Fall 2020 – CSE3055 Database Systems
Project Step 3

Logical database design and mapping & Physical design and database implementation + (Req. analysis & Concep. db. des.)

Group Members:

150117007 – Edanur Öztürk

150117033 – Feyza Nur Bulgurcu

150117044 – Sueda Bilen

Project Description and Scope:

The database system holds product ,customer ,wholesaler ,employee ,bill, and shopping infos.

Our goal with this project was ,creating a practical ,applicable and arrangable database for a local mini market .We made our database based on our goal .It stores the customer infos and their preferences of buying products in shopping and shoppingInfo tables .There are wholesalers who supply products to the mini market therefore the database keeps data of wholesalers and the products that supplied from the wholesaler.

Our objective was observing choices and keeping the data of the customers who live around Cerrahpasa .After implementing the database we satisfied that objective .The database demonstrates the most purchased products by customers, the most preferred product type by customers and average customer age for keeping track of preferences from view tables. We included every requirement in step1 and 2.

Data and Requirement Analysis:

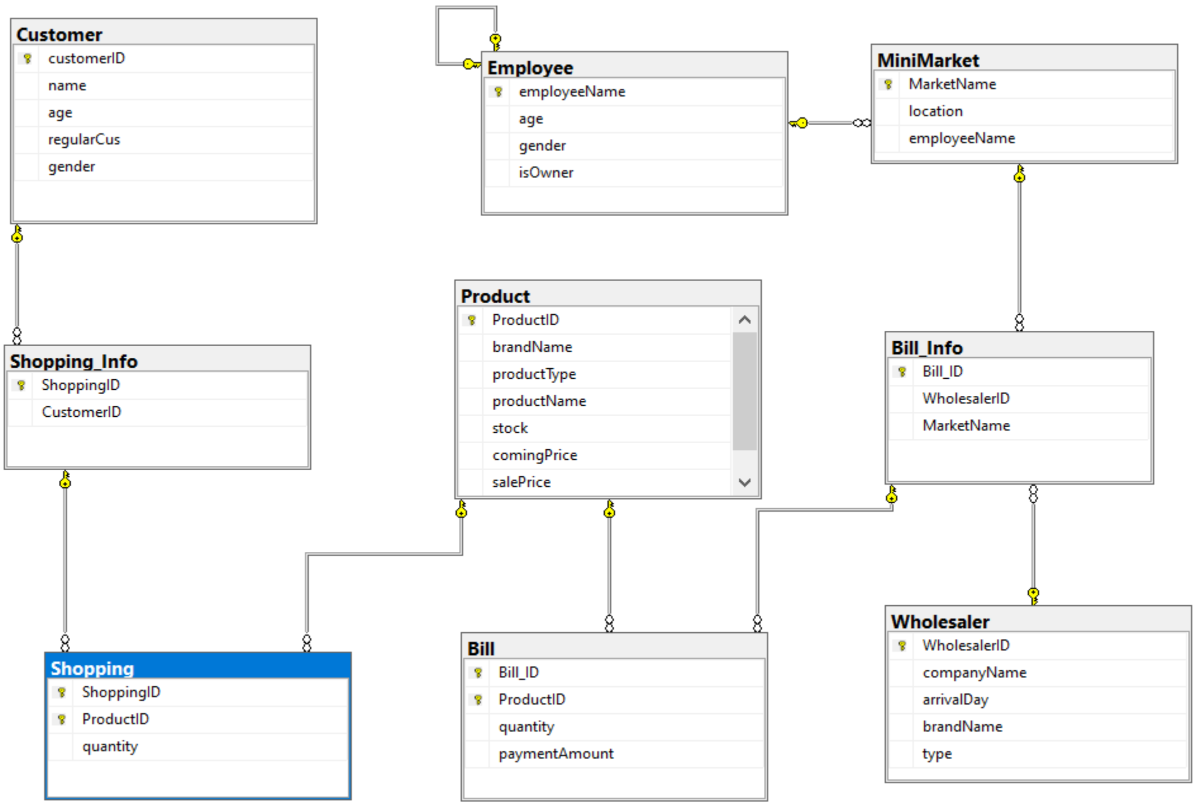
As business process ,firstly Mini Market owner makes deals with wholesalers .Wholesalers supply products determined day of each week .Owner calculates the prices of the products .After that ,customers come and buy products.

According to them ,we thought keeping bill of products that market owner ordered by wholesalers in Product table .Unfortunately , this condition should be normalized to the 3rd normal form .Then we keep the data of products and wholesaler separately ,also we have tables for Bill and Bill Information that determines the relation between owner, wholesaler and products.

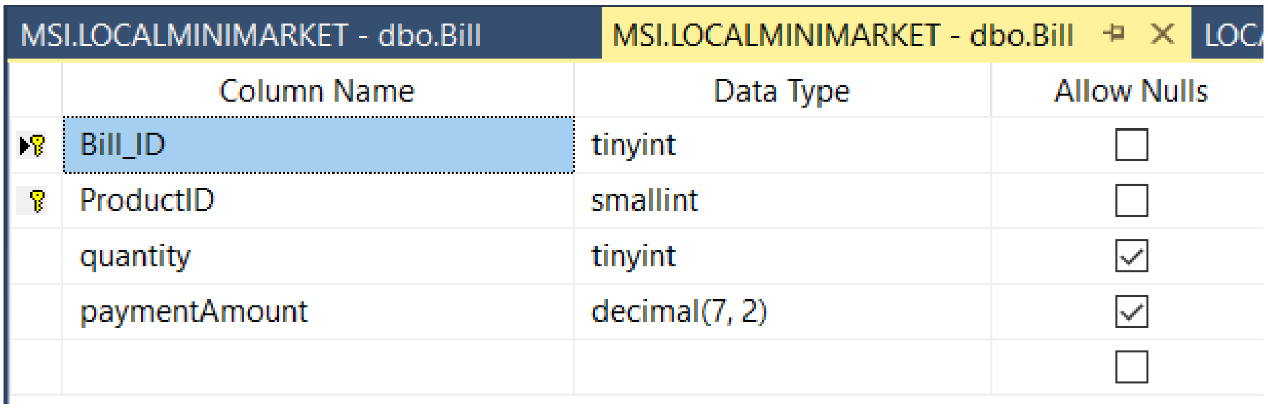
After that ,we thought about keeping shopping information of customers for accessing preferences in the Customer table. Likewise other condition ,in this step we needed to normalize customer table to the 3rd normal form .Then, we keep the data of products and customers separately ,also we have tables for Shopping and Shopping Info to hold information of purchased products.

System keeps stock records for frequently bought products so that we don't have a stock-out problem later .We have planned to use trigger to handle this problem so every time a product entered to system that trigger can increase the stock by taking quantity of entered product .Also ,every time a product purchased to system that trigger can decrease the stock by taking quantity of purchased product.

Diagram of whole database:



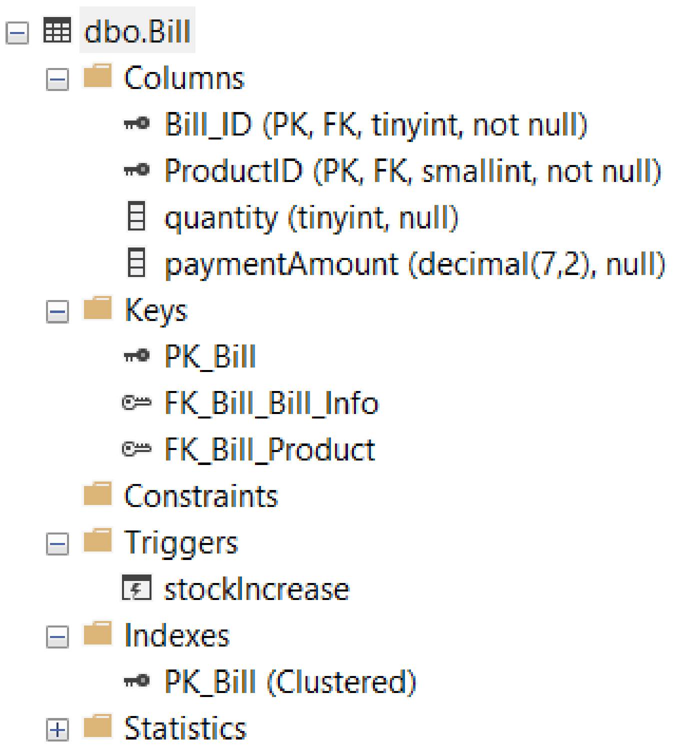
Tables:



MSI.LOCALMINIMARKET - dbo.Bill | MSI.LOCALMINIMARKET - dbo.Bill | LOC

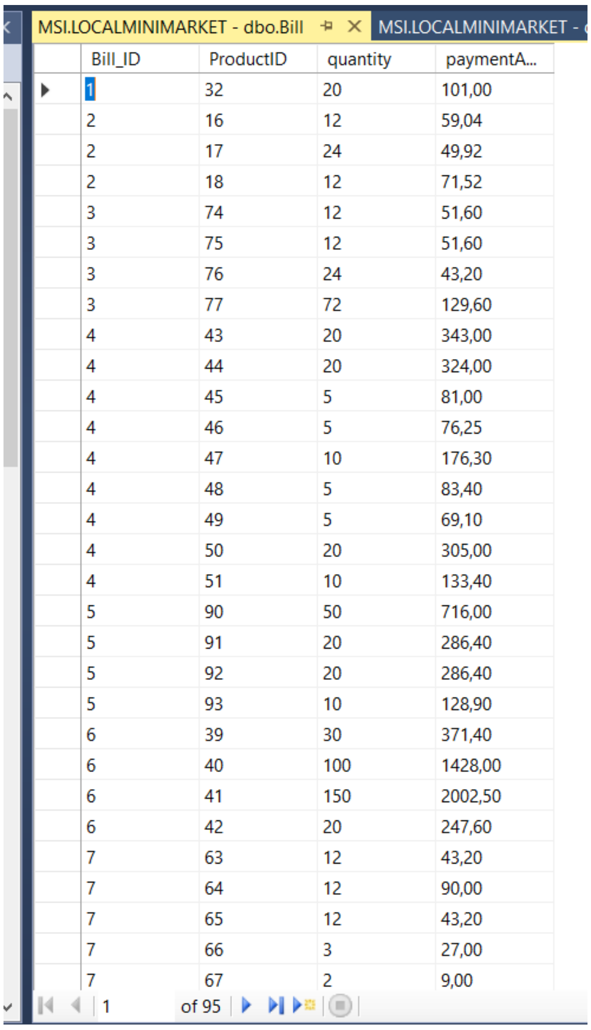
Column Name	Data Type	Allow Nulls
Bill_ID	tinyint	☐
ProductID	smallint	☐
quantity	tinyint	✓
paymentAmount	decimal(7, 2)	✓
		☐

Bill Table:This table keeps data of bills ,calculates payment amount according to quantities and product table’s coming price value.



```
dbo.Bill
  Columns
    Bill_ID (PK, FK, tinyint, not null)
    ProductID (PK, FK, smallint, not null)
    quantity (tinyint, null)
    paymentAmount (decimal(7,2), null)
  Keys
    PK_Bill
    FK_Bill_Bill_Info
    FK_Bill_Product
  Constraints
  Triggers
    stockIncrease
  Indexes
    PK_Bill (Clustered)
  Statistics
```

Information about indexes, primary key, foreign key.(Above)



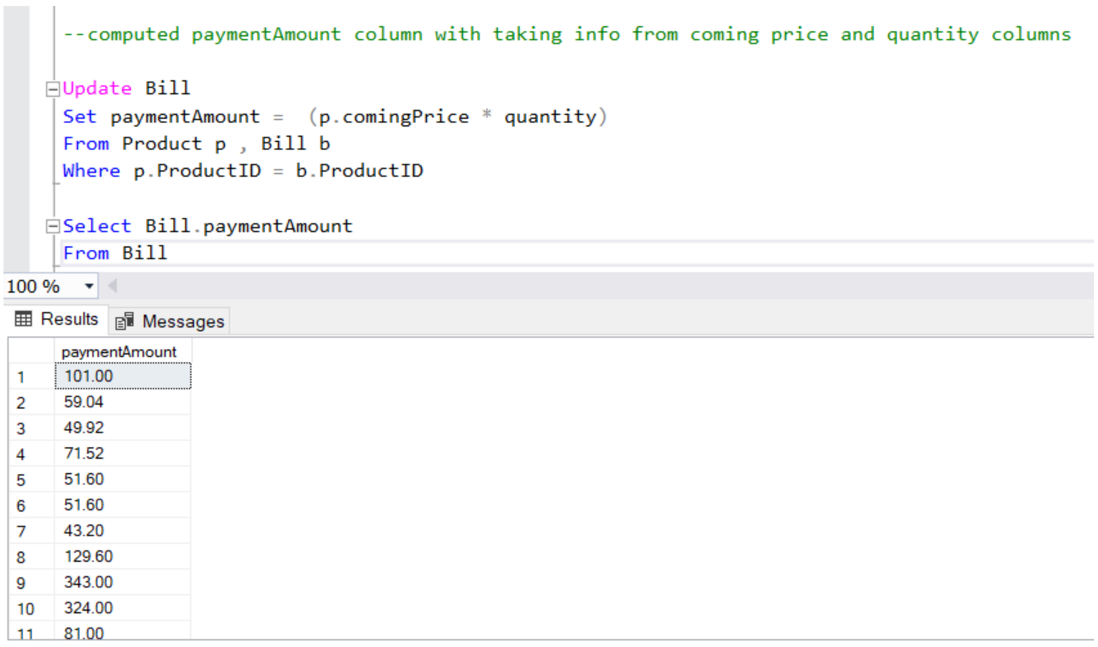
MSI.LOCALMINIMARKET - dbo.Bill | MSI.LOCALMINIMARKET -

Bill_ID	ProductID	quantity	paymentA...
1	32	20	101,00
2	16	12	59,04
2	17	24	49,92
2	18	12	71,52
3	74	12	51,60
3	75	12	51,60
3	76	24	43,20
3	77	72	129,60
4	43	20	343,00
4	44	20	324,00
4	45	5	81,00
4	46	5	76,25
4	47	10	176,30
4	48	5	83,40
4	49	5	69,10
4	50	20	305,00
4	51	10	133,40
5	90	50	716,00
5	91	20	286,40
5	92	20	286,40
5	93	10	128,90
6	39	30	371,40
6	40	100	1428,00
6	41	150	2002,50
6	42	20	247,60
7	63	12	43,20
7	64	12	90,00
7	65	12	43,20
7	66	3	27,00
7	67	2	9,00

1 of 95

Figure1:General table

In this table,salePrice is a computed column like shown below.



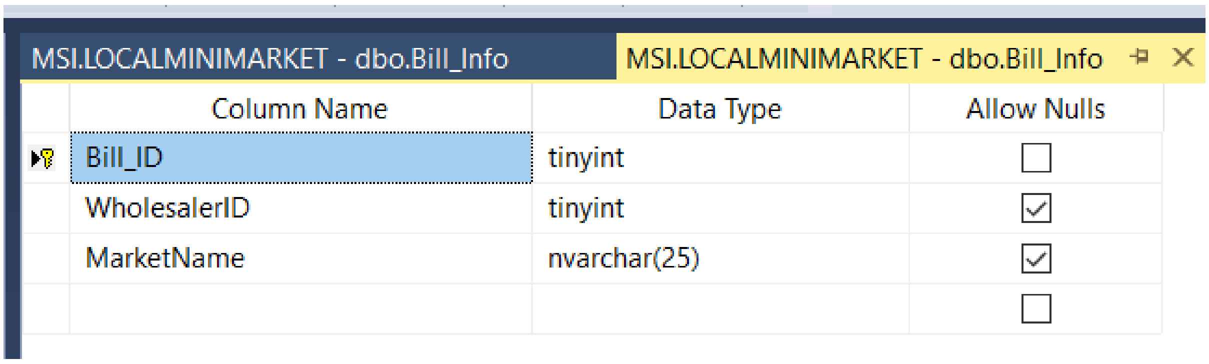
```
--computed paymentAmount column with taking info from coming price and quantity columns

Update Bill
Set paymentAmount =  (p.comingPrice * quantity)
From Product p , Bill b
Where p.ProductID = b.ProductID

Select Bill.paymentAmount
From Bill
```

100 %

Results | Messages

	paymentAmount
1	101.00
2	59.04
3	49.92
4	71.52
5	51.60
6	51.60
7	43.20
8	129.60
9	343.00
10	324.00
11	81.00

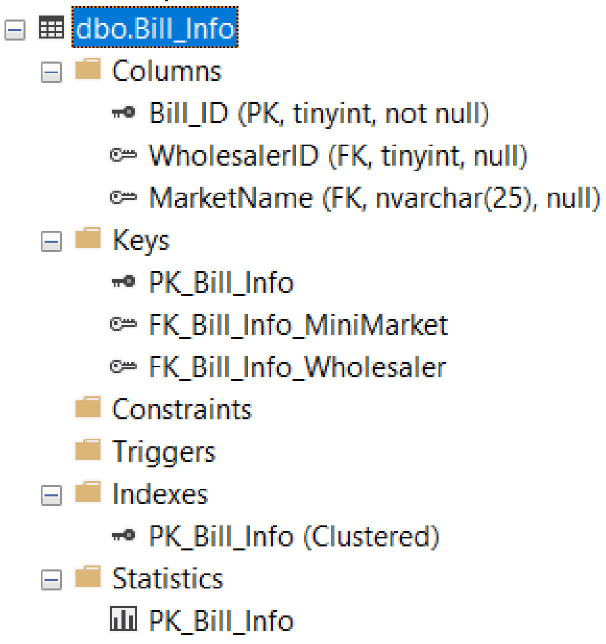
Bill_Info Table:



MSI.LOCALMINIMARKET - dbo.Bill_Info | MSI.LOCALMINIMARKET - dbo.Bill_Info

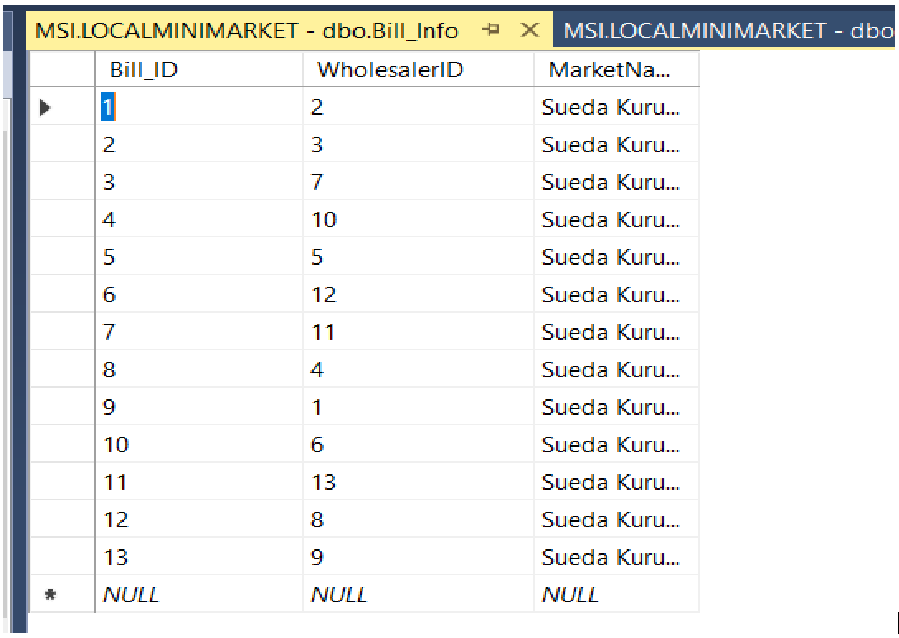
Column Name	Data Type	Allow Nulls
Bill_ID	tinyint	☐
WholesalerID	tinyint	✓
MarketName	nvarchar(25)	✓
		☐

This table keeps relation between wholesaler,market.



```
dbo.Bill_Info
  Columns
    Bill_ID (PK, tinyint, not null)
    WholesalerID (FK, tinyint, null)
    MarketName (FK, nvarchar(25), null)
  Keys
    PK_Bill_Info
    FK_Bill_Info_MiniMarket
    FK_Bill_Info_Wholesaler
  Constraints
  Triggers
  Indexes
    PK_Bill_Info (Clustered)
  Statistics
    PK_Bill_Info
```

Information about indexes, primary key, foreign key.(Above)

