

# **Database Systems Project Step 3**

## **Kesgin Kuafor Database**

### **Members:**

**150116836 – Enes Nadir SANLI as Representative**

**150116844 – Abdurrahman ŞAKAR**

**150113852 – Doğukan DENİZ**

### **Content**

<b>Project Description .....</b>	<b>1</b>
<b>Scope .....</b>	<b>1</b>
<b>Data Requirements Analysis .....</b>	<b>2</b>
<b>Tables with Their Definitions .....</b>	<b>2</b>
<b>Tables with Their Definitions .....</b>	<b>3</b>
<b>Views .....</b>	<b>3</b>
<b>Triggers .....</b>	<b>6</b>
<b>Stored Procedures .....</b>	<b>8</b>

## **Project Description**

Kesgin Kuafor is a barber store. There are a lot of people that come here. But the problem here is that they didn't keep any data of their customers up to now. Reservation situations are made by phone and they are written on the papers. Also, they sell care products like wax, hair spray and shampoo etc. for increasing their incomes. All these operations are kept on the papers or receipt so they can't trace monthly customer count, income or data at a specific month easily. We have a project for them. We will implement a database for all their records. In our database project we have Customer, Staff, Owner, Barber, Salary, Receipt, CategoryChoice, Category, ProductChoice, Reservation, Storage, InChargeOfStorage and Product tables. After the implemented database we will present this project to the client within a web application. Web application will have a useful and simple interface so our client can use on all their work instead of paper.

## **Scope**

While we were planning our database with its requirements and data, we determined most of the required field. We have included what we planned but during the process of completing our project we had some additional requirements. Thinking about the project from the beginning, we were supported about the project by the owner of company during the processes. But for some parts like salaries of owners and barbers we created data different from the real one.

## Data Requirements Analysis

We provided unreal data for some relations like Reservation, Receipt, Customer and Staff tables. Because of the business rule we also couldn't provide real Salary data for the table. Current state of the database can be enough for demos with client.

## Tables with Their Definitions

There is a file named "DatabaseTables.jpg" in the project file, It has tables and information about attributes, Primary/Foreign Key, datatypes, unique and etc. Here we add definitions of tables.

**Staff:** This table keeps information about the people of the company and has two types one is Owner and other one is Barber.

**Salary:** This table keeps owner's determined salaries for the barber and also himself.

**Customer:** This table keeps information about customers that come to the barber store.

**Reservation:** This table keeps information about reservation of customers done.

**Receipt:** This table keeps information about the prices of bought products and done reservations.

**Storage:** This table keeps information about the storage that products kept.

**InChargeOfStorage:** Each storage has to have a responsible person, and this table keeps information about that person.

**Product:** This table holds information about product to be sold.

**Category:** This table holds information about categories that the customer can choose for their needs.

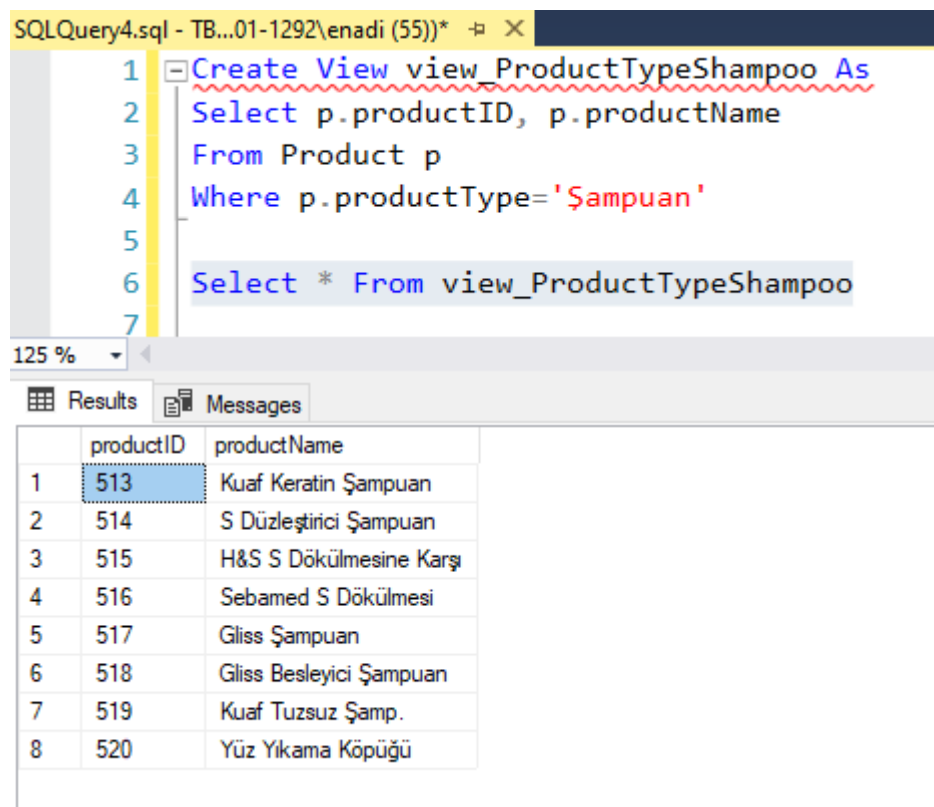
**ProductChoice:** This table keeps the products that written a specific receipt.

