



MARMARA UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

CSE3035

DATABASE SYSTEMS

PROJECT STEP-3

-DETAILED REPORT-

Ahsen Yağmur Kahyaoğlu – 150119788

Alperen Kağan Kara – 150119608

Fatih Satı – 150119625

Mehmet Selman Baysan - 150120841

Project Description

In this project, we are created a database that handles properties of Cleaning Product Supplier System. Information about orders, products, depots, employees, suppliers, payments are recorded in an organized way. Thanks to this system, managers of Cleaning Product Supplier will be able to process payments, order deliveries and other operations.

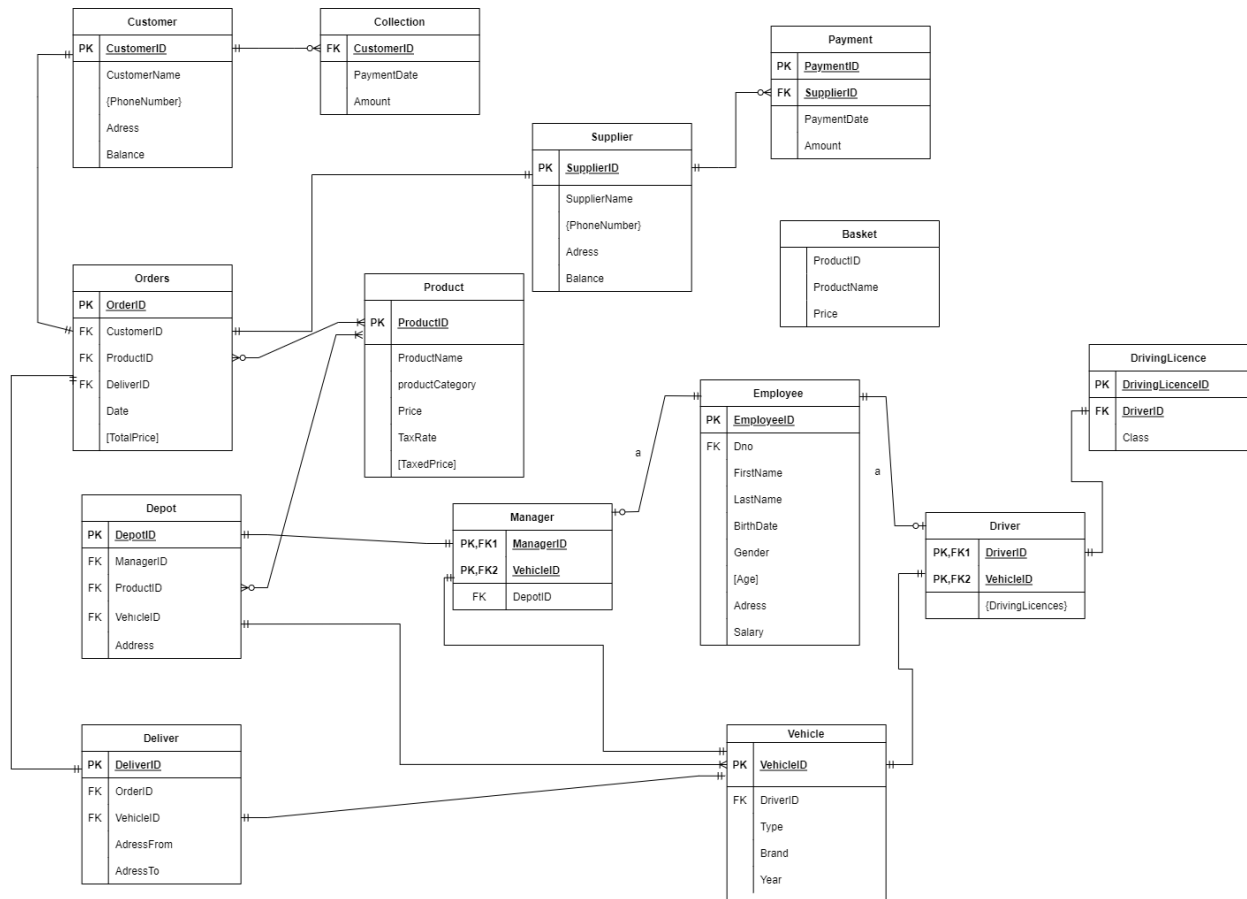
Scope

We've generated a number of entities relating to Customer Supplier System as part of our database system. To summarize, when an order occurs from our customers, we are creating order consist of products. Payment making operations will be done according to amount of price of products. After that, orders will be delivered to customers' addresses from our depots.

Data And Requirements Analysis for The Database and Business Processes

The database system that requested from us develop a "Customer Supplier System" that is reliable, scalable, adaptable, and effective. We established various restrictions, relationships, and stored procedures to fulfill their demands. In this document we will provide a more detailed explanation of these distinct components.

Diagram of Database



Tables

In our database there are tables called “Basket”, “Collection”, “Customer”, “Deliver”, “Depot”, “Driver”, “DrivingLicence”, “Employee”, “Manager”, “Orders”, “Payment”, “Product”, “Supplier”, “Vehicle”. In this part definition of tables will be stated.

