Project Description

The purpose of this project is creating a database system, which will be implemented in MS SQL and proper integrations to a website for Volkan Kozmetik. Database will store the sell and stock information of the company that is specialized in personal care products and cologne.

Scope of the Project

We will be using Microsoft SQL to create our database. After entities and relations are described, the website should be ready. Each customer and company should see which categories are available. Each category should also include products that are produced and not out of stock. At the last step of the project we must implement both systems to each other.

Business Requirements

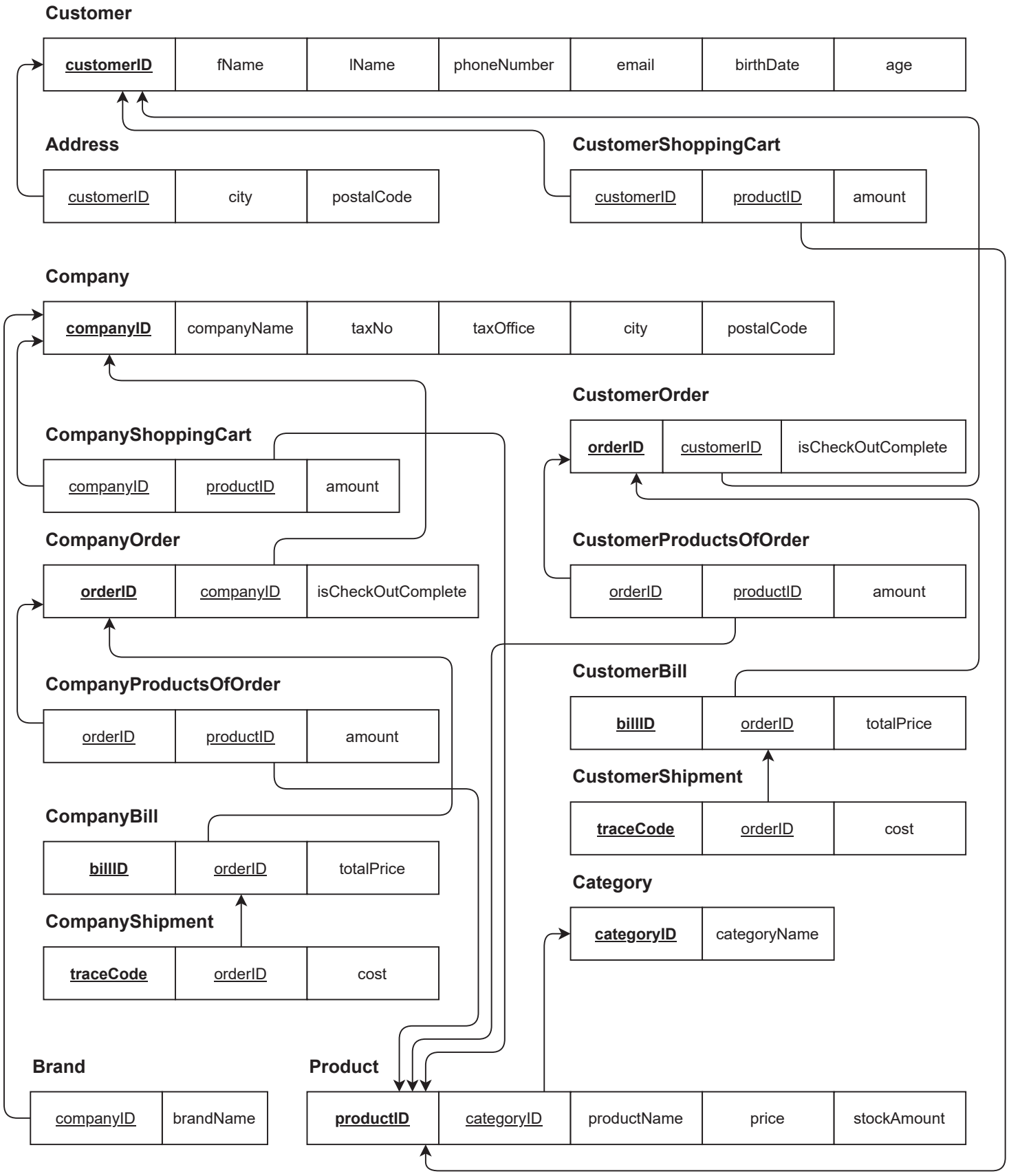
Functional Business Requirement

1. **Extensible:** The database must be restatable and changeable according to the information of customers and companies. (Must)
2. **Accessible:** The employees who have access can reach the database from the website. (Must)
3. **Multiple User Support:** The employees or the customers can make work on the website at the same time. (Must)
4. **Data Change Support:** Forms will be used to allow users to change records according to their access level. (Must)
5. **Sorting:** Sorts should be generated quickly for allowing managers to reach first order given, highest amount etc. (Must)

Non-Functional Business Requirement

1. **Microsoft SQL Server:** The database must be created using Microsoft SQL Server 2019. (Must)
2. **Entity Relation Diagram:** Create a diagram that includes the relations of the entities in the database. (Must)
3. **Table for Each Entity:** Create tables in the database according to the entity names with correct attributes. (Must)
4. **Achievable from Website:** Data must be achievable from website. (Must)
5. **Requirement Analysis Design:** Introduction of the project, list of entities and their definitions, list of functional and nonfunctional requirements. (Must)

Volkan Kozmetik Database Mapping Diagram



Tables

Address: The address of the customer that can be multivalued.
Columns: customerID (int), city (nvarchar(50)), postalCode (int).
Keys: customerID (FK)

Brand: The company's brand for representing produced products.
Columns: companyID (int), brandName (nvarchar(50))
Keys: companyID (FK)

Category: Cluster of products.
Columns: categoryID (int), categoryName (nvarchar(50))
Keys: companyID (FK)
Company: The table that buys huge amounts of products and uses the factory for their brands to be produced.
Columns: companyID (int) (+Identity) , taxNo (bigint) (+Unique), companyName (nvarchar(50))(+Unique), taxOffice (nvarchar(50)), address (nvarchar(50)).
Keys: companyID (PK)

Company Bill: Company bills are produced when the company checkouts the order.
Columns: billID(int) (+Identity), orderID(int), totalPrice(decimal(10,2)).
Keys: billID(PK), orderID(FK)

Company Order: Company orders are produced when the company creates an order for a product to be produced.
Columns: orderID(int) (+Identity) , companyID(int), isCheckOutComplete(int).
Keys: orderID(PK), companyID(FK)

Company Product Of Order: The product order that is the single element of the order.
Columns: orderID(int) , productID(int), amount(int).
Keys: orderID(FK), productID(FK)
Trigger: *sp_autoDecrementStockAfterCompanyOrder*

Company Shipment: The shipment of the company that is created after the checkout is complete.
Columns: traceCode(int) (+Identity) , orderID(int), cost(int).
Keys: traceCode(PK), orderID(FK)

Company Shopping Card: The shopping card of the company created when the company representative is adding products.
Columns: companyID(int), productID(int), amount(int).
Keys: companyID(FK), productID(FK)

Customer: The table for the person who buys small amounts of products and regular customers.
Columns: customerID(int) (+Identity)), fName (nvarchar(50)), lName (nvarchar(50)), phoneNumber(nvarchar(50)), email(nvarchar(50)), birthDate(smalldatetime), age (int)(+Computed)
Keys: customerID (PK)

Customer Bill: Customer bills are produced when the customer checkouts the order.
Columns: billID(int) (+Identity), orderID(int), totalPrice(decimal(10,2)).
Keys: billID(PK), orderID(FK)

Customer Order: Customer orders are produced when the customer creates an order for a product to be produced.
Columns: orderID(int) (+Identity) , customerID(int), isCheckOutComplete(int).
Keys: orderID(PK), customerID(FK)
Trigger: *sp_autoDecrementStockAfterCustomerOrder*

Customer Product Of Order: The product order that is the single element of the order.
Columns: orderID(int) , productID(int), amount(int).
Keys: orderID(FK), productID(FK)
Customer Shipment: The shipment of the customer that is created after the checkout is complete.
Columns: traceCode(int) (+Identity) , orderID(int), cost(int).
Keys: traceCode(PK), orderID(FK)

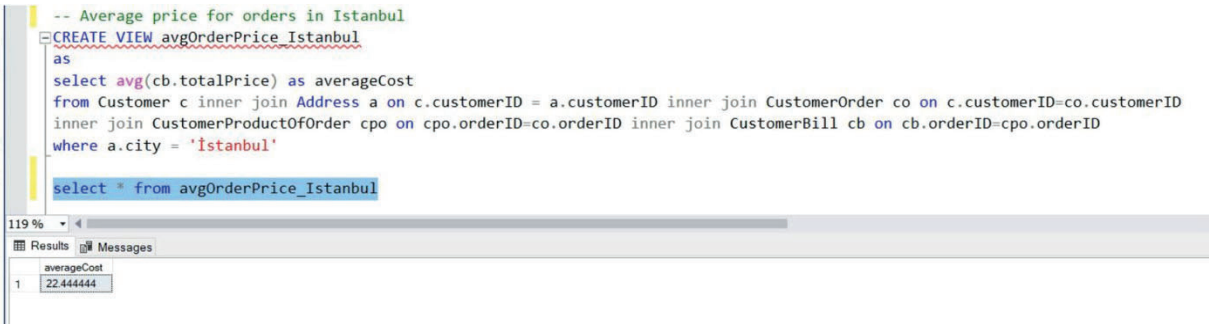
Customer Shopping Card: The shopping card of the customer created when the customer is adding products.
Columns: customerID(int), productID(int), amount(int).
Keys: customerID(FK), productID(FK)

Product: The shopping card of the customer created when the customer is adding products.
Columns: productID(int), categoryID(int), stockAmount(int), productName (nvarchar(50)), price (decimal(10,2)).
Keys: productID(PK), categoryID(FK)

