentError, creditCardNoID) VALUES (2,
2,'VISA','Approved','02/11/2020','10:35:02','NO ERROR', 2);
INSERT INTO payment (paymentID, orderID, paymentMethod, paymentStatus, paymentDate,
paymentTime, paymentError, creditCardNoID) VALUES (3,
3,'VISA','Pending','01/01/2020','09:35:02','Pending', 3);
INSERT INTO payment (paymentID, orderID, paymentMethod, paymentStatus, paymentDate,
paymentTime, paymentError, creditCardNoID) VALUES (4,
4,'MASTERCARD','Failed','01/17/2020','12:35:02','Card Invalid', 4);
```

```sql
INSERT INTO payment (paymentID, orderID, paymentMethod, paymentStatus, paymentDate,
paymentTime, paymentError, creditCardNoID) VALUES (5,
5,'APPEX','Approved','01/21/2020','21:30:02','NO ERROR', 5);
INSERT INTO payment (paymentID, orderID, paymentMethod, paymentStatus, paymentDate,
paymentTime, paymentError, creditCardNoID) VALUES (6,
6,'VISA','Pending','05/01/2020','13:10:52','Pending', 6);
INSERT INTO payment (paymentID, orderID, paymentMethod, paymentStatus, paymentDate,
paymentTime, paymentError, creditCardNoID) VALUES
(7,7,'MASTERCARD','Approved','02/26/2020','15:12:02','NO ERROR', 7);
INSERT INTO payment (paymentID, orderID, paymentMethod, paymentStatus, paymentDate,
paymentTime, paymentError, creditCardNoID) VALUES
(8,8,'APPEX','Failed','03/10/2020','16:32:02','Server Time Out', 8);
INSERT INTO payment (paymentID, orderID, paymentMethod, paymentStatus, paymentDate,
paymentTime, paymentError, creditCardNoID) VALUES
(9,9,'APPEX','Approved','01/06/2020','20:32:02','NO ERROR', 9);
INSERT INTO payment (paymentID, orderID, paymentMethod, paymentStatus, paymentDate,
paymentTime, paymentError, creditCardNoID) VALUES
(11,11,'VISA','Failed','04/29/2020','18:20:40','Unable to onnect', 10);
INSERT INTO payment (paymentID, orderID, paymentMethod, paymentStatus, paymentDate,
paymentTime, paymentError, creditCardNoID) VALUES
(13,13,'MASTERCARD','Approved','07/29/2020','11:45:02','NO ERROR', 11);
INSERT INTO payment (paymentID, orderID, paymentMethod, paymentStatus, paymentDate,
paymentTime, paymentError, creditCardNoID) VALUES
(14,14,'APPEX','Approved','05/02/2020','10:20:04','NO ERROR', 12);
INSERT INTO payment (paymentID, orderID, paymentMethod, paymentStatus, paymentDate,
paymentTime, paymentError, creditCardNoID) VALUES
(15,15,'VISA','Failed','08/02/2020','09:08:13','Time Out', 13);
```

Displaying payment table: The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



	paymentID	orderID	paymentMethod	paymentStatus	paymentDate	paymentTime	paymentError	creditCardNoID
1	1	1	VISA	Approved	2020-01-17	20:32:02.0000000	NO ERROR	1
2	2	2	VISA	Approved	2020-02-11	10:35:02.0000000	NO ERROR	2
3	3	3	VISA	Pending	2020-01-01	09:35:02.0000000	Pending	3
4	4	4	MASTERCARD	Failed	2020-01-17	12:35:02.0000000	Card Invalid	4
5	5	5	APPEX	Approved	2020-01-21	21:30:02.0000000	NO ERROR	5
6	6	6	VISA	Pending	2020-05-01	13:10:52.0000000	Pending	6
7	7	7	MASTERCARD	Approved	2020-02-26	15:12:02.0000000	NO ERROR	7
8	8	8	APPEX	Failed	2020-03-10	16:32:02.0000000	Server Time Out	8
9	9	9	APPEX	Approved	2020-01-06	20:32:02.0000000	NO ERROR	9
10	11	11	VISA	Failed	2020-04-29	18:20:40.0000000	Unable to onnect	10
11	13	13	MASTERCARD	Approved	2020-07-29	11:45:02.0000000	NO ERROR	11
12	14	14	APPEX	Approved	2020-05-02	10:20:04.0000000	NO ERROR	12
13	15	15	VISA	Failed	2020-08-02	09:08:13.0000000	Time Out	13

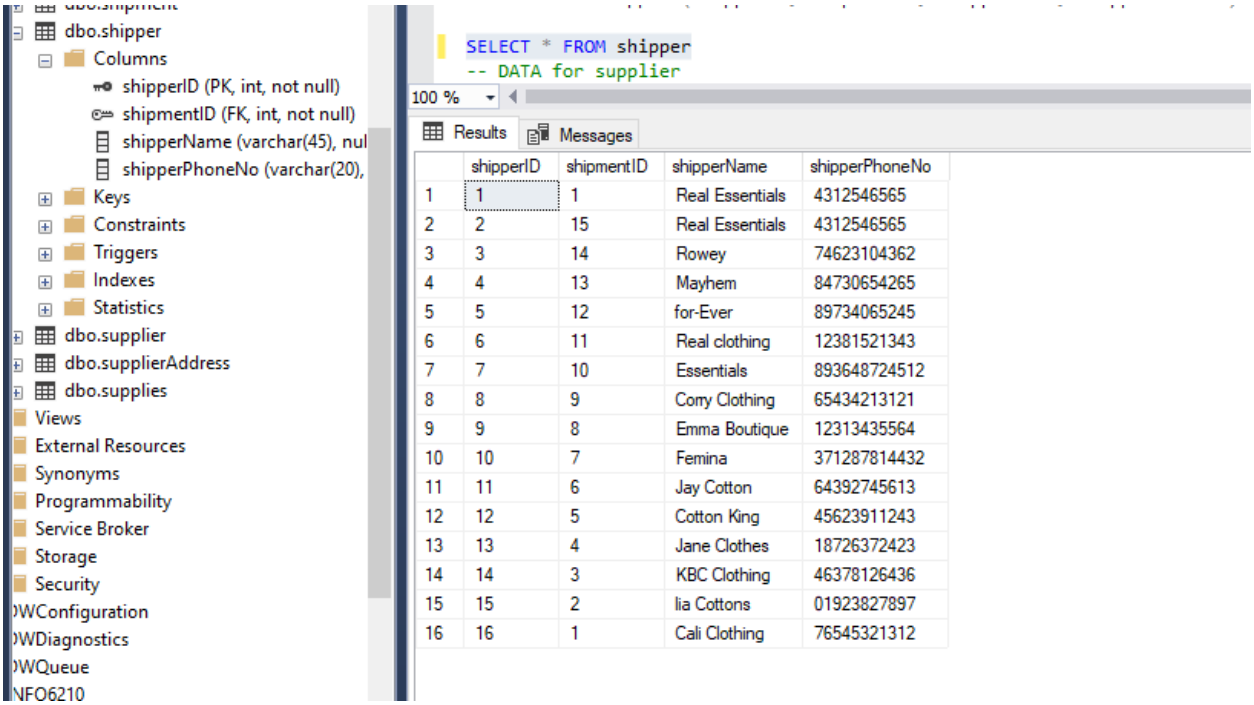
Inserting into shipper table:

```sql
-- DATA for shipper


INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (1,
1,'Real Essentials','4312546565');
```

```
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (2,
15,'Real Essentials','4312546565');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (3,
14,'Rowey','74623104362');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (4,
13,'Mayhem','84730654265');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (5,
12,'for-Ever','89734065245');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (6,
11,'Real clothing','12381521343');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (7,
10,'Essentials','893648724512');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (8,
9,'Corry Clothing','65434213121');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (9,
8,'Emma Boutique','12313435564');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (10,
7,'Femina','371287814432');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (11,
6,'Jay Cotton','64392745613');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (12,
5,'Cotton King','45623911243');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (13,
4,'Jane Clothes','18726372423');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (14,
3,'KBC Clothing','46378126436');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (15,
2,'lia Cottons','01923827897');
INSERT INTO shipper (shipperID, shipmentID, shipperName, shipperPhoneNo) VALUES (16,
1,'Cali Clothing','76545321312');
```

<u>Displaying shipper table:</u> The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



	shipperID	shipmentID	shipperName	shipperPhoneNo
1	1	1	Real Essentials	4312546565
2	2	15	Real Essentials	4312546565
3	3	14	Rowey	74623104362
4	4	13	Mayhem	84730654265
5	5	12	for-Ever	89734065245
6	6	11	Real clothing	12381521343
7	7	10	Essentials	893648724512
8	8	9	Corry Clothing	65434213121
9	9	8	Emma Boutique	12313435564
10	10	7	Femina	371287814432
11	11	6	Jay Cotton	64392745613
12	12	5	Cotton King	45623911243
13	13	4	Jane Clothes	18726372423
14	14	3	KBC Clothing	46378126436
15	15	2	lia Cottons	01923827897
16	16	1	Cali Clothing	76545321312

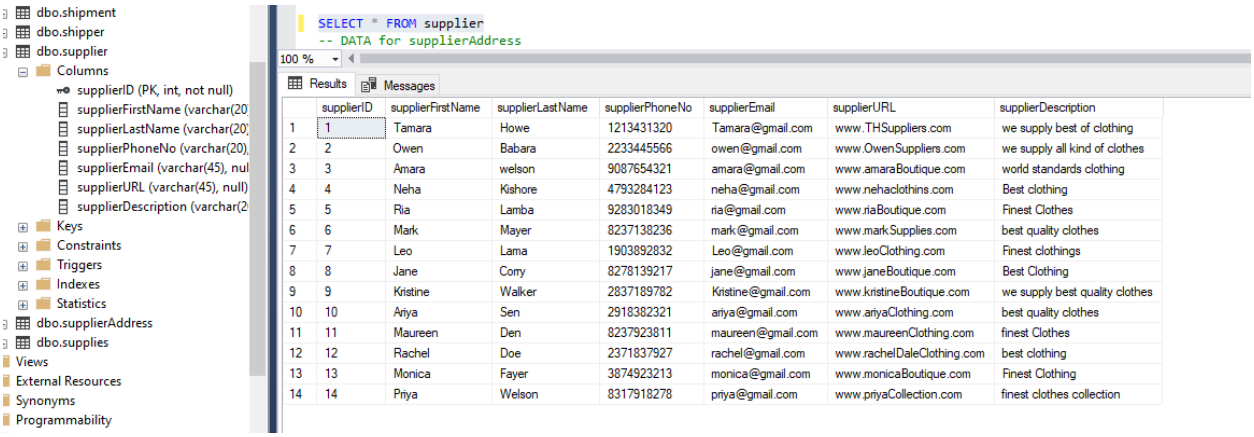
Inserting into supplier table:

```
-- DATA for supplier


INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES (1,'Tamara','Howe','1213431320','Tamara@gmail.com','www.THSuppliers.com','we
supply best of clothing');
INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES(2,'Owen','Babara','2233445566','owen@gmail.com','www.OwenSuppliers.com','we supply
all kind of clothes');
INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES(3,'Amara','welson','9087654321','amara@gmail.com','www.amaraBoutique.com','world
standards clothing');
INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES(4,'Neha','Kishore','4793284123','neha@gmail.com','www.nehaclothins.com','Best
clothing');
INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES(5,'Ria','Lamba','9283018349','ria@gmail.com','www.riaBoutique.com','Finest
Clothes');
INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES(6,'Mark','Mayer','8237138236','mark@gmail.com','www.markSupplies.com','best
quality clothes');
INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES(7,'Leo','Lama','1903892832','Leo@gmail.com','www.leoClothing.com','Finest
clothings');
INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES(8,'Jane','Corry','8278139217','jane@gmail.com','www.janeBoutique.com','Best
Clothing');
INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES(9,'Kristine','Walker','2837189782','Kristine@gmail.com','www.kristineBoutique.com'
,'we supply best quality clothes');
INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES(10,'Ariya','Sen','2918382321','ariya@gmail.com','www.ariyaClothing.com','best
quality clothes');
INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES(11,'Maureen','Den','8237923811','maureen@gmail.com','www.maureenClothing.com','fin
est Clothes');
INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES(12,'Rachel','Doe','2371837927','rachel@gmail.com','www.rachelDaleClothing.com','be
st clothing');
```

