MARMARA UNIVERSITY
FACULTY OF ENGINEERING

DEPARTMENT OF
COMPUTER ENGINEERING

CSE3055 Database Systems Project
- Step 3

Travel Agency

150118015 - Hasan Fatih Başar
150118042 - Bahadır Alacan
150119508 - Mert Sağlam
150119824 - Zeynep Ferah Akkurt

Project Name: Travel Agency

Entities

- **Customer**: the traveler who makes the booking. Customers have a CustomerId as unique identifier, CustomerName, a CustomerSurname, a Mail, a PhoneNumber, an Address (composed of city and state), a VaccinationCard (Covid etc.), a HESCode, a BirthDate, an Age (derived from birthdate), a Gender, and a Visa (Germany, France etc. (County names)). A customer can have more than one VaccinationCard or Visa.
- **Group_Info**: travelers may be in a group of people (company, family event etc.). A group has an GroupId as unique identifier and a GroupName.
- **Passport**: passport information for the customer who is going to a trip abroad. A Passport has an PassportId as unique identifier and an ExpireDate.
- **Booking**: reservation for a trip and services. It has a BookingId as unique identifier, StartDate and EndDate of the booking and a TotalPrice. StartDate and EndDate is the time period for trips and services. For instance, trip starts at February 15, 2020 and ends on February 27, 2020
- **Trip**: organized trip that a travel agency can offer or can create for the customer. Each trip has an TripId as unique identifier, start point as StartLocation, end point as Destination, Price (except transportation and accommodation), and an information about if it is inside of the country or out as isAbroad.
- **Payment**: customer payment information for the trip. Payment has a PaymentId as unique identifier, Amount of payment, Date, and a payment Type (cash or installment etc.)
- **Service**: any kind of additional service to give to a customer. A service can be a Hotel reservation, Flight, or a Car. Services have a ServiceId as unique identifier, Price and ServiceType. ServiceType is a subtype discriminator
- **Hotel**: customer's place to stay along the trip. Hotels have a HotelName, an Address (composed of BuildingNo, Street, City and Country).
- **Flight:** customer flight information. Flights have an AirlineCompany, FlightDate, FlightTime, departure as FromAirport, arrival as ToAirport.
- **Car:** customer may want to rent a car. Cars has a carType (panelvan, automobile etc.) and a CarModel (renault clio, bmw 3 etc.).

SCOPE

The aim in this project is to create a database for a travel agency. An agency keeps track of the customers and their bookings along with lots of services. The database has 16 tables with 3 Trigger, 10 stored procedures and 5 views. Each stored procedures are the functions for calculating/listing some customer or booking feeatures based on the request(inputs to the function.) Each view shows another perspective of the tables for instance customers who has not finished their payment of their bookings. The most important table in the database is Booking. Because it keeps the records for everyone and that is why it is like a bus to everywhere. Each booking can belong to a trip or a service. Bookings of a customer and groups keeps in a different table.

Business processes

Travel agency makes necessary arrangements for travelers. They organize your trip according to your requests and wishes, so it is essential to keep all those records inside of a system. A group of customer or maybe just one person can book a trip along with other services like flight ticket, hotel reservations. Travel agency has organized trips to offer. It also reaches and reserves a plane ticket and a hotel room for the customer's trip (depending on customers' choices. There might be several hotel and flight options.). A customer can also rent a car.

Business Rules and Constraints

- A Customer is person.
- Customer may be member of Group (Family or company).
- Customer can make same Booking with his/her Group members or her/himself.
- Each Customer makes a Payment for the booking.
- Passport is required from the Customer who goes to a trip which is abroad. A customer can only have one passport information. And it may be required.
- A customer can book many Trips and Booking Trip can belong to more than one customer or a group.
- A Trip is offered by travel agency. An agency can offer many trips.
- Travel agency can provide different kinds of services. These services can be Hotel reservation, Flight tickets, renting a Car.
- A Booking can include one or more services. A Service can be reserved/organized for many bookings.
- A customer makes a Payment to book a Trip, a Payment can be made only for one Booking.

Functional & non-functional business requirements 1. Functional Requirements

- Customer logs into the system
- System shows some options to customer trips.
- Customer chooses what he/she wants
- Customer decides the services (hotel reservation, flight tickets etc.) or trip from the agency such as:
    - The system checks for the available hotels and offers to the customer. Customer chooses based on those offers.
    - The system checks for the flight with a given destination and departure location, system may ask for passport if the destination is international
    - Systems shows the cars that can be rent.
- System gives output to customer depends on his/her request
- If everything is okay, then customer provides personal information's for booking the trip that is organized then jumps to the payment process
- System asks for payment info
- After payment is done, booking is successfully

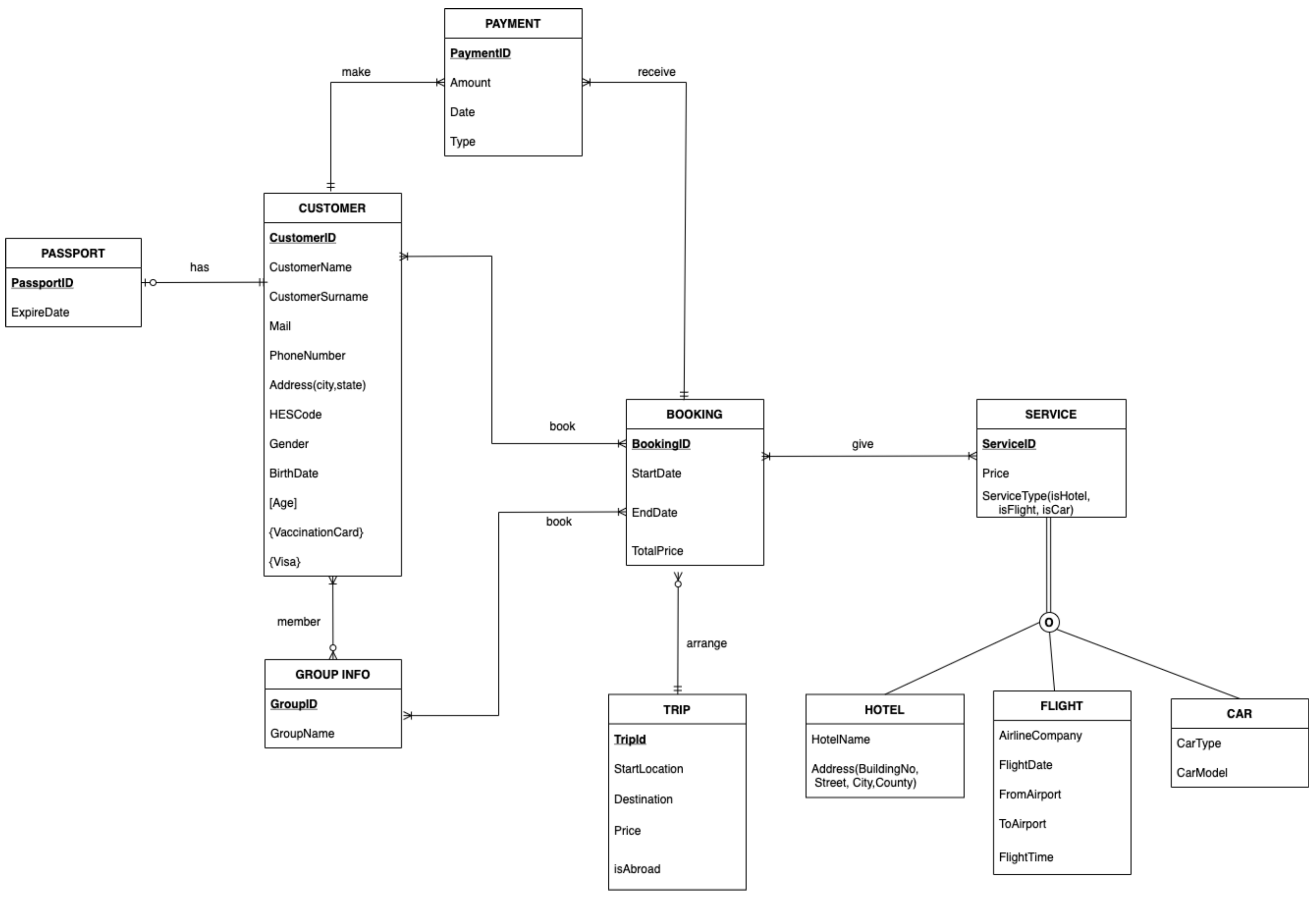
2. Non-functional Requirements

- Performance
- Scalability
- Capacity
- Availability
- Reliability
- Recoverability
- Maintainability
- Serviceability

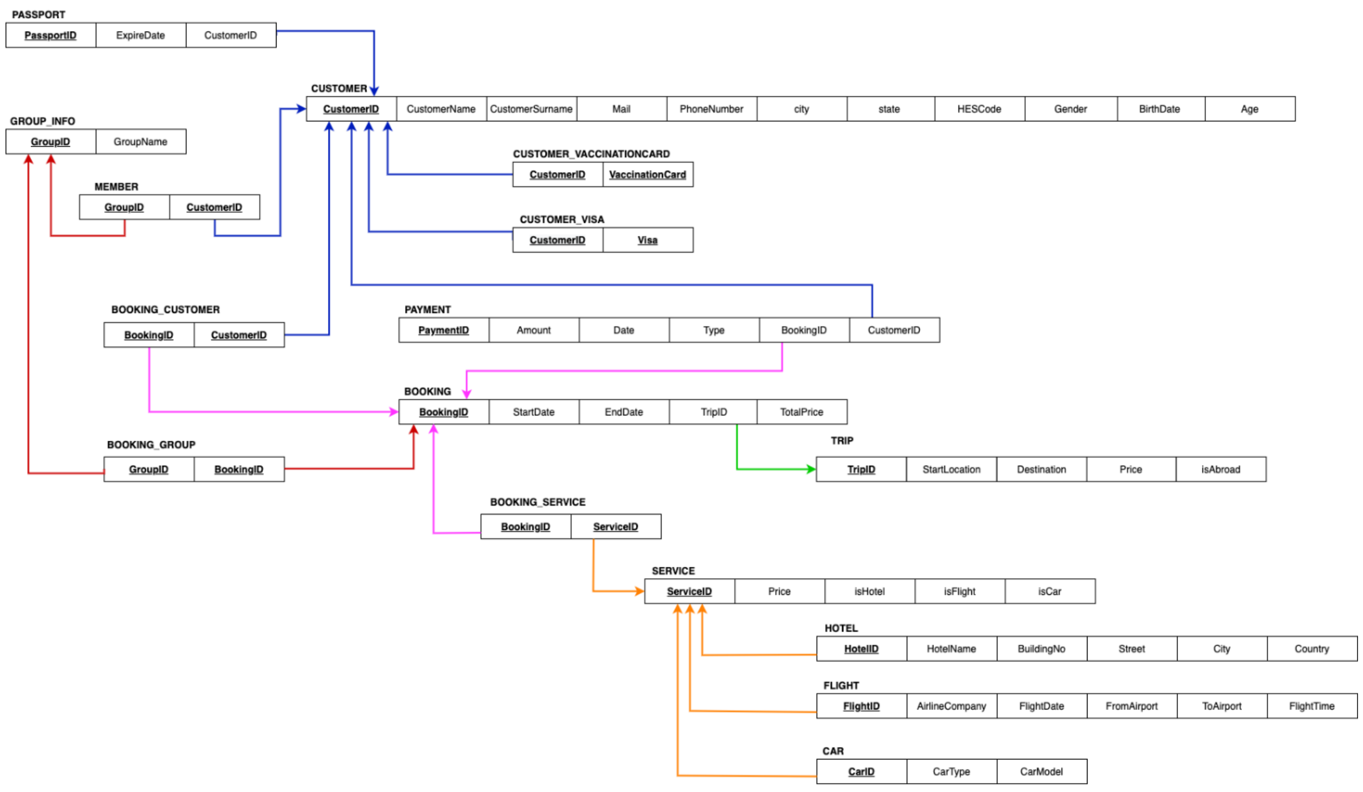
Customer Related Documents

In this project, our customer is an actual travelling agency (STH Travel) employee who is willing to guide us about how their booking system works.

ER Diagram



Tables



Customer: All information about our customers is kept in this table, such as CustomerID, CustomerName, CustomerSurname, customer's mail, customer's phone number, city, state, HES code, Gender, birthdate, age. Age variable is computed by the difference of today's date and customer's birthdate.

	CustomerID	CustomerName	CustomerSurname	Mail	PhoneNumber
Data type	bigint	nvarchar(50)	nvarchar(50)	nvarchar(50)	nvarchar(50)

	City	State	HESCode	Gender	BirthDate	Age
Data type	nvarchar(50)	nvarchar(50)	nvarchar(50)	char(1)	smalldatetime	Computed

- Primary key: CustomerID
- Indexes: Age
- Unique index: HESCode can not be null
- Constraint: Gender (F for female or M for male)
- Default: State is by default 'Türkiye'
- Computed: Age is derived from BirthDate
- Composite: City, State

Customer_Vaccination_Card: This table keeps track of the vaccination cards of who had vaccinated. A customer can have more than one vaccination. This table is created this because customer's vaccination card is multivalued data type in our ER diagram.