Views

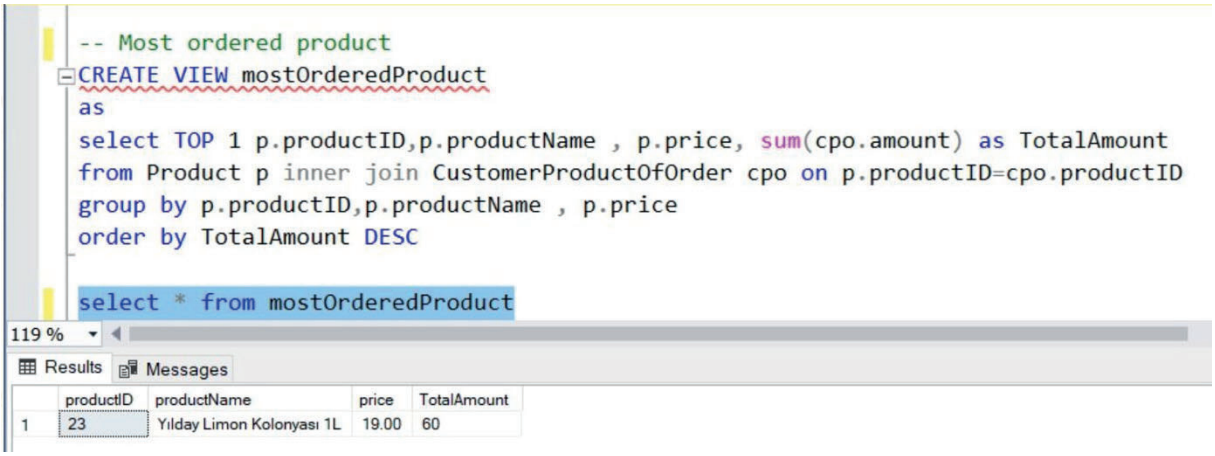
1-) avgOrderPrice_Istanbul

Display the average price of the all orders which was given in Istanbul.



2-) mostOrderedProduct

Display the most ordered product



3-) orderCountByCity

Display the order counts according to the cities

```
CREATE VIEW orderCountByCity
as
select a.city , count(*) as totalOrderCity
from Customer c inner join Address a on c.customerID= a.customerID inner join CustomerOrder co on c.customerID=co.customerID
inner join CompanyProductOfOrder cpo on co.orderID = cpo.orderID
inner join CustomerBill cb on cb.orderID = cpo.orderID inner join CustomerShipment cs on cs.orderID=cb.orderID
group by a.city

select * from orderCountByCity
```

119 %

Results | Messages

	city	totalOrderCity
1	İstanbul	18
2	İzmir	11

4-) orderCountByCategory

Display the order counts according to the categories

```
-- Number of total orders according to their categories
CREATE VIEW orderCountByCategory
as
select  c.categoryName  , sum(cpo.amount) as TotalCategorOrderAmont
from Product p inner join Category c on p.categoryID = c.categoryID
inner join CustomerProductOfOrder cpo on cpo.productID = p.productID
group by c.categoryName

select * from orderCountByCategory
```

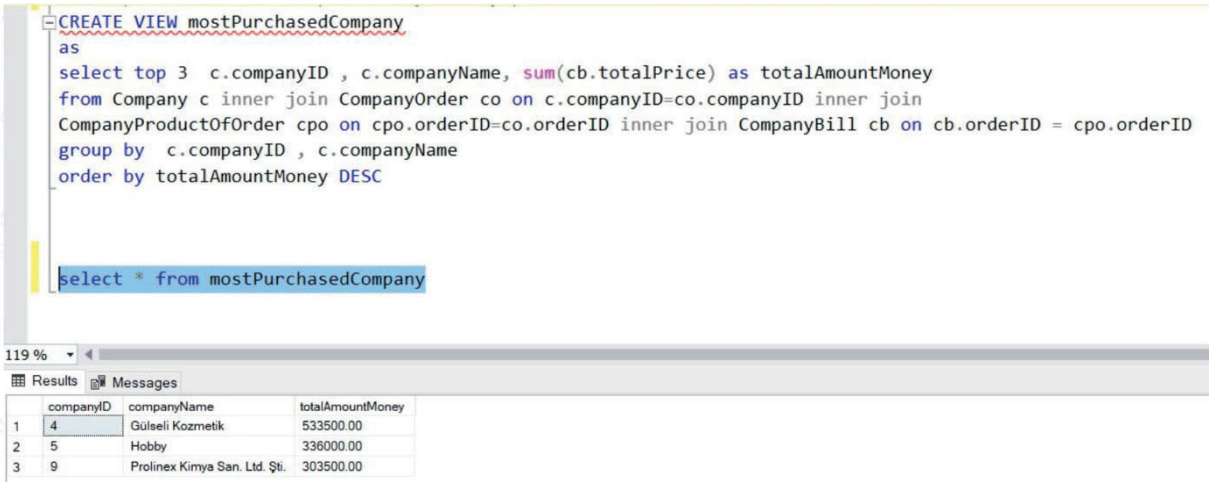
119 %

Results | Messages

	categoryName	TotalCategorOrderAmont
1	Cologne	129
2	Personal Care Product	18

5-) mostPurchasedCompany

Display the top 3 companies that spended most money



```
CREATE VIEW mostPurchasedCompany
as
select top 3  c.companyID , c.companyName, sum(cb.totalPrice) as totalAmountMoney
from Company c inner join CompanyOrder co on c.companyID=co.companyID inner join
CompanyProductOfOrder cpo on cpo.orderID=co.orderID inner join CompanyBill cb on cb.orderID = cpo.orderID
group by  c.companyID , c.companyName
order by totalAmountMoney DESC



select * from mostPurchasedCompany
```

119 %

Results | Messages

	companyID	companyName	totalAmountMoney
1	4	Gülseli Kozmetik	533500.00
2	5	Hobby	336000.00
3	9	Prolinex Kimya San. Ltd. Şti.	303500.00

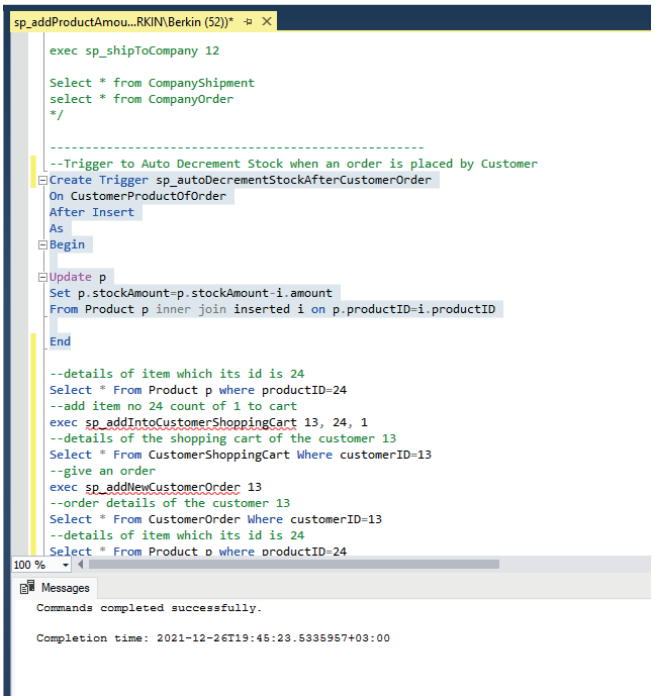
Triggers

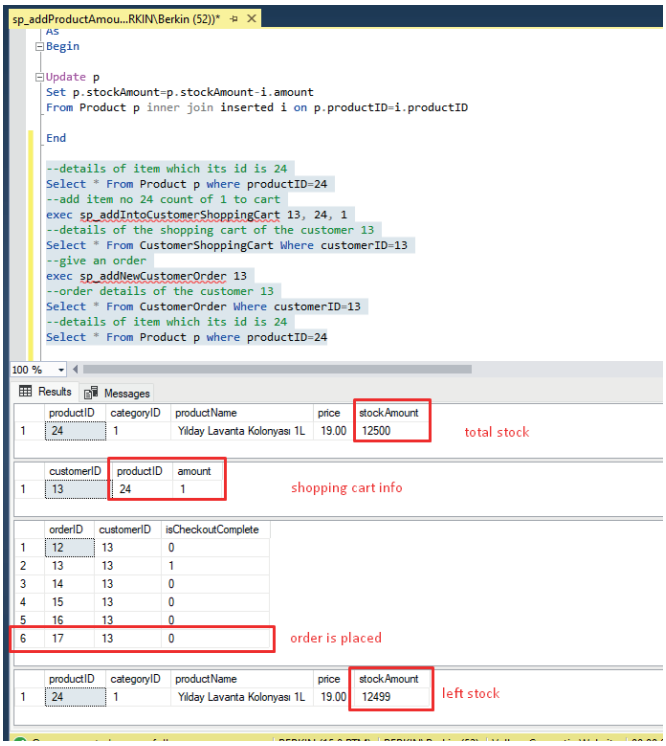
Trigger-1 : *sp_autoDecrementStockAfterCustomerOrder*

Definition: This trigger updates the product stock amount when a customer orders a product.

How it works: By keeping track of what has been inserted to the customer's ordered product(s) list, trigger gets an inserted table. With the help of this table, trigger finds which product(s) have been purchased and decrements the stock amount of those products according to this situation.