```
INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES(13,'Monica','Fayer','3874923213','monica@gmail.com','www.monicaBoutique.com','Fine
st Clothing');
INSERT INTO supplier (supplierID, supplierFirstName, supplierLastName, supplierPhoneNo,
supplierEmail, supplierURL, supplierDescription)
VALUES(14,'Priya','Welson','8317918278','priya@gmail.com','www.priyaCollection.com','fine
st clothes collection ');
```

Displaying supplier table: The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



	supplierID	supplierFirstName	supplierLastName	supplierPhoneNo	supplierEmail	supplierURL	supplierDescription
1	1	Tamara	Howe	1213431320	Tamara@gmail.com	www.THSuppliers.com	we supply best of clothing
2	2	Owen	Babara	2233445566	owen@gmail.com	www.OwenSuppliers.com	we supply all kind of clothes
3	3	Amara	welson	9087654321	amara@gmail.com	www.amaraBoutique.com	world standards clothing
4	4	Neha	Kishore	4793284123	neha@gmail.com	www.nehaclothins.com	Best clothing
5	5	Ria	Lamba	9283018349	ria@gmail.com	www.riaBoutique.com	Finest Clothes
6	6	Mark	Mayer	8237138236	mark@gmail.com	www.markSupplies.com	best quality clothes
7	7	Leo	Lama	1903892832	Leo@gmail.com	www.leoClothing.com	Finest clothings
8	8	Jane	Corry	8278139217	jane@gmail.com	www.janeBoutique.com	Best Clothing
9	9	Kristine	Walker	2837189782	Kristine@gmail.com	www.kristineBoutique.com	we supply best quality clothes
10	10	Ariya	Sen	2918382321	ariya@gmail.com	www.ariyaClothing.com	best quality clothes
11	11	Maureen	Den	8237923811	maureen@gmail.com	www.maureenClothing.com	finest Clothes
12	12	Rachel	Doe	2371837927	rachel@gmail.com	www.rachelDaleClothing.com	best clothing
13	13	Monica	Fayer	3874923213	monica@gmail.com	www.monicaBoutique.com	Finest Clothing
14	14	Priya	Welson	8317918278	priya@gmail.com	www.priyaCollection.com	finest clothes collection

---

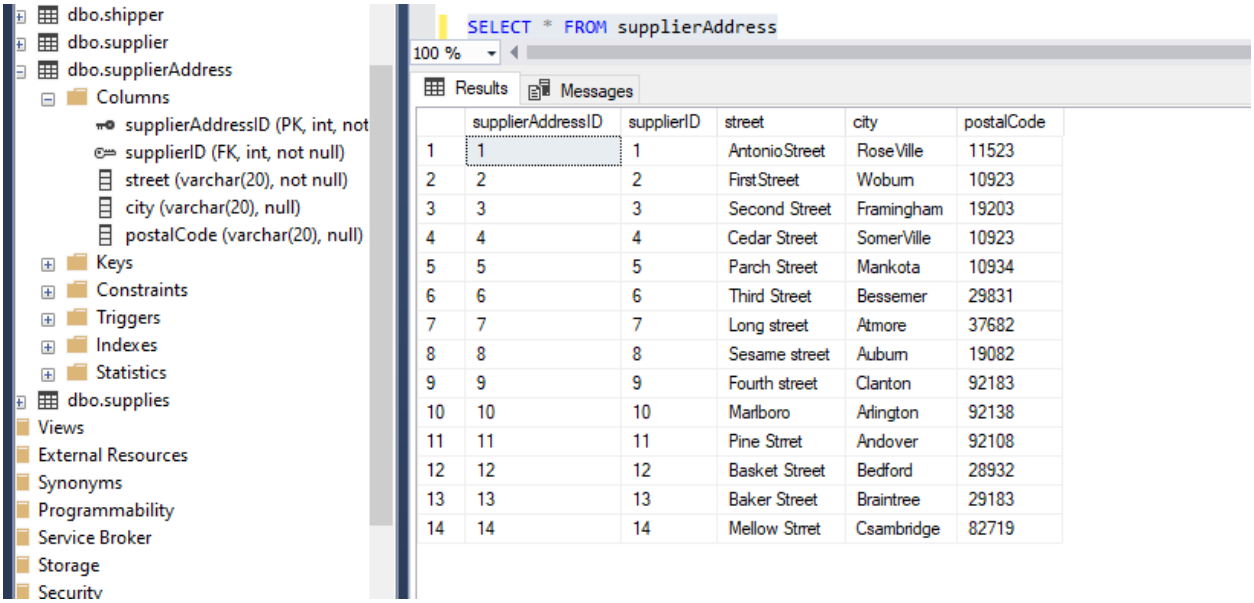
Inserting into supplierAddress table:

```
-- DATA for supplierAddress


INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (1, 1,'AntonioStreet','RoseVille','11523');
INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (2, 2,'FirstStreet','Woburn','10923');
INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (3, 3,'Second Street','Framingham','19203');
INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (4, 4,'Cedar Street','SomerVille','10923');
INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (5, 5,'Parch Street','Mankota','10934');
INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (6, 6,'Third Street','Bessemer','29831');
INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (7, 7,'Long street','Atmore','37682');
INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (8, 8,'Sesame street','Auburn','19082');
INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (9, 9,'Fourth street','Clanton','92183');
INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (10, 10,'Marlboro','Arlington','92138');
INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (11, 11,'Pine Strret','Andover','92108');
INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (12, 12,'Basket Street','Bedford','28932');
```

```sql
INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (13, 13,'Baker Street','Braintree','29183');
INSERT INTO supplierAddress (supplierAddressID, supplierID, street,city, postalCode)
VALUES (14, 14,'Mellow Strret','Csambridge','82719');
```

Displaying supplierAddress table: The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.



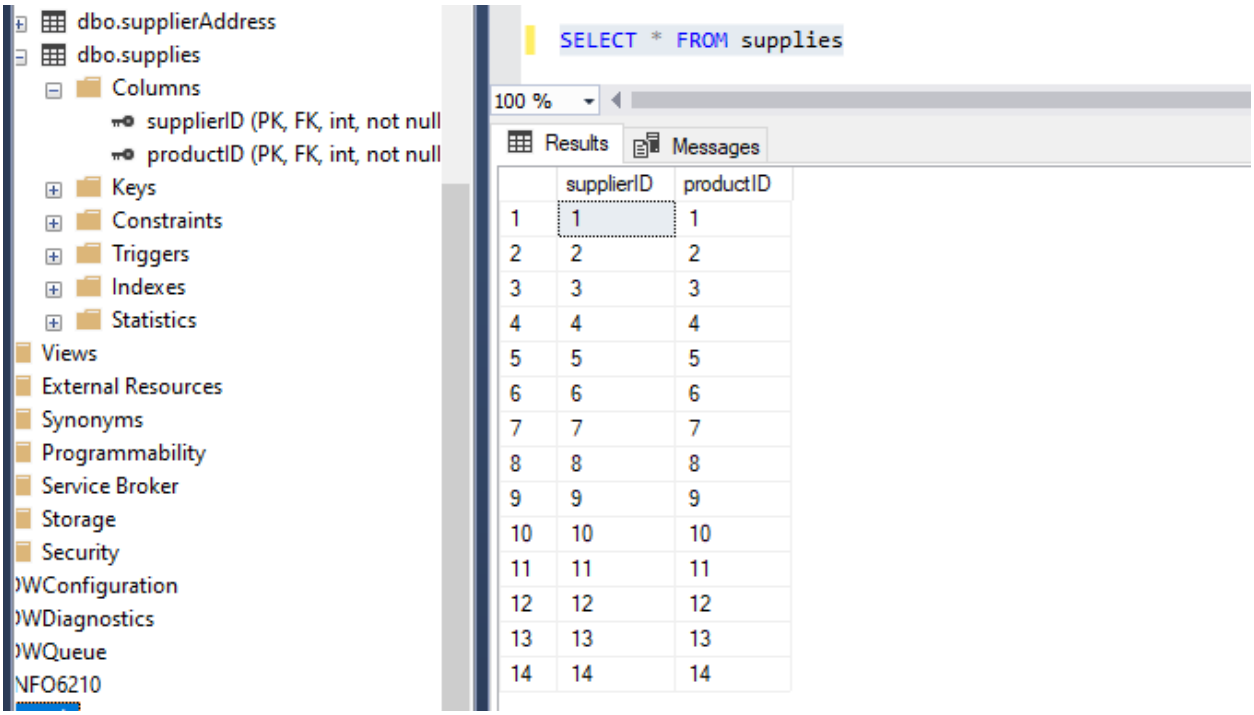
	supplierAddressID	supplierID	street	city	postalCode
1	1	1	Antonio Street	Rose Ville	11523
2	2	2	First Street	Woburn	10923
3	3	3	Second Street	Framingham	19203
4	4	4	Cedar Street	Somer Ville	10923
5	5	5	Parch Street	Mankota	10934
6	6	6	Third Street	Bessemer	29831
7	7	7	Long street	Atmore	37682
8	8	8	Sesame street	Auburn	19082
9	9	9	Fourth street	Clanton	92183
10	10	10	Marlboro	Arlington	92138
11	11	11	Pine Strret	Andover	92108
12	12	12	Basket Street	Bedford	28932
13	13	13	Baker Street	Braintree	29183
14	14	14	Mellow Strret	Csambridge	82719

Inserting into supplies table:

```sql
-- DATA for supplies


INSERT INTO supplies (supplierID, productID) VALUES (1, 1);
INSERT INTO supplies (supplierID, productID) VALUES (2, 2);
INSERT INTO supplies (supplierID, productID) VALUES (3, 3);
INSERT INTO supplies (supplierID, productID) VALUES (4, 4);
INSERT INTO supplies (supplierID, productID) VALUES (5, 5);
INSERT INTO supplies (supplierID, productID) VALUES (6, 6);
INSERT INTO supplies (supplierID, productID) VALUES (7, 7);
INSERT INTO supplies (supplierID, productID) VALUES (8, 8);
INSERT INTO supplies (supplierID, productID) VALUES (9, 9);
INSERT INTO supplies (supplierID, productID) VALUES (10, 10);
INSERT INTO supplies (supplierID, productID) VALUES (11, 11);
INSERT INTO supplies (supplierID, productID) VALUES (12, 12);
INSERT INTO supplies (supplierID, productID) VALUES (13, 13);
INSERT INTO supplies (supplierID, productID) VALUES (14, 14);
```

Displaying supplies table: The columns are also displayed in the Object Explorer where the Primary and Foreign keys are depicted.


dbo.supplierAddress
dbo.supplies
Columns
supplierID (PK, FK, int, not null
productID (PK, FK, int, not null
Keys
Constraints
Triggers
Indexes
Statistics
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
WConfiguration
WDiagnostics
WQueue
NFO6210
SELECT * FROM supplies
100 %
Results
Messages
supplierID productID
1 1 1
2 2 2
3 3 3
4 4 4
5 5 5
6 6 6
7 7 7
8 8 8
9 9 9
10 10 10
11 11 11
12 12 12
13 13 13
14 14 14

DATA ENCRYPTION:

```
USE DMDDP4

GO

-- Create database Key
CREATE MASTER KEY
ENCRYPTION BY PASSWORD = 'DMDDP4Encrypt';

--verify that master key has been created
SELECT name KeyName,
symmetric_key_id KeyID,
key_length KeyLength,
algorithm_desc KeyAlgorithm
FROM sys.symmetric_keys;

-------------------------------------------------------END OF STEP 1---------------------
-----
-- Create self signed certificate
USE DMDDP4;
GO
CREATE CERTIFICATE CreditCardNumber
WITH SUBJECT = 'EncryptCreditCardData';
GO
-----------------------------------------------END OF STEP 2---------------------------
----
-- Create symmetric Key
CREATE SYMMETRIC KEY CustCC_SM
 WITH ALGORITHM = AES_256
 ENCRYPTION BY CERTIFICATE CreditCardNumber;

----------------------------------------------END OF STEP 3----------------------------
 --ADD new column for encrypted data
 ALTER TABLE creditcard
 ADD encryptedCreditCardNo varbinary(MAX)

 ----------------------------------------------END OF STEP 4----------------------------
---
-- Opens the symmetric key for use
OPEN SYMMETRIC KEY CustCC_SM
DECRYPTION BY CERTIFICATE CreditCardNumber;


-- Populating encrypted credit card no into new column
UPDATE dbo.creditcard
SET encryptedCreditCardNo = EncryptByKey (Key_GUID('CustCC_SM'), creditCardNo)
FROM dbo.creditcard;
GO

----------------------------------------------END OF STEP 5----------------------------
---------
-- Closing the symmetric key
```

```sql
CLOSE SYMMETRIC KEY CustCC_SM;
GO

---DROPPING CreditCardNo----
ALTER TABLE creditCard
DROP COLUMN creditCardNo;
GO

----CHECK THE NEW ENCRYPTED DATA-------
SELECT * FROM creditCard
```

```sql
----CHECK THE NEW ENCRYPTED DATA-------
SELECT * FROM creditCard
```

100 %

Results | Messages

	creditCardNoID	customerID	SetAsPrimary	creditCardType	cardExpiry	encryptedCreditCardNo
1	1	1	yes	VISA	11/21	0x009DA9A989D1844C90A5CC5A032D4A730200000063BF1A5...
2	2	2	yes	VISA	01/23	0x009DA9A989D1844C90A5CC5A032D4A7302000000C5423B2...
3	3	3	yes	VISA	01/23	0x009DA9A989D1844C90A5CC5A032D4A7302000000BA3A8C...
4	4	4	no	MASTERCARD	03/25	0x009DA9A989D1844C90A5CC5A032D4A73020000008EEC38...
5	5	5	yes	APPEX	09/25	0x009DA9A989D1844C90A5CC5A032D4A7302000000F618713...
6	6	1	no	MASTERCARD	11/25	0x009DA9A989D1844C90A5CC5A032D4A7302000000EEF1150...
7	7	6	yes	VISA	09/25	0x009DA9A989D1844C90A5CC5A032D4A730200000083A0E2...
8	8	7	no	MASTERCARD	08/23	0x009DA9A989D1844C90A5CC5A032D4A7302000000B2CB6B...
9	9	8	yes	APPEX	02/24	0x009DA9A989D1844C90A5CC5A032D4A73020000000F59E3E...
10	10	9	no	APPEX	04/27	0x009DA9A989D1844C90A5CC5A032D4A7302000000A26A1C8...
11	11	10	no	MASTERCARD	04/27	0x009DA9A989D1844C90A5CC5A032D4A7302000000CA27325...
12	12	11	yes	VISA	05/25	0x009DA9A989D1844C90A5CC5A032D4A73020000006310AB4...
13	13	12	no	VISA	06/23	0x009DA9A989D1844C90A5CC5A032D4A73020000009890370...
14	14	13	no	MASTERCARD	07/22	0x009DA9A989D1844C90A5CC5A032D4A73020000001609FA2...
15	15	14	no	APPEX	02/25	0x009DA9A989D1844C90A5CC5A032D4A73020000008773231...