**CategoryChoice:** This table keeps the categories that written a specific receipt.

## Views

1- **Name:** view\_ProductTypeShampoo

**Def:** This view lists only the product type shampoo.



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows the SQL query editor with the following code:

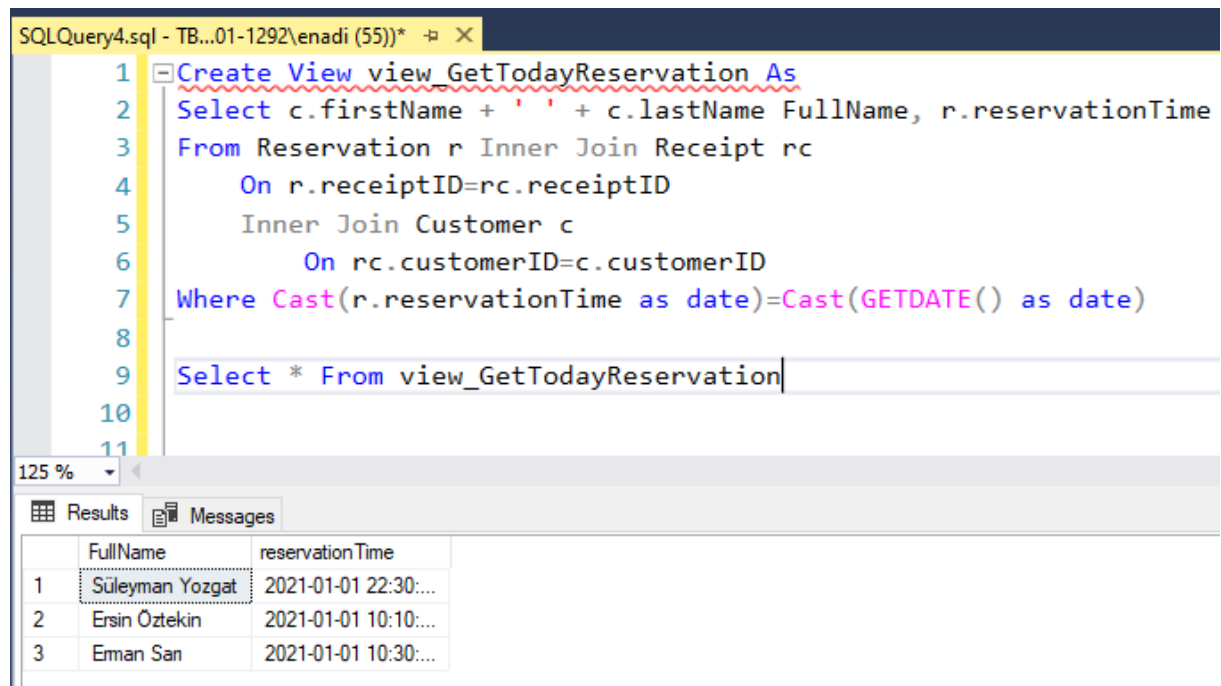
```
1 Create View view_ProductTypeShampoo As
2 Select p.productID, p.productName
3 From Product p
4 Where p.productType='Şampuan'
5
6 Select * From view_ProductTypeShampoo
7
```

The bottom pane shows the 'Results' tab with a table containing 8 rows of data. The first row is highlighted.

	productID	productName
1	513	Kuaf Keratin Şampuan
2	514	S Düzleştirici Şampuan
3	515	H&S S Dökülmesine Karşı
4	516	Sebamed S Dökülmesi
5	517	Gliss Şampuan
6	518	Gliss Besleyici Şampuan
7	519	Kuaf Tuzsuz Şamp.
8	520	Yüz Yıkama Köpüğü

## 2- Name: view\_GetTodayReservation

**Def:** This view lists the reservations of the day



```
SQLQuery4.sql - TB...01-1292\enadi (55))* -> X
1 Create View view_GetTodayReservation As
2 Select c.firstName + ' ' + c.lastName FullName, r.reservationTime
3 From Reservation r Inner Join Receipt rc
4     On r.receiptID=rc.receiptID
5     Inner Join Customer c
6     On rc.customerID=c.customerID
7 Where Cast(r.reservationTime as date)=Cast(GETDATE() as date)
8
9 Select * From view_GetTodayReservation
10
11
```