Screenshots:



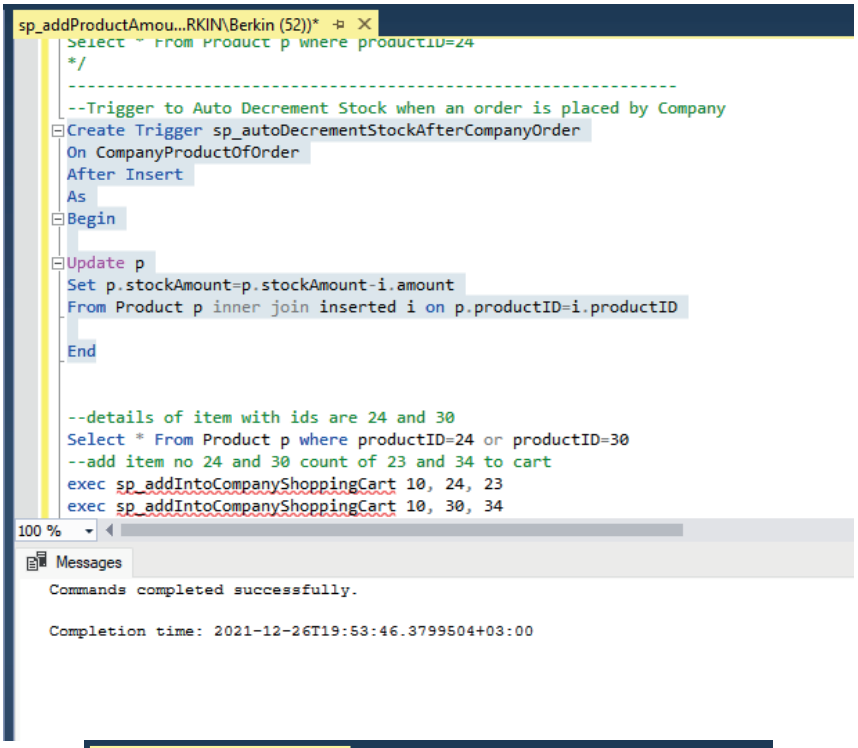


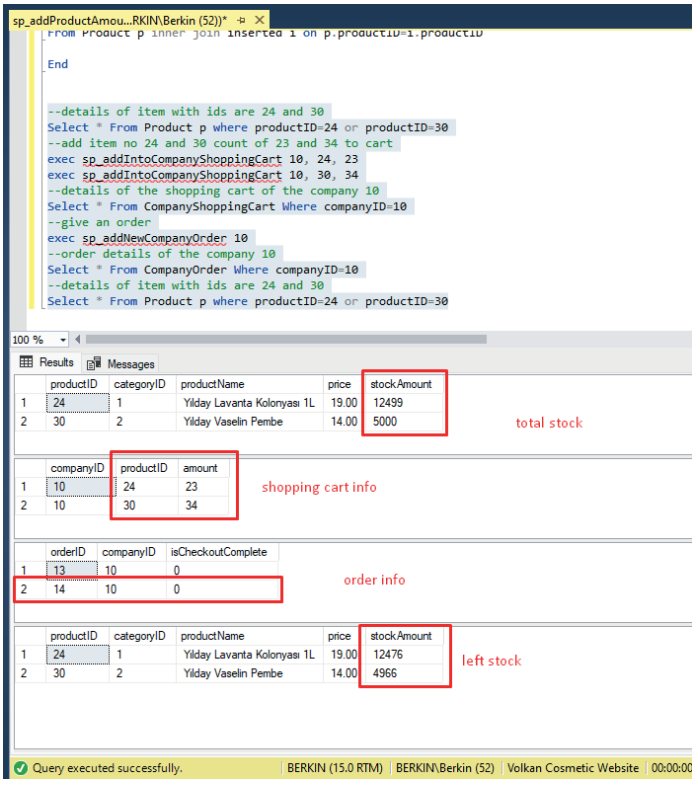
Trigger-2 : *sp_autoDecrementStockAfterCompanyOrder*

Definition: This trigger updates the product stock amount when a company orders a product.

How it works: By keeping track of what has been inserted to the company's ordered product(s) list, trigger gets an inserted table. With the help of this table, trigger finds which product(s) have been purchased and decrements the stock amount of those products according to this situation.

Screenshots:





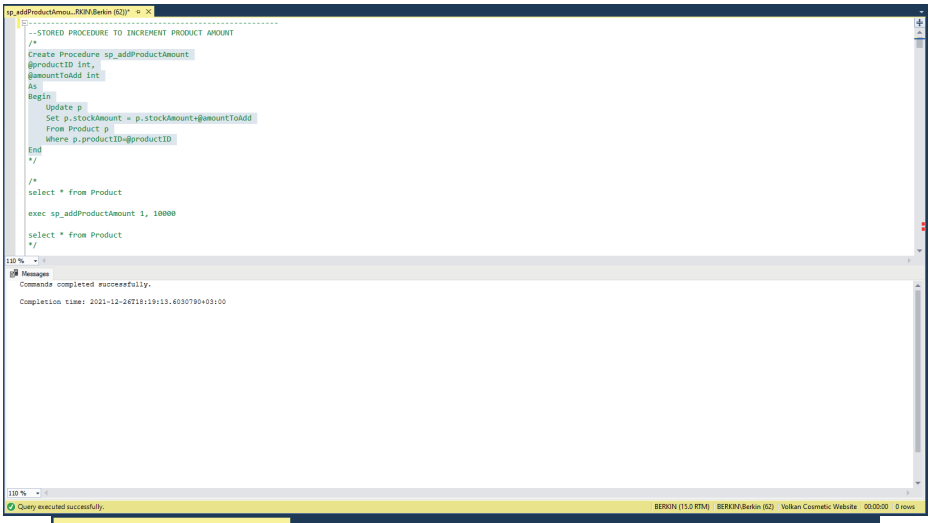
Stored Procedures

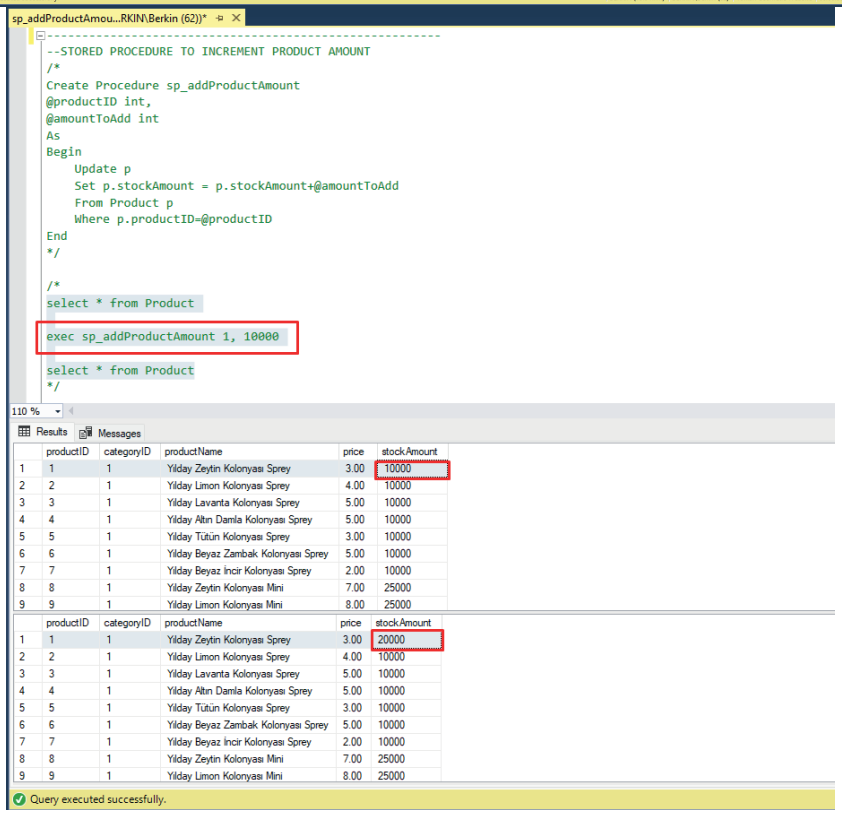
Stored Procedure-1: *sp_addProductAmount(productId, amountToAdd)*

Definition: Increments a product's stock amount by the given amount.

How it works: Works by updating the product table's stock amount.

Screenshots:





```
--STORED PROCEDURE TO INCREMENT PRODUCT AMOUNT
/*
Create Procedure sp_addProductAmount
@productID int,
@amountToAdd int
As
Begin
    Update p
    Set p.stockAmount = p.stockAmount+@amountToAdd
    From Product p
    Where p.productID=@productID
End
*/

/*
select * from Product

exec sp_addProductAmount 1, 10000

select * from Product
*/
```

Results | Messages

	productID	categoryID	productName	price	stockAmount
1	1	1	Yıldız Zeytin Kolonyası Sprey	3.00	10000
2	2	1	Yıldız Limon Kolonyası Sprey	4.00	10000
3	3	1	Yıldız Lavanta Kolonyası Sprey	5.00	10000
4	4	1	Yıldız Altın Damla Kolonyası Sprey	5.00	10000
5	5	1	Yıldız Tütün Kolonyası Sprey	3.00	10000
6	6	1	Yıldız Beyaz Zambak Kolonyası Sprey	5.00	10000
7	7	1	Yıldız Beyaz İncir Kolonyası Sprey	2.00	10000
8	8	1	Yıldız Zeytin Kolonyası Mini	7.00	25000
9	9	1	Yıldız Limon Kolonyası Mini	8.00	25000

	productID	categoryID	productName	price	stockAmount
1	1	1	Yıldız Zeytin Kolonyası Sprey	3.00	20000
2	2	1	Yıldız Limon Kolonyası Sprey	4.00	10000
3	3	1	Yıldız Lavanta Kolonyası Sprey	5.00	10000
4	4	1	Yıldız Altın Damla Kolonyası Sprey	5.00	10000
5	5	1	Yıldız Tütün Kolonyası Sprey	3.00	10000
6	6	1	Yıldız Beyaz Zambak Kolonyası Sprey	5.00	10000
7	7	1	Yıldız Beyaz İncir Kolonyası Sprey	2.00	10000
8	8	1	Yıldız Zeytin Kolonyası Mini	7.00	25000
9	9	1	Yıldız Limon Kolonyası Mini	8.00	25000

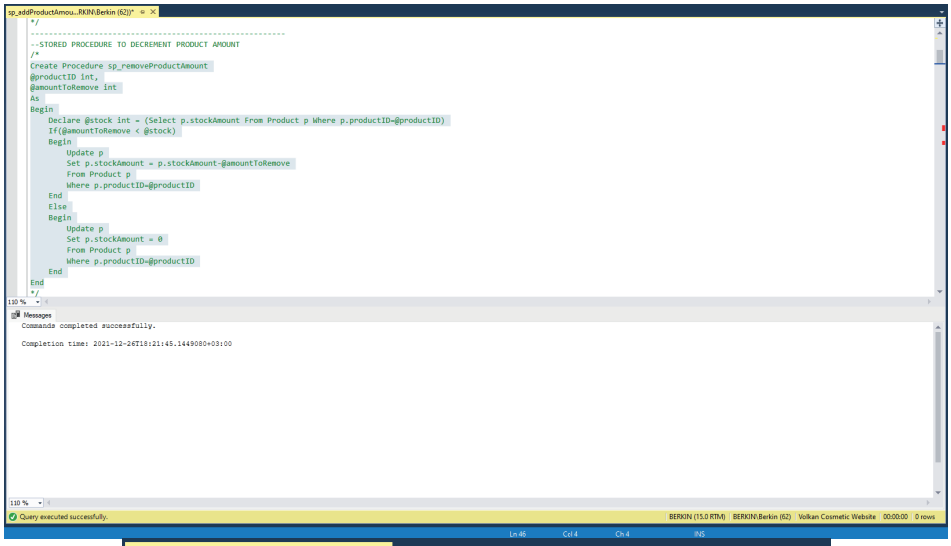
Query executed successfully.