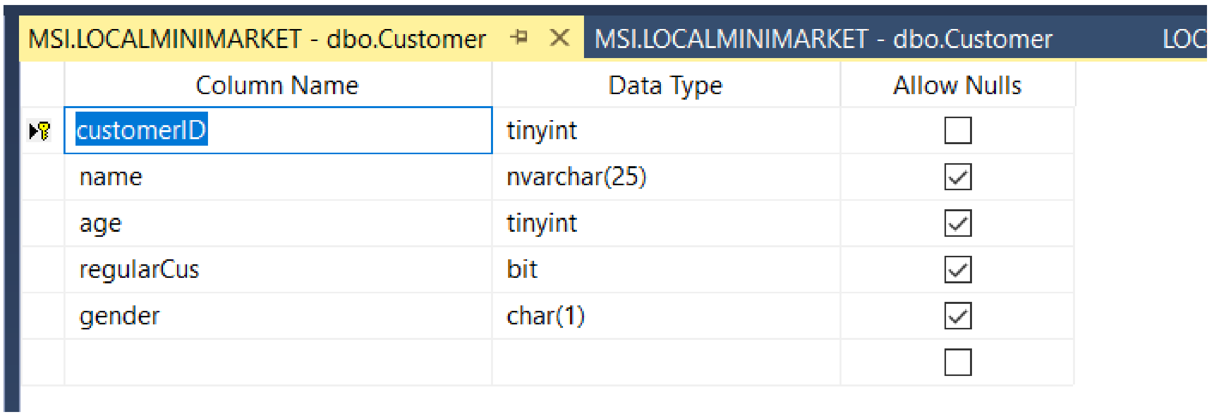


MSI.LOCALMINIMARKET - dbo.Bill_Info | MSI.LOCALMINIMARKET - dbo

Bill_ID	WholesalerID	MarketNa...
1	2	Sueda Kuru...
2	3	Sueda Kuru...
3	7	Sueda Kuru...
4	10	Sueda Kuru...
5	5	Sueda Kuru...
6	12	Sueda Kuru...
7	11	Sueda Kuru...
8	4	Sueda Kuru...
9	1	Sueda Kuru...
10	6	Sueda Kuru...
11	13	Sueda Kuru...
12	8	Sueda Kuru...
13	9	Sueda Kuru...
NULL	*NULL*	*NULL*

Figure:General Table

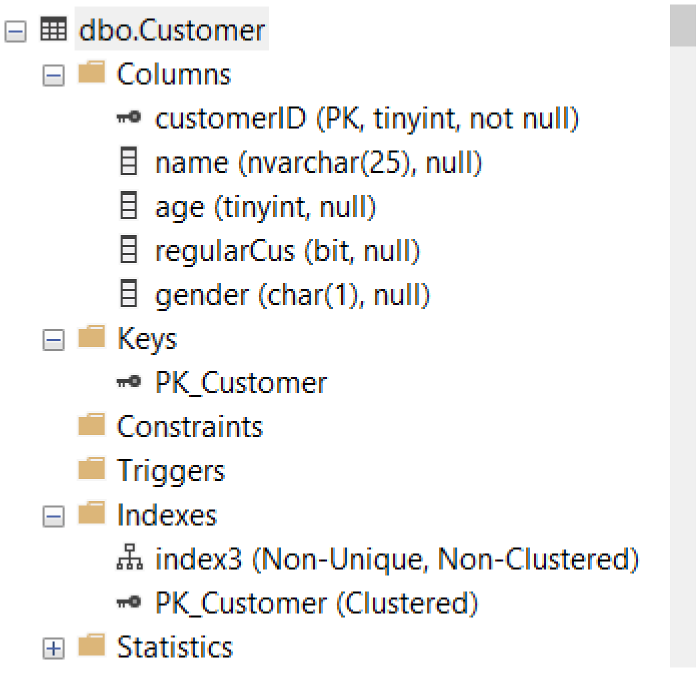
Customer Table:



MSI.LOCALMINIMARKET - dbo.Customer | MSI.LOCALMINIMARKET - dbo.Customer | LOC

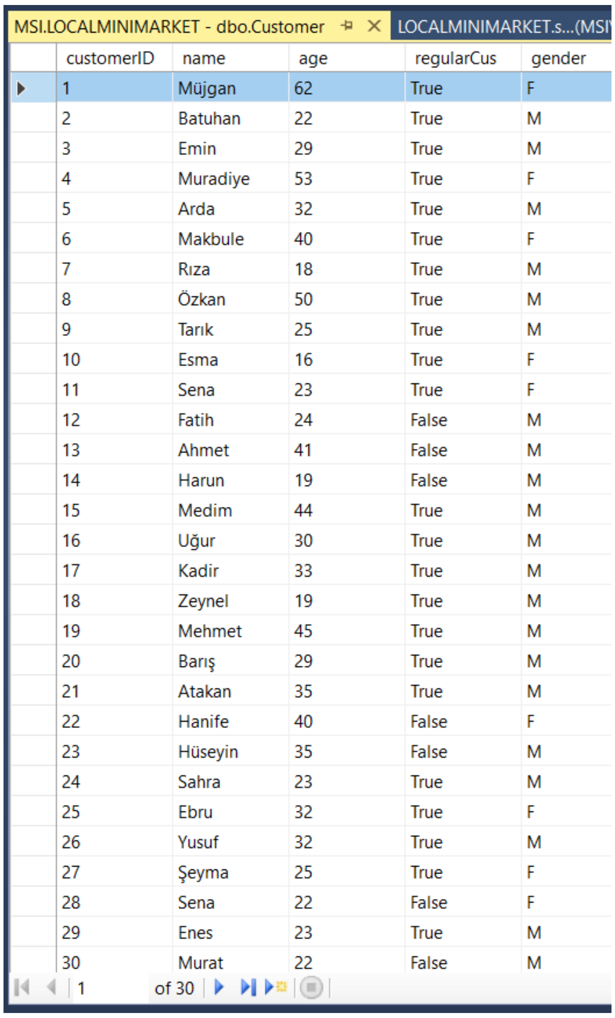
Column Name	Data Type	Allow Nulls
customerID	tinyint	☐
name	nvarchar(25)	✓
age	tinyint	✓
regularCus	bit	✓
gender	char(1)	✓
		☐

This table holds data of customer with name,age,gender values.



- dbo.Customer
  - Columns
    - customerID (PK, tinyint, not null)
    - name (nvarchar(25), null)
    - age (tinyint, null)
    - regularCus (bit, null)
    - gender (char(1), null)
  - Keys
    - PK_Customer
  - Constraints
  - Triggers
  - Indexes
    - index3 (Non-Unique, Non-Clustered)
    - PK_Customer (Clustered)
  - Statistics

Information about indexes, primary key, foreign key.(Above)

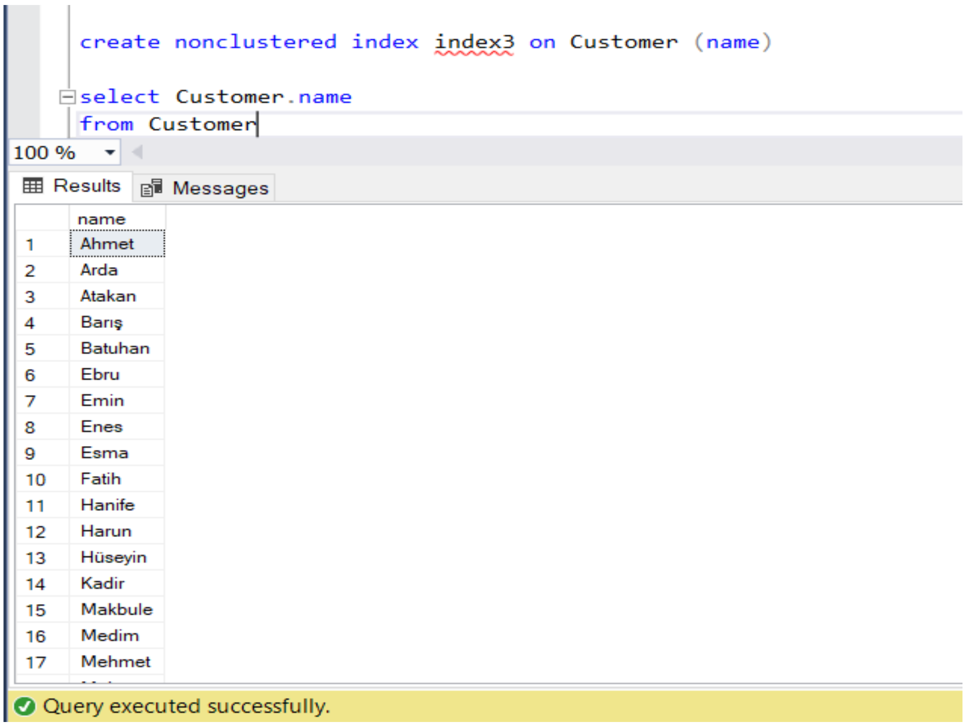

MSI.LOCALMINIMARKET - dbo.Customer LOCALMINIMARKET.s...(MSI

customerID	name	age	regularCus	gender
1	Müjgan	62	True	F
2	Batuhan	22	True	M
3	Emin	29	True	M
4	Muradiye	53	True	F
5	Arda	32	True	M
6	Makbule	40	True	F
7	Rıza	18	True	M
8	Özkan	50	True	M
9	Tarık	25	True	M
10	Esma	16	True	F
11	Sena	23	True	F
12	Fatih	24	False	M
13	Ahmet	41	False	M
14	Harun	19	False	M
15	Medim	44	True	M
16	Uğur	30	True	M
17	Kadir	33	True	M
18	Zeynel	19	True	M
19	Mehmet	45	True	M
20	Barış	29	True	M
21	Atakan	35	True	M
22	Hanife	40	False	F
23	Hüseyin	35	False	M
24	Sahra	23	True	M
25	Ebru	32	True	F
26	Yusuf	32	True	M
27	Şeyma	25	True	F
28	Sena	22	False	F
29	Enes	23	True	M
30	Murat	22	False	M

1 of 30

Figure:General Table

Nonclustured index for Customer’s name.


```
create nonclustered index index3 on Customer (name)

select Customer.name
from Customer
```

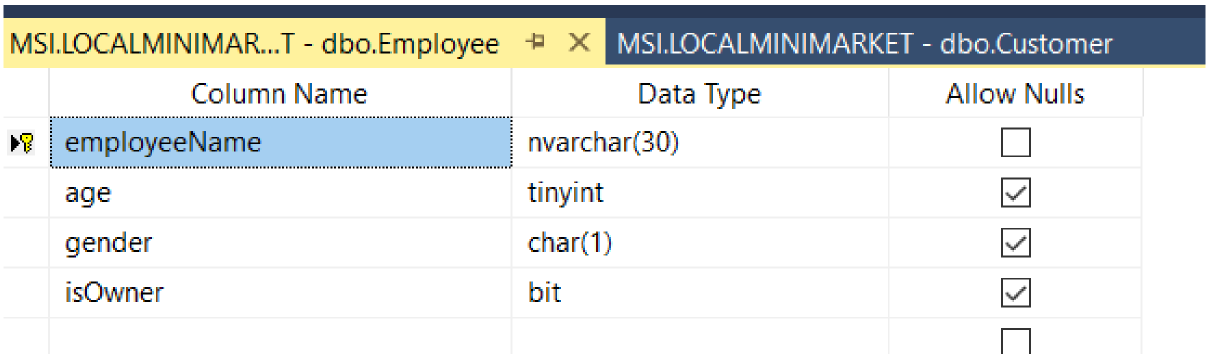
100 %

Results Messages

	name
1	Ahmet
2	Arda
3	Atakan
4	Barış
5	Batuhan
6	Ebru
7	Emin
8	Enes
9	Esma
10	Fatih
11	Hanife
12	Harun
13	Hüseyin
14	Kadir
15	Makbule
16	Medim
17	Mehmet

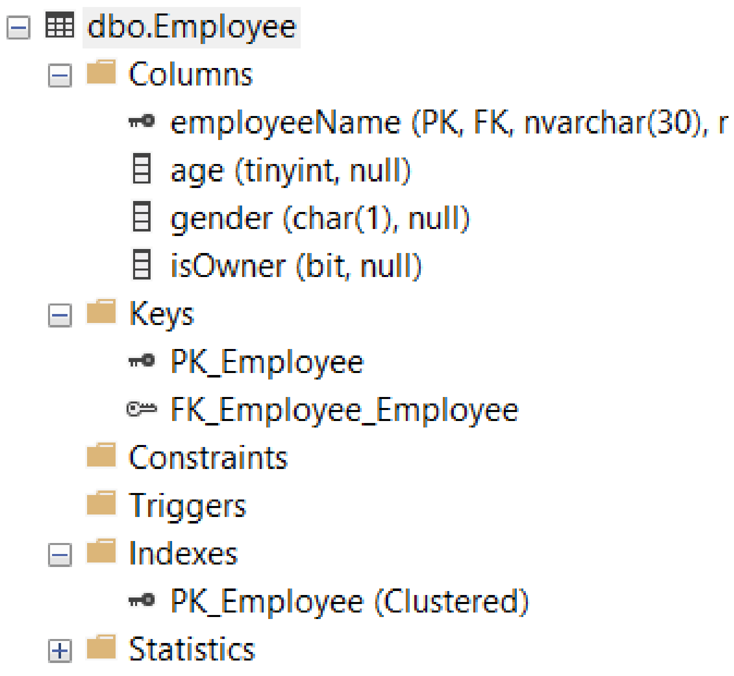
Query executed successfully.

Employee Table:

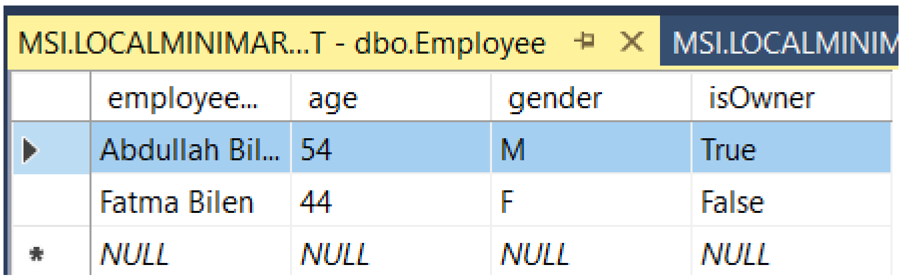

MSI.LOCALMINIMAR...T - dbo.Employee | MSI.LOCALMINIMARKET - dbo.Customer

Column Name	Data Type	Allow Nulls
employeeName	nvarchar(30)	☐
age	tinyint	✓
gender	char(1)	✓
isOwner	bit	✓
		☐

Employee table to hold informations of employees.


dbo.Employee
- Columns
  - employeeName (PK, FK, nvarchar(30), r
  - age (tinyint, null)
  - gender (char(1), null)
  - isOwner (bit, null)
- Keys
  - PK_Employee
  - FK_Employee_Employee
- Constraints
- Triggers
- Indexes
  - PK_Employee (Clustered)
- Statistics

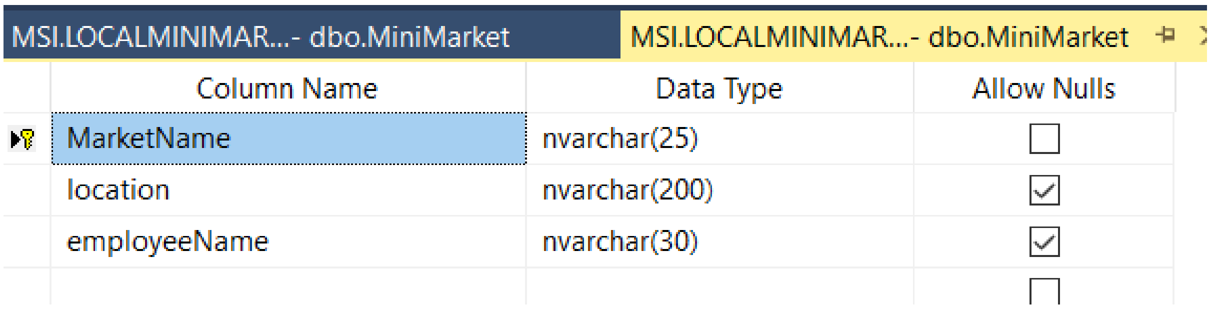
Information about indexes, primary key, foreign key.(Above)


MSI.LOCALMINIMAR...T - dbo.Employee | MSI.LOCALMINIM

	employee...	age	gender	isOwner
▶	Abdullah Bil...	54	M	True
	Fatma Bilen	44	F	False
*	*NULL*	*NULL*	*NULL*	*NULL*

Figure:General Table

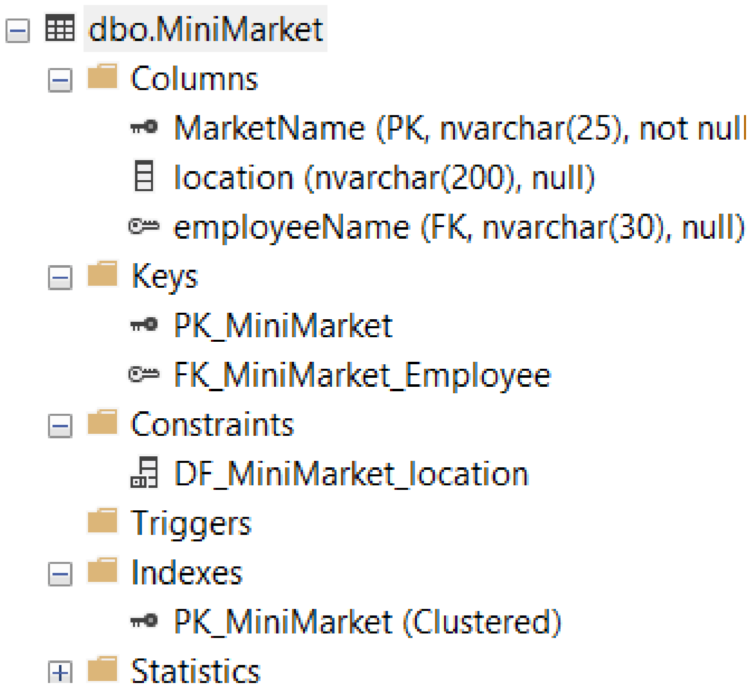
MiniMarket Table:



MSI.LOCALMINIMAR...- dbo.MiniMarket | MSI.LOCALMINIMAR...- dbo.MiniMarket

Column Name	Data Type	Allow Nulls
MarketName	nvarchar(25)	☐
location	nvarchar(200)	✓
employeeName	nvarchar(30)	✓
		☐

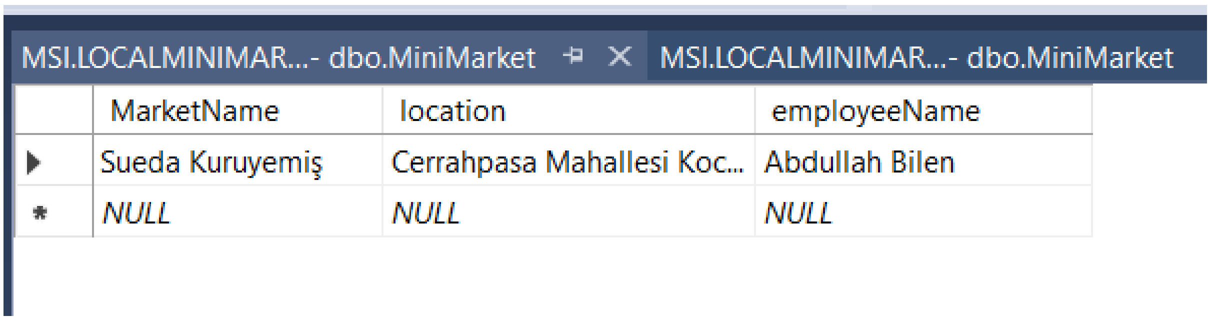
This table holds information of minimarket.



```
dbo.MiniMarket
    Columns
        MarketName (PK, nvarchar(25), not null
        location (nvarchar(200), null)
        employeeName (FK, nvarchar(30), null)
    Keys
        PK_MiniMarket
        FK_MiniMarket_Employee
    Constraints
        DF_MiniMarket_location
    Triggers
    Indexes
        PK_MiniMarket (Clustered)
    Statistics
```

Information about indexes, primary key, foreign key.(Above)

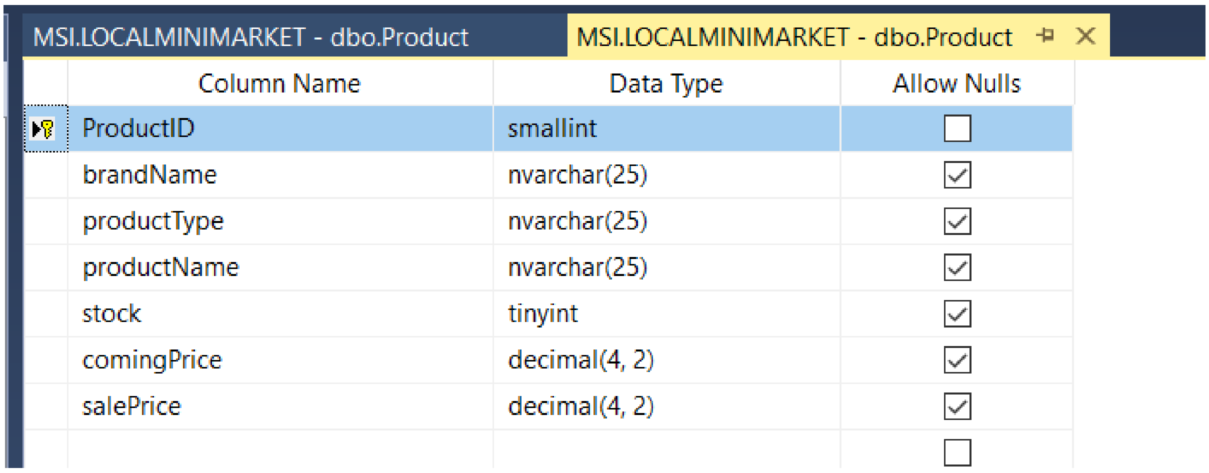
There is a default key of location also.