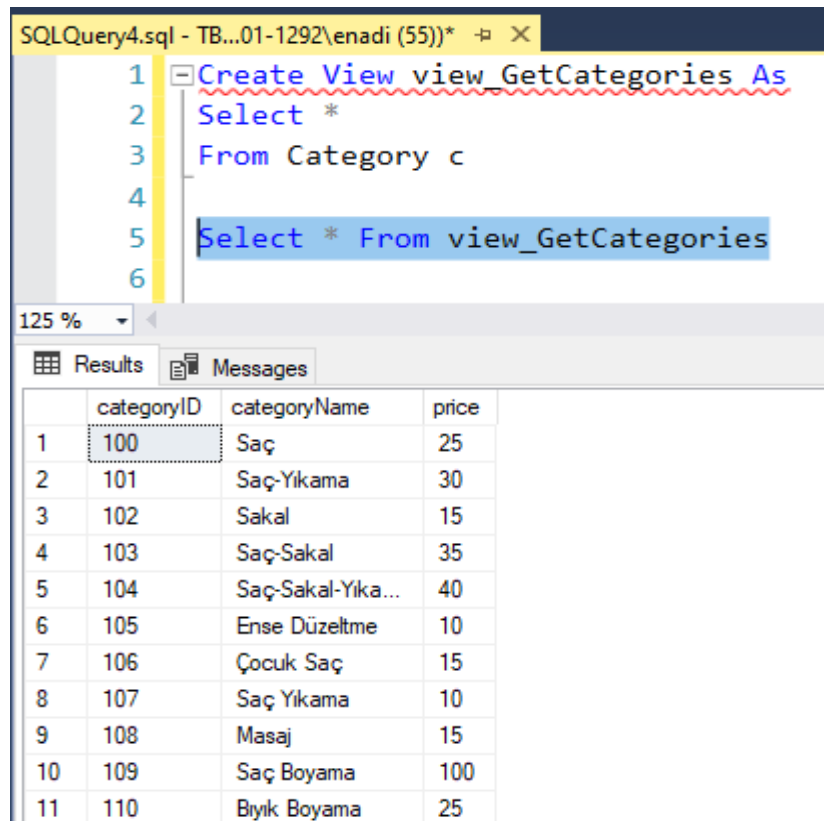
125 %

Results Messages

	FullName	reservationTime
1	Süleyman Yozgat	2021-01-01 22:30:...
2	Ersin Öztekin	2021-01-01 10:10:...
3	Eman San	2021-01-01 10:30:...

## 3- Name: view\_GetCategories

**Def:** This view lists all categories from category table



```
SQLQuery4.sql - TB...01-1292\enadi (55))* -> X
1 Create View view_GetCategories As
2 Select *
3 From Category c
4
5 Select * From view_GetCategories
6
```

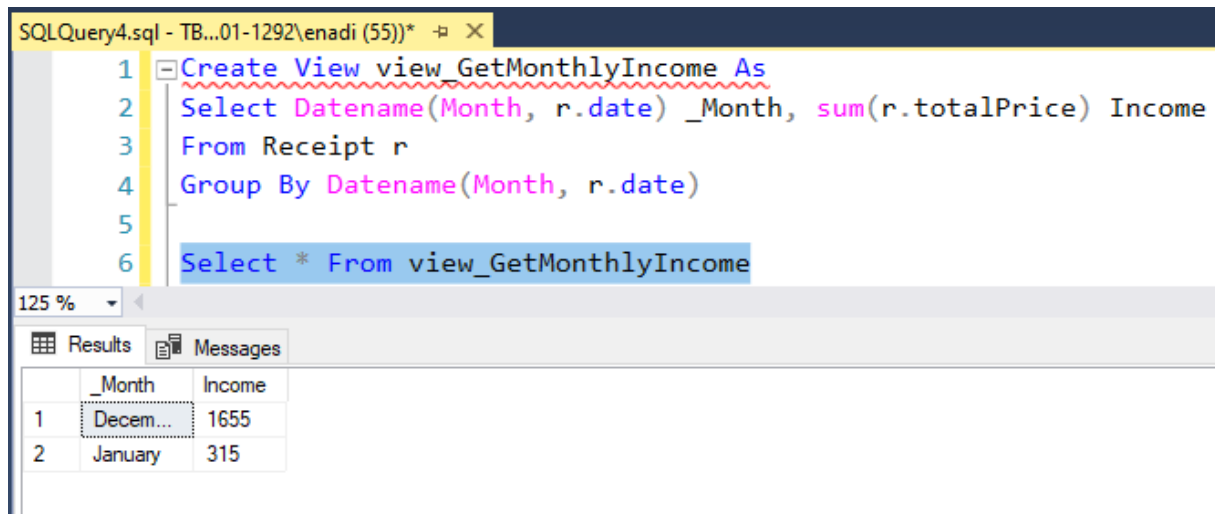
125 %

Results Messages

	categoryID	categoryName	price
1	100	Saç	25
2	101	Saç-Yıkama	30
3	102	Sakal	15
4	103	Saç-Sakal	35
5	104	Saç-Sakal-Yika...	40
6	105	Ense Düzeltme	10
7	106	Çocuk Saç	15
8	107	Saç Yıkama	10
9	108	Masaj	15
10	109	Saç Boyama	100
11	110	Biyık Boyama	25

#### 4- Name: view\_GetMonthlyIncome

**Def:** This view lists income of each month



```
SQLQuery4.sql - TB...01-1292\enadi (55))*  X
1 Create View view_GetMonthlyIncome As
2 Select Datename(Month, r.date) _Month, sum(r.totalPrice) Income
3 From Receipt r
4 Group By Datename(Month, r.date)
5
6 Select * From view_GetMonthlyIncome
```

