Company Overview

The company selected is called 'Everyone Everywhere Rentals' (imaginary company for the sake of this project); it is a chain company with multiple in-person locations that provide car rental services to people across the country. The line of operation would be vehicle rental. Everyone Everywhere Rentals is not a real company and has been created for the purpose of this project.

In this system, data will be collected on the person who rents the car, all the cars, the car types, plate numbers, the methods of payment and the payment details of the customer, rewards program, reservation histories, cars that are available for reservation, and customer history. The company will be able to see previous, as well as current, customer approvals or denials based on their credit score. The most recent credit score will be used to determine the customer status, but the customer's social security number will not be stored in the database by the company.

Requirements

Customers will be able to:

a. Reserve specific cars;

b. See past car rental details;

c. See upcoming rental details;

d. Check out the status and conditions of variety of cars;

e. Sign up for rewards;

f. See the locations of the rental company.

The company will be able to access all the information above and it will also be able to:

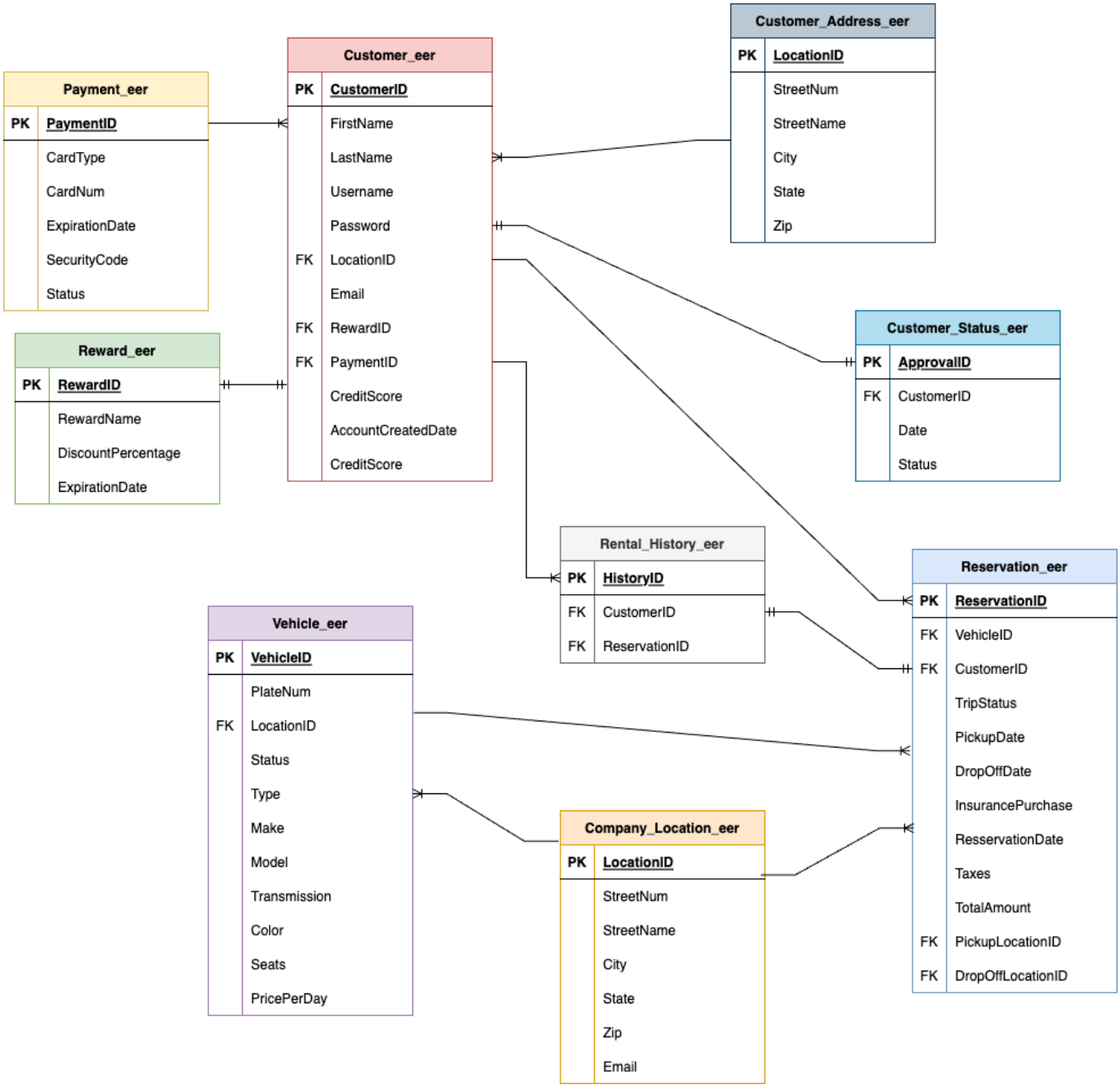
a. Store personal data on the customers;

b. Store payment information;

c. See the history of each customer;

d. Keep a list of customers who passed and did not pass the verification process.

E-R


Customer_Address_eer
PK LocationID
StreetNum
StreetName
City
State
Zip
Payment_eer
PK PaymentID
CardType
CardNum
ExpirationDate
SecurityCode
Status
Customer_eer
PK CustomerID
FirstName
LastName
Username
Password
FK LocationID
Email
FK RewardID
FK PaymentID
CreditScore
AccountCreatedDate
CreditScore
Customer_Status_eer
PK ApprovalID
FK CustomerID
Date
Status
Reward_eer
PK RewardID
RewardName
DiscountPercentage
ExpirationDate
Rental_History_eer
PK HistoryID
FK CustomerID
FK ReservationID
Reservation_eer
PK ReservationID
FK VehicleID
FK CustomerID
TripStatus
PickupDate
DropOffDate
InsurancePurchase
ResservationDate
Taxes
TotalAmount
FK PickupLocationID
FK DropOffLocationID
Vehicle_eer
PK VehicleID
PlateNum
FK LocationID
Status
Type
Make
Model
Transmission
Color
Seats
PricePerDay
Company_Location_eer
PK LocationID
StreetNum
StreetName
City
State
Zip
Email

*Note: all tables are suffixed with _eer to clearly distinguish them from other tables that might carry a similar name in the database. See attached file called 'lauraBuibas_ERDiagram_drawio.png' for the E-R Diagram image.

a. Database design - see attached file named 'lauraBuibas_BCS360_DatabaseDesign.pdf'

b. Data- see attached Excel data file named 'lauraBuibas_BCS360_Data.xlsx'

Implementation sql logic

Note: see attached file named 'lauraBuibas_BCS360_SQL_CODE_FINAL.sql' for all SQL code.

Table Creation

Fig. 1-1 Creating the Customer_eer and Payment_eer tables

```
CREATE TABLE Customer_eer
    (
    CustomerID INT NOT NULL PRIMARY KEY IDENTITY(1,1),
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Username VARCHAR(20) NOT NULL,
    Password VARCHAR(20) NOT NULL,
    LocationID INT NOT NULL,
    Email VARCHAR(100) NOT NULL,
    RewardID INT NOT NULL,
    PaymentID INT NOT NULL,
    AccountCreatedDate DATE NOT NULL,
    CreditScore INT NOT NULL
    );


CREATE TABLE Payment_eer
    (
    PaymentID INT NOT NULL PRIMARY KEY IDENTITY(1,1),
    CardType VARCHAR(4) NOT NULL,
    CardNum VARCHAR(16) NOT NULL,
    ExpirationDate VARCHAR(5) NOT NULL,
    SecurityCode VARCHAR(4) NOT NULL,
    Status CHAR(1) NOT NULL
    );
```

Fig. 1-2 Creating the reward_eer, customer_address_eer, customer_status_eer, and company_location_eer tables

```
CREATE TABLE Reward_eer
    (
    RewardID INT NOT NULL PRIMARY KEY IDENTITY(1,1),
    RewardName VARCHAR(10) NOT NULL,
    DiscountPercentage INT NOT NULL,
    ExpirationDate VARCHAR(5) NOT NULL
    );

CREATE TABLE Customer_Address_eer
    (
    LocationID INT NOT NULL PRIMARY KEY IDENTITY(1,1),
    StreetNum INT NOT NULL,
    StreetName VARCHAR(50) NOT NULL,
    City VARCHAR(50) NOT NULL,
    State CHAR(2) NOT NULL,
    Zip INT NOT NULL
    );

CREATE TABLE Customer_Status_eer
    (
    ApprovalID INT NOT NULL PRIMARY KEY IDENTITY(1,1),
    CustomerID INT NOT NULL REFERENCES Customer_eer(CustomerID),
    Date DATE NOT NULL,
    Status CHAR(1) NOT NULL
    );

CREATE TABLE Company_Location_eer
    (
    LocationID INT NOT NULL PRIMARY KEY IDENTITY(1,1),
    StreetNum INT NOT NULL,
    StreetName VARCHAR(50) NOT NULL,
    City VARCHAR(50) NOT NULL,
    State CHAR(2) NOT NULL,
    Zip INT NOT NULL,
    Email VARCHAR(100) NOT NULL,
    );
```

Fig. 1-3 creating the vehicle_eer, reservation_eer, and rental_history_eer tables

```
CREATE TABLE Vehicle_eer
    (
    VehicleID INT NOT NULL PRIMARY KEY IDENTITY(1,1),
    PlateNum VARCHAR(8) NOT NULL,
    LocationID INT NOT NULL REFERENCES Company_Location_eer(LocationID),
    Status CHAR(1) NOT NULL,
    Type CHAR(1) NOT NULL,
    Make VARCHAR(20) NOT NULL,
    Model VARCHAR(20) NOT NULL,
    Color VARCHAR(20) NOT NULL,
    Seats INT NOT NULL,
    PricePerDay MONEY NOT NULL
    );

CREATE TABLE Reservation_eer
    (
    ReservationID INT NOT NULL PRIMARY KEY IDENTITY(1,1),
    VehicleID INT NOT NULL REFERENCES Vehicle_eer(VehicleID),
    CustomerID INT NOT NULL REFERENCES Customer_eer(CustomerID),
    TripStatus CHAR(1) NOT NULL,
    PickupDate DATETIME NOT NULL,
    DropOffDate DATETIME NOT NULL,
    InsurancePurchase CHAR(1) NOT NULL,
    ReservationDate DATETIME NOT NULL,
    Taxes MONEY NOT NULL,
    TotalAmount MONEY NOT NULL,
    PickupLocationID INT NOT NULL REFERENCES Company_Location_eer(LocationID),
    DropOffLocationID INT NOT NULL REFERENCES Company_Location_eer(LocationID)
    );

CREATE TABLE Rental_History_eer
    (
    HistoryID INT NOT NULL PRIMARY KEY IDENTITY(1,1),
    CustomerID INT NOT NULL REFERENCES Customer_eer(CustomerID),
    ReservationID INT NOT NULL REFERENCES Reservation_eer(ReservationID)
    );
```

Inserting Data into Created Tables

Fig. 1-4 inserting data into the customer_eer, payment_eer, and reward_eer tables

```
INSERT INTO Customer_eer
VALUES ('Jane', 'Doe', 'jdoe', 'abcdef', 2, 'jdo@gmail.com', 1, 3, '2021-03-27', 700),
       ('John', 'Doe', 'jodoe', 'abcdef', 1, 'jodoe@gmail.com', 2, 1, '2021-02-27', 750),
       ('Amie', 'May', 'amay', 'abcdef', 4, 'amay@gmail.com', 3, 2, '2021-02-26', 800),
       ('Clair', 'Montclair', 'cmont', 'abcdef', 3, 'cmont@gmail.com', 4, 10, '2022-02-27', 650),
       ('Jamie', 'Fraser', 'jfras', 'qrstuv', 9, 'jfras@gmail.com', 6, 4, '2021-07-26', 730),
       ('Ollie', 'Frankie', 'ofrank', 'qrstuv', 10, 'ofrank@gmail.com', 10, 5, '2022-02-27', 620),
       ('Alexander', 'Hamilton', 'ahal', 'qrstuv', 8, 'ahal@gmail.com', 5, 6, '2022-07-14', 550),
       ('George', 'Wallflower', 'gwall', 'qrstuv', 7, 'gwall@gmail.com', 7, 7, '2021-02-17', 590),
       ('Annette', 'Italone', 'aital', 'abcdef', 6, 'aital@gmail.com', 8, 8, '2021-02-03', 730),
       ('Charlie', 'Mayflower', 'cmayf', 'abcdef', 5, 'cmayf@gmail.com', 9, 9, '2021-02-02', 750);


INSERT INTO Payment_eer
VALUES ('V', '1921910212139810', '04/27', 313, 'A'),
       ('AMEX', '7896910213981080', '04/26', 321, 'A'),
       ('V', '4200625829358760', '03/25', 123, 'A'),
       ('AMEX', '7891237854309070', '03/25', 234, 'A'),
       ('V', '6300211898731260', '03/25', 432, 'A'),
       ('AMEX', '7891986543890920', '07/27', 324, 'A'),
       ('V', '6300675478116530', '11/26', 567, 'A'),
       ('MC', '7813211898711280', '11/25', 654, 'A'),
       ('MC', '7812349872568890', '11/27', 467, 'A'),
       ('MC', '7812987656783420', '03/25', 876, 'A');

INSERT INTO Reward_eer
VALUES ('Platinum', 20, '05/23'),
       ('Platinum', 20, '05/23'),
       ('Platinum', 20, '04/24'),
       ('Platinum', 20, '03/24'),
       ('Silver', 15, '11/23'),
       ('Silver', 15, '11/23'),
       ('Silver', 15, '11/23'),
       ('Silver', 15, '05/23'),
       ('Bronze', 10, '05/23'),
       ('Gold', 5, '05/23');
```

Fig 1-5 inserting data into company_location_eer and vehicle_eer tables

```
INSERT INTO Company_Location_eer
VALUES (23, 'May ave', 'Melville', 'NY', 11765, 'melville@eer.com'),
       (32, 'June str', 'New Orleans', 'LA', 12340, 'neworleans@eer.com'),
       (45, 'Eight str', 'George Post', 'NY', 67219, 'georgepost@eer.com'),
       (54, 'Carle ave', 'Gardden City', 'NY', 67192, 'gardencity@eer.com');



INSERT INTO Vehicle_eer
VALUES ('HNG6576', 1, 'A', 'A', 'Chevrolet', 'Silverado', 'black', 6, 120),
       ('AZY6789', 2, 'A', 'A', 'Chevrolet', 'Colorado', 'blue', 6, 120),
       ('ZZY1234', 3, 'A', 'A', 'Chevrolet', 'Silverado', 'black', 6, 120),
       ('JKL4786', 4, 'A', 'A', 'Chevrolet', 'Equinox', 'white', 5, 115),
       ('LKJ6780', 1, 'A', 'A', 'Chevrolet', 'Equinox', 'blue', 5, 115),
       ('BGH4567', 2, 'A', 'A', 'Audi', 'A4', 'grey', 5, 300),
       ('NHG2345', 3, 'A', 'A', 'Audi', 'A4', 'blue', 5, 300),
       ('MJH6754', 4, 'T', 'A', 'Audi', 'A6', 'red', 5, 250),
       ('MLK2349', 1, 'T', 'A', 'Audi', 'A6', 'grey', 5, 250),
       ('LKJ4210', 2, 'T', 'A', 'Toyota', 'Corolla', 'black', 5, 115),
       ('ZAN4532', 3, 'A', 'A', 'Toyota', 'Corolla', 'red', 5, 115),
       ('LKH3976', 4, 'A', 'A', 'Toyota', 'Camry', 'red', 5, 120),
       ('GHJ9876', 1, 'A', 'A', 'Toyota', 'Corolla', 'blue', 5, 115),
       ('FGJ7830', 1, 'A', 'A', 'Toyota', 'Highlander', 'black', 6, 130),
       ('NJF3987', 1, 'T', 'A', 'Toyota', 'Tundra', 'black', 6, 150),
       ('NIH2423', 2, 'A', 'A', 'Toyota', 'Corolla', 'black', 5, 110),
       ('HJK4390', 2, 'A', 'A', 'Honda', 'Civic', 'black', 5, 200),
       ('HFS9732', 3, 'T', 'A', 'Honda', 'Civic', 'red', 5, 130),
       ('BFJ9370', 3, 'A', 'A', 'Honda', 'Passport', 'grey', 6, 250),
       ('FJA3890', 3, 'T', 'M', 'Honda', 'Civic', 'grey', 5, 120);
```

Fig 1-6 inserting data into the reservation_eer and rental _history_eer tables

```
INSERT INTO Reservation_eer
VALUES (1, 1, 'C', '2023-04-01', '2023-04-14', 'Y', '2023-03-23', 100, 1835, 1, 1),
       (2, 2, 'C', '2023-04-01', '2023-04-14', 'N', '2023-02-23', 150, 3510, 1, 1),
       (3, 3, 'C', '2023-04-03', '2034-04-20', 'N', '2023-04-01', 20, 2180, 1, 1),
       (4, 4, 'C', '2023-04-01', '2023-04-15', 'N', '2023-03-02', 50, 1775, 1, 2),
       (5, 5, 'C', '2023-04-15', '2023-04-30', 'N', '2023-04-10', 60, 1900, 2, 2),
       (6, 6, 'C', '2023-04-17', '2023-04-30', 'N', '2023-04-16', 70, 4270, 2, 2),
       (7, 1, 'C', '2023-04-15', '2023-04-30', 'N', '2023-04-14', 80, 4580, 2, 2),
       (8, 1, 'P', '2023-07-23', '2023-07-30', 'N', '2023-04-14', 90, 6890, 2, 1),
       (9, 2, 'P', '2023-09-07', '2023-09-17', 'N', '2023-03-12', 100, 2850, 3, 3),
       (1, 3, 'P', '2023-10-06', '2023-10-20', 'N', '2023-05-01', 20, 1820, 3, 3),
       (20, 4, 'P', '2023-07-06', '2023-07-15', 'N', '2023-05-01', 20, 1250, 3, 2),
       (15, 10, 'P', '2023-09-07', '2023-09-17', 'N', '2023-05-01', 50, 1700, 4, 2),
       (18, 9, 'P', '2023-09-07', '2023-09-17', 'N', '2023-03-12', 50, 1480, 4, 2),
       (17, 8, 'C', '2023-02-25', '2023-02-28', 'Y', '2022-12-30', 50, 900, 4, 1),
       (15, 7, 'C', '2023-03-14', '2023-03-24', 'N', '2022-11-15', 50, 1700, 1, 1),
       (19, 6, 'C', '2023-03-25', '2023-03-30', 'N', '2023-01-05', 50, 1550, 1, 1);


INSERT INTO Rental_History_eer
VALUES (1, 1),
       (2, 2),
       (3, 3),
       (4, 4),
       (5, 5),
       (6, 6),
       (1, 7),
       (1, 8),
       (2, 9),
       (3, 10),
       (4, 11),
       (10, 12),
       (9, 13),
       (8, 14),
       (7, 15),
       (6, 16);
```

Database Management and Manipulation Code

Note: it is strongly recommended to read the next section called 'Use Cases' first to understand the purpose of each use case described in this subsection.

Fig. 1-7 the code for the creation of use case 1: customer creates an account

Note: picture extends to the next page

```
CREATE PROC spCustomerAccountCreation
        (@FirstName VARCHAR(60)
        , @LastName VARCHAR(60)
        , @Username VARCHAR(20)
        , @Password VARCHAR(20)
        , @Email VARCHAR(100)
        , @CreditScore INT
        , @StreetNum INT
        , @StreetName VARCHAR(50)
        , @City VARCHAR(50)
        , @State CHAR(2)
        , @Zip INT
        , @CardType VARCHAR(4)
        , @CardNum VARCHAR(16)
        , @ExpirationDate VARCHAR(5)
        , @SecurityCode VARCHAR(4)
        )
AS
BEGIN TRY
BEGIN
DECLARE @LocationID INT
DECLARE @PaymentID INT
DECLARE @RewardID INT
        IF NOT EXISTS (SELECT 1 FROM Customer_eer WHERE Email = @Email)
            BEGIN
            -- if customer does not exist, MUST ALSO CHECK IF USERNAME IS UNIQUE
            IF NOT EXISTS (SELECT 1 FROM Customer_eer WHERE Username = @Username)
                BEGIN
                -- if location info does not exist, create it and get the locationID
                IF NOT EXISTS (SELECT 1
                                FROM Customer_Address_eer
                                WHERE StreetNum = @StreetNum AND StreetName = @StreetName AND City = @City AND State = @State AND Zip = @Zip)
                    BEGIN
                    INSERT INTO Customer_Address_eer
                    VALUES (@StreetNum, @StreetName, @City, @State, @Zip)
                    SET @LocationID = SCOPE_IDENTITY()
                    PRINT 'New location record inserted successfully'
                    END
                -- else, get the locationID of the existing location
                ELSE
                    BEGIN
                    SET @LocationID = (SELECT LocationID
```

```
                                    FROM Customer_Address_eer
                                    WHERE StreetNum = @StreetNum AND StreetName = @StreetName AND City = @City AND State = @State AND Zip = @Zip)
                    END

                -- if payment info does not exist, create it and get the paymentID
                IF NOT EXISTS (SELECT 1 FROM Payment_eer WHERE CardNum = @CardNum)
                    BEGIN
                    INSERT INTO Payment_eer
                    VALUES (@CardType, @CardNum, @ExpirationDate, @SecurityCode, 'A')
                    SET @PaymentID = SCOPE_IDENTITY()
                    PRINT 'New payment record inserted successfully';
                    END
                -- else, get the paymentid of the existing payment
                ELSE
                    BEGIN
                    SET @PaymentID = (SELECT PaymentID
                                    FROM Payment_eer
                                    WHERE CardNum = @CardNum)
                    END

                -- create customer with locationID, paymentID, and generate new reward information
                    -- all new customers get a new reward level named 'ruby' with 0% discount, and an expiration value of 'n/a'
                INSERT INTO Reward_eer
                VALUES ('Ruby', 0, 'n/a');
                SET @RewardID = SCOPE_IDENTITY()

                DECLARE @customerID INT
                INSERT INTO Customer_eer
                VALUES (@FirstName, @LastName, @Username, @Password, @LocationID, @Email, @RewardID, @PaymentID, GETDATE(), @CreditScore)
                SET @customerID = SCOPE_IDENTITY()
                PRINT 'Customer created with ID' + CONVERT(VARCHAR(5), @customerID)
                --SELECT * FROM Customer_eer WHERE CustomerID = @customerID
                END
        ELSE
            PRINT 'Username already exists. Pick a different username'
            END
    -- else, display customer already exists
    ELSE
        PRINT 'Customer already exists. try again with different email addresss'
END

END TRY
```

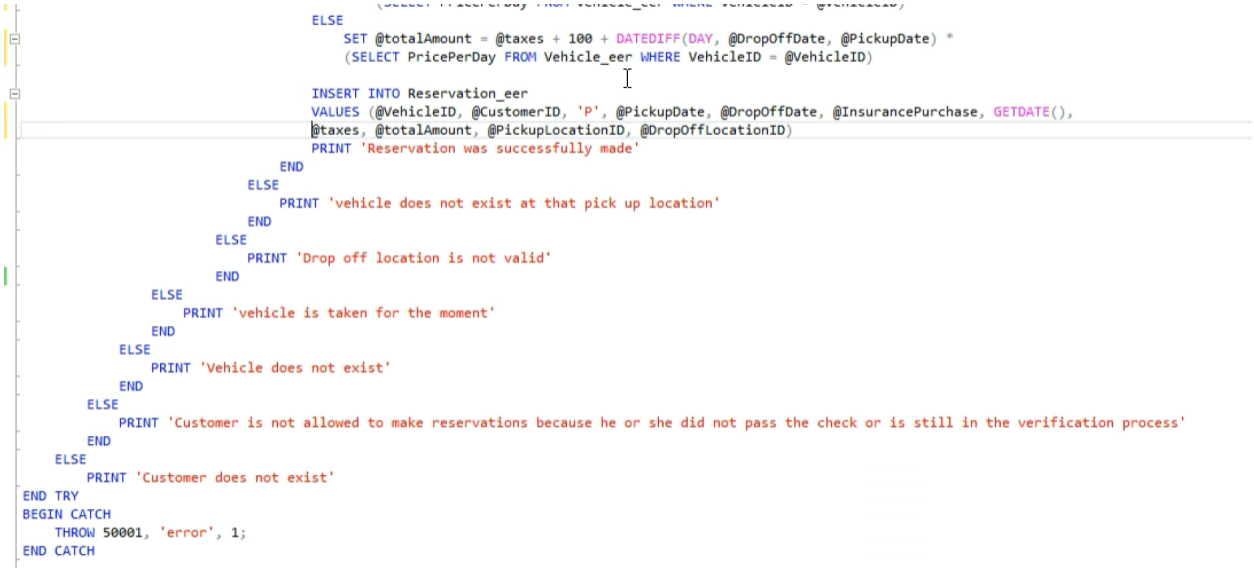
```
BEGIN CATCH
    throw 50001, 'error', 1;
END CATCH
```

Fig 1-8 the code for the creation of use case 2: upcoming rental management

```
/
-- customers use this procedure to cancel reservations
CREATE PROC spCancelReservation
    (@ReservationID INT)
AS
BEGIN TRY
    -- if reservation exists
    IF EXISTS (SELECT 1 FROM Reservation_eer WHERE ReservationID = @ReservationID)
        BEGIN
        -- if reservation is pending
        IF (SELECT TripStatus FROM Reservation_eer WHERE ReservationID = @ReservationID) = 'P'
            BEGIN
            UPDATE Reservation_eer
            SET TripStatus = 'S'
            WHERE ReservationID = @ReservationID
            PRINT 'Reservation successfully canceled, now has an S value, meaning it was stopped.';
            END
        ELSE
            PRINT 'Cannot cancel a reservation that was already completed or canceled'
        END
    ELSE
        PRINT 'Reservation does not exist'
END TRY
BEGIN CATCH
    THROW 50001, 'error', 1;
END CATCH
```

Fig 1-9 the code for the creation of use case 3: vehicle is rented (note: picture extends to the next page)

```
*/
CREATE PROC spCreateRental
       (@CustomerID INT
       , @VehicleID INT
       , @PickupDate DATETIME
       , @DropOffDate DATETIME
       , @InsurancePurchase CHAR(1)
       , @PickupLocationID INT
       , @DropOffLocationID INT
       )
AS
BEGIN TRY
    -- if customer exists
    IF EXISTS (SELECT 1 FROM Customer_eer WHERE CustomerID = @CustomerID)
        BEGIN
        -- if customer has a passing value in the customer status table
        IF ((SELECT MAX(Status) FROM Customer_Status_eer WHERE CustomerID = @CustomerID) = 'P')
            BEGIN
            --if vehicle exists
            IF EXISTS (SELECT 1 FROM Vehicle_eer WHERE VehicleID = @VehicleID)
                BEGIN
                -- if vehicle is not part of a pending reservation
                IF NOT EXISTS (SELECT 1 FROM Reservation_eer WHERE TripStatus = 'P' AND VehicleID = @VehicleID)
                    BEGIN
                    -- if drop off location ID is valid
                        IF EXISTS (SELECT 1 FROM Company_Location_eer WHERE LocationID = @DropOffLocationID)
                            BEGIN
                            -- if vehicle exists at the given pick up location
                            IF EXISTS (SELECT 1 FROM Vehicle_eer WHERE LocationID = @PickupLocationID AND VehicleID = @VehicleID)
                                BEGIN
                                    -- Do taxes
                                    DECLARE @taxes MONEY
                                    SET @taxes = (SELECT PricePerDay FROM Vehicle_eer WHERE VehicleID = @VehicleID) * .15 *
                                        DATEDIFF(DAY, @DropOffDate, @PickupDate)
                                    -- determine price if $100 insurance was purchased
                                    DECLARE @totalAmount MONEY
                                    IF @InsurancePurchase = 'N'
                                        SET @totalAmount = @taxes + DATEDIFF(DAY, @DropOffDate, @PickupDate) *
                                            (SELECT PricePerDay FROM Vehicle_eer WHERE VehicleID = @VehicleID)
```



```
                                ELSE
                                    SET @totalAmount = @taxes + 100 + DATEDIFF(DAY, @DropOffDate, @PickupDate) *
                                    (SELECT PricePerDay FROM Vehicle_eer WHERE VehicleID = @VehicleID)

                                INSERT INTO Reservation_eer
                                VALUES (@VehicleID, @CustomerID, 'P', @PickupDate, @DropOffDate, @InsurancePurchase, GETDATE(),
                                @taxes, @totalAmount, @PickupLocationID, @DropOffLocationID)
                                PRINT 'Reservation was successfully made'
                            END
                        ELSE
                            PRINT 'vehicle does not exist at that pick up location'
                        END
                    ELSE
                        PRINT 'Drop off location is not valid'
                    END
                ELSE
                    PRINT 'vehicle is taken for the moment'
                END
            ELSE
                PRINT 'Vehicle does not exist'
            END
        ELSE
            PRINT 'Customer is not allowed to make reservations because he or she did not pass the check or is still in the verification process'
        END
    ELSE
        PRINT 'Customer does not exist'
END TRY
BEGIN CATCH
    THROW 50001, 'error', 1;
END CATCH
```

Fig 1-10 the code for the creation of use case 4: available vehicle summary

```
*/
CREATE PROC spSeeAvailableVehicles
AS
BEGIN TRY
    DECLARE @tableAvailableVehicles TABLE
        (VehicleID INT
        , Type CHAR(1)
        , Make VARCHAR(20)
        , Model VARCHAR(20)
        , Color VARCHAR(20)
        , Seats INT
        , PricePerDay MONEY
        , LocationID INT
        , StreetNum INT
        , StreetName VARCHAR(50)
        , City VARCHAR(50)
        , State CHAR(2)
        , Zip INT
        )
    INSERT @tableAvailableVehicles
    SELECT VehicleID, Type, Make, Model, Color, Seats, PricePerDay, v.LocationID, StreetNum, StreetName, City, State, Zip
    FROM Vehicle_eer v
    JOIN Company_Location_eer cl
        ON v.LocationID = cl.LocationID
    WHERE Status = 'A';

    SELECT * FROM @tableAvailableVehicles;

END TRY
BEGIN CATCH
    throw 50001, 'error', 1;
END CATCH
```

Fig 1- 11 the code for the creation of use case 5: vehicle availability status

```
-- this trigger goes hand in hand with use case 2. when the reservation is stopped, the vehicle availability
-- will change to 'A' from 'T'
-- drop trigger tUpdateReservation
CREATE TRIGGER tUpdateReservation
ON Reservation_eer
AFTER UPDATE
AS
BEGIN TRY
    IF EXISTS (SELECT 1 FROM Reservation_eer WHERE ReservationID = (SELECT ReservationID FROM Inserted))
    BEGIN
        UPDATE Vehicle_eer
        SET Status = 'A'
        WHERE VehicleID IN (SELECT VehicleID from Inserted)
        PRINT 'vehicle status updated'
    END
END TRY
BEGIN CATCH
    THROW 50001, 'error', 1;
END CATCH
```

Fig 1-12 the code for the creation of use case 6: vehicle taken status

```
*/
CREATE TRIGGER tInsertReservation
ON Reservation_eer
AFTER INSERT
AS
BEGIN TRY
    IF EXISTS (SELECT 1 FROM Reservation_eer WHERE ReservationID = (SELECT ReservationID FROM Inserted))
    BEGIN
        UPDATE Vehicle_eer
        SET Status = 'T'
        WHERE VehicleID IN (SELECT VehicleID from Inserted)
    END
END TRY
BEGIN CATCH
    THROW 50001, 'error', 1;
END CATCH
```

Fig 1-13 the code for the creation of use case 7: see upcoming rentals

```
*/
-- customers use this function to see their upcoming rentals
CREATE FUNCTION fnUpcomingReservations (@CustomerID INT)
    RETURNS table
RETURN
    SELECT *
    FROM Reservation_eer
    WHERE CustomerID = @CustomerID AND TripStatus = 'P'
;

-- stored procedure to run function above
CREATE PROC spSeeUpcomingRentals
    (@CustomerID INT)
AS
BEGIN TRY
    SELECT * FROM dbo.fnUpcomingReservations(@CustomerID)
END TRY
BEGIN CATCH
    THROW 50001, 'error', 1;
END CATCH
```

Use Cases

Use Case 1: Customer Creates an Account

Customers are able to create a customer account using their information. If the fields required are taken by another customer, the customer will be prompted to try again with different information. In order to create an account, the user must not be an existing customer, must have a unique username, must provide a location, and must provide a payment.

Use Case 2: Upcoming Rental Management

Customers will be able to see their upcoming rentals and cancel these rentals if their plans change. Could also be called ‘Upcoming Rental Cancellation’.

The customer will provide an existing reservation whose status is still pending and the reservation will be canceled.

Use Case 3: Vehicle is Rented

Customers are able to rent and reserve cars for a time period in which the car desired is available

An existing customer must have a passing value after the credit score check, must wish to rent an existing vehicle that is not currently pending for another reservation and that is available at the specified location. After this, proper money management will be calculated and the reservation will be successfully completed.

Use Case 4: Available Vehicles Summary

Users are able to view cars that are currently available to rent

Use Case 5: Vehicle Availability Status

When a customer cancels his or her reservation, the vehicle will become available to rent once again. Everything so far has been a stored procedure, but this will be a trigger that listens for when a reservation is canceled and when it is, the vehicle's status of the canceled reservation will become available ‘A’ once again. This use case goes hand in hand with use case 2.

Use Case 6: Vehicle Taken Status

When a reservation is inserted, the availability of that reserved vehicle is changed to taken so that the vehicle cannot be rented by another customer.

This is another trigger that listens for when a new reservation row is inserted, and when it is, the vehicle associated with that particular reservation will have a status that becomes taken ‘T’ so that no other reservations can be made on the upcoming vehicle because it will not show in the table that shows the available vehicles (use case 4). This use case goes hand in hand with use case 3.

Use Case 7: See Upcoming Rentals

Customers are able to see a list of their upcoming reservation details by calling a function within a stored procedure.

Test implementation

Note: the testing was not done in the order in which the use cases were listed. Instead, testing was done in the order such that use cases can flow into each other.

The testing of use case 1: customer creates an account

Fig. 2-1 - Result set if the customer wants to register with an existing email, meaning that the customer already exists.

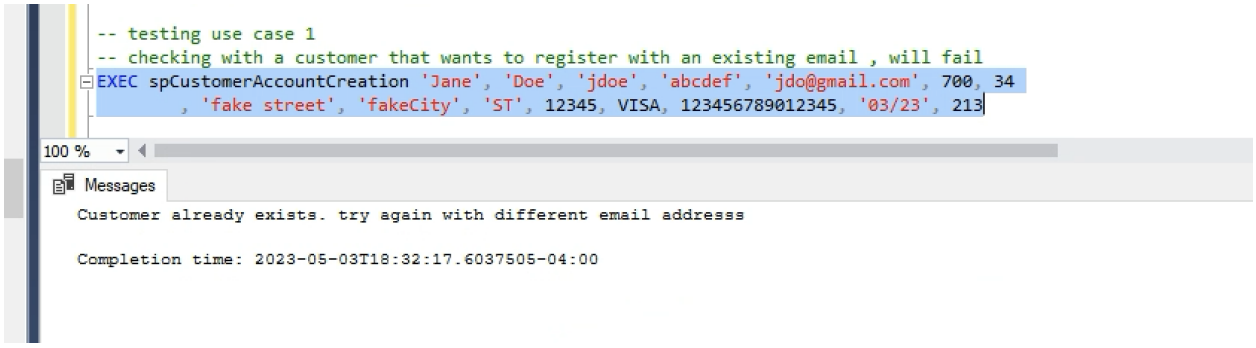


Fig. 2-2 - Result set if the customer wants to register with a new email, but the username is taken by another user

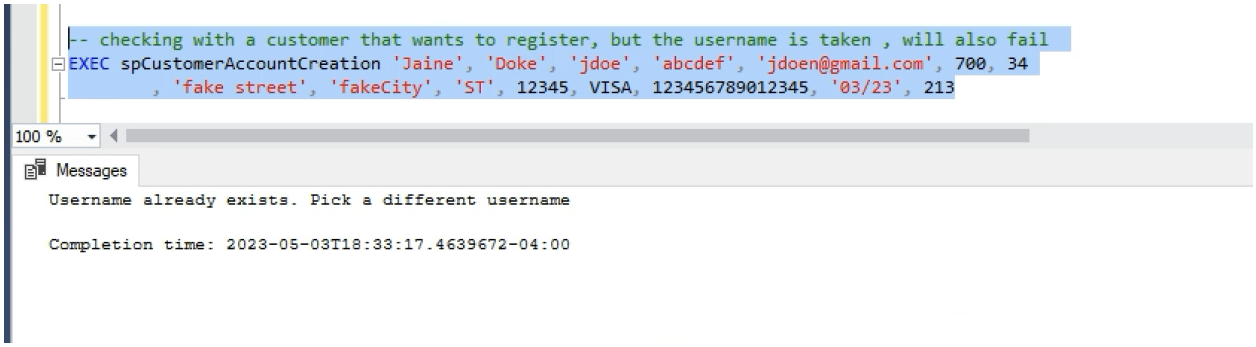
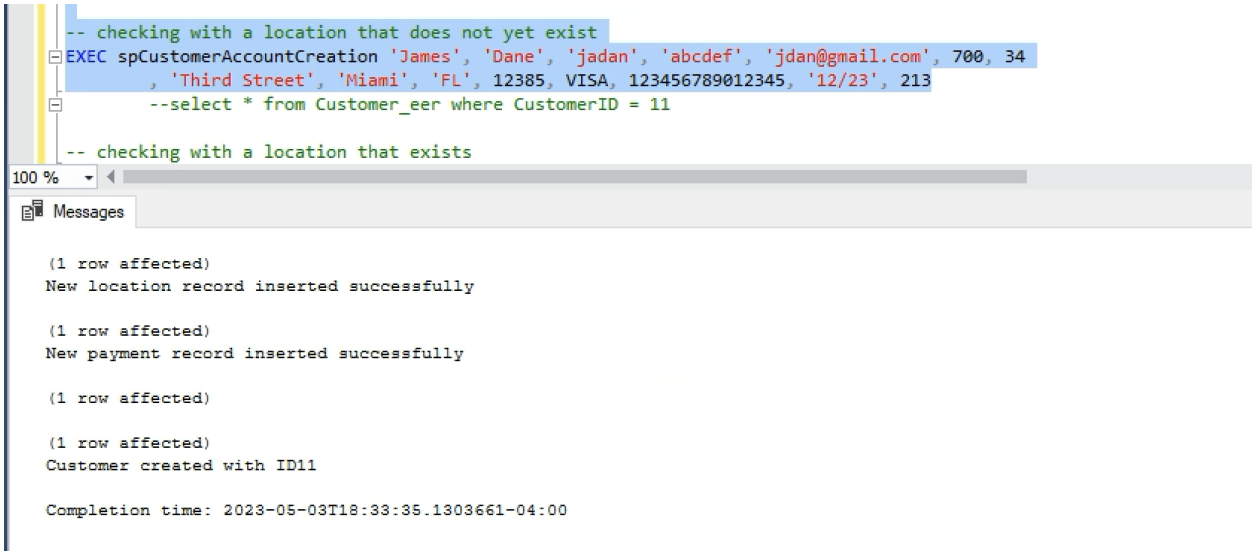


Fig. 2-3 - Result set when the customer wants to register with new email, username, location, and payment information , registration successful



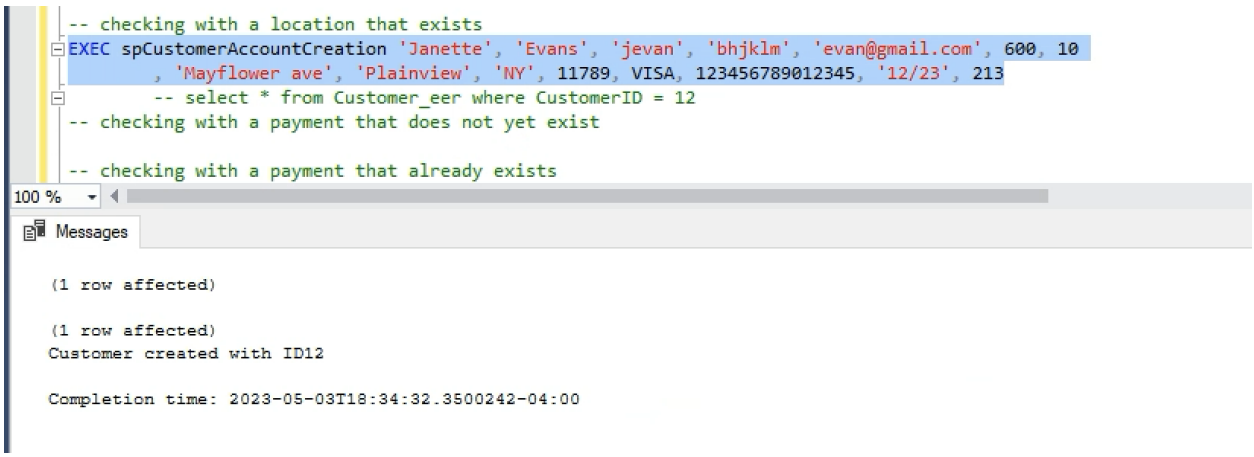
```
-- checking with a location that does not yet exist
EXEC spCustomerAccountCreation 'James', 'Dane', 'jadan', 'abcdef', 'jdan@gmail.com', 700, 34
        , 'Third Street', 'Miami', 'FL', 12385, VISA, 123456789012345, '12/23', 213
        --select * from Customer_eer where CustomerID = 11

-- checking with a location that exists
```

100 %

Messages

```
(1 row affected)
New location record inserted successfully

(1 row affected)
New payment record inserted successfully

(1 row affected)

(1 row affected)
Customer created with ID11

Completion time: 2023-05-03T18:33:35.1303661-04:00
```

Fig. 2-5 - Result set when registering with a location that already exists, and payment also already exists (reusing some data from the customer in Fig. 2-4), registration still successful



```
-- checking with a location that exists
EXEC spCustomerAccountCreation 'Janette', 'Evans', 'jevan', 'bhjklm', 'evan@gmail.com', 600, 10
        , 'Mayflower ave', 'Plainview', 'NY', 11789, VISA, 123456789012345, '12/23', 213
        -- select * from Customer_eer where CustomerID = 12
-- checking with a payment that does not yet exist

-- checking with a payment that already exists
```

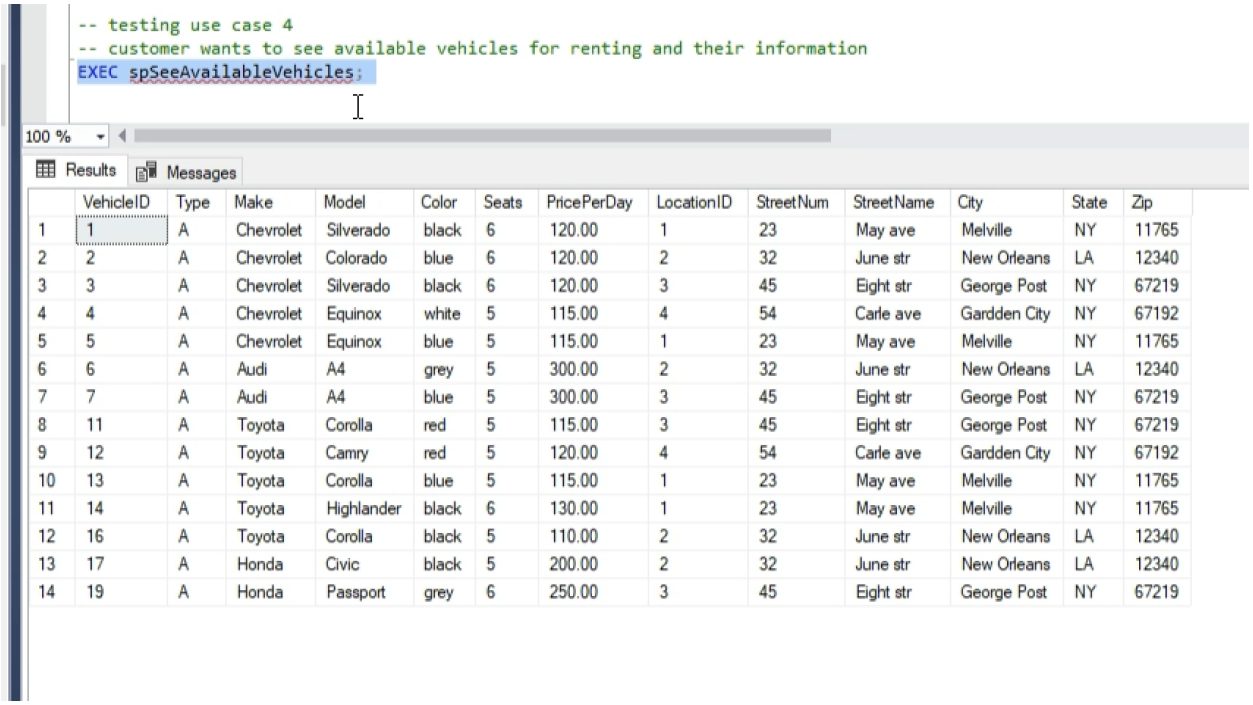
100 %

Messages

```
(1 row affected)

(1 row affected)
Customer created with ID12

Completion time: 2023-05-03T18:34:32.3500242-04:00
```

The testing of use case 4: available vehicles summary

Fig. 2-7 - Here is a summary of all the vehicles that are available to rent



```
-- testing use case 4
-- customer wants to see available vehicles for renting and their information
EXEC spSeeAvailableVehicles;
```

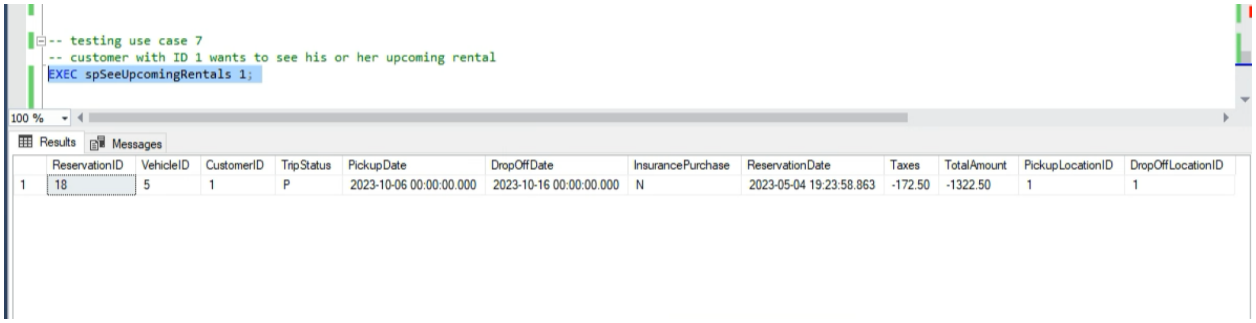
100 %

Results | Messages

	VehicleID	Type	Make	Model	Color	Seats	PricePerDay	LocationID	StreetNum	StreetName	City	State	Zip
1	1	A	Chevrolet	Silverado	black	6	120.00	1	23	May ave	Melville	NY	11765
2	2	A	Chevrolet	Colorado	blue	6	120.00	2	32	June str	New Orleans	LA	12340
3	3	A	Chevrolet	Silverado	black	6	120.00	3	45	Eight str	George Post	NY	67219
4	4	A	Chevrolet	Equinox	white	5	115.00	4	54	Carle ave	Gardden City	NY	67192
5	5	A	Chevrolet	Equinox	blue	5	115.00	1	23	May ave	Melville	NY	11765
6	6	A	Audi	A4	grey	5	300.00	2	32	June str	New Orleans	LA	12340
7	7	A	Audi	A4	blue	5	300.00	3	45	Eight str	George Post	NY	67219
8	11	A	Toyota	Corolla	red	5	115.00	3	45	Eight str	George Post	NY	67219
9	12	A	Toyota	Camry	red	5	120.00	4	54	Carle ave	Gardden City	NY	67192
10	13	A	Toyota	Corolla	blue	5	115.00	1	23	May ave	Melville	NY	11765
11	14	A	Toyota	Highlander	black	6	130.00	1	23	May ave	Melville	NY	11765
12	16	A	Toyota	Corolla	black	5	110.00	2	32	June str	New Orleans	LA	12340
13	17	A	Honda	Civic	black	5	200.00	2	32	June str	New Orleans	LA	12340
14	19	A	Honda	Passport	grey	6	250.00	3	45	Eight str	George Post	NY	67219

Testing Use Case 7: See Upcoming Rentals

Fig. 2-8 - Customers can see details about their upcoming reservations based on their ID



```
-- testing use case 7
-- customer with ID 1 wants to see his or her upcoming rental
EXEC spSeeUpcomingRentals 1;
```

100 %

Results | Messages

	ReservationID	VehicleID	CustomerID	TripStatus	PickupDate	DropOffDate	InsurancePurchase	ReservationDate	Taxes	TotalAmount	PickupLocationID	DropOffLocationID
1	18	5	1	P	2023-10-06 00:00:00.000	2023-10-16 00:00:00.000	N	2023-05-04 19:23:58.863	-172.50	-1322.50	1	1

Testing Use Case 2 : Upcoming Rental Management

Fig 2-9 - When customers try to cancel an already completed or canceled reservation, a message is displayed saying that such action cannot be taken

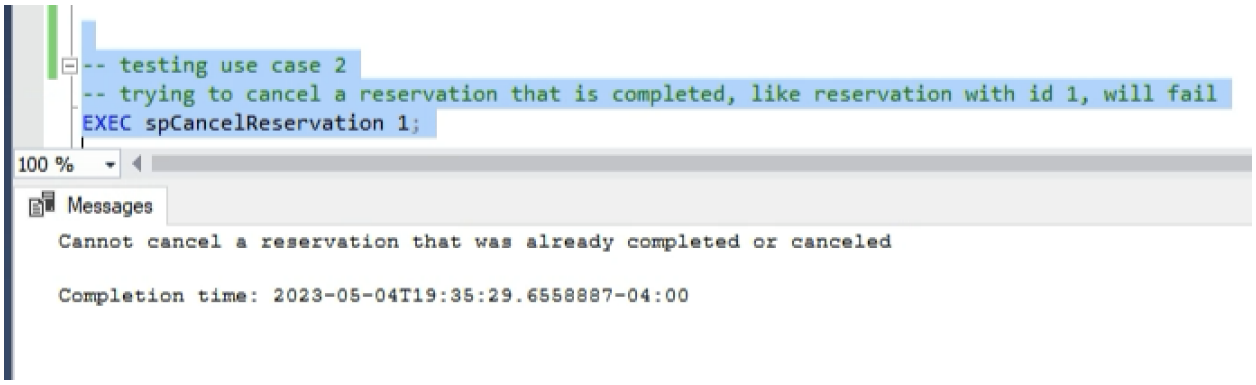
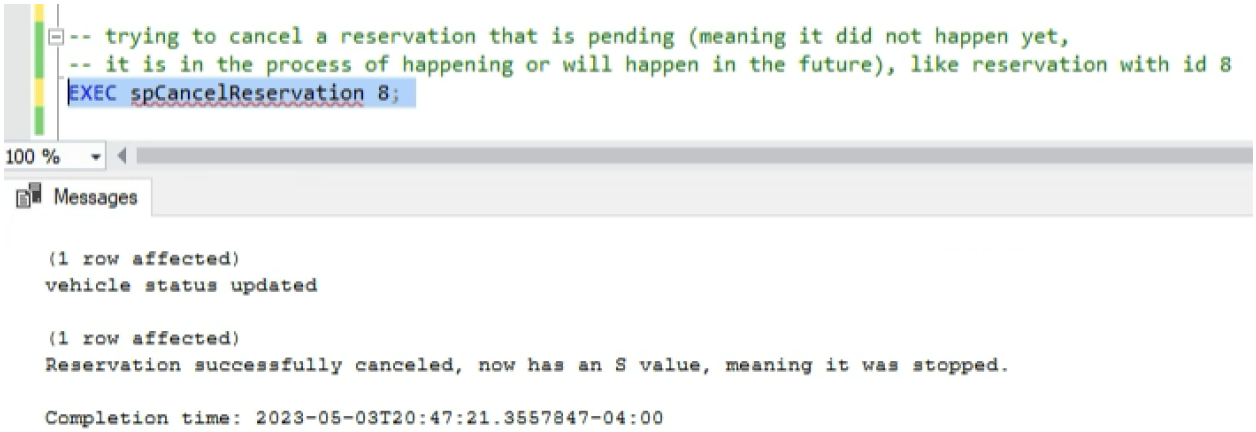


Fig. 2-10 - When a customer tries to cancel a reservation that is currently in the pending process, the value of the reservation changes to 'S' meaning the reservation was stopped, and thus, successfully canceled.



Testing Use Case 5: Vehicle Availability Status

Fig. 2-11 - once a reservation was successfully canceled, the vehicle status that was assigned to that reservation is changed to available - 'A' - and its presence can be seen through the stored procedure that shows the available vehicles.

```
-- testing use case 5
-- cancelling/ stopping a reservation, as is the case in the previous test of use case 2, changes the vehicle status to 'A'
EXEC spSeeAvailableVehicles;
```

100 %

Results | Messages

	VehicleID	Type	Make	Model	Color	Seats	PricePerDay	LocationID	StreetNum	StreetName	City	State	Zip
1	1	A	Chevrolet	Silverado	black	6	120.00	1	23	May ave	Melville	NY	11765
2	2	A	Chevrolet	Colorado	blue	6	120.00	2	32	June str	New Orleans	LA	12340
3	3	A	Chevrolet	Silverado	black	6	120.00	3	45	Eight str	George Post	NY	67219
4	4	A	Chevrolet	Equinox	white	5	115.00	4	54	Carle ave	Gardden City	NY	67192
5	5	A	Chevrolet	Equinox	blue	5	115.00	1	23	May ave	Melville	NY	11765
6	6	A	Audi	A4	grey	5	300.00	2	32	June str	New Orleans	LA	12340
7	7	A	Audi	A4	blue	5	300.00	3	45	Eight str	George Post	NY	67219
8	8	A	Audi	A6	red	5	250.00	4	54	Carle ave	Gardden City	NY	67192
9	11	A	Toyota	Corolla	red	5	115.00	3	45	Eight str	George Post	NY	67219
10	12	A	Toyota	Camry	red	5	120.00	4	54	Carle ave	Gardden City	NY	67192
11	13	A	Toyota	Corolla	blue	5	115.00	1	23	May ave	Melville	NY	11765
12	14	A	Toyota	Highlander	black	6	130.00	1	23	May ave	Melville	NY	11765
13	16	A	Toyota	Corolla	black	5	110.00	2	32	June str	New Orleans	LA	12340
14	17	A	Honda	Civic	black	5	200.00	2	32	June str	New Orleans	LA	12340
15	19	A	Honda	Passport	grey	6	250.00	3	45	Eight str	George Post	NY	67219

Testing Use Case 3: Vehicle is Rented

Fig. 2-12 Trying to rent a vehicle in which the user does not exist results in a message that shows the issue.

```
-- testing use case 3
-- if customer does not exist
EXEC spCreateRental 14, 25, '2023-05-02', '2023-05-15', 'N', 4, 4
-- if customer did not pass credit score check or is still processing
```

00 %

Messages

```
Customer does not exist

Completion time: 2023-05-04T18:57:50.2164290-04:00
```

Fig. 2-13 - if the customer wants to rent a vehicle that is taken by someone else in a pending reservation, a message is printed saying that the vehicle is taken

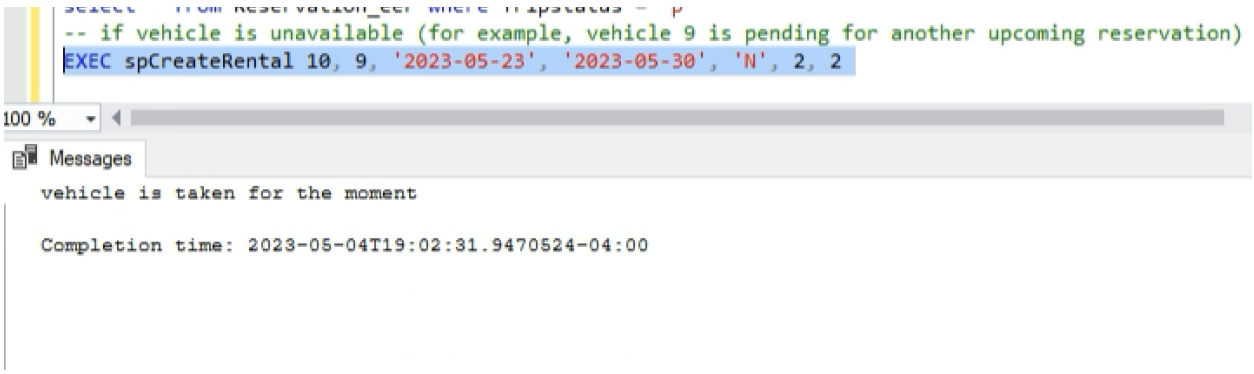


Fig. 2-14 - if the customer wants to rent a vehicle that does not exist at a location, it prints a message

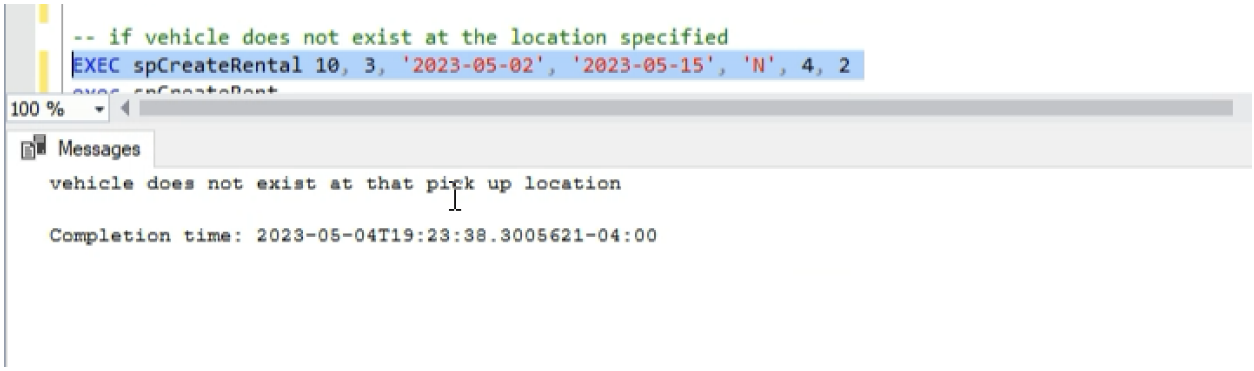
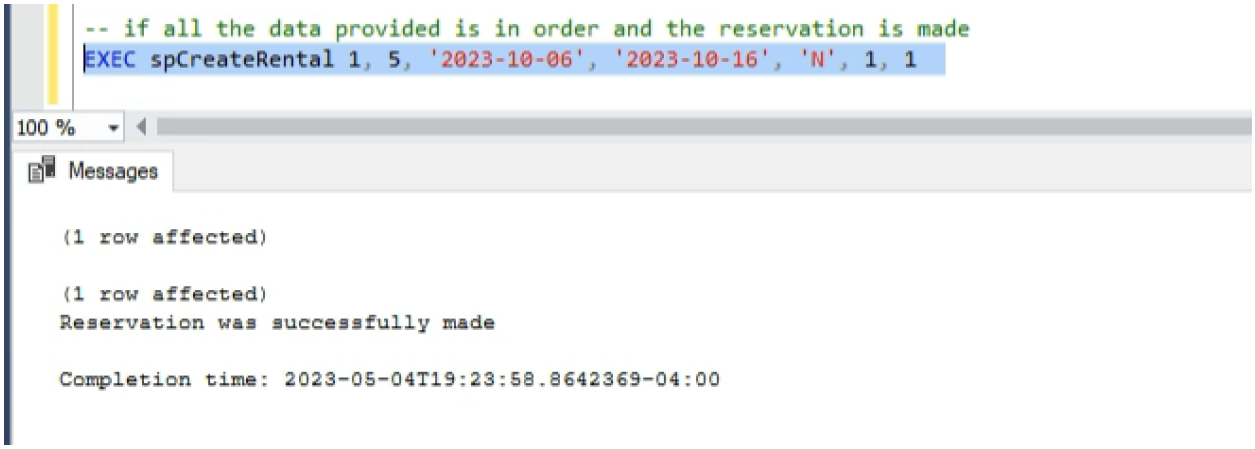