MSI.LOCALMINIMAR...- dbo.MiniMarket | MSI.LOCALMINIMAR...- dbo.MiniMarket

	MarketName	location	employeeName
▶	Sueda Kuruyemiş	Cerrahpasa Mahallesi Koc...	Abdullah Bilen
*	*NULL*	*NULL*	*NULL*

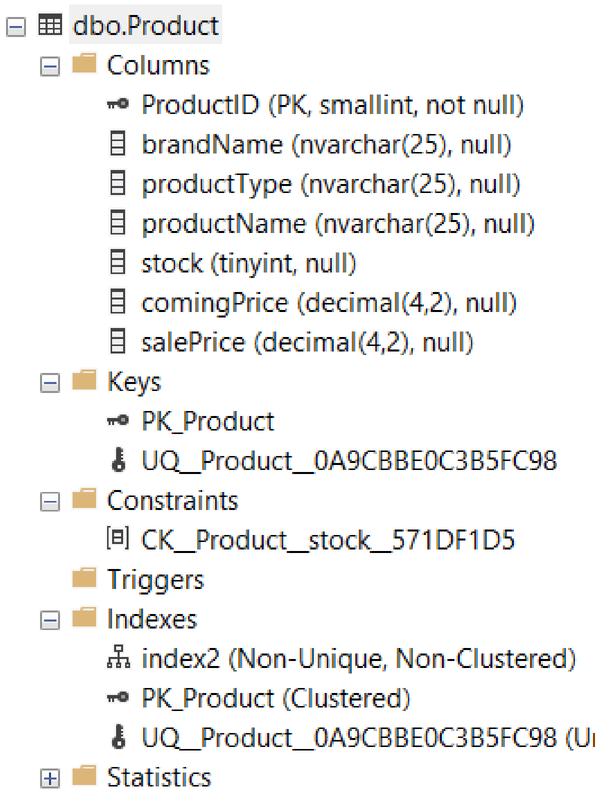
Product Table:



MSI.LOCALMINIMARKET - dbo.Product | MSI.LOCALMINIMARKET - dbo.Product

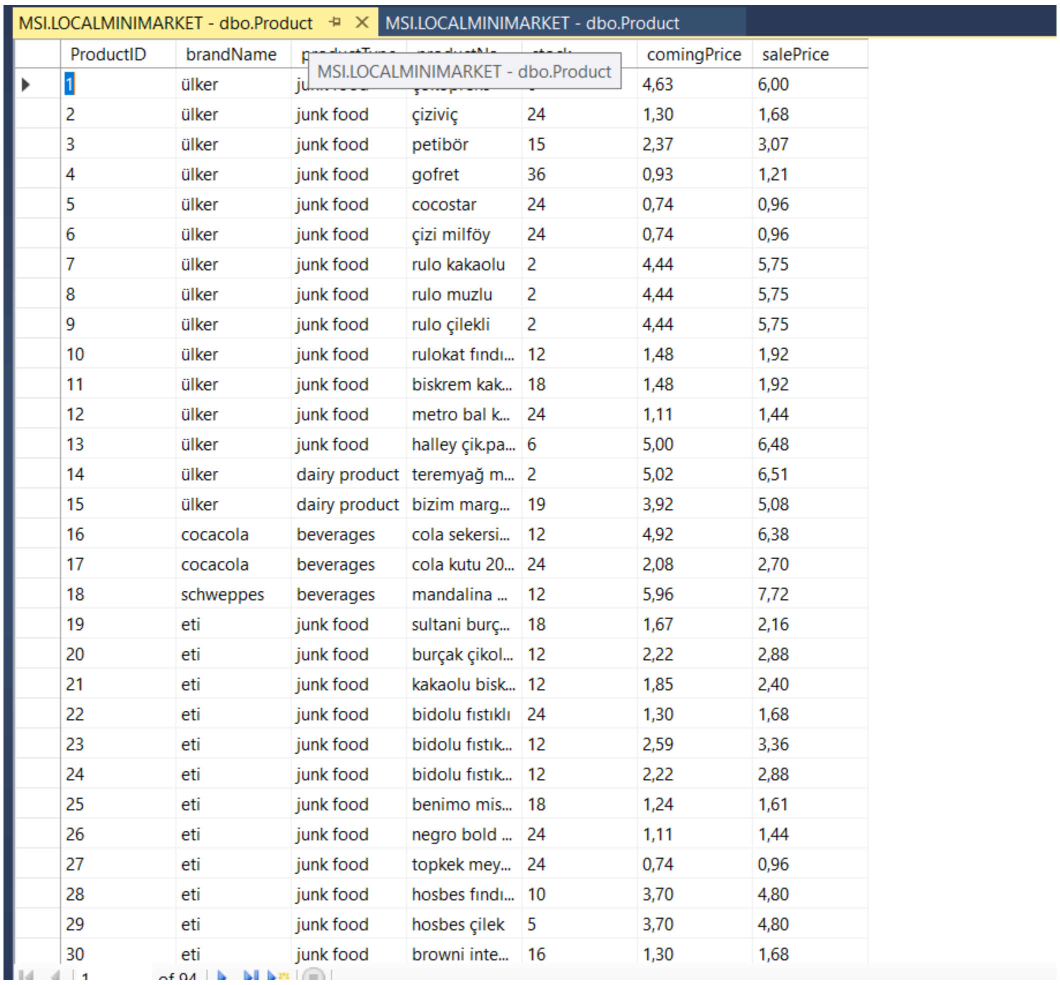
Column Name	Data Type	Allow Nulls
ProductID	smallint	☐
brandName	nvarchar(25)	✓
productType	nvarchar(25)	✓
productName	nvarchar(25)	✓
stock	tinyint	✓
comingPrice	decimal(4, 2)	✓
salePrice	decimal(4, 2)	✓
		☐

This table holds informations of products.



```
dbo.Product
  Columns
    ProductID (PK, smallint, not null)
    brandName (nvarchar(25), null)
    productType (nvarchar(25), null)
    productName (nvarchar(25), null)
    stock (tinyint, null)
    comingPrice (decimal(4,2), null)
    salePrice (decimal(4,2), null)
  Keys
    PK_Product
    UQ__Product__0A9CBBE0C3B5FC98
  Constraints
    CK__Product__stock__571DF1D5
  Triggers
  Indexes
    index2 (Non-Unique, Non-Clustered)
    PK_Product (Clustered)
    UQ__Product__0A9CBBE0C3B5FC98 (U
  Statistics
```

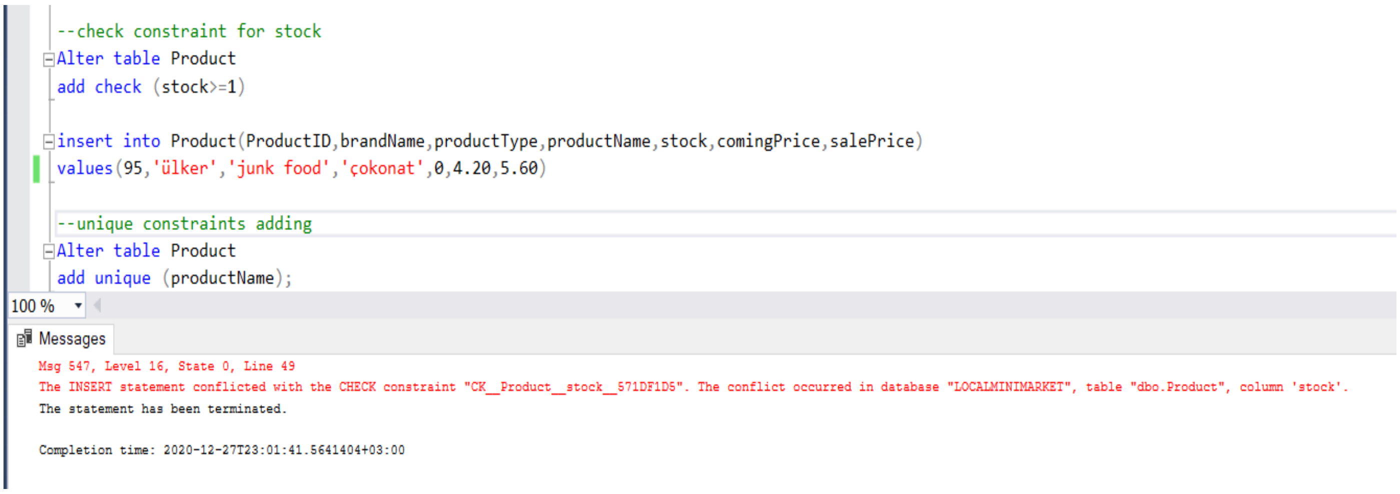
Information about indexes, primary key, foreign key.(Above)


MSI.LOCALMINIMARKET - dbo.Product | MSI.LOCALMINIMARKET - dbo.Product

	ProductID	brandName	productType	productNa...	stock	comingPrice	salePrice
▶	1	ülker	junk food	çokoprens	[illegible]	4,63	6,00
	2	ülker	junk food	çiziviç	24	1,30	1,68
	3	ülker	junk food	petibör	15	2,37	3,07
	4	ülker	junk food	gofret	36	0,93	1,21
	5	ülker	junk food	cocostar	24	0,74	0,96
	6	ülker	junk food	çizi milföy	24	0,74	0,96
	7	ülker	junk food	rulo kakaolu	2	4,44	5,75
	8	ülker	junk food	rulo muzlu	2	4,44	5,75
	9	ülker	junk food	rulo çilekli	2	4,44	5,75
	10	ülker	junk food	rulokat fındı...	12	1,48	1,92
	11	ülker	junk food	biskrem kak...	18	1,48	1,92
	12	ülker	junk food	metro bal k...	24	1,11	1,44
	13	ülker	junk food	halley çik.pa...	6	5,00	6,48
	14	ülker	dairy product	teremyağ m...	2	5,02	6,51
	15	ülker	dairy product	bizim marg...	19	3,92	5,08
	16	cocacola	beverages	cola sekersi...	12	4,92	6,38
	17	cocacola	beverages	cola kutu 20...	24	2,08	2,70
	18	schweppes	beverages	mandalina ...	12	5,96	7,72
	19	eti	junk food	sultani burç...	18	1,67	2,16
	20	eti	junk food	burçak çikol...	12	2,22	2,88
	21	eti	junk food	kakaolu bisk...	12	1,85	2,40
	22	eti	junk food	bidolu fıstıklı	24	1,30	1,68
	23	eti	junk food	bidolu fıstık...	12	2,59	3,36
	24	eti	junk food	bidolu fıstık...	12	2,22	2,88
	25	eti	junk food	benimo mis...	18	1,24	1,61
	26	eti	junk food	negro bold ...	24	1,11	1,44
	27	eti	junk food	topkek mey...	24	0,74	0,96
	28	eti	junk food	hosbes fındı...	10	3,70	4,80
	29	eti	junk food	hosbes çilek	5	3,70	4,80
	30	eti	junk food	browni inte...	16	1,30	1,68

1 of 94

Figure: General Table


```
--check constraint for stock
Alter table Product
add check (stock>=1)

insert into Product(ProductID,brandName,productType,productName,stock,comingPrice,salePrice)
values(95,'ülker','junk food','çokonat',0,4.20,5.60)

--unique constraints adding
Alter table Product
add unique (productName);
```

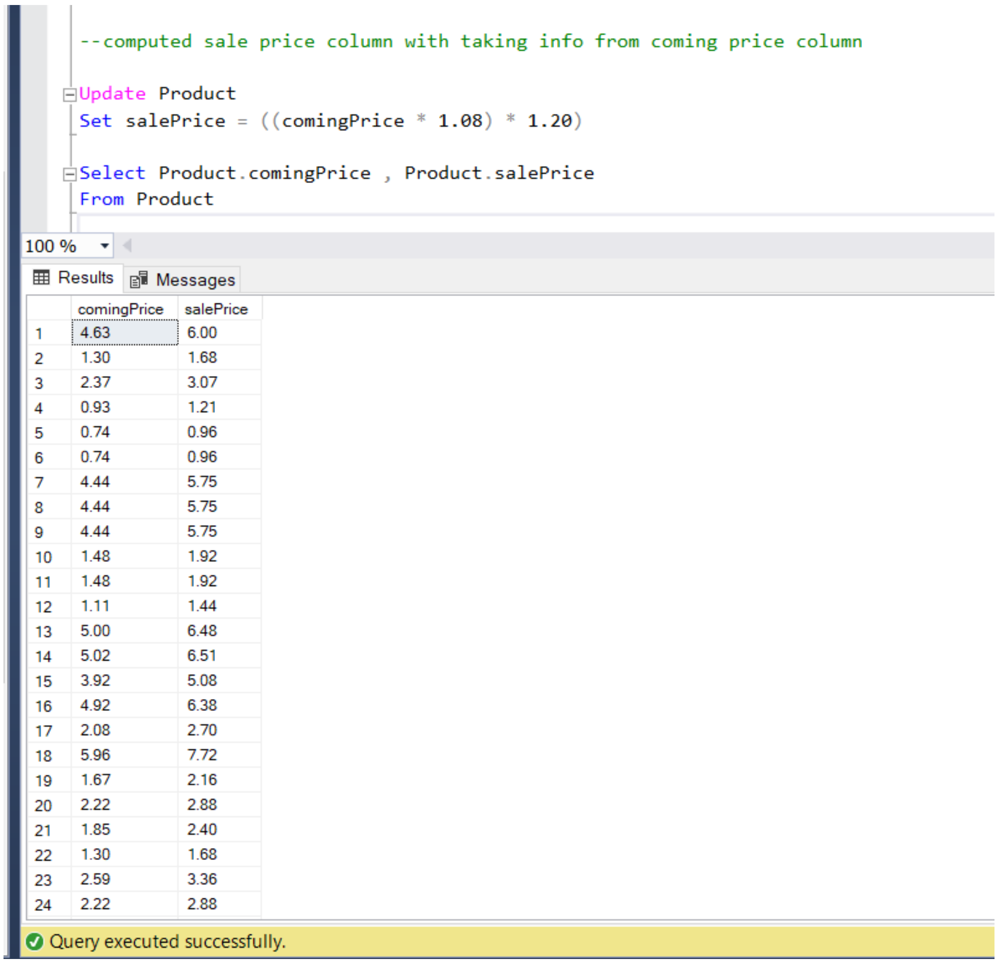
100 %

Messages

```
Msg 547, Level 16, State 0, Line 49
The INSERT statement conflicted with the CHECK constraint "CK__Product__stock__571DF1D5". The conflict occurred in database "LOCALMINIMARKET", table "dbo.Product", column 'stock'.
The statement has been terminated.

Completion time: 2020-12-27T23:01:41.5641404+03:00
```

Check constraint defined.This error occurred because we tried to insert stock value as 0,then check constaint didn’t accept it because of stock should be higher than zero condition.

Unique key added.

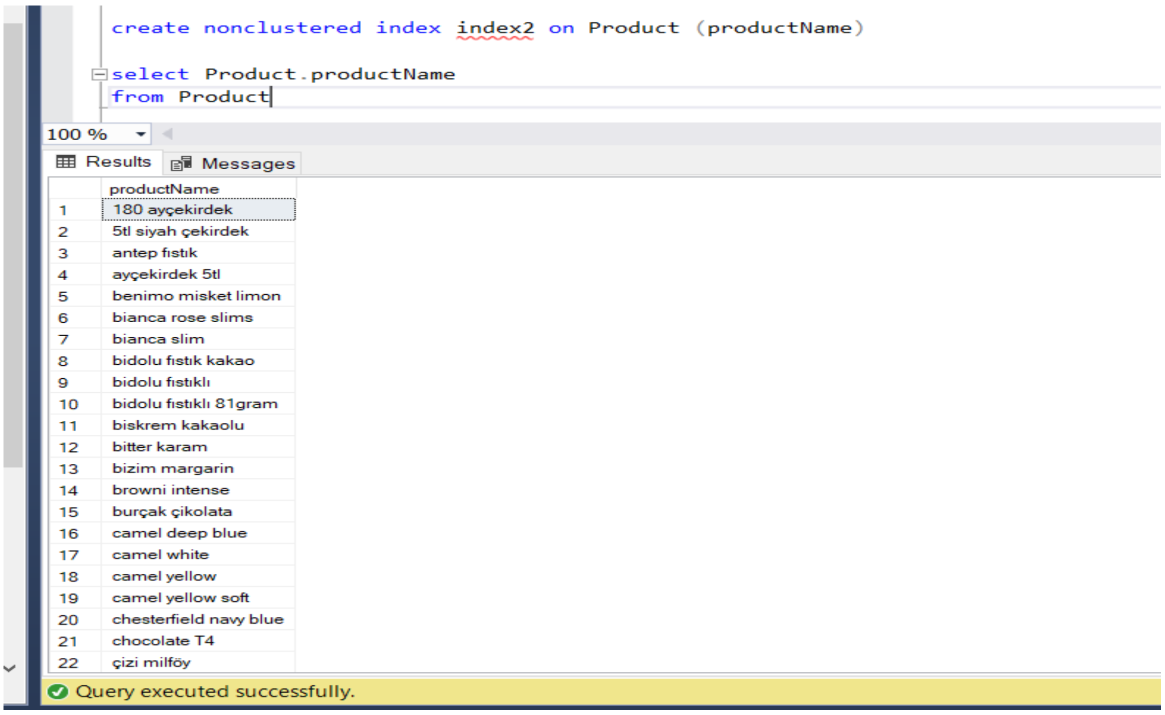


```
--computed sale price column with taking info from coming price column

Update Product
Set salePrice = ((comingPrice * 1.08) * 1.20)

Select Product.comingPrice , Product.salePrice
From Product
```

	comingPrice	salePrice
1	4.63	6.00
2	1.30	1.68
3	2.37	3.07
4	0.93	1.21
5	0.74	0.96
6	0.74	0.96
7	4.44	5.75
8	4.44	5.75
9	4.44	5.75
10	1.48	1.92
11	1.48	1.92
12	1.11	1.44
13	5.00	6.48
14	5.02	6.51
15	3.92	5.08
16	4.92	6.38
17	2.08	2.70
18	5.96	7.72
19	1.67	2.16
20	2.22	2.88
21	1.85	2.40
22	1.30	1.68
23	2.59	3.36
24	2.22	2.88

Query executed successfully.

Computed salePrice column.This column computed from comingPrice values.



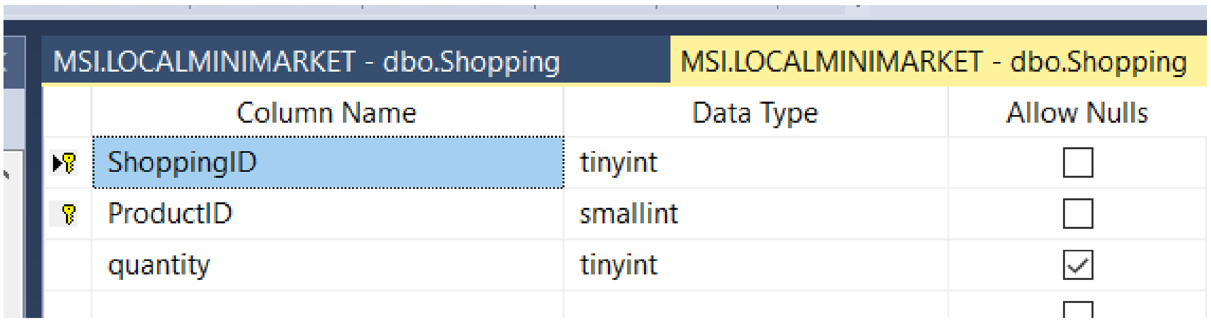
```
create nonclustered index index2 on Product (productName)

select Product.productName
from Product
```

	productName
1	180 ayçekirdek
2	5tl siyah çekirdek
3	antep fıstık
4	ayçekirdek 5tl
5	benimo misket limon
6	bianca rose slims
7	bianca slim
8	bidolu fıstık kakao
9	bidolu fıstıklı
10	bidolu fıstıklı 81gram
11	biskrem kakaolu
12	bitter karam
13	bizim margarin
14	browni intense
15	burçak çikolata
16	camel deep blue
17	camel white
18	camel yellow
19	camel yellow soft
20	chesterfield navy blue
21	chocolate T4
22	çizi milföy

Query executed successfully.

Index for productName to find easily and minimize executing time.