Basket Table

ProductID	ProductName	Price
int	varchar(50)	decimal(18,2)

Collection Table

CollectionID	PaymentDate	Amount	CustomerID
int	date	decimal(18,2)	int

Collection table stores information about the amount of payment that collected from the customers. Primary key of the Collection table is CollectionID. Foreign key is CustomerID.

Customer Table

CustomerID	CustomerName	Address	Balance	PhoneNumber
int	nvarchar(50)	nvarchar(50)	decimal(18,2)	varchar(15)

Customer table stores information about the customers of our business. Primary key is CustomerID.

Deliver Table

DeliverID	OrderID	VehicleID	AddressFrom	AddressTo
int	int	int	nvarchar(50)	nvarchar(50)

Deliver table stores information about the deliver of the orders that our customers have requested. DeliverID is primary key of the table. VehicleID is the foreign key of the table. Delivers are carried by vehicles.

Depot Table

DepotID	ManagerID	ProductID	VehicleID	Address
int	int	int	int	nvarchar(50)

Depot table stores information about the depot that stocks the products. Primary key of table is DepotID. Foreign keys of the table are ManagerID, ProductID and VehicleID.

Driver Table

DriverID	VehicleID
int	Int

Driver table stores information about the drivers who delivers the orders to the customers. Drivers are also an employee in our system. Primary key of our table is DriverID. Foreign key of the table is VehicleID.

DrivingLicence Table

DrivingLicenceID	Class	DriverID
int	char(1)	Int

DrivingLicence table stores information about the driving licence of drivers. DrivingLicenceID is primary key of the table. DriverID is foreign key .

Employee Table

EmployeeID	Dno	FirstName	LastName	BirthDate	Gender	Age	Address	Salary
int	int	nvarchar(50)	nvarchar(50)	date	char(1)	tinyint	nvarchar(50)	decimal(18,2)

Employee table stores information about the employees of our system. EmployeeID is primary key of the table.

Manager Table

ManagerID	VehicleID	DepotID
int	int	Int

Managaer table stores information about the manager of our system. Managers are also an employee. Primary key is ManagerID. VehicleID and DepotID are foreign keys.

Orders Table

OrderID	CustomerID	ProductID	DeliverID	Date	DepotID	TotalPrice
int	int	int	int	date	tinyint	decimal(18,2)

Order table stores information about orders that customers requested from our business. Orders consist of multiple products. OrderID is primary key. CustomerID, ProductID, DeliverID, DepotID are foreign keys.

Payment Table

PaymentID	SupplierID	PaymentDate	Amount
int	int	date	decimal(18,2)

Payment table stores information about amount of payment that need to be paid by customers. Primary key is PaymentID. Foreign key is SupplierID.

Product Table

ProductID	ProductName	ProductCategory	Price	TaxRate	TaxedPrice
Int	varchar(50)	varchar(50)	decimal(18,2)	tinyint	

Product table stores information about products of our system. Primary key is ProductID. TaxedPrice is a computed column.

Supplier Table

SupplierID	SupplierName	PhoneNumber	Address	Balance
int	nvarchar(50)	varchar(15)	nvarchar(50)	decimal(18, 2)

Supplier table stores information about suppliers of our business. SupplierID is primary key.

Vehicle Table

VehicleID	DriverID	Type	Brand	Year
int	int	nvarchar(25)	nvarchar(25)	smallint

Vehicle table stores information about vehicles of our business. VehicleID is primary key. DriverID is foreign key.

Views

In our database systems, there are 5 views that are called “Ambalaj Products That Have Ordered Less Than 3”, “Customers Need To Pay Because They Didn't do Payment Of 2 Orders Past”, “Customers That Have Order But No Payment”, “Drivers That Older Than 35 and Having Less Than 3 Licence” and “Managers That Their Car Model Less Than 2018”.

Ambalaj Products That Have Ordered Less Than 3

It shows orders that are contains less than 3 products.

```
CREATE VIEW [Ambalaj Products That Have Ordered Less Than 3]
AS
SELECT p.ProductID, OrderProductCount.Total
FROM      dbo.Product AS p INNER JOIN
          (SELECT p1.ProductID, COUNT(*) AS Total
           FROM      dbo.Product AS p1 INNER JOIN
                     dbo.Orders AS o ON p1.ProductID = o.ProductID
           GROUP BY p1.ProductID) AS OrderProductCount ON p.ProductID = OrderProductCount.ProductID
WHERE (p.ProductCategory LIKE '%Ambalaj') AND (OrderProductCount.Total < 3)
```

	ProductID	Total
1	111	2
2	113	2
3	114	2

Customers Need To Pay Because They Didn't do Payment Of 2 Orders Past

It shows customers who need to pay past 2 unpaid orders.

```
CREATE VIEW [Customers Need To Pay Because They Didn't do Payment Of 2 Orders Past]
AS
SELECT numoforder.CustomerID
FROM      (SELECT CustomerID, COUNT(*) AS num
           FROM      (SELECT CustomerID, OrderID
                     FROM      dbo.Orders AS o
                     GROUP BY CustomerID, OrderID) AS custworder
           GROUP BY CustomerID) AS numoforder INNER JOIN
          (SELECT CustomerID, COUNT(*) AS num
           FROM      dbo.Collection AS c
           GROUP BY CustomerID) AS numofcol ON numoforder.CustomerID = numofcol.CustomerID
WHERE (numoforder.num - numofcol.num > 2)
```