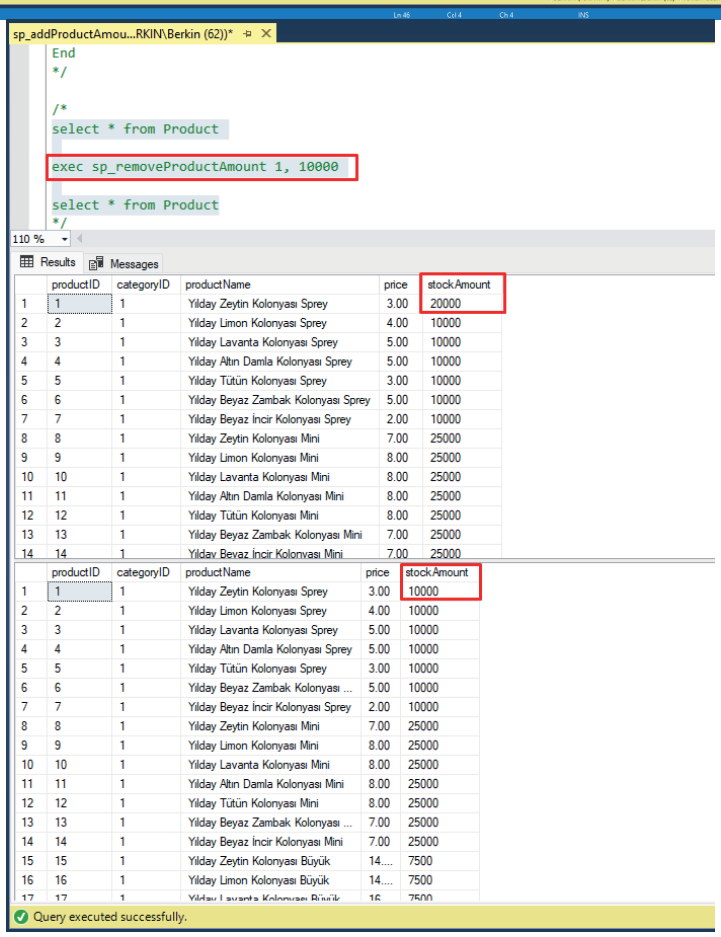
Stored Procedure-2: *sp_removeProductAmount (productId, amountToAdd)*

Definition: Decrements a product's stock amount by the given amount.

How it works: Works by updating the product table's stock amount.

Screenshots:



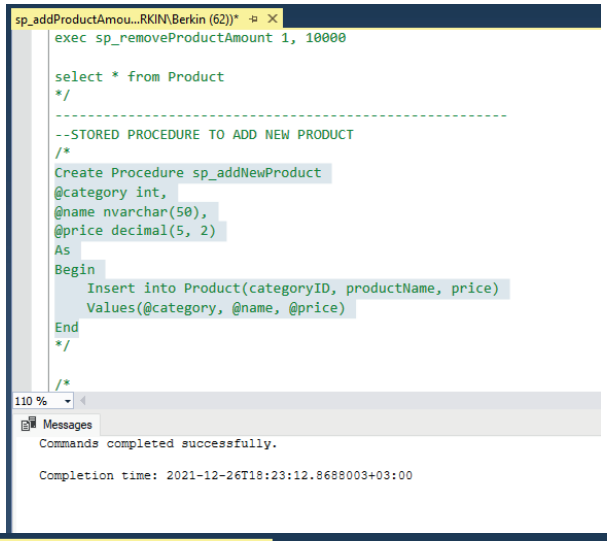


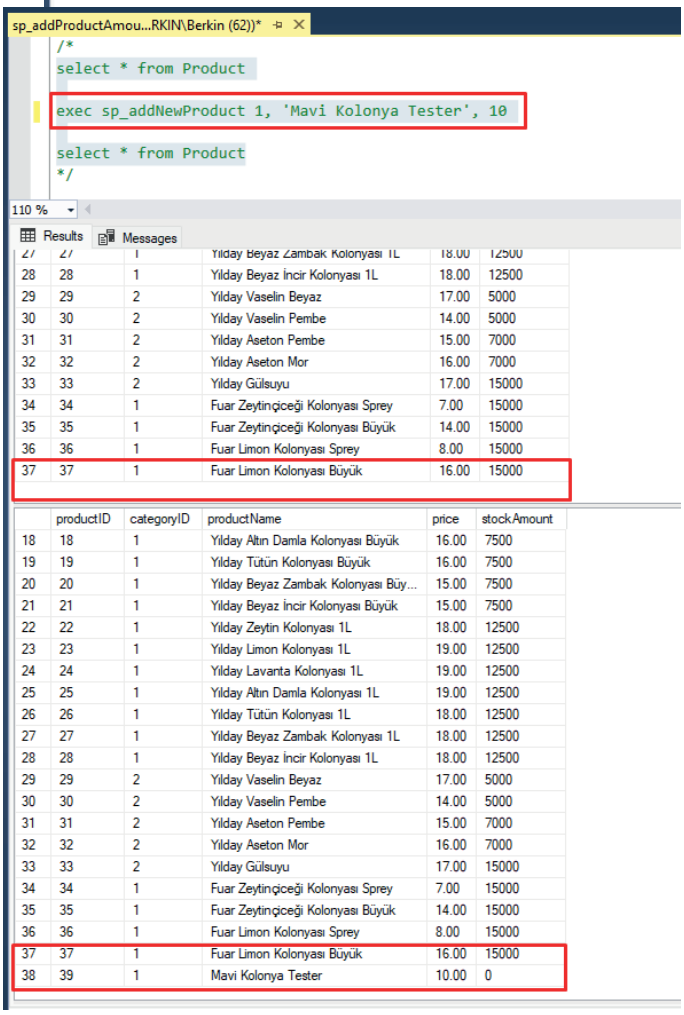
Stored Procedure-3: *sp_addNewProduct (category, name, price)*

Definition: Adds a new product.

How it works: Works by inserting given parametes into product table.

Screenshots:





28	28	1	Yılday Beyaz İncir Kolonyası 1L	18.00	12500
29	29	2	Yılday Vaselin Beyaz	17.00	5000
30	30	2	Yılday Vaselin Pembe	14.00	5000
31	31	2	Yılday Aseton Pembe	15.00	7000
32	32	2	Yılday Aseton Mor	16.00	7000
33	33	2	Yılday Gülsuyu	17.00	15000
34	34	1	Fuar Zeytinçiceği Kolonyası Sprey	7.00	15000
35	35	1	Fuar Zeytinçiceği Kolonyası Büyük	14.00	15000
36	36	1	Fuar Limon Kolonyası Sprey	8.00	15000
37	37	1	Fuar Limon Kolonyası Büyük	16.00	15000

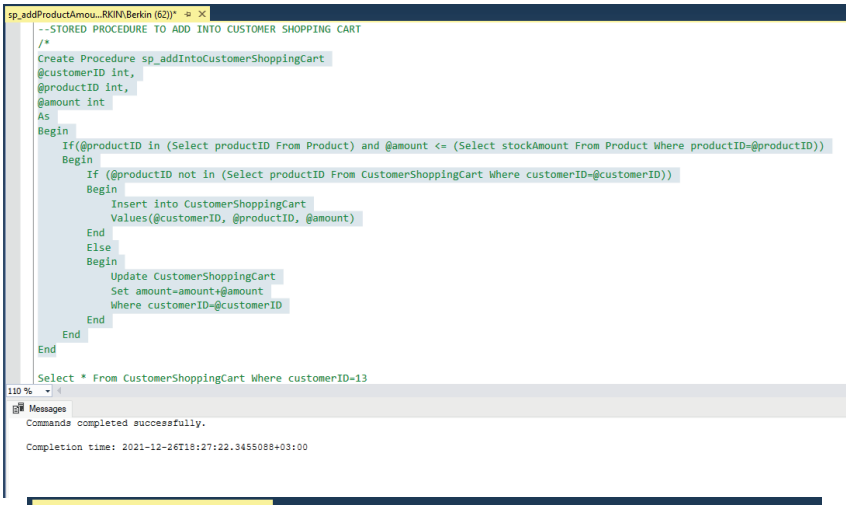
	productID	categoryID	productName	price	stockAmount
18	18	1	Yılday Altın Damla Kolonyası Büyük	16.00	7500
19	19	1	Yılday Tütün Kolonyası Büyük	16.00	7500
20	20	1	Yılday Beyaz Zambak Kolonyası Büy...	15.00	7500
21	21	1	Yılday Beyaz İncir Kolonyası Büyük	15.00	7500
22	22	1	Yılday Zeytin Kolonyası 1L	18.00	12500
23	23	1	Yılday Limon Kolonyası 1L	19.00	12500
24	24	1	Yılday Lavanta Kolonyası 1L	19.00	12500
25	25	1	Yılday Altın Damla Kolonyası 1L	19.00	12500
26	26	1	Yılday Tütün Kolonyası 1L	18.00	12500
27	27	1	Yılday Beyaz Zambak Kolonyası 1L	18.00	12500
28	28	1	Yılday Beyaz İncir Kolonyası 1L	18.00	12500
29	29	2	Yılday Vaselin Beyaz	17.00	5000
30	30	2	Yılday Vaselin Pembe	14.00	5000
31	31	2	Yılday Aseton Pembe	15.00	7000
32	32	2	Yılday Aseton Mor	16.00	7000
33	33	2	Yılday Gülsuyu	17.00	15000
34	34	1	Fuar Zeytinçiceği Kolonyası Sprey	7.00	15000
35	35	1	Fuar Zeytinçiceği Kolonyası Büyük	14.00	15000
36	36	1	Fuar Limon Kolonyası Sprey	8.00	15000
37	37	1	Fuar Limon Kolonyası Büyük	16.00	15000
38	39	1	Mavi Kolonya Tester	10.00	0