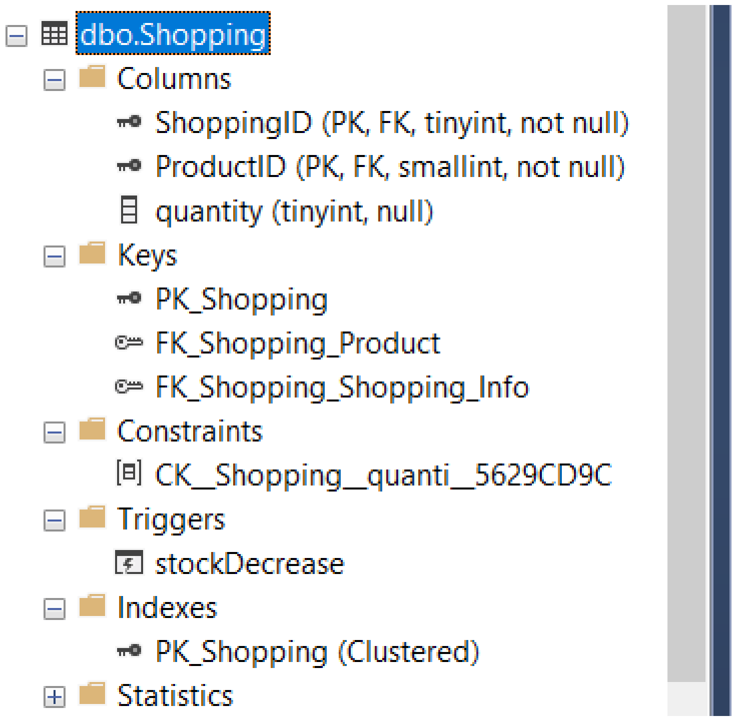
Shopping Table:



MSI.LOCALMINIMARKET - dbo.Shopping | MSI.LOCALMINIMARKET - dbo.Shopping

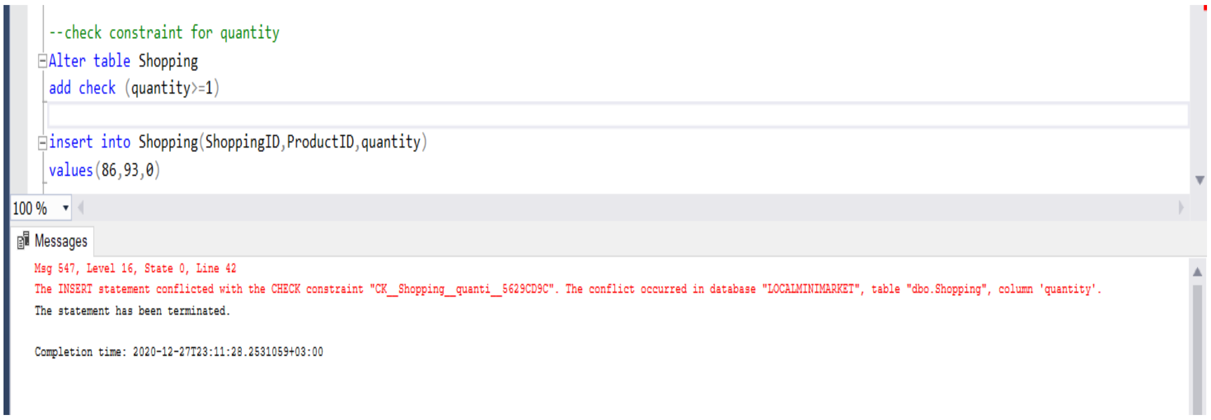
Column Name	Data Type	Allow Nulls
ShoppingID	tinyint	☐
ProductID	smallint	☐
quantity	tinyint	✓

This table keeps data of every shopping.



- dbo.Shopping
  - Columns
    - ShoppingID (PK, FK, tinyint, not null)
    - ProductID (PK, FK, smallint, not null)
    - quantity (tinyint, null)
  - Keys
    - PK_Shopping
    - FK_Shopping_Product
    - FK_Shopping_Shopping_Info
  - Constraints
    - CK__Shopping__quanti__5629CD9C
  - Triggers
    - stockDecrease
  - Indexes
    - PK_Shopping (Clustered)
  - Statistics

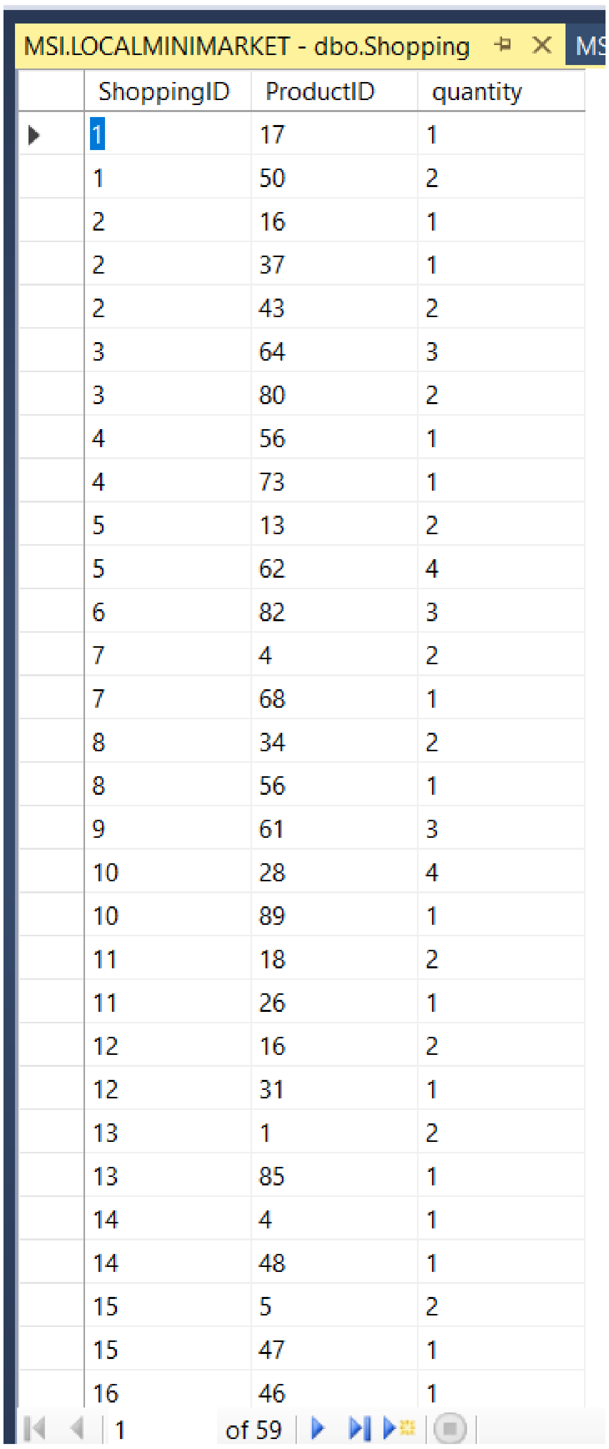
Information about indexes, primary key, foreign key.(Above)



```
--check constraint for quantity
Alter table Shopping
add check (quantity>=1)

insert into Shopping(ShoppingID,ProductID,quantity)
values(86,93,0)
```

```
Msg 547, Level 16, State 0, Line 42
The INSERT statement conflicted with the CHECK constraint "CK__Shopping__quanti__5629CD9C". The conflict occurred in database "LOCALMINIMARKET", table "dbo.Shopping", column 'quantity'.
The statement has been terminated.

Completion time: 2020-12-27T23:11:28.2531059+03:00
```

Check constraint added for checking quantity of product ,if we try to add an product with quantity equals to 0 then check constraint will give an error.

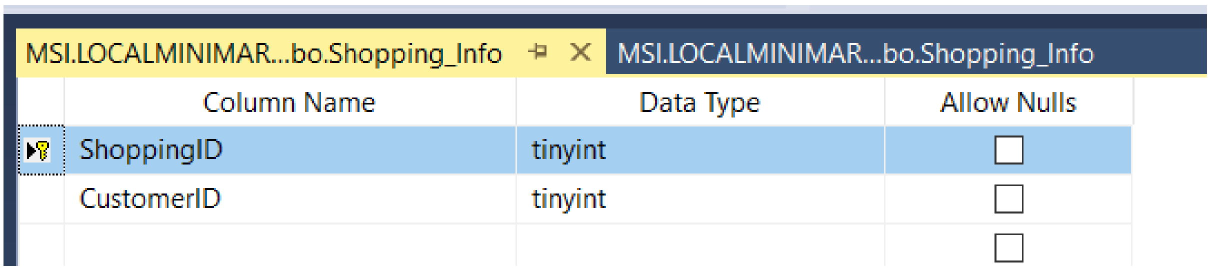

MSI.LOCALMINIMARKET - dbo.Shopping

ShoppingID	ProductID	quantity
1	17	1
1	50	2
2	16	1
2	37	1
2	43	2
3	64	3
3	80	2
4	56	1
4	73	1
5	13	2
5	62	4
6	82	3
7	4	2
7	68	1
8	34	2
8	56	1
9	61	3
10	28	4
10	89	1
11	18	2
11	26	1
12	16	2
12	31	1
13	1	2
13	85	1
14	4	1
14	48	1
15	5	2
15	47	1
16	46	1

1 of 59

Figure : General Table

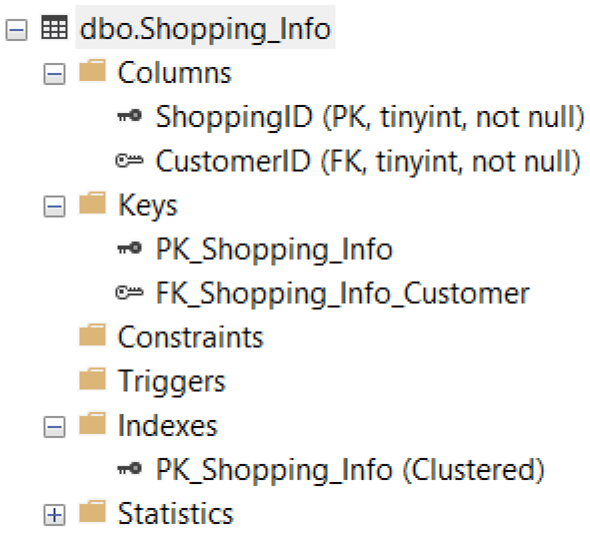
Shopping_Info Table:



MSI.LOCALMINIMAR...bo.Shopping_Info | MSI.LOCALMINIMAR...bo.Shopping_Info

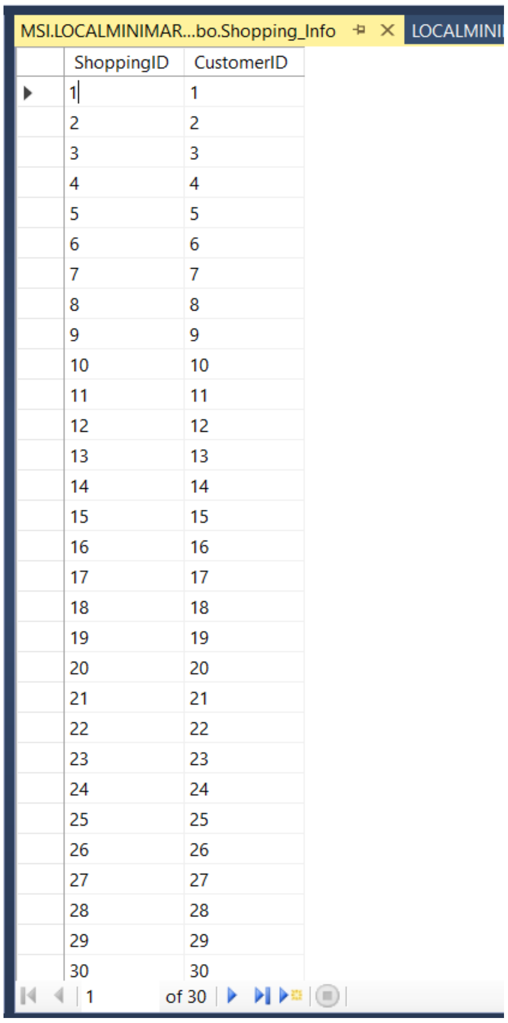
Column Name	Data Type	Allow Nulls
ShoppingID	tinyint	☐
CustomerID	tinyint	☐
		☐

This table keeps data of customer and shopping ids.



```
dbo.Shopping_Info
  Columns
    ShoppingID (PK, tinyint, not null)
    CustomerID (FK, tinyint, not null)
  Keys
    PK_Shopping_Info
    FK_Shopping_Info_Customer
  Constraints
  Triggers
  Indexes
    PK_Shopping_Info (Clustered)
  Statistics
```

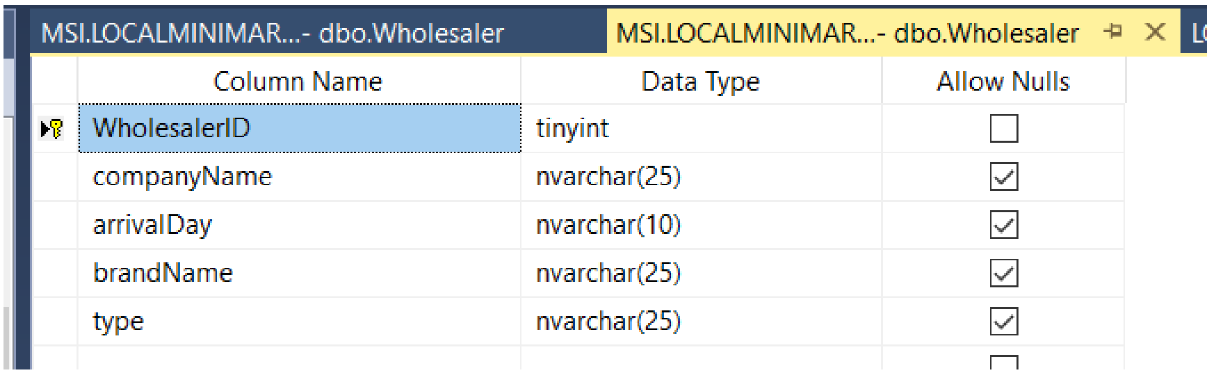
Information about indexes, primary key, foreign key.(Above)



ShoppingID	CustomerID
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30

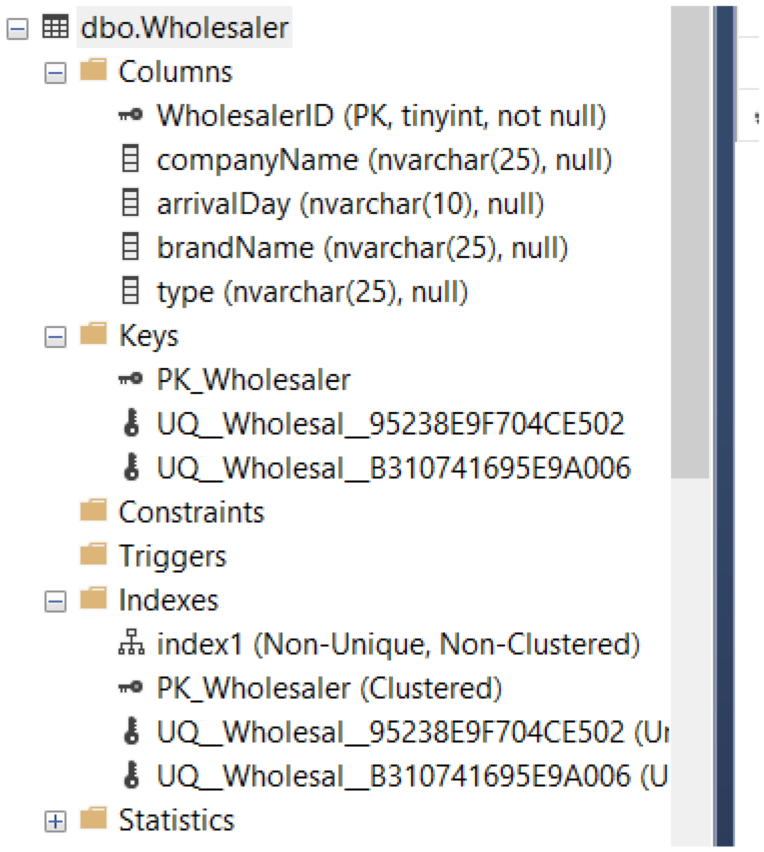
Figure : General Table

Wholesaler Table:


MSI.LOCALMINIMAR...- dbo.Wholesaler
MSI.LOCALMINIMAR...- dbo.Wholesaler

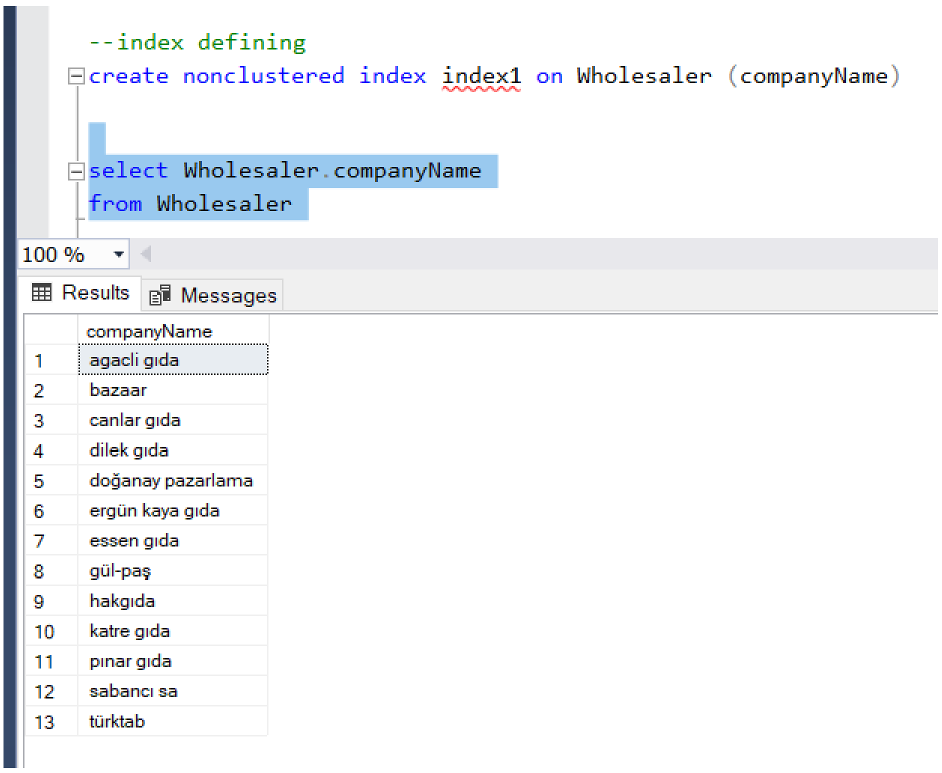
Column Name	Data Type	Allow Nulls
WholesalerID	tinyint	☐
companyName	nvarchar(25)	✓
arrivalDay	nvarchar(10)	✓
brandName	nvarchar(25)	✓
type	nvarchar(25)	✓

This table holds information of wholesalers.



```
dbo.Wholesaler
  Columns
    WholesalerID (PK, tinyint, not null)
    companyName (nvarchar(25), null)
    arrivalDay (nvarchar(10), null)
    brandName (nvarchar(25), null)
    type (nvarchar(25), null)
  Keys
    PK_Wholesaler
    UQ__Wholesal__95238E9F704CE502
    UQ__Wholesal__B310741695E9A006
  Constraints
  Triggers
  Indexes
    index1 (Non-Unique, Non-Clustered)
    PK_Wholesaler (Clustered)
    UQ__Wholesal__95238E9F704CE502 (Ur
    UQ__Wholesal__B310741695E9A006 (U
  Statistics
```

Information about indexes, primary key, foreign key.(Above)



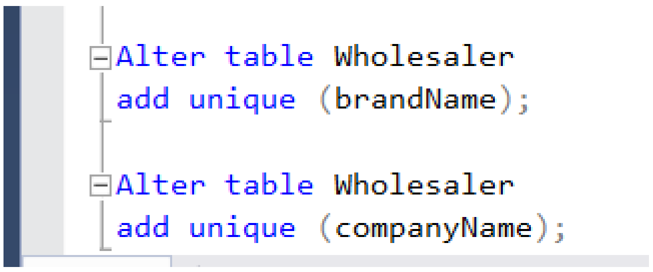
```
--index defining
create nonclustered index index1 on Wholesaler (companyName)

select Wholesaler.companyName
from Wholesaler
```

100 %

Results | Messages

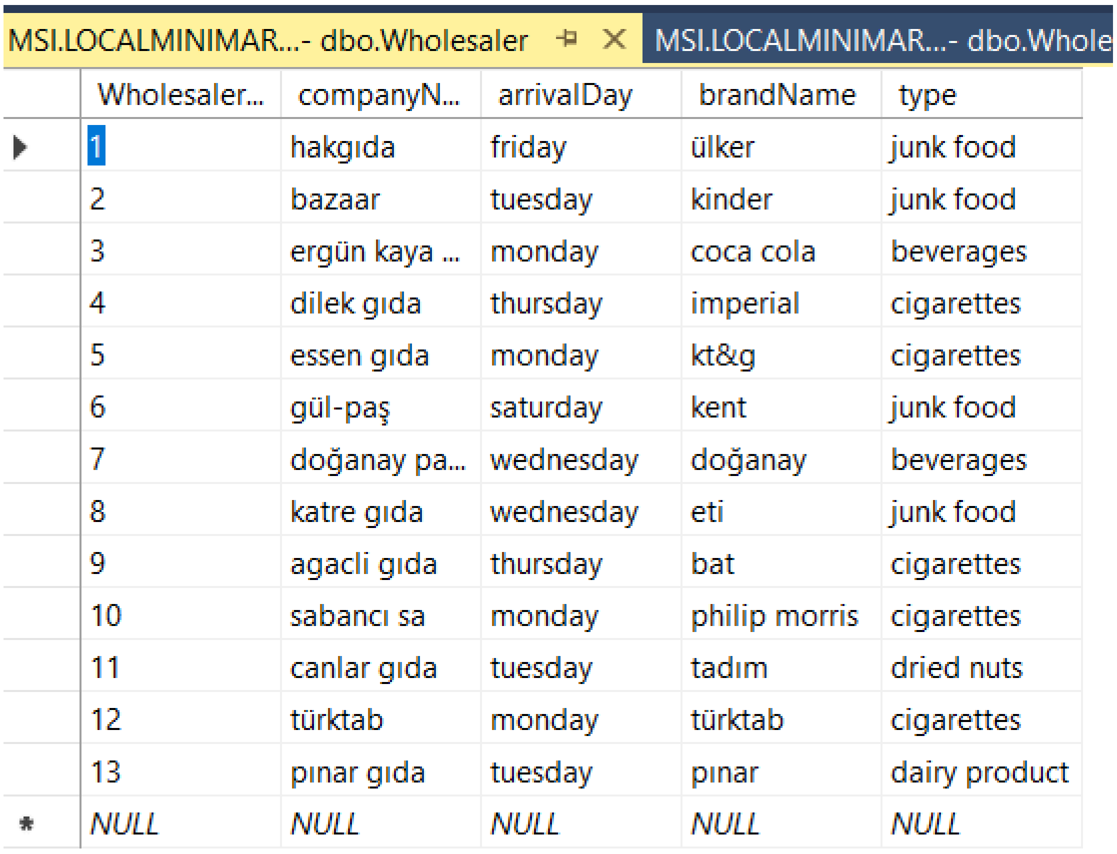
	companyName
1	agacli gıda
2	bazaar
3	canlar gıda
4	dilek gıda
5	doğanay pazarlama
6	ergün kaya gıda
7	essen gıda
8	gül-paş
9	hakgıda
10	katre gıda
11	pınar gıda
12	sabancı sa
13	türktab

Nonclustured Index for companyName.To reach easily and minimizing execute time.



```
Alter table Wholesaler
add unique (brandName);

Alter table Wholesaler
add unique (companyName);
```

Unique key added.