Customers That Have Order But No Payment

It shows the customers who did not pay for what they order yet.

```
CREATE VIEW [Customers That Have Order But No Payment]
AS
SELECT DISTINCT CustomerID
FROM      (SELECT DISTINCT CustomerID
           FROM      dbo.Orders AS o
           WHERE      (CustomerID LIKE '1%')) AS customerwithorder
WHERE      (CustomerID NOT IN (SELECT CustomerID FROM dbo.Collection))
```

	CustomerID
1	1018
2	1022
3	1024

Drivers That Older Than 35 and Having Less Than 3 Licence

It shows drivers who are older than 35 and have less than 3 licenses.

```
CREATE VIEW [Drivers That Older Than 35 and Having Less Than 3 Licence]
AS
SELECT d.DriverID
FROM dbo.Driver AS d INNER JOIN
      dbo.Employee AS e ON d.DriverID = e.EmployeeID INNER JOIN
      (SELECT DriverID, COUNT(*) AS TotalLicence
       FROM      dbo.DrivingLicence AS dl
       GROUP BY DriverID) AS LicenceCount ON d.DriverID = LicenceCount.DriverID
WHERE      (e.Age > 35) AND (LicenceCount.TotalLicence < 3)
```

	DriverID
1	100025

Managers That Their Car Model Less Than 2018

It shows vehicles of managers which have model less than 2018.

```
CREATE VIEW [Managers That Their Car Model Less Than 2018]
AS
SELECT m.ManagerID, v.VehicleID, v.Year
FROM      dbo.Manager AS m INNER JOIN
           dbo.Vehicle AS v ON m.VehicleID = v.VehicleID
WHERE      (v.Year < 2018)
```

	ManagerID	VehicleID	Year
1	100022	2	2016
2	100024	4	2017

Trigger

In our database system, there is one trigger which is called “trg_updatedelivery”.

trg_updatedelivery

Every time an order been added, it creates a delivery for that order.

```
create trigger trg_updatedelivery
on Orders
after insert
as
    if (select i.OrderID from inserted i) not in (select d.OrderID from Deliver d)
    begin
        declare @orderid int
        set @orderid = (select i.OrderID from inserted i)

        insert into Deliver(DeliverID,OrderID)
        values((select MAX(DeliverID)+1 from Deliver),
        (select i.orderID from inserted i))

        exec sp_UpdateAddress @orderid
    end
```

Stored Procedures

In our database system, there are 13 stored procedures that are called “sp_AddNewProduct”, “sp_AddEmployee”, “sp_UpdateTotalPrice”, “sp_AddOrder”, “sp_DeleteCustomer”, “sp_AddnewCustomer”, “sp_DeleteOrder”, “sp_SelectOrder”, “sp_AddSupplier”, “sp_DeleteSupplier”, “sp_MakeCollection”, “sp_UpdateAddress”, “sp_addpayment”.

sp_AddNewProduct

It adds new product according to given parameters.

```
create procedure sp_AddNewProduct (  
@ProductName varchar(15),  
@ProductCategory varchar(15),  
@Price decimal(18,2))  
  
as  
begin  
    insert into Product  
        (ProductID,  
        ProductName,  
        ProductCategory,  
        Price)  
  
    values ((select (max(productID)+1)  
            from Product),  
            @ProductName,  
            @ProductCategory,  
            @Price)  
  
end  
exec sp_AddNewProduct 'Kokulu Sabun', 'Sabun', 29.50
```

	ProductID	ProductName	ProductCategory	Price	TaxRate	TaxedPrice
15	114	Plastik Kaşık	Plastik Ambalaj	10.00	18	11.80
16	115	Renkli Peçete	Peçete	7.50	18	8.85
17	116	Ace Çamaşır Suyu	Çamaşır Suyu	40.00	18	47.20
18	117	Duru El Sabunu	Sabun	28.00	18	33.04
19	118	Babysoft Yumuşatıcı	Yumuşatıcı	45.00	18	53.10
20	119	Tex Yüzey Temizleyici	Genel Temizlik	25.00	18	29.50
21	120	Kokulu Sabun	Sabun	29.50	18	34.81

sp_AddEmployee

It adds new employee according to given parameters. Age derived from birthdate. EmployeeID increases according to the biggest EmployeeID.

```
CREATE PROCEDURE sp_AddEmployee (  
    @Dno int,  
    @FirstName nvarchar(50),  
    @LastName nvarchar(50),  
    @BirthDate date,  
    @Gender char(1),  
    @Adress nvarchar(50),  
    @Salary decimal(18,2))  
  
as  
begin  
    insert into Employee (  
        EmployeeID,  
        Dno,  
        FirstName,  
        LastName,  
        BirthDate,  
        Gender,  
        Age,  
        Adress,  
        Salary)  
  
    values (  
        (select (max(EmployeeID)+1)from Employee),  
        @Dno,  
        @FirstName,  
        @LastName,  
        @BirthDate,  
        @Gender,  
        DATEDIFF(YY, @Birthdate, GETDATE()),  
        @Adress,  
        @Salary)  
  
end  
exec sp_AddEmployee 9, 'Ahmet', 'Aksoy', '1999-07-17', 'M', 'Pendik', 5600.00
```

30	100029	6	Faruk Doğukan	Öreroğlu	1992-12-02	M	29	2.ORGANİZE SANAYİ BÖLGESİ KIRIM CD SARIÇİÇE...	8450.00
31	100030	7	Mustafa Selman	Öyüş	1993-06-08	M	28	YUZBAŞILAR C KUMALI APT ALTI N 4 DEĞİRMEND...	9000.00
32	100031	6	Fatih	Uçar	1989-05-29	M	32	necip fazıl mahallesi aşçıbaşı mektebi sokak no:18	10000.00
33	100032	7	Berfin Elif	Özay	1969-04-06	F	52	GÖLGELİ S N 37/16, Gaziosmanpaşa	16000.00
34	100033	6	Işılray	Özbey	1977-04-26	F	44	Saraçlar San.Sit.9.Blok No:689 İkitelli	12000.00
35	100034	7	Murathan	Paksoy	1990-11-16	M	31	5 TELSİZ MAH. ZEYTİNBURNU	8700.00
36	100035	9	Ahmet	Aksoy	1999-07-17	M	22	Pendik	5600.00

sp_UpdateTotalPrice

It updates total price of an order by adding price of products in that order.

```
CREATE PROCEDURE sp_UpdateTotalPrice (  
    @OrderID int)  
AS  
BEGIN  
    UPDATE ORDERS  
    SET TotalPrice=calculations.TotalPrice  
    FROM Orders o inner join (SELECT o.OrderID, sum(p.TaxedPrice) as TotalPrice FROM Orders o INNER JOIN Product p ON o.ProductID=p.ProductID  
    Group by o.OrderID) as calculations on o.OrderID=calculations.OrderID  
    where o.OrderID=@OrderID;  
END  
  
exec sp_UpdateTotalPrice 10009
```

	OrderID	CustomerID	ProductID	DeliverID	Date	DepotID	TotalPrice
1	10008	5010	101	1000009	2021-12-12	6	153.40
2	10009	1018	108	1000010	2021-12-13	6	94.40
3	10009	1018	110	1000010	2021-12-13	6	94.40
4	10009	1018	113	1000010	2021-12-13	6	94.40
5	10009	1018	114	1000010	2021-12-13	6	94.40

sp_AddOrder

It adds new order according to given parameters. Also, the price of new order will be added to customers' taxed price and delivery properties will be created.

```
create procedure sp_AddOrder(
@OrderID int,
@CustomerID int,
@ProductID int,
@DepotID tinyint)

as
begin
    insert into Orders
        (OrderID,
        CustomerID,
        ProductID,
        DeliverID,
        orders.Date,
        DepotID)

        values (@OrderID,
        @CustomerID,
        @ProductID,
        (select (max(deliverID)+1)
        from deliver),
        GETDATE(),
        @DepotID)
    exec sp_UpdateTotalPrice @OrderID=@OrderID
end
begin
    update Customer
    set Balance=Balance+pprice.TaxedPrice
    from (select p.productID, p.TaxedPrice
    from Product p
    where p.productID=@ProductID) pprice
    where CustomerID=@CustomerID
end

exec sp_AddOrder 10020, 1018, 110, 4
```

	OrderID	CustomerID	ProductID	DeliverID	Date	DepotID	TotalPrice
1	10020	1018	110	1000015	2021-12-26	4	29.50

sp_AddnewCustomer

It adds new customer according to given parameters. CustomerID increases according to last customer's CustomerID.

```
create procedure sp_AddnewCustomer (@CustomerName varchar(50),
                                     @Address varchar(50),
                                     @Phonenumber varchar(15))
as
begin
    insert into Customer(CustomerID,
                        CustomerName,
                        Address,
                        Balance,
                        PhoneNumber)

    values ((select (MAX(CustomerID)+1)
            from Customer),
            @customerName,
            @Address,
            0,
            @Phonenumber)

end

exec sp_AddnewCustomer 'Moda Avm', 'Kartal', 2164568312
```

23	1023	AYC Oyuncak	ANAFARTALAR BULVARI NO: 734, Antalya	0.45	2423331122
24	1024	Kemer Otel	K.KARABEKİR Cad. No:28, Antalya	177.00	2426452368
25	1025	Moda Avm	Kartal	0.00	2164568312

sp_DeleteCustomer

It deletes the customer by taking CustomerID parameter.

```
create procedure sp_DeleteCustomer (@CustomerID int)
as
begin
    delete
    from Customer
    where CustomerID=@CustomerID

end

exec sp_DeleteCustomer 1025
```

%

Messages

(1 row affected)

21	1021	Ametal Asansör	Y.YALOVA YOLU BUTTİM İŞ MERKEZİ A.BLOK 2.K ...	0.00	2242110443
22	1022	Aydın Torna	CUMHURİYET M ATATÜRK BULVARI N 69, Aydın	115.05	2562142849
23	1023	AYC Oyuncak	ANAFARTALAR BULVARI NO: 734, Antalya	0.45	2423331122
24	1024	Kemer Otel	K.KARABEKİR Cad. No:28, Antalya	177.00	2426452368

sp_DeleteOrder

It deletes order according to OrderID parameter. It will be deleted from customers' orders as well.

```
create procedure sp_DeleteOrder (@OrderID int)
as
begin
    update Customer
    set Balance=c.Balance-price.TotalPrice
    from (select OrderID, CustomerID, TotalPrice
          from Orders) price inner join Customer c on price.CustomerID=c.CustomerID
    End
    Begin
        Delete
        from Orders
        where OrderID=@OrderID
    End
    exec sp_DeleteOrder 10020
```

	OrderID	CustomerID	ProductID	DeliverID	Date	DepotID	TotalPrice
1	10008	5010	101	1000009	2021-12-12	6	153.40
2	10009	1018	108	1000010	2021-12-13	6	94.40
3	10009	1018	110	1000010	2021-12-13	6	94.40

sp_SelectOrder

It shows order information according to selected OrderID.

```
create procedure sp_SelectOrder (@OrderID int)
as
begin
    select o.OrderID, o.CustomerID, o.ProductID, o.Date, o.DepotID
    from orders o
    where OrderID=@OrderID
    group by OrderID, o.CustomerID, o.ProductID, o.Date, o.DepotID
    End
    exec sp_SelectOrder 10007
```

83 %

Results Messages

	OrderID	CustomerID	ProductID	Date	DepotID
1	10007	5008	104	2021-12-11	2
2	10007	5008	105	2021-12-11	2
3	10007	5008	107	2021-12-11	2
4	10007	5008	110	2021-12-11	2
5	10007	5008	113	2021-12-11	2

sp_AddSupplier

It adds new supplier according to given parameters.

```
create procedure sp_AddSupplier (@SupplierID int,
                                @SupplierName nvarchar(50),
                                @PhoneNumber varchar(15),
                                @Adress nvarchar(50))
as
Begin
    insert into Supplier(SupplierID,
                        SupplierName,
                        PhoneNumber,
                        Adress,
                        Balance)

    values(@SupplierID,
           @SupplierName,
           @PhoneNumber,
           @Adress,
           0)

End

exec sp_AddSupplier 5015, 'Selen Temizlik', 2167864532, 'Maltepe'
```

ID	SupplierID	SupplierName	PhoneNumber	Adress	Balance
14	5014	Başkent Toptan	3120193810	EBU ZİYA TEVFIK S N 3/2, Ankara	0.00
15	5015	Selen Temizlik	2167864532	Maltepe	0.00

sp_DeleteSupplier

It deletes a supplier according to SupplierID parameter.

```
create procedure sp_DeleteSupplier (@SupplierID int)
as
Begin
    DELETE
    FROM Supplier
    WHERE SupplierID=@SupplierID
END

exec sp_DeleteSupplier 5015
```

ID	SupplierID	SupplierName	PhoneNumber	Adress	Balance
13	5013	Otrivant Ltd. Şti	2569974439	Nazili Aydın, Çiçek sokak No:7 Aydın	0.00
14	5014	Başkent Toptan	3120193810	EBU ZİYA TEVFIK S N 3/2, Ankara	0.00

sp_MakeCollection

It makes collection from customers for their orders. Procedure shows the customers that make payment and amount of the payment.

```
create procedure sp_MakeCollection (@CustomerID int,@Amount decimal(18,2))
as

    if @CustomerID in (select CustomerID
                        from Collection)
    begin
        declare @collectionID int
        select @collectionID = (MAX(collectionID)+1)
                        from collection
                        where CustomerID=@CustomerID

        insert into Collection(CustomerID,PaymentDate,Amount,CollectionID)
        values(@CustomerID,getdate(),@Amount,@collectionID)

        update Customer
        set Balance=balance-@Amount
        where CustomerID=@CustomerID
    end
    else
    begin
        declare @collectionIDvarchar varchar(10)
        select @collectionIDvarchar = (cast(@customerID as varchar(5)) + CAST(1 as varchar(1)))
        insert into Collection (CustomerID,PaymentDate,Amount,CollectionID)
        values(@CustomerID,GETDATE(),@Amount,@collectionIDvarchar)

        update Customer
        set Balance=balance-@Amount
        where CustomerID=@CustomerID
    end
end

exec sp_MakeCollection 1012, 150.00
```

	CustomerID	PaymentDate	Amount	CollectionID
1	1004	2021-12-09	100.00	10041
2	1004	2021-12-22	200.00	10042
3	1010	2021-12-22	50.00	10101
4	1010	2021-12-22	45.00	10102
5	1012	2021-12-26	150.00	10121

sp_UpdateAddress

This procedure used in trigger that we created for making new delivery of an order. In this procedure, address to and address from parts will be updated according to depot and customer addresses.

```
create procedure sp_UpdateAddress
    @orderID int
As
Begin
    update Deliver
    set VehicleID=vhcl.VehicleID
    from(select dpt.DepotID,dpt.VehicleID,o.DeliverID,o.OrderID
    from Orders o inner join Depot dpt on o.DepotID=dpt.DepotID
    group by dpt.DepotID,dpt.VehicleID,o.DeliverID,o.OrderID) vhcl
    where Deliver.DeliverID=vhcl.DeliverID and @orderID=vhcl.OrderID

    update Deliver
    set AddressFrom=adrs.Address
    from (select distinct(dpt.Address), o.OrderID
    from Orders o inner join Depot dpt on o.DepotID=dpt.DepotID
    where o.OrderID=@orderID) adrs
    where Deliver.OrderID=adrs.OrderID

    Update d
    Set d.AddressTo=c.Address
    From Deliver d inner join Orders ord on d.OrderID=ord.OrderID
        inner join Customer c on ord.CustomerID=c.CustomerID
    Where ord.CustomerID like '1%' and @orderID=d.OrderID

    Update d
    Set d.AddressTo=s.Address
    From Deliver d inner join Orders ord on d.OrderID=ord.OrderID
        inner join Supplier s on ord.CustomerID=s.SupplierID
    Where ord.CustomerID like '5%' and @orderID=d.OrderID
End

exec sp_UpdateAddress 10006
```

sp_addpayment

It makes the payment to supplier according to SupplierID and amount of money. It creates new payment process to payment table and Id of process increases according to last payment.

```
create procedure sp_addpayment (@supplierID int, @amount decimal(18,2))
as
    if @supplierID in (select SupplierID
                      from Payment)
    begin
        declare @paymentID int
        select @paymentID = (MAX(PaymentID)+1)
                      from Payment
                      where SupplierID=@supplierID

        insert into Payment(SupplierID,PaymentDate,Amount,PaymentID)
        values(@supplierID,getdate(),@Amount,@paymentID)

        update Supplier
        set Balance=balance-@Amount
        where SupplierID=@supplierID
    end
    else
    begin
        declare @paymentIDvchar varchar(10)
        select @paymentIDvchar= (cast(@supplierID as varchar(5)) + CAST(1 as varchar(1)))

        insert into Payment(SupplierID,PaymentDate,Amount,PaymentID)
        values(@supplierID,getdate(),@Amount,@paymentIDvchar)

        update Supplier
        set Balance=balance-@Amount
        where SupplierID=@supplierID
    end

end

exec sp_addpayment 5009, 2500.00
```

	PaymentID	SupplierID	PaymentDate	Amount
1	50021	5002	2021-12-10	150.00
2	50051	5005	2021-12-22	50.00
3	50052	5005	2021-12-22	25.00
4	50081	5008	2021-12-11	75.00
5	50082	5008	2021-12-22	50.00
6	50091	5009	2021-12-26	2500.00
7	50101	5010	2021-12-15	100.00

Last Added Stored Procedures

In addition to our previous stored procedures, we created new ones to improve our web interface. They are called “sp_AddItemToBasket”, “sp_AllCustomerView”, “sp_AllOrderView”, “sp_CountOrderNumber”, “sp_DeleteBasketItems”, “sp_DeleteProduct”, “sp_SearchOrderByCustomerID”, “sp_UpdateCustomer”, “sp_UpdateProduct”, “sp_UpdateSupplier”, “sp_AllOrderView”.

sp_AddItemToBasket

It adds product to basket according to given parameter.

```
ALTER procedure [dbo].[sp_AddItemToBasket] @ProductID int, @ProductName varchar(50), @Price decimal(18,2)
as
begin
    insert into Basket ("ProductID","ProductName","Price")
    values (@ProductID,@ProductName,@Price)
end

exec sp_AddItemToBasket 100, 'Sedef Foam', 41.30
```

	ProductID	ProductName	Price
1	100	Sedef Foam	41.30

sp_AllCustomerView

It shows all customers.

```
ALTER procedure [dbo].[sp_AllCustomerView]
as
begin
    select *
    from Customer
end

exec sp_AllCustomerView
```