```sql
---------------------------------------------END OF STEP 6-----------------------------
--
---Decryting Credit Card No-
OPEN SYMMETRIC KEY CustCC_SM
DECRYPTION BY CERTIFICATE CreditCardNumber;
SELECT creditCardNoID, customerID, SetAsPrimary, creditCardType, cardExpiry,
encryptedCreditCardNo AS 'Encrypted CC', CONVERT(varchar(50),
DecryptByKey(encryptedCreditCardNo)) AS 'Decrypted CC'
FROM creditcard
```

```
---Decryting Credit Card No-
OPEN SYMMETRIC KEY CustCC_SM
DECRYPTION BY CERTIFICATE CreditCardNumber;
SELECT creditCardNoID, customerID, SetAsPrimary, creditCardType, cardExpiry, encryptedCreditCardNo AS 'Encrypted CC', CONVERT(varcha
FROM creditcard
```

Results | Messages

creditCardNoID	customerID	SetAsPrimary	creditCardType	cardExpiry	Encrypted CC	Decrypted CC
1	1	yes	VISA	11/21	0x009DA9A989D1844C90A5CC5A032D4A730200000063BF1A5...	1234567891234567
2	2	yes	VISA	01/23	0x009DA9A989D1844C90A5CC5A032D4A7302000000C5423B2...	2222405343248877
3	3	yes	VISA	01/23	0x009DA9A989D1844C90A5CC5A032D4A7302000000BA3A8C...	2222990905257051
4	4	no	MASTERCARD	03/25	0x009DA9A989D1844C90A5CC5A032D4A73020000008EEC38...	2223007648726984
5	5	yes	APPEX	09/25	0x009DA9A989D1844C90A5CC5A032D4A7302000000F618713...	2223577120017656
6	1	no	MASTERCARD	11/25	0x009DA9A989D1844C90A5CC5A032D4A7302000000EEF1150...	378282246310005
7	6	yes	VISA	09/25	0x009DA9A989D1844C90A5CC5A032D4A730200000083A0E2...	5105105105105100
8	7	no	MASTERCARD	08/23	0x009DA9A989D1844C90A5CC5A032D4A7302000000B2CB6B...	5111010030175156
9	8	yes	APPEX	02/24	0x009DA9A989D1844C90A5CC5A032D4A73020000000F59E3E...	5185540810000019
10	9	no	APPEX	04/27	0x009DA9A989D1844C90A5CC5A032D4A7302000000A26A1C8...	5200828282828210
11	10	no	MASTERCARD	04/27	0x009DA9A989D1844C90A5CC5A032D4A7302000000CA27325...	5204230080000017
12	11	yes	VISA	05/25	0x009DA9A989D1844C90A5CC5A032D4A73020000006310AB4...	5204740009900014
13	12	no	VISA	06/23	0x009DA9A989D1844C90A5CC5A032D4A73020000009890370...	5420923878724339
14	13	no	MASTERCARD	07/22	0x009DA9A989D1844C90A5CC5A032D4A73020000001609FA2...	5455330760000018
15	14	no	APPEX	02/25	0x009DA9A989D1844C90A5CC5A032D4A73020000008773231...	5506900490000436

FUNCTIONS, STORED PROCEDURES, TRIGGERS AND VIEWS:

USER-DEFINED FUNCTION 1:

Explaination: GETS CustomerID as input parameter and returns CUSTOMER FULL NAME

```
------ GETS CustomerID as input parameter and returns CUSTOMER FULL NAME----------------
------
CREATE FUNCTION CustomerFullName (@customerID int) RETURNS varchar(50)
AS
BEGIN
DECLARE @FullName varchar(50)
SELECT @FullName = c.customerFirstName + ' ' + c.customerLastName
FROM customer c
WHERE customerID = @customerID
RETURN @FullName
END
```

Result:

```
SELECT dbo.CustomerFullName (c.customerID) AS CustomerFullName
FROM customer c
-----------------------------------END OF UDF 1--------------------
------ GETS SupplierID as input parameter and returns SUPPLIER FULL NAME
CREATE FUNCTION SupplierFullName (@supplierID int) RETURNS varchar(50)
AS
BEGIN
DECLARE @FullName varchar(50)
SELECT @FullName = s.supplierFirstName + ' ' + s.supplierLastName
FROM supplier s
```

100 %

Results | Messages

	CustomerFullName
1	Cecelia Chapman
2	Iris Watson
3	Celeste Slater
4	Theodore Lowe
5	Kyla Olsen
6	Hiroko Potter
7	Nyssa Vazquez
8	Lawrence Moreno
9	Ian Somerhalder
10	Aaron Hawkins
11	Hedy Greene
12	Melvin Porter
13	Keefe Sellers
14	Joan Romero

USER-DEFINED FUNCTION 2:

Explanation: GETS SupplierID as input parameter and returns SUPPLIER FULL NAME

```
------ GETS SupplierID as input parameter and returns SUPPLIER FULL NAME----------------
------
CREATE FUNCTION SupplierFullName (@supplierID int) RETURNS varchar(50)
AS
BEGIN
DECLARE @FullName varchar(50)
SELECT @FullName = s.supplierFirstName + ' ' + s.supplierLastName
FROM supplier s
WHERE supplierID = @supplierID
RETURN @FullName
END
```

Result:

```
SELECT dbo.SupplierFullName (s.supplierID) AS supplierFullName
FROM supplier s
-----------------------------------END OF UDF 2--------------------
```

100 %

Results | Messages

	supplierFullName
1	Tamara Howe
2	Owen Babara
3	Amara welson
4	Neha Kishore
5	Ria Lamba
6	Mark Mayer
7	Leo Lama
8	Jane Corry
9	Kristine Walker
10	Ariya Sen
11	Maureen Den
12	Rachel Doe
13	Monica Fayer
14	Priya Welson

USER-DEFINED FUNCTION 3:

Explanation: Take OrderID as input and returns Order Total

```
CREATE FUNCTION
GetOrderTotal (@orderID int)
RETURNS Float
AS
BEGIN
DECLARE @OrderTotal float
SELECT @OrderTotal = SUM ((o.orderQuantity) * (p.productPrice))
FROM orderDetails o JOIN product p
ON orderID = @orderID AND o.productID = p.productID
RETURN @OrderTotal
END
```

Result:

```
--------------------QUERY TO CALL UDF-------------------
SELECT dbo.GetOrderTotal (o.orderID) AS OrderTotal
FROM [order] o
---------------------------------------------------------
```

100 %

Results | Messages

	OrderTotal
1	140
2	140
3	80
4	126
5	78
6	39
7	80
8	103
9	120
10	35
11	140
12	140
13	60
14	180
15	90

STORED PROCEDURE NO:1

Explanation: Gets CustomerID as parameter and displays CUSTOMER INFORMATION and FEEDBACK ON ORDER ID based on the @customer_ID as the input parameter.

```
CREATE PROCEDURE
GetCustomerFeedbackWith @customer_ID INT
AS
BEGIN
SELECT customer.customerID, dbo.CustomerFullName (customer.customerID) AS
CustomerFullName, customerPhoneNo, customerEmail, orderID, productRating, shippingRating,
experienceRating
FROM customer JOIN customerFeedback
ON [customerFeedback].[customerID] = @customer_ID
AND [customer].[customerID]= @customer_ID;
END
```

Result:

```
EXEC GetCustomerFeedbackWith 6
-----------------------------END OF SP1-----------------------------
```

100 %

Results | Messages

	customerID	CustomerFullName	customerPhoneNo	customerEmail	orderID	productRating	shippingRating	experienceRating
1	6	Hiroko Potter	3142446306	hiroko@gmail.com	7	5.0	4.5	5.0

STORED PROCEDURE NO:2

Explanation: Gets CustomerID as parameter and displays CUSTOMER INFORMATION and ADDRESS BASED ON THE @customer_ID input parameter

```
CREATE PROCEDURE
GetCustomerInformation @customer_ID INT
AS
BEGIN
Select customer.customerID, dbo.CustomerFullName (customer.customerID) AS
CustomerFullName, customerPhoneNO, customerEmail, street, city, postalCode
FROM Customer JOIN customerAddress
ON [Customer].[customerID] = @customer_ID and [customerAddress].[customerID] =
@customer_ID;
END
```

Result:

```
END
EXEC GetCustomerInformation 10
----------------------------END OF SP2------------------------
------ GETS orderID as parameter and displays PAYMENT STATUS INFORMATION BASED ON THE (
```

100 %

Results | Messages

	customerID	CustomerFullName	customerPhoneNO	customerEmail	street	city	postalCode
1	10	Aaron Hawkins	6606634518	aaron@gmail.com	Main street	Louisiana	67973

STORED PROCEDURE NO:3

Explanation:GETS orderID as parameter and displays PAYMENT STATUS INFORMATION BASED ON THE @order_ID inpur parameter.

```
CREATE PROCEDURE
GetPaymentStatusWith @order_ID INT
AS
BEGIN
```

```
SELECT paymentStatus, paymentMethod, paymentDate, paymentTime,
paymentError,p.creditCardNoID, c.encryptedCreditCardNo, orderTime, fulfillmentStatus
FROM orderDetails, payment p, [order], creditcard c
WHERE [orderDetails].[orderID] = @order_ID and p.[orderID] = @order_ID and
[order].orderID = @order_ID and p.creditCardNoID = c.creditCardNoID
END
```

Result:

```
END
EXEC GetPaymentStatusWith 11
```

100 %

Results | Messages

	paymentStatus	paymentMethod	paymentDate	paymentTime	paymentError	creditCardNoID	encryptedCreditCardNo	orderTime	fulfillmentStatus
1	Failed	VISA	2020-04-29	18:20:40.0000000	Unable to connect	10	0x009DA9A989D1844C90A5CC5A032D4A7302000000A26A1C...	17:15:36.0000000	Not Applied

STORED PROCEDURE NO:4

Explanation: Gets productID and new productPrice as parameters and UPDATES PRODUCT PRICE.

```
CREATE PROCEDURE
updateProductPrice @product_ID INT, @new_product_Price VARCHAR(10)
AS
BEGIN
DECLARE @currProductPrice VARCHAR(10);
SET @currProductPrice = (SELECT productPrice from product where productID = @product_ID);
Update product SET productPrice = @new_product_Price where productID = @product_ID;
SELECT productName, p.productID, productPrice AS NewProductPrice, productDescription,
unitsInStock, unitsInORder FROM product p JOIN
productStock ON [productStock].[productID] = @product_ID and p.[productID] = @product_ID;
END
```

RESULT:

```
BEGIN TRANSACTION
EXEC updateProductPrice 10, 1500
ROLLBACK
-----------------------END OF SP4-----------------
```

00 %

Results | Messages

	productName	productID	NewProductPrice	productDescription	unitsInStock	unitsInORder
1	Jayleane Shorts	10	1500	90% cotton, machine wash.	25	14

STORED PROCEDURE NO:5

Explanation:
Gets Experience Rating as parameter and displays the CustomerID and the OrderID based on that Experience Rating.

```
CREATE PROCEDURE
GetCustomerIDandOrderIDWithExperienceRating @Exp_Rating decimal(2,1)
AS
BEGIN
SELECT customerID, orderID, ExperienceRating FROM customerFeedback
WHERE [customerFeedback].[ExperienceRating] = @Exp_Rating;
END
```

Result:

```
EXEC GetCustomerIDandOrderIDWithExperienceRating 4.0
-----------------------------END OF SP5--------------------------
```

100 %

Results | Messages

	customerID	orderID	ExperienceRating
1	1	1	4.0
2	4	5	4.0
3	5	6	4.0
4	11	12	4.0
5	13	14	4.0
6	14	15	4.0

TRIGGERS:

Explanation:
This trigger is called on update of the product price. Check if the product price is not less than 0 and not greater that specified limit.

```
CREATE TRIGGER
CheckProductPriceChanges
ON product
AFTER UPDATE
AS
    DECLARE @productPrice INT
    SET @productPrice=(select productPrice from inserted)
    IF( @productPrice < 0)
       BEGIN
              UPDATE product SET productPrice = 0
       END
    IF(@productPrice > 10000)
       BEGIN
              UPDATE product SET productPrice=10000
       END
```

Result:

a.When the product price is given below 0, for example, say -5, the price gets updated as 0. This is because of the trigger "`CheckProductPriceChanges`" which checks the update on the price change of the product.

```
EXEC updateProductPrice 3,-5
```

100 %

Results | Messages

	productName	productPrice	productDescription	unitsInStock	unitsInORder
1	Roadster Shirts	0	88% polyester,12% spandex,machine wash.	20	6

b.When the product price is given above 10,000, for example, say 12,000, the price gets updated as 10,000. This is because of the trigger "`CheckProductPriceChanges`" which checks the update on the price change of the product.

```
EXEC updateProductPrice 3,12000
```

100 %

Results | Messages

	productName	productPrice	productDescription	unitsInStock	unitsInORder
1	Roadster Shirts	10000	88% polyester,12% spandex,machine wash.	20	6

VIEWS:

VIEW 1:

Explanation: This view displays all customer information and their credit card information.

```
CREATE VIEW
CustomersAndTheirCreditCards
AS
Select dbo.CustomerFullName (customer.customerID) AS CustomerFullName, customerPhoneNo,
customerEmail, C.encryptedcreditCardNo, creditCardType, setAsPrimary, cardExpiry
FROM Customer JOIN creditCard C
ON [Customer].[customerID] = C.[customerID];
```

Result:

```
SELECT * FROM CustomersAndTheirCreditCards
---------------------------------END OF VIEW 1-------------------------
```

100 %

Results | Messages

	CustomerFullName	customerPhoneNo	customerEmail	encryptedcreditCardNo	creditCardType	setAsPrimary	cardExpiry
1	Cecelia Chapman	8493221093	cecelia@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A730200000063BF1A5...	VISA	yes	11/21
2	Iris Watson	3725872335	iris@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A7302000000C5423B2...	VISA	yes	01/23
3	Celeste Slater	7867138616	celeste@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A7302000000BA3A8C...	VISA	yes	01/23
4	Theodore Lowe	7867138616	Theodore@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A73020000008EEC38...	MASTERCARD	no	03/25
5	Kyla Olsen	6543935734	kyla@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A7302000000F618713...	APPEX	yes	09/25
6	Cecelia Chapman	8493221093	cecelia@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A7302000000EEF1150...	MASTERCARD	no	11/25
7	Hiroko Potter	3142446306	hiroko@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A730200000083A0E2...	VISA	yes	09/25
8	Nyssa Vazquez	9472785929	nyssa@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A7302000000B2CB6B...	MASTERCARD	no	08/23
9	Lawrence Moreno	6845791879	lawrence@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A73020000000F59E3E...	APPEX	yes	02/24
10	Ian Somerhalder	3142444006	Ian@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A7302000000A26A1C8...	APPEX	no	04/27
11	Aaron Hawkins	6606634518	aaron@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A7302000000CA27325...	MASTERCARD	no	04/27
12	Hedy Greene	6082652215	hedy@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A730200000006310AB4...	VISA	yes	05/25
13	Melvin Porter	9591198364	melvin@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A7302000000989037O...	VISA	no	06/23
14	Keefe Sellers	4683532641	keefe@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A730200000001609FA2...	MASTERCARD	no	07/22
15	Joan Romero	2486754007	joan@gmail.com	0x009DA9A989D1844C90A5CC5A032D4A7302000000877323 1...	APPEX	no	02/25

VIEW 2:

Explanation: This view display all supplier information with their addresses.

```
CREATE VIEW
ShowSupplierInfoAndTheirAddress
AS
SELECT s.supplierID, dbo.SupplierFullName (s.supplierID) AS SupplierName,
supplierPhoneNo, supplierEmail, supplierUrl, street, city, postalCode
FROM supplier s JOIN supplierAddress
ON s.[supplierID] = [supplierAddress].[supplierID];
```

Result:

```
SELECT * FROM ShowSupplierInfoAndTheirAddress
```

100 %

Results | Messages

	supplierID	SupplierName	supplierPhoneNo	supplierEmail	supplierUrl	street	city	postalCode
1	1	Tamara Howe	1213431320	Tamara@gmail.com	www.THSuppliers.com	Antonio Street	Rose Ville	11523
2	2	Owen Babara	2233445566	owen@gmail.com	www.OwenSuppliers.com	First Street	Woburn	10923
3	3	Amara welson	9087654321	amara@gmail.com	www.amaraBoutique.com	Second Street	Framingham	19203
4	4	Neha Kishore	4793284123	neha@gmail.com	www.nehaclothins.com	Cedar Street	SomerVille	10923
5	5	Ria Lamba	9283018349	ria@gmail.com	www.riaBoutique.com	Parch Street	Mankota	10934
6	6	Mark Mayer	8237138236	mark@gmail.com	www.markSupplies.com	Third Street	Bessemer	29831
7	7	Leo Lama	1903892832	Leo@gmail.com	www.leoClothing.com	Long street	Atmore	37682
8	8	Jane Corry	8278139217	jane@gmail.com	www.janeBoutique.com	Sesame street	Auburn	19082
9	9	Kristine Walker	2837189782	Kristine@gmail.com	www.kristineBoutique.com	Fourth street	Clanton	92183
10	10	Ariya Sen	2918382321	ariya@gmail.com	www.ariyaClothing.com	Marlboro	Arlington	92138
11	11	Maureen Den	8237923811	maureen@gmail.com	www.maureenClothing.com	Pine Strret	Andover	92108
12	12	Rachel Doe	2371837927	rachel@gmail.com	www.rachelDaleClothing.com	Basket Street	Bedford	28932
13	13	Monica Fayer	3874923213	monica@gmail.com	www.monicaBoutique.com	Baker Street	Braintree	29183
14	14	Priya Welson	8317918278	priya@gmail.com	www.priyaCollection.com	Mellow Strret	Csambridge	82719

VIEW 3:

Explanation: This view displays all product information

```
CREATE VIEW
ViewProductInfoWithUnits
AS
SELECT p.productID, productName, categoryID, productPrice, productSize, discount,
productWeight, productPicture, productDescription, unitsInStock, unitsInOrder
FROM product p JOIN productStock
ON p.[productID] = [productStock].[productID];
```

Result:

```
SELECT * FROM ViewProductInfoWithUnits
--------------------------------END OF VIEW 3------------------------
```

100 %

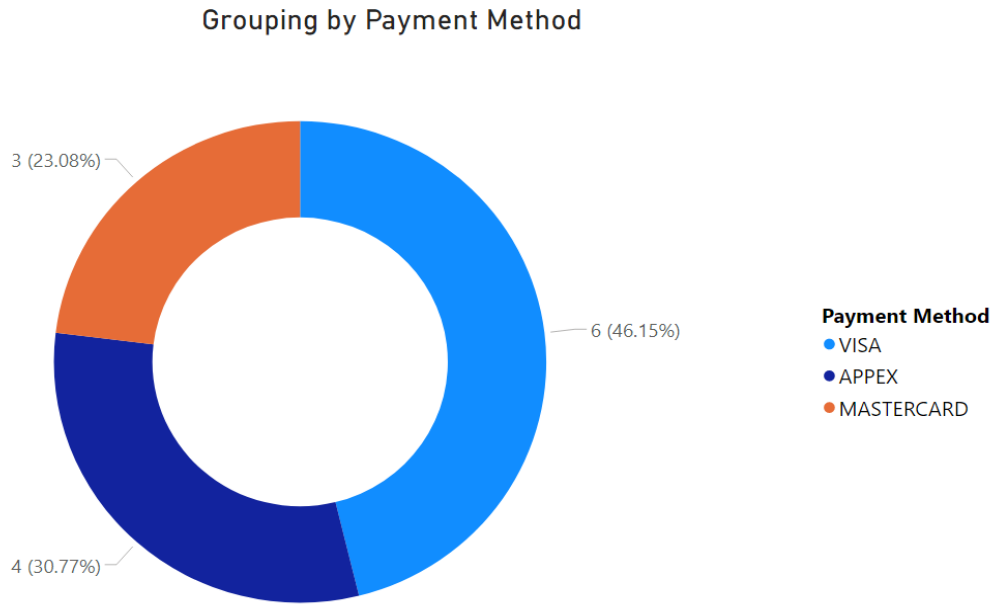
Results | Messages

	productID	productName	categoryID	productPrice	productSize	discount	productWeight	productPicture	productDescription	unitsInStock	unitsInOrder
1	1	Mustard T-Shirt	1	35	Small	10%	120gms	0xFFD8FFE000104A4649460001010100600600000FFDB008...	88% Polyster, 12% Spandex, Machine Wash	10	4
2	2	Mustard T-Shirt	1	40	Large	10%	130gms	0xFFD8FFE000104A4649460001010100600600000FFDB008...	88% polyester,12% spandex,machine wash.	10	4
3	3	Roadster Shirts	2	103	Medium	10%	130gms	0xFFD8FFDB00430005030404040305040404050505060700C0...	88% polyester,12% spandex,machine wash.	20	6
4	4	Forever Sweaters	3	40	large	10%	120gms	0xFFD8FFE000104A4649460001010000480048000 0FFED003...	88% polyester,12% wool, Hand wash.	40	15
5	5	Forever Sweatshirts	4	39	Medium	10%	130gms	0xFFD8FFE000104A4649460001010100600600000FFFE003...	90% cotton, machine wash.	10	2
6	6	Forever Sweatshirts	4	39	Medium	10%	130gms	0xFFD8FFE000104A4649460001010000010001000 0FFDB008...	90% cotton, machine wash.	30	20
7	7	Forever Sweatshirts	4	42	large	10%	130gms	0xFFD8FFDB00430008060607060508070707090908 0A0C140...	90% cotton, machine wash.	35	25
8	8	Polo Jackets	5	40	medium	10%	150gms	0xFFD8FFE000104A4649460001010100480048000 0FFDB008...	90% Denim, 10% Cotton machine wash.	30	5
9	9	Casual Trousers	6	70	medium	20%	130gms	0xFFD8FFE000104A4649460001010100480048000 0FFDB004...	90% cotton, machine wash.	20	14
10	10	Jayleane Shorts	7	70	medium	20%	130gms	0xFFD8FFE000104A4649460001010100480048000 0FFFE003...	90% cotton, machine wash.	25	14
11	11	Joggers	8	60	medium	20%	130gms	0xFFD8FFE000104A4649460001010100480048000 0FFDB004...	90% cotton, machine wash.	30	24
12	12	Dresses	9	60	medium	20%	130gms	0xFFD8FFE000104A4649460001010000010001000 0FFDB004...	90% cotton, machine wash.	35	4
13	13	Jumpsuits	10	30	medium	20%	130gms	0xFFD8FFE000104A4649460001010000480048000 0FFED003...	90% cotton, machine wash.	40	14
14	14	Skrits	11	35	medium	20%	130gms	0xFFD8FFE000104A4649460001010000480048000 0FFDB004...	90% cotton, machine wash.	20	13
15	15	Shrugs	12	35	medium	20%	130gms	0xFFD8FFE000104A4649460001010100600600000FFE110D...	90% cotton, machine wash.	30	10

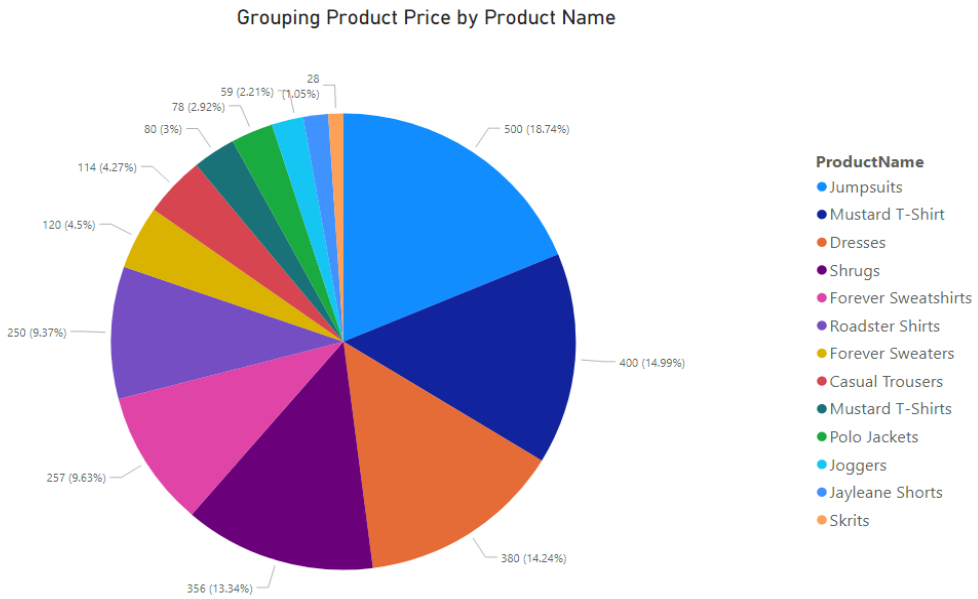
POWER BI VIEWS:

The server and the database were imported in the PowerBI which aided in creating several vies of visualization to present the database.

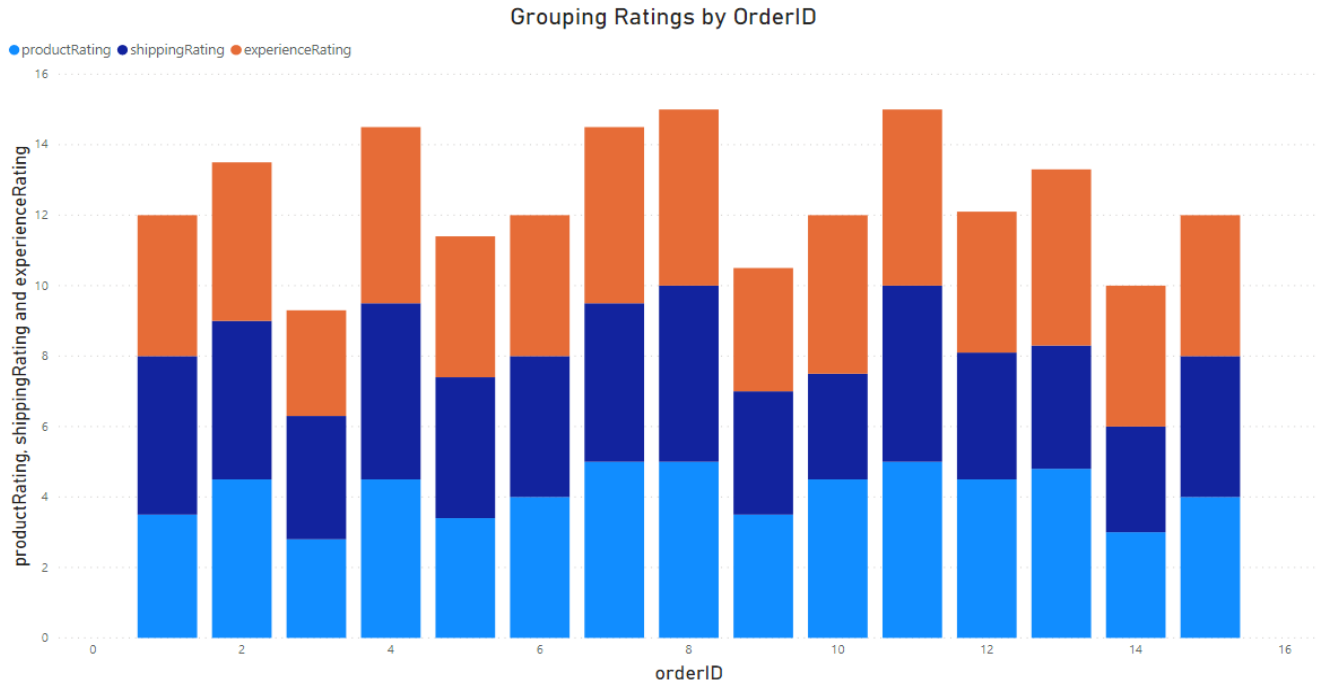
Visualization 1: Displaying the percentage of the payment methods (VISA, APPEX and MASTERCARD. This shows that most of the payments are made by VISA.



Visualization 2: Displaying the Pie chart of the product name based on the price of the product.



Visualization 3: This bar chart helps us to know the different ratings (according to product, shipping and experience) given by the customer , which is grouped by the orderID



Visualization 4: This bar chart helps us to know the count of the products grouped by the product size. We see that the medium size has the highest number (11) of products as per the given data.