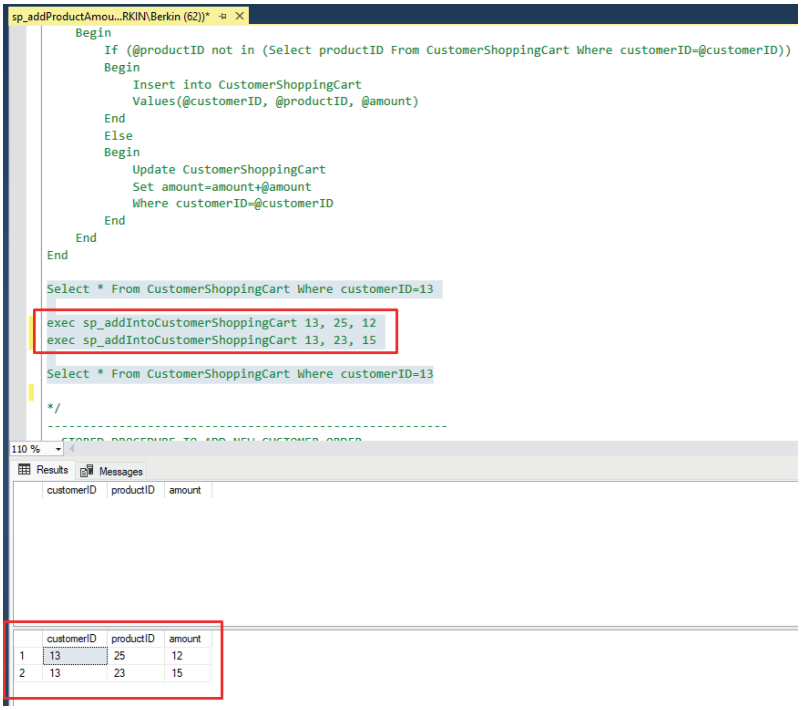
Stored Procedure-4: *sp_addIntoCustomerShoppingCart (customerID, productId, amount)*

Definition: Adds the given product by the given amount into the customer's shopping cart or updates existing amount.

How it works: Works by inserting to CustomerShoppingCart table or updating existing amount in the cart.

Screenshots:



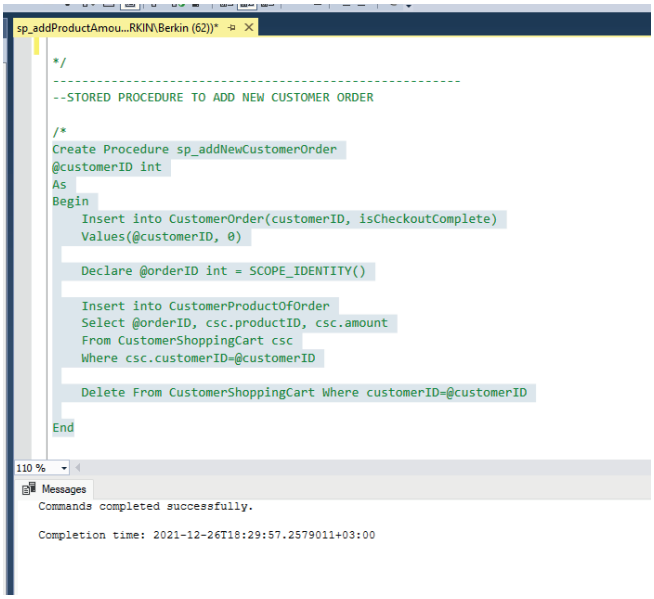


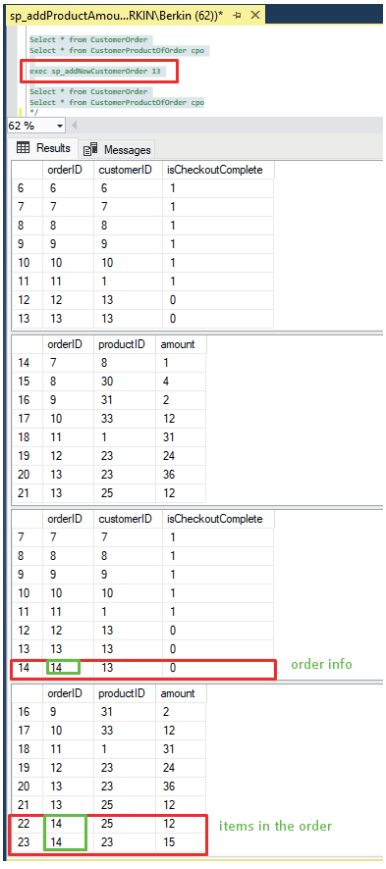
Stored Procedure-5: *sp_addNewCustomerOrder (customerID)*

Definition: Creates a new order for the customer by using the values in customer's shopping cart. Empties the cart.

How it works: Works by inserting to customer order table and products of the order table. Deletes the cart's values.

Screenshots:



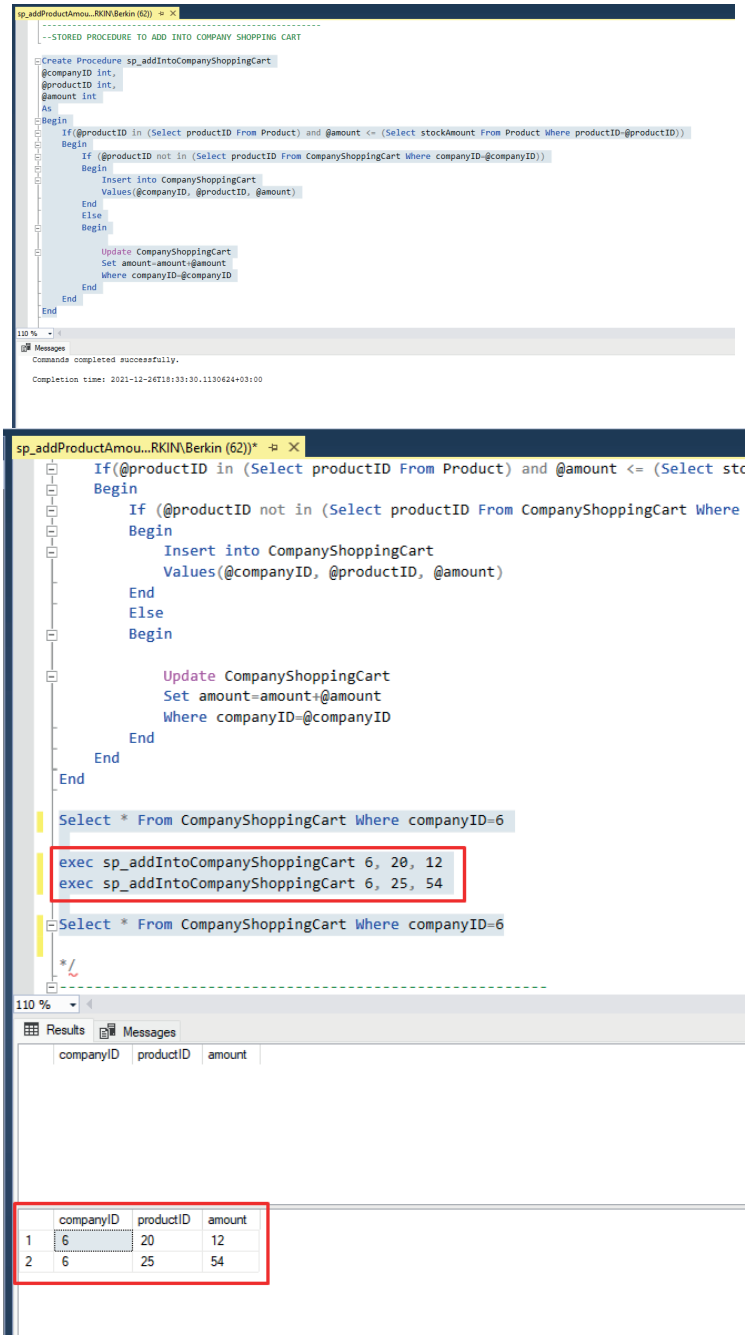


Stored Procedure-6: *sp_addIntoCompanyShoppingCart (companyID, productId, amount)*

Definition: Adds the given product by the given amount into the company's shopping cart or updates existing amount.

How it works: Works by inserting to CompanyShoppingCart table or updating existing amount in the cart.