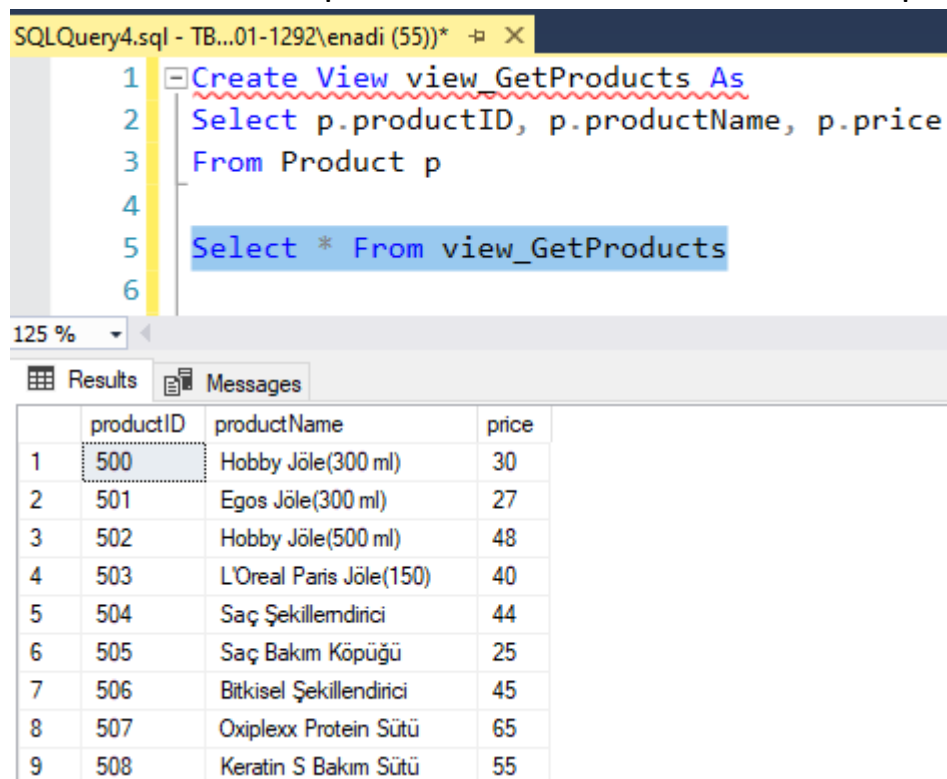
125 %

Results Messages

	_Month	Income
1	Decem...	1655
2	January	315

#### 5- Name: view\_GetProducts

**Def:** This view list products with their id, name and price



```
SQLQuery4.sql - TB...01-1292\enadi (55))*  X
1 Create View view_GetProducts As
2 Select p.productID, p.productName, p.price
3 From Product p
4
5 Select * From view_GetProducts
6
```

125 %

Results Messages

	productID	productName	price
1	500	Hobby Jöle(300 ml)	30
2	501	Egos Jöle(300 ml)	27
3	502	Hobby Jöle(500 ml)	48
4	503	L'Oreal Paris Jöle(150)	40
5	504	Saç Şekillemdirici	44
6	505	Saç Bakım Köpüğü	25
7	506	Bitkisel Şekillendirici	45
8	507	Oxiplexx Protein Sütü	65
9	508	Keratin S Bakım Sütü	55

# Triggers

## 1- Name: TrgSetReceiptDate

Def: This trigger work after insertion to receipt table and sets the date of receipt table.

```
SQLQuery4.sql - TB...01-1292\enadi (55))* -# X
1 Create Trigger TrgSetReceiptDate
2   On Receipt
3   After Insert
4   As
5   Begin
6       Set Nocount On;
7       update r
8       set date=Cast(GETDATE() as smalldatetime)
9       From Receipt r inner join inserted i
10          on r.receiptID=i.receiptID
11   End
12
```

## 2- Name: TrgDecreaseSoldProductCount

Def: When there is an insertion or deletion on the productChoice table this trigger adds or decreases the quantity from product table.

```
SQLQuery4.sql - TB...01-1292\enadi (55))* -# X
1 Create Trigger TrgDecreaseSoldProductCount
2   On ProductChoice
3   After Insert, Delete, Update
4   As
5   Begin
6       Set Nocount On;
7       Update p
8       Set quantity= p.quantity-i.quantity
9       From Product p Inner Join inserted i
10          On p.productID=i.productID
11
12       Update p
13       Set quantity= p.quantity+d.quantity
14       From Product p Inner Join deleted d
15          On p.productID=d.productID
16   End
17
```

### 3- Name: TrgSetStaffSubtypeContent

Def: This trigger inserts the new staff record to the Owner or Barber table according to its staffType.

```
SQLQuery4.sql - TB...01-1292\enadi (55))* -p X
1 Create Trigger TrgSetStaffSubtypeContent
2 On Staff
3 After Insert
4 As
5 Begin
6     Declare @newSsn smallint, @staffType char(1)
7     Set Nocount On;
8     If Exists(Select * From inserted)
9     Begin
10         set @newSsn = (Select i.SSN From inserted i)
11         set @staffType = (Select i.staffType From inserted i)
12
13         If @staffType='o'
14         Begin
15             Insert Into Ownerr(OSSN) values(@newSsn)
16         End
17         Else If @staffType='b'
18         Begin
19             Insert Into Barber(BSSN) values(@newSsn)
20         End
21     End
22 End
```



# Stored Procedures

1- Name: Sp\_CreateReceipt

Def: This SP creates receipt with given customerID as parameter.

Before:

35	39	1000	2021-01-01 14:00:...	115
36	40	1001	2021-01-01 14:24:...	200

✓ Query executed successfully.