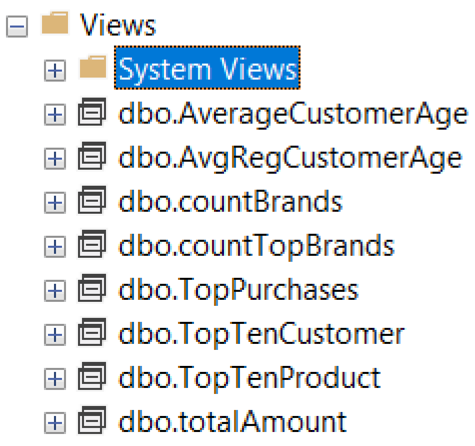


MSI.LOCALMINIMAR...- dbo.Wholesaler | MSI.LOCALMINIMAR...- dbo.Whole

Wholesaler...	companyN...	arrivalDay	brandName	type
1	hakgıda	friday	ülker	junk food
2	bazaar	tuesday	kinder	junk food
3	ergün kaya ...	monday	coca cola	beverages
4	dilek gıda	thursday	imperial	cigarettes
5	essen gıda	monday	kt&g	cigarettes
6	gül-paş	saturday	kent	junk food
7	doğanay pa...	wednesday	doğanay	beverages
8	katre gıda	wednesday	eti	junk food
9	agacli gıda	thursday	bat	cigarettes
10	sabancı sa	monday	philip morris	cigarettes
11	canlar gıda	tuesday	tadım	dried nuts
12	türktab	monday	türktab	cigarettes
13	pınar gıda	tuesday	pınar	dairy product
NULL	NULL	NULL	NULL	NULL

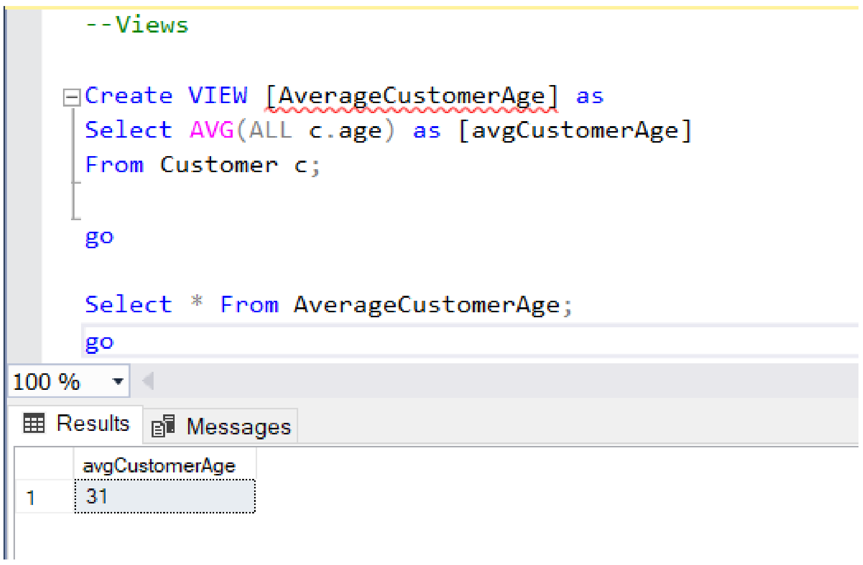
Figure : General Table

Views:



```
Views
    System Views
    dbo.AverageCustomerAge
    dbo.AvgRegCustomerAge
    dbo.countBrands
    dbo.countTopBrands
    dbo.TopPurchases
    dbo.TopTenCustomer
    dbo.TopTenProduct
    dbo.totalAmount
```

1.AverageCustomerAge:

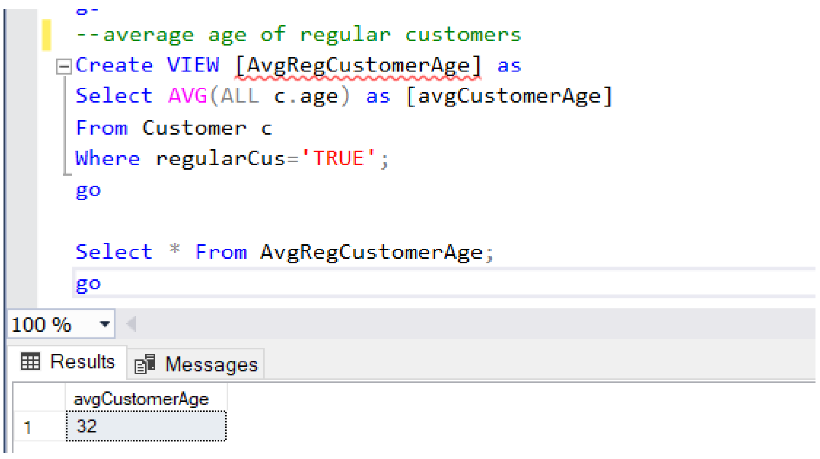


```
--Views

Create VIEW [AverageCustomerAge] as
Select AVG(ALL c.age) as [avgCustomerAge]
From Customer c;

go

Select * From AverageCustomerAge;
go
```

```
   avgCustomerAge
1  31
```

This view calculates average age of all customers.

2.AvgRegCustomerAge:

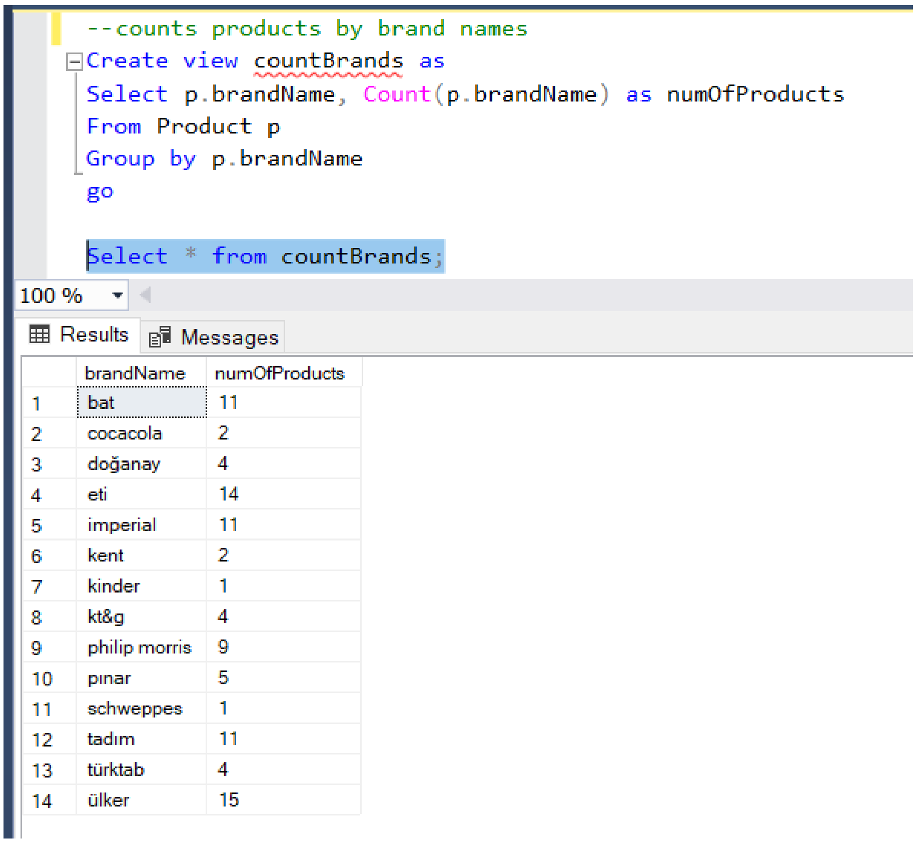


```
--average age of regular customers
Create VIEW [AvgRegCustomerAge] as
Select AVG(ALL c.age) as [avgCustomerAge]
From Customer c
Where regularCus='TRUE';
go

Select * From AvgRegCustomerAge;
go
```

```
   avgCustomerAge
1  32
```

This view calculates average age of regular customers.

3.countBrands:

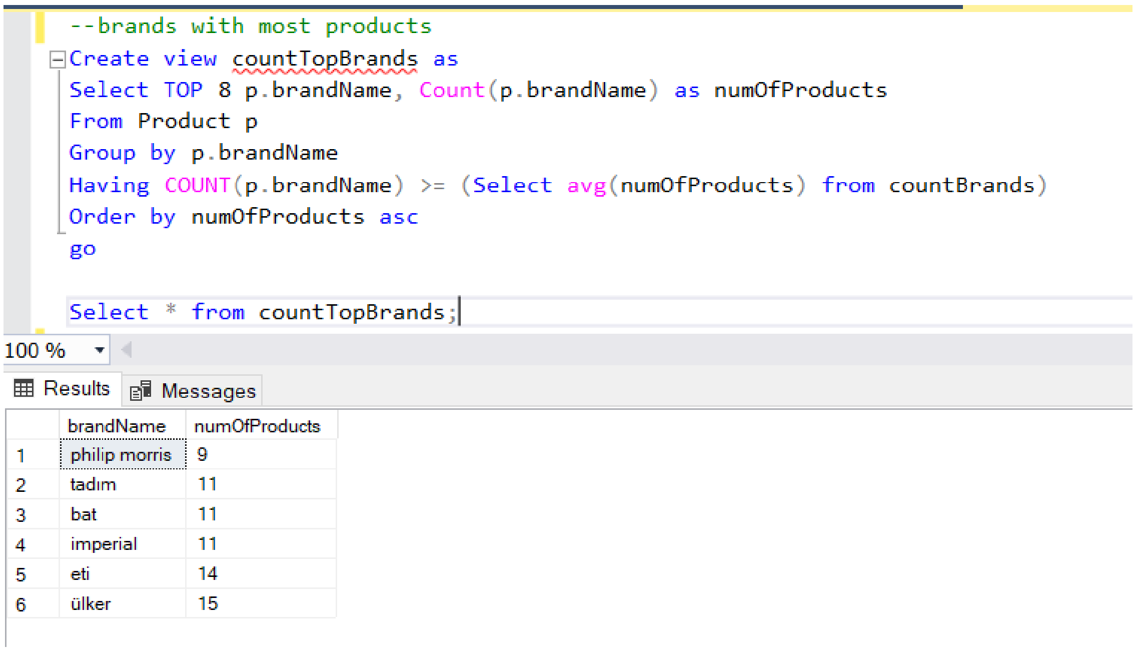


```
--counts products by brand names
Create view countBrands as
Select p.brandName, Count(p.brandName) as numOfProducts
From Product p
Group by p.brandName
go

Select * from countBrands;
```

	brandName	numOfProducts
1	bat	11
2	cocacola	2
3	doğanay	4
4	eti	14
5	imperial	11
6	kent	2
7	kinder	1
8	kt&g	4
9	philip morris	9
10	pınar	5
11	schweppes	1
12	tadım	11
13	türktab	4
14	ülker	15

This view counts products by their brand names.

4.countTopBrands

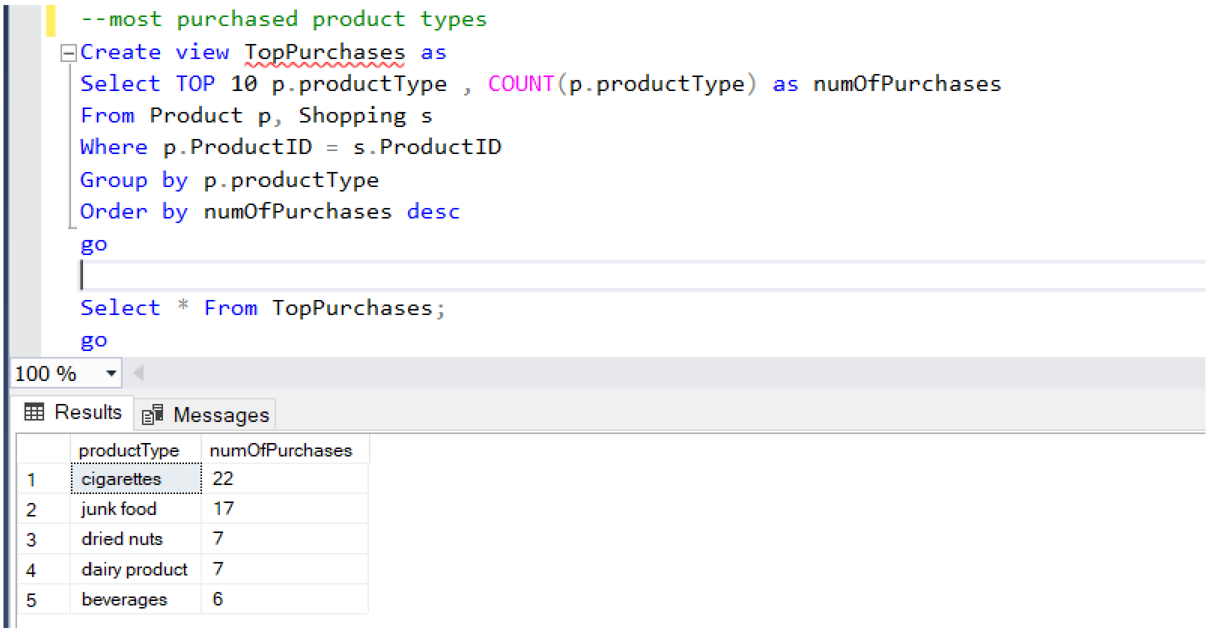


```
--brands with most products
Create view countTopBrands as
Select TOP 8 p.brandName, Count(p.brandName) as numOfProducts
From Product p
Group by p.brandName
Having COUNT(p.brandName) >= (Select avg(numOfProducts) from countBrands)
Order by numOfProducts asc
go

Select * from countTopBrands;
```

	brandName	numOfProducts
1	philip morris	9
2	tadım	11
3	bat	11
4	imperial	11
5	eti	14
6	ülker	15

This view counts brands with the most products.

5.TopPurchases:

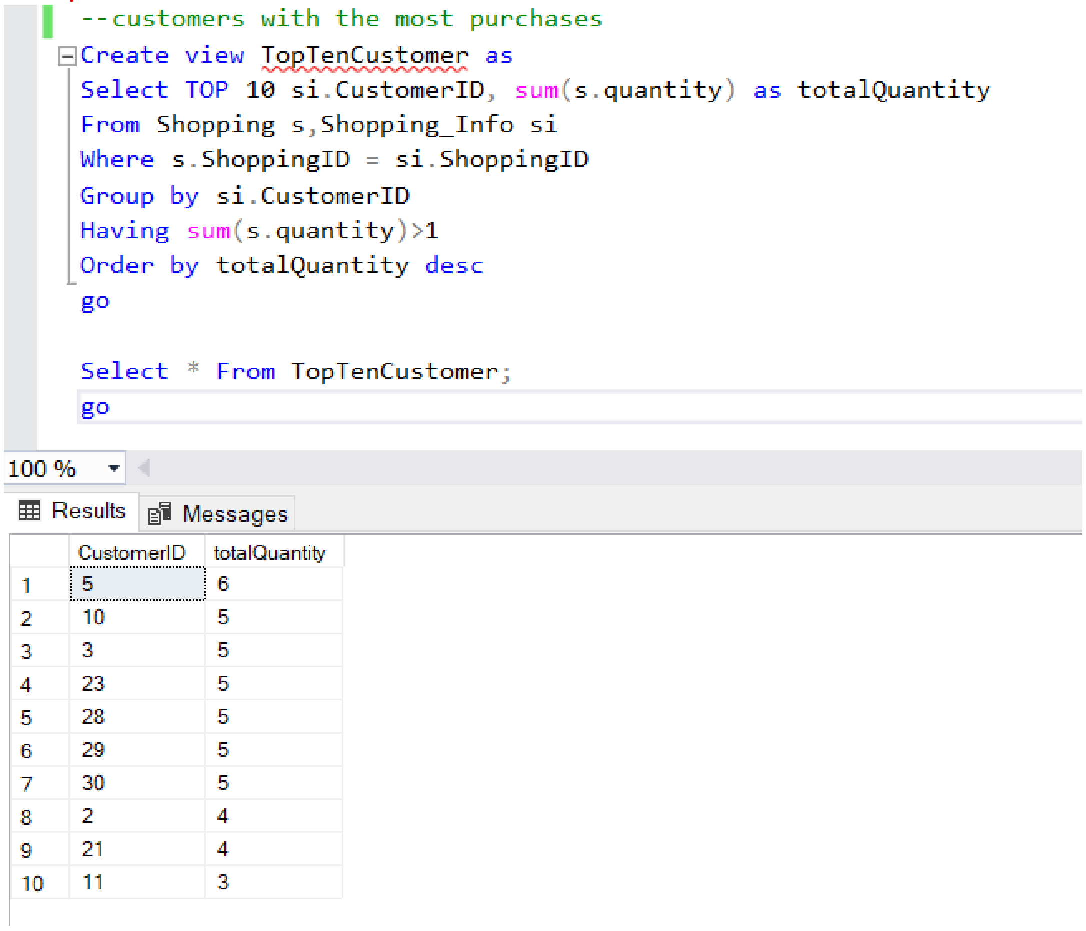


```
--most purchased product types
Create view TopPurchases as
Select TOP 10 p.productType , COUNT(p.productType) as numOfPurchases
From Product p, Shopping s
Where p.ProductID = s.ProductID
Group by p.productType
Order by numOfPurchases desc
go

Select * From TopPurchases;
go
```

	productType	numOfPurchases
1	cigarettes	22
2	junk food	17
3	dried nuts	7
4	dairy product	7
5	beverages	6

This view shows most purchased product types.