	CustomerID	VaccinationCard
Data type	bigint	nvarchar(50)

- Primary key: CustomerID, VaccinationCard
- Foreign Key: CustomerID (Table Customer)

Customer_Visa: This table keeps track of the visa type who had visa among our customers. This table is created because customer's visa is multivalued data type in our ER diagram.

	CustomerID	Visa
Data type	bigint	nvarchar(50)

- Primary key: CustomerID, Visa
- Foreign Key: CustomerID (Table Customer)

Passport: Customers who is going abroad has passport and their passports are unique for each customer. That's why we have a passport id in this table.

	PassportID	ExpireDate	CustomerID
Data type	int	smalldatetime	bigint

- Primary key: PassportID
- Foreign Key: CustomerID (to Table Customer)

Group_Info: Families, societies, national teams, companies etc. are counted as groups. That's why we keep track of all those things.

	GroupID	GroupName
Data type	int	nvarchar(50)

- Primary key: GroupID

Member: This table indicates the member of groups by their group id.

	GroupID	CustomerID
Data type	int	bigint

- Primary key: GroupID, CustomerID
- Foreign Key: GroupID (to Table Group) and CustomerID (to Table Customer)

Trip: Travel agency can arrange some trips such as Balkan tour, Africa tour, Europe tour. We keep track of start location and destination of our trip. Price is stored as well.

	TripID	StartLocation	Destination	Price	isAbroad
Data type	int	nvarchar(50)	nvarchar(50)	int	bit

- Primary key: TripID
- Identities: TripID increases 1 by 1 for each time.
- Trigger: When a trip is booked, the TotalPrice in the Booking table will be updated by adding the price of the trip.
- Destination can not be null

Service: Since our travel agency can serve many services such as hotel reservation, flight tickets, car rental, we keep track of them. Customer may want more than one service. For example, he wants to reserve a room in a hotel in Erzurum also he wants to buy flight tickets from Istanbul to Erzurum. if he wants to rent a car in Erzurum, travel agency can arrange it for the customer.

	ServiceID	Price	isHotel	isFlight	isCar
Data type	int	int	bit	bit	bit

- Primary key: ServiceID
- isHotel, isFlight ,isCar are subtype discriminator elements. A service can be hotel, flight or car.
- Trigger: When service price has updated, booking TotalPrice has changed.

Hotel: Agency can serve hotel reservation as service. The hotels that can be served are in this table, no matter abroad or domestic. Each hotel is a service.

	HotelID	HotelName	BuildingNo	Street	City	Country
Data type	int	nvarchar(50)	Smallint	nvarchar(50)	nvarchar(50)	nvarchar(50)

- Primary key: HotelID
- Foreign key: HotelId (ServideID in Service Table)
- Composite: Street, City, BuildingNo, Country

Flight: Agency can arrange flight tickets for customers, we keep track of flights as a service. Those flights can be abroad or domestic. Flight time is the time when plane will take off. From airport and arrival airports are kept as well.

	FlightID	AirlineCompany	FlightDate	FromAirport	toAirport	FlightTime
Data type	int	nvarchar(50)	smalldatetime	nvarchar(50)	nvarchar(50)	time(0)

- Primary key: FlightID
- Foreign key: FligthID (ServideID in Service Table)

Car: Cars which is made rent by our travel agency to customers are kept in this table.

	CarID	CarType	CarModel
Data type	int	nvarchar(50)	nvarchar(50)

- Primary key: CarID
- Foreign key: CarID (ServideID in Service Table)

Booking: Track of booking identifier, booking start and end date, trip id and total price of booking is kept in Booking table.

	BookingID	StartDate	EndDate	TripID	TotalPrice
Data type	int	smalldatetime	smalldatetime	int	int

- Primary key: BookingID
- Foreign key: TripId (Table Trip)
- Trigger: TotalPrice is updated each time a service or a trip(just one) is added or removed. A booking can include many services.
- Start and End Date can not be null.

Booking_Customer: This table keeps track of customers who made booking only individually.

	BookingID	CustomerID
Data type	int	bigint

- Primary key: BookingId and CustomerID
- Foreign key: BookingId (to Table Booking) and CustomerID (to Table Customer)

Booking_Group: Track of customers who made booking with group or family etc. are kept in Booking_Group table.

	BookingID	GroupID
Data type	int	int

- Primary key: BookingId and CustomerID
- Foreign key: BookingId (Table Booking) and CustomerID (Table Customer)

Booking_Service: This table keeps track of all bookings for services.

	BookingID	ServiceID
Data type	int	int

- Primary key: BookingId and ServiceID
- Foreign key: BookingId (Table Booking) and ServiceID (Table Service)

- Trigger: when a service is booked the total price in Booking Table will be updated by adding the service price.

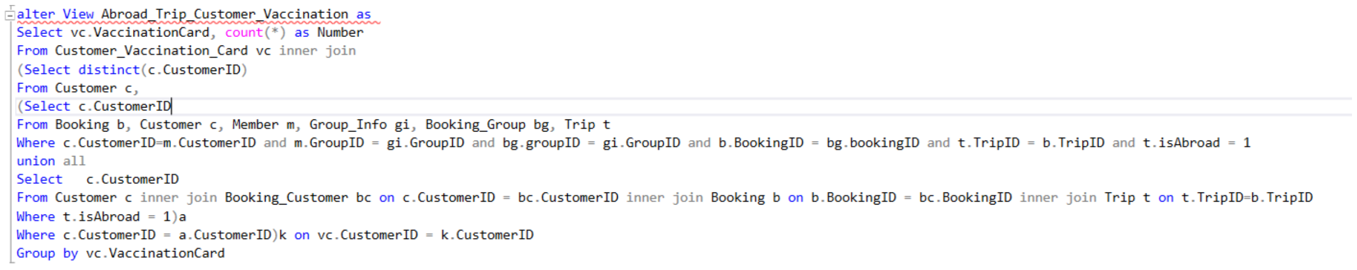
Payment: All customers have to pay for the services and prices. The payment type can be cash or installment. We keep track of the payment date and amount in this table.

	PaymentID	Amount	Date	Type	BookingID	CustomerID
Data type	int	bigint	smalldatetime	nvarchar(50)	int	bigint

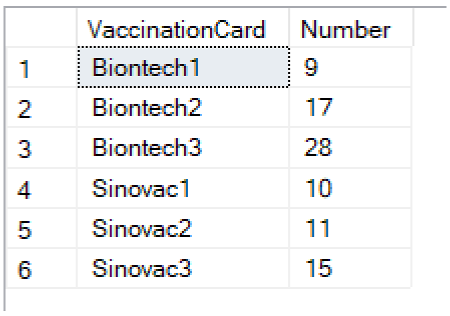
- Primary key: PaymentID
- Foreign key: BookingID (to Table Booking) and CustomerID (to Table Customer)

D.) VIEWS

1.) Abroad_Trip_Customer_Vaccination:
It groups customers who went trip to abroad by their vaccinations.



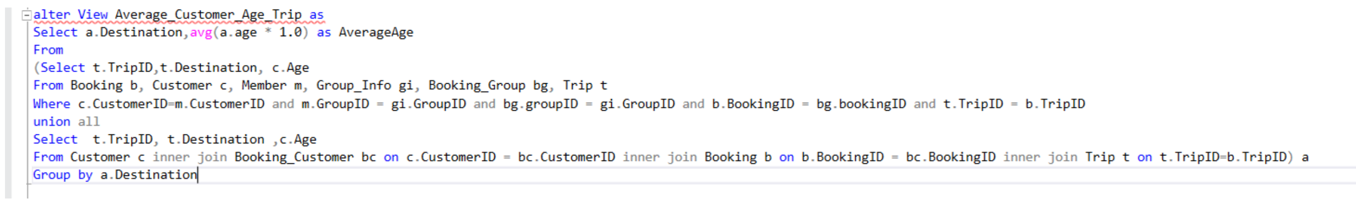
```sql
alter View Abroad_Trip_Customer_Vaccination as
Select vc.VaccinationCard, count(*) as Number
From Customer_Vaccination_Card vc inner join
(Select distinct(c.CustomerID)
From Customer c,
(Select c.CustomerID
From Booking b, Customer c, Member m, Group_Info gi, Booking_Group bg, Trip t
Where c.CustomerID=m.CustomerID and m.GroupID = gi.GroupID and bg.groupID = gi.GroupID and b.BookingID = bg.bookingID and t.TripID = b.TripID and t.isAbroad = 1
union all
Select   c.CustomerID
From Customer c inner join Booking_Customer bc on c.CustomerID = bc.CustomerID inner join Booking b on b.BookingID = bc.BookingID inner join Trip t on t.TripID=b.TripID
Where t.isAbroad = 1)a
Where c.CustomerID = a.CustomerID)k on vc.CustomerID = k.CustomerID
Group by vc.VaccinationCard
```



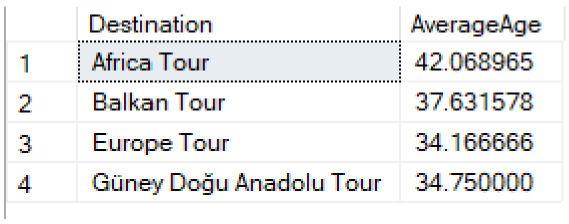
	VaccinationCard	Number
1	Biontech1	9
2	Biontech2	17
3	Biontech3	28
4	Sinovac1	10
5	Sinovac2	11
6	Sinovac3	15

2.) Average_Customer_Age_Trip:

It calculates average customer age by trip destination.



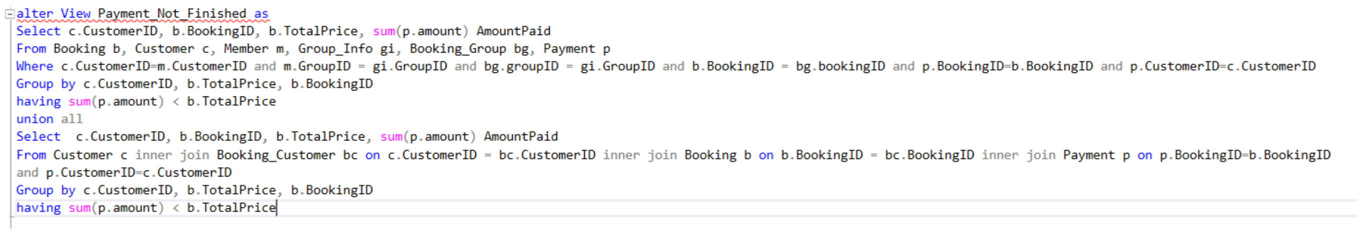
```sql
alter View Average_Customer_Age_Trip as
Select a.Destination,avg(a.age * 1.0) as AverageAge
From
(Select t.TripID,t.Destination, c.Age
From Booking b, Customer c, Member m, Group_Info gi, Booking_Group bg, Trip t
Where c.CustomerID=m.CustomerID and m.GroupID = gi.GroupID and bg.groupID = gi.GroupID and b.BookingID = bg.bookingID and t.TripID = b.TripID
union all
Select  t.TripID, t.Destination ,c.Age
From Customer c inner join Booking_Customer bc on c.CustomerID = bc.CustomerID inner join Booking b on b.BookingID = bc.BookingID inner join Trip t on t.TripID=b.TripID) a
Group by a.Destination
```



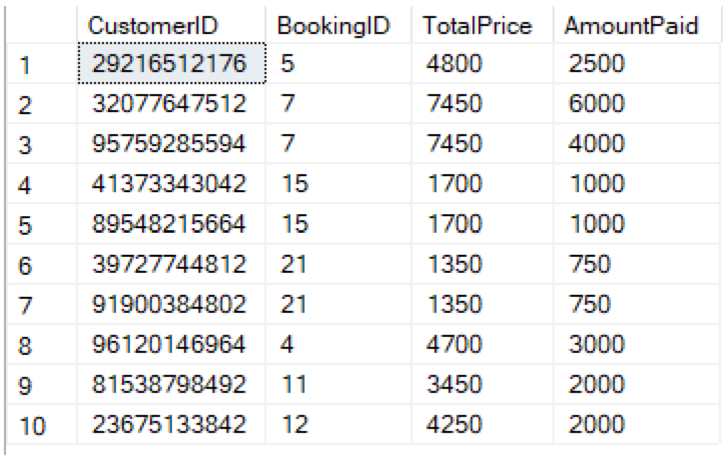
	Destination	AverageAge
1	Africa Tour	42.068965
2	Balkan Tour	37.631578
3	Europe Tour	34.166666
4	Güney Doğu Anadolu Tour	34.750000

3.) Payment_Not_Finished

It shows customers who has not finished their payment and their paid amount.

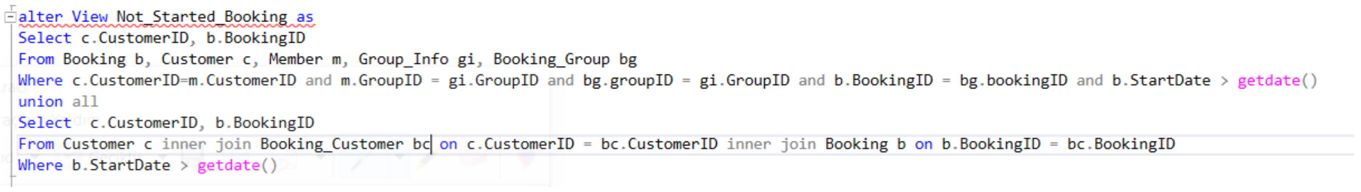


```sql
alter View Payment_Not_Finished as
Select c.CustomerID, b.BookingID, b.TotalPrice, sum(p.amount) AmountPaid
From Booking b, Customer c, Member m, Group_Info gi, Booking_Group bg, Payment p
Where c.CustomerID=m.CustomerID and m.GroupID = gi.GroupID and bg.groupID = gi.GroupID and b.BookingID = bg.bookingID and p.BookingID=b.BookingID and p.CustomerID=c.CustomerID
Group by c.CustomerID, b.TotalPrice, b.BookingID
having sum(p.amount) < b.TotalPrice
union all
Select  c.CustomerID, b.BookingID, b.TotalPrice, sum(p.amount) AmountPaid
From Customer c inner join Booking_Customer bc on c.CustomerID = bc.CustomerID inner join Booking b on b.BookingID = bc.BookingID inner join Payment p on p.BookingID=b.BookingID
and p.CustomerID=c.CustomerID
Group by c.CustomerID, b.TotalPrice, b.BookingID
having sum(p.amount) < b.TotalPrice
```

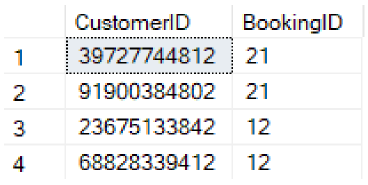


	CustomerID	BookingID	TotalPrice	AmountPaid
1	29216512176	5	4800	2500
2	32077647512	7	7450	6000
3	95759285594	7	7450	4000
4	41373343042	15	1700	1000
5	89548215664	15	1700	1000
6	39727744812	21	1350	750
7	91900384802	21	1350	750
8	96120146964	4	4700	3000
9	81538798492	11	3450	2000
10	23675133842	12	4250	2000

4.) Not_Started_Booking
It shows booking which customer book but booking has not started.

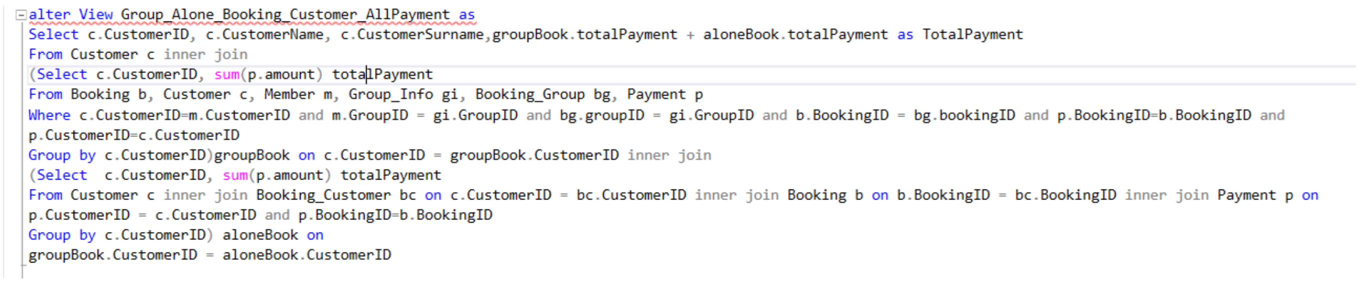


```
alter View Not_Started_Booking as
Select c.CustomerID, b.BookingID
From Booking b, Customer c, Member m, Group_Info gi, Booking_Group bg
Where c.CustomerID=m.CustomerID and m.GroupID = gi.GroupID and bg.groupID = gi.GroupID and b.BookingID = bg.bookingID and b.StartDate > getdate()
union all
Select  c.CustomerID, b.BookingID
From Customer c inner join Booking_Customer bc on c.CustomerID = bc.CustomerID inner join Booking b on b.BookingID = bc.BookingID
Where b.StartDate > getdate()
```

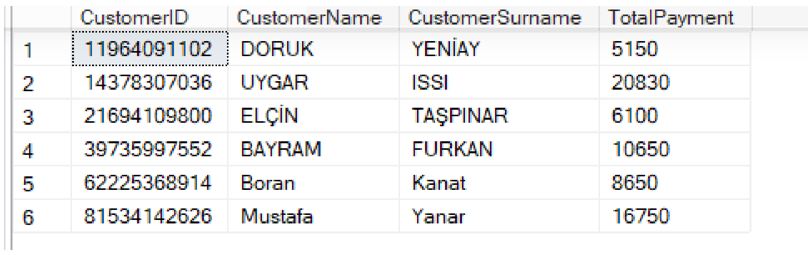


	CustomerID	BookingID
1	39727744812	21
2	91900384802	21
3	23675133842	12
4	68828339412	12

5.) Group_Alone_Booking_Customer_AllPayment
This code list the total payment who made booking either individually or with group



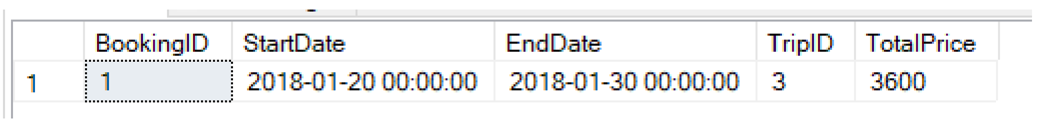
```
alter View Group_Alone_Booking_Customer_AllPayment as
Select c.CustomerID, c.CustomerName, c.CustomerSurname,groupBook.totalPayment + aloneBook.totalPayment as TotalPayment
From Customer c inner join
(Select c.CustomerID, sum(p.amount) totalPayment
From Booking b, Customer c, Member m, Group_Info gi, Booking_Group bg, Payment p
Where c.CustomerID=m.CustomerID and m.GroupID = gi.GroupID and bg.groupID = gi.GroupID and b.BookingID = bg.bookingID and p.BookingID=b.BookingID and
p.CustomerID=c.CustomerID
Group by c.CustomerID)groupBook on c.CustomerID = groupBook.CustomerID inner join
(Select  c.CustomerID, sum(p.amount) totalPayment
From Customer c inner join Booking_Customer bc on c.CustomerID = bc.CustomerID inner join Booking b on b.BookingID = bc.BookingID inner join Payment p on
p.CustomerID = c.CustomerID and p.BookingID=b.BookingID
Group by c.CustomerID) aloneBook on
groupBook.CustomerID = aloneBook.CustomerID
```



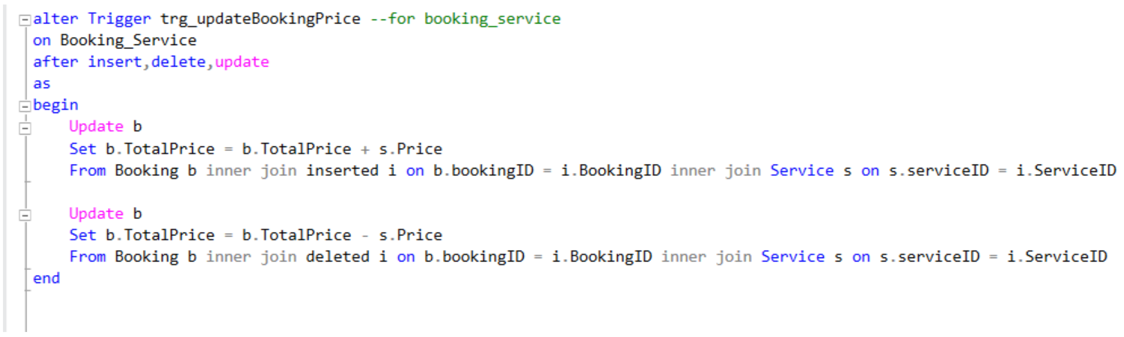
	CustomerID	CustomerName	CustomerSurname	TotalPayment
1	11964091102	DORUK	YENİAY	5150
2	14378307036	UYGAR	ISSI	20830
3	21694109800	ELÇİN	TAŞPINAR	6100
4	39735997552	BAYRAM	FURKAN	10650
5	62225368914	Boran	Kanat	8650
6	81534142626	Mustafa	Yanar	16750

E.) TRIGGER

1.)

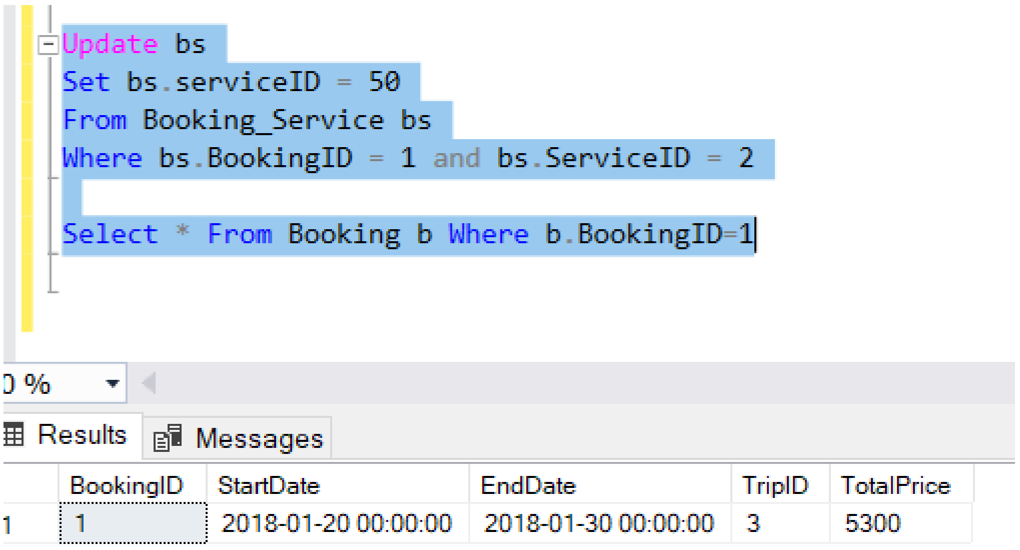


	BookingID	StartDate	EndDate	TripID	TotalPrice
1	1	2018-01-20 00:00:00	2018-01-30 00:00:00	3	3600



```
alter Trigger trg_updateBookingPrice --for booking_service
on Booking_Service
after insert,delete,update
as
begin
    Update b
    Set b.TotalPrice = b.TotalPrice + s.Price
    From Booking b inner join inserted i on b.bookingID = i.BookingID inner join Service s on s.serviceID = i.ServiceID

    Update b
    Set b.TotalPrice = b.TotalPrice - s.Price
    From Booking b inner join deleted i on b.bookingID = i.BookingID inner join Service s on s.serviceID = i.ServiceID
end
```

We add trigger to booking service when we update insert or delete bookingID or serviceID on booking service it updates the value of totalPrice on booking.

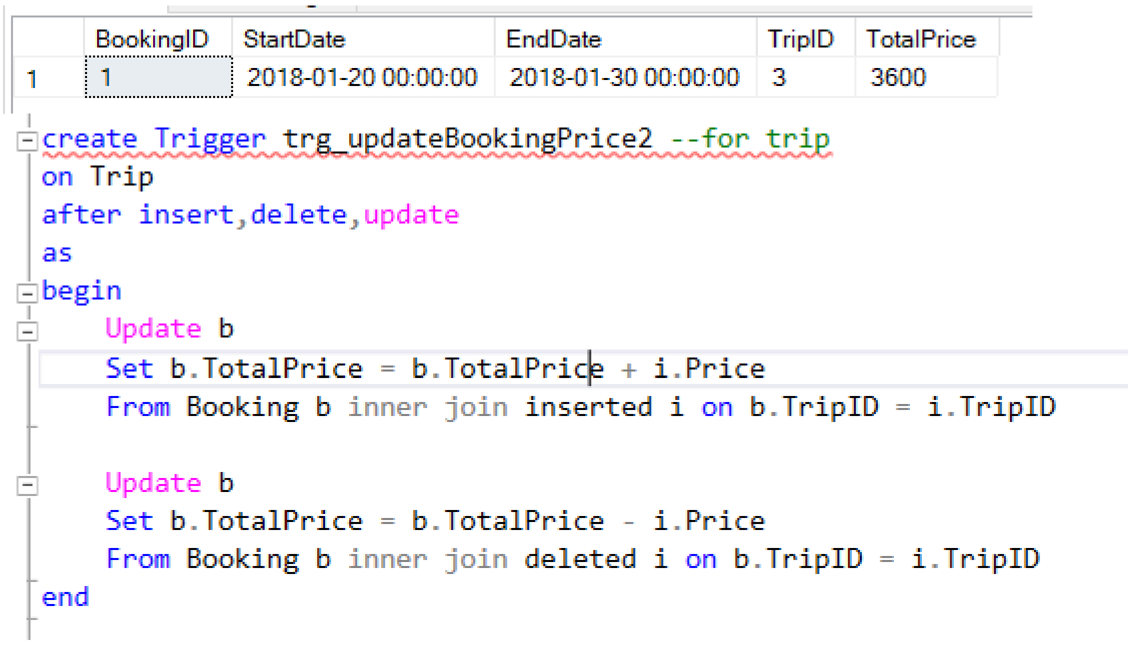


```
Update bs
Set bs.serviceID = 50
From Booking_Service bs
Where bs.BookingID = 1 and bs.ServiceID = 2

Select * From Booking b Where b.BookingID=1
```

	BookingID	StartDate	EndDate	TripID	TotalPrice
1	1	2018-01-20 00:00:00	2018-01-30 00:00:00	3	5300

Change the service id of booking 1.

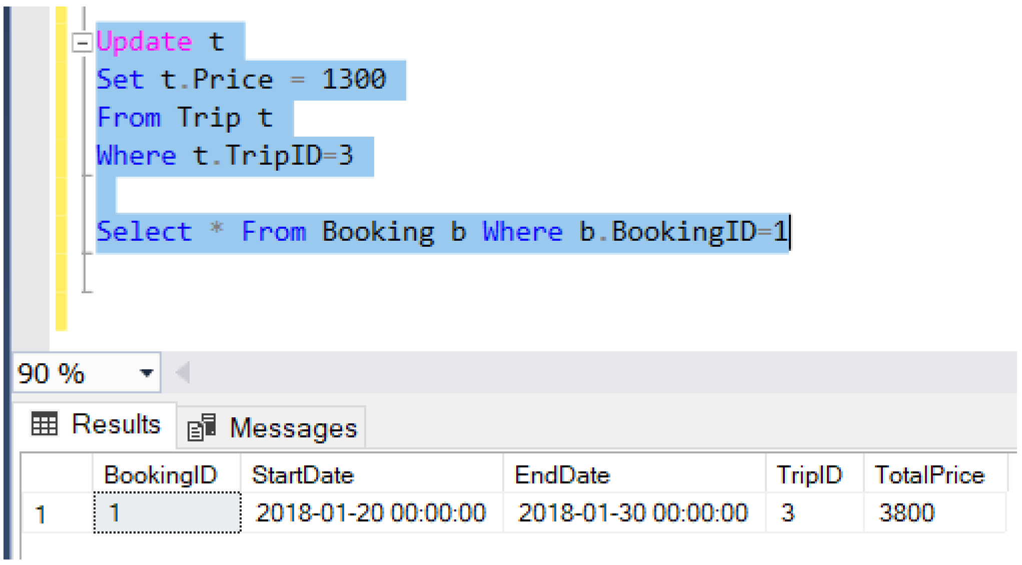
2.)



	BookingID	StartDate	EndDate	TripID	TotalPrice
1	1	2018-01-20 00:00:00	2018-01-30 00:00:00	3	3600

```
create Trigger trg_updateBookingPrice2 --for trip
on Trip
after insert,delete,update
as
begin
    Update b
    Set b.TotalPrice = b.TotalPrice + i.Price
    From Booking b inner join inserted i on b.TripID = i.TripID

    Update b
    Set b.TotalPrice = b.TotalPrice - i.Price
    From Booking b inner join deleted i on b.TripID = i.TripID
end
```

We add trigger to trip when its price change booking's total price has changed as well.



```
Update t
Set t.Price = 1300
From Trip t
Where t.TripID=3

Select * From Booking b Where b.BookingID=1
```

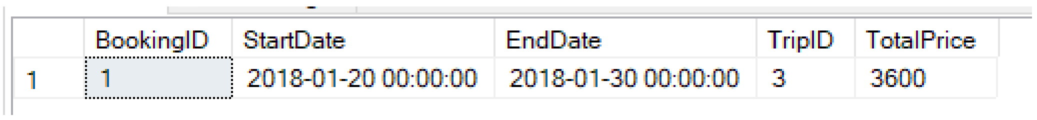
90 %

Results | Messages

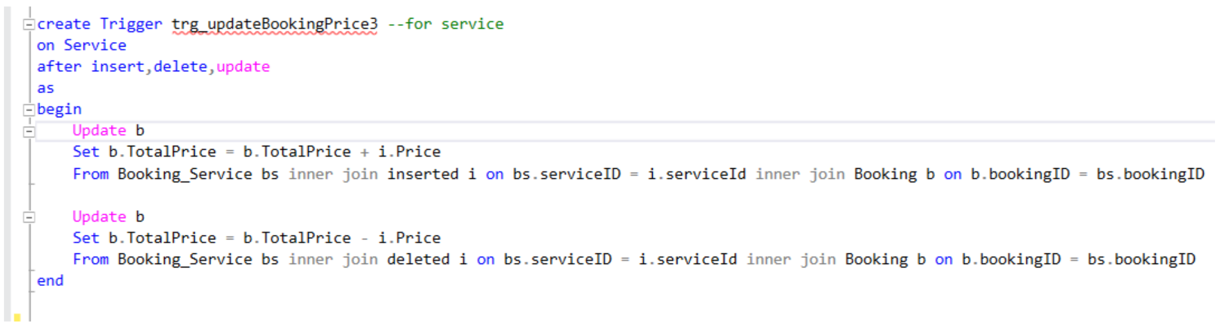
	BookingID	StartDate	EndDate	TripID	TotalPrice
1	1	2018-01-20 00:00:00	2018-01-30 00:00:00	3	3800

Change the trip price 1100 to 1300.

3.)

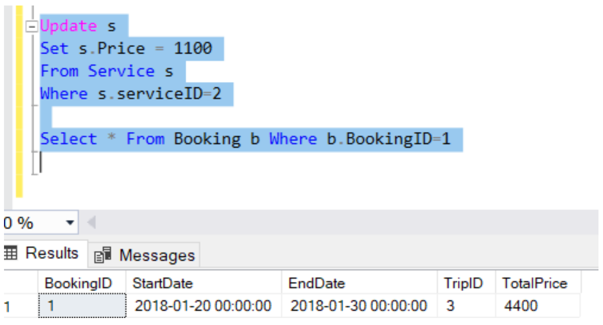


	BookingID	StartDate	EndDate	TripID	TotalPrice
1	1	2018-01-20 00:00:00	2018-01-30 00:00:00	3	3600



```sql
create Trigger trg_updateBookingPrice3 --for service
on Service
after insert,delete,update
as
begin
    Update b
    Set b.TotalPrice = b.TotalPrice + i.Price
    From Booking_Service bs inner join inserted i on bs.serviceID = i.serviceId inner join Booking b on b.bookingID = bs.bookingID

    Update b
    Set b.TotalPrice = b.TotalPrice - i.Price
    From Booking_Service bs inner join deleted i on bs.serviceID = i.serviceId inner join Booking b on b.bookingID = bs.bookingID
end
```

We add trigger to service when its price change booking's total price has changed as well.



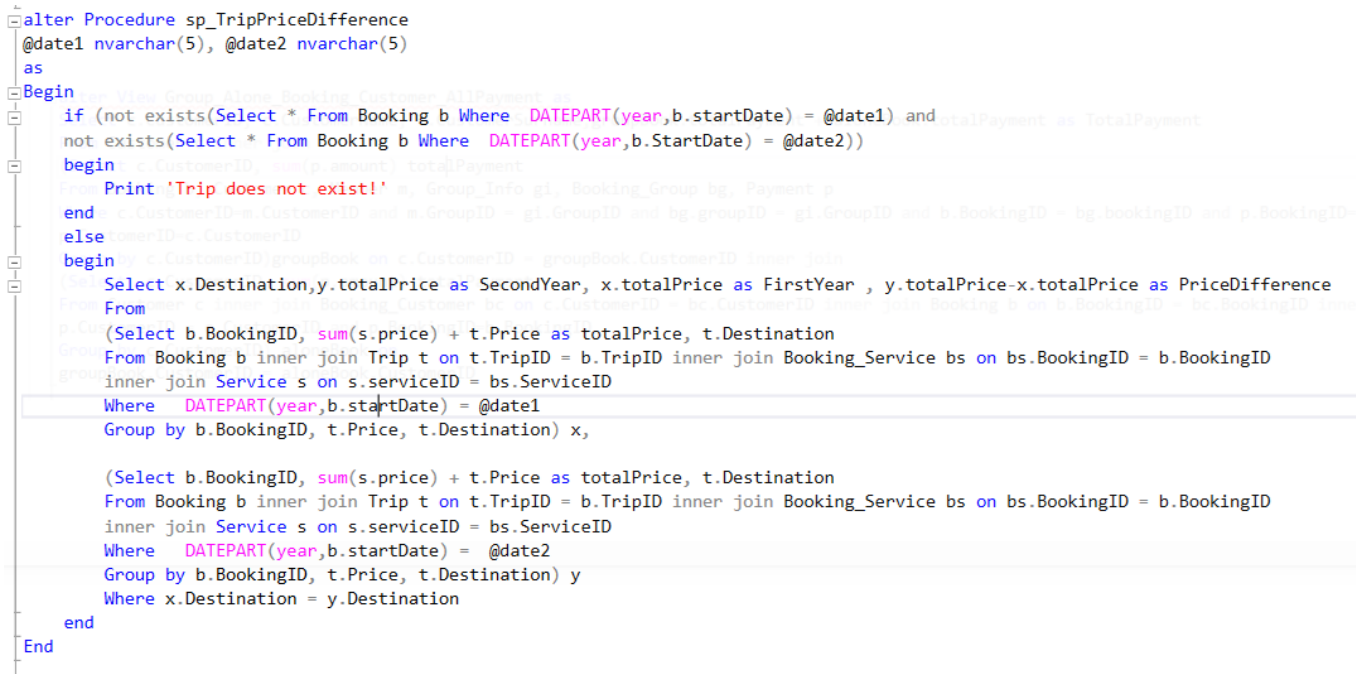
```sql
Update s
Set s.Price = 1100
From Service s
Where s.serviceID=2

Select * From Booking b Where b.BookingID=1
```

```
0 %
Results   Messages
   BookingID  StartDate            EndDate              TripID  TotalPrice
1  1          2018-01-20 00:00:00  2018-01-30 00:00:00  3       4400
```

We changed service price 300 to 1100.

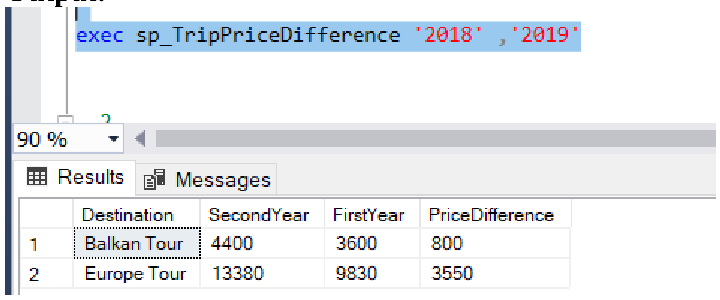
F.)Stored procedures

1.) sp_TripPriceDifference
Inputs: Two different dates of the same trip (Ex: same trip one year later)
For the given years, it shows the price difference of same trip bookings. If trip doesn't arrange in given years, it doesn't show.



```sql
alter Procedure sp_TripPriceDifference
@date1 nvarchar(5), @date2 nvarchar(5)
as
Begin
    if (not exists(Select * From Booking b Where  DATEPART(year,b.startDate) = @date1) and
    not exists(Select * From Booking b Where  DATEPART(year,b.StartDate) = @date2))
    begin
        Print 'Trip does not exist!'
    end
    else
    begin
        Select x.Destination,y.totalPrice as SecondYear, x.totalPrice as FirstYear , y.totalPrice-x.totalPrice as PriceDifference
        From
        (Select b.BookingID, sum(s.price) + t.Price as totalPrice, t.Destination
        From Booking b inner join Trip t on t.TripID = b.TripID inner join Booking_Service bs on bs.BookingID = b.BookingID
        inner join Service s on s.serviceID = bs.ServiceID
        Where   DATEPART(year,b.startDate) = @date1
        Group by b.BookingID, t.Price, t.Destination) x,

        (Select b.BookingID, sum(s.price) + t.Price as totalPrice, t.Destination
        From Booking b inner join Trip t on t.TripID = b.TripID inner join Booking_Service bs on bs.BookingID = b.BookingID
        inner join Service s on s.serviceID = bs.ServiceID
        Where   DATEPART(year,b.startDate) =  @date2
        Group by b.BookingID, t.Price, t.Destination) y
        Where x.Destination = y.Destination
    end
End
```

Output:

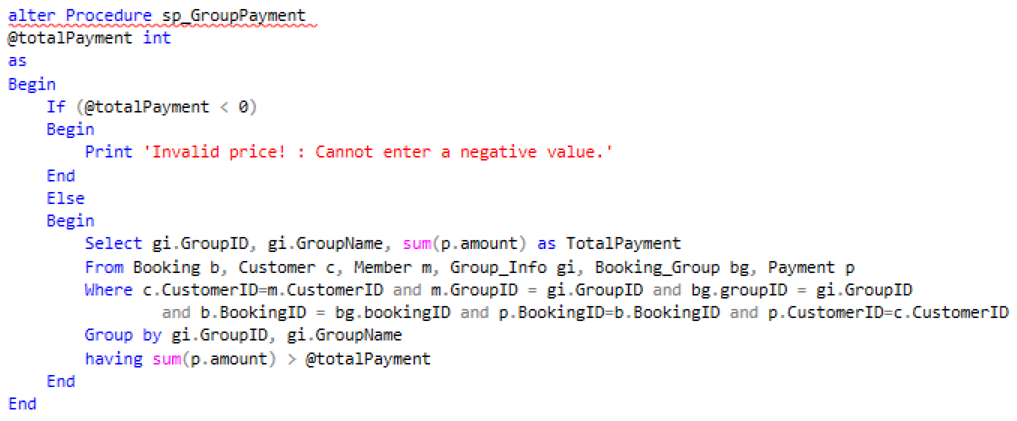


```sql
exec sp_TripPriceDifference '2018' ,'2019'
```

	Destination	SecondYear	FirstYear	PriceDifference
1	Balkan Tour	4400	3600	800
2	Europe Tour	13380	9830	3550

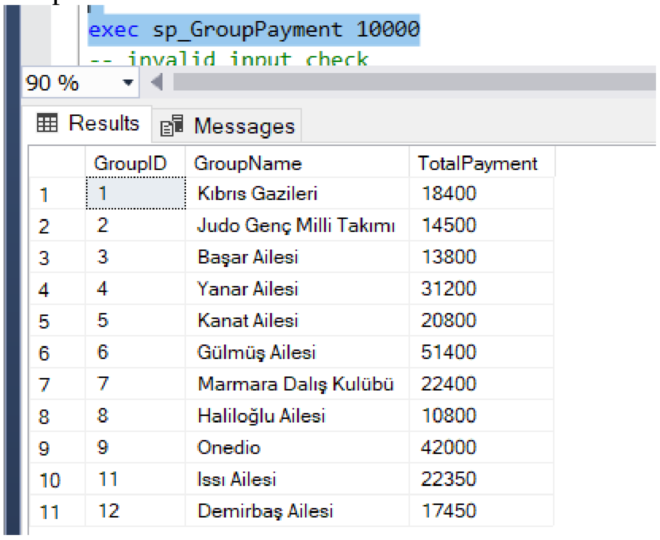
2.) sp_GroupPayment

Input: Minimum price for the group
It shows the groups which totalPayment is bigger than given input.



```
alter Procedure sp_GroupPayment
@totalPayment int
as
Begin
    If (@totalPayment < 0)
    Begin
        Print 'Invalid price! : Cannot enter a negative value.'
    End
    Else
    Begin
        Select gi.GroupID, gi.GroupName, sum(p.amount) as TotalPayment
        From Booking b, Customer c, Member m, Group_Info gi, Booking_Group bg, Payment p
        Where c.CustomerID=m.CustomerID and m.GroupID = gi.GroupID and bg.groupID = gi.GroupID
                and b.BookingID = bg.bookingID and p.BookingID=b.BookingID and p.CustomerID=c.CustomerID
        Group by gi.GroupID, gi.GroupName
        having sum(p.amount) > @totalPayment
    End
End
```

Output:

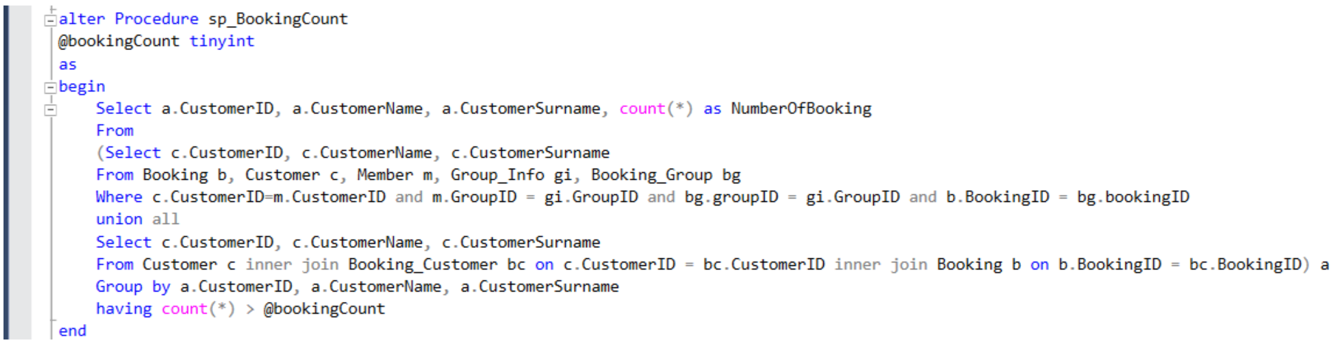


```
exec sp_GroupPayment 10000
-- invalid input check
```

	GroupID	GroupName	TotalPayment
1	1	Kıbrıs Gazileri	18400
2	2	Judo Genç Milli Takımı	14500
3	3	Başar Ailesi	13800
4	4	Yanar Ailesi	31200
5	5	Kanat Ailesi	20800
6	6	Gülmüş Ailesi	51400
7	7	Marmara Dalış Kulübü	22400
8	8	Haliloğlu Ailesi	10800
9	9	Onedio	42000
10	11	Issı Ailesi	22350
11	12	Demirbaş Ailesi	17450

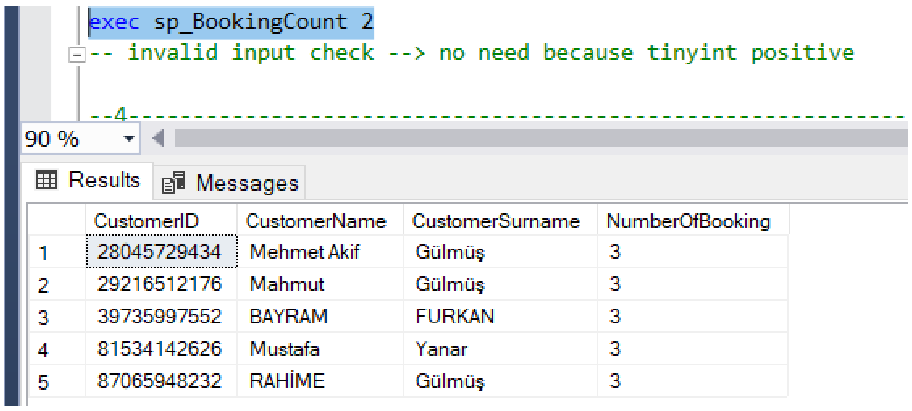
3.) sp_BookingCount

It shows the customer who book more than given input.



```
alter Procedure sp_BookingCount
@bookingCount tinyint
as
begin
    Select a.CustomerID, a.CustomerName, a.CustomerSurname, count(*) as NumberOfBooking
    From
    (Select c.CustomerID, c.CustomerName, c.CustomerSurname
    From Booking b, Customer c, Member m, Group_Info gi, Booking_Group bg
    Where c.CustomerID=m.CustomerID and m.GroupID = gi.GroupID and bg.groupID = gi.GroupID and b.BookingID = bg.bookingID
    union all
    Select c.CustomerID, c.CustomerName, c.CustomerSurname
    From Customer c inner join Booking_Customer bc on c.CustomerID = bc.CustomerID inner join Booking b on b.BookingID = bc.BookingID) a
    Group by a.CustomerID, a.CustomerName, a.CustomerSurname
    having count(*) > @bookingCount
end
```

Output:



```
exec sp_BookingCount 2
-- invalid input check --> no need because tinyint positive

--4----------------------------------------------------------------
```

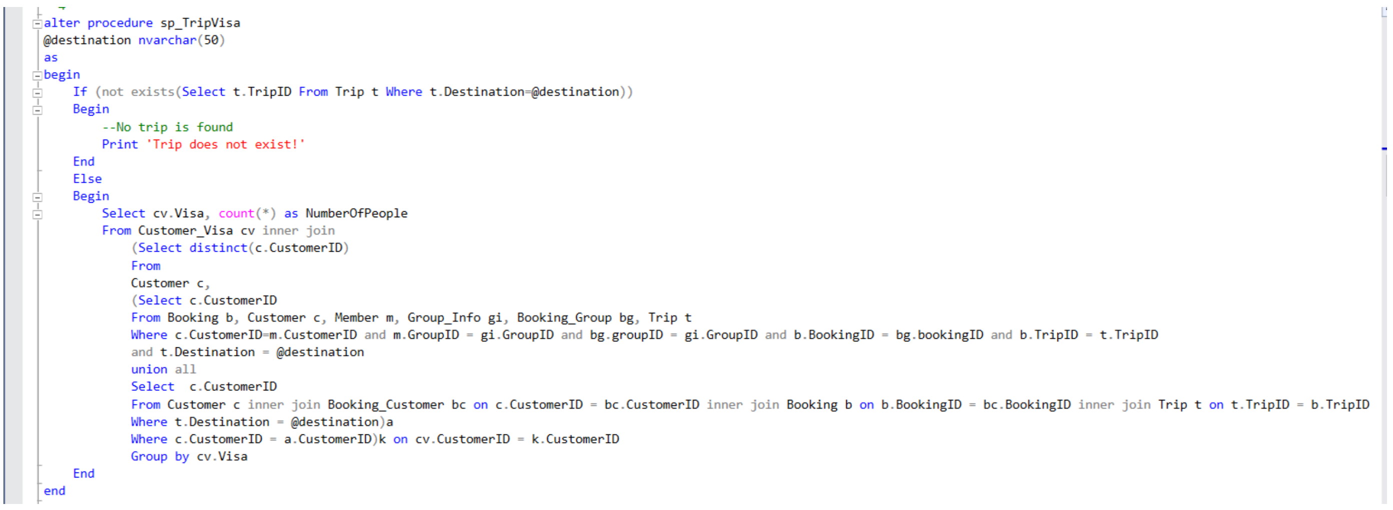
90 %

Results | Messages

	CustomerID	CustomerName	CustomerSurname	NumberOfBooking
1	28045729434	Mehmet Akif	Gülmüş	3
2	29216512176	Mahmut	Gülmüş	3
3	39735997552	BAYRAM	FURKAN	3
4	81534142626	Mustafa	Yanar	3
5	87065948232	RAHİME	Gülmüş	3

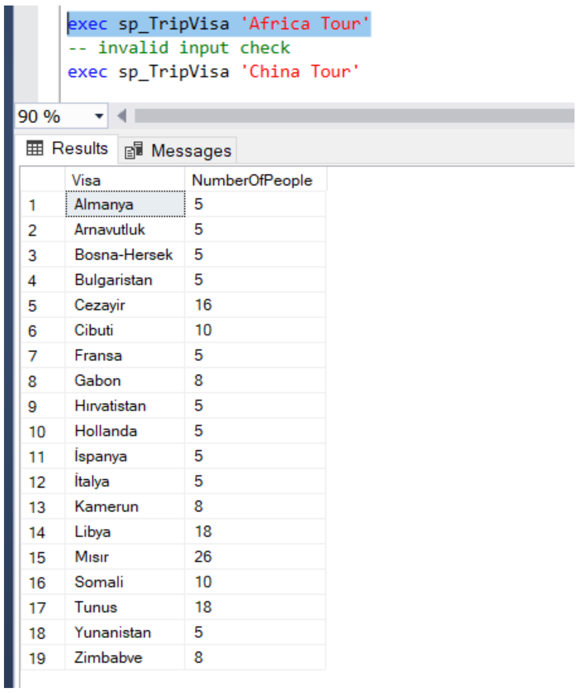
4.) sp_TripVisa

It groups visas by given trip's destination.



```
alter procedure sp_TripVisa
@destination nvarchar(50)
as
begin
    If (not exists(Select t.TripID From Trip t Where t.Destination=@destination))
    Begin
        --No trip is found
        Print 'Trip does not exist!'
    End
    Else
    Begin
        Select cv.Visa, count(*) as NumberOfPeople
        From Customer_Visa cv inner join
            (Select distinct(c.CustomerID)
            From
            Customer c,
            (Select c.CustomerID
            From Booking b, Customer c, Member m, Group_Info gi, Booking_Group bg, Trip t
            Where c.CustomerID=m.CustomerID and m.GroupID = gi.GroupID and bg.groupID = gi.GroupID and b.BookingID = bg.bookingID and b.TripID = t.TripID
            and t.Destination = @destination
            union all
            Select  c.CustomerID
            From Customer c inner join Booking_Customer bc on c.CustomerID = bc.CustomerID inner join Booking b on b.BookingID = bc.BookingID inner join Trip t on t.TripID = b.TripID
            Where t.Destination = @destination)a
            Where c.CustomerID = a.CustomerID)k on cv.CustomerID = k.CustomerID
            Group by cv.Visa
    End
end
```

Output:



```
exec sp_TripVisa 'Africa Tour'
-- invalid input check
exec sp_TripVisa 'China Tour'
```

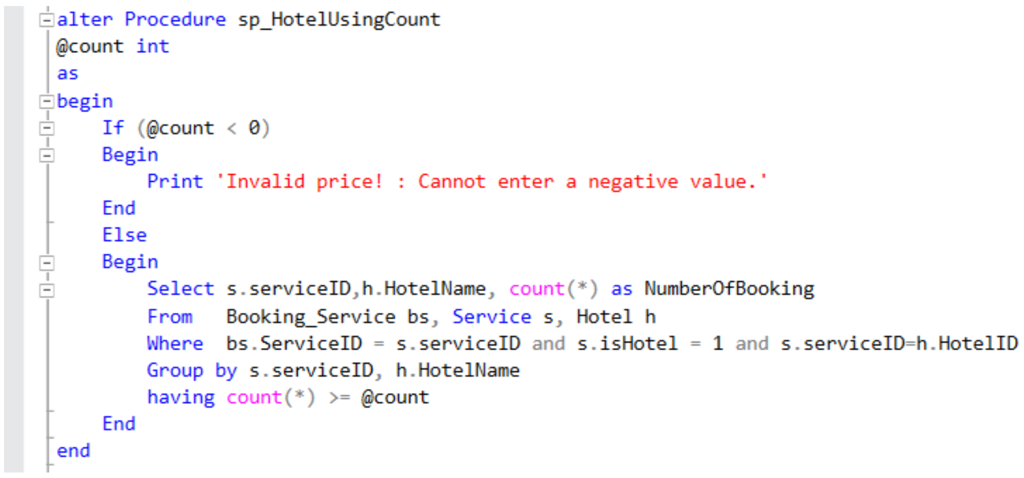
90 %

Results | Messages

	Visa	NumberOfPeople
1	Almanya	5
2	Arnavutluk	5
3	Bosna-Hersek	5
4	Bulgaristan	5
5	Cezayir	16
6	Cibuti	10
7	Fransa	5
8	Gabon	8
9	Hırvatistan	5
10	Hollanda	5
11	İspanya	5
12	İtalya	5
13	Kamerun	8
14	Libya	18
15	Mısır	26
16	Somali	10
17	Tunus	18
18	Yunanistan	5
19	Zimbabve	8

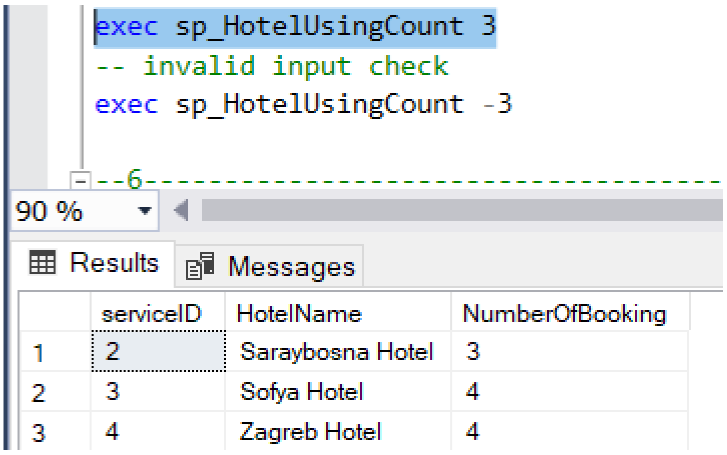
5.) sp_HotelUsingCount

It shows hotel which is more than book from given input.



```
alter Procedure sp_HotelUsingCount
@count int
as
begin
    If (@count < 0)
    Begin
        Print 'Invalid price! : Cannot enter a negative value.'
    End
    Else
    Begin
        Select s.serviceID,h.HotelName, count(*) as NumberOfBooking
        From   Booking_Service bs, Service s, Hotel h
        Where  bs.ServiceID = s.serviceID and s.isHotel = 1 and s.serviceID=h.HotelID
        Group by s.serviceID, h.HotelName
        having count(*) >= @count
    End
end
```

Output:



```
exec sp_HotelUsingCount 3
-- invalid input check
exec sp_HotelUsingCount -3

--6-----------------------------------
```

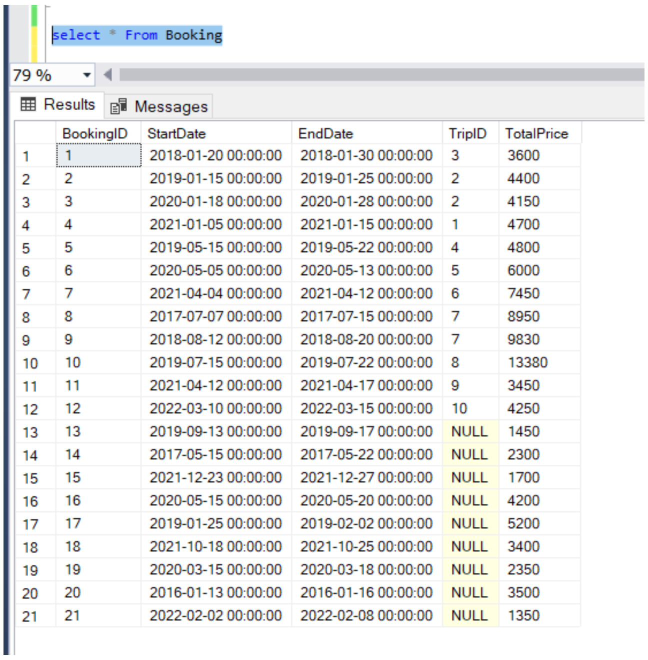
	serviceID	HotelName	NumberOfBooking
1	2	Saraybosna Hotel	3
2	3	Sofya Hotel	4
3	4	Zagreb Hotel	4

6.) sp_PostponeTrip
Inputs: trip id, current start and end date of the trip and new dates for start and end to be updated.
Trips can postpone so that dates should be changeable. This procedure finds the trip that wanted to be postponed and updates the dates of it with given input.

SQL Code:

```
alter Procedure sp_PostponeTrip
    @tripId int,
    @startDate smalldatetime,
    @newSDate smalldatetime,
    @newEDate smalldatetime
As
Begin

    If (not exists(Select t.TripID From Trip t Where t.TripID=@tripId))
    Begin
        --No trip is found
        Print 'Trip does not exist!'
    End
    Else
    Begin
        Update b
        Set b.StartDate=@newSDate, b.EndDate=@newEDate
        From Booking b inner join Trip t on b.TripID=t.TripID
        Where b.StartDate=@startDate

    End
End
```

Before:



```sql
select * From Booking
```

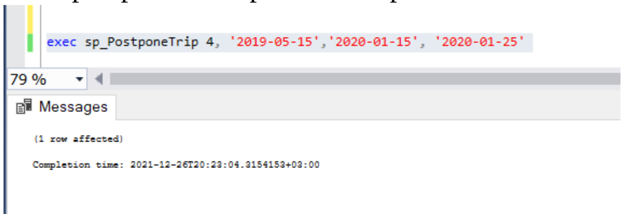
79 %

Results | Messages

	BookingID	StartDate	EndDate	TripID	TotalPrice
1	1	2018-01-20 00:00:00	2018-01-30 00:00:00	3	3600
2	2	2019-01-15 00:00:00	2019-01-25 00:00:00	2	4400
3	3	2020-01-18 00:00:00	2020-01-28 00:00:00	2	4150
4	4	2021-01-05 00:00:00	2021-01-15 00:00:00	1	4700
5	5	2019-05-15 00:00:00	2019-05-22 00:00:00	4	4800
6	6	2020-05-05 00:00:00	2020-05-13 00:00:00	5	6000
7	7	2021-04-04 00:00:00	2021-04-12 00:00:00	6	7450
8	8	2017-07-07 00:00:00	2017-07-15 00:00:00	7	8950
9	9	2018-08-12 00:00:00	2018-08-20 00:00:00	7	9830
10	10	2019-07-15 00:00:00	2019-07-22 00:00:00	8	13380
11	11	2021-04-12 00:00:00	2021-04-17 00:00:00	9	3450
12	12	2022-03-10 00:00:00	2022-03-15 00:00:00	10	4250
13	13	2019-09-13 00:00:00	2019-09-17 00:00:00	NULL	1450
14	14	2017-05-15 00:00:00	2017-05-22 00:00:00	NULL	2300
15	15	2021-12-23 00:00:00	2021-12-27 00:00:00	NULL	1700
16	16	2020-05-15 00:00:00	2020-05-20 00:00:00	NULL	4200
17	17	2019-01-25 00:00:00	2019-02-02 00:00:00	NULL	5200
18	18	2021-10-18 00:00:00	2021-10-25 00:00:00	NULL	3400
19	19	2020-03-15 00:00:00	2020-03-18 00:00:00	NULL	2350
20	20	2016-01-13 00:00:00	2016-01-16 00:00:00	NULL	3500
21	21	2022-02-02 00:00:00	2022-02-08 00:00:00	NULL	1350

Figure 1. Trips that are booked with their dates.

Let's postpone and update the trip whose id is 4 and starts on 2019-05-15:



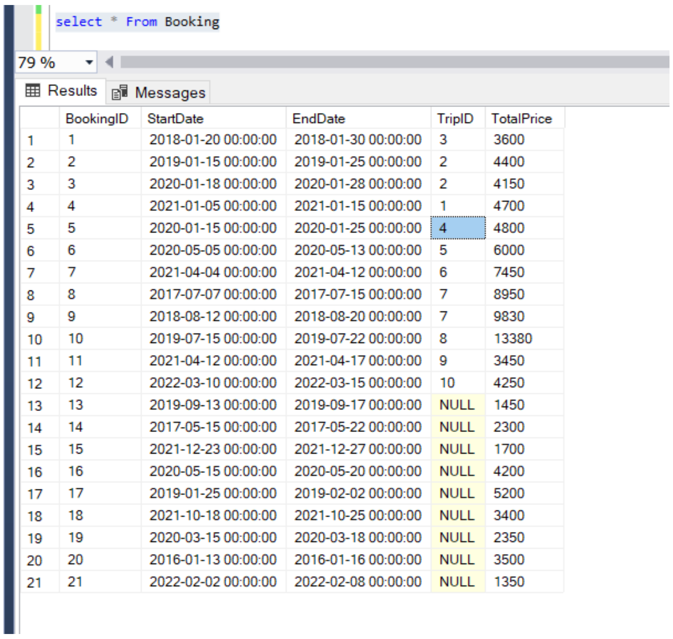
```sql
exec sp_PostponeTrip 4, '2019-05-15','2020-01-15', '2020-01-25'
```

79 %

Messages

```
(1 row affected)

Completion time: 2021-12-26T20:23:04.3154153+03:00
```

After:



```sql
select * From Booking
```

79 %

Results | Messages

	BookingID	StartDate	EndDate	TripID	TotalPrice
1	1	2018-01-20 00:00:00	2018-01-30 00:00:00	3	3600
2	2	2019-01-15 00:00:00	2019-01-25 00:00:00	2	4400
3	3	2020-01-18 00:00:00	2020-01-28 00:00:00	2	4150
4	4	2021-01-05 00:00:00	2021-01-15 00:00:00	1	4700
5	5	2020-01-15 00:00:00	2020-01-25 00:00:00	4	4800
6	6	2020-05-05 00:00:00	2020-05-13 00:00:00	5	6000
7	7	2021-04-04 00:00:00	2021-04-12 00:00:00	6	7450
8	8	2017-07-07 00:00:00	2017-07-15 00:00:00	7	8950
9	9	2018-08-12 00:00:00	2018-08-20 00:00:00	7	9830
10	10	2019-07-15 00:00:00	2019-07-22 00:00:00	8	13380
11	11	2021-04-12 00:00:00	2021-04-17 00:00:00	9	3450
12	12	2022-03-10 00:00:00	2022-03-15 00:00:00	10	4250
13	13	2019-09-13 00:00:00	2019-09-17 00:00:00	NULL	1450
14	14	2017-05-15 00:00:00	2017-05-22 00:00:00	NULL	2300
15	15	2021-12-23 00:00:00	2021-12-27 00:00:00	NULL	1700
16	16	2020-05-15 00:00:00	2020-05-20 00:00:00	NULL	4200
17	17	2019-01-25 00:00:00	2019-02-02 00:00:00	NULL	5200
18	18	2021-10-18 00:00:00	2021-10-25 00:00:00	NULL	3400
19	19	2020-03-15 00:00:00	2020-03-18 00:00:00	NULL	2350
20	20	2016-01-13 00:00:00	2016-01-16 00:00:00	NULL	3500
21	21	2022-02-02 00:00:00	2022-02-08 00:00:00	NULL	1350

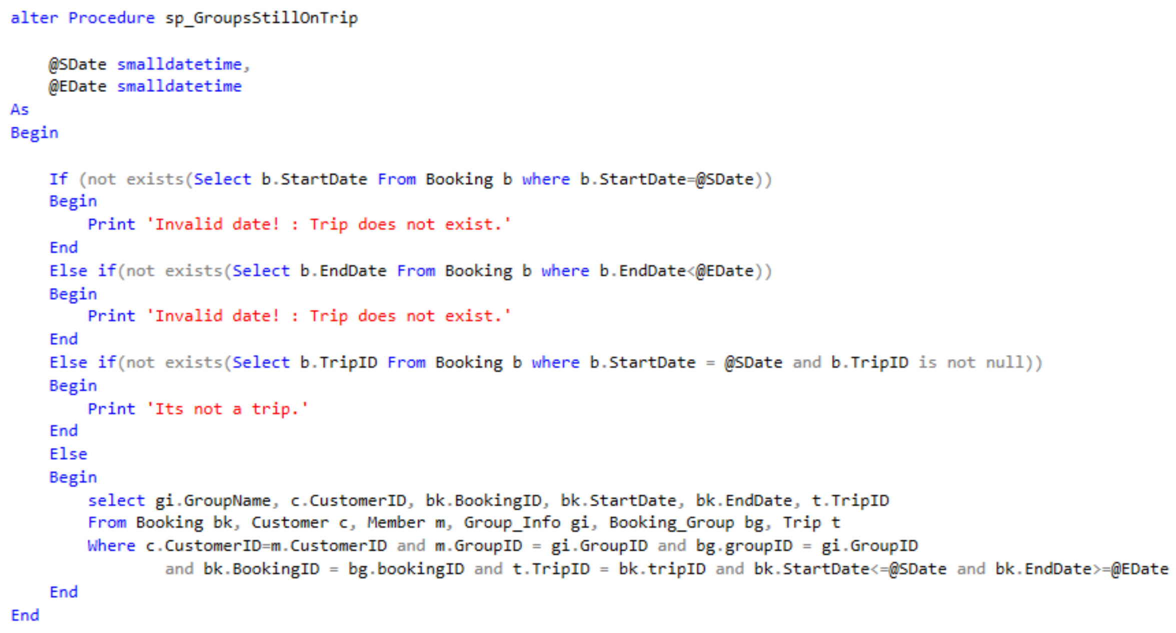
Figure 2. Updated Booking table.

Date has changed from 2019-05-15, 2019-05-22 to 2020-01-15, 2020-01-25.

7.) sp_GroupsStillOnTrip

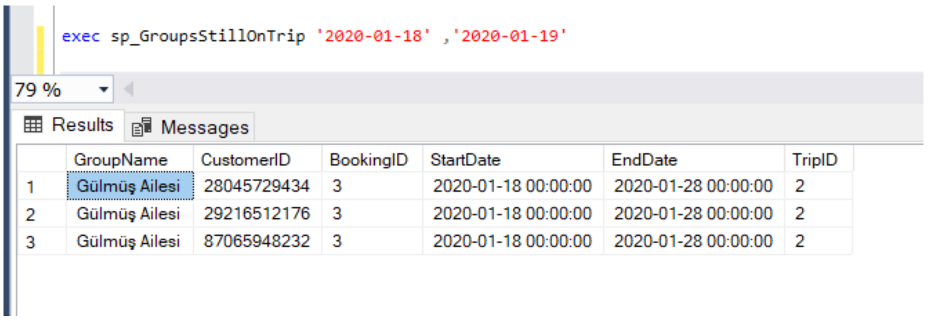
Takes two dates as an input. This procedure checks the families or groups whose trip is started but not ended between given input dates:

Code:



```
alter Procedure sp_GroupsStillOnTrip

    @SDate smalldatetime,
    @EDate smalldatetime
As
Begin

    If (not exists(Select b.StartDate From Booking b where b.StartDate=@SDate))
    Begin
        Print 'Invalid date! : Trip does not exist.'
    End
    Else if(not exists(Select b.EndDate From Booking b where b.EndDate<@EDate))
    Begin
        Print 'Invalid date! : Trip does not exist.'
    End
    Else if(not exists(Select b.TripID From Booking b where b.StartDate = @SDate and b.TripID is not null))
    Begin
        Print 'Its not a trip.'
    End
    Else
    Begin
        select gi.GroupName, c.CustomerID, bk.BookingID, bk.StartDate, bk.EndDate, t.TripID
        From Booking bk, Customer c, Member m, Group_Info gi, Booking_Group bg, Trip t
        Where c.CustomerID=m.CustomerID and m.GroupID = gi.GroupID and bg.groupID = gi.GroupID
                and bk.BookingID = bg.bookingID and t.TripID = bk.tripID and bk.StartDate<=@SDate and bk.EndDate>=@EDate
    End
End
```

Output:



```
exec sp_GroupsStillOnTrip '2020-01-18' ,'2020-01-19'
```

79 %

Results | Messages

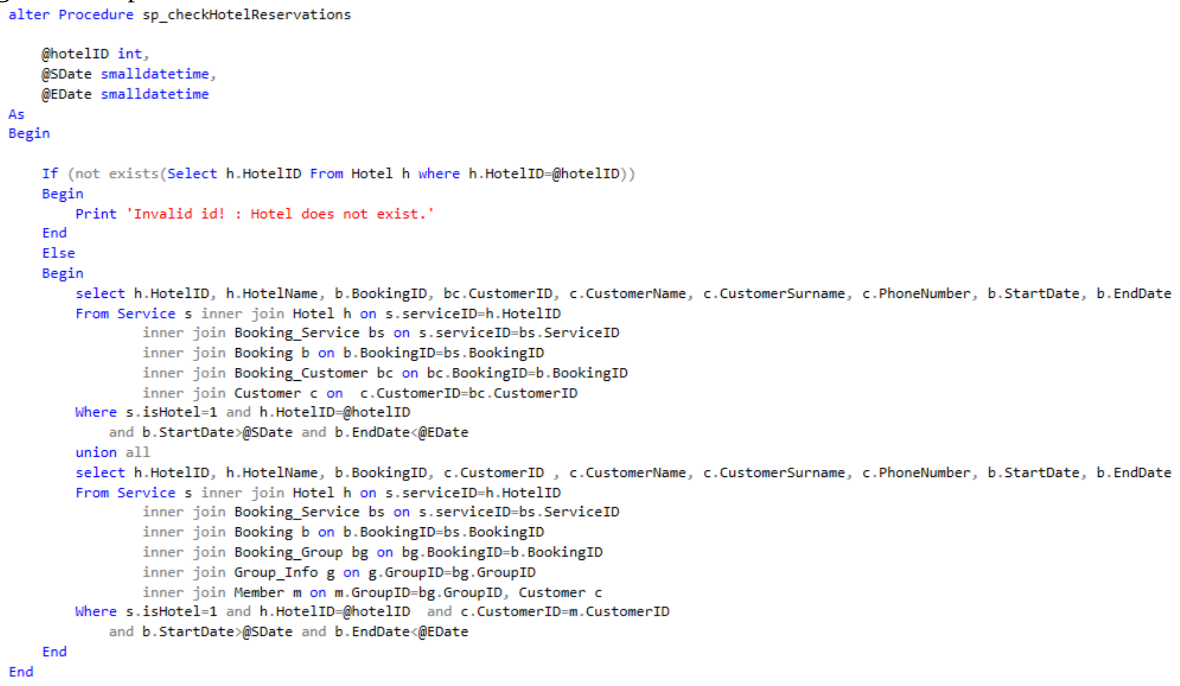
	GroupName	CustomerID	BookingID	StartDate	EndDate	TripID
1	Gülmüş Ailesi	28045729434	3	2020-01-18 00:00:00	2020-01-28 00:00:00	2
2	Gülmüş Ailesi	29216512176	3	2020-01-18 00:00:00	2020-01-28 00:00:00	2
3	Gülmüş Ailesi	87065948232	3	2020-01-18 00:00:00	2020-01-28 00:00:00	2

Family Gülmüş's trip started after or in the given date but not ended yet.

8.) sp_checkHotelReservations

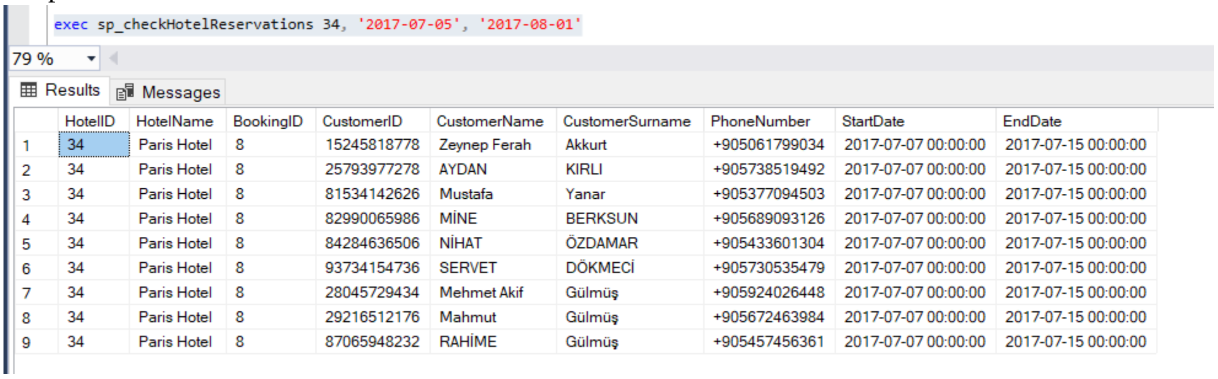
Inputs: hotel id, start and end date of the hotel booking

This procedure finds the customers(in a group or individual) who has a booking on that specific hotel in a given time period.



```
alter Procedure sp_checkHotelReservations

    @hotelID int,
    @SDate smalldatetime,
    @EDate smalldatetime
As
Begin

    If (not exists(Select h.HotelID From Hotel h where h.HotelID=@hotelID))
    Begin
        Print 'Invalid id! : Hotel does not exist.'
    End
    Else
    Begin
        select h.HotelID, h.HotelName, b.BookingID, bc.CustomerID, c.CustomerName, c.CustomerSurname, c.PhoneNumber, b.StartDate, b.EndDate
        From Service s inner join Hotel h on s.serviceID=h.HotelID
                inner join Booking_Service bs on s.serviceID=bs.ServiceID
                inner join Booking b on b.BookingID=bs.BookingID
                inner join Booking_Customer bc on bc.BookingID=b.BookingID
                inner join Customer c on  c.CustomerID=bc.CustomerID
        Where s.isHotel=1 and h.HotelID=@hotelID
            and b.StartDate>@SDate and b.EndDate<@EDate
        union all
        select h.HotelID, h.HotelName, b.BookingID, c.CustomerID , c.CustomerName, c.CustomerSurname, c.PhoneNumber, b.StartDate, b.EndDate
        From Service s inner join Hotel h on s.serviceID=h.HotelID
                inner join Booking_Service bs on s.serviceID=bs.ServiceID
                inner join Booking b on b.BookingID=bs.BookingID
                inner join Booking_Group bg on bg.BookingID=b.BookingID
                inner join Group_Info g on g.GroupID=bg.GroupID
                inner join Member m on m.GroupID=bg.GroupID, Customer c
        Where s.isHotel=1 and h.HotelID=@hotelID  and c.CustomerID=m.CustomerID
            and b.StartDate>@SDate and b.EndDate<@EDate
    End
End
```

Output:



```
exec sp_checkHotelReservations 34, '2017-07-05', '2017-08-01'
```

79 %

Results | Messages

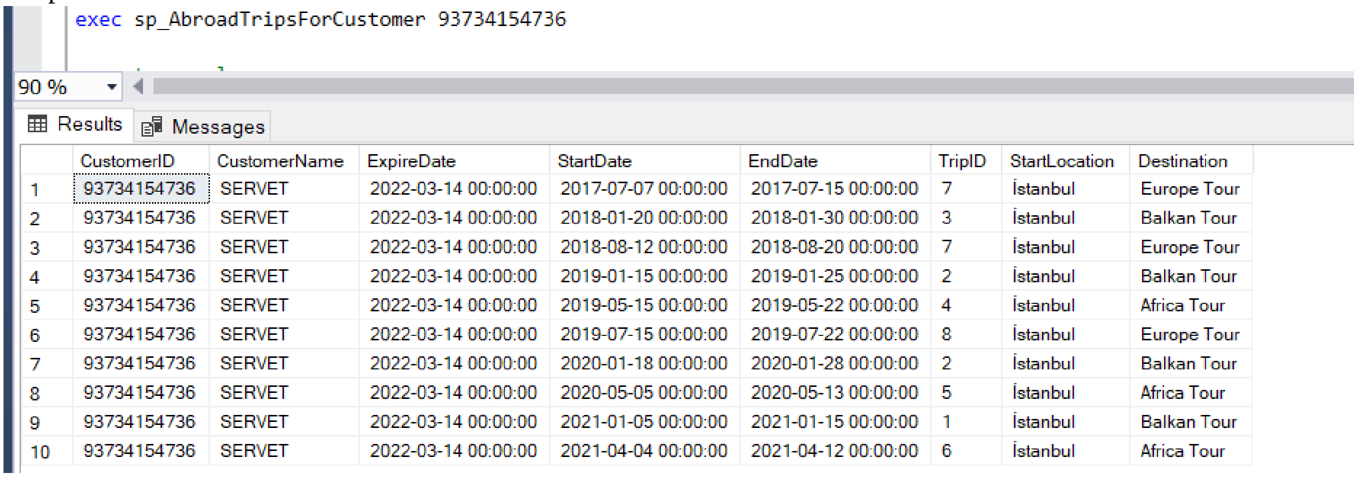
	HotelID	HotelName	BookingID	CustomerID	CustomerName	CustomerSurname	PhoneNumber	StartDate	EndDate
1	34	Paris Hotel	8	15245818778	Zeynep Ferah	Akkurt	+905061799034	2017-07-07 00:00:00	2017-07-15 00:00:00
2	34	Paris Hotel	8	25793977278	AYDAN	KIRLI	+905738519492	2017-07-07 00:00:00	2017-07-15 00:00:00
3	34	Paris Hotel	8	81534142626	Mustafa	Yanar	+905377094503	2017-07-07 00:00:00	2017-07-15 00:00:00
4	34	Paris Hotel	8	82990065986	MİNE	BERKSUN	+905689093126	2017-07-07 00:00:00	2017-07-15 00:00:00
5	34	Paris Hotel	8	84284636506	NİHAT	ÖZDAMAR	+905433601304	2017-07-07 00:00:00	2017-07-15 00:00:00
6	34	Paris Hotel	8	93734154736	SERVET	DÖKMECİ	+905730535479	2017-07-07 00:00:00	2017-07-15 00:00:00
7	34	Paris Hotel	8	28045729434	Mehmet Akif	Gülmüş	+905924026448	2017-07-07 00:00:00	2017-07-15 00:00:00
8	34	Paris Hotel	8	29216512176	Mahmut	Gülmüş	+905672463984	2017-07-07 00:00:00	2017-07-15 00:00:00
9	34	Paris Hotel	8	87065948232	RAHİME	Gülmüş	+905457456361	2017-07-07 00:00:00	2017-07-15 00:00:00

Paris Hotel with an id 34, is booked by several customers in a given time period.

9.) sp_AbroadTripsForCustomer
Inputs: customer id
This procedure finds the available abroad trips that a customer can book without having a passport expire date problem. (trips that is before the expire date of the passport.).

```
alter Procedure sp_AbroadTripsForCustomer

    @cId bigint
As
Begin

    If (not exists(Select c.CustomerID From Customer c where c.CustomerID=@cId))
    Begin
        Print 'Invalid customer id! : Customer does not exist.'
    End
    Else if(not exists(select p.ExpireDate
                       from Passport p where p.CustomerID=@cId))
    Begin
        Print 'Customer does not have a passport data! : Passport does not exist.'
    End
    Else
    Begin

        Declare @CusExpirePassDate smalldatetime
        Set @CusExpirePassDate = (select p.ExpireDate
                                  from Customer c inner join Passport p on c.CustomerID=p.CustomerID
                                  where c.CustomerID=@cId)

        select  c.CustomerID, c.CustomerName, @CusExpirePassDate ExpireDate,
                abrTrip.StartDate, abrTrip.EndDate , abrTrip.TripID, abrTrip.StartLocation, abrTrip.Destination
        from Customer c,(select b.BookingID, t.TripID, b.StartDate, b.EndDate, t.StartLocation, t.Destination
                         From Booking b inner join Trip t on b.TripID=t.TripID
                         where t.isAbroad=1) abrTrip
        where abrTrip.EndDate<@CusExpirePassDate and c.CustomerID=@cId
        order by abrTrip.StartDate
    End
End
```

Output:



```
exec sp_AbroadTripsForCustomer 93734154736
```

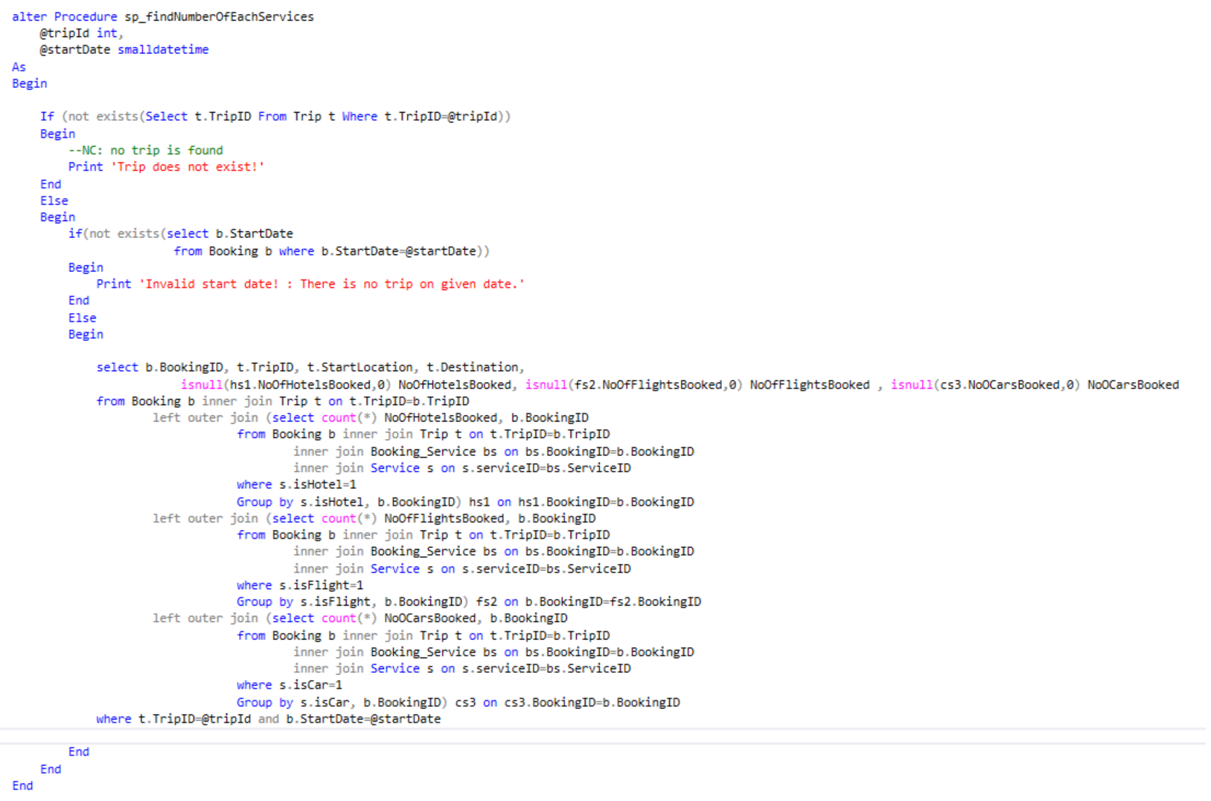
90 %

Results | Messages

	CustomerID	CustomerName	ExpireDate	StartDate	EndDate	TripID	StartLocation	Destination
1	93734154736	SERVET	2022-03-14 00:00:00	2017-07-07 00:00:00	2017-07-15 00:00:00	7	İstanbul	Europe Tour
2	93734154736	SERVET	2022-03-14 00:00:00	2018-01-20 00:00:00	2018-01-30 00:00:00	3	İstanbul	Balkan Tour
3	93734154736	SERVET	2022-03-14 00:00:00	2018-08-12 00:00:00	2018-08-20 00:00:00	7	İstanbul	Europe Tour
4	93734154736	SERVET	2022-03-14 00:00:00	2019-01-15 00:00:00	2019-01-25 00:00:00	2	İstanbul	Balkan Tour
5	93734154736	SERVET	2022-03-14 00:00:00	2019-05-15 00:00:00	2019-05-22 00:00:00	4	İstanbul	Africa Tour
6	93734154736	SERVET	2022-03-14 00:00:00	2019-07-15 00:00:00	2019-07-22 00:00:00	8	İstanbul	Europe Tour
7	93734154736	SERVET	2022-03-14 00:00:00	2020-01-18 00:00:00	2020-01-28 00:00:00	2	İstanbul	Balkan Tour
8	93734154736	SERVET	2022-03-14 00:00:00	2020-05-05 00:00:00	2020-05-13 00:00:00	5	İstanbul	Africa Tour
9	93734154736	SERVET	2022-03-14 00:00:00	2021-01-05 00:00:00	2021-01-15 00:00:00	1	İstanbul	Balkan Tour
10	93734154736	SERVET	2022-03-14 00:00:00	2021-04-04 00:00:00	2021-04-12 00:00:00	6	İstanbul	Africa Tour

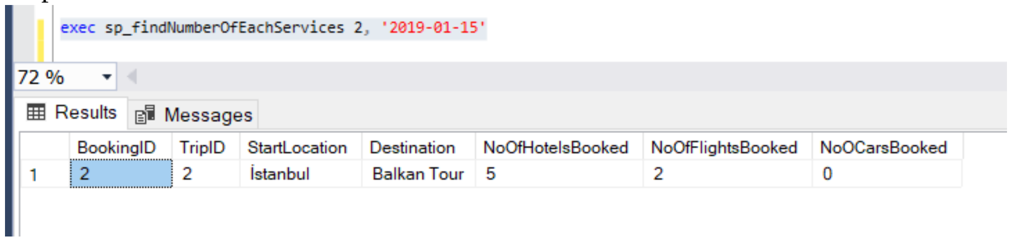
Servet can book the trips 1,2,3,4,5,6,7 and 8 without any problem. But she cannot book any trip which is after 2020-01-17.

10) sp_findNumberOfEachServices
Inputs: trip id and start date of trip
This procedure counts the number of services for each kind in a given trip.
Code:



```
alter Procedure sp_findNumberOfEachServices
    @tripId int,
    @startDate smalldatetime
As
Begin

    If (not exists(Select t.TripID From Trip t Where t.TripID=@tripId))
    Begin
        --NC: no trip is found
        Print 'Trip does not exist!'
    End
    Else
    Begin
        if(not exists(select b.StartDate
                      from Booking b where b.StartDate=@startDate))
        Begin
            Print 'Invalid start date! : There is no trip on given date.'
        End
        Else
        Begin

            select b.BookingID, t.TripID, t.StartLocation, t.Destination,
                        isnull(hs1.NoOfHotelsBooked,0) NoOfHotelsBooked, isnull(fs2.NoOfFlightsBooked,0) NoOfFlightsBooked , isnull(cs3.NoOCarsBooked,0) NoOCarsBooked
            from Booking b inner join Trip t on t.TripID=b.TripID
                    left outer join (select count(*) NoOfHotelsBooked, b.BookingID
                                from Booking b inner join Trip t on t.TripID=b.TripID
                                        inner join Booking_Service bs on bs.BookingID=b.BookingID
                                        inner join Service s on s.serviceID=bs.ServiceID
                                where s.isHotel=1
                                Group by s.isHotel, b.BookingID) hs1 on hs1.BookingID=b.BookingID
                    left outer join (select count(*) NoOfFlightsBooked, b.BookingID
                                from Booking b inner join Trip t on t.TripID=b.TripID
                                        inner join Booking_Service bs on bs.BookingID=b.BookingID
                                        inner join Service s on s.serviceID=bs.ServiceID
                                where s.isFlight=1
                                Group by s.isFlight, b.BookingID) fs2 on b.BookingID=fs2.BookingID
                    left outer join (select count(*) NoOCarsBooked, b.BookingID
                                from Booking b inner join Trip t on t.TripID=b.TripID
                                        inner join Booking_Service bs on bs.BookingID=b.BookingID
                                        inner join Service s on s.serviceID=bs.ServiceID
                                where s.isCar=1
                                Group by s.isCar, b.BookingID) cs3 on cs3.BookingID=b.BookingID
            where t.TripID=@tripId and b.StartDate=@startDate

        End
    End
End
```

Ouput:



```
exec sp_findNumberOfEachServices 2, '2019-01-15'
```

72 %

Results | Messages

	BookingID	TripID	StartLocation	Destination	NoOfHotelsBooked	NoOfFlightsBooked	NoOCarsBooked
1	2	2	İstanbul	Balkan Tour	5	2	0

For trip who has an id 2 and starts on 2019-01-15, there are 5 hotels, 2 flights booked.