After:

36	40	1001	2021-01-01 14:24:...	200
37	41	1001	2021-01-01 22:16:...	0

✓ Query executed successfully.

```
SQLQuery13.sql - T...01-1292\enadi (56))*  SQLQuery4.sql - TB...01-1292\enadi (55))*
1 CREATE PROCEDURE Sp_CreateReceipt
2     @customerID smallint,
3     @receiptID smallint Output
4 AS
5 BEGIN
6     Set Nocount On
7     Insert Into Receipt(customerID, totalPrice) Values(@customerID, 0)
8     Select @receiptID=@@Identity
9     Return
10 END
11 GO
12 Declare @receiptID smallint
13 Exec Sp_CreateReceipt 1001, @receiptID = @receiptID Output;
```

## 2- Name: Sp\_CreateProductReceipt

Def: This SP adds products to ProductChoice table and updates receipt totalPrice of given receiptID

Before:

8	39	505	1	1	25
9	39	511	1	1	30

After:

36	40	1001	2021-01-01 14:24:...	200
37	41	1001	2021-01-01 22:16:...	22

```
SQLQuery13.sql - T...01-1292\enadi (56))*  SQLQuery4.sql - TB...01-1292\enadi (55))*
1 Create Proc Sp_CreateProductReceipt
2     @receiptID smallint, @newProductID smallint, @newStorageID smallint, @price smallint
3 As
4 Begin
5     Set Nocount On
6     If Exists(Select * From ProductChoice pc
7         where pc.receiptID=@receiptID and pc.productID=@newProductID and pc.storageID=@newStorageID)
8     Begin
9         Update pc
10        Set pc.quantity = pc.quantity + 1
11        From ProductChoice pc
12        Where pc.receiptID=@receiptID and pc.productID=@newProductID and pc.storageID=@newStorageID
13    End
14    Else
15    Begin
16        Insert Into ProductChoice Values(@receiptID, @newProductID, @newStorageID, 1, @price)
17    End
18    Update r
19    Set r.totalPrice = r.totalPrice + @price
20    From Receipt r
21    Where r.receiptID=@receiptID
22 End
23 GO
24
25 Exec Sp_CreateProductReceipt 41, 500, 1, 22
26 Select * From Receipt
```

### 3- Name: Sp\_CreateReservationReceipt

Def: This SP adds categories to CategoryChoice table and updates receipt totalPrice of given receiptID

Before:

36	40	1001	2021-01-01 14:24:...	200
37	41	1001	2021-01-01 22:16:...	22

After:

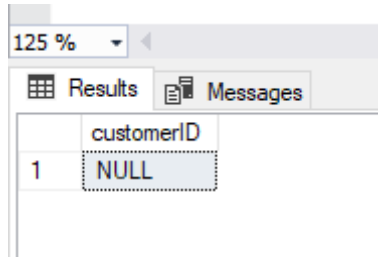
36	40	1001	2021-01-01 14:24:...	200
37	41	1001	2021-01-01 22:16:...	44

```
SQLQuery13.sql - T...01-1292\enadi (56))*  SQLQuery4.sql - TB...01-1292\enadi (55))*
1 Create Proc Sp_CreateReservationReceipt
2 @receiptID smallint, @newCategoryID smallint, @price smallint
3 As
4 Begin
5     Set Nocount On
6
7     If Exists(Select * From CategoryChoice cc
8               where cc.receiptID=@receiptID and cc.categoryID=@newCategoryID)
9     Begin
10        Update cc
11        Set cc.quantity = cc.quantity + 1
12        From CategoryChoice cc
13        Where cc.receiptID=@receiptID and cc.categoryID=@newCategoryID
14    End
15    Else
16    Begin
17        Insert Into CategoryChoice Values(@receiptID, @newCategoryID, 1, @price)
18    End
19
20    Update r
21    Set r.totalPrice = r.totalPrice + @price
22    From Receipt r
23    Where r.receiptID=@receiptID
24 End
25 GO
26 Exec Sp_CreateReservationReceipt 41, 109, 22
27 Select * From Receipt
```

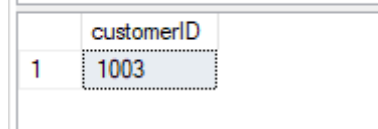
#### 4- Name: Sp\_GetCustomerID

Def: This SP returns Existing Customer or Creates new one's customerID.

Before/After:



	customerID
1	NULL

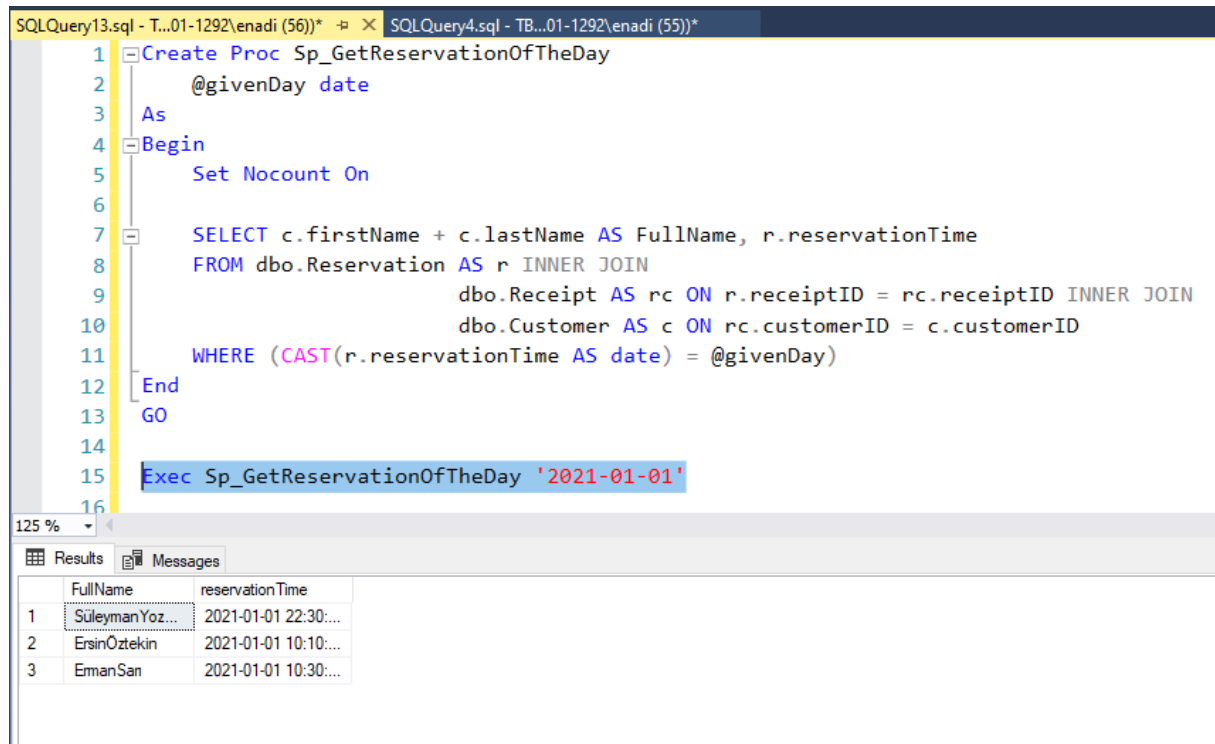
  


	customerID
1	1003

```
SQLQuery13.sql - T...01-1292\enadi (56))*  SQLQuery4.sql - TB...01-1292\enadi (55))*
1 Create Proc Sp_GetCustomerID
2     @customerName nvarchar(25), @customerLastName nvarchar(25),
3     @customerID smallint Output
4 As
5 Begin
6     Set Nocount On
7
8     If Exists(Select * From Customer c
9         Where c.firstName like @customerName and c.lastName like @customerLastName)
10    Begin
11        Select @customerID = c.customerID
12        From Customer c
13        Where c.firstName like @customerName and c.lastName like @customerLastName
14        Return
15    End
16    Else
17    Begin
18        Insert Into Customer(firstName, lastName) Values(@customerName, @customerLastName)
19        Select @customerID=@@Identity
20        Return
21    End
22 End
23 GO
24 Declare @customerID smallint
25 select @customerID customerID
26 Exec Sp_GetCustomerID 'hakan', 'sarı', @customerID = @customerID Output;
27 select @customerID customerID
```