Screenshots:

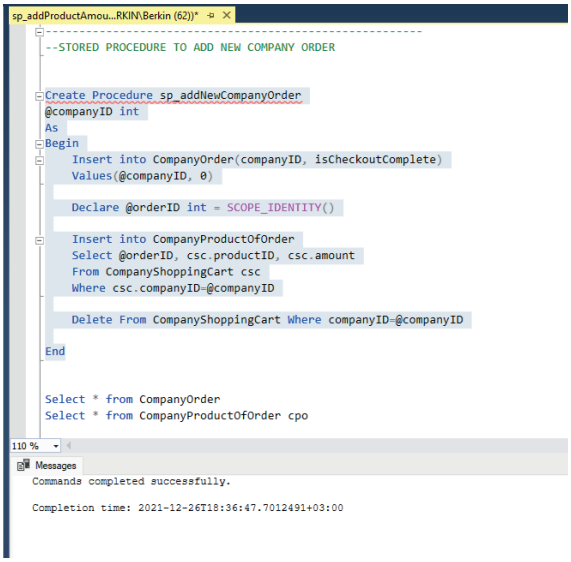


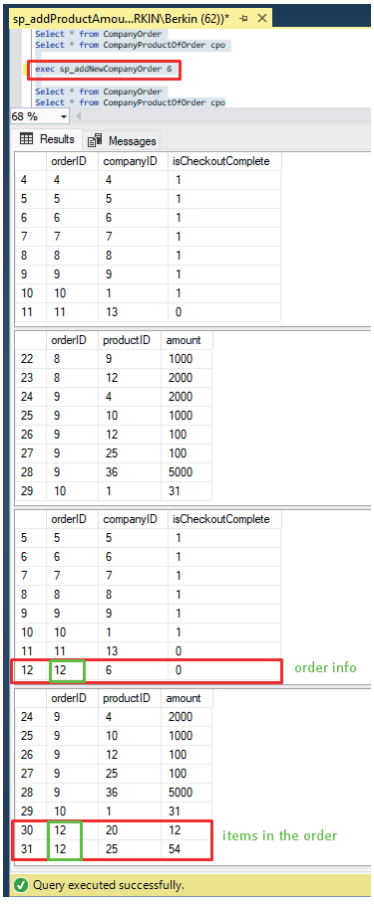
Stored Procedure-7: *sp_addNewCompanyOrder (companyID)*

Definition: Creates a new order for the company by using the values in company's shopping cart. Empties the cart.

How it works: Works by inserting to company order table and products of the order table. Deletes the cart's values.

Screenshots:



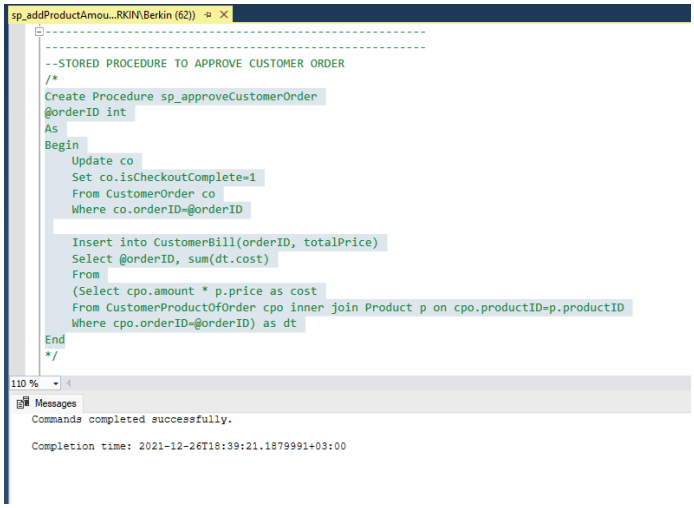


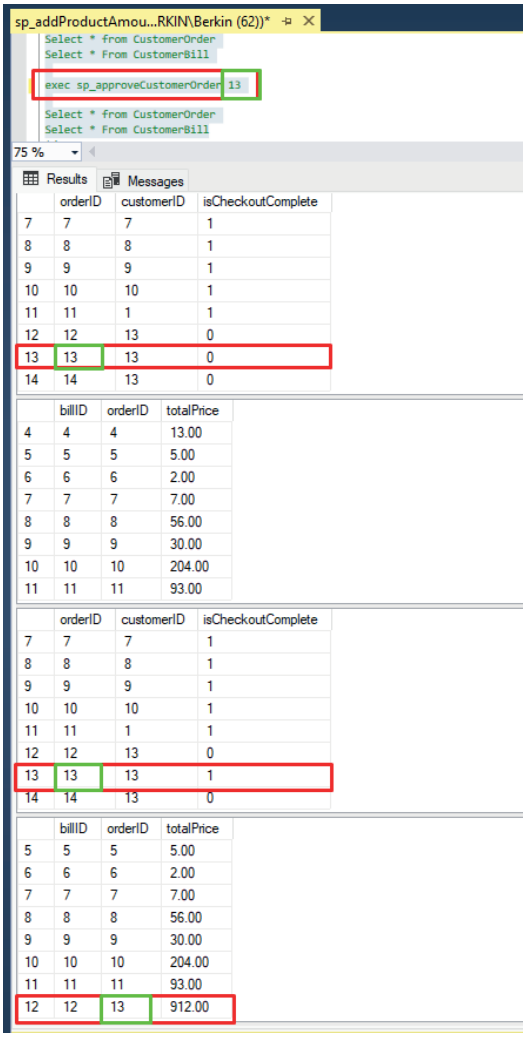
Stored Procedure-8: *sp_approveCustomerOrder (customerID)*

Definition: Completes the customer's checkout and creates a bill for the order.

How it works: Works by updating customer order table and inserting a new bill into the bill table. Calculates total cost for the products.

Screenshots:



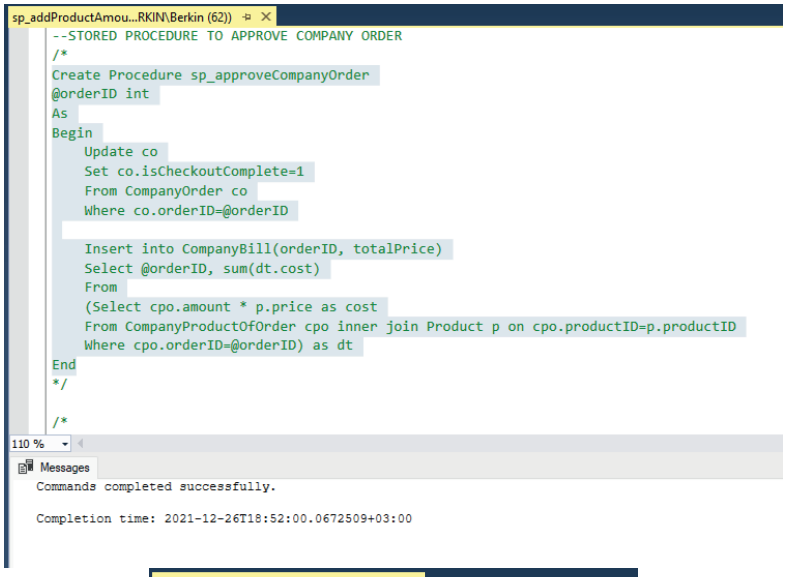


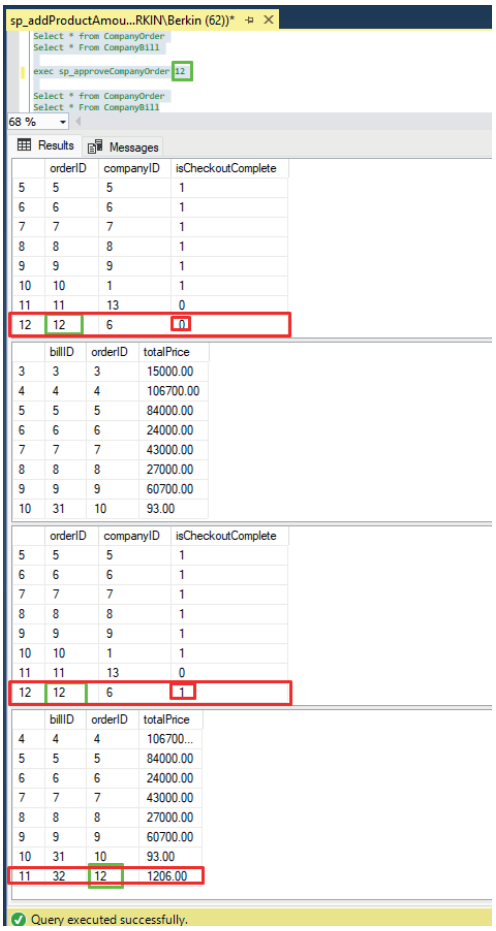
Stored Procedure-9: *sp_approveCompanyOrder (companyID)*

Definition: Completes the company's checkout and creates a bill for the order.

How it works: Works by updating company order table and inserting a new bill into the bill table. Calculates total cost for the products.

Screenshots:



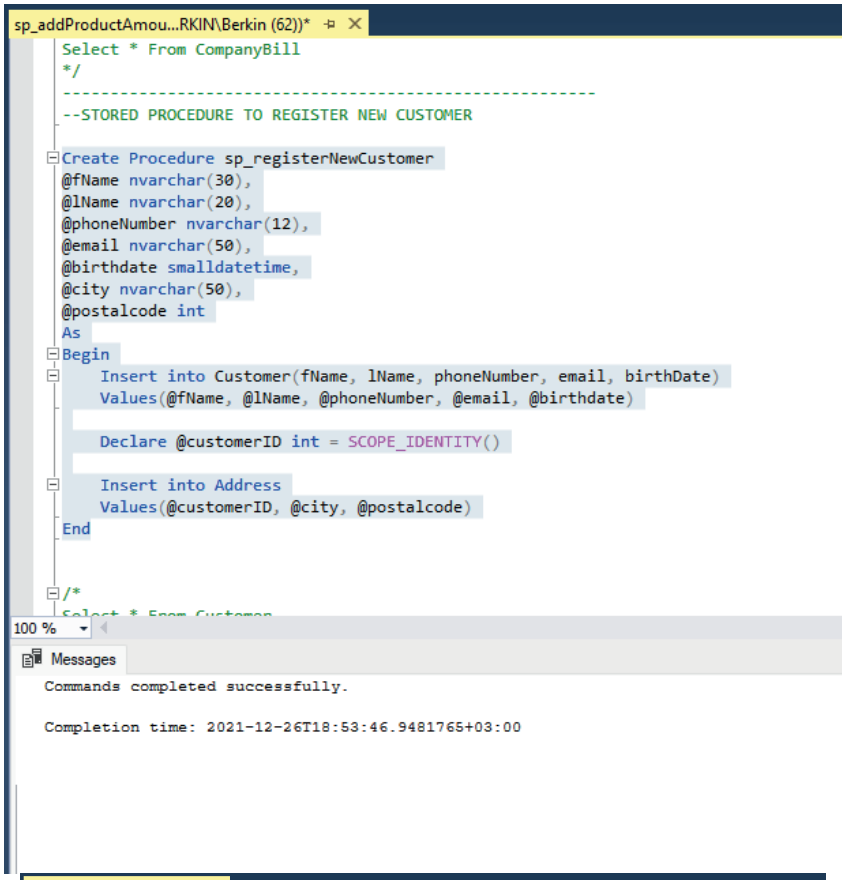


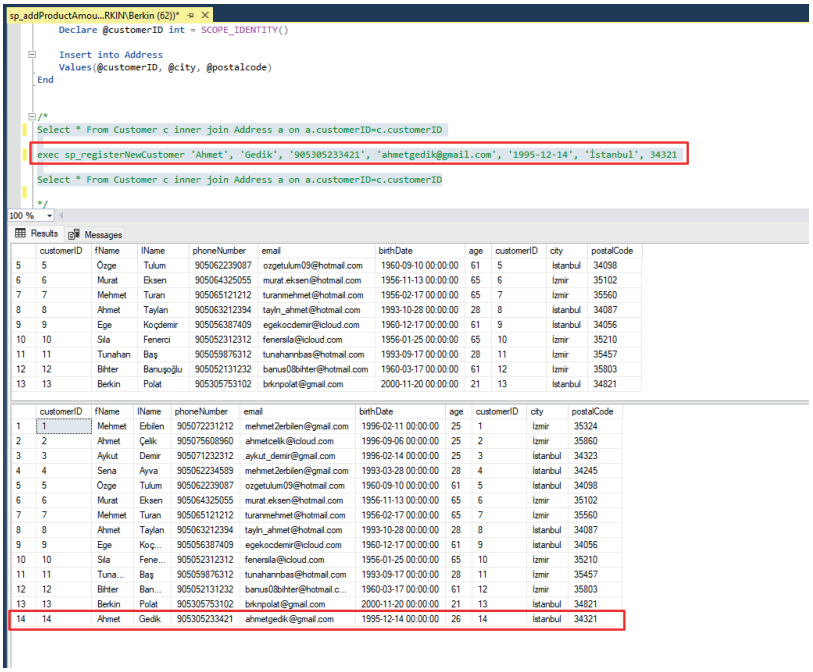
Stored Procedure-10: *sp_registerNewCustomer (fName, lName, phoneNumber, email, birthDate)*

Definition: Adds a new customer to the system.

How it works: Works by inserting personal details into Customer table and addres details into address table.

Screenshots:





	customerID	fName	lName	phoneNumber	email	birthDate	age	customerID	city	postalCode
5	5	Özge	Tulum	905062239087	ozgetulum09@hotmail.com	1960-09-10 00:00:00	61	5	İstanbul	34098
6	6	Murat	Eksen	905064325055	murat.eksen@hotmail.com	1956-11-13 00:00:00	65	6	İzmir	35102
7	7	Mehmet	Turan	905065121212	turanmehmet@hotmail.com	1956-02-17 00:00:00	65	7	İzmir	35560
8	8	Ahmet	Taylan	905063212394	tayln_ahmet@hotmail.com	1993-10-28 00:00:00	28	8	İstanbul	34087
9	9	Ege	Koçdemir	905056387409	egekocdemir@icloud.com	1960-12-17 00:00:00	61	9	İstanbul	34056
10	10	Sıla	Fenerci	905052312312	fenersila@icloud.com	1956-01-25 00:00:00	65	10	İzmir	35210
11	11	Tunahan	Baş	905059876312	tunahannbas@hotmail.com	1993-09-17 00:00:00	28	11	İzmir	35457
12	12	Bihter	Banuşoğlu	905052131232	banus08bihter@hotmail.com	1960-03-17 00:00:00	61	12	İzmir	35803
13	13	Berkin	Polat	905305753102	brknpolat@gmail.com	2000-11-20 00:00:00	21	13	İstanbul	34821

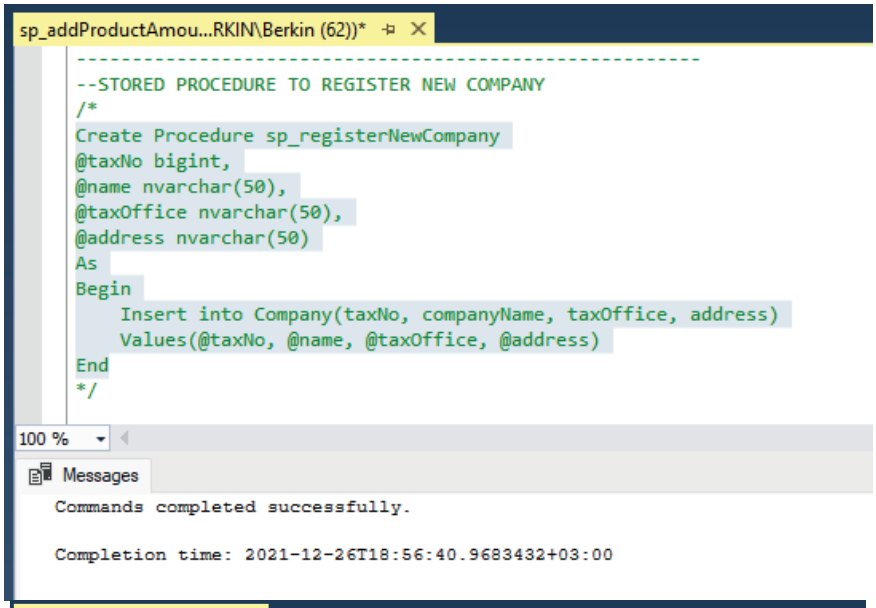
	customerID	fName	lName	phoneNumber	email	birthDate	age	customerID	city	postalCode
1	1	Mehmet	Erbilen	905072231212	mehmet2erbilen@gmail.com	1996-02-11 00:00:00	25	1	İzmir	35324
2	2	Ahmet	Çelik	905075608960	ahmetcelik@icloud.com	1996-09-06 00:00:00	25	2	İzmir	35860
3	3	Aykut	Demir	905071232312	aykut_demir@gmail.com	1996-02-14 00:00:00	25	3	İstanbul	34323
4	4	Sena	Ayva	905062234589	mehmet2erbilen@gmail.com	1993-03-28 00:00:00	28	4	İstanbul	34245
5	5	Özge	Tulum	905062239087	ozgetulum09@hotmail.com	1960-09-10 00:00:00	61	5	İstanbul	34098
6	6	Murat	Eksen	905064325055	murat.eksen@hotmail.com	1956-11-13 00:00:00	65	6	İzmir	35102
7	7	Mehmet	Turan	905065121212	turanmehmet@hotmail.com	1956-02-17 00:00:00	65	7	İzmir	35560
8	8	Ahmet	Taylan	905063212394	tayln_ahmet@hotmail.com	1993-10-28 00:00:00	28	8	İstanbul	34087
9	9	Ege	Koç...	905056387409	egekocdemir@icloud.com	1960-12-17 00:00:00	61	9	İstanbul	34056
10	10	Sıla	Fene...	905052312312	fenersila@icloud.com	1956-01-25 00:00:00	65	10	İzmir	35210
11	11	Tuna...	Baş	905059876312	tunahannbas@hotmail.com	1993-09-17 00:00:00	28	11	İzmir	35457
12	12	Bihter	Ban...	905052131232	banus08bihter@hotmail.c...	1960-03-17 00:00:00	61	12	İzmir	35803
13	13	Berkin	Polat	905305753102	brknpolat@gmail.com	2000-11-20 00:00:00	21	13	İstanbul	34821
14	14	Ahmet	Gedik	905305233421	ahmetgedik@gmail.com	1995-12-14 00:00:00	26	14	İstanbul	34321

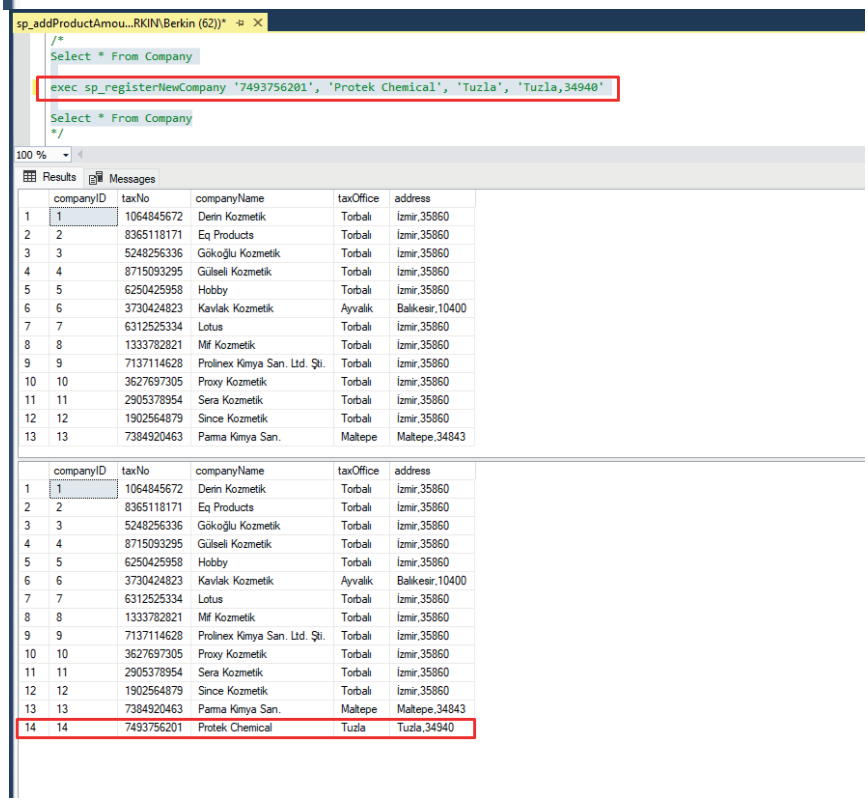
Stored Procedure-11: *sp_registerNewCompany (taxNo, companyName, taxOffice, address)*

Definition: Adds a new company to the system.

How it works: Works by inserting given company details into Company table.

Screenshots:





	companyID	taxNo	companyName	taxOffice	address
1	1	1064845672	Derin Kozmetik	Torbalı	İzmir,35860
2	2	8365118171	Eq Products	Torbalı	İzmir,35860
3	3	5248256336	Gökoğlu Kozmetik	Torbalı	İzmir,35860
4	4	8715093295	Gülseli Kozmetik	Torbalı	İzmir,35860
5	5	6250425958	Hobby	Torbalı	İzmir,35860
6	6	3730424823	Kavlak Kozmetik	Ayvalık	Balıkesir,10400
7	7	6312525334	Lotus	Torbalı	İzmir,35860
8	8	1333782821	Mif Kozmetik	Torbalı	İzmir,35860
9	9	7137114628	Prolinex Kimya San. Ltd. Şti.	Torbalı	İzmir,35860
10	10	3627697305	Proxy Kozmetik	Torbalı	İzmir,35860
11	11	2905378954	Sera Kozmetik	Torbalı	İzmir,35860
12	12	1902564879	Since Kozmetik	Torbalı	İzmir,35860
13	13	7384920463	Parma Kimya San.	Maltepe	Maltepe,34843

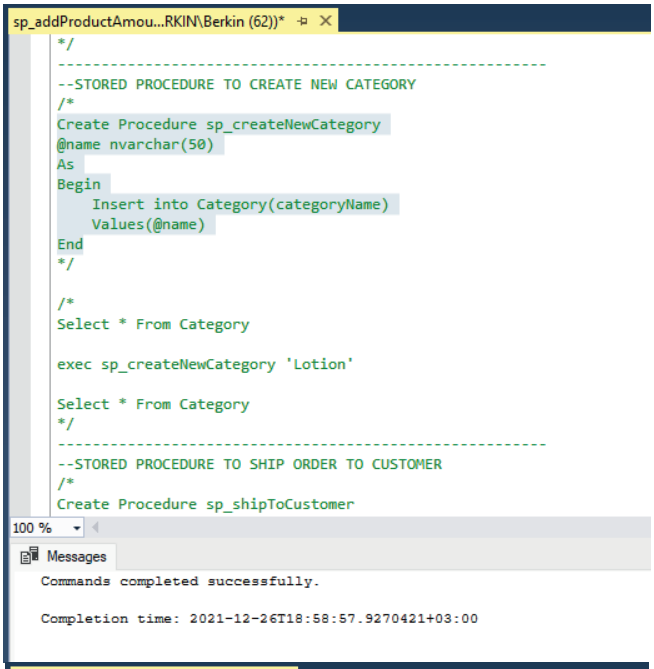
	companyID	taxNo	companyName	taxOffice	address
1	1	1064845672	Derin Kozmetik	Torbalı	İzmir,35860
2	2	8365118171	Eq Products	Torbalı	İzmir,35860
3	3	5248256336	Gökoğlu Kozmetik	Torbalı	İzmir,35860
4	4	8715093295	Gülseli Kozmetik	Torbalı	İzmir,35860
5	5	6250425958	Hobby	Torbalı	İzmir,35860
6	6	3730424823	Kavlak Kozmetik	Ayvalık	Balıkesir,10400
7	7	6312525334	Lotus	Torbalı	İzmir,35860
8	8	1333782821	Mif Kozmetik	Torbalı	İzmir,35860
9	9	7137114628	Prolinex Kimya San. Ltd. Şti.	Torbalı	İzmir,35860
10	10	3627697305	Proxy Kozmetik	Torbalı	İzmir,35860
11	11	2905378954	Sera Kozmetik	Torbalı	İzmir,35860
12	12	1902564879	Since Kozmetik	Torbalı	İzmir,35860
13	13	7384920463	Parma Kimya San.	Maltepe	Maltepe,34843
14	14	7493756201	Protek Chemical	Tuzla	Tuzla,34940

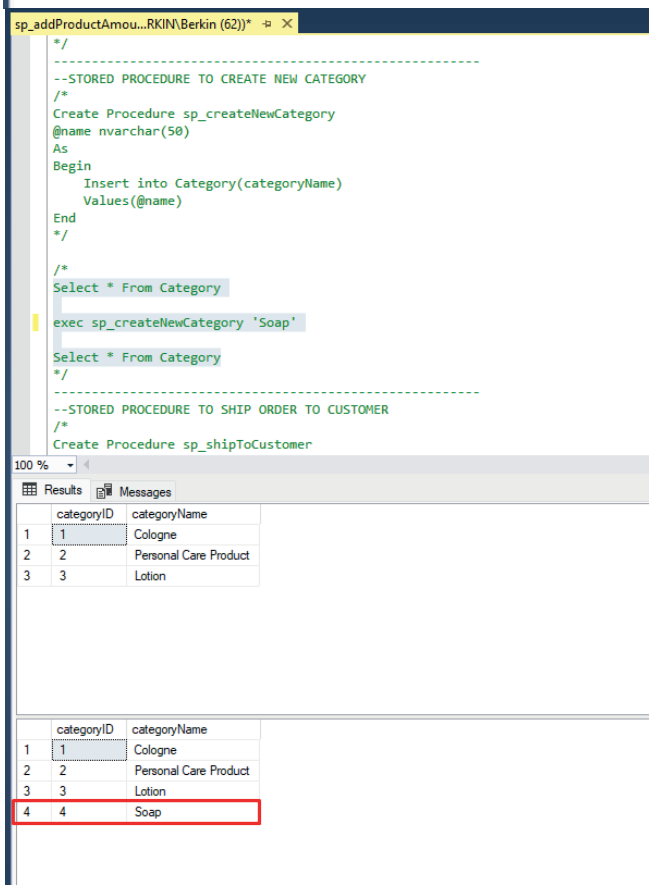
Stored Procedure-12: *sp_createNewCategory (categoryName)*

Definition: Adds a new category to the system.

How it works: Works by inserting given category name into category table.

Screenshots:



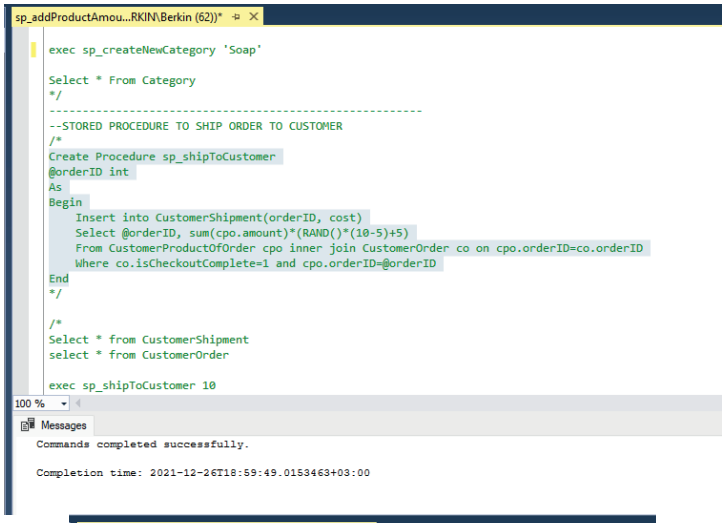


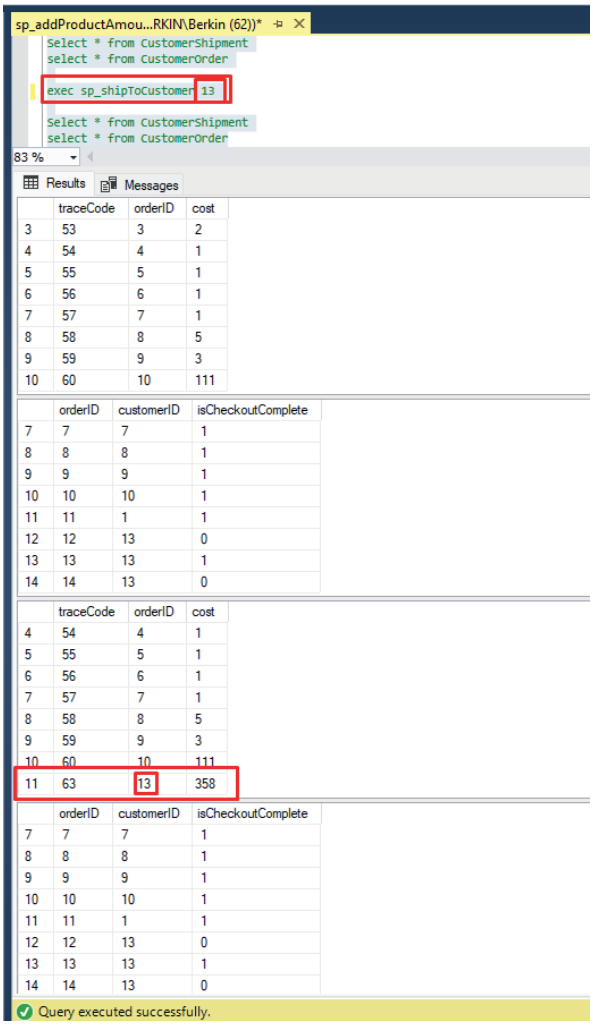
Stored Procedure-13: *sp_shipToCustomer (orderID)*

Definition: Ships the billed-order to the customer. Calculates a shipment cost.

How it works: Works by inserting a new row into shipment table. SP links the given orderID to shipment details. Calculates a random cost depending on the total amount of the items.

Screenshots:





Stored Procedure-14: *sp_shipToCompany (orderID)*

Definition: Ships the billed-order to the company. Calculates a shipment cost.

How it works: Works by inserting a new row into shipment table. SP links the given orderID to shipment details. Calculates a random cost depending on the total amount of the items.

Screenshots: