STEP 3 REPORT

a) Project description: explain what your database project is about.

We wanted to create Fritolay's database. There are 3 main employee kind. RegionalManager> SalesChief>Salesman. This sorting is about management. Also there are products. We have taken the list of products from the person we talked to (They are using oracle). Actually we wanted to do a database of stock exchange but it was really hard. So we kept it simple. Customers make a list of products that they want and Salesmans make a list of products that they want. There are also warehouses where the warehouse employees work.

b) Scope: what is included/exclude? Which processes are supported, which ones are not?

We have deleted some of the entities from the previous step. Because they were just a bunch of useless data that we cant to anything with them. Actually they would be useful in real life to keep track of everything simultaneously. It would take weeks to do the real project i guess. To sum up we have kept what we can use and we deleted what is useless for the project.

c) Tables

i) Name of the fields/columns.

ii) Definition of the table.

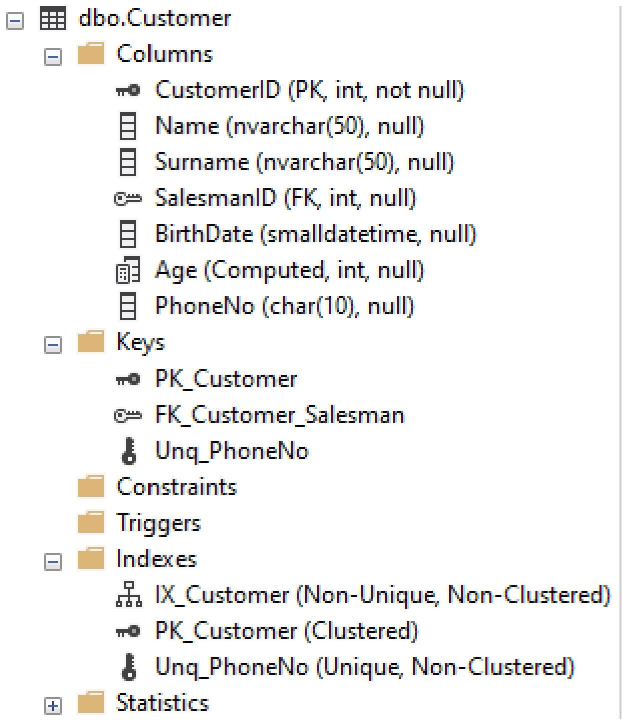
iii) Data types of the fields.

iv) Information about indexes, primary key, foreign key.

v) Information about uniques, identity, check constraints, defaults, computed columns, if any.

vi) Information about triggers, if any.

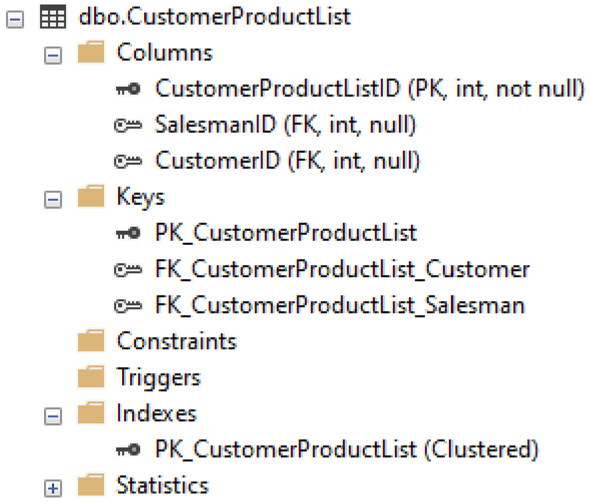
Customer Table



```
⊟ dbo.Customer
    ⊟ Columns
          CustomerID (PK, int, not null)
          Name (nvarchar(50), null)
          Surname (nvarchar(50), null)
          SalesmanID (FK, int, null)
          BirthDate (smalldatetime, null)
          Age (Computed, int, null)
          PhoneNo (char(10), null)
    ⊟ Keys
          PK_Customer
          FK_Customer_Salesman
          Unq_PhoneNo
      Constraints
      Triggers
    ⊟ Indexes
          IX_Customer (Non-Unique, Non-Clustered)
          PK_Customer (Clustered)
          Unq_PhoneNo (Unique, Non-Clustered)
    ⊞ Statistics
```

Customers have these attributes. They have also a salesman who is providing chips to them. Age column is computed from birthdate. Phone number is unique and salesmanId is foreign key here. We have also indexed surname ascending in this table. Primary key is identity it increase as you enter new data.

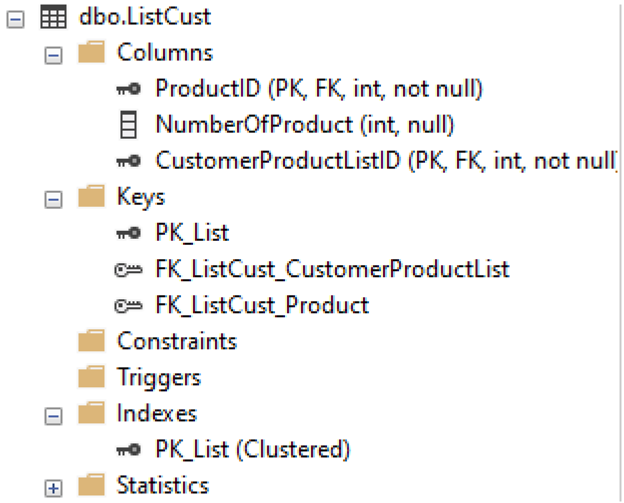
CustomerProductList Table



```
⊟ dbo.CustomerProductList
    ⊟ Columns
          CustomerProductListID (PK, int, not null)
          SalesmanID (FK, int, null)
          CustomerID (FK, int, null)
    ⊟ Keys
          PK_CustomerProductList
          FK_CustomerProductList_Customer
          FK_CustomerProductList_Salesman
      Constraints
      Triggers
    ⊟ Indexes
          PK_CustomerProductList (Clustered)
    ⊞ Statistics
```

This table is basically a ladder. It connects salesmanID and customerID to listCust table via customerProductListId. Nothing special here. We had to create this table in order to table associative entity (product list).

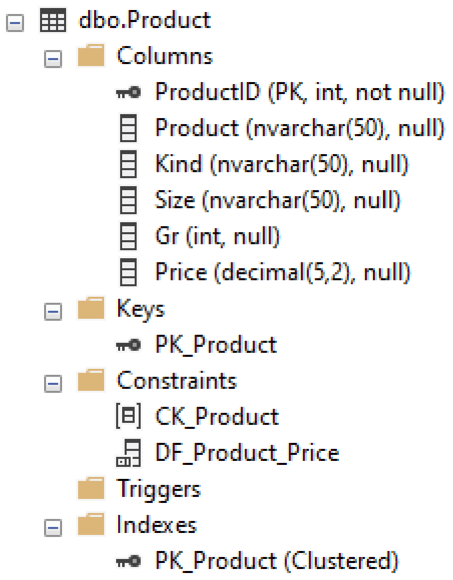
ListCust Table



```
⊟ dbo.ListCust
    ⊟ Columns
          ProductID (PK, FK, int, not null)
          NumberOfProduct (int, null)
          CustomerProductListID (PK, FK, int, not null
    ⊟ Keys
          PK_List
          FK_ListCust_CustomerProductList
          FK_ListCust_Product
      Constraints
      Triggers
    ⊟ Indexes
          PK_List (Clustered)
    ⊞ Statistics
```

This table keeps data of the products in a product list. There are 2 primary key(productID and customerProductListId) here because of associative entity. Also there are 2 foreign key one of them connects productID to product table and the other one connects customerProductListID to customerProductList table.

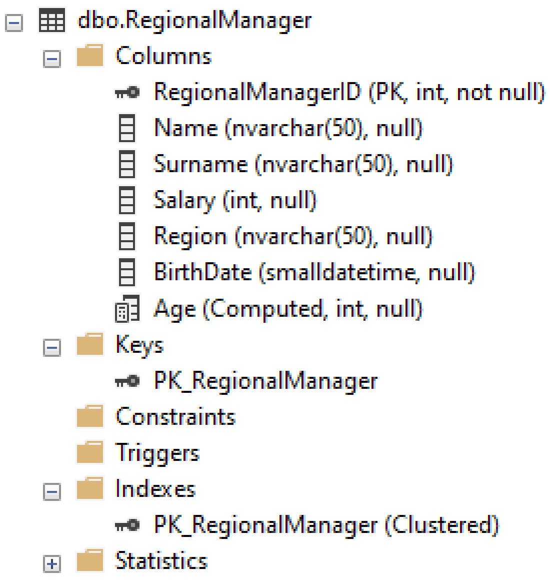
Product Table



```
⊟ dbo.Product
    ⊟ Columns
          ProductID (PK, int, not null)
          Product (nvarchar(50), null)
          Kind (nvarchar(50), null)
          Size (nvarchar(50), null)
          Gr (int, null)
          Price (decimal(5,2), null)
    ⊟ Keys
          PK_Product
    ⊟ Constraints
          CK_Product
          DF_Product_Price
      Triggers
    ⊟ Indexes
          PK_Product (Clustered)
```

In this table we keep data of products. Such as doritos(product) taco (kind) Süper (size) 107 (gr) 8 (price). We have taken all these from real person. There is also a constraint which is not so sensible but a products price can not be higher than 15 liras. Primary key is identity it increase as you enter new data.

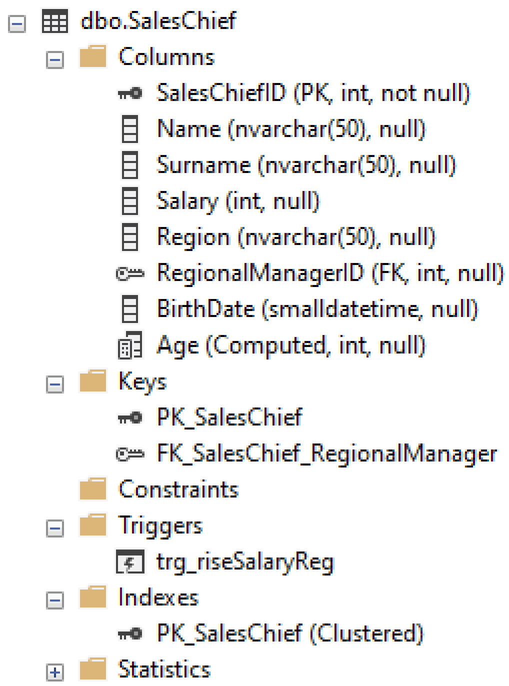
RegionalManager Table



```
⊟ dbo.RegionalManager
    ⊟ Columns
          RegionalManagerID (PK, int, not null)
          Name (nvarchar(50), null)
          Surname (nvarchar(50), null)
          Salary (int, null)
          Region (nvarchar(50), null)
          BirthDate (smalldatetime, null)
          Age (Computed, int, null)
    ⊟ Keys
          PK_RegionalManager
      Constraints
      Triggers
    ⊟ Indexes
          PK_RegionalManager (Clustered)
    ⊞ Statistics
```

Regional manager manages saleschief. There are some attributes above. Age is computed from birthdate. Primary key is identity it increase as you enter new data.

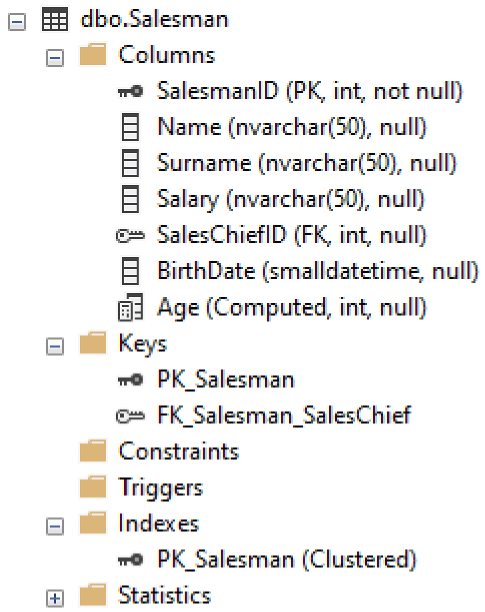
SalesChief Table



```
dbo.SalesChief
    Columns
        SalesChiefID (PK, int, not null)
        Name (nvarchar(50), null)
        Surname (nvarchar(50), null)
        Salary (int, null)
        Region (nvarchar(50), null)
        RegionalManagerID (FK, int, null)
        BirthDate (smalldatetime, null)
        Age (Computed, int, null)
    Keys
        PK_SalesChief
        FK_SalesChief_RegionalManager
    Constraints
    Triggers
        trg_riseSalaryReg
    Indexes
        PK_SalesChief (Clustered)
    Statistics
```

Sales chief is managed by regional manager and also sales chief manages salesmans. Age is computed. RegionalManagerID is foreign key which connects to regionalManager table. There is one trigger here which raises the salary of regionalManager after the insertion of saleschief. Primary key is identity it increase as you enter new data.

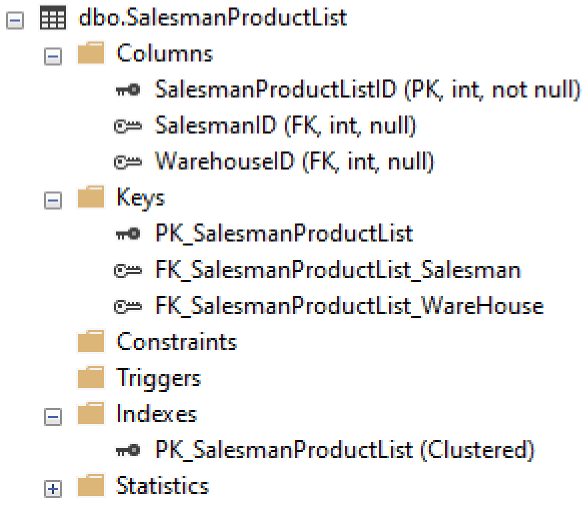
Salesman Table



```
dbo.Salesman
    Columns
        SalesmanID (PK, int, not null)
        Name (nvarchar(50), null)
        Surname (nvarchar(50), null)
        Salary (nvarchar(50), null)
        SalesChiefID (FK, int, null)
        BirthDate (smalldatetime, null)
        Age (Computed, int, null)
    Keys
        PK_Salesman
        FK_Salesman_SalesChief
    Constraints
    Triggers
    Indexes
        PK_Salesman (Clustered)
    Statistics
```

Salesman is managed by salesChief so there is foreign key about that. Primary key is identity it increase as you enter new data.

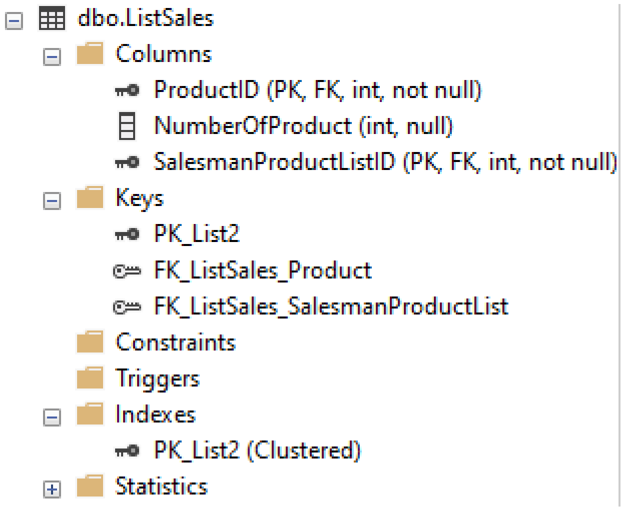
SalesmanProductList



```
⊟ dbo.SalesmanProductList
    ⊟ Columns
          SalesmanProductListID (PK, int, not null)
          SalesmanID (FK, int, null)
          WarehouseID (FK, int, null)
    ⊟ Keys
          PK_SalesmanProductList
          FK_SalesmanProductList_Salesman
          FK_SalesmanProductList_WareHouse
      Constraints
      Triggers
    ⊟ Indexes
          PK_SalesmanProductList (Clustered)
    ⊞ Statistics
```

This table is similar to CustomerProductList but we have warehouse here. Salesman gets the products from warehouse. There are two foreign key which connects to related tables. Primary key is identity it increase as you enter new data.

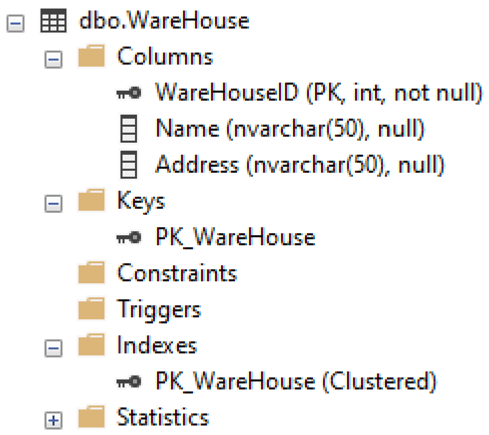
ListSales Table



```
⊟ dbo.ListSales
    ⊟ Columns
          ProductID (PK, FK, int, not null)
          NumberOfProduct (int, null)
          SalesmanProductListID (PK, FK, int, not null)
    ⊟ Keys
          PK_List2
          FK_ListSales_Product
          FK_ListSales_SalesmanProductList
      Constraints
      Triggers
    ⊟ Indexes
          PK_List2 (Clustered)
    ⊞ Statistics
```

This table is created to keep track of products taken from warehouses by salesmans. There are 2 primary key because of associative entity.

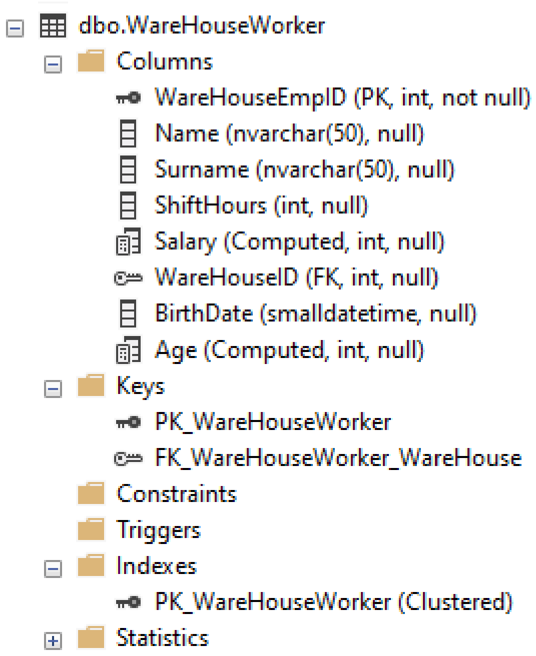
WareHouse Table



```
⊟ dbo.WareHouse
    ⊟ Columns
          WareHouseID (PK, int, not null)
          Name (nvarchar(50), null)
          Address (nvarchar(50), null)
    ⊟ Keys
          PK_WareHouse
      Constraints
      Triggers
    ⊟ Indexes
          PK_WareHouse (Clustered)
    ⊞ Statistics
```

This table keeps data of warehouses around Turkey. Primary key is identity it increase as you enter new data.

WareHouseWorker Table



```
⊟ dbo.WareHouseWorker
    ⊟ Columns
          WareHouseEmpID (PK, int, not null)
          Name (nvarchar(50), null)
          Surname (nvarchar(50), null)
          ShiftHours (int, null)
          Salary (Computed, int, null)
          WareHouseID (FK, int, null)
          BirthDate (smalldatetime, null)
          Age (Computed, int, null)
    ⊟ Keys
          PK_WareHouseWorker
          FK_WareHouseWorker_WareHouse
      Constraints
      Triggers
    ⊟ Indexes
          PK_WareHouseWorker (Clustered)
    ⊞ Statistics
```

These employees have shifthours. They have salary of asgari ücret. As they work more shifhour increases and so the salary. We have set default salary to 4250. And additionaly shifthours*10 is added to their salary. Foreign key connects to warehouse table. Primary key is identity it increase as you enter new data.

d) Views

e) Triggers

i) Name of the trigger.

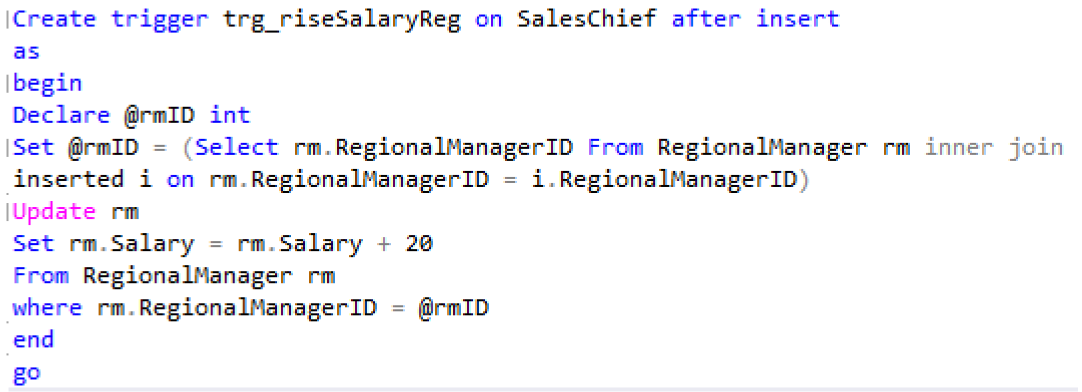
`trg_riseSalaryReg`

ii) Definition, and when/how it works.

This trigger works when a new saleschief is inserted. We basically increased the salary of regional manager. If there is more saleschief to be managed by regional managers than they should earn more money.

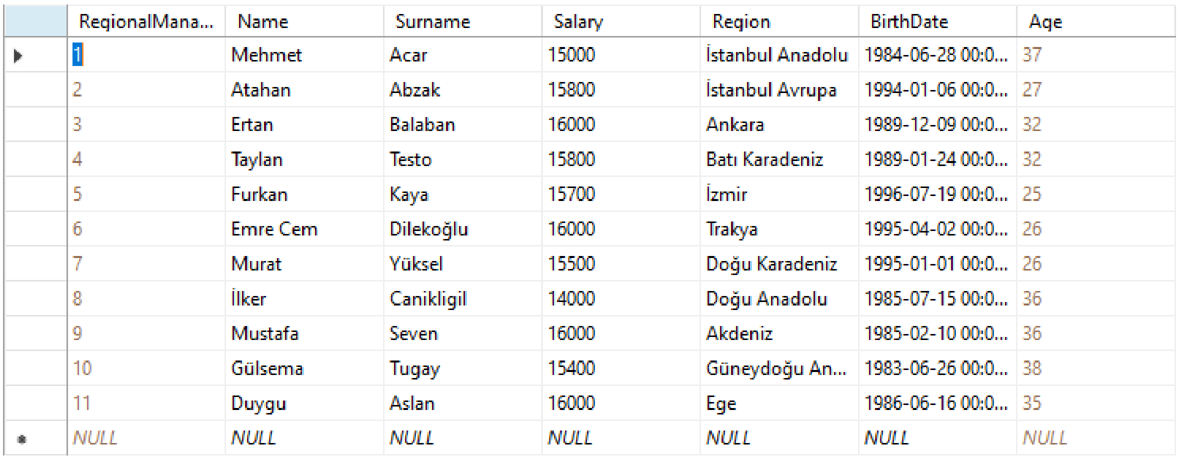
iii) Screenshot of the code of each trigger, and screenshots of before and after states of the table data that each

trigger works on.



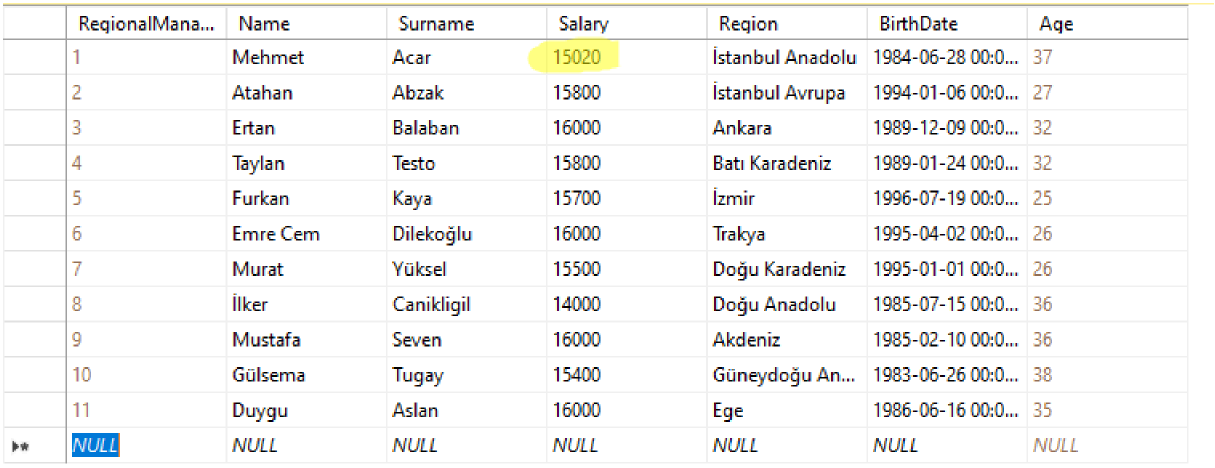
```
Create trigger trg_riseSalaryReg on SalesChief after insert
as
begin
Declare @rmID int
Set @rmID = (Select rm.RegionalManagerID From RegionalManager rm inner join
inserted i on rm.RegionalManagerID = i.RegionalManagerID)
Update rm
Set rm.Salary = rm.Salary + 20
From RegionalManager rm
where rm.RegionalManagerID = @rmID
end
go
```

Before



	RegionalMana...	Name	Surname	Salary	Region	BirthDate	Age
▶	1	Mehmet	Acar	15000	İstanbul Anadolu	1984-06-28 00:0...	37
	2	Atahan	Abzak	15800	İstanbul Avrupa	1994-01-06 00:0...	27
	3	Ertan	Balaban	16000	Ankara	1989-12-09 00:0...	32
	4	Taylan	Testo	15800	Batı Karadeniz	1989-01-24 00:0...	32
	5	Furkan	Kaya	15700	İzmir	1996-07-19 00:0...	25
	6	Emre Cem	Dilekoğlu	16000	Trakya	1995-04-02 00:0...	26
	7	Murat	Yüksel	15500	Doğu Karadeniz	1995-01-01 00:0...	26
	8	İlker	Canikligil	14000	Doğu Anadolu	1985-07-15 00:0...	36
	9	Mustafa	Seven	16000	Akdeniz	1985-02-10 00:0...	36
	10	Gülsema	Tugay	15400	Güneydoğu An...	1983-06-26 00:0...	38
	11	Duygu	Aslan	16000	Ege	1986-06-16 00:0...	35
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

After the insertion of a new salesChief



	RegionalMana...	Name	Surname	Salary	Region	BirthDate	Age
	1	Mehmet	Acar	15020	İstanbul Anadolu	1984-06-28 00:0...	37
	2	Atahan	Abzak	15800	İstanbul Avrupa	1994-01-06 00:0...	27
	3	Ertan	Balaban	16000	Ankara	1989-12-09 00:0...	32
	4	Taylan	Testo	15800	Batı Karadeniz	1989-01-24 00:0...	32
	5	Furkan	Kaya	15700	İzmir	1996-07-19 00:0...	25
	6	Emre Cem	Dilekoğlu	16000	Trakya	1995-04-02 00:0...	26
	7	Murat	Yüksel	15500	Doğu Karadeniz	1995-01-01 00:0...	26
	8	İlker	Canikligil	14000	Doğu Anadolu	1985-07-15 00:0...	36
	9	Mustafa	Seven	16000	Akdeniz	1985-02-10 00:0...	36
	10	Gülsema	Tugay	15400	Güneydoğu An...	1983-06-26 00:0...	38
	11	Duygu	Aslan	16000	Ege	1986-06-16 00:0...	35
▶*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

f) Stored procedures

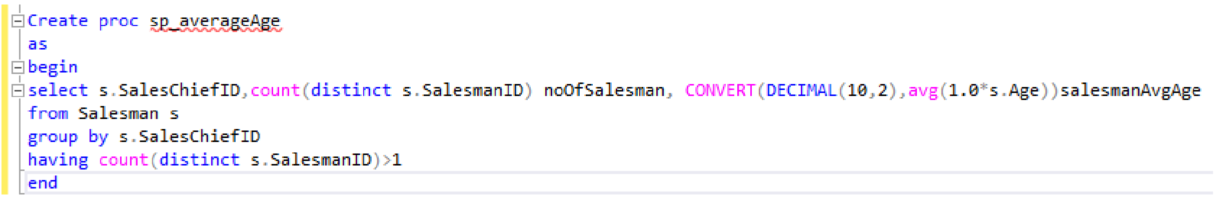
i) Name of the store procedure.

sp_averageAge

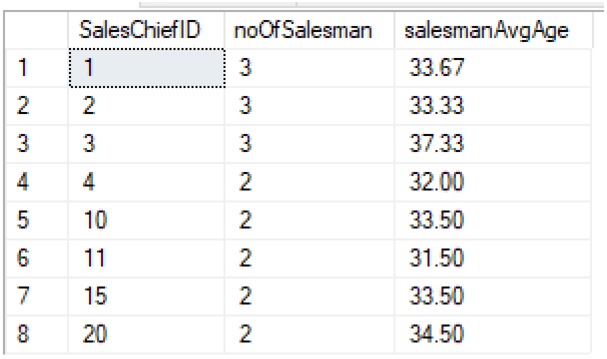
ii) Definition.

This stored procedure calculates the average age of the salesmans grouped by salesChiefs. If saleschief has only one salesman than procedure does not calculate avg of age.

iii) Screenshots of the code of each stored procedure, and screenshots of before and after states of the data that each stored procedure works on.



```
Create proc sp_averageAge
as
begin
select s.SalesChiefID,count(distinct s.SalesmanID) noOfSalesman, CONVERT(DECIMAL(10,2),avg(1.0*s.Age))salesmanAvgAge
from Salesman s
group by s.SalesChiefID
having count(distinct s.SalesmanID)>1
end
```



	SalesChiefID	noOfSalesman	salesmanAvgAge
1	1	3	33.67
2	2	3	33.33
3	3	3	37.33
4	4	2	32.00
5	10	2	33.50
6	11	2	31.50
7	15	2	33.50
8	20	2	34.50

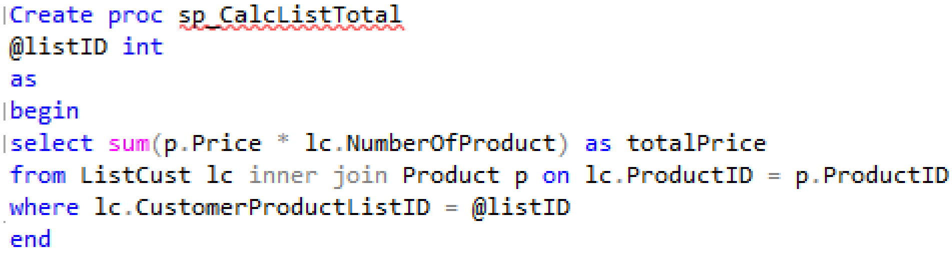
i) Name of the store procedure.

sp_CalcListTotal

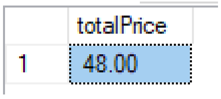
ii) Definition.

This sp calculates the total price of the product list that customer has with given listID

iii) Screenshots of the code of each stored procedure, and screenshots of before and after states of the data that each stored procedure works on.



```
Create proc sp_CalcListTotal
@listID int
as
begin
select sum(p.Price * lc.NumberOfProduct) as totalPrice
from ListCust lc inner join Product p on lc.ProductID = p.ProductID
where lc.CustomerProductListID = @listID
end
```



	totalPrice
1	48.00

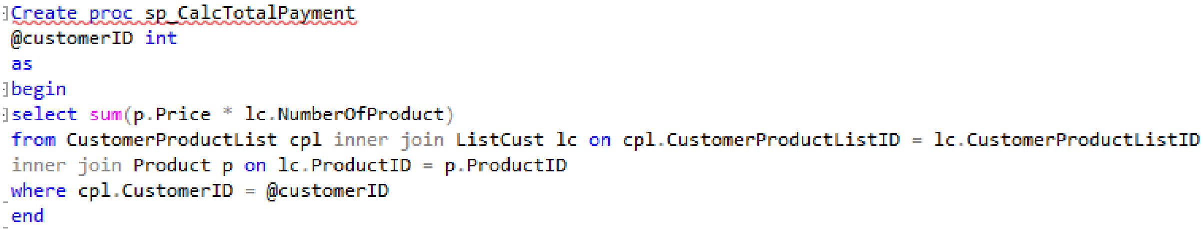
i) Name of the store procedure.

sp_CalcTotalPayment

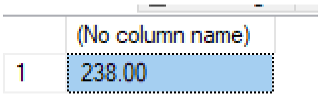
ii) Definition.

This sp calculates the total price of the all lists that customer has with the given customerID

iii) Screenshots of the code of each stored procedure, and screenshots of before and after states of the data that each stored procedure works on.



```
Create proc sp_CalcTotalPayment
@customerID int
as
begin
select sum(p.Price * lc.NumberOfProduct)
from CustomerProductList cpl inner join ListCust lc on cpl.CustomerProductListID = lc.CustomerProductListID
inner join Product p on lc.ProductID = p.ProductID
where cpl.CustomerID = @customerID
end
```



	(No column name)
1	238.00

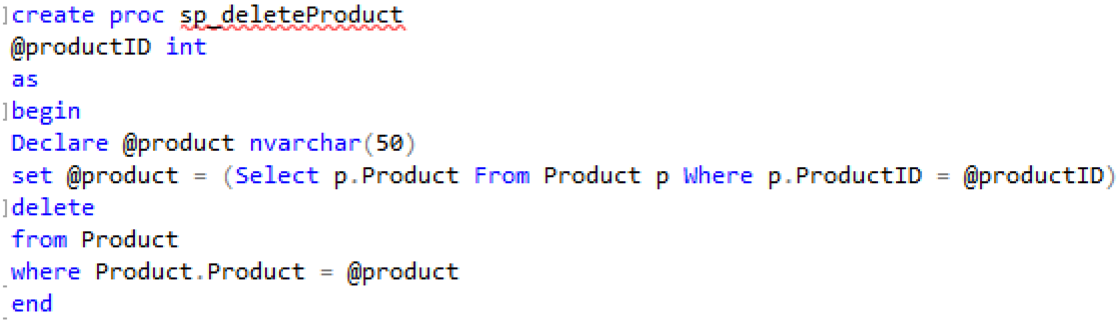
i) Name of the store procedure.

sp_deleteProduct

ii) Definition.

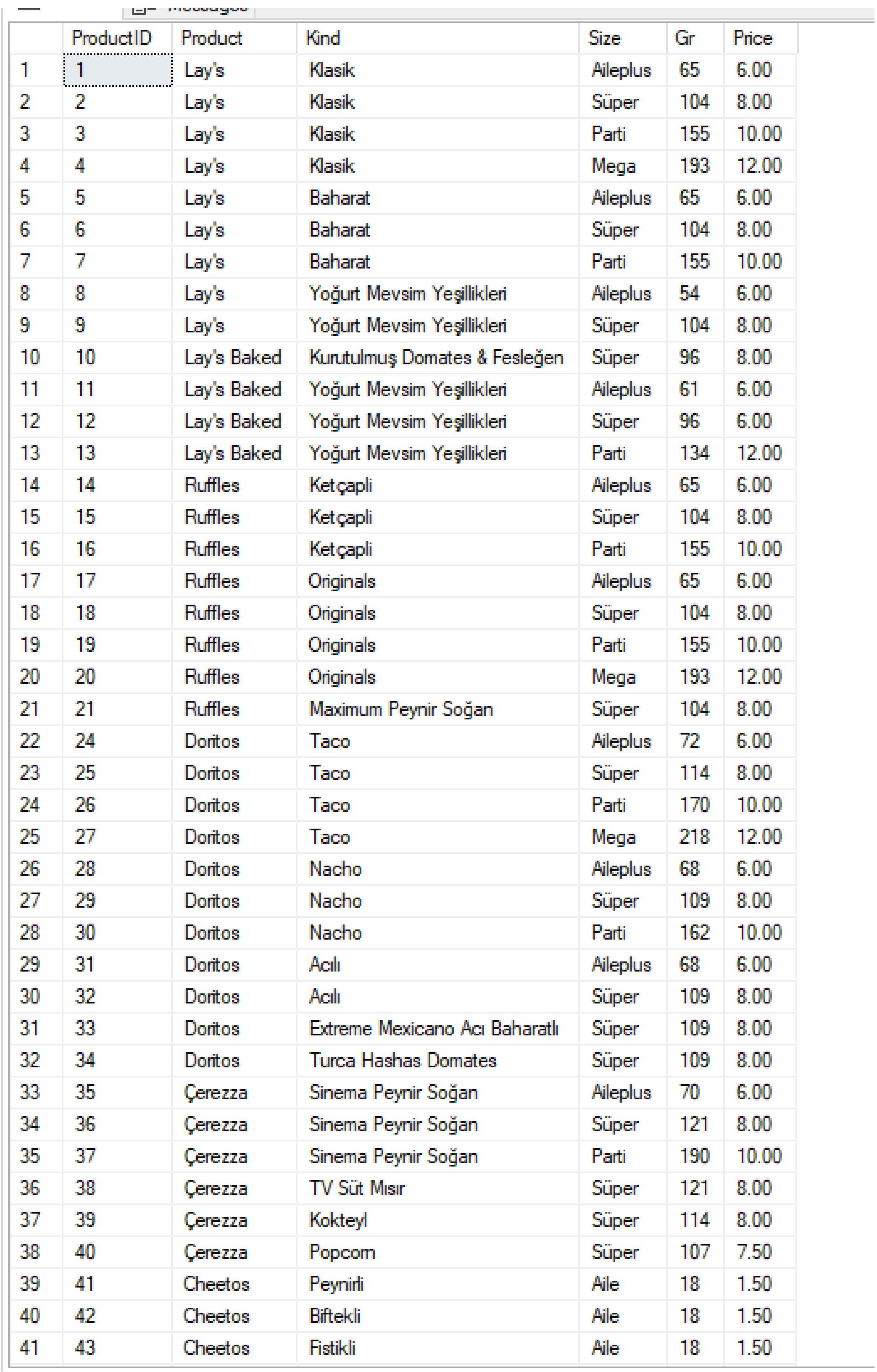
This sp deletes a product group from list with the given produc name.

iii) Screenshots of the code of each stored procedure, and screenshots of before and after states of the data that each stored procedure works on.



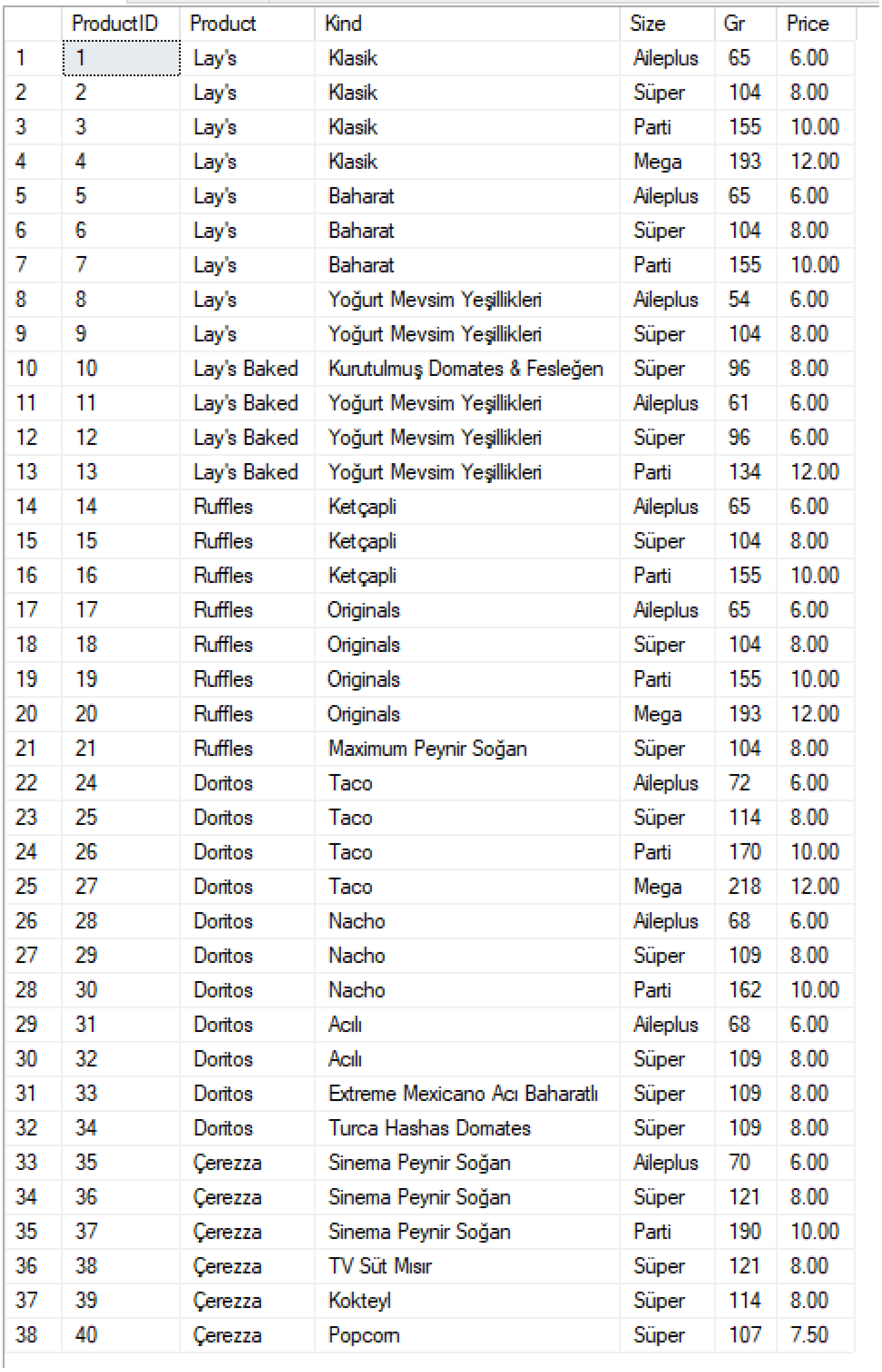
```
create proc sp_deleteProduct
@productID int
as
begin
Declare @product nvarchar(50)
set @product = (Select p.Product From Product p Where p.ProductID = @productID)
delete
from Product
where Product.Product = @product
end
```

Before the deletion



	ProductID	Product	Kind	Size	Gr	Price
1	1	Lay's	Klasik	Aileplus	65	6.00
2	2	Lay's	Klasik	Süper	104	8.00
3	3	Lay's	Klasik	Parti	155	10.00
4	4	Lay's	Klasik	Mega	193	12.00
5	5	Lay's	Baharat	Aileplus	65	6.00
6	6	Lay's	Baharat	Süper	104	8.00
7	7	Lay's	Baharat	Parti	155	10.00
8	8	Lay's	Yoğurt Mevsim Yeşillikleri	Aileplus	54	6.00
9	9	Lay's	Yoğurt Mevsim Yeşillikleri	Süper	104	8.00
10	10	Lay's Baked	Kurutulmuş Domates & Fesleğen	Süper	96	8.00
11	11	Lay's Baked	Yoğurt Mevsim Yeşillikleri	Aileplus	61	6.00
12	12	Lay's Baked	Yoğurt Mevsim Yeşillikleri	Süper	96	6.00
13	13	Lay's Baked	Yoğurt Mevsim Yeşillikleri	Parti	134	12.00
14	14	Ruffles	Ketçapli	Aileplus	65	6.00
15	15	Ruffles	Ketçapli	Süper	104	8.00
16	16	Ruffles	Ketçapli	Parti	155	10.00
17	17	Ruffles	Originals	Aileplus	65	6.00
18	18	Ruffles	Originals	Süper	104	8.00
19	19	Ruffles	Originals	Parti	155	10.00
20	20	Ruffles	Originals	Mega	193	12.00
21	21	Ruffles	Maximum Peynir Soğan	Süper	104	8.00
22	24	Doritos	Taco	Aileplus	72	6.00
23	25	Doritos	Taco	Süper	114	8.00
24	26	Doritos	Taco	Parti	170	10.00
25	27	Doritos	Taco	Mega	218	12.00
26	28	Doritos	Nacho	Aileplus	68	6.00
27	29	Doritos	Nacho	Süper	109	8.00
28	30	Doritos	Nacho	Parti	162	10.00
29	31	Doritos	Acılı	Aileplus	68	6.00
30	32	Doritos	Acılı	Süper	109	8.00
31	33	Doritos	Extreme Mexicano Acı Baharatlı	Süper	109	8.00
32	34	Doritos	Turca Hashas Domates	Süper	109	8.00
33	35	Çerezza	Sinema Peynir Soğan	Aileplus	70	6.00
34	36	Çerezza	Sinema Peynir Soğan	Süper	121	8.00
35	37	Çerezza	Sinema Peynir Soğan	Parti	190	10.00
36	38	Çerezza	TV Süt Mısır	Süper	121	8.00
37	39	Çerezza	Kokteyl	Süper	114	8.00
38	40	Çerezza	Popcorn	Süper	107	7.50
39	41	Cheetos	Peynirli	Aile	18	1.50
40	42	Cheetos	Biftekli	Aile	18	1.50
41	43	Cheetos	Fistikli	Aile	18	1.50

after the deletion of cheetos



	ProductID	Product	Kind	Size	Gr	Price
1	1	Lay's	Klasik	Aileplus	65	6.00
2	2	Lay's	Klasik	Süper	104	8.00
3	3	Lay's	Klasik	Parti	155	10.00
4	4	Lay's	Klasik	Mega	193	12.00
5	5	Lay's	Baharat	Aileplus	65	6.00
6	6	Lay's	Baharat	Süper	104	8.00
7	7	Lay's	Baharat	Parti	155	10.00
8	8	Lay's	Yoğurt Mevsim Yeşillikleri	Aileplus	54	6.00
9	9	Lay's	Yoğurt Mevsim Yeşillikleri	Süper	104	8.00
10	10	Lay's Baked	Kurutulmuş Domates & Fesleğen	Süper	96	8.00
11	11	Lay's Baked	Yoğurt Mevsim Yeşillikleri	Aileplus	61	6.00
12	12	Lay's Baked	Yoğurt Mevsim Yeşillikleri	Süper	96	6.00
13	13	Lay's Baked	Yoğurt Mevsim Yeşillikleri	Parti	134	12.00
14	14	Ruffles	Ketçapli	Aileplus	65	6.00
15	15	Ruffles	Ketçapli	Süper	104	8.00
16	16	Ruffles	Ketçapli	Parti	155	10.00
17	17	Ruffles	Originals	Aileplus	65	6.00
18	18	Ruffles	Originals	Süper	104	8.00
19	19	Ruffles	Originals	Parti	155	10.00
20	20	Ruffles	Originals	Mega	193	12.00
21	21	Ruffles	Maximum Peynir Soğan	Süper	104	8.00
22	24	Doritos	Taco	Aileplus	72	6.00
23	25	Doritos	Taco	Süper	114	8.00
24	26	Doritos	Taco	Parti	170	10.00
25	27	Doritos	Taco	Mega	218	12.00
26	28	Doritos	Nacho	Aileplus	68	6.00
27	29	Doritos	Nacho	Süper	109	8.00
28	30	Doritos	Nacho	Parti	162	10.00
29	31	Doritos	Acılı	Aileplus	68	6.00
30	32	Doritos	Acılı	Süper	109	8.00
31	33	Doritos	Extreme Mexicano Acı Baharatlı	Süper	109	8.00
32	34	Doritos	Turca Hashas Domates	Süper	109	8.00
33	35	Çerezza	Sinema Peynir Soğan	Aileplus	70	6.00
34	36	Çerezza	Sinema Peynir Soğan	Süper	121	8.00
35	37	Çerezza	Sinema Peynir Soğan	Parti	190	10.00
36	38	Çerezza	TV Süt Mısır	Süper	121	8.00
37	39	Çerezza	Kokteyl	Süper	114	8.00
38	40	Çerezza	Popcorn	Süper	107	7.50

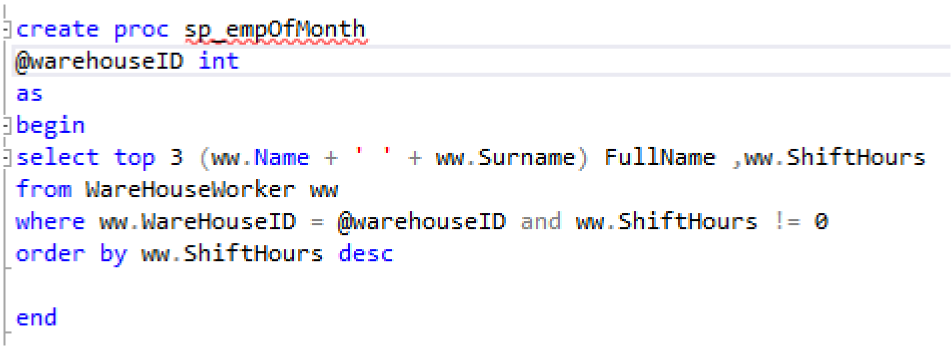
i) Name of the store procedure.

sp_empOfMonth

ii) Definition.

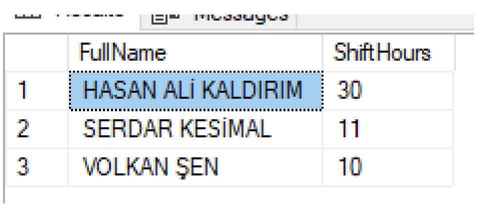
This sp shows top 3 employee who worked the most around warehouses workers calculated by shifthours with the given warehouseid.

iii) Screenshots of the code of each stored procedure, and screenshots of before and after states of the data that each stored procedure works on.



```
create proc sp_empOfMonth
@warehouseID int
as
begin
select top 3 (ww.Name + ' ' + ww.Surname) FullName ,ww.ShiftHours
from WareHouseWorker ww
where ww.WareHouseID = @warehouseID and ww.ShiftHours != 0
order by ww.ShiftHours desc

end
```



	FullName	ShiftHours
1	HASAN ALİ KALDIRIM	30
2	SERDAR KESİMAL	11
3	VOLKAN ŞEN	10

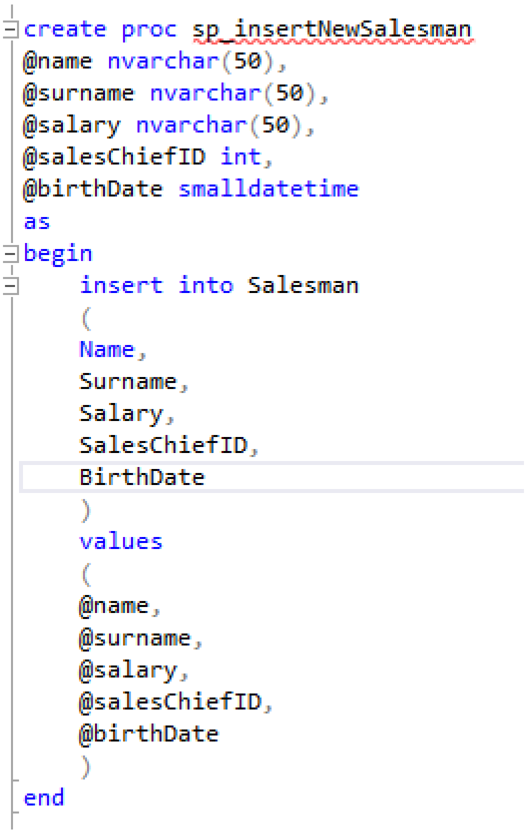
i) Name of the store procedure.

sp_insertNewSalesman

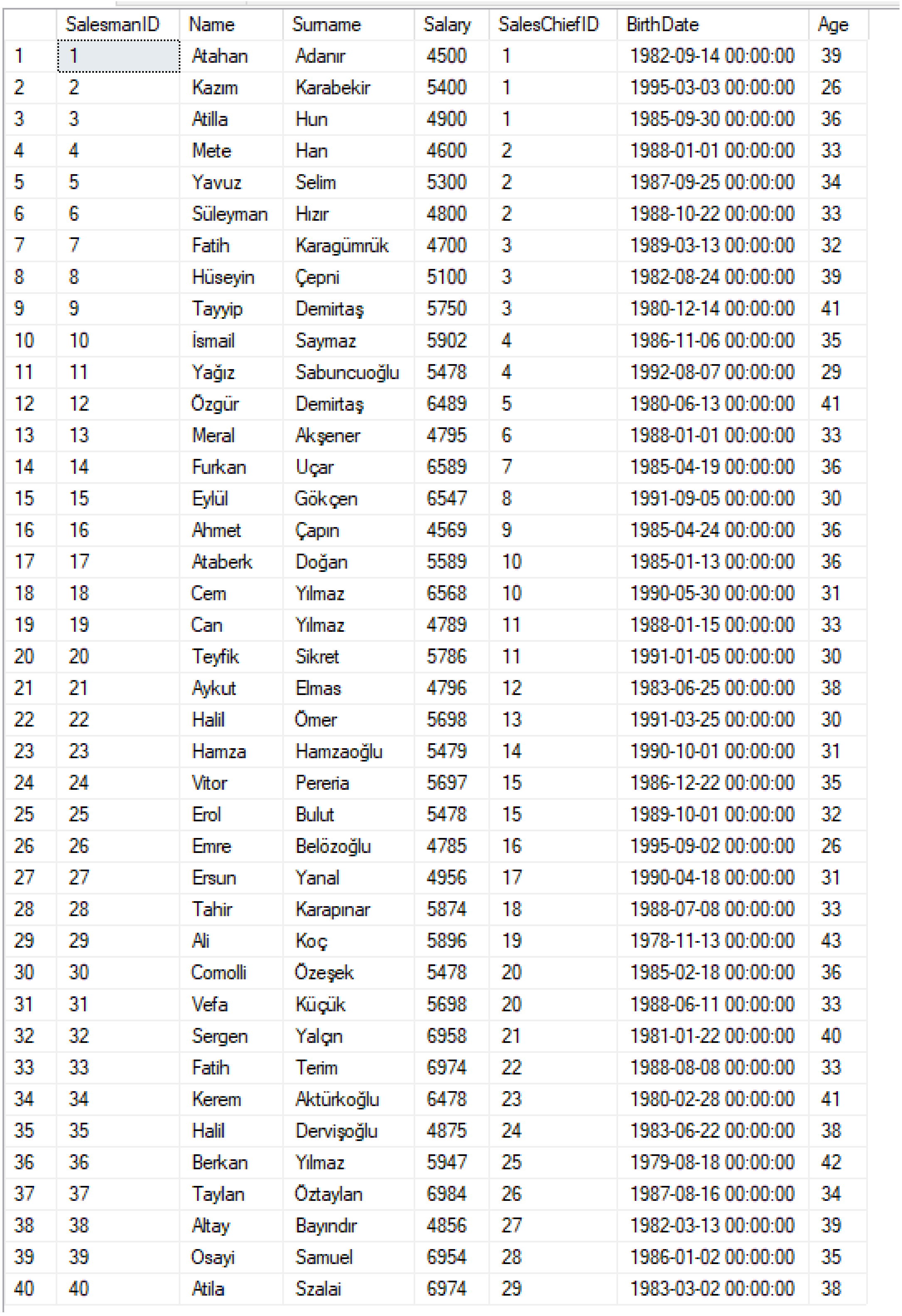
ii) Definition.

This sp inserts a new Salesman to table with the given parameters.

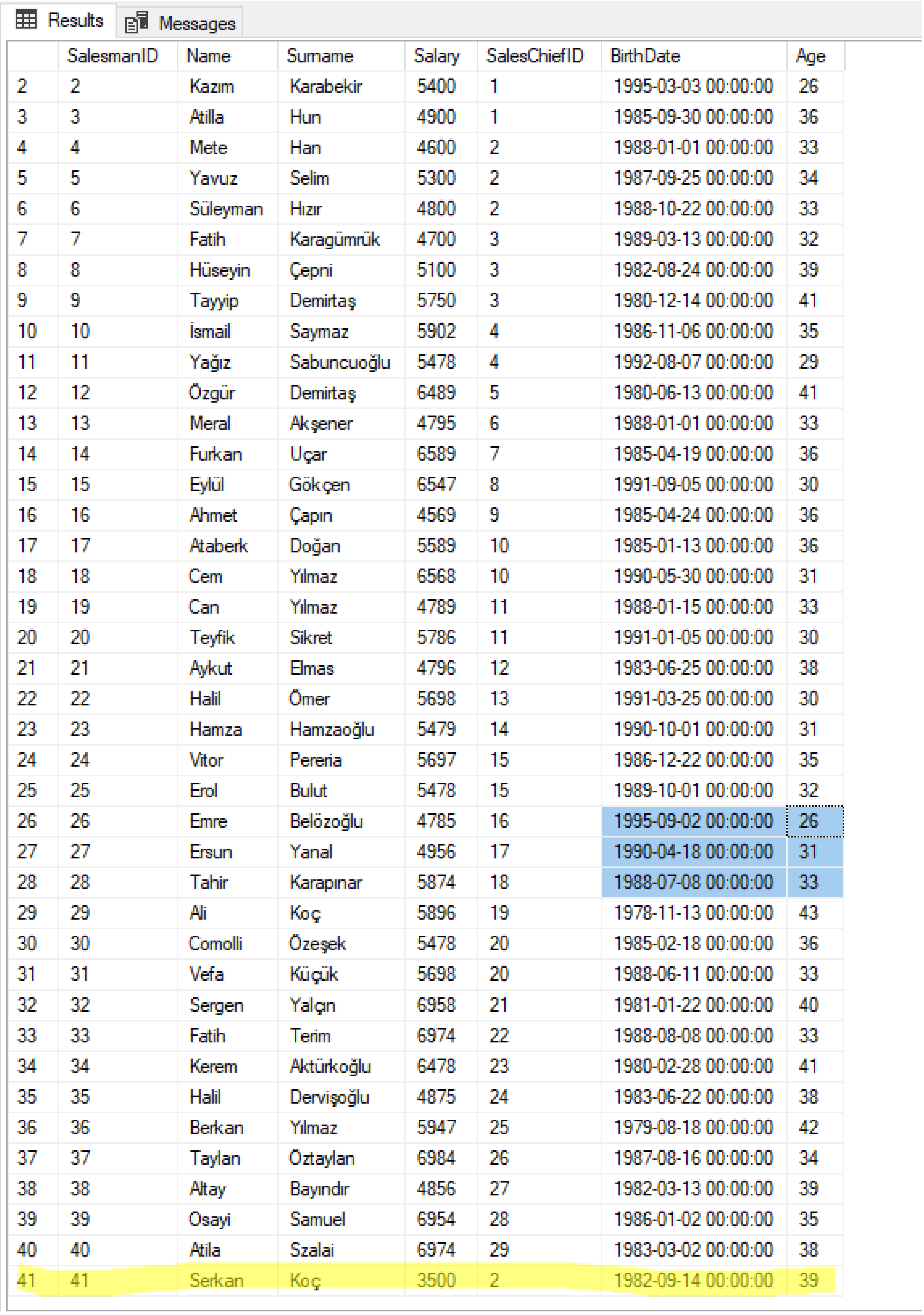
iii) Screenshots of the code of each stored procedure, and screenshots of before and after states of the data that each stored procedure works on.



```
create proc sp_insertNewSalesman
@name nvarchar(50),
@surname nvarchar(50),
@salary nvarchar(50),
@salesChiefID int,
@birthDate smalldatetime
as
begin
    insert into Salesman
    (
    Name,
    Surname,
    Salary,
    SalesChiefID,
    BirthDate
    )
    values
    (
    @name,
    @surname,
    @salary,
    @salesChiefID,
    @birthDate
    )
end
```



	SalesmanID	Name	Surname	Salary	SalesChiefID	BirthDate	Age
1	1	Atahan	Adanır	4500	1	1982-09-14 00:00:00	39
2	2	Kazım	Karabekir	5400	1	1995-03-03 00:00:00	26
3	3	Atilla	Hun	4900	1	1985-09-30 00:00:00	36
4	4	Mete	Han	4600	2	1988-01-01 00:00:00	33
5	5	Yavuz	Selim	5300	2	1987-09-25 00:00:00	34
6	6	Süleyman	Hızır	4800	2	1988-10-22 00:00:00	33
7	7	Fatih	Karagümrük	4700	3	1989-03-13 00:00:00	32
8	8	Hüseyin	Çepni	5100	3	1982-08-24 00:00:00	39
9	9	Tayyip	Demirtaş	5750	3	1980-12-14 00:00:00	41
10	10	İsmail	Saymaz	5902	4	1986-11-06 00:00:00	35
11	11	Yağız	Sabuncuoğlu	5478	4	1992-08-07 00:00:00	29
12	12	Özgür	Demirtaş	6489	5	1980-06-13 00:00:00	41
13	13	Meral	Akşener	4795	6	1988-01-01 00:00:00	33
14	14	Furkan	Uçar	6589	7	1985-04-19 00:00:00	36
15	15	Eylül	Gökçen	6547	8	1991-09-05 00:00:00	30
16	16	Ahmet	Çapın	4569	9	1985-04-24 00:00:00	36
17	17	Ataberk	Doğan	5589	10	1985-01-13 00:00:00	36
18	18	Cem	Yılmaz	6568	10	1990-05-30 00:00:00	31
19	19	Can	Yılmaz	4789	11	1988-01-15 00:00:00	33
20	20	Teyfik	Sikret	5786	11	1991-01-05 00:00:00	30
21	21	Aykut	Elmas	4796	12	1983-06-25 00:00:00	38
22	22	Halil	Ömer	5698	13	1991-03-25 00:00:00	30
23	23	Hamza	Hamzaoğlu	5479	14	1990-10-01 00:00:00	31
24	24	Vitor	Pereria	5697	15	1986-12-22 00:00:00	35
25	25	Erol	Bulut	5478	15	1989-10-01 00:00:00	32
26	26	Emre	Belözoğlu	4785	16	1995-09-02 00:00:00	26
27	27	Ersun	Yanal	4956	17	1990-04-18 00:00:00	31
28	28	Tahir	Karapınar	5874	18	1988-07-08 00:00:00	33
29	29	Ali	Koç	5896	19	1978-11-13 00:00:00	43
30	30	Comolli	Özeşek	5478	20	1985-02-18 00:00:00	36
31	31	Vefa	Küçük	5698	20	1988-06-11 00:00:00	33
32	32	Sergen	Yalçın	6958	21	1981-01-22 00:00:00	40
33	33	Fatih	Terim	6974	22	1988-08-08 00:00:00	33
34	34	Kerem	Aktürkoğlu	6478	23	1980-02-28 00:00:00	41
35	35	Halil	Dervişoğlu	4875	24	1983-06-22 00:00:00	38
36	36	Berkan	Yılmaz	5947	25	1979-08-18 00:00:00	42
37	37	Taylan	Öztaylan	6984	26	1987-08-16 00:00:00	34
38	38	Altay	Bayındır	4856	27	1982-03-13 00:00:00	39
39	39	Osayi	Samuel	6954	28	1986-01-02 00:00:00	35
40	40	Atila	Szalai	6974	29	1983-03-02 00:00:00	38



	SalesmanID	Name	Surname	Salary	SalesChiefID	BirthDate	Age
2	2	Kazım	Karabekir	5400	1	1995-03-03 00:00:00	26
3	3	Atilla	Hun	4900	1	1985-09-30 00:00:00	36
4	4	Mete	Han	4600	2	1988-01-01 00:00:00	33
5	5	Yavuz	Selim	5300	2	1987-09-25 00:00:00	34
6	6	Süleyman	Hızır	4800	2	1988-10-22 00:00:00	33
7	7	Fatih	Karagümrük	4700	3	1989-03-13 00:00:00	32
8	8	Hüseyin	Çepni	5100	3	1982-08-24 00:00:00	39
9	9	Tayyip	Demirtaş	5750	3	1980-12-14 00:00:00	41
10	10	İsmail	Saymaz	5902	4	1986-11-06 00:00:00	35
11	11	Yağız	Sabuncuoğlu	5478	4	1992-08-07 00:00:00	29
12	12	Özgür	Demirtaş	6489	5	1980-06-13 00:00:00	41
13	13	Meral	Akşener	4795	6	1988-01-01 00:00:00	33
14	14	Furkan	Uçar	6589	7	1985-04-19 00:00:00	36
15	15	Eylül	Gökçen	6547	8	1991-09-05 00:00:00	30
16	16	Ahmet	Çapın	4569	9	1985-04-24 00:00:00	36
17	17	Ataberk	Doğan	5589	10	1985-01-13 00:00:00	36
18	18	Cem	Yılmaz	6568	10	1990-05-30 00:00:00	31
19	19	Can	Yılmaz	4789	11	1988-01-15 00:00:00	33
20	20	Teyfik	Sikret	5786	11	1991-01-05 00:00:00	30
21	21	Aykut	Elmas	4796	12	1983-06-25 00:00:00	38
22	22	Halil	Ömer	5698	13	1991-03-25 00:00:00	30
23	23	Hamza	Hamzaoğlu	5479	14	1990-10-01 00:00:00	31
24	24	Vitor	Pereria	5697	15	1986-12-22 00:00:00	35
25	25	Erol	Bulut	5478	15	1989-10-01 00:00:00	32
26	26	Emre	Belözoğlu	4785	16	1995-09-02 00:00:00	26
27	27	Ersun	Yanal	4956	17	1990-04-18 00:00:00	31
28	28	Tahir	Karapınar	5874	18	1988-07-08 00:00:00	33
29	29	Ali	Koç	5896	19	1978-11-13 00:00:00	43
30	30	Comolli	Özeşek	5478	20	1985-02-18 00:00:00	36
31	31	Vefa	Küçük	5698	20	1988-06-11 00:00:00	33
32	32	Sergen	Yalçın	6958	21	1981-01-22 00:00:00	40
33	33	Fatih	Terim	6974	22	1988-08-08 00:00:00	33
34	34	Kerem	Aktürkoğlu	6478	23	1980-02-28 00:00:00	41
35	35	Halil	Dervişoğlu	4875	24	1983-06-22 00:00:00	38
36	36	Berkan	Yılmaz	5947	25	1979-08-18 00:00:00	42
37	37	Taylan	Öztaylan	6984	26	1987-08-16 00:00:00	34
38	38	Altay	Bayındır	4856	27	1982-03-13 00:00:00	39
39	39	Osayi	Samuel	6954	28	1986-01-02 00:00:00	35
40	40	Atila	Szalai	6974	29	1983-03-02 00:00:00	38
41	41	Serkan	Koç	3500	2	1982-09-14 00:00:00	39

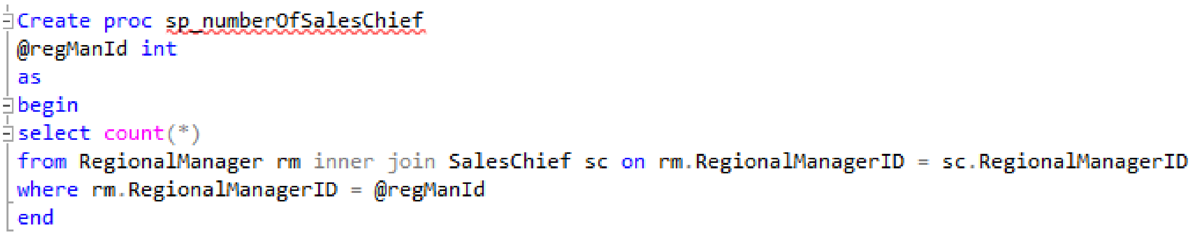
i) Name of the store procedure.

sp_numberOfSalesChief

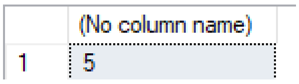
ii) Definition.

This sp calculates the number of salesChiefs that a regional manager manages with the given regionalManagerID input.

iii) Screenshots of the code of each stored procedure, and screenshots of before and after states of the data that each stored procedure works on.



```
Create proc sp_numberOfSalesChief
@regManId int
as
begin
select count(*)
from RegionalManager rm inner join SalesChief sc on rm.RegionalManagerID = sc.RegionalManagerID
where rm.RegionalManagerID = @regManId
end
```



	(No column name)
1	5

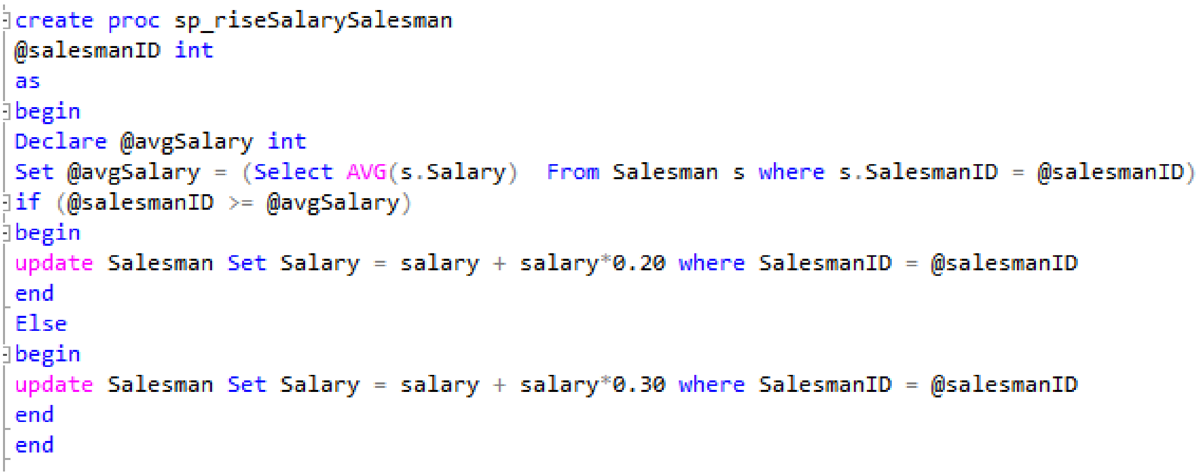
i) Name of the store procedure.

sp_riseSalarySalesman

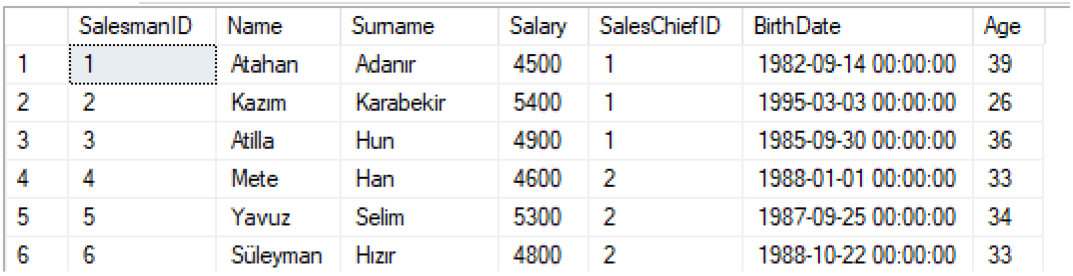
ii) Definition.