## 5- Name: Sp\_GetReservationOfTheDay

Def: This SP returns Reservation of the given day

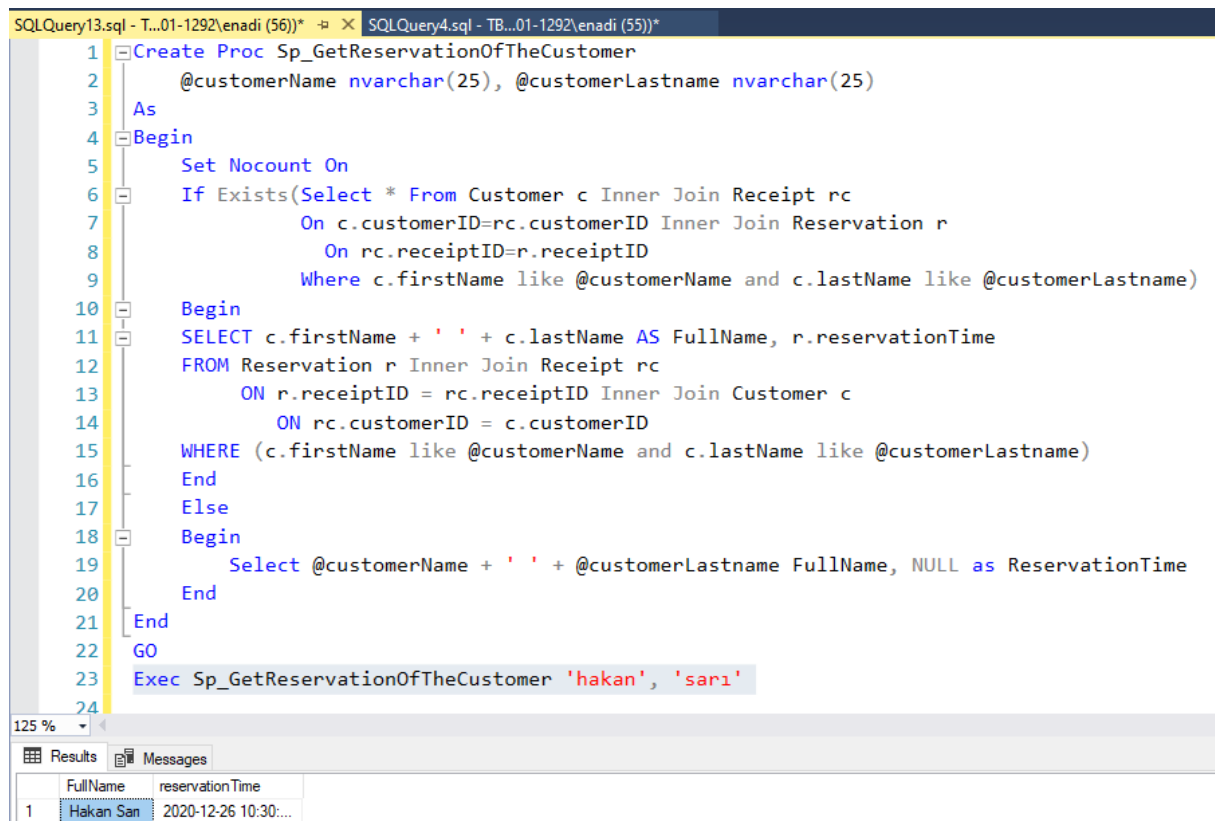


```
1 Create Proc Sp_GetReservationOfTheDay
2     @givenDay date
3 As
4 Begin
5     Set Nocount On
6
7     SELECT c.firstName + c.lastName AS FullName, r.reservationTime
8     FROM dbo.Reservation AS r INNER JOIN
9           dbo.Receipt AS rc ON r.receiptID = rc.receiptID INNER JOIN
10          dbo.Customer AS c ON rc.customerID = c.customerID
11     WHERE (CAST(r.reservationTime AS date) = @givenDay)
12 End
13 GO
14
15 Exec Sp_GetReservationOfTheDay '2021-01-01'
```

	FullName	reservationTime
1	SüleymanYoz...	2021-01-01 22:30:...
2	ErsinÖztekin	2021-01-01 10:10:...
3	EmanSan	2021-01-01 10:30:...

## 6- Name: Sp\_GetReservationOfTheCustomer

Def: This SP returns Reservation of the given customer



```
1 Create Proc Sp_GetReservationOfTheCustomer
2     @customerName nvarchar(25), @customerLastname nvarchar(25)
3 As
4 Begin
5     Set Nocount On
6
7     If Exists(Select * From Customer c Inner Join Receipt rc
8              On c.customerID=rc.customerID Inner Join Reservation r
9              On rc.receiptID=r.receiptID
10             Where c.firstName like @customerName and c.lastName like @customerLastname)
11     Begin
12         SELECT c.firstName + ' ' + c.lastName AS FullName, r.reservationTime
13         FROM Reservation r Inner Join Receipt rc
14              ON r.receiptID = rc.receiptID Inner Join Customer c
15              ON rc.customerID = c.customerID
16         WHERE (c.firstName like @customerName and c.lastName like @customerLastname)
17     End
18     Else
19     Begin
20         Select @customerName + ' ' + @customerLastname FullName, NULL as ReservationTime
21     End
22 End
23 GO
24 Exec Sp_GetReservationOfTheCustomer 'hakan', 'sarı'
```

	FullName	reservationTime
1	Hakan San	2020-12-26 10:30:...

## 7- Name: Sp\_GetTotalSalaryOfGivenStaffType

Def: This SP returns the salary of given staffType

The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows the definition of a stored procedure named `Sp_GetTotalSalaryOfGivenStaffType`. The procedure takes two parameters: `@staffType char(1)` and `@totalSalary int Output`. The logic involves setting `Nocount On`, selecting the sum of salaries for a given staff type, and returning the result. The bottom pane shows the execution results, which include a table with one row containing the value `10040`.

```
1 Create Proc Sp_GetTotalSalaryOfGivenStaffType
2     @staffType char(1),
3     @totalSalary int Output
4 As
5 Begin
6     Set Nocount On
7     Select @totalSalary = sum(sl.amount)
8         From Staff st Inner Join Salary sl
9             On st.salaryID=sl.salaryID
10        Where st.staffType like @staffType
11
12    Return
13 End
14 GO
15 Declare @totalSalary int
16 select @totalSalary totalSalary
17 Exec Sp_GetTotalSalaryOfGivenStaffType 'b', @totalSalary=@totalSalary Output;
18 select @totalSalary totalSalary
```

totalSalary
1

totalSalary
1

## 8- Name: Sp\_GetIncomeOfGivenMonth

Def: This SP returns the income of given month

The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows the definition of the stored procedure `Sp_GetIncomeOfGivenMonth`. The bottom pane shows the results of the procedure's execution, which returned a single row with the total income for January.

```
SQLQuery13.sql - T...01-1292\enadi (56))* X SQLQuery4.sql - TB...01-1292\enadi (55))*
SQLQuery13.sql - TB701-1292\MYMSSQLSERVER.KESGIN_KUAFOR (TB701-1292\enadi (56))*
2  @month varchar(11),
3  @totalIncome int Output
4  As
5  Begin
6      Set Nocount On
7      Select @totalIncome = sum(r.totalPrice)
8      From Receipt r
9      Where DATENAME(MONTH, Cast(r.date as datetime)) like @month
10
11      Return
12  End
13  GO
14  Declare @totalIncome int
15  select @totalIncome totalIncome
16  Exec Sp_GetIncomeOfGivenMonth 'January', @totalIncome=@totalIncome Output;
17  select @totalIncome totalIncome
18
```