6.TopTenCustomer:

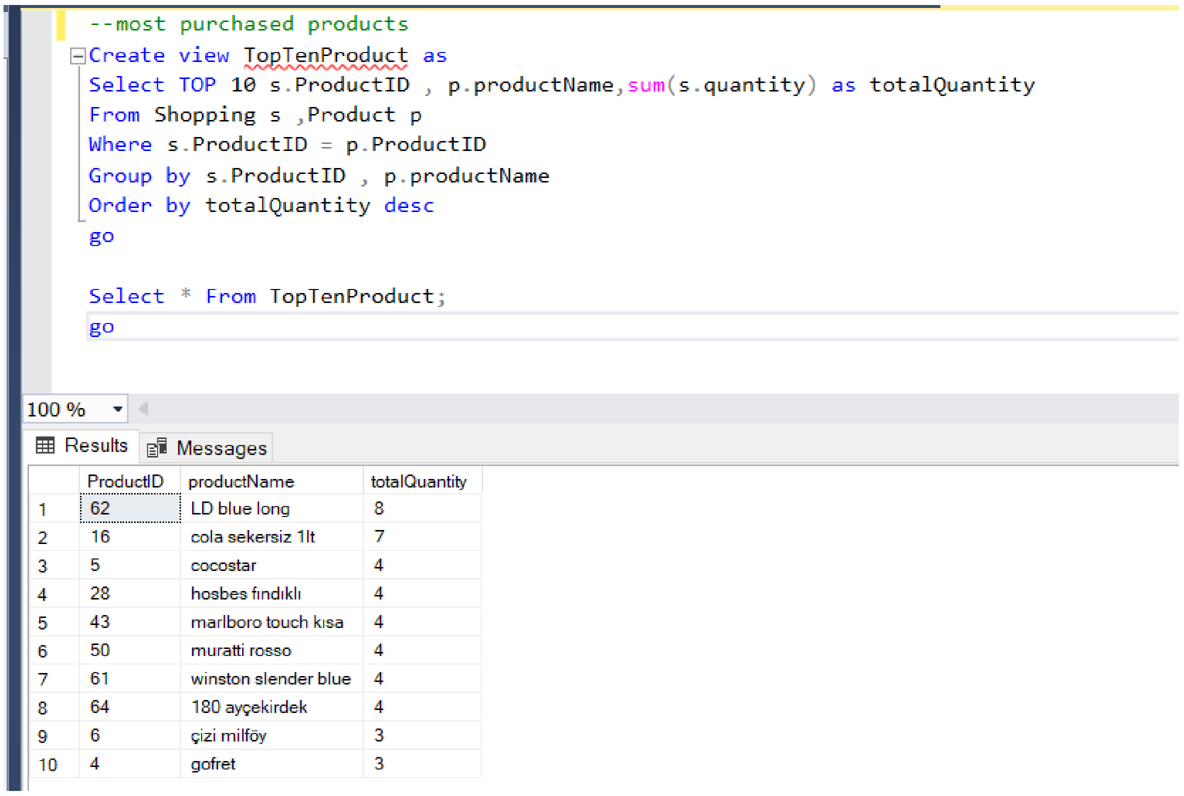


```
--customers with the most purchases
Create view TopTenCustomer as
Select TOP 10 si.CustomerID, sum(s.quantity) as totalQuantity
From Shopping s,Shopping_Info si
Where s.ShoppingID = si.ShoppingID
Group by si.CustomerID
Having sum(s.quantity)>1
Order by totalQuantity desc
go

Select * From TopTenCustomer;
go
```

	CustomerID	totalQuantity
1	5	6
2	10	5
3	3	5
4	23	5
5	28	5
6	29	5
7	30	5
8	2	4
9	21	4
10	11	3

This view shows customers with the most purchases.

7.TopTenProduct:



```
--most purchased products
Create view TopTenProduct as
Select TOP 10 s.ProductID , p.productName,sum(s.quantity) as totalQuantity
From Shopping s ,Product p
Where s.ProductID = p.ProductID
Group by s.ProductID , p.productName
Order by totalQuantity desc
go

Select * From TopTenProduct;
go
```

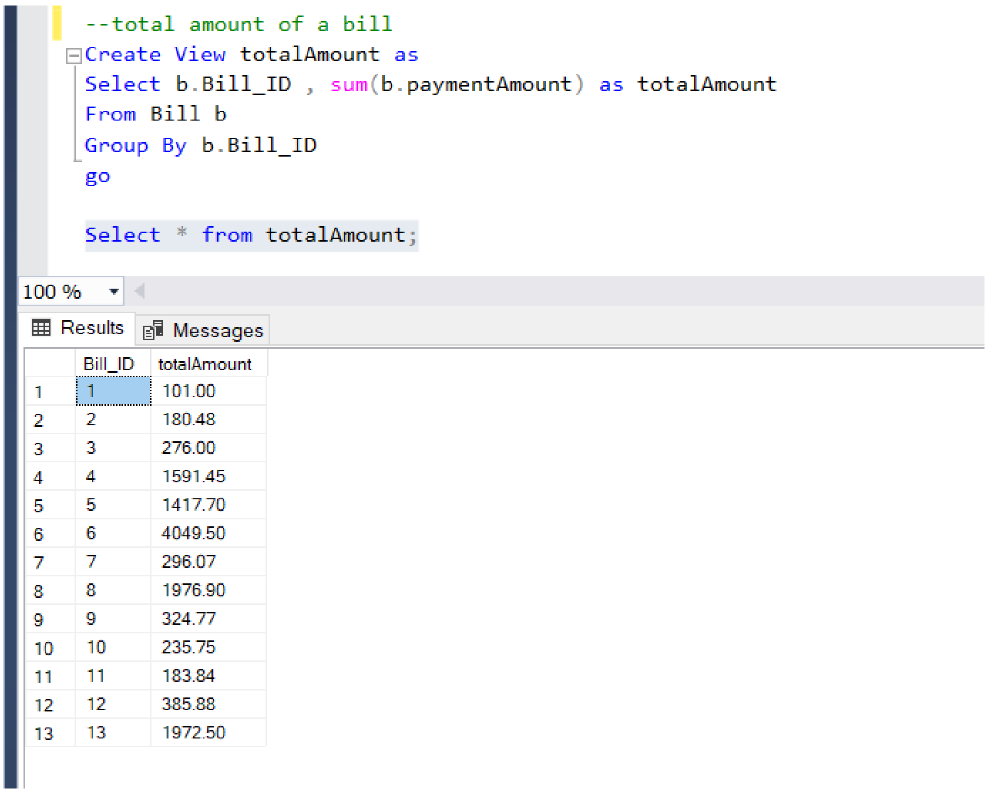
100 %

Results | Messages

	ProductID	productName	totalQuantity
1	62	LD blue long	8
2	16	cola sekersiz 1lt	7
3	5	cocostar	4
4	28	hosbes fındıklı	4
5	43	marlboro touch kısa	4
6	50	muratti rosso	4
7	61	winston slender blue	4
8	64	180 ayçekirdek	4
9	6	çizi milföy	3
10	4	gofret	3

This view shows most purchased products.

8.TotalAmount



```
--total amount of a bill
Create View totalAmount as
Select b.Bill_ID , sum(b.paymentAmount) as totalAmount
From Bill b
Group By b.Bill_ID
go

Select * from totalAmount;
```

100 %

Results | Messages

	Bill_ID	totalAmount
1	1	101.00
2	2	180.48
3	3	276.00
4	4	1591.45
5	5	1417.70
6	6	4049.50
7	7	296.07
8	8	1976.90
9	9	324.77
10	10	235.75
11	11	183.84
12	12	385.88
13	13	1972.50

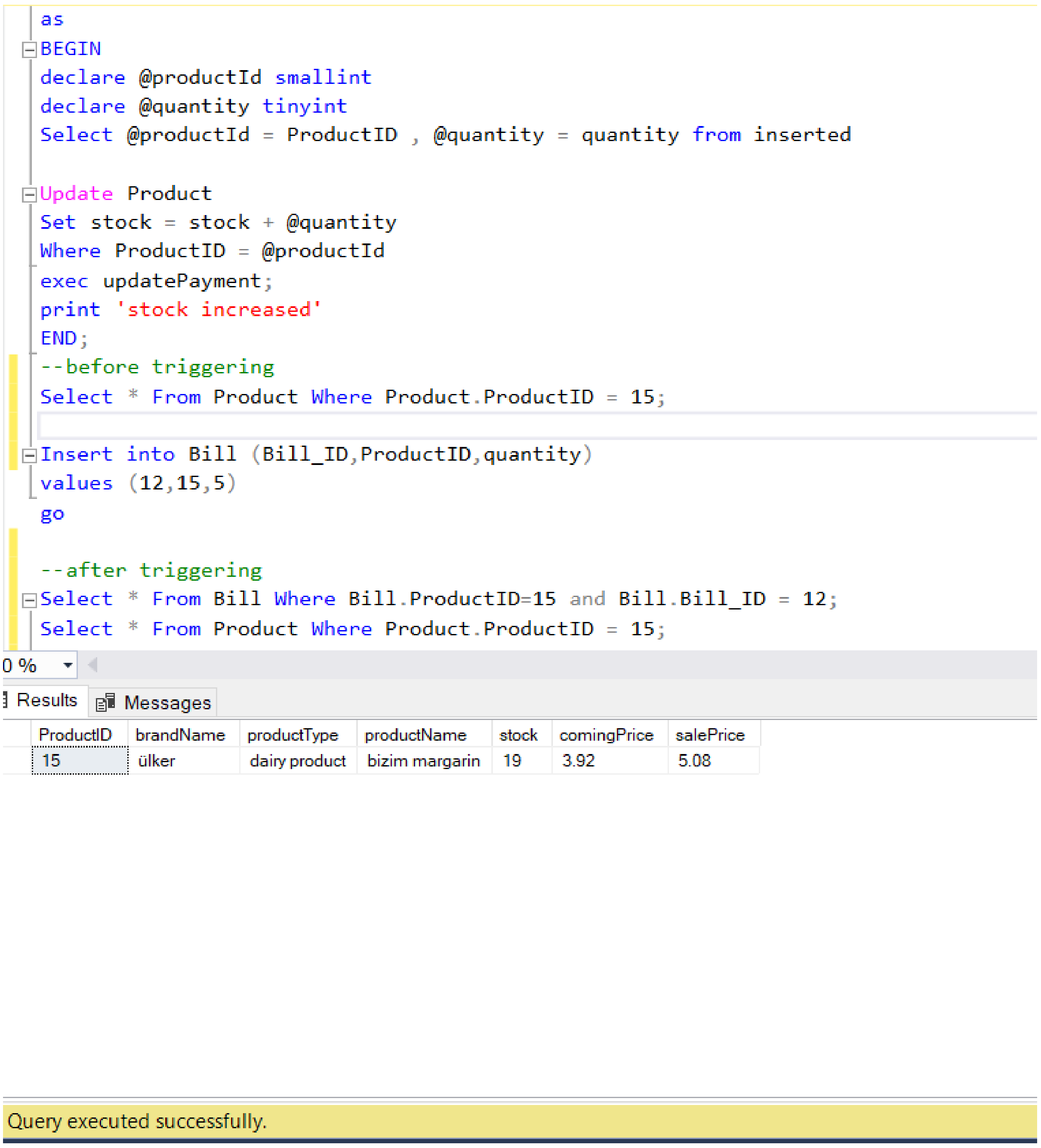
This view calculates total amount of a bill.

Triggers:

1.stockIncrease

When minimarket purchases products from wholesaler,these product purchases are inserting to Bill table.This trigger works when a purchase happened and it increases the stock of that product with ordered quantity.

Before triggering:



```
as
BEGIN
declare @productId smallint
declare @quantity tinyint
Select @productId = ProductID , @quantity = quantity from inserted

Update Product
Set stock = stock + @quantity
Where ProductID = @productId
exec updatePayment;
print 'stock increased'
END;
--before triggering
Select * From Product Where Product.ProductID = 15;

Insert into Bill (Bill_ID,ProductID,quantity)
values (12,15,5)
go

--after triggering
Select * From Bill Where Bill.ProductID=15 and Bill.Bill_ID = 12;
Select * From Product Where Product.ProductID = 15;
```

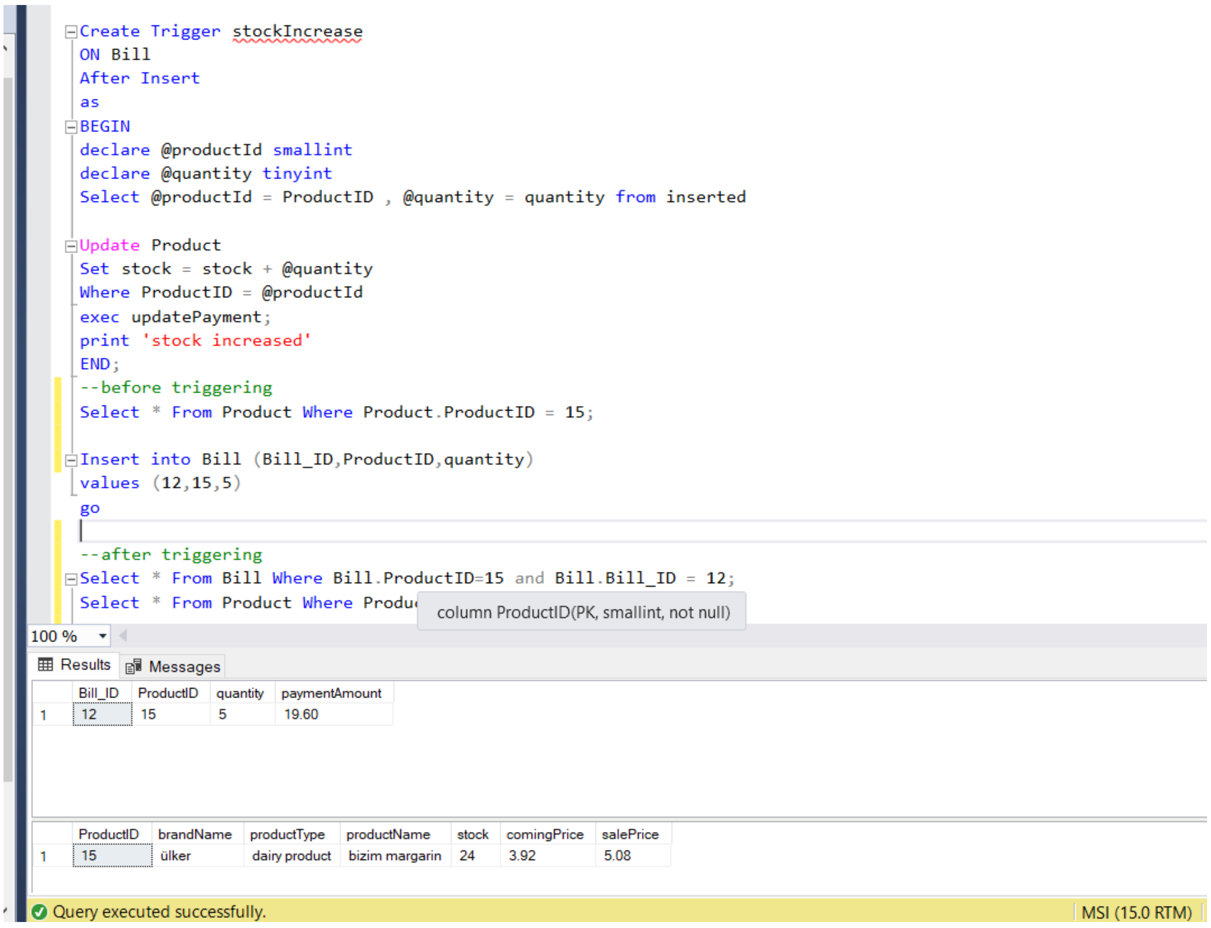
0 %

Results | Messages

ProductID	brandName	productType	productName	stock	comingPrice	salePrice
15	ülker	dairy product	bizim margarin	19	3.92	5.08

Query executed successfully.

After triggering:



```
Create Trigger stockIncrease
ON Bill
After Insert
as
BEGIN
declare @productId smallint
declare @quantity tinyint
Select @productId = ProductID , @quantity = quantity from inserted

Update Product
Set stock = stock + @quantity
Where ProductID = @productId
exec updatePayment;
print 'stock increased'
END;
--before triggering
Select * From Product Where Product.ProductID = 15;

Insert into Bill (Bill_ID,ProductID,quantity)
values (12,15,5)
go

--after triggering
Select * From Bill Where Bill.ProductID=15 and Bill.Bill_ID = 12;
Select * From Product Where Produ
```

column ProductID(PK, smallint, not null)

100 %

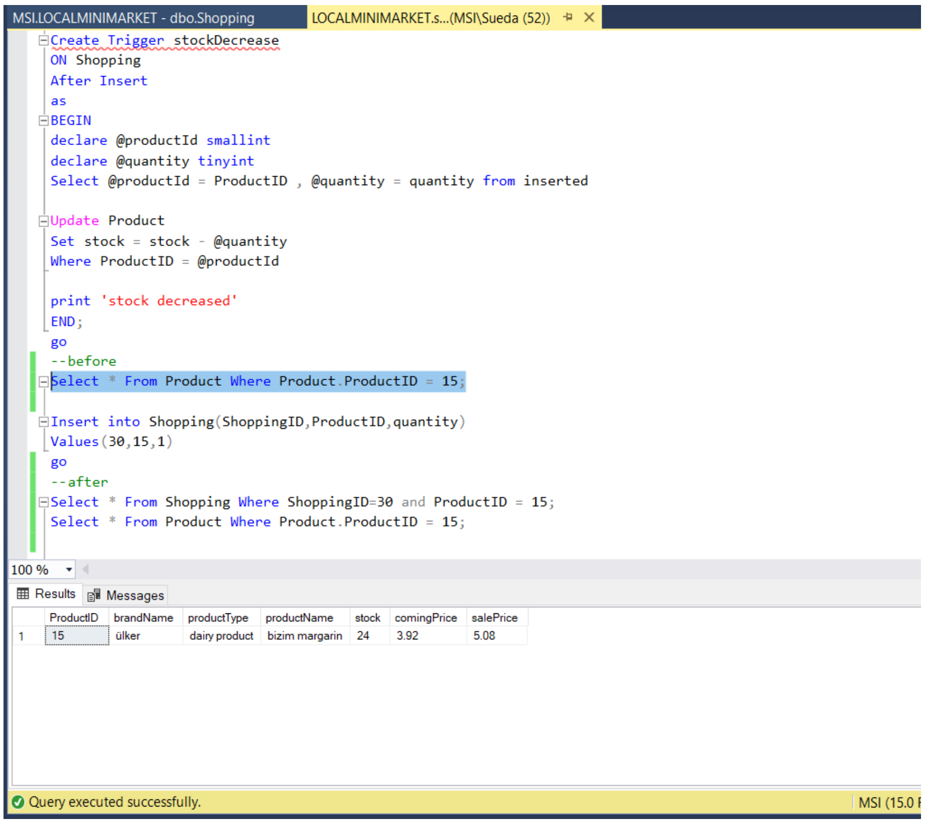
Results | Messages

	Bill_ID	ProductID	quantity	paymentAmount
1	12	15	5	19.60

	ProductID	brandName	productType	productName	stock	comingPrice	salePrice
1	15	ülker	dairy product	bizim margarin	24	3.92	5.08

Query executed successfully. MSI (15.0 RTM)

2.stockDecrease:When customer purchases products from minimarket,these product purchases are inserting to Shopping table.This trigger works when a purchase happened and it decreases the stock of that product with ordered quantity.



MSI.LOCALMINIMARKET - dbo.Shopping   LOCALMINIMARKET.s...(MSI\Sueda (52))

```
Create Trigger stockDecrease
ON Shopping
After Insert
as
BEGIN
declare @productId smallint
declare @quantity tinyint
Select @productId = ProductID , @quantity = quantity from inserted

Update Product
Set stock = stock - @quantity
Where ProductID = @productId

print 'stock decreased'
END;
go
--before
Select * From Product Where Product.ProductID = 15;

Insert into Shopping(ShoppingID,ProductID,quantity)
Values(30,15,1)
go
--after
Select * From Shopping Where ShoppingID=30 and ProductID = 15;
Select * From Product Where Product.ProductID = 15;
```

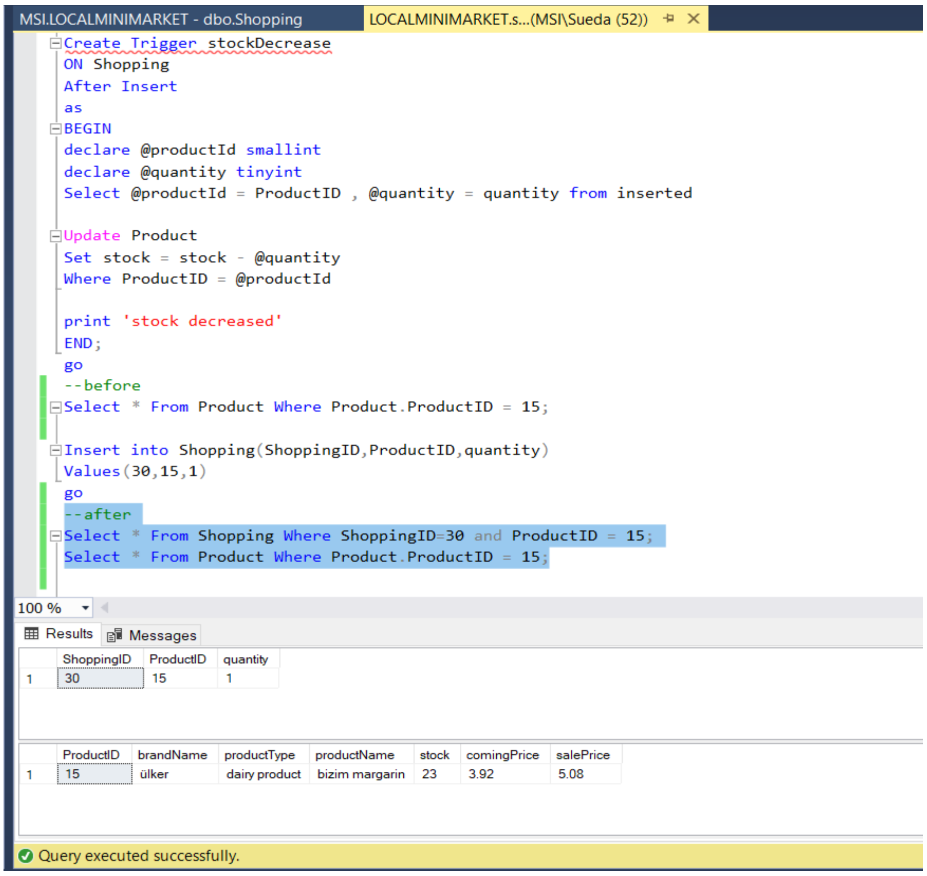
100 %

Results | Messages

	ProductID	brandName	productType	productName	stock	comingPrice	salePrice
1	15	ülker	dairy product	bizim margarin	24	3.92	5.08

Query executed successfully.   MSI (15.0

Before Triggering



MSI.LOCALMINIMARKET - dbo.Shopping   LOCALMINIMARKET.s...(MSI\Sueda (52))

```
Create Trigger stockDecrease
ON Shopping
After Insert
as
BEGIN
declare @productId smallint
declare @quantity tinyint
Select @productId = ProductID , @quantity = quantity from inserted

Update Product
Set stock = stock - @quantity
Where ProductID = @productId

print 'stock decreased'
END;
go
--before
Select * From Product Where Product.ProductID = 15;

Insert into Shopping(ShoppingID,ProductID,quantity)
Values(30,15,1)
go
--after
Select * From Shopping Where ShoppingID=30 and ProductID = 15;
Select * From Product Where Product.ProductID = 15;
```

100 %

Results | Messages

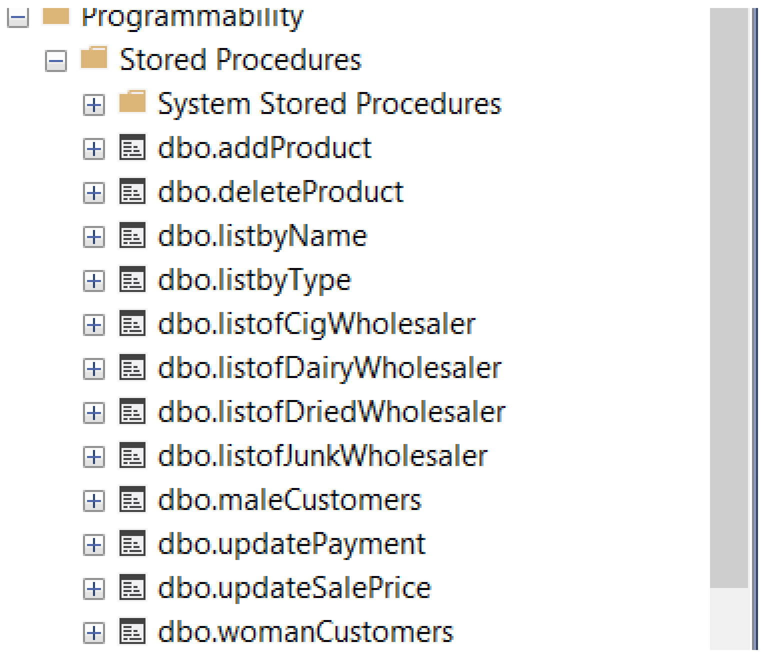
	ShoppingID	ProductID	quantity
1	30	15	1

	ProductID	brandName	productType	productName	stock	comingPrice	salePrice
1	15	ülker	dairy product	bizim margarin	23	3.92	5.08

Query executed successfully.

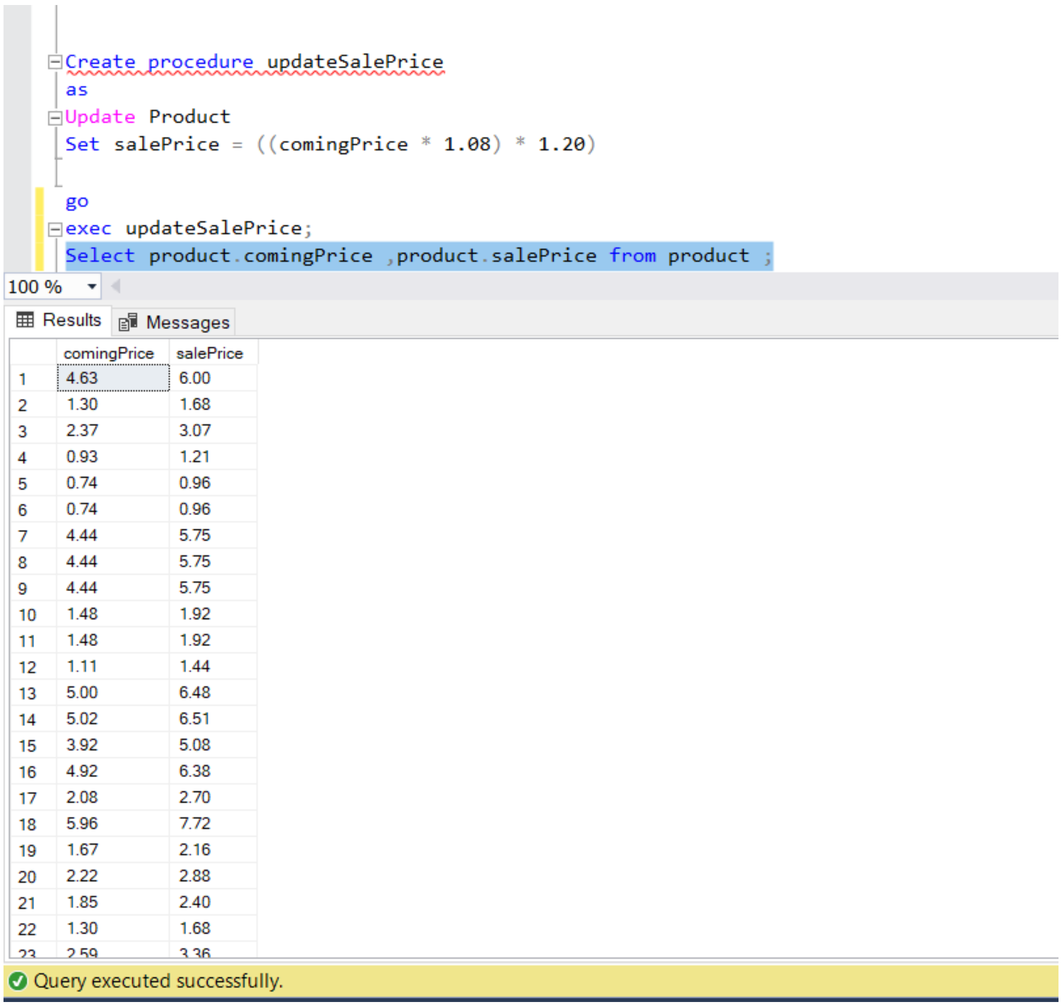
After Triggering

Store Procedures:



```
⊟ Programmability
    ⊟ Stored Procedures
        ⊞ System Stored Procedures
        ⊞ dbo.addProduct
        ⊞ dbo.deleteProduct
        ⊞ dbo.listbyName
        ⊞ dbo.listbyType
        ⊞ dbo.listofCigWholesaler
        ⊞ dbo.listofDairyWholesaler
        ⊞ dbo.listofDriedWholesaler
        ⊞ dbo.listofJunkWholesaler
        ⊞ dbo.maleCustomers
        ⊞ dbo.updatePayment
        ⊞ dbo.updateSalePrice
        ⊞ dbo.womanCustomers
```

1.updateSalePrice



```sql
Create procedure updateSalePrice
as
Update Product
Set salePrice = ((comingPrice * 1.08) * 1.20)

go
exec updateSalePrice;
Select product.comingPrice ,product.salePrice from product ;
```

100 %

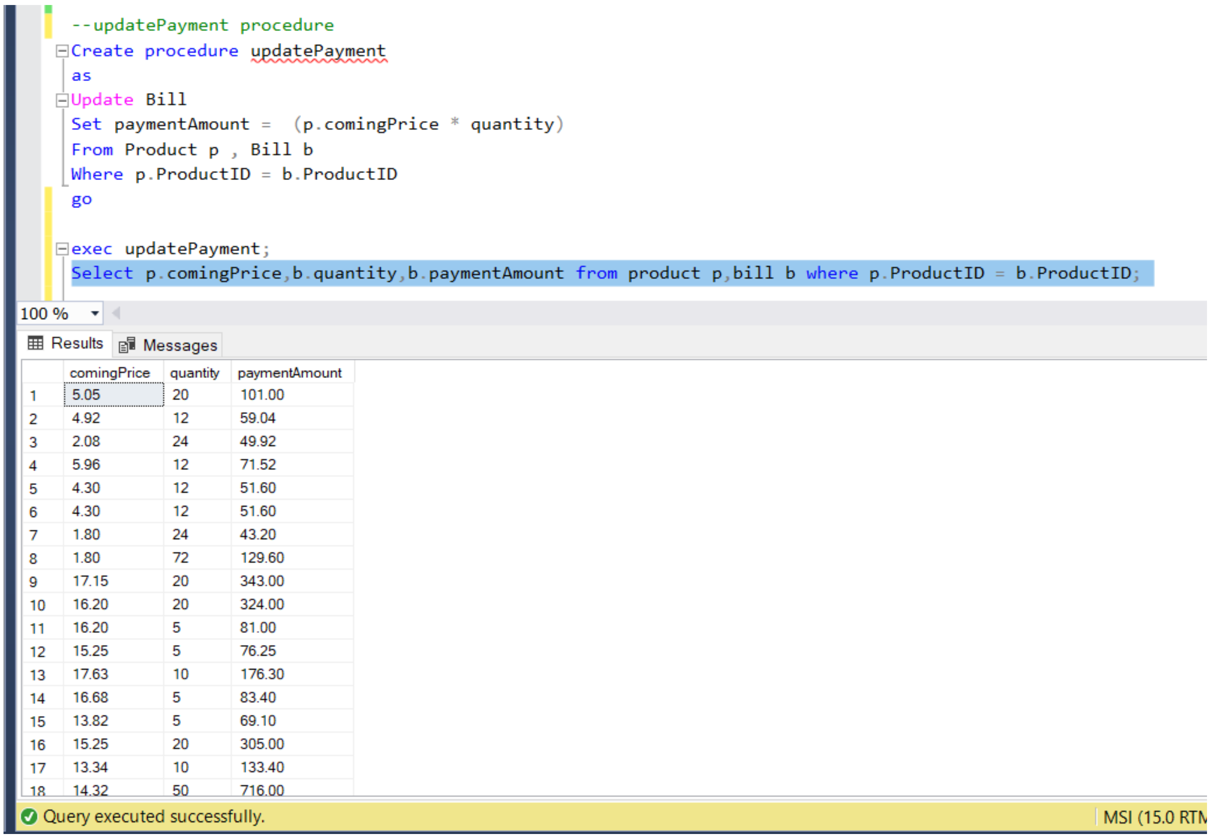
Results | Messages

	comingPrice	salePrice
1	4.63	6.00
2	1.30	1.68
3	2.37	3.07
4	0.93	1.21
5	0.74	0.96
6	0.74	0.96
7	4.44	5.75
8	4.44	5.75
9	4.44	5.75
10	1.48	1.92
11	1.48	1.92
12	1.11	1.44
13	5.00	6.48
14	5.02	6.51
15	3.92	5.08
16	4.92	6.38
17	2.08	2.70
18	5.96	7.72
19	1.67	2.16
20	2.22	2.88
21	1.85	2.40
22	1.30	1.68
23	2.59	3.36

Query executed successfully.

Updates sale prices by taking coming prices.

2.updatePayment



```sql
--updatePayment procedure
Create procedure updatePayment
as
Update Bill
Set paymentAmount =  (p.comingPrice * quantity)
From Product p , Bill b
Where p.ProductID = b.ProductID
go

exec updatePayment;
Select p.comingPrice,b.quantity,b.paymentAmount from product p,bill b where p.ProductID = b.ProductID;
```

100 %

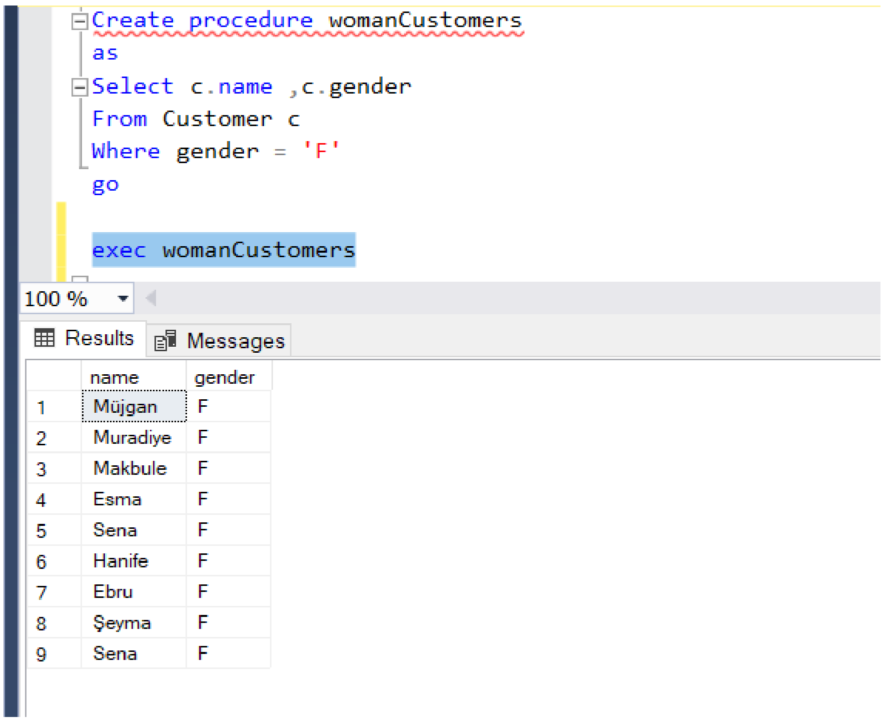
Results | Messages

	comingPrice	quantity	paymentAmount
1	5.05	20	101.00
2	4.92	12	59.04
3	2.08	24	49.92
4	5.96	12	71.52
5	4.30	12	51.60
6	4.30	12	51.60
7	1.80	24	43.20
8	1.80	72	129.60
9	17.15	20	343.00
10	16.20	20	324.00
11	16.20	5	81.00
12	15.25	5	76.25
13	17.63	10	176.30
14	16.68	5	83.40
15	13.82	5	69.10
16	15.25	20	305.00
17	13.34	10	133.40
18	14.32	50	716.00

Query executed successfully. | MSI (15.0 RTN

Updates payment by taking coming price and quantity values.

3.womanCustomers



```sql
Create procedure womanCustomers
as
Select c.name ,c.gender
From Customer c
Where gender = 'F'
go

exec womanCustomers
```

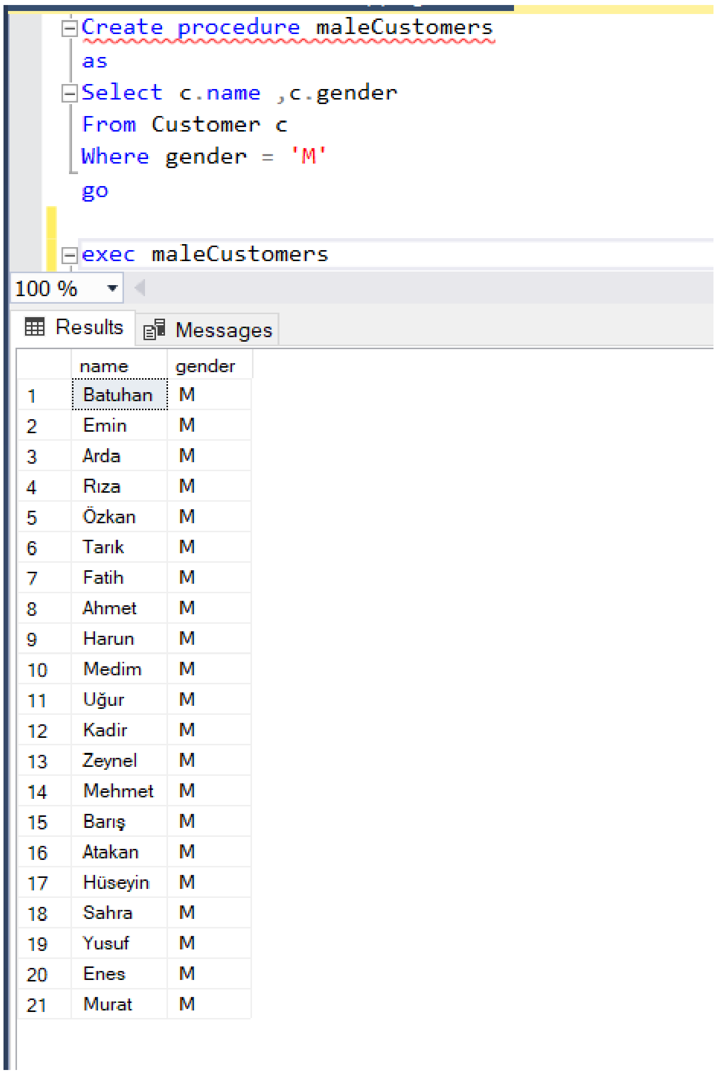
100 %

Results | Messages

	name	gender
1	Müjgan	F
2	Muradiye	F
3	Makbule	F
4	Esma	F
5	Sena	F
6	Hanife	F
7	Ebru	F
8	Şeyma	F
9	Sena	F

Lists woman customers.

4.maleCustomers



```
Create procedure maleCustomers
as
Select c.name ,c.gender
From Customer c
Where gender = 'M'
go

exec maleCustomers
```

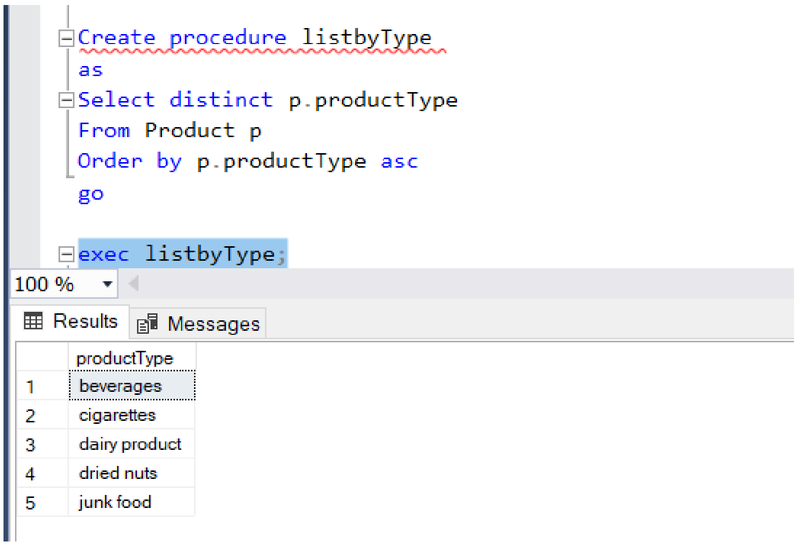
100 %

Results | Messages

```
    name     gender
1   Batuhan  M
2   Emin     M
3   Arda     M
4   Rıza     M
5   Özkan    M
6   Tarık    M
7   Fatih    M
8   Ahmet    M
9   Harun    M
10  Medim    M
11  Uğur     M
12  Kadir    M
13  Zeynel   M
14  Mehmet   M
15  Barış    M
16  Atakan   M
17  Hüseyin  M
18  Sahra    M
19  Yusuf    M
20  Enes     M
21  Murat    M
```

Lists male customers.

5.listbyType



```
Create procedure listbyType
as
Select distinct p.productType
From Product p
Order by p.productType asc
go

exec listbyType;
```

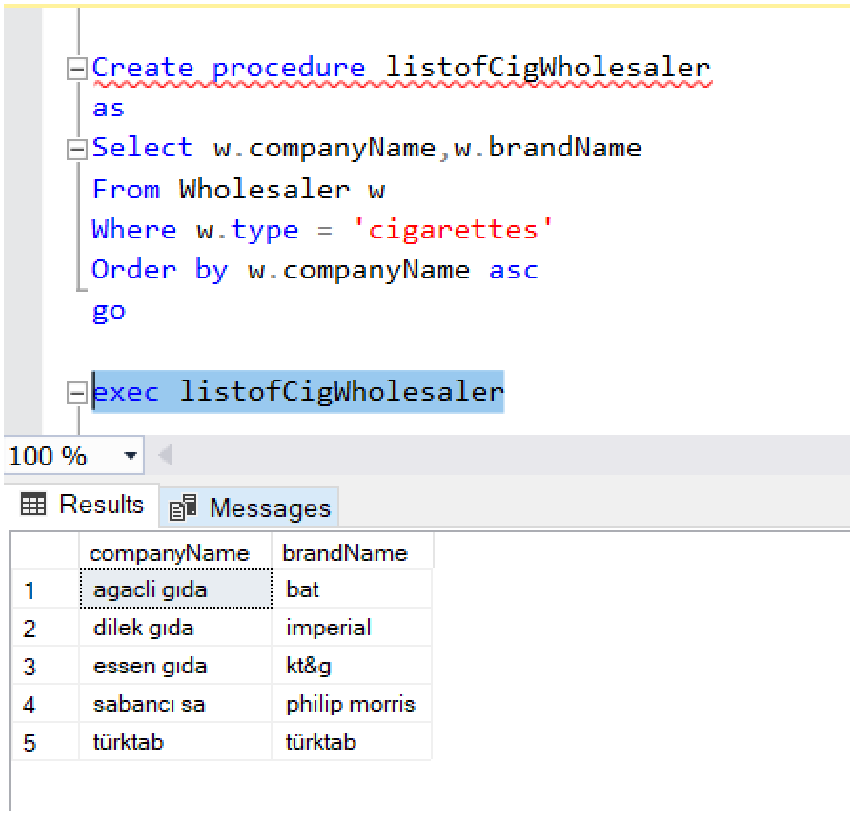
100 %

Results | Messages

```
   productType
1  beverages
2  cigarettes
3  dairy product
4  dried nuts
5  junk food
```

Lists products by their product types.

6.listofCigWholesaler



```
Create procedure listofCigWholesaler
as
Select w.companyName,w.brandName
From Wholesaler w
Where w.type = 'cigarettes'
Order by w.companyName asc
go

exec listofCigWholesaler
```

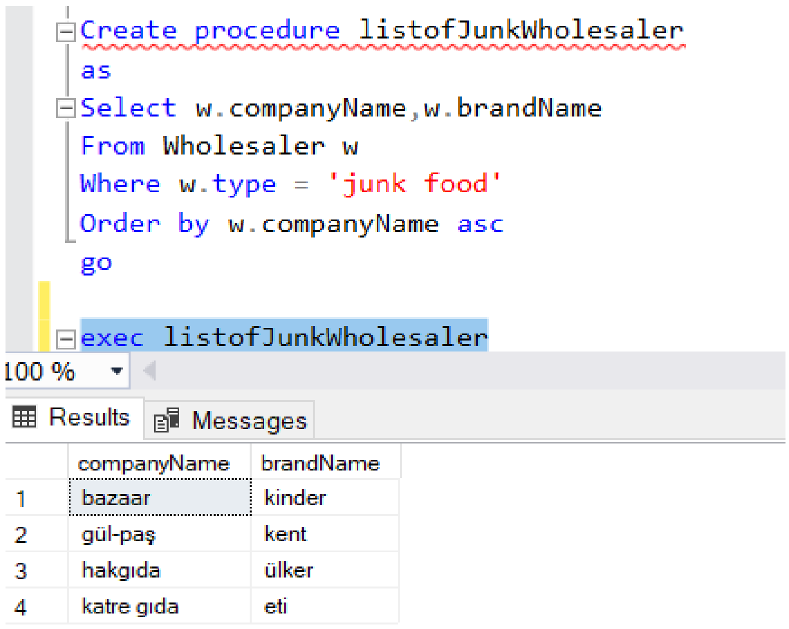
100 %

Results Messages

	companyName	brandName
1	agacli gıda	bat
2	dilek gıda	imperial
3	essen gıda	kt&g
4	sabancı sa	philip morris
5	türktab	türktab

Lists wholesalers which sells cigarettes.

6.listofJunkWholesaler



```
Create procedure listofJunkWholesaler
as
Select w.companyName,w.brandName
From Wholesaler w
Where w.type = 'junk food'
Order by w.companyName asc
go

exec listofJunkWholesaler
```

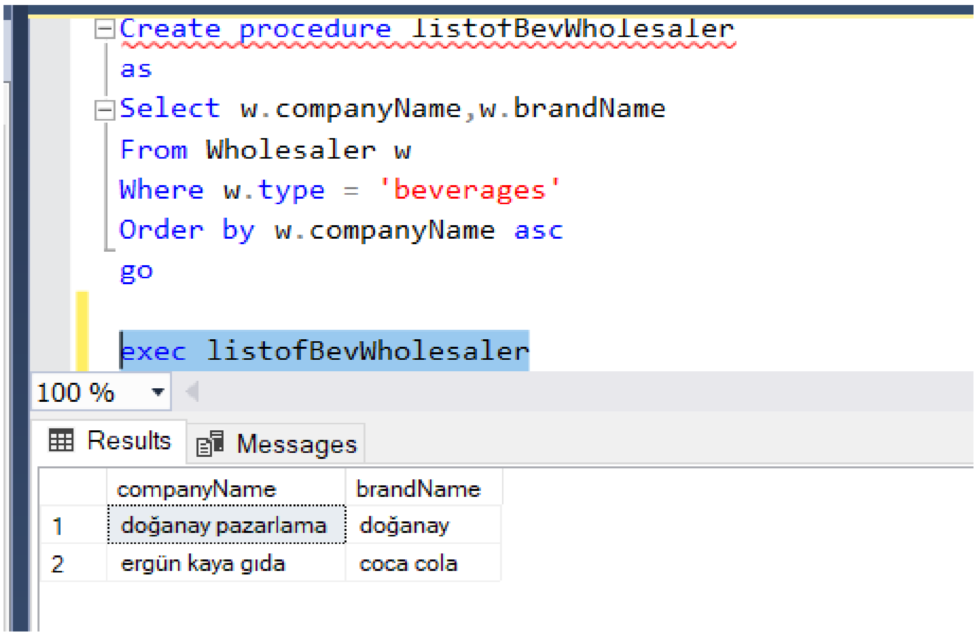
100 %

Results Messages

	companyName	brandName
1	bazaar	kinder
2	gül-paş	kent
3	hakgıda	ülker
4	katre gıda	eti

Lists wholesalers which sells junk food.

7.listofBevWholesaler:



```
Create procedure listofBevWholesaler
as
Select w.companyName,w.brandName
From Wholesaler w
Where w.type = 'beverages'
Order by w.companyName asc
go

exec listofBevWholesaler
```

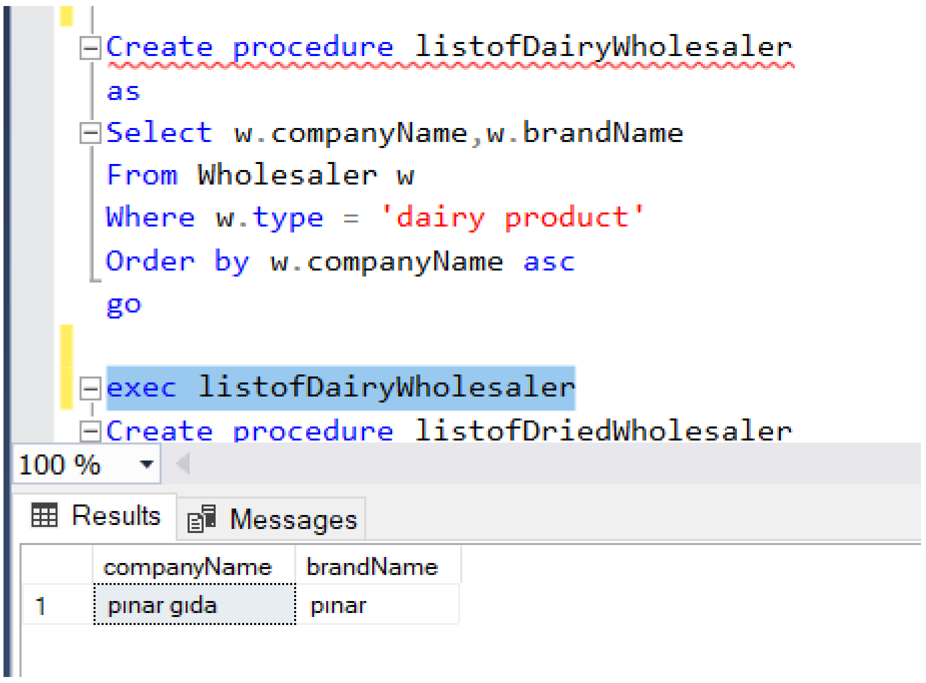
100 %

Results | Messages

	companyName	brandName
1	doğanay pazarlama	doğanay
2	ergün kaya gıda	coca cola

Lists wholesalers which sells beverages.

8.listofDairyWholesaler



```
Create procedure listofDairyWholesaler
as
Select w.companyName,w.brandName
From Wholesaler w
Where w.type = 'dairy product'
Order by w.companyName asc
go

exec listofDairyWholesaler
Create procedure listofDriedWholesaler
```

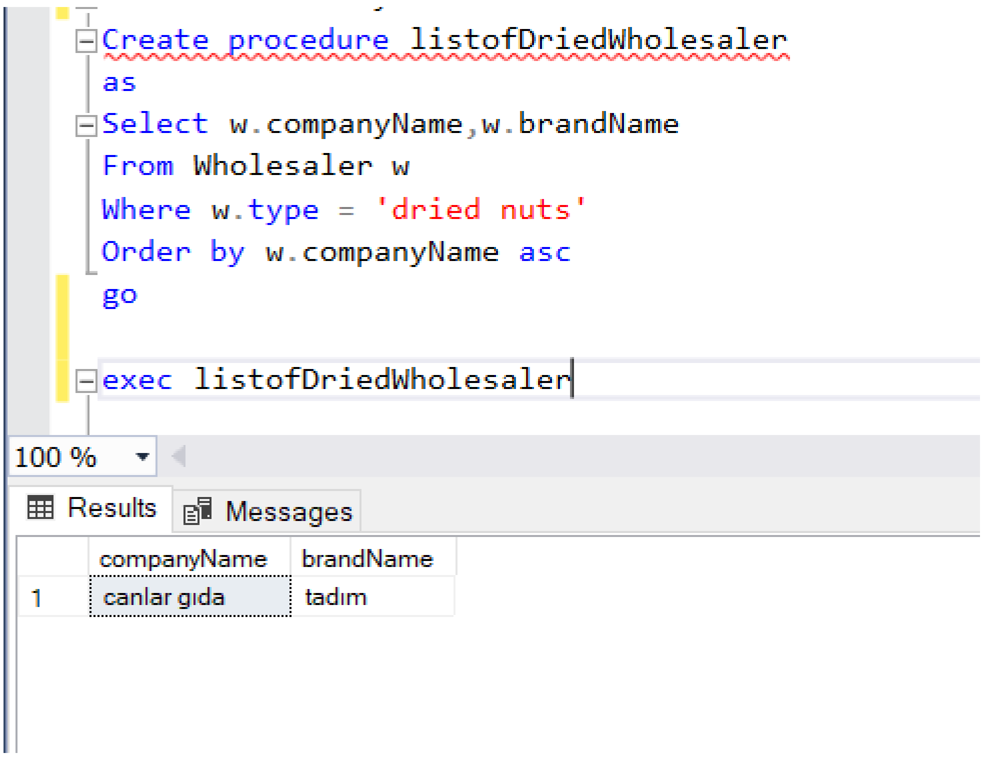
100 %

Results | Messages

	companyName	brandName
1	pınar gıda	pınar

Lists wholesalers which sells dairy product.

9.listofDriedWholesaler

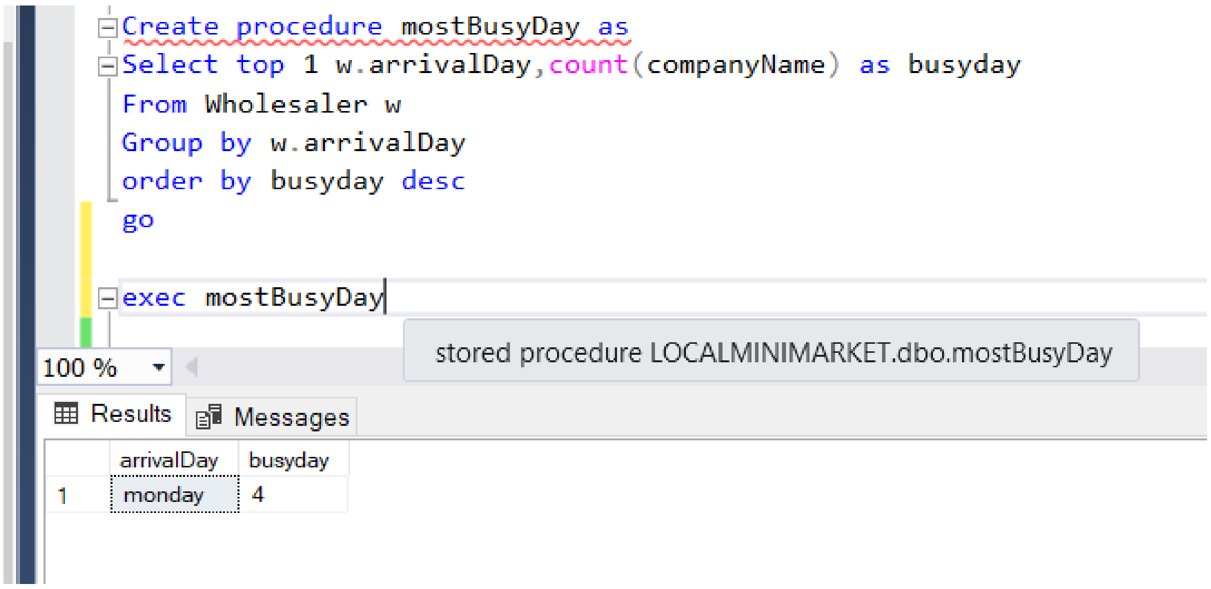


```
Create procedure listofDriedWholesaler
as
Select w.companyName,w.brandName
From Wholesaler w
Where w.type = 'dried nuts'
Order by w.companyName asc
go

exec listofDriedWholesaler
```

```
   companyName   brandName
1  canlar gıda   tadım
```

Lists wholesalers which sells dried nuts.

10.mostBusyDay



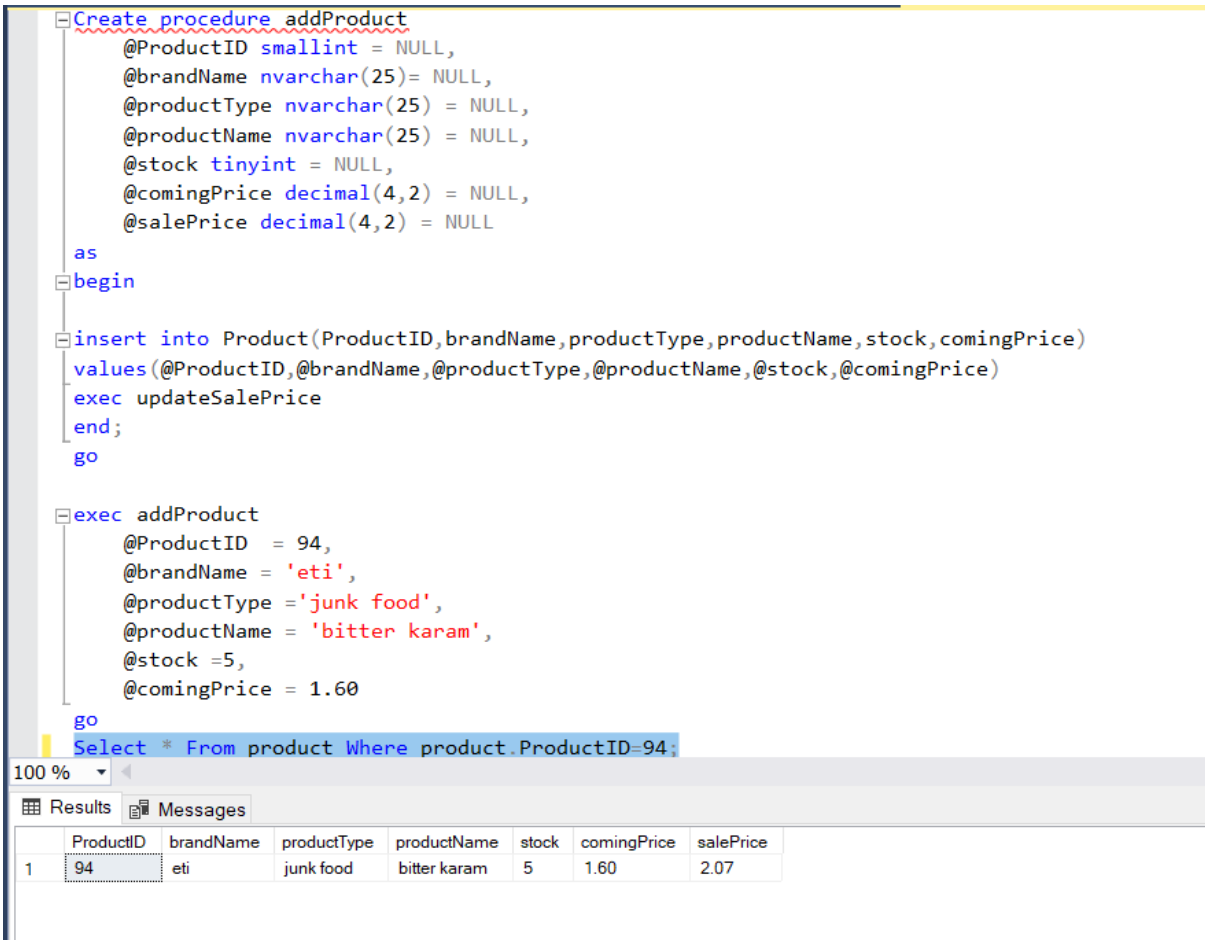
```
Create procedure mostBusyDay as
Select top 1 w.arrivalDay,count(companyName) as busyday
From Wholesaler w
Group by w.arrivalDay
order by busyday desc
go

exec mostBusyDay
```

stored procedure LOCALMINIMARKET.dbo.mostBusyDay

```
   arrivalDay   busyday
1  monday       4
```

Finds most busy day which is the day many wholesaler comes to take orders.

11.addProduct



```
Create procedure addProduct
        @ProductID smallint = NULL,
        @brandName nvarchar(25)= NULL,
        @productType nvarchar(25) = NULL,
        @productName nvarchar(25) = NULL,
        @stock tinyint = NULL,
        @comingPrice decimal(4,2) = NULL,
        @salePrice decimal(4,2) = NULL
as
begin

insert into Product(ProductID,brandName,productType,productName,stock,comingPrice)
values(@ProductID,@brandName,@productType,@productName,@stock,@comingPrice)
exec updateSalePrice
end;
go

exec addProduct
        @ProductID  = 94,
        @brandName = 'eti',
        @productType ='junk food',
        @productName = 'bitter karam',
        @stock =5,
        @comingPrice = 1.60
go
Select * From product Where product.ProductID=94;
```

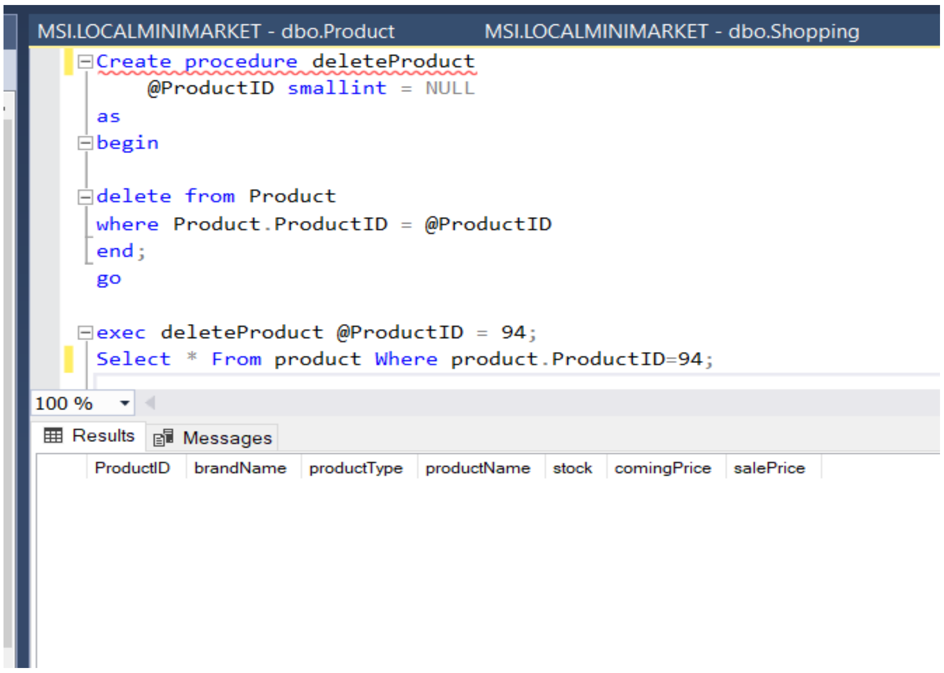
100 %

Results | Messages

	ProductID	brandName	productType	productName	stock	comingPrice	salePrice
1	94	eti	junk food	bitter karam	5	1.60	2.07

This stored procedure is to adding new product to the product table.

12.deleteProduct



MSI.LOCALMINIMARKET - dbo.Product   MSI.LOCALMINIMARKET - dbo.Shopping

```
Create procedure deleteProduct
        @ProductID smallint = NULL
as
begin

delete from Product
where Product.ProductID = @ProductID
end;
go

exec deleteProduct @ProductID = 94;
Select * From product Where product.ProductID=94;
```

100 %

Results | Messages

ProductID	brandName	productType	productName	stock	comingPrice	salePrice

This stored procedure deletes a product from the product table.