100 %						
		Results Messages				
	CustomerID	CustomerName	Adress	Balance	PhoneNumber	
1	1001	Akdeniz Temizlik	Necip fazıl Mah. İlim Sokak. No:32, Çekmeköy	0.00	2164121122	
2	1002	Ozkan Makina	Şemsi Paşa Mah. Burhanettin sok. No:2, Ümraniye	0.00	2167851235	
3	1003	Dört Mevsim Manav	Bağdat Caddesi No:52, Kadıköy	0.00	2163452315	
4	1004	Tekim Temizlik	Atatürk Mahallesi, Çavuşbaşı Caddesi No:12, Ümrani	62.85	2161452365	
5	1005	Bireysel Akademi	Madenler Mah. Cicek Sokak No:33, Çekmeköy	0.00	2163562684	
6	1006	Prizma	Alemdağ Caddesi, No: 98, Çekmeköy	0.00	2169846521	
7	1007	Nebioglu Temizlik	Bankalar Cad. Yanıkkapı, Porsuk Sok.No.6/1 80020 K	0.00	2124563377	
8	1008	ARD Asansör	Gebze Sanayi Sitesi A blok No:198,Gebze	0.00	2624356018	
9	1009	Penta A.Ş	SULTAN ORHAN MAH. 1176/2 SOK. No:107, Gebze	0.00	2625557816	
10	1010	Macfit	Çakmak Mah. Samanyolu Cad. No:26; Ümraniye	2.35	2164569874	
11	1011	Arı Temizlik	Beylikdüzü, Birlik Sanayi Sitesi 2. Cad. No: 89, B	0.00	2128751789	
12	1012	Fen Bilimleri VIP	Ambarlı Mahallesi, Bayıldım Caddesi No:3, Avcılar	0.00	2123478546	
13	1013	Işık Plastik	Cihangir Mahallesi, Çatık Kaş Sokağı No:12/A, Avcı	0.00	2123547896	
14	1014	Nafi Yılmaz Diş Poliklinik	Çınar Mahallesi, Gül Sokak No:25, Bahçeşehir	0.00	2125647893	
15	1015	Fatih Temizlik ve Ambalaj	MODERN İŞHANI KAT 3 N 421, Ulus, Ankara	173.69	3123110477	
16	1016	Faruk Ambalaj	YENİ ZİRAAT MAHALLESİ N 24/B, Dışkapı, Ankara	0.00	3122058719	
17	1017	Başkent Manav	REŞİT GALİP C N 128/1, Gaziosmanpaşa, Ankara	0.00	3124473091	
18	1018	Bostancı Ambalaj	Bostancı Sanayi Sitesi, Huzurhoca sok. No.44 İçere	94.40	2168764523	
19	1019	Esenyurt Yenilenebilir E...	EVREN OTO SANAYİ SİTESİ 2 KISIM 28/A BLOK N...	0.00	2123339981	
20	1020	Turex Taşımacılık	ORG.SAN.BÖL.YEŞİL C N 31, Bursa	158.05	2247891130	
21	1021	Ametal Asansör	Y.YALOVA YOLU BUTTİM İŞ MERKEZİ A BLOK 2.K ...	0.00	2242110443	
22	1022	Aydın Torna	CUMHURİYET M ATATÜRK BULVARI N 69, Aydın	115.05	2562142849	
23	1023	AYC Oyuncak	ANAFARTALAR BULVARI NO: 734, Antalya	0.45	2423331122	
24	1024	Kemer Otel	K.KARABEKİR Cad. No:28, Antalya	177.00	2426452368	
25	1025	Fatih	Ev	159.30	7878787	
26	1026	ali	Pendik	0.00	12324341	

sp_AllOrderView

It shows all orders.

```
ALTER procedure [dbo].[sp_AllOrderView]
as
begin
    select *
    from Orders
end
```

exec sp_AllOrderView

100 %

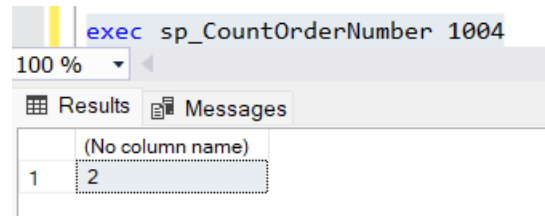
Results Messages

	OrderID	CustomerID	ProductID	DeliverID	Date	DepotID	TotalPrice
1	10016	1025	107	1000019	2022-01-05	1	112.10
2	10008	5010	101	1000009	2021-12-12	6	153.40
3	10009	1018	108	1000010	2021-12-13	6	94.40
4	10009	1018	110	1000010	2021-12-13	6	94.40
5	10009	1018	113	1000010	2021-12-13	6	94.40
6	10009	1018	114	1000010	2021-12-13	6	94.40
7	10010	5005	111	1000011	2021-12-14	6	132.75
8	10010	5005	112	1000011	2021-12-14	6	132.75
9	10010	5005	107	1000011	2021-12-14	6	132.75
10	10010	5005	108	1000011	2021-12-14	6	132.75
11	10010	5005	110	1000011	2021-12-14	6	132.75
12	10011	1015	117	1000012	2021-12-14	7	230.69
13	10011	1015	108	1000012	2021-12-14	7	230.69
14	10011	1015	109	1000012	2021-12-14	7	230.69
15	10011	1015	115	1000012	2021-12-14	7	230.69
16	10011	1015	116	1000012	2021-12-14	7	230.69
17	10011	1015	118	1000012	2021-12-14	7	230.69
18	10011	1015	119	1000012	2021-12-14	7	230.69
19	10012	1010	102	1000013	2021-12-15	1	97.35
20	10012	1010	105	1000013	2021-12-15	1	97.35
21	10012	1010	110	1000013	2021-12-15	1	97.35
22	10013	1022	102	1000014	2021-12-15	1	115.05
23	10013	1022	105	1000014	2021-12-15	1	115.05
24	10013	1022	109	1000014	2021-12-15	1	115.05
25	10013	1022	110	1000014	2021-12-15	1	115.05
26	10016	1025	101	1000019	2022-01-05	1	112.10
27	10016	1025	100	1000019	2022-01-05	1	112.10
28	10000	1004	118	1000015	2021-12-22	4	259.60
29	10000	1004	101	1000015	2021-12-22	4	259.60
30	10000	1004	102	1000015	2021-12-22	4	259.60
31	10014	1025	101	1000019	2022-01-04	1	109.15
32	10014	1025	110	1000019	2022-01-04	1	109.15
33	10014	1025	110	1000019	2022-01-04	1	109.15

sp_CountOrderNumber

It counts the number of orders that customer made.

```
ALTER procedure [dbo].[sp_CountOrderNumber] @CustomerID int
as
begin
    select count(distinct(orderID)) from Orders where CustomerID=@CustomerID
end
```



exec sp_CountOrderNumber 1004

100 %


Results Messages

	(No column name)
1	2

sp_DeleteBasketItems

It empties the items in basket.

```
ALTER procedure [dbo].[sp_DeleteBasketItems]
as
begin
    delete Basket
end
```



exec sp_DeleteBasketItems

ProductID	ProductName	Price
-----------	-------------	-------

sp_DeleteProduct

It deletes product from Product Table via given ProductID.

```
ALTER procedure [dbo].[sp_DeleteProduct] @ProductID int
as
begin
    delete Product
    where ProductID=@ProductID
end
```

```
exec sp_DeleteProduct 100
```

	ProductID	ProductName	ProductCategory	Price	TaxRate	TaxedPrice
1	101	Sedef Foam	Sabun	35.00	18	41.30
2	102	Klino Flora	Genel Temizlik	30.00	18	35.40
3	103	Klino Bahar	Genel Temizlik	30.00	18	35.40
4	104	Prill Bulaşık Deterjanı	Bulaşık Deterjanı	25.00	18	29.50
5	105	Fairy Bulaşık Deterjanı	Bulaşık Deterjanı	27.50	18	32.45

sp_SearchOrderByCustomerID

It shows the orders of customer via given CustomerID.

```
ALTER procedure [dbo].[sp_SearchOrderByCustomerID] @CustomerID int
as
begin
    select *
    from Orders
    where CustomerID=@customerID
end
```

```
exec sp_SearchOrderByCustomerID 1004
```

100 %

Results Messages

	OrderID	CustomerID	ProductID	DeliverID	Date	DepotID	TotalPrice
1	10000	1004	118	1000015	2021-12-22	4	259.60
2	10000	1004	101	1000015	2021-12-22	4	259.60
3	10000	1004	102	1000015	2021-12-22	4	259.60
4	10000	1004	100	1000001	2021-12-05	4	259.60
5	10000	1004	105	1000001	2021-12-05	4	259.60
6	10000	1004	106	1000001	2021-12-05	4	259.60
7	10000	1004	115	1000001	2021-12-05	4	259.60
8	10003	1004	105	1000004	2021-12-08	5	103.25
9	10003	1004	108	1000004	2021-12-08	5	103.25
10	10003	1004	110	1000004	2021-12-08	5	103.25

sp_UpdateCustomer

It updates the selected customer's information.

```
ALTER procedure [dbo].[sp_UpdateCustomer] (@CustomerID int,
                                           @CustomerName nvarchar(50),
                                           @Adress nvarchar(50),
                                           @PhoneNumber varchar(15))
as
begin
    update customer
    set CustomerName=@CustomerName, Adress=@Adress, PhoneNumber=@PhoneNumber
    where CustomerID=@CustomerID
end
```

```
exec sp_UpdateCustomer 1026, 'Ali', 'Maltepe', '983560'
```

24	1024	Kemer Otel	K.KARABEKİR Cad. No:28, Antalya	177.00	2426452368
25	1025	Fatih	Ev	159.30	7878787
26	1026	Ali	Maltepe	0.00	983560

sp_UpdateProduct

It updates the selected product's information.

```
ALTER procedure [dbo].[sp_UpdateProduct]
@ProductID int,
@ProductName varchar(50),
@ProductCategory varchar(50),
@Price decimal(18,2)
as
begin
    update Product
    set ProductName=@ProductName, ProductCategory=@ProductCategory, Price=@Price
    where ProductID=@ProductID
end
```

```
exec sp_UpdateProduct 102, 'Klino Flora', 'Genel Temizlik', 45.90
```

	ProductID	ProductName	ProductCategory	Price	TaxRate	TaxedPrice
1	101	Sedef Foam	Sabun	35.00	18	41.30
2	102	Klino Flora	Genel Temizlik	45.90	18	54.16

sp_UpdateSupplier

It updates the selected supplier's information.

```
ALTER procedure [dbo].[sp_UpdateSupplier] (@SupplierID int,  
                                           @SupplierName nvarchar(50),  
                                           @Adress nvarchar(50),  
                                           @PhoneNumber varchar(15))  
  
as  
begin  
    update Supplier  
    set SupplierName=@SupplierName, Adress=@Adress, PhoneNumber=@PhoneNumber  
    where SupplierID=@SupplierID  
end
```

```
exec sp_UpdateSupplier 5002, 'Akın Temizlik', 'Ümraniye', 2163487960
```

	SupplierID	SupplierName	PhoneNumber	Adress
1	5001	Deniz Kimya	2165789463	İmes sanayi sitesi 102 sokak A blok no:4 Ümraniye
2	5002	Akın Temizlik	2163487960	Ümraniye