ITC607 Data Management

Assignment 2

Faculty of Health, Humanities and Computing

Name / Student ID: Shawn Chen / 2017003399 **Date:16 June 2019**

Task	Mark	Result
1. Requirements	10	
2. Logical Design	10	
3. SQL	10	
Sum	**30**	

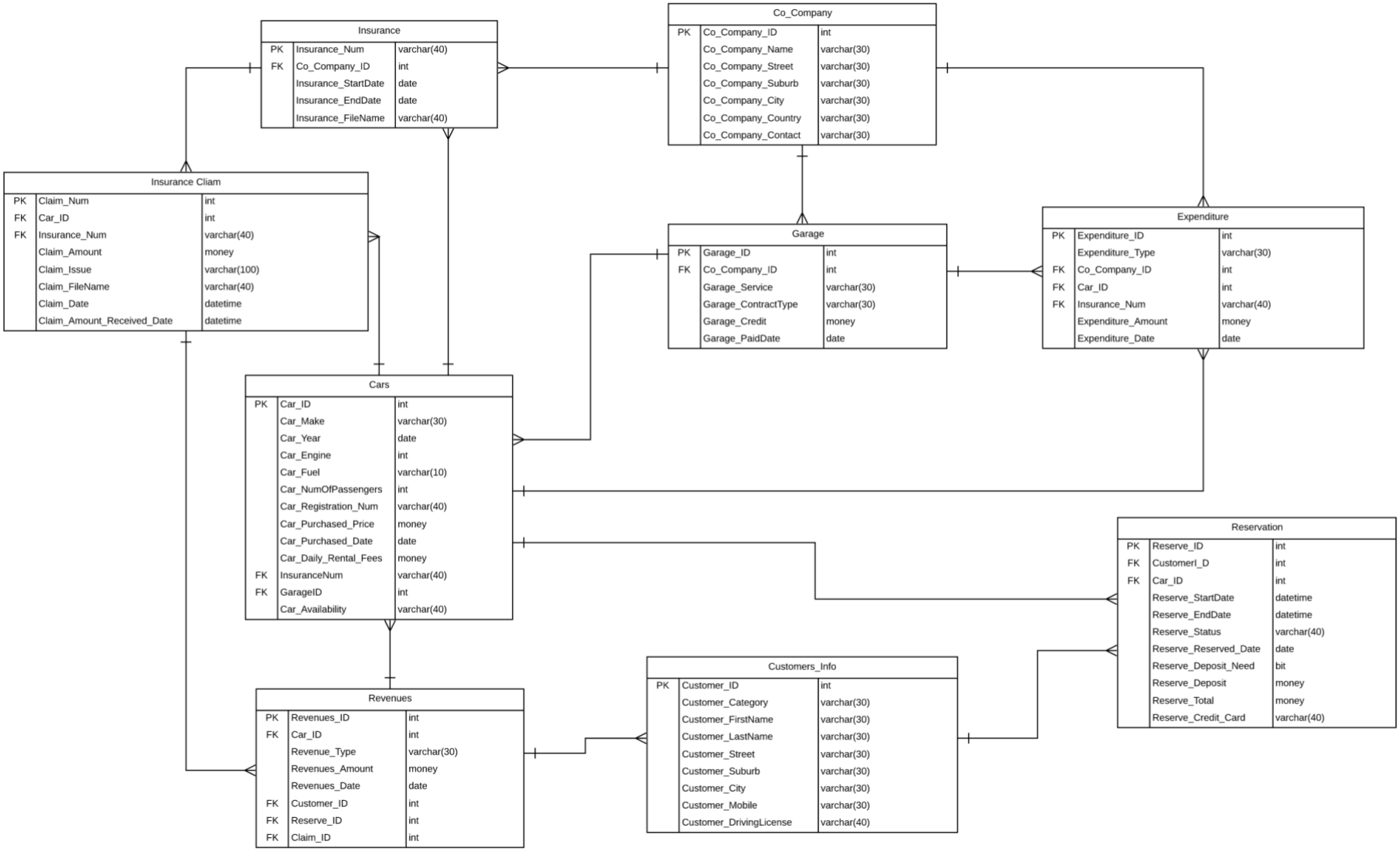
Declaration

I declare that this assignment submission will be my own work and I will not collude with anyone else on the preparation of this Assignment.

1. **Requirements:** List and work out the requirements for the database. This will require some imagination and assumption as unfortunately the client brief does not seem to be 100 % complete, though it is reasonably thorough and clear.

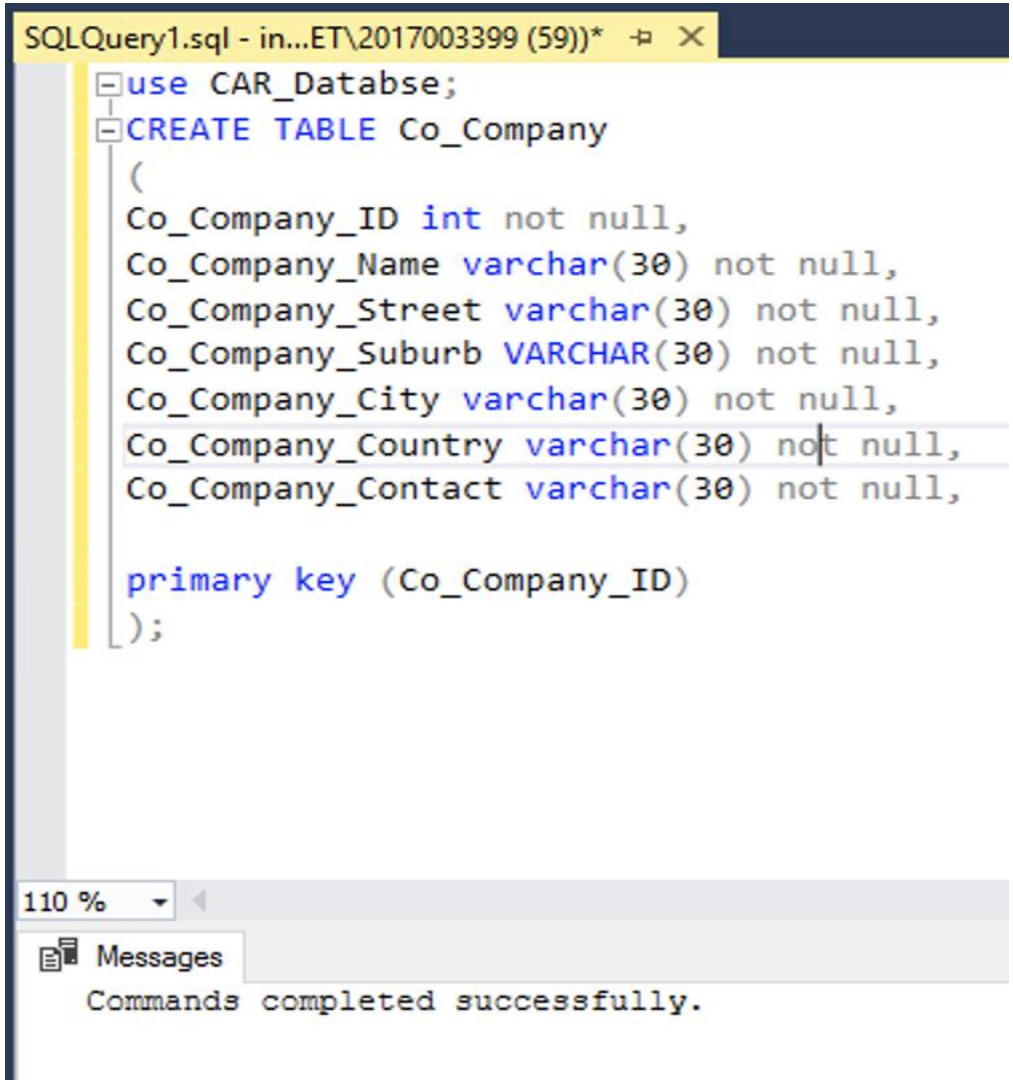
1. All client's details are required to be hold in database.
2. There are two types of customer group: privileged customers and casual customers.
3. Privileged customer has the special credit and can make booking in advance for a particular car at any time up to 1 month (assume 1 month = 30 days).
4. Casual customers are not allowed for the privileges as above, they must pay a deposit for any rental, unless they will pay by credit card.
5. Assume no customer will exceed the period that they have booked.
6. Customer needs to sign the desired start date and time and the desired end date and time.
7. Assume that the customer desired start date and time, cannot be less than 13 hours before the next reservation started on the same car.
8. Assume that the customer desired end date and time, cannot be less than 1 hours before the next reservation started on the same car.
9. Assume that customer can cancel their reservation before 24 hours.
10. All cars details will be kept in database.
11. All cars will not be hold for a period exceeding one year (assume 1 year = 365 days).
12. Assume that the administrator needs to execute a procedure manually to update the cars' status from available or rental to waiting for sale if meets the condition above (No. 10).
13. Each car will require an insurance.
14. Insurance claims must be hold on file.
15. Insurance company details need to be kept in database.
16. All major repairs and maintenance are done by subcontractors with long-term agreements.
17. Subcontractor details need to be stored in database.
18. Agreement of payment is in 2 types: immediately paid after repair, credit allowance for a period.
19. All purchases need to be registered as company expenditures, regards to repair, maintenance, insurance and so on.
20. Cash inflow needs to be assigned as company revenues, relates to car hire, car sales and insurance claims.

2. **Logical Design:** Design the database and produce the Entity Relationship Diagram (ERD).

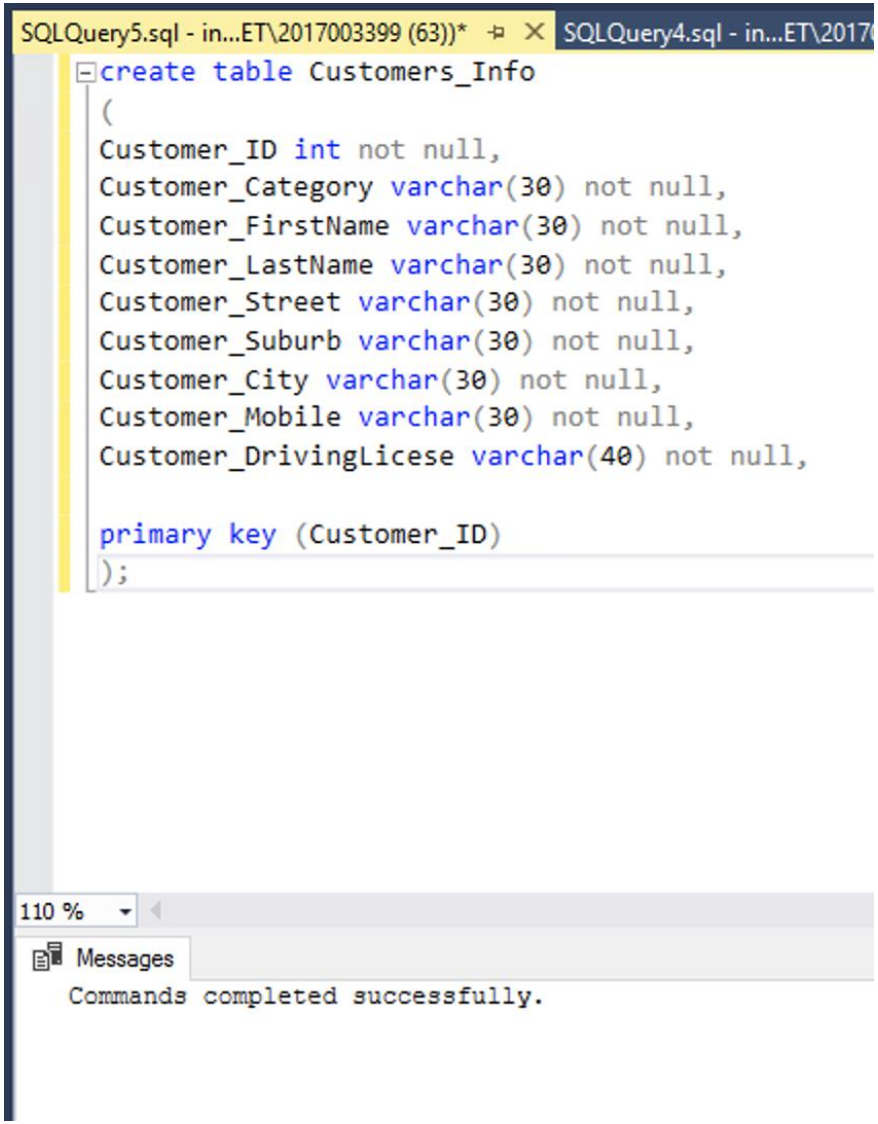


3. **SQL:** Create the Database using SQL server. You must take screen shot of each step and explain your code.

Part1: Tables



The "Co_Company" table represents other co-operation companies' details including **their ID** (primary key), **full name**, **address** and **contact number**.



"Customers_Info" contains all the details about including the **ID** number (primary key), **customer category** (Privileged or Not privileged), **name**, **address**, **mobile** and the **driving license** number.

SQLQuery1.sql - in...ET\2017003399 (57))*

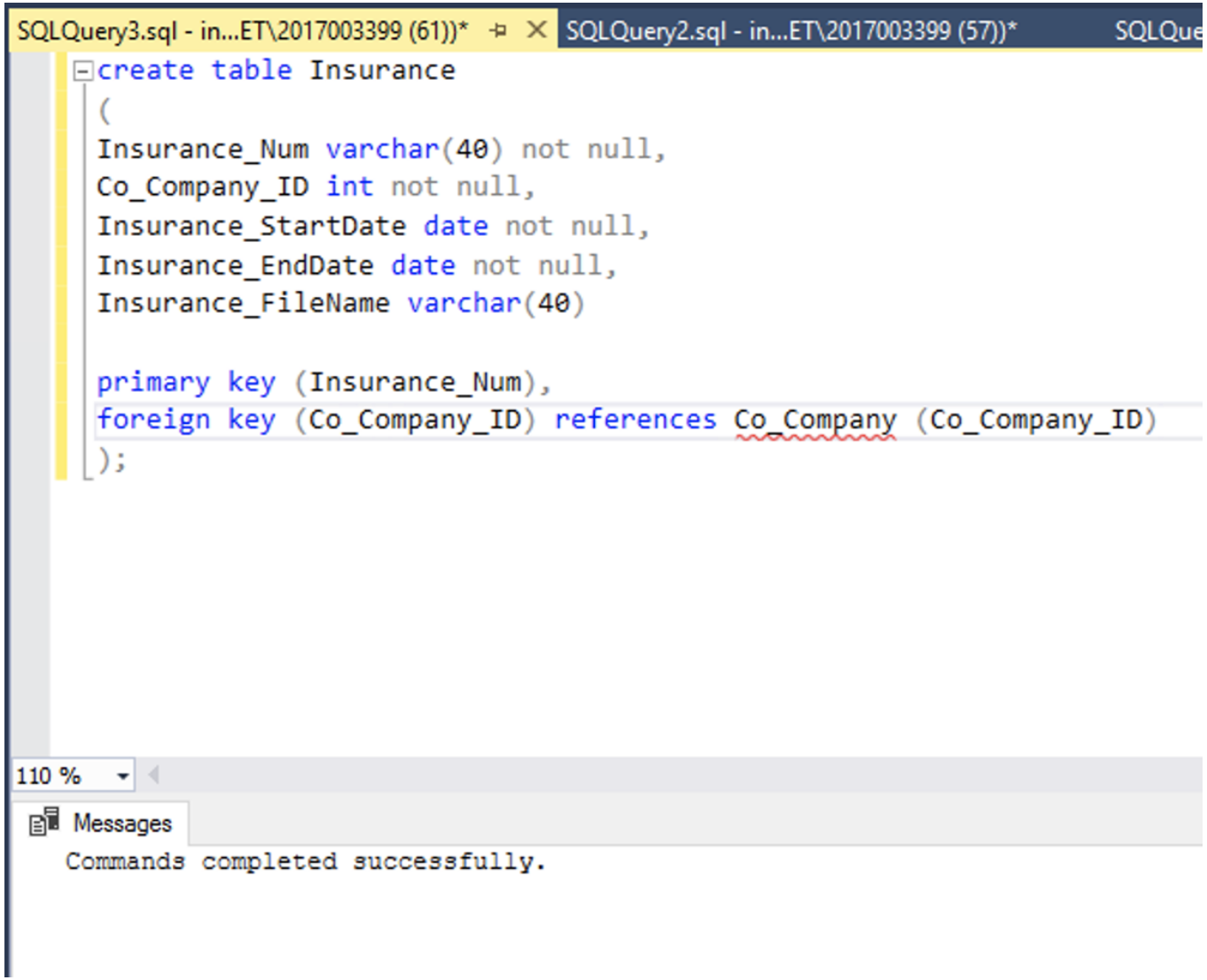
```
create table Garage
(
Garage_ID int not null,
Co_Company_ID int not null,
Garage_Service varchar(30) not null,
Garage_ContractType varchar(30) not null,
Garage_Credit money null,
Garage_PaidDate date null,

primary key (Garage_ID),
foreign key (Co_Company_ID) references Co_Company (Co_Company_ID)
);
```

110 %

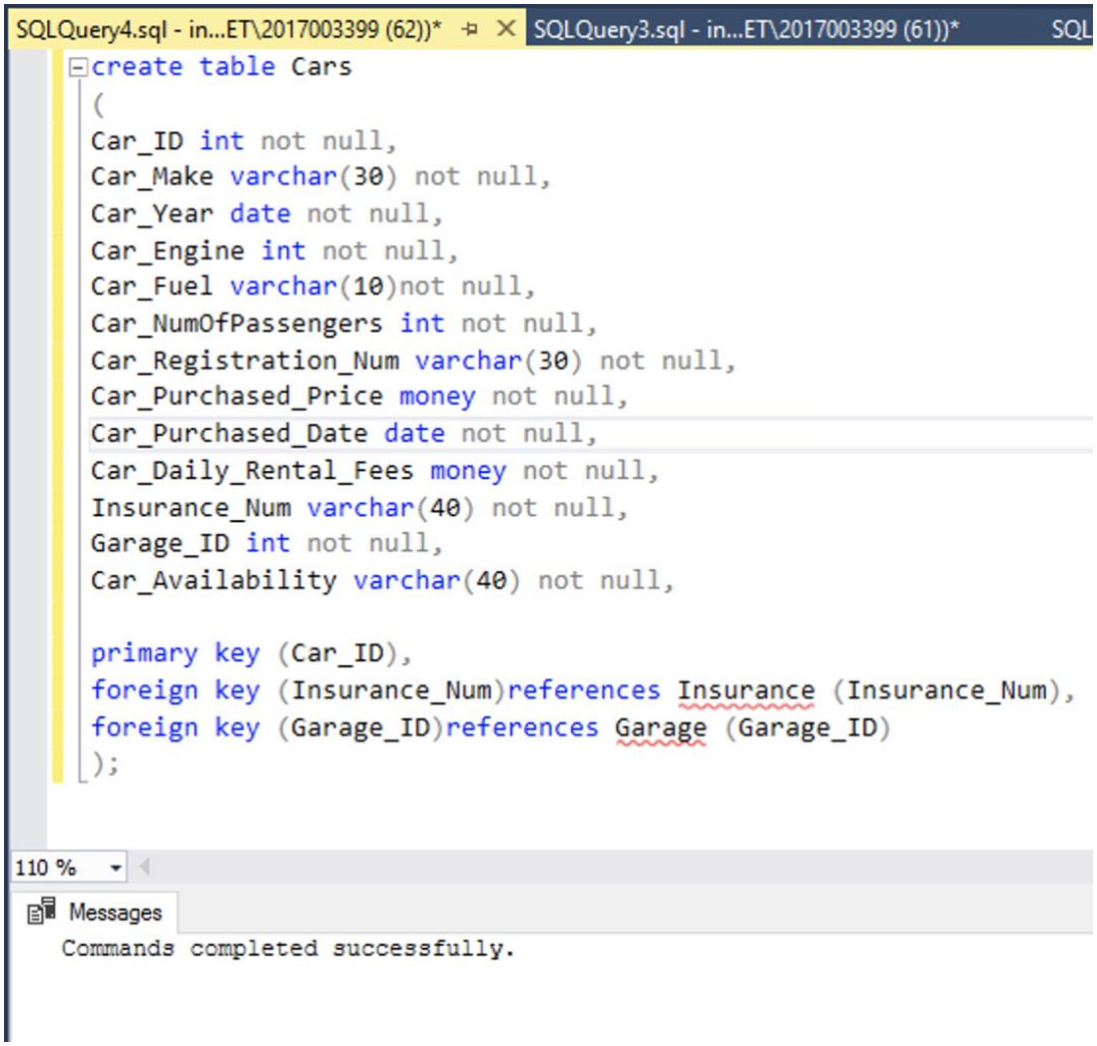
Messages

```
Commands completed successfully.
```

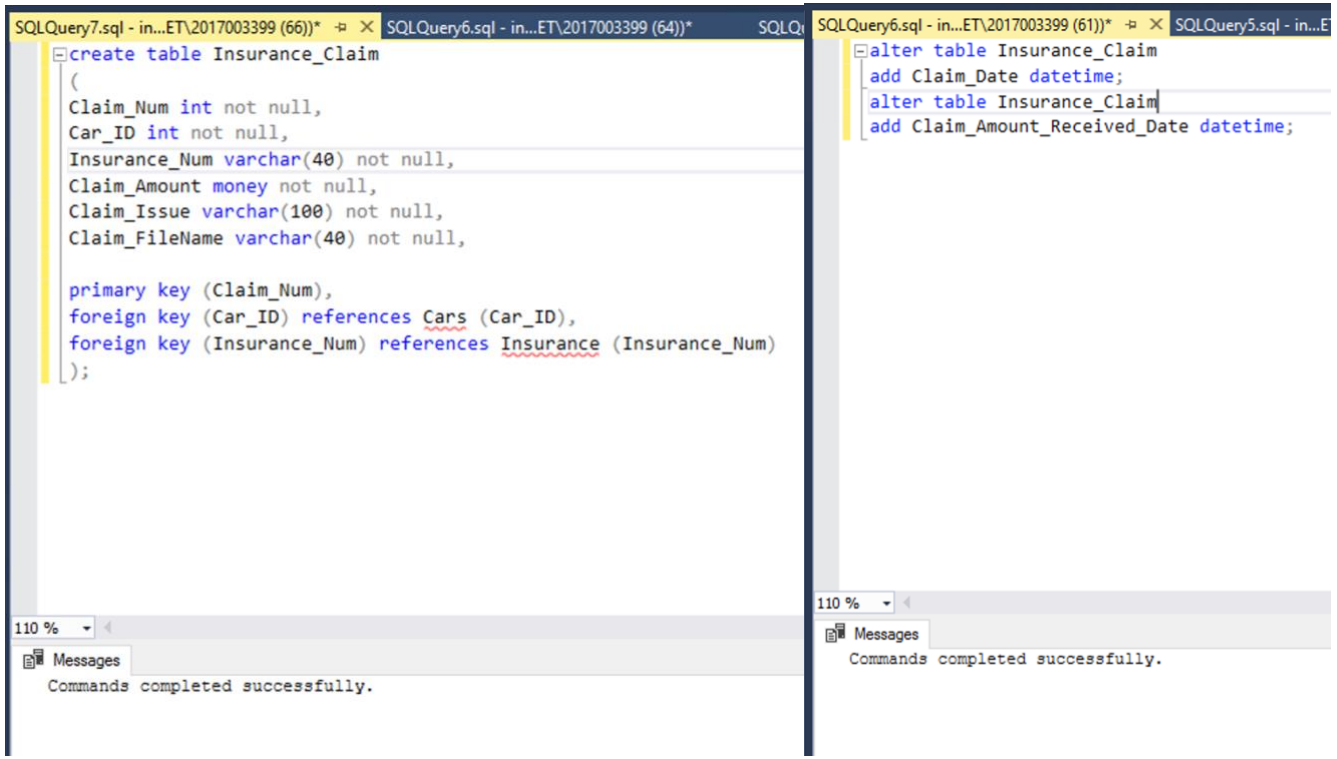
"Garage" table represents the garage further information including the a **ID number** (primary key), **a co-operation company ID** (foreign key, relates to the Co-company information), the **range of service**, the **contract type** (pay immediately, or pay monthly), the **allowance of monthly credit**, and the **fixed date** that need to pay for repair fees.



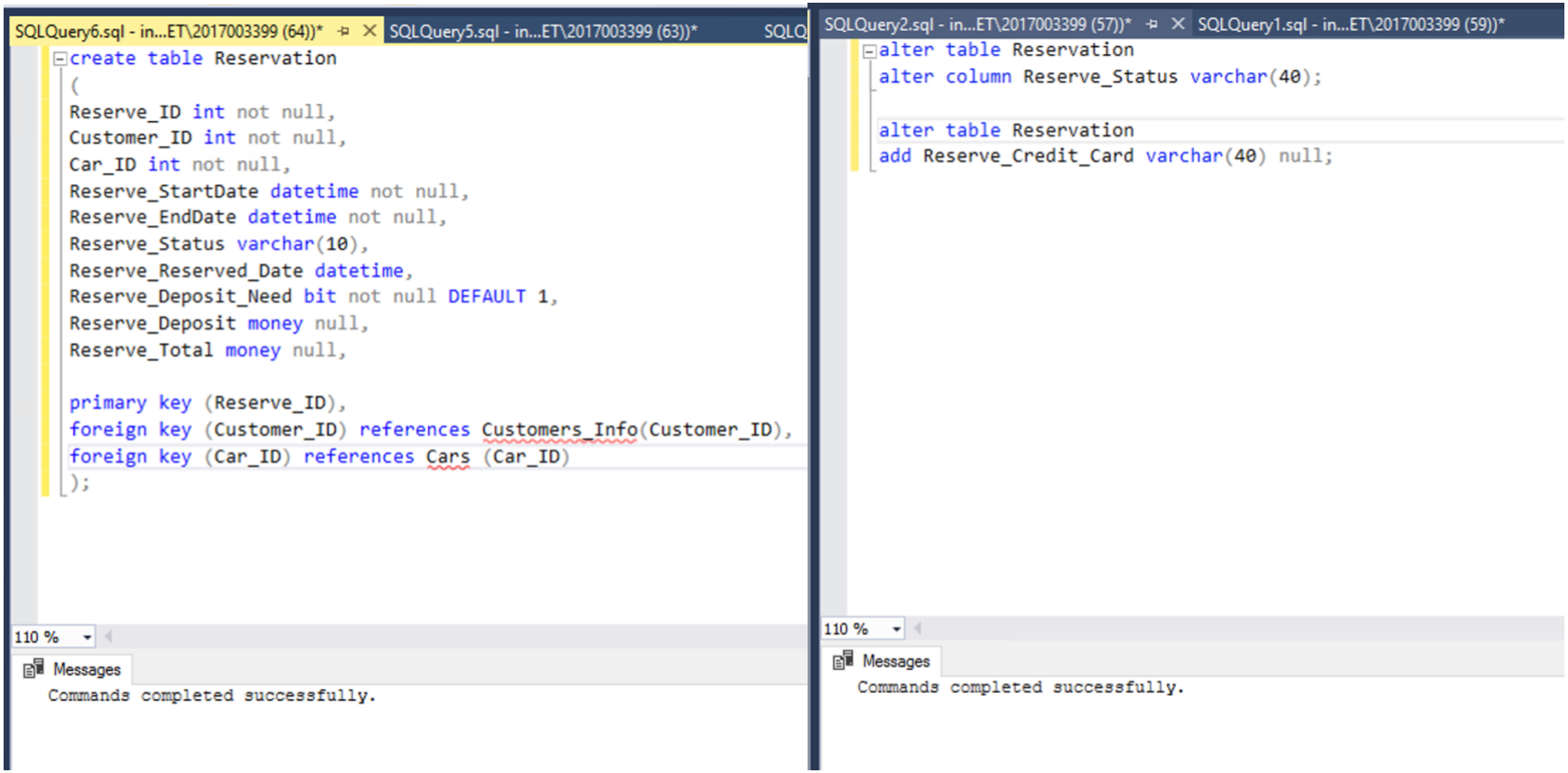
The “Insurance” table represents the insurance information includes the insurance number (primary key), the Co-operation company information (foreign key, relates to the insurance company details), the start date of this insurance, the end date of this insurance, and the insurance file name that store on the local disk.



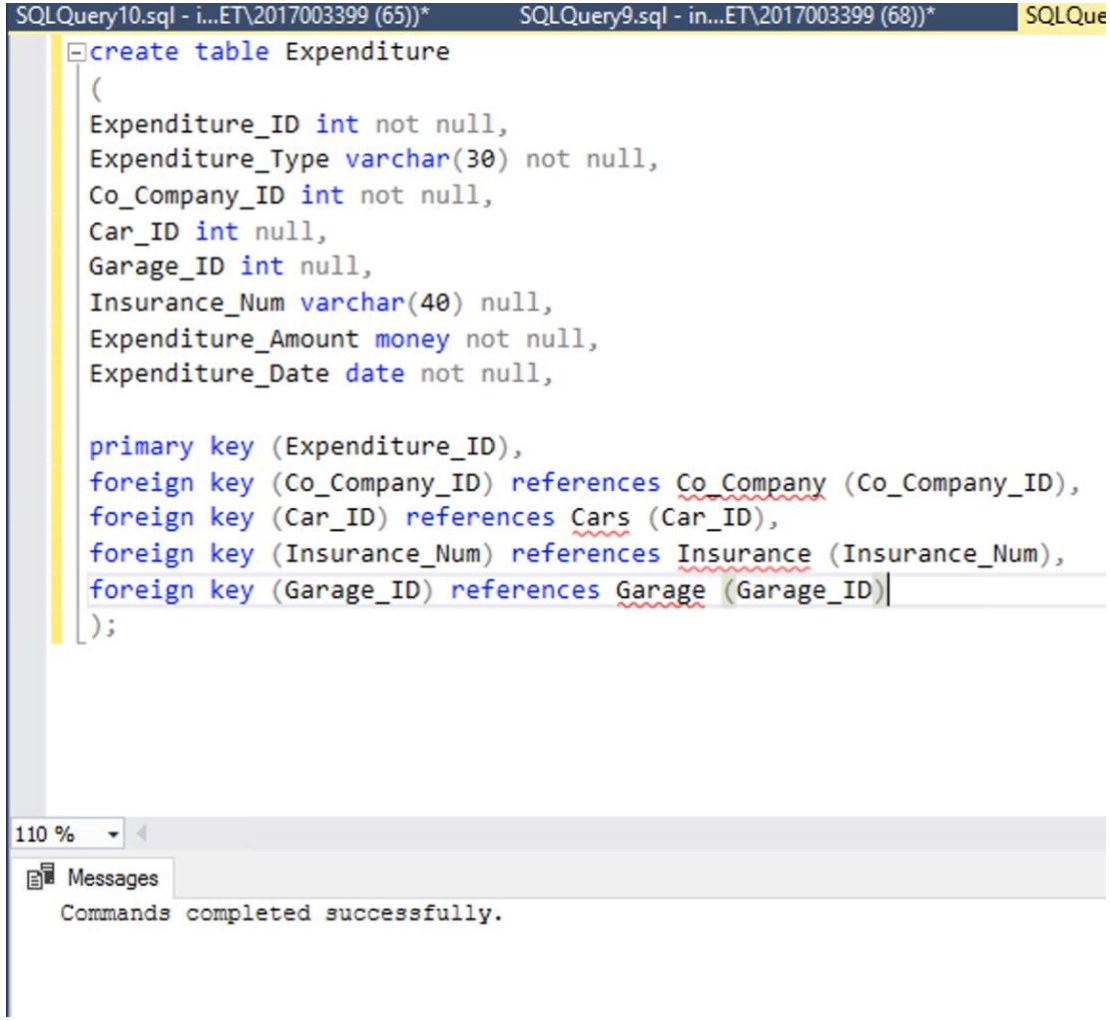
The "Cars" table represents all the cars details including a unique ID for each car (primary key), the factory that the car made by, the year of the car, the engine size, the fuel type, the maximum number of passengers, the registration number of this car, the price and date that this car was purchased by CAR, the established price for daily rental, the insurance number for this car (foreign key, relates to the insurance details), the garage ID (foreign key, relates to the garage further details), and the availability of this car (waiting for sale, sold, on used, or available).



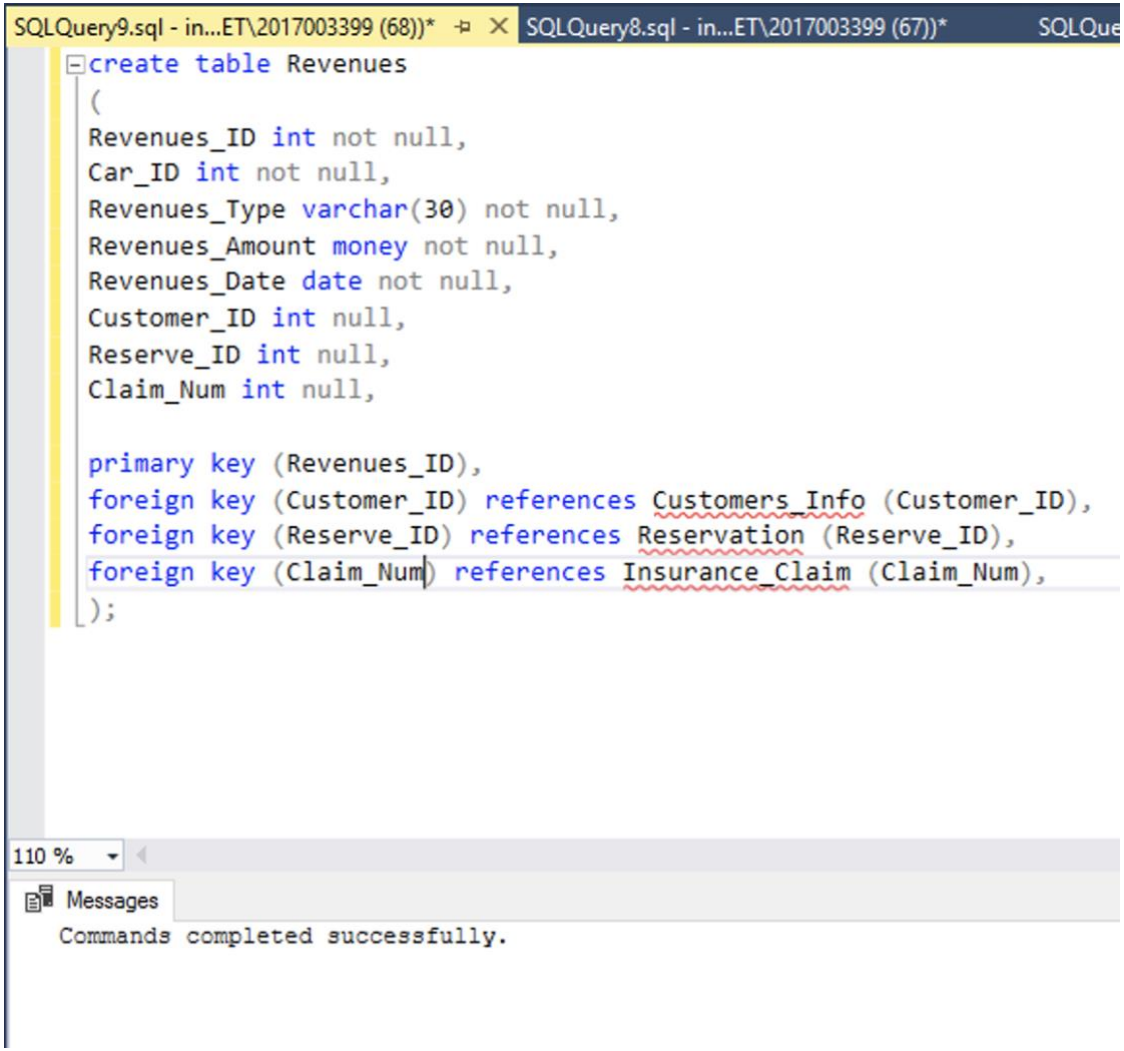
The “Insurance Claim” table represents the information about any insurance been claimed, including the claim ID (primary key), car ID (foreign key, relates to the car information), Insurance number (foreign key, relates to the insurance details), the about of money been claimed, what the issue(s) been claimed, the claim file name related to the relevant document after a claim, the claimed date, and the date that received the fund by insurance company.



The "Reservation" Table contains a set of information about customers reservations, including the **Reservation ID** (primary key), the **Customer ID** (foreign key, relates to the customer information, especially for checking the customer type to get the privilege or not), the **Car ID** (foreign key, relates to the car information, and shows which car was reserved). The **Start date** for this reservation, the **End date** for this reservation, the **Reservation status** (which can be reserved, reserved with credit card, reserved with deposit, ongoing with deposit, ongoing with credit card, privileged reservation ongoing, cancelled, completed not paid rest, and completed paid), the **date of this reservation issued**, **whether the customer needs to paid the deposit** based on the customer type in customer table, the **amount of deposit** (can be null value if the customer has privilege), **the total amount** needs to pay, and the credit card number (if applicable, can be null value).

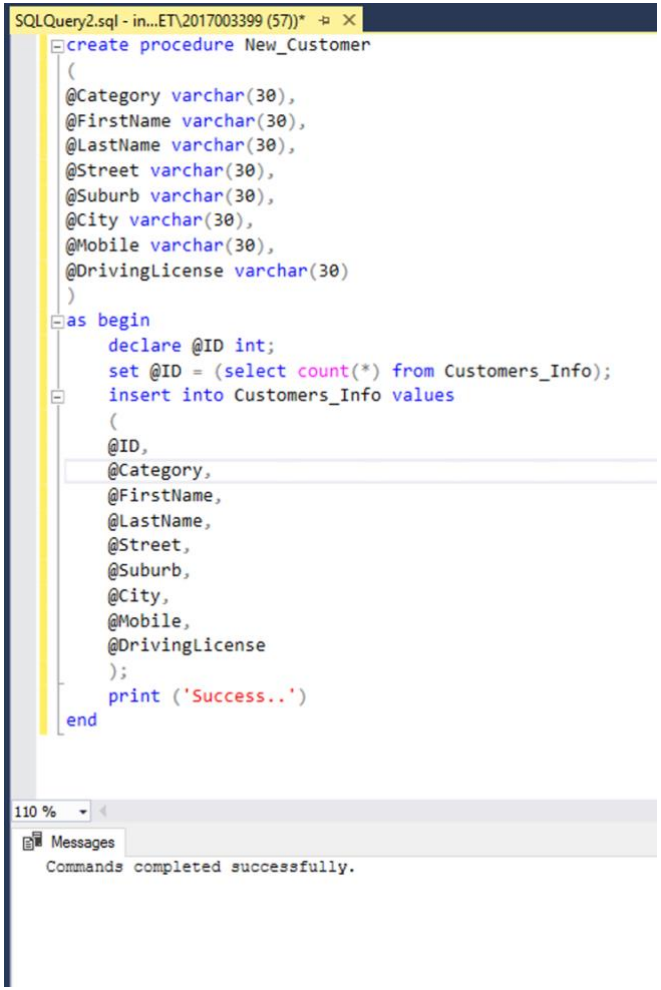


The "Expenditure" table represents all the information regards to the company cost, including the **Expenditure ID** (primary key), the **type of this expenditure**, the **company ID** that they pay for (foreign key, relates to the co-operation company details), the **car ID** (foreign key, relates to the car information. This column can be null value, if the purchase is not related to car, like the daily consumption of stationary), the **garage ID** (foreign key, relates to the garage information, this is also can be null, if the expenditure is not for repair), the **insurance num** (foreign key, relates to the insurance details, it is can be null value, when the purchase is not about to get or update the insurance for the car), the **amount** of this expenditure, and **the date** of this expenditure.

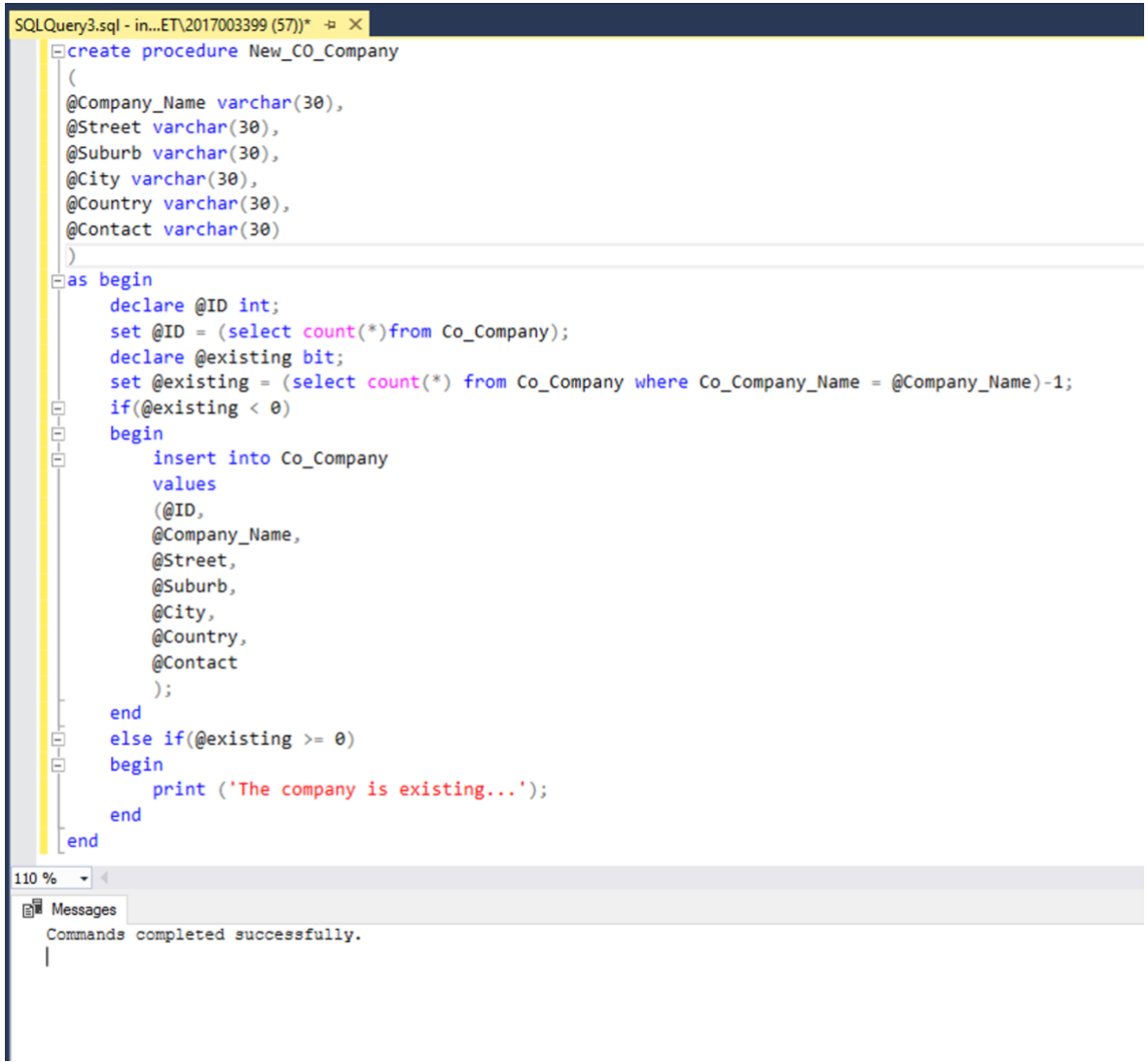


The "Revenue" table relates to the information about company incomes, including the **revenues ID** (primary key), **the car ID** (foreign key, relates to the car information), the **revenue type** (can be rental, reserved rental, car sale, and insurance claim), the **amount** of income received, the **date** of the income received, the **customer ID** (foreign key, relates to customer information, this is can be null value, if the income is insurance claim), **reserve ID** (foreign key, relates to the reservation information, this is can be null value, if the income is not the type of reserved rental), and the **claim num**ber (foreign key, relates to the insurance claim information. This is can be a null value if the income is not the type of insurance claim).

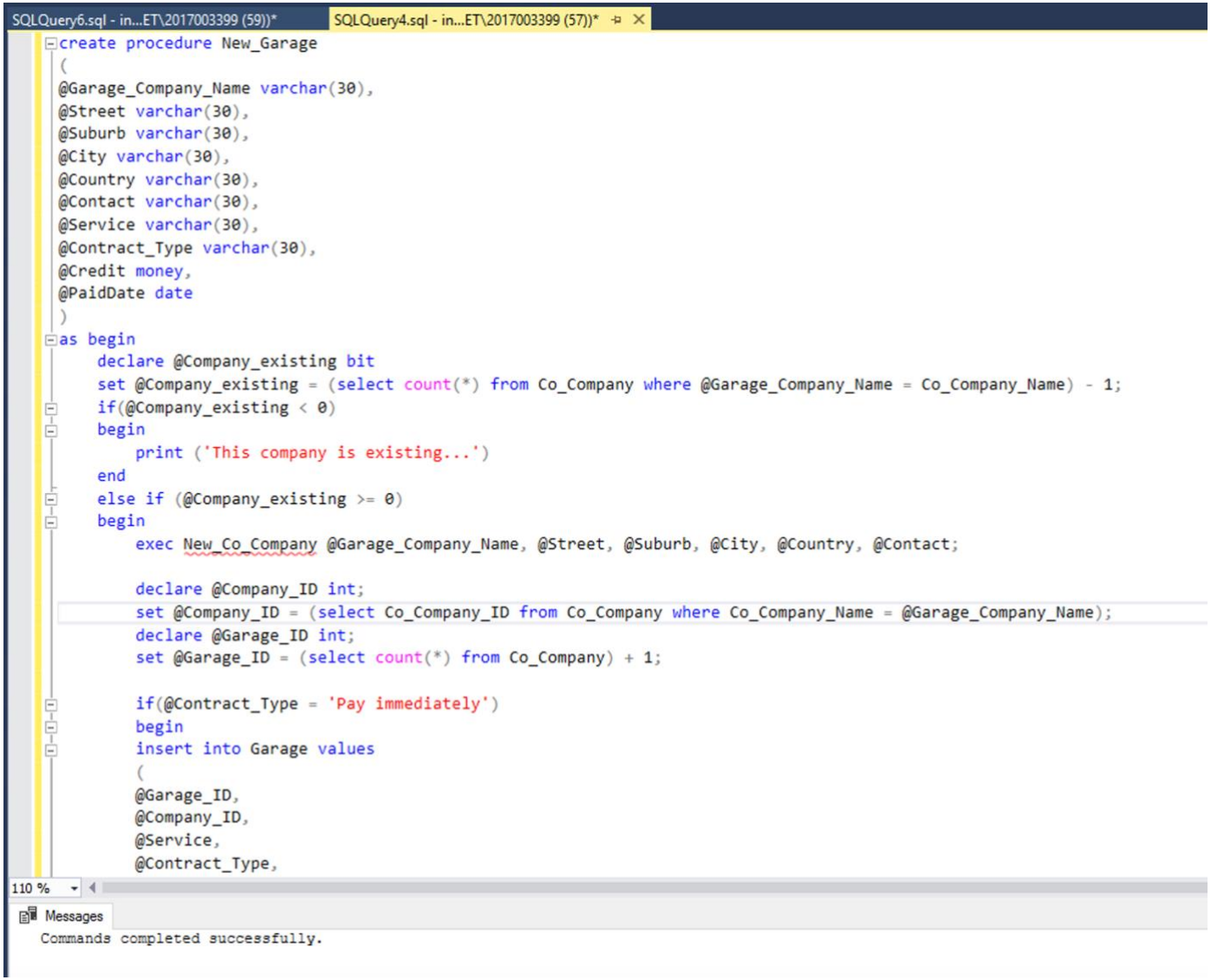
Part2: Procedures



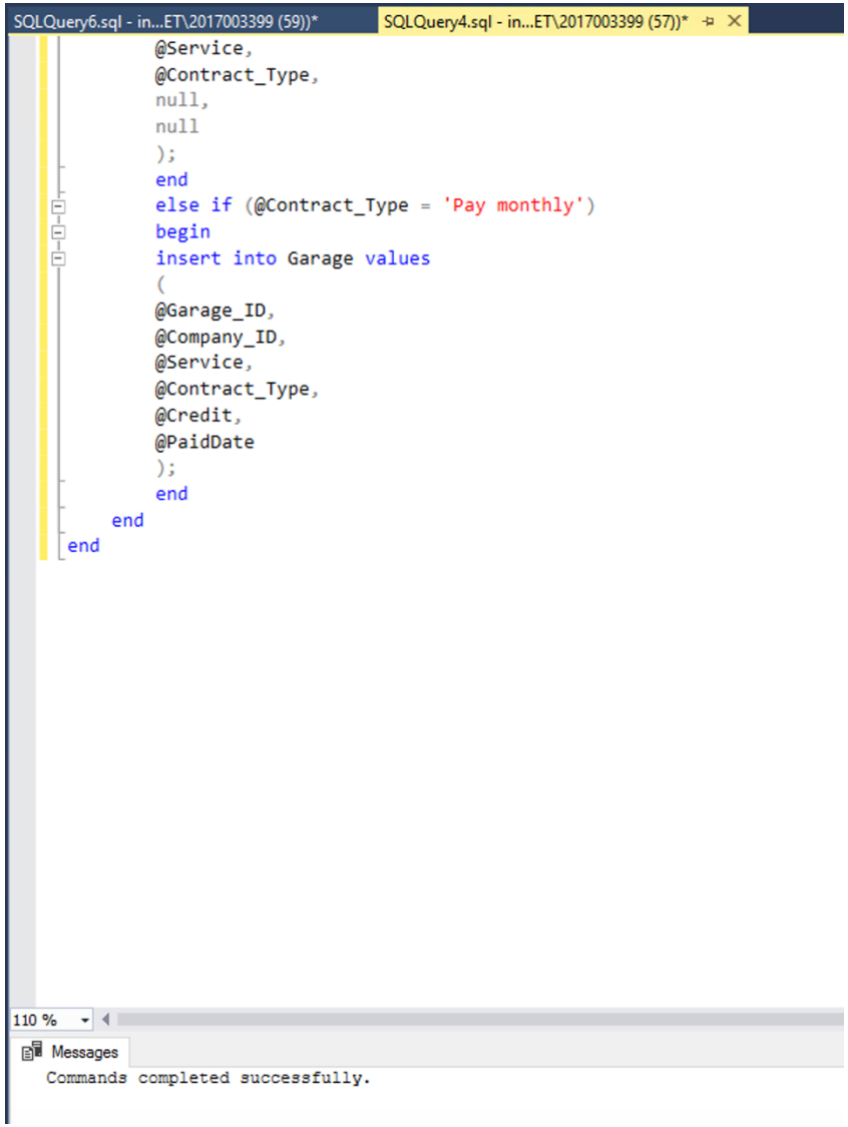
This procedure mainly designed for inserting new customer.



This procedure just for inserting a new company, it will help user to automatically check the company name if it exists.



New Garage procedure Part 1



New Garage procedure Part 2

This procedure will help to insert a new garage based on the contract categories which are Immediately pay or pay monthly. Also, it is designed to check if the garage company name is invalid, then will suggest user to insert company information first.

SQLQuery4.sql - in...ET\2017003399 (57))*

```
create procedure New_Car
(
@Car_Company_Name varchar(30),
@Make varchar(30),
@Year date,
@Engin int,
@Fuel varchar(10),
@Passengers int,
@Resgitration varchar(30),
@Purcharsed_Price money,
@Purcharsed_Date date,
@Daily_Rental_Fees money,
@Insurance_Num varchar(40),
@Insurance_Price money,
@Insurance_Company_Name varchar(40),
@Insurance_StartDate date,
@Insurance_EndDate date,
@Insurance_FileName varchar(40),
@Garage_ID int
)
as begin
    declare @Garage_valid bit;
    set @Garage_valid = (select count(*) from Garage where Garage_ID = @Garage_ID);

    /*check garage*/
    if(@Garage_valid < 0)
    begin
        print('Invalid Garage ID or Garage is not existing...')
    end
    else if (@Garage_valid >= 0)
    begin
        declare @Insurance_Company_existing bit;
        set @Insurance_Company_existing =
        (select count(*) from Co_Company where Co_Company_Name = @Insurance_Company_Name) - 1;
        /*check insurance company*/
        if(@Insurance_Company_existing < 0)
        begin
```

110 %

Messages

```
Commands completed successfully.
```

Purchase New Car Part 1

SQLQuery4.sql - in...ET\2017003399 (57))*

```
        begin
            print ('Unknown insurance company, please insert information of this company first...')
        end
        else if (@Insurance_Company_existing >= 0)
        begin
            declare @Car_Company_exisitng bit;
            set @Car_Company_exisitng = (select count(*) from Co_Company where Co_Company_Name = @Car_Company_Name) - 1;
            if(@Car_Company_exisitng < 0)
            begin
                print ('Unknow Car Company, please complete record in the Co_Company first...')
            end
            else if(@Car_Company_exisitng >= 0)
                begin

                declare @Insurance_Company_ID int;
                set @Insurance_Company_ID = (select Co_Company_ID from Co_Company where Co_Company_Name = @Insurance_Company_Name);

                declare @Car_ID int;
                set @Car_ID = (select count(*) from Cars) + 1;

                /*insert record in Insurance*/
                insert into Insurance values
                (
                @Insurance_Num,
                @Insurance_Company_ID,
                @Insurance_StartDate,
                @Insurance_EndDate,
                @Insurance_FileName
                );
                /*insert record in Cars*/
                insert into Cars values
                (
                @Car_ID,
                @Make,
                @Year,
                @Engin,
                @Fuel,
```

110 %

Messages

```
Commands completed successfully.
```

Purchase New Car Part 2

SQLQuery4.sql - in...ET\2017003399 (57))*

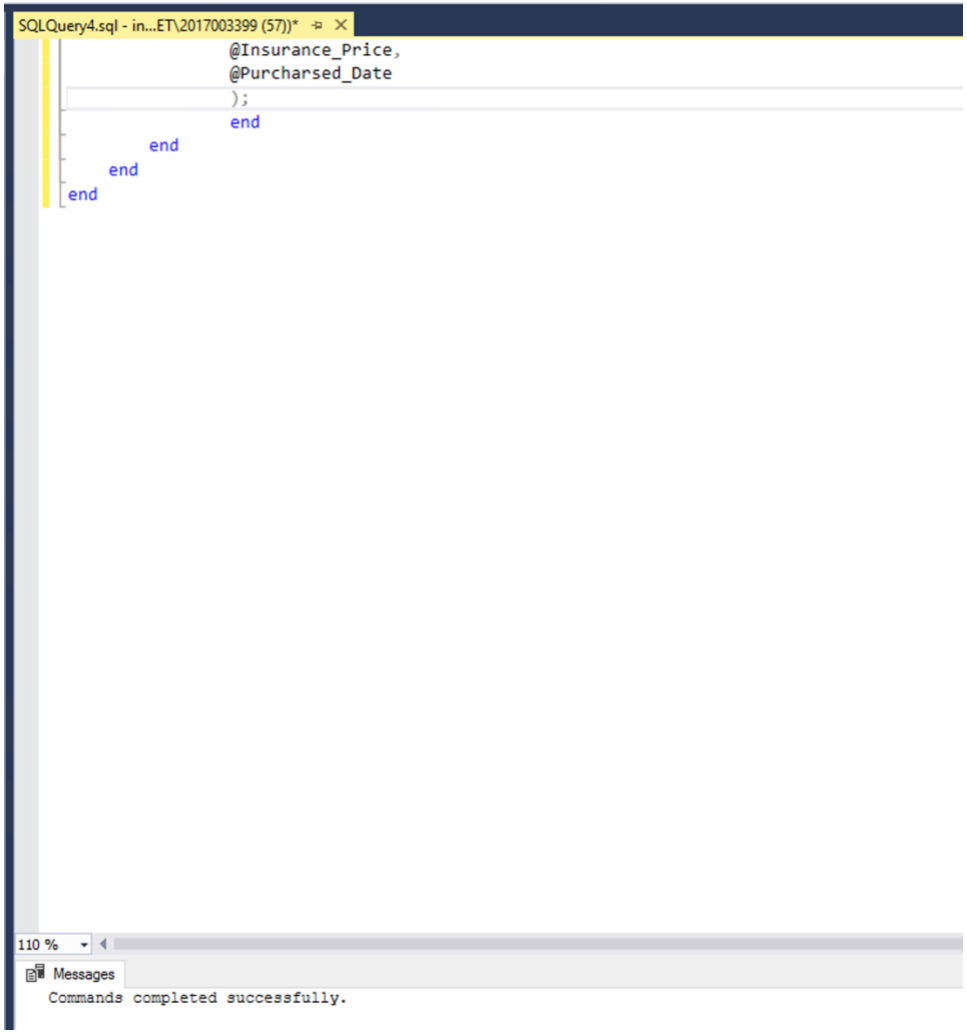
```
@Fuel,
@Passengers,
@Resgitration,
@Purcharsed_Price,
@Purcharsed_Date,
@Daily_Rental_Fees,
@Insurance_Num,
@Garage_ID,
'Available'
);
/*insert record in Expenditure*/
declare @E_ID int;
set @E_ID = (select count(*) from Expenditure) + 1;
declare @Car_Company_ID int;
set @Car_Company_ID = (select Co_Company_ID from Co_Company where @Car_Company_Name = Co_Company_Name);
insert into Expenditure values
(
@E_ID,
'Purchase car',
@Car_Company_ID,
@Car_ID,
'',
'',
@Purcharsed_Price,
@Purcharsed_Date
);
insert into Expenditure values
(
@E_ID + 1,
'Purchase car insurance',
@Insurance_Company_ID,
@Car_ID,
'',
@Insurance_Num,
@Insurance_Price,
@Purcharsed_Date
);
```

110 %

Messages

```
Commands completed successfully.
```

Purchase New Car Part 3



Purchase New Car Part 4

This procedure will help to check if any information is invalid like insurance company or car factory name, then will recommend user to insert relative information first. If everything correct, then to insert all information related to new car, car's insurance and the garage assigned for car repair, then will generate an expenditure record for this purchase.

SQLQuery5.sql - in...ET\2017003399 (60))* | SQLQuery4.sql - in...ET\2017003399 (59))* | SQLQuery3.sql - in...ET\2017003399 (57))* | SQLQuery2.sql - in...ET\2017003399 (56))*

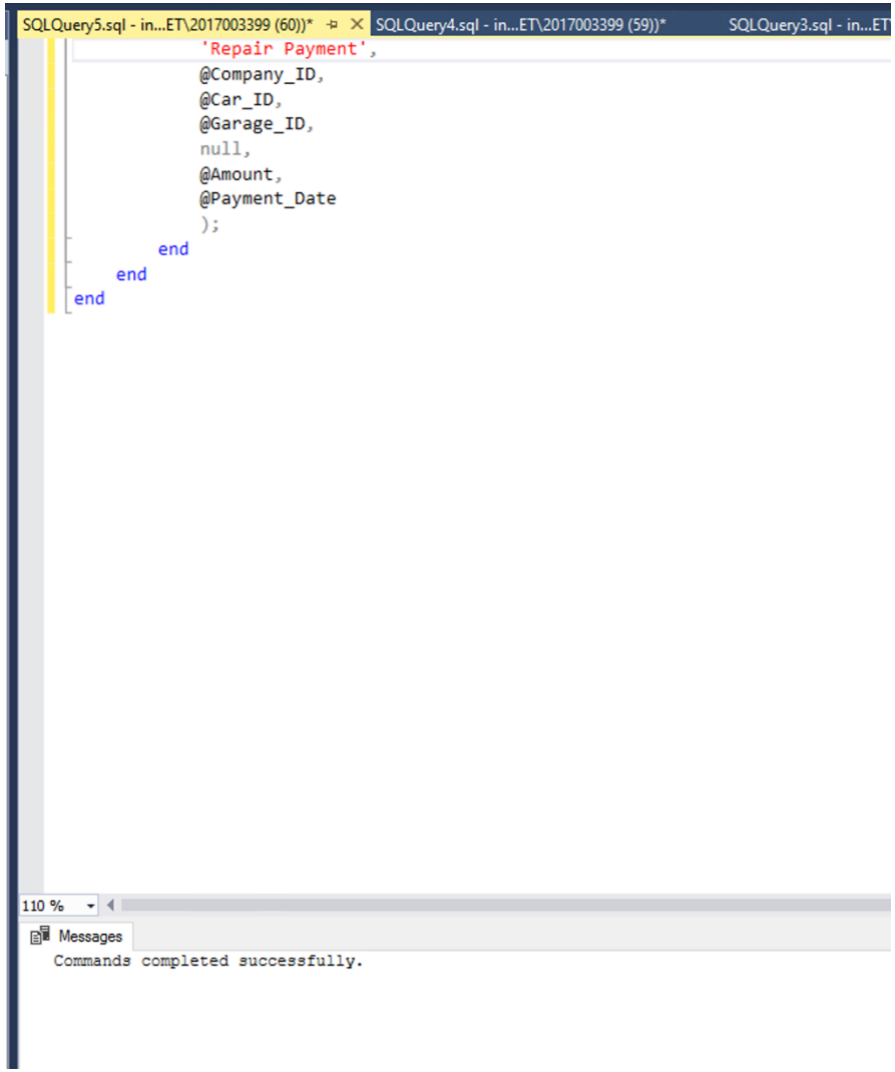
```
create procedure Repair_Payment
(
@Garage_Company_Name varchar(30),
@Garage_ID int,
@Car_ID int,
@Amount money,
@Payment_Date datetime
)
as begin
    declare @Garage_Company_existing int;
    declare @Garage_existing int;
    set @Garage_Company_existing = (select count(*) from Co_Company where @Garage_Company_Name = Co_Company_Name) - 1;
    set @Garage_existing = (select count(*) from Garage where Garage_ID = @Garage_ID) - 1;

    if(@Garage_Company_existing < 0)
    begin
        print ('Cannot find Company information...')
    end
    else if(@Garage_Company_existing = 0)
    begin
        if(@Garage_existing < 0)
        begin
            print ('Cannot find garage information...')
        end
        else if(@Garage_existing = 0)
        begin
            declare @E_ID int;
            set @E_ID = (select count(*) from Expenditure) + 1;
            declare @Company_ID int;
            set @Company_ID = (select Co_Company_ID from Co_Company where @Garage_Company_Name = Co_Company_Name);
            insert into Expenditure values
            (
            @E_ID,
            'Repair Payment',
```

110 %

Messages

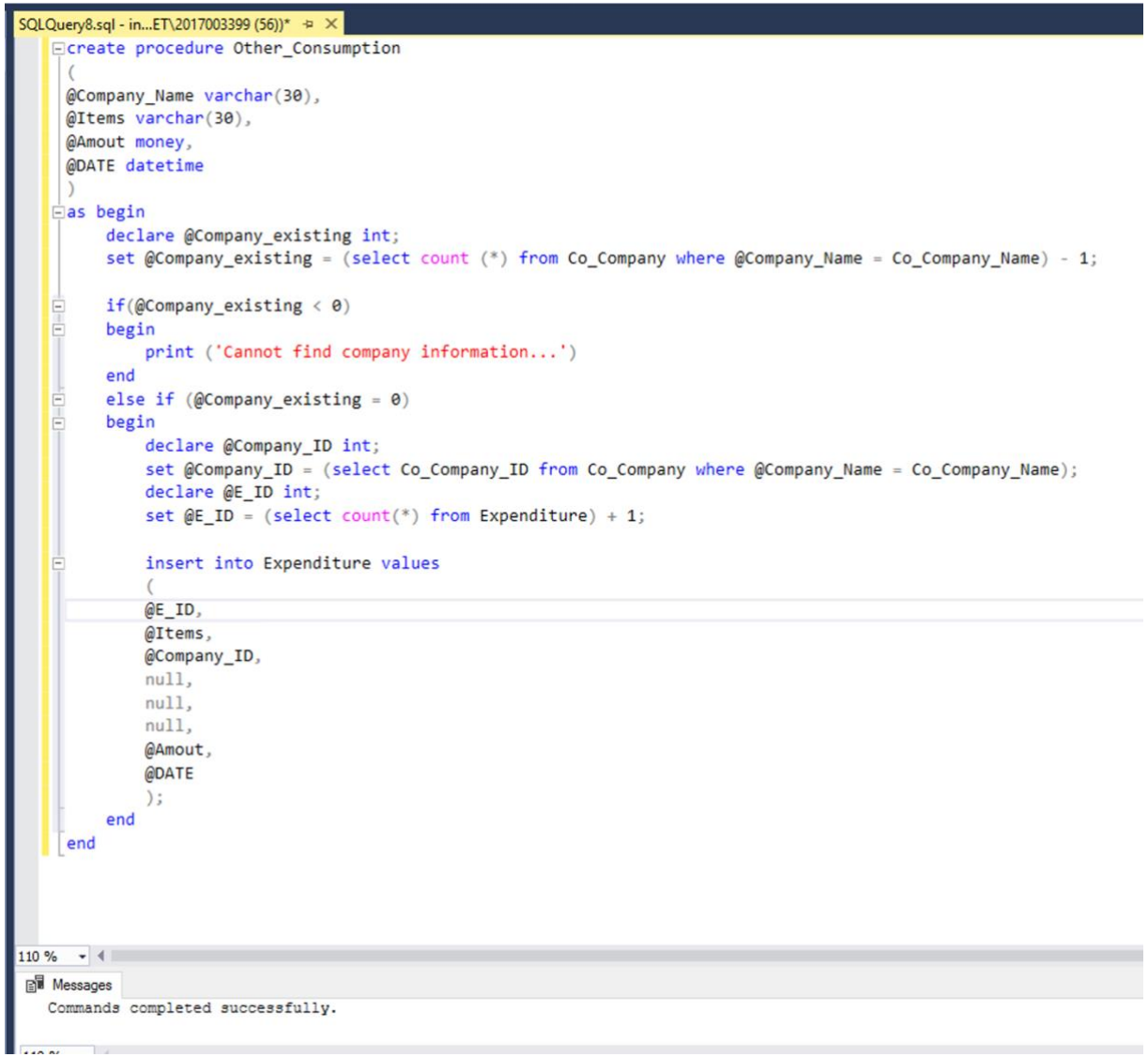
```
Commands completed successfully.
```

Repair Payment procedure part 1



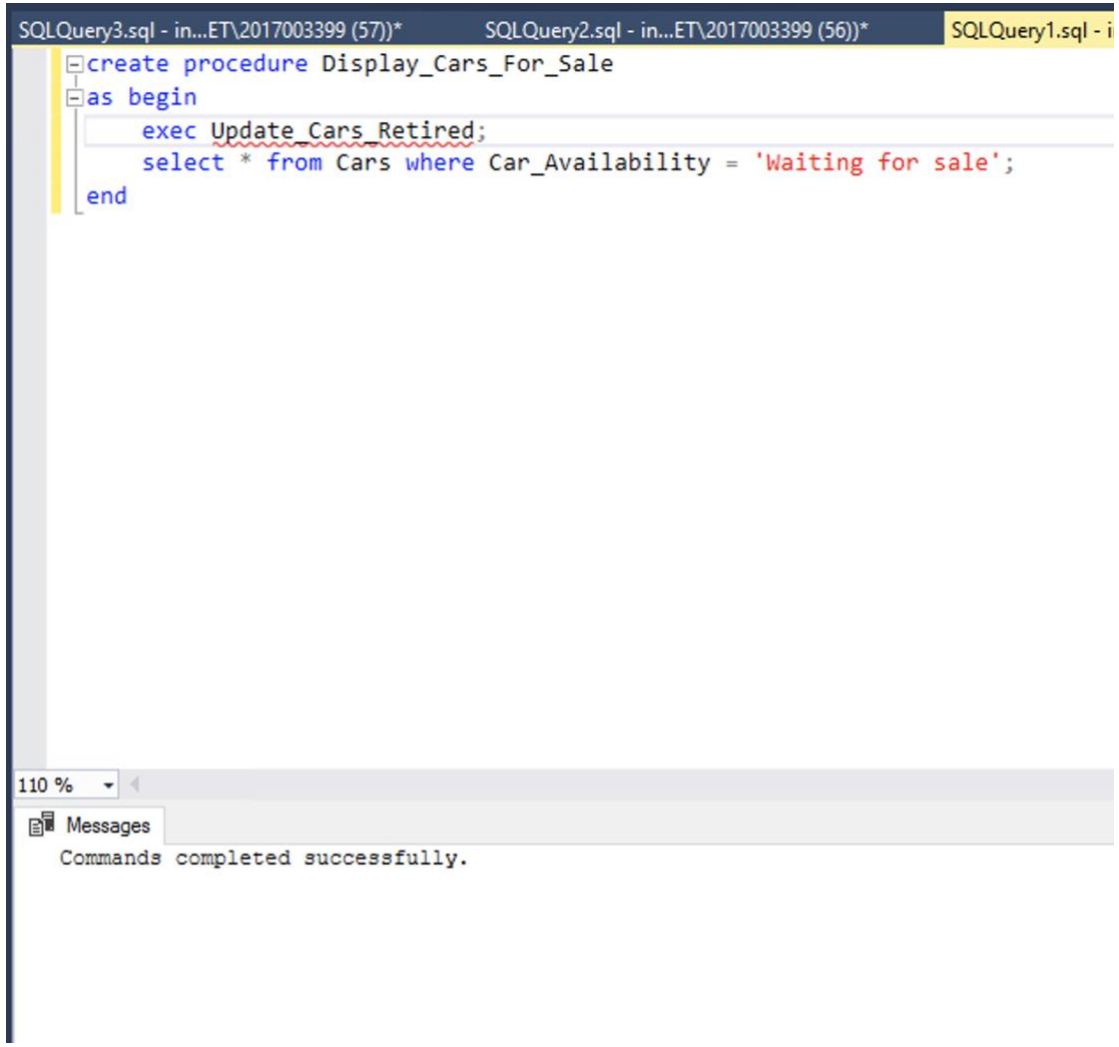
Repair Payment procedure part 2

This procedure is used for when CAR needs to pay for repair, then to create a new expenditure to sign relevant information in the relative columns. Also, it is designed to check if garage company does not exist in the Co_Company table, then recommend to insert those information first.



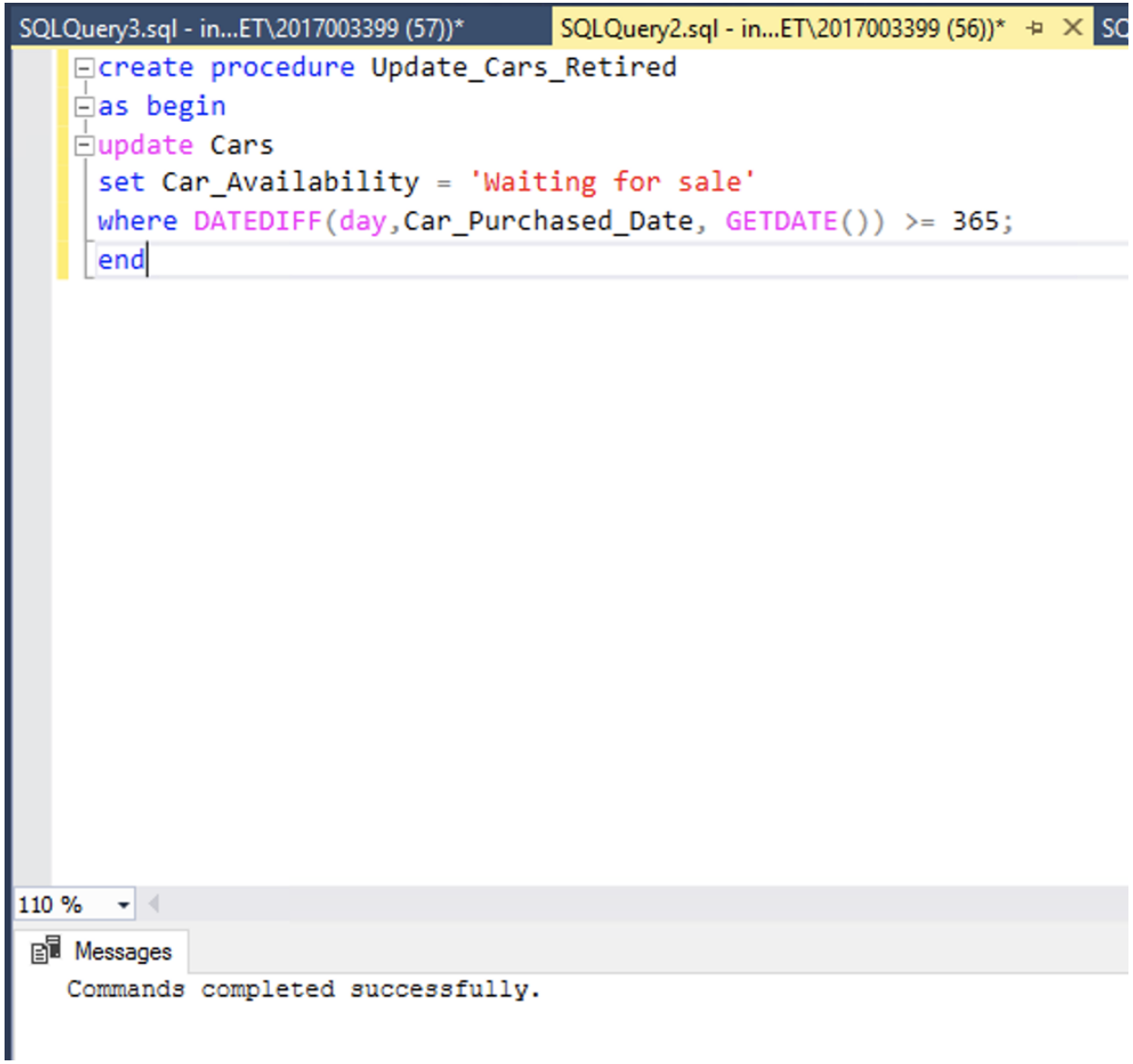
Other consumption procedure

In case of CAR might have some other daily consumptions, so this procedure is designed for this aim.

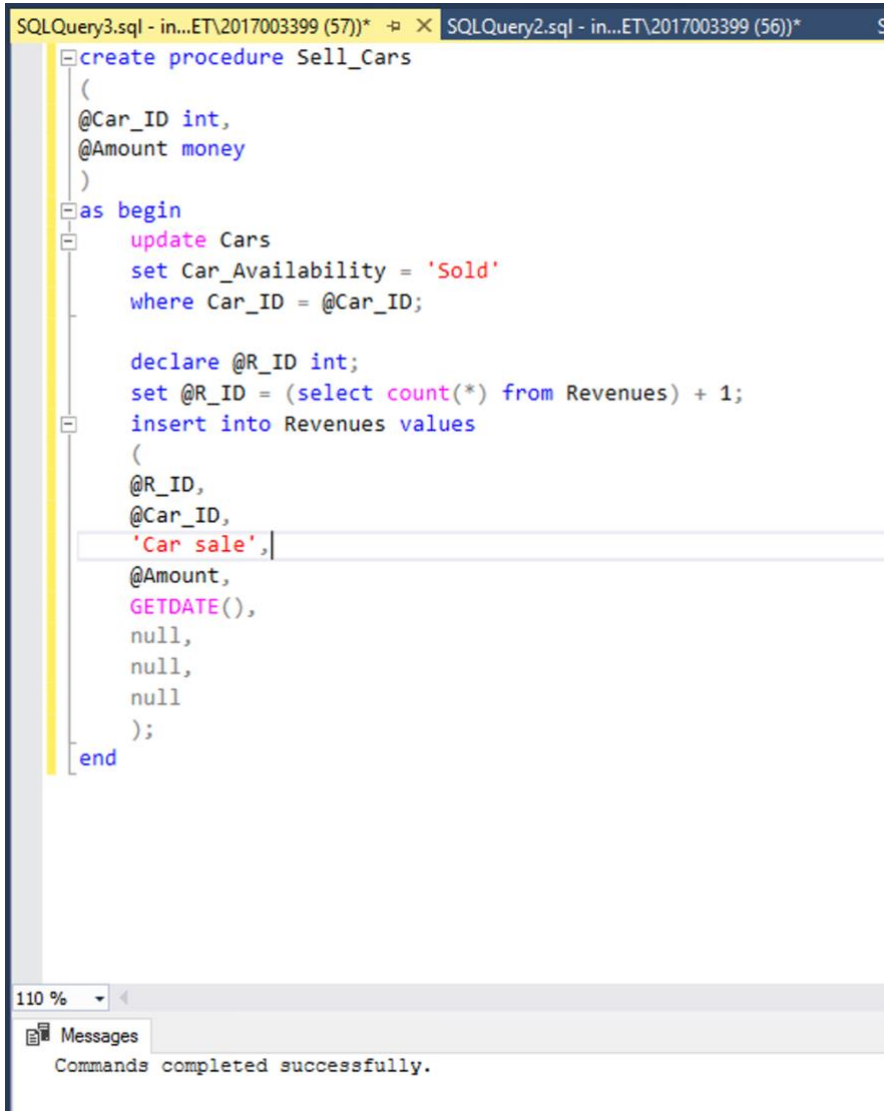


Display Cars need to sell

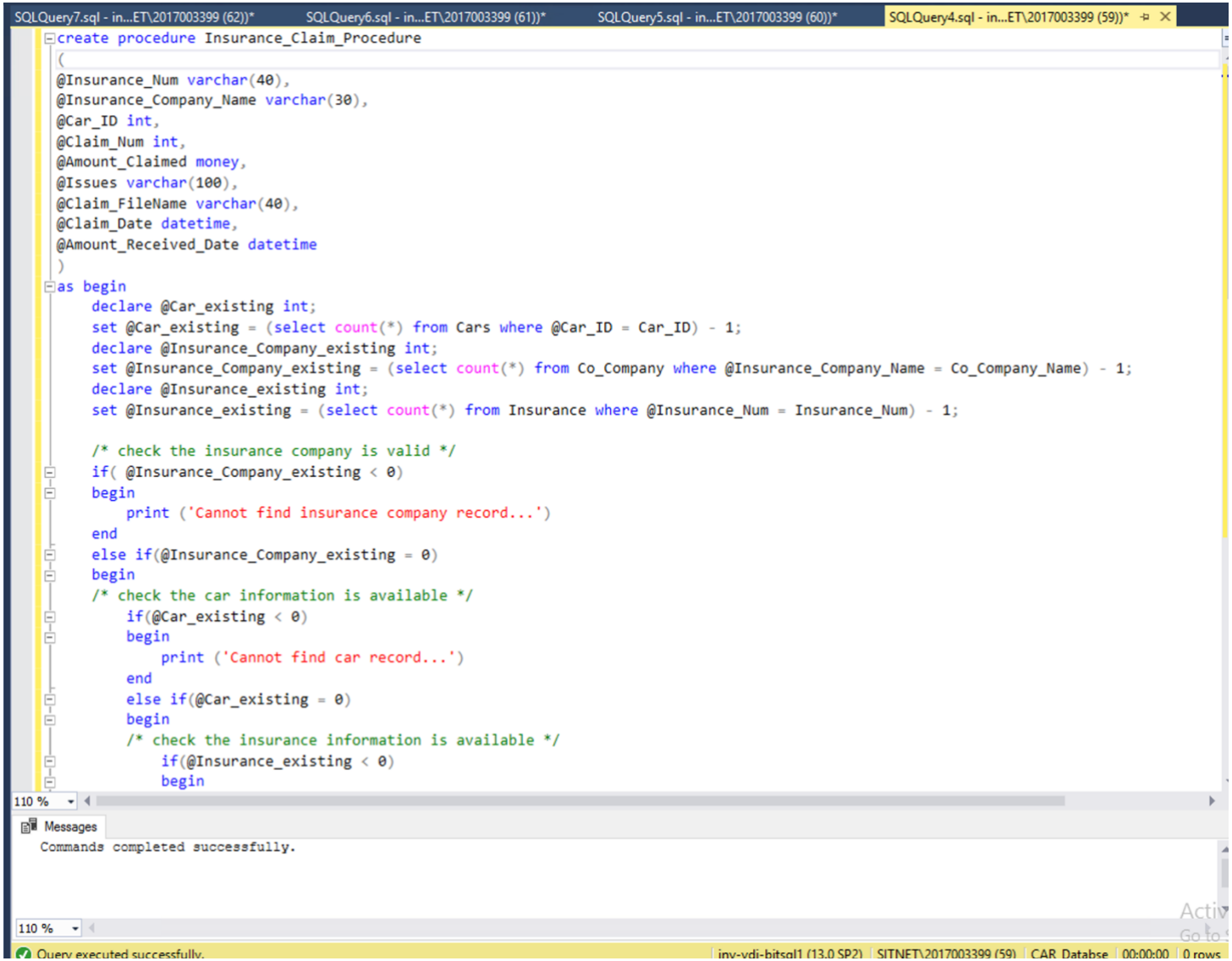
This procedure is designed to show how many cars need to sell, according to the policy of CAR, “All cars will not be hold for a period exceeding one year”.



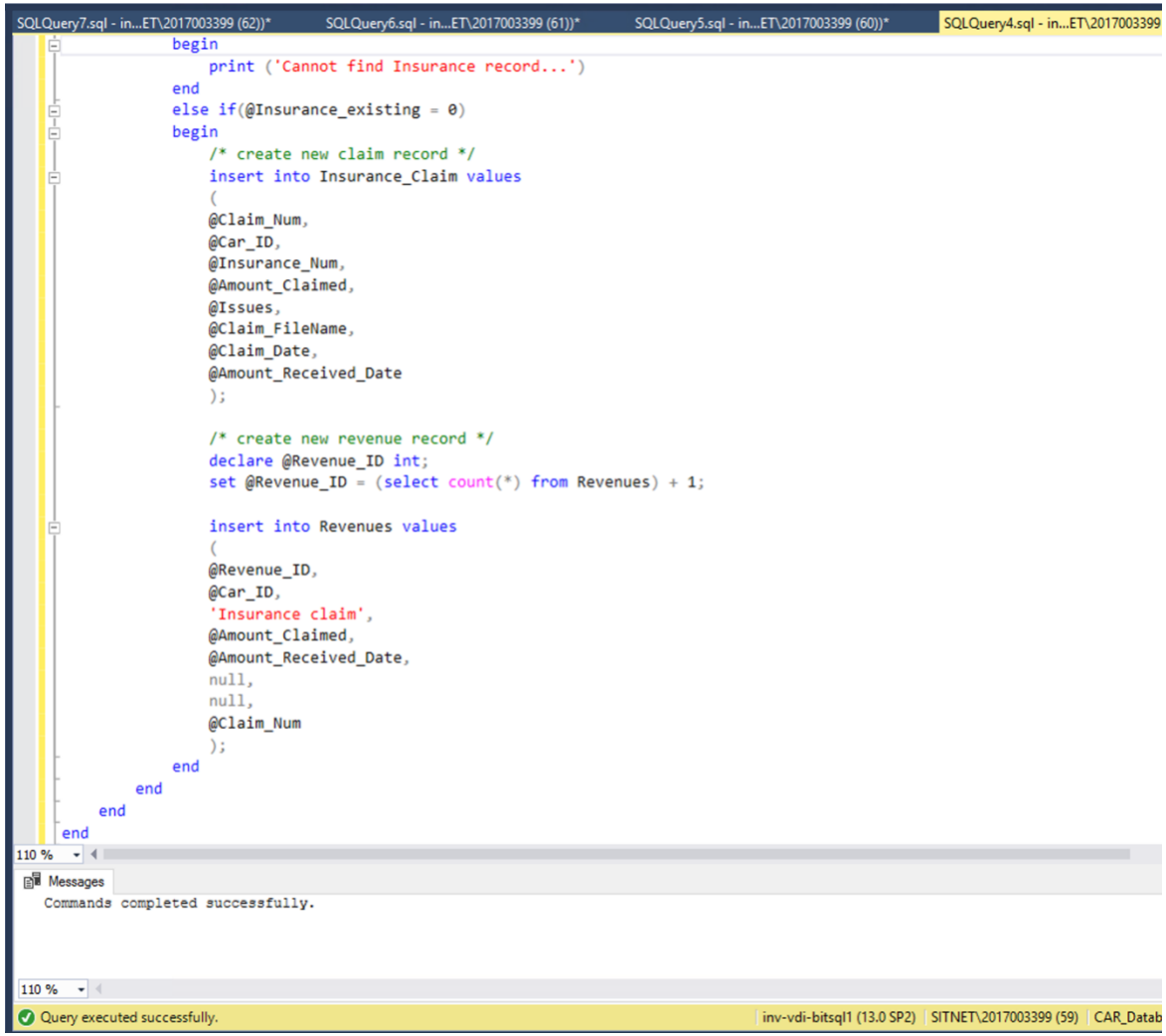
This procedure aims for supporting the procedure above, to update all cars need to sell. (days on hold over 364 days)



When the car has been sold, user needs to insert the car ID and amount of money been received to this procedure, then it will automatically update the car status to sold and create a new revenue record with relevant information.



Insurance Claim procedure Part 1



Insurance claim procedure Part 2

This procedure is for inserting a new insurance claim record in the Insurance Claim table and generate a new revenue record in Revenue table.

SQLQuery9.sql - in...ET\2017003399 (62))* | SQLQuery8.sql - in...ET\2017003399 (57))* | SQLQuery7.sql - in...ET\2017003399 (61))* | SQLQuery6.sql - in...ET\201700

```
create procedure Display_Cars_Availability
as begin
    /*Update car availability if the car has been reserved, and the reservation will be started within 60 minutes*/
    update Cars
    set Car_Availability = 'Reserved'
    where Car_ID =
    (select Car_ID from Reservation where datediff(MINUTE,getdate(),Reserve_StartDate) < 60);
    /*Update car availability if the car exceeds 365 days since CAR purchased*/
    exec Update_Cars_Retired;
    /*display the cars available for booking*/
    select Car_ID,Car_Make,Car_Year, Car_Engine, Car_Fuel, Car_NumOfPassengers
    , Car_Daily_Rental_Fees, Car_Availability from Cars
    where Car_Availability = 'Available'
end
```

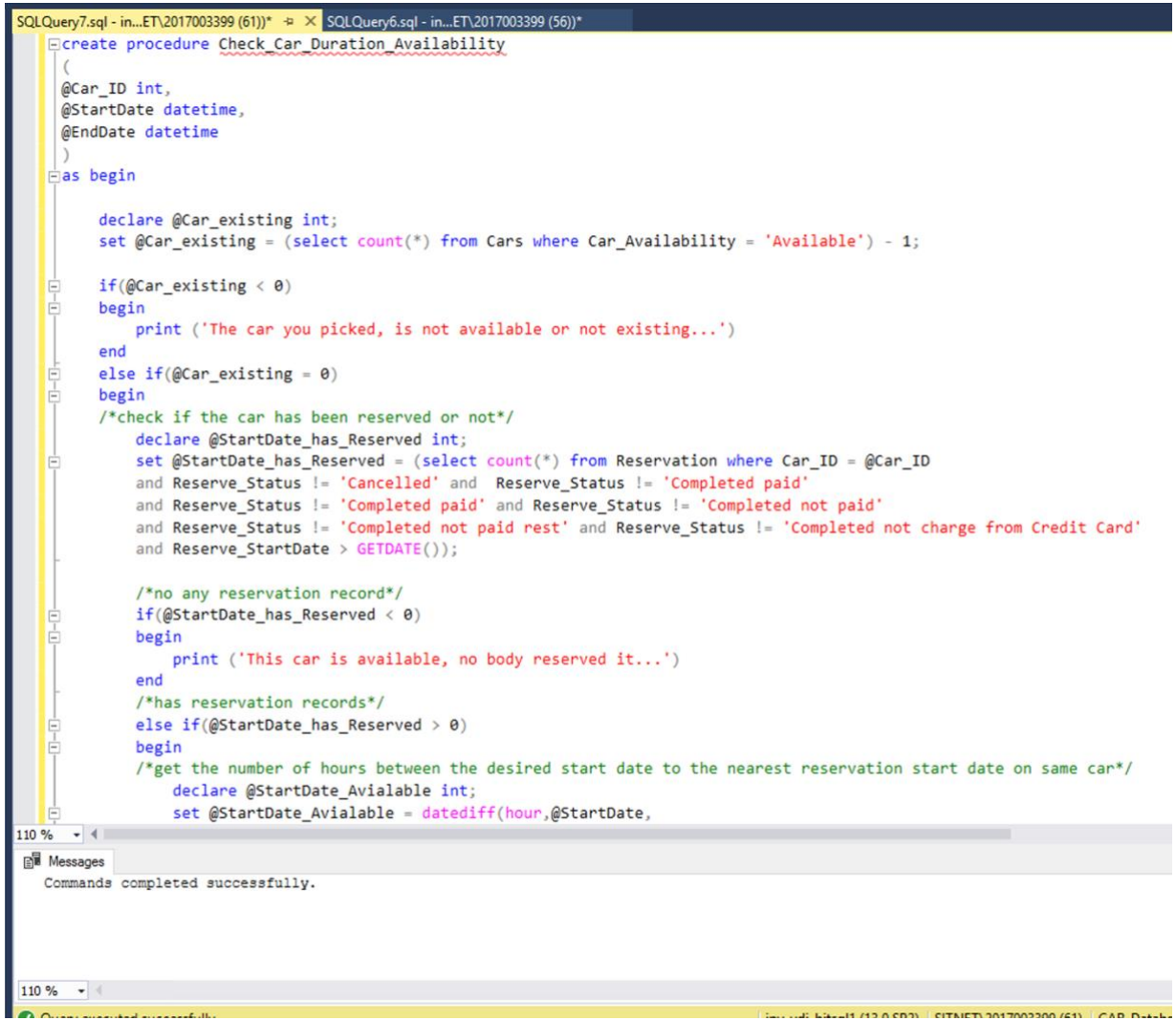
110 %

Messages

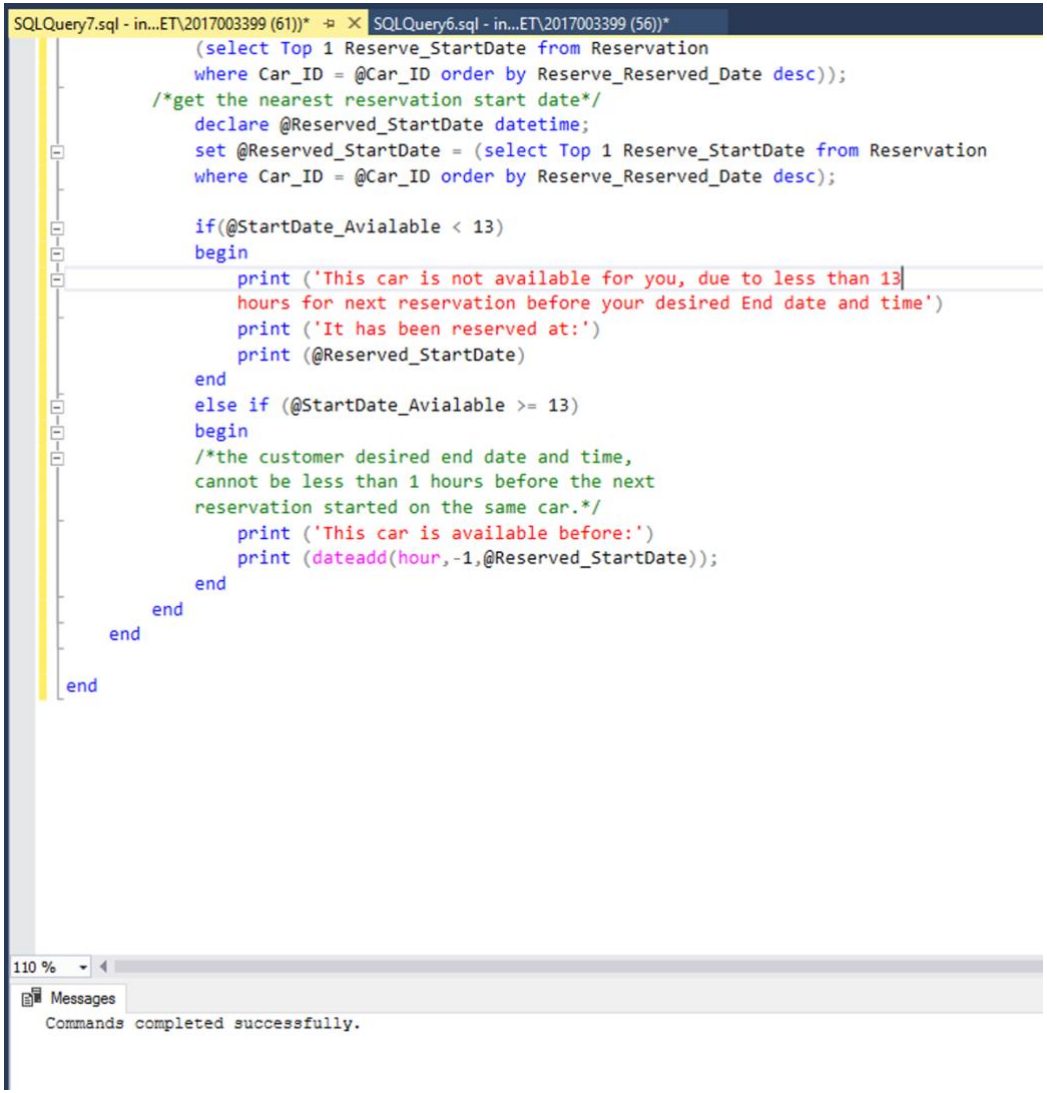
```
Commands completed successfully.
```

Booking system 1 – Display Cars Availability

It will update every car to “Reserved” if it has reservation which will be started within 60 minutes, then display the cars are available for rental.

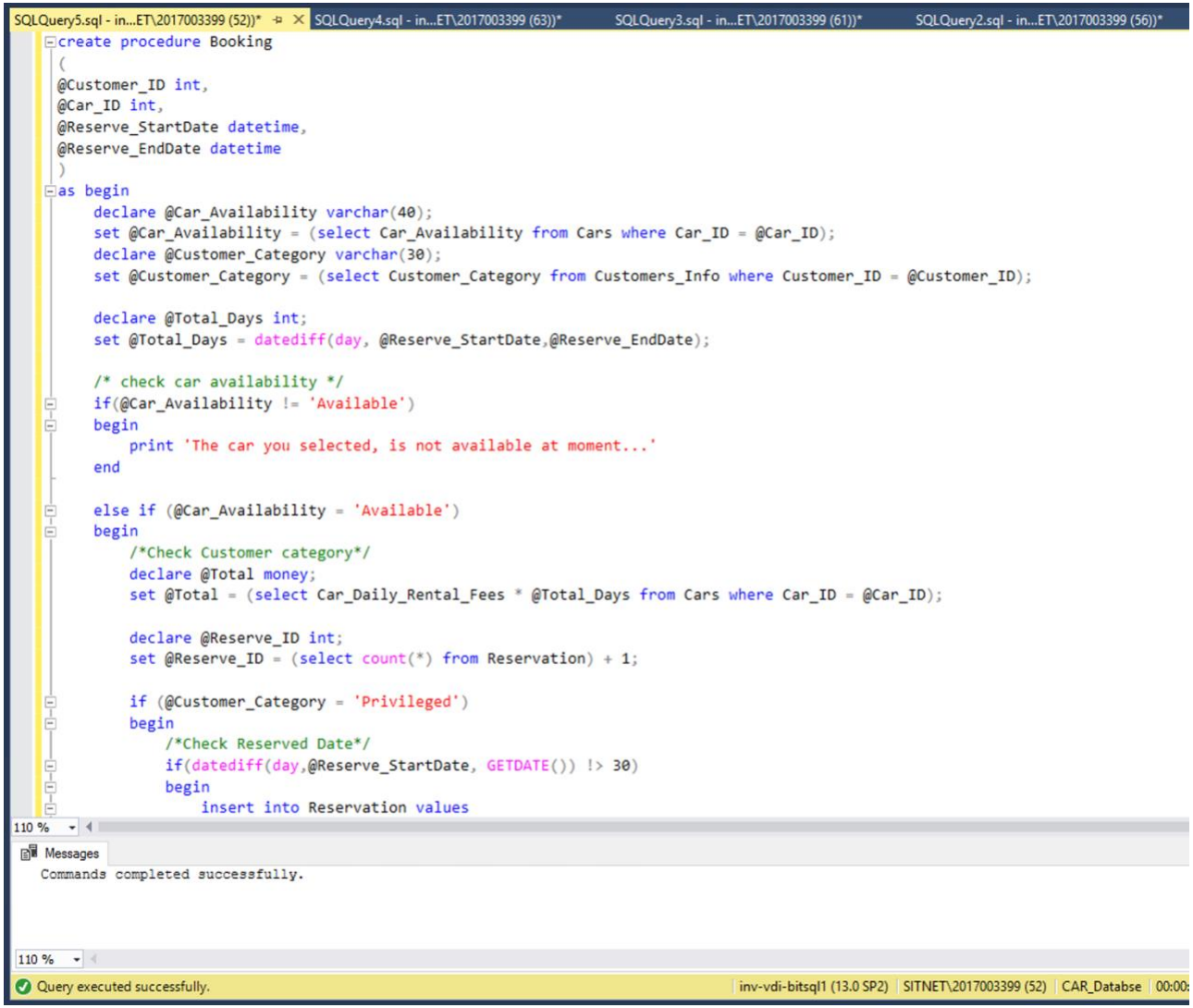


Booking system 2 – Check Desired Rental duration availability of the car Part 1

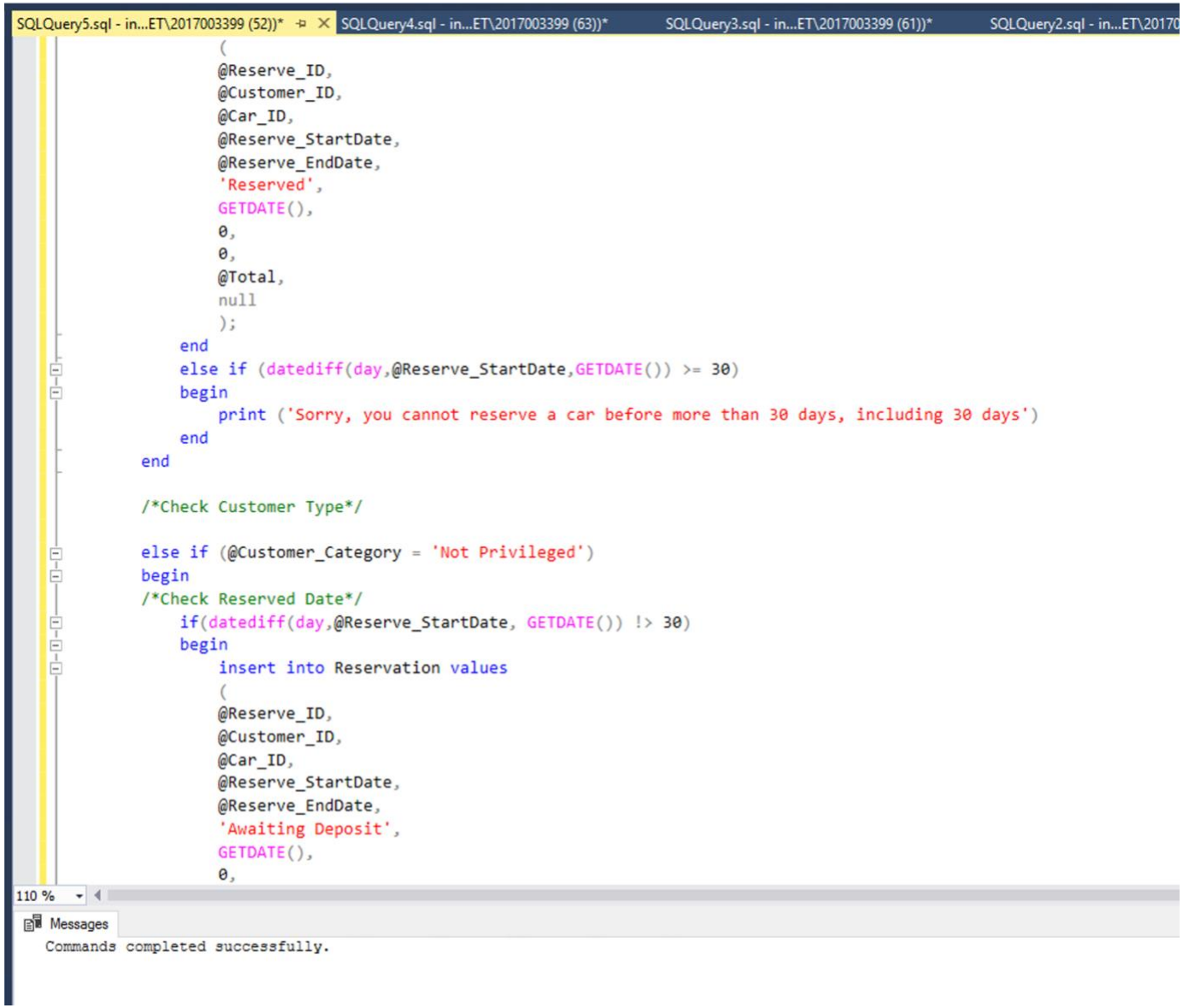


Booking system 3 – Check desired rental duration availability of the car Part 2

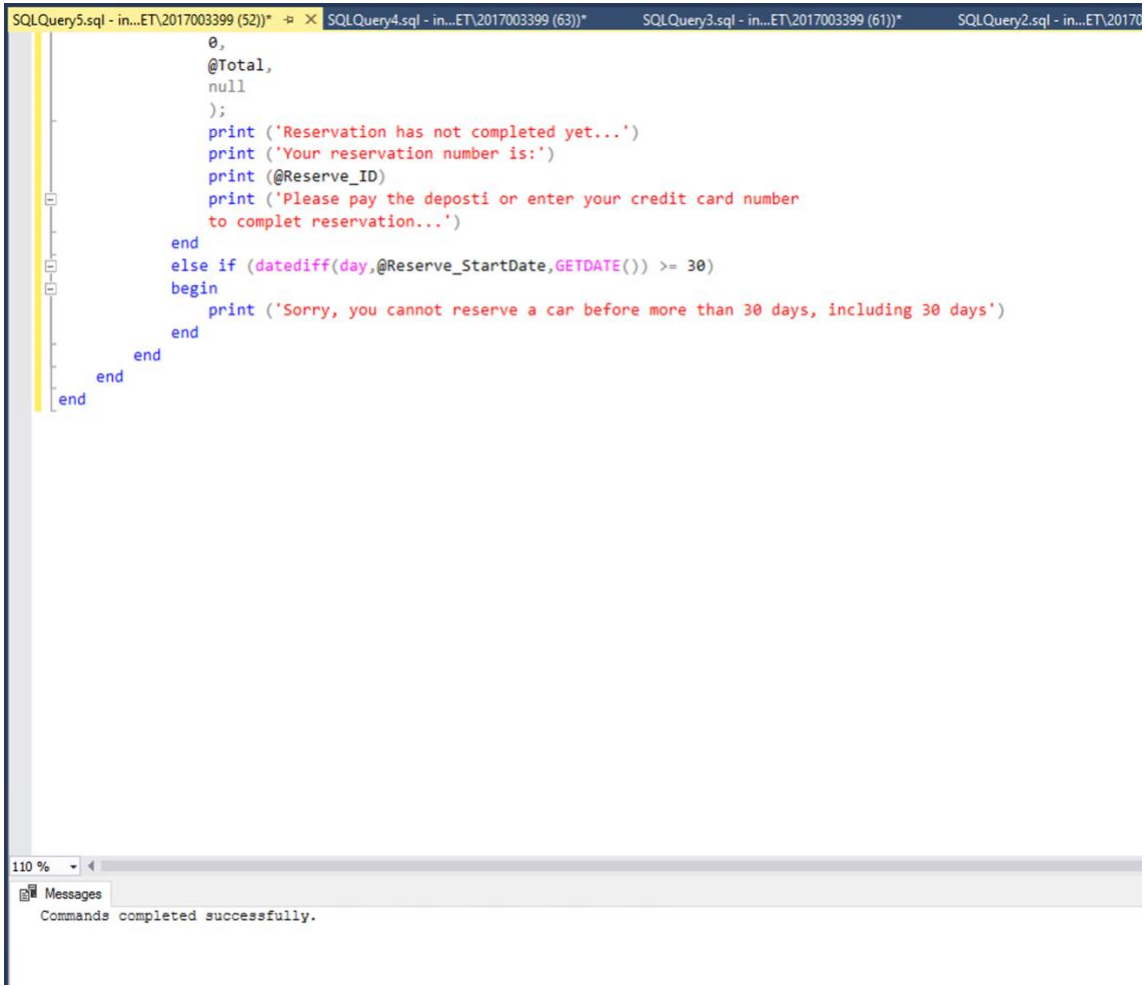
This procedure will help customer to check whether their desired period of time for car rental is suitable for the car itself, and also can tell when will the nearest reservation started.



Booking system 4 – Booking procedure Part 1

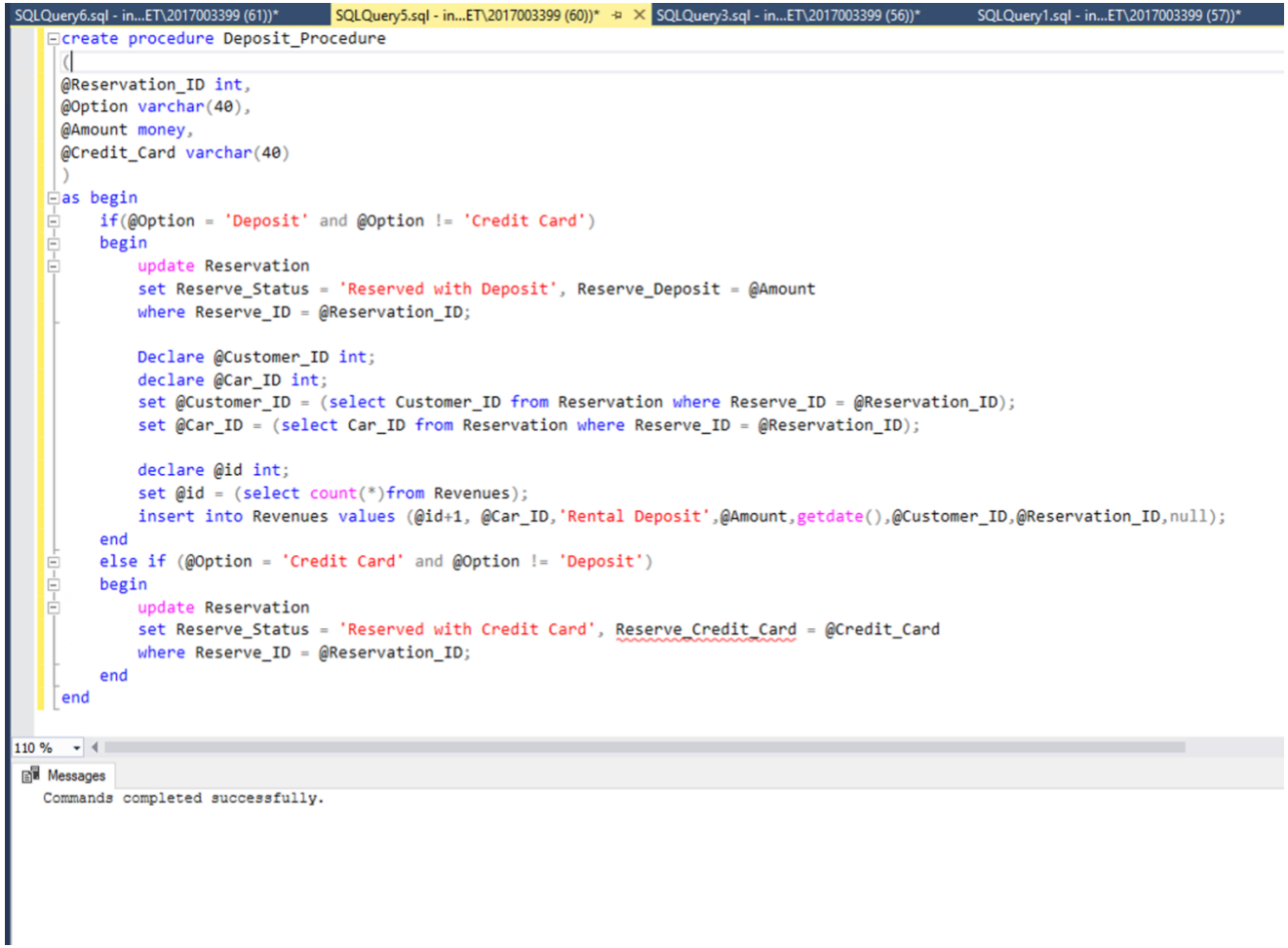


Booking system 5 – Booking procedure Part 2



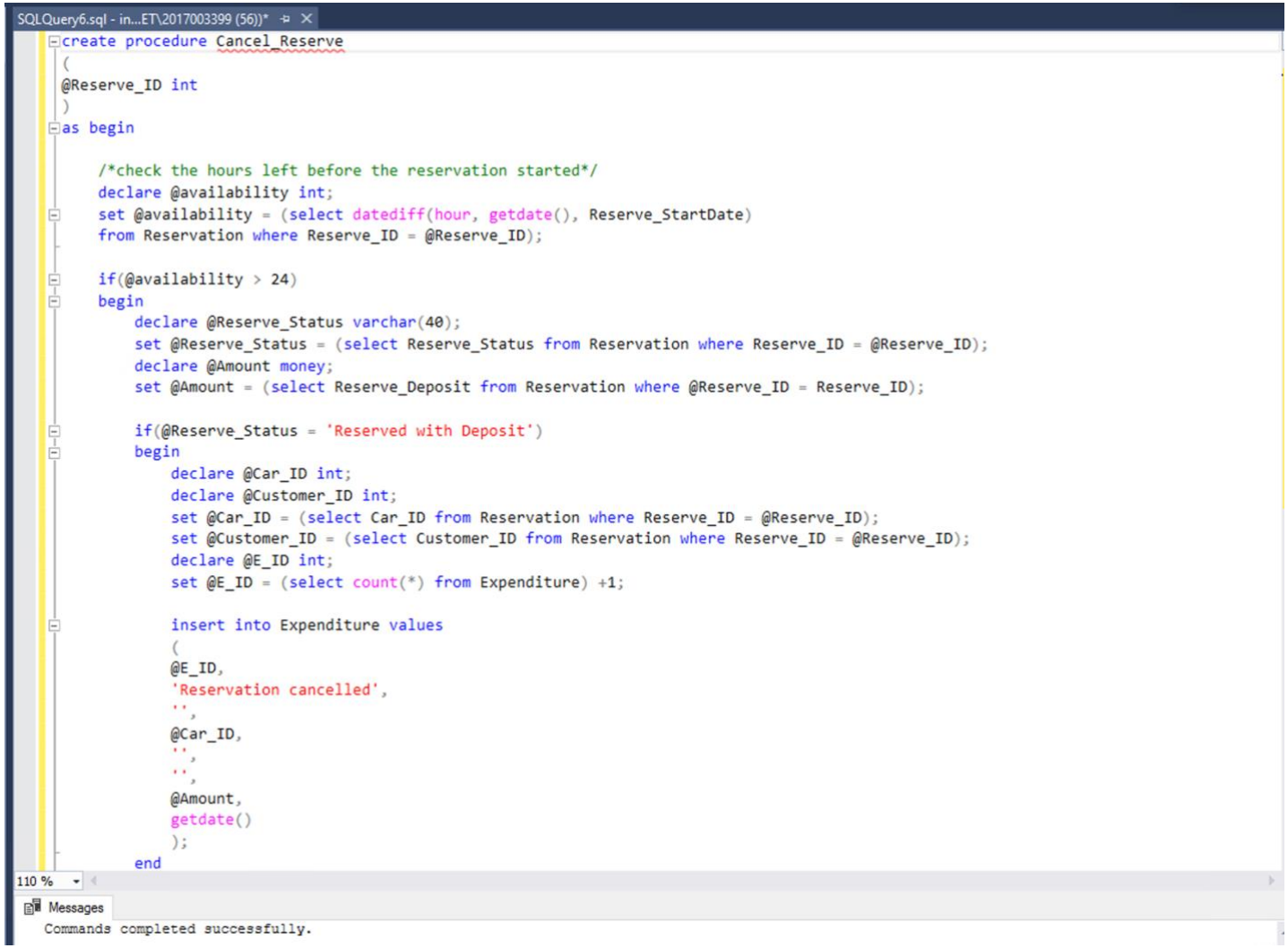
Booking system 6 – Booking procedure Part 3

This procedure will interact with Reservation table with the customer input. It will create a new reservation record based on ‘customerID’ which can help to find out the customer category in ‘Customer_Info’ table. If the customer is privileged, then will check the reservation start date if over 30 days from the current date which CAR does not allow privileged user to book a car before 30 days. Or if the customer is not privileged, then will sign the reservation status as ‘waiting deposit’, then will recommend to jump to deposit procedure.

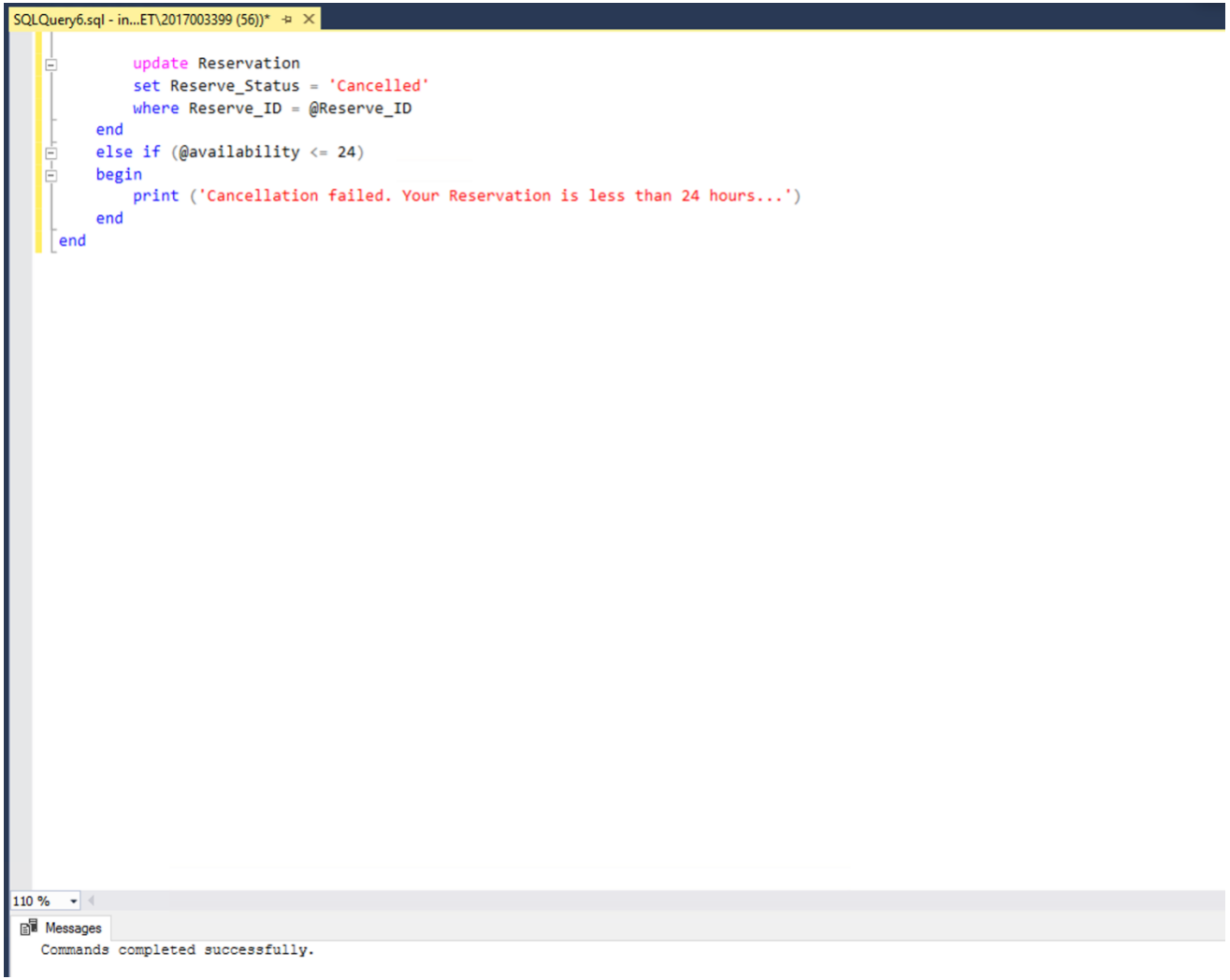


Booking system 7 – Deposit procedure

Once the CAR received the customer option which can be ‘Place deposit’ with the amount of deposit or ‘Provide credit card number’ with the credit card number, then the procedure will fill in the information based on the options. And finally, change the reservation status to either ‘Reserved with Deposit’ or ‘Reserved with Credit Card’. If customer pays the deposit on the reservation, procedure will automatically generate a revenue record in Revenue table.

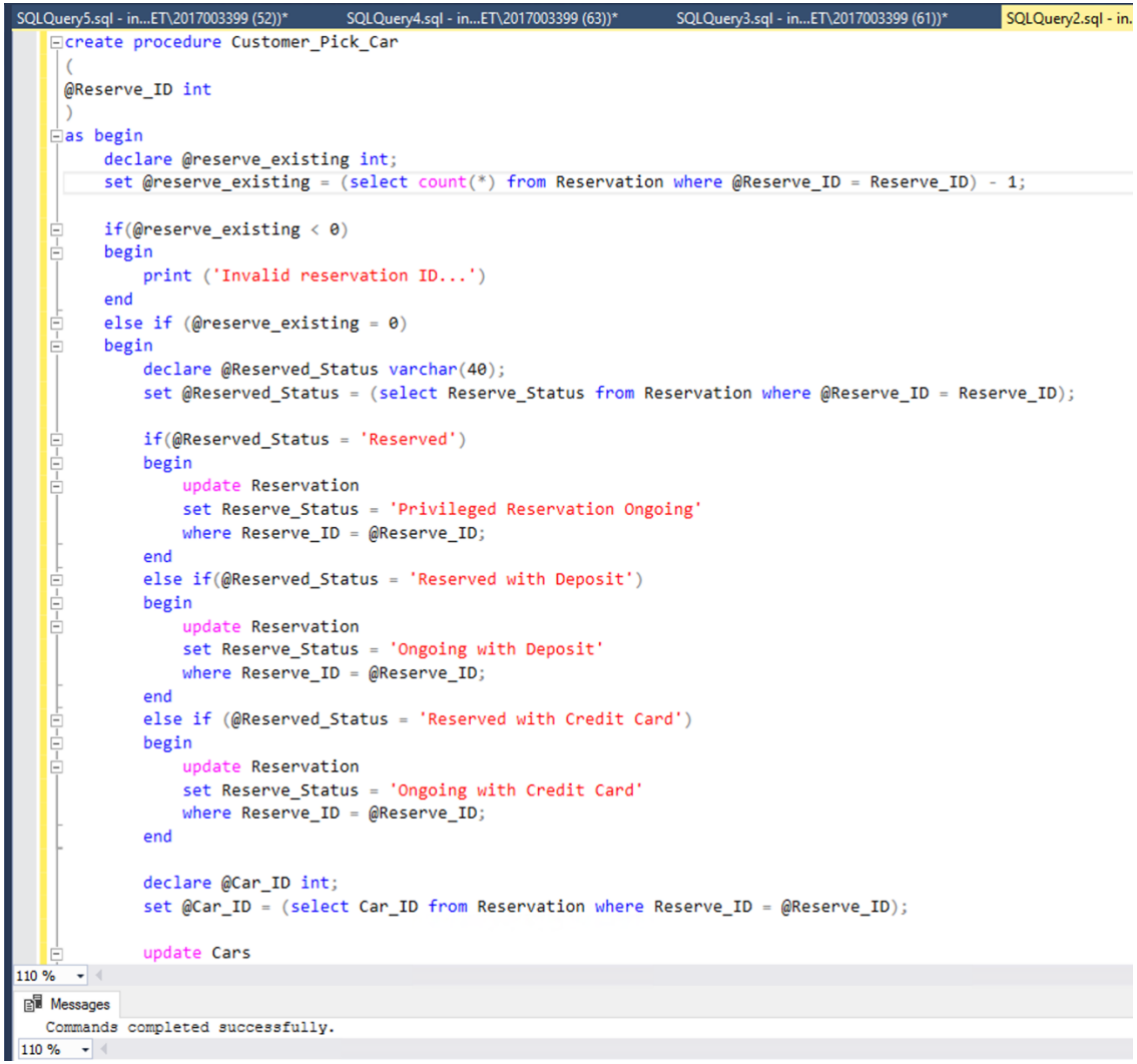


Reservation Cancellation Part 1

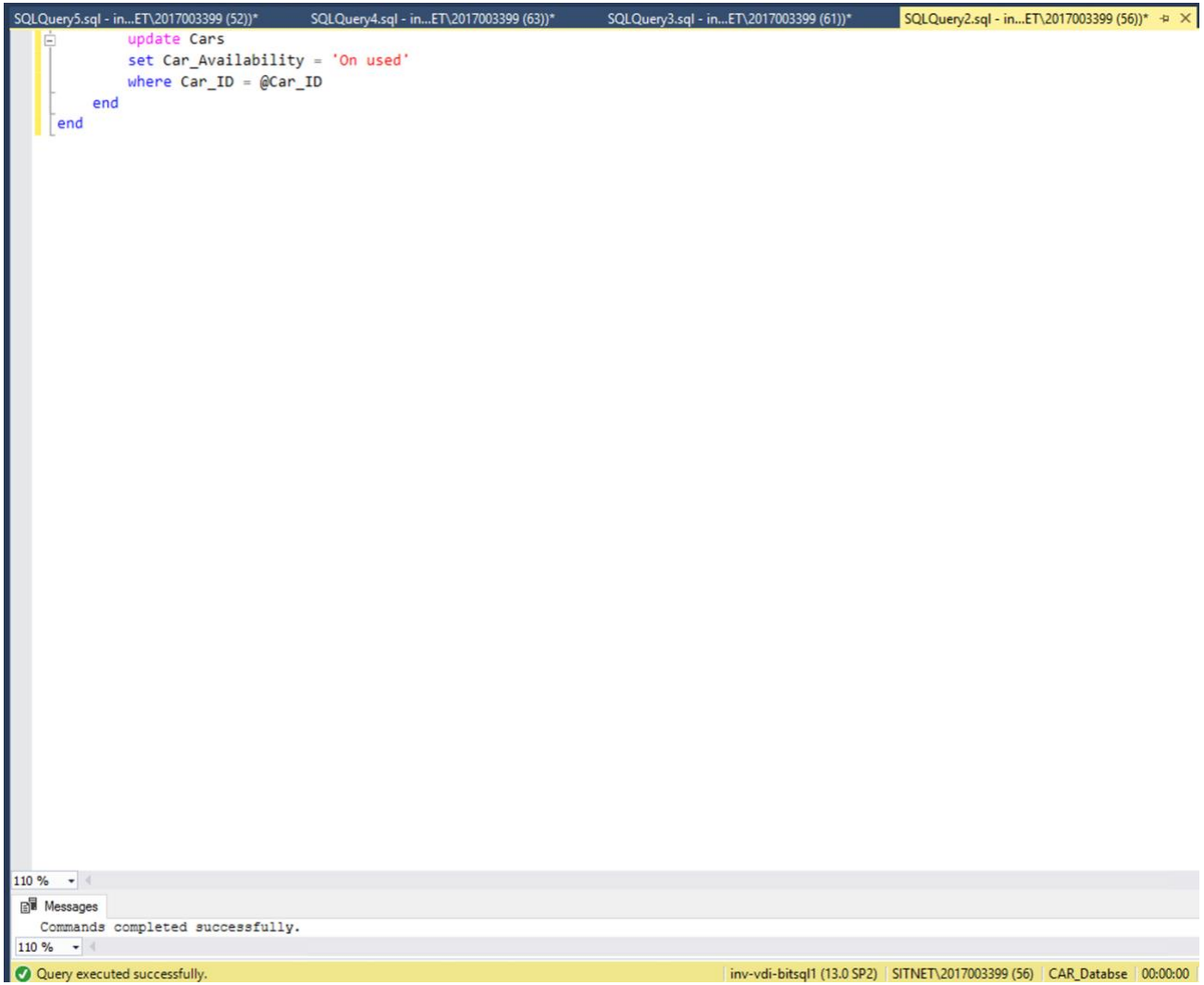


Reservation Cancellation Part 2

This procedure will allow customers to cancel their reservation only before 24 hours. If it can be cancelled and also reserved with deposit, it will generate an new Expenditure record with the Expenditure type ‘Reservation cancelled’.

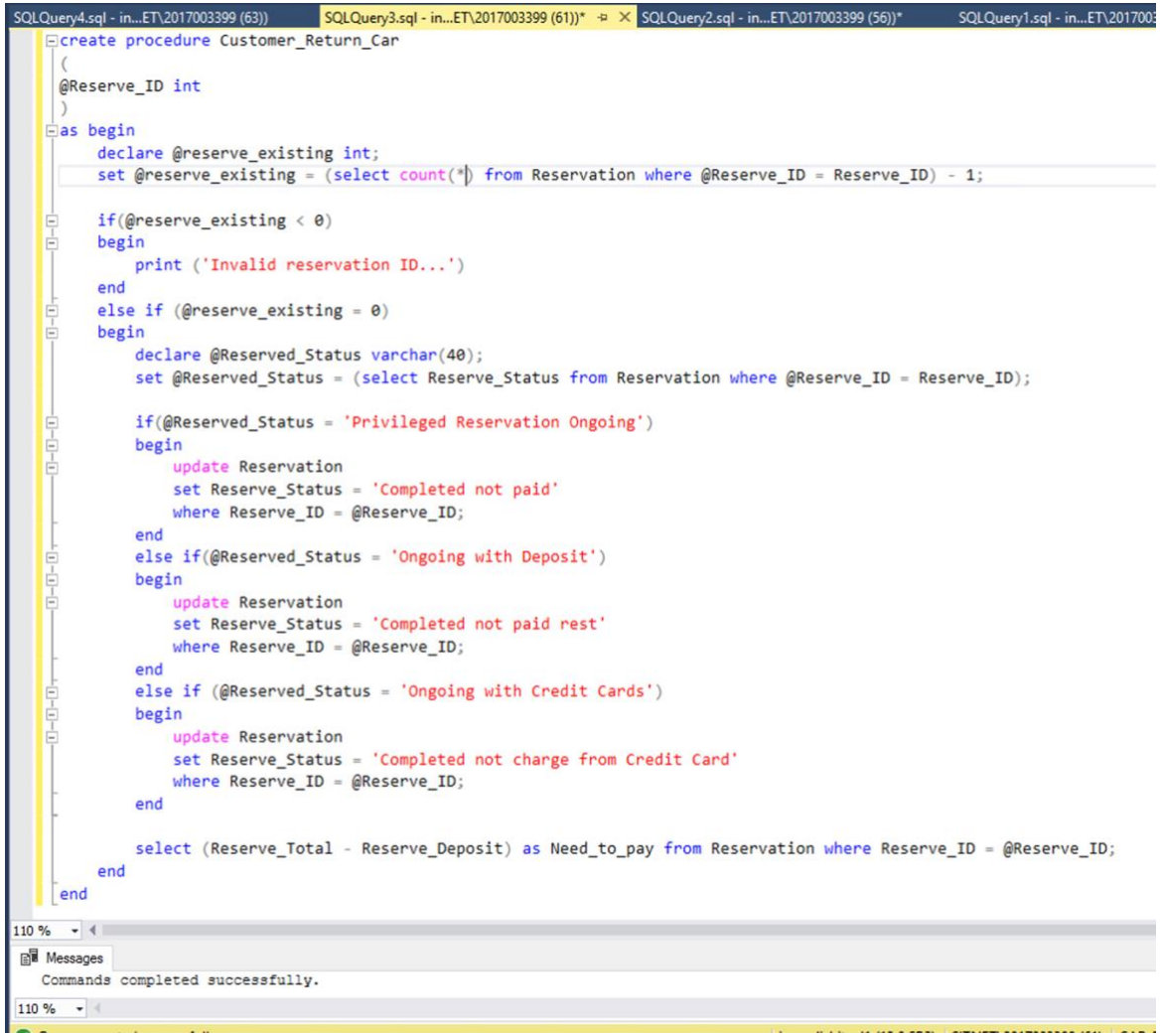


A procedure for customers when they come to pick the car with the reservation ID – Part 1



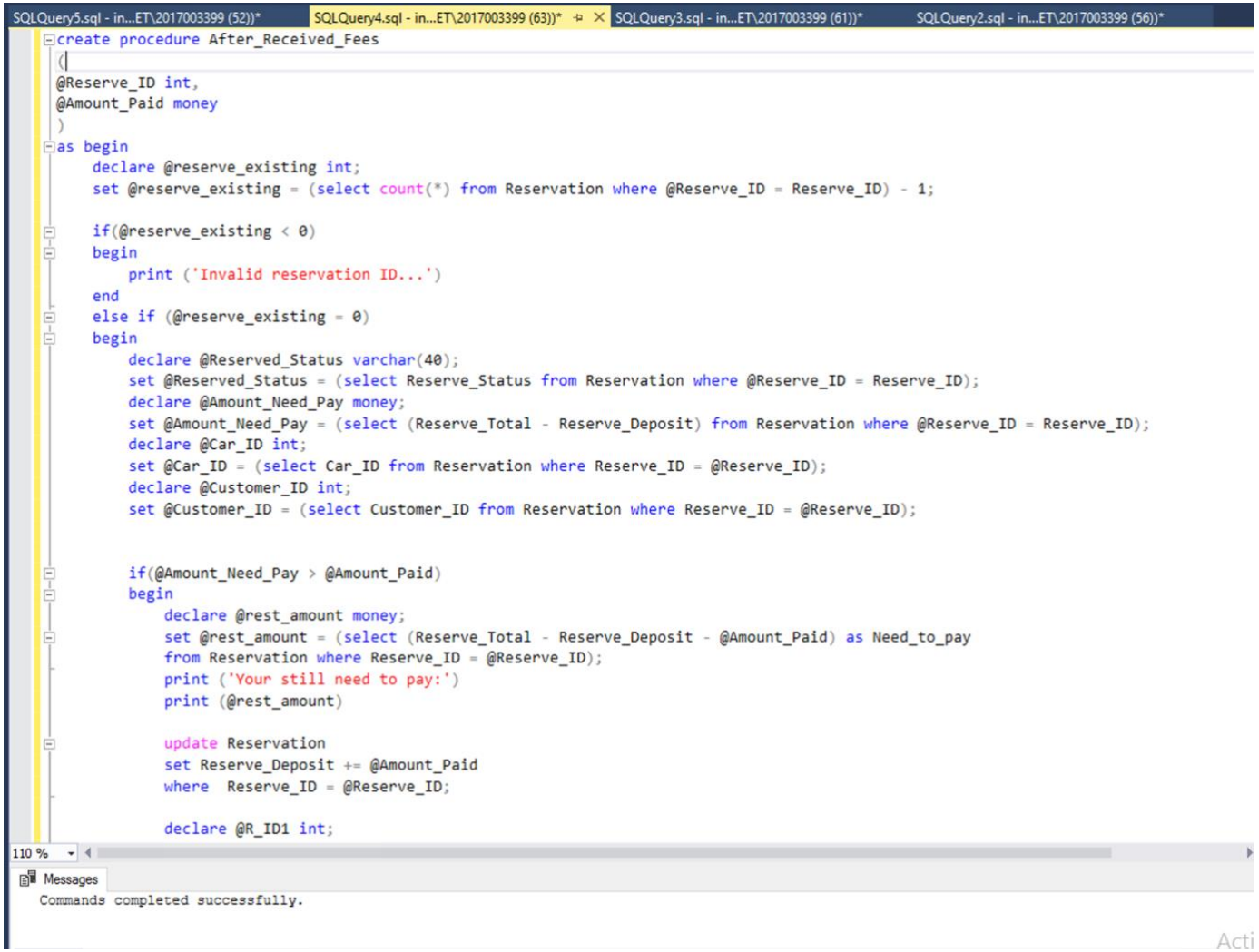
A procedure for customers when they come to pick the car with the reservation ID – Part 2

Just need to insert Reservation ID, then the procedure will automatically validate the reservation ID, if valid, then will update all relative information regards to Car's status and Reservation status.



Customer Return a car

When customer return a car , this procedure will update the reservation status to ‘completed not paid’, ‘completed not paid rest’ or ‘completed not charge form credit card’ based on reservation ID, and the reservation type.



CAR Received Fees Part 1

SQLQuery5.sql - in...ET\2017003399 (52))* | SQLQuery4.sql - in...ET\2017003399 (63))* | SQLQuery3.sql - in...ET\2017003399 (61))* | SQLQuery2.sql - in...ET\2017003399 (56))*

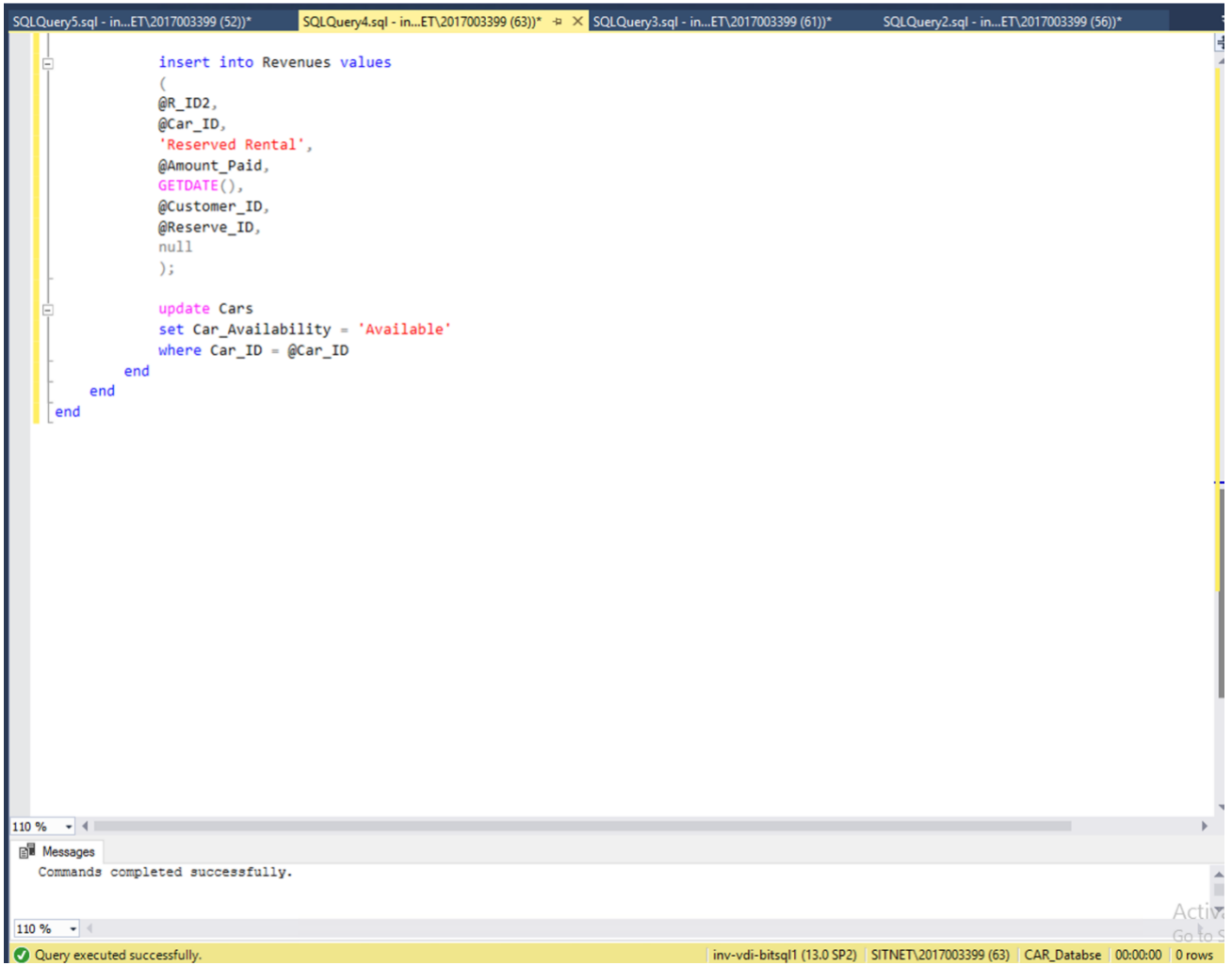
```
            set @R_ID1 = (select count(*) from Revenues) + 1;

            insert into Revenues values
            (
            @R_ID1,
            @Car_ID,
            'Reserved Rental',
            @Amount_Paid,
            GETDATE(),
            @Customer_ID,
            @Reserve_ID,
            null
            );
        end
        else if (@Amount_Need_Pay = @Amount_Paid)
        begin
            if(@Reserved_Status = 'Completed not paid')
            begin
                update Reservation
                set Reserve_Status = 'Completed paid'
                where Reserve_ID = @Reserve_ID;
            end
            else if(@Reserved_Status = 'Completed not paid rest')
            begin
                update Reservation
                set Reserve_Status = 'Completed paid'
                where Reserve_ID = @Reserve_ID;
            end
            else if (@Reserved_Status = 'Completed not charge from Credit Cards')
            begin
                update Reservation
                set Reserve_Status = 'Completed paid'
                where Reserve_ID = @Reserve_ID;
            end

            set @Customer_ID = (select Customer_ID from Reservation where Reserve_ID = @Reserve_ID);
            declare @R_ID2 int;
            set @R_ID2 = (select count(*) from Revenues) + 1;
```

110 %

Messages

```
Commands completed successfully.
```

CAR Received Fees Part 2



CAR Received Fees Part 3

Once the CAR received the amount of rental fees, procedure will check if the reservation ID is invalid, then check whether the customer pay enough fees, and update the reservation status to completed, and create a new revenue record in Revenue table.