Results

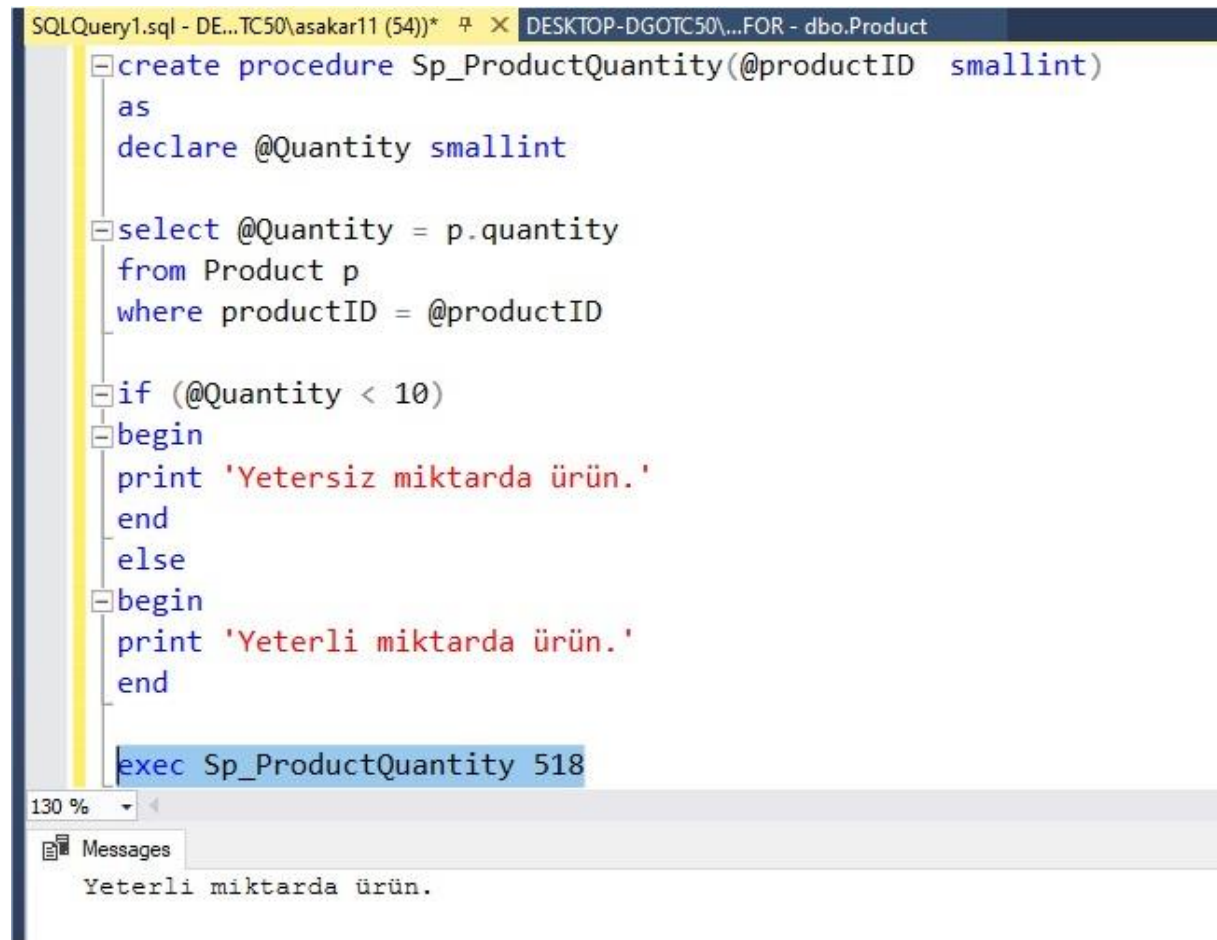
	totalIncome
1	NULL

Messages

	totalIncome
1	359

## 9- Name: Sp\_ProductQuantity

Def: This SP returns if the given product is enough or not



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the SQL code for creating a stored procedure named Sp\_ProductQuantity. The code declares a variable @Quantity, selects its value from the Product table, and then uses an if statement to print a message based on whether the quantity is less than 10. The bottom pane shows the execution of the procedure with the product ID 518, resulting in the message 'Yeterli miktarda ürün.'

```
SQLQuery1.sql - DE...TC50\asakar11 (54))* X DESKTOP-DGOTC50\...FOR - dbo.Product
create procedure Sp_ProductQuantity(@productID smallint)
as
declare @Quantity smallint
select @Quantity = p.quantity
from Product p
where productID = @productID
if (@Quantity < 10)
begin
print 'Yetersiz miktarda ürün.'
end
else
begin
print 'Yeterli miktarda ürün.'
end

exec Sp_ProductQuantity 518
```

130 %

Messages

Yeterli miktarda ürün.