This sp increases the salary of a salesman with the if else statements with the amount of percentages. SalesmanId is input parameter.

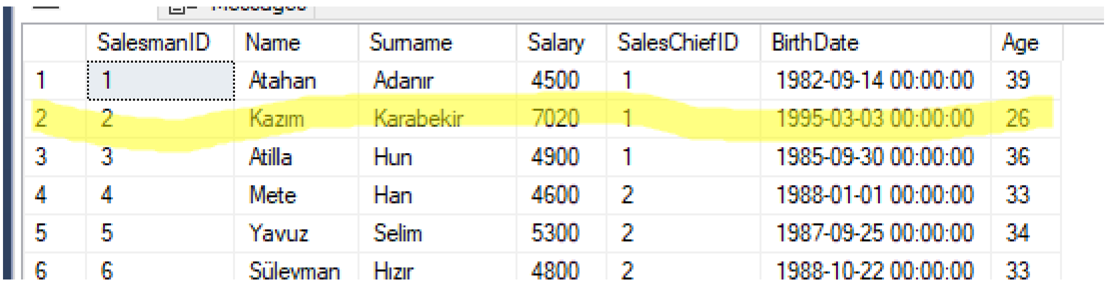
iii) Screenshots of the code of each stored procedure, and screenshots of before and after states of the data that each stored procedure works on.



```
create proc sp_riseSalarySalesman
@salesmanID int
as
begin
Declare @avgSalary int
Set @avgSalary = (Select AVG(s.Salary)  From Salesman s where s.SalesmanID = @salesmanID)
if (@salesmanID >= @avgSalary)
begin
update Salesman Set Salary = salary + salary*0.20 where SalesmanID = @salesmanID
end
Else
begin
update Salesman Set Salary = salary + salary*0.30 where SalesmanID = @salesmanID
end
end
```



	SalesmanID	Name	Surname	Salary	SalesChiefID	BirthDate	Age
1	1	Atahan	Adanır	4500	1	1982-09-14 00:00:00	39
2	2	Kazım	Karabekir	5400	1	1995-03-03 00:00:00	26
3	3	Atilla	Hun	4900	1	1985-09-30 00:00:00	36
4	4	Mete	Han	4600	2	1988-01-01 00:00:00	33
5	5	Yavuz	Selim	5300	2	1987-09-25 00:00:00	34
6	6	Süleyman	Hızır	4800	2	1988-10-22 00:00:00	33



	SalesmanID	Name	Surname	Salary	SalesChiefID	BirthDate	Age
1	1	Atahan	Adanır	4500	1	1982-09-14 00:00:00	39
2	2	Kazım	Karabekir	7020	1	1995-03-03 00:00:00	26
3	3	Atilla	Hun	4900	1	1985-09-30 00:00:00	36
4	4	Mete	Han	4600	2	1988-01-01 00:00:00	33
5	5	Yavuz	Selim	5300	2	1987-09-25 00:00:00	34
6	6	Süleyman	Hızır	4800	2	1988-10-22 00:00:00	33

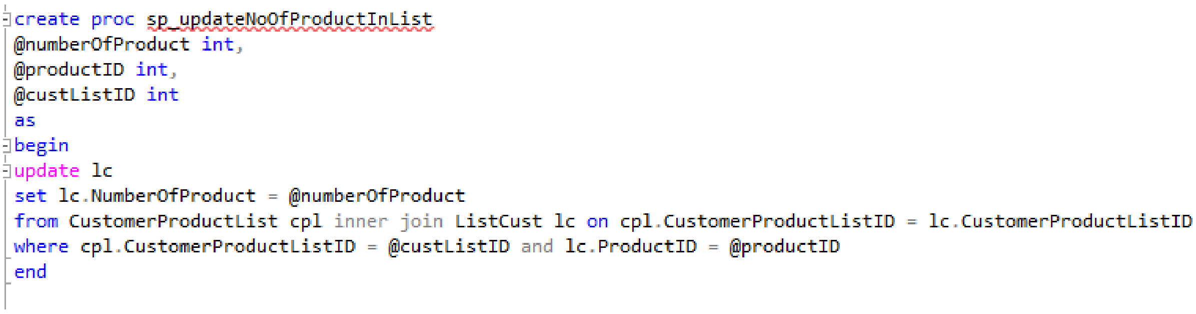
i) Name of the store procedure.

sp_updateNoOfProductInList

ii) Definition.

This sp updates the number of a product in a list with the input parameters.

iii) Screenshots of the code of each stored procedure, and screenshots of before and after states of the data that each stored procedure works on.



```
create proc sp_updateNoOfProductInList
@numberOfProduct int,
@productID int,
@custListID int
as
begin
update lc
set lc.NumberOfProduct = @numberOfProduct
from CustomerProductList cpl inner join ListCust lc on cpl.CustomerProductListID = lc.CustomerProductListID
where cpl.CustomerProductListID = @custListID and lc.ProductID = @productID
end
```

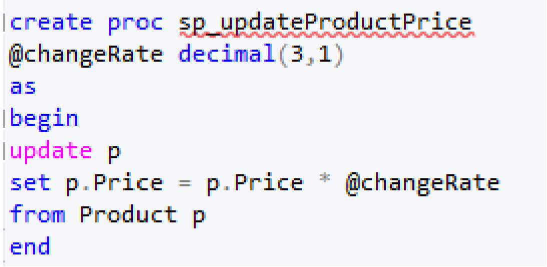
i) Name of the store procedure.

sp_updateProductPrice

ii) Definition.

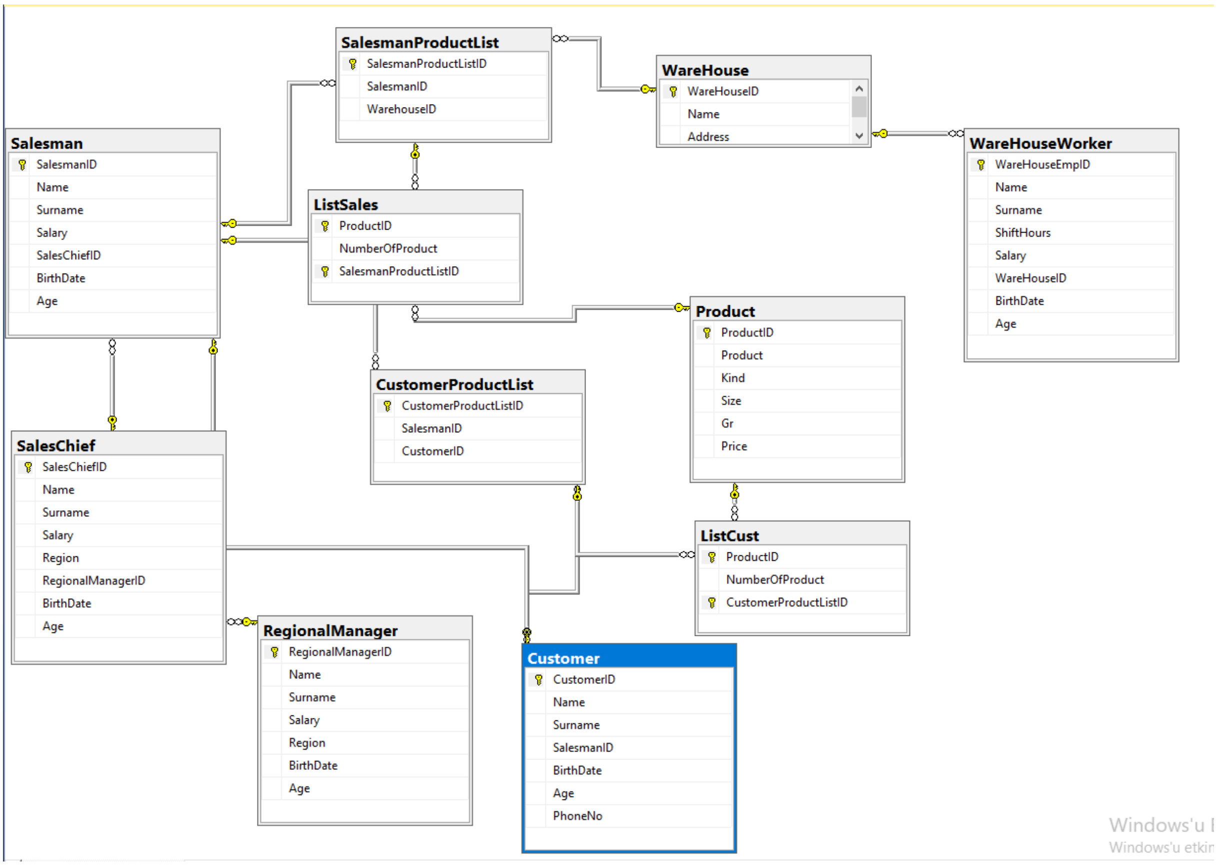
This sp updates the price of a product with the given percentage.

iii) Screenshots of the code of each stored procedure, and screenshots of before and after states of the data that each stored procedure works on.



```
create proc sp_updateProductPrice
@changeRate decimal(3,1)
as
begin
update p
set p.Price = p.Price * @changeRate
from Product p
end
```

DIAGRAM OF WHOLE DATABASE


SalesmanProductList
SalesmanProductListID
SalesmanID
WarehouseID
WareHouse
WareHouseID
Name
Address
Salesman
SalesmanID
Name
Surname
Salary
SalesChiefID
BirthDate
Age
WareHouseWorker
WareHouseEmpID
Name
Surname
ShiftHours
Salary
WareHouseID
BirthDate
Age
ListSales
ProductID
NumberOfProduct
SalesmanProductListID
Product
ProductID
Product
Kind
Size
Gr
Price
CustomerProductList
CustomerProductListID
SalesmanID
CustomerID
SalesChief
SalesChiefID
Name
Surname
Salary
Region
RegionalManagerID
BirthDate
Age
ListCust
ProductID
NumberOfProduct
CustomerProductListID
RegionalManager
RegionalManagerID
Name
Surname
Salary
Region
BirthDate
Age
Customer
CustomerID
Name
Surname
SalesmanID
BirthDate
Age
PhoneNo
Windows'u l
Windows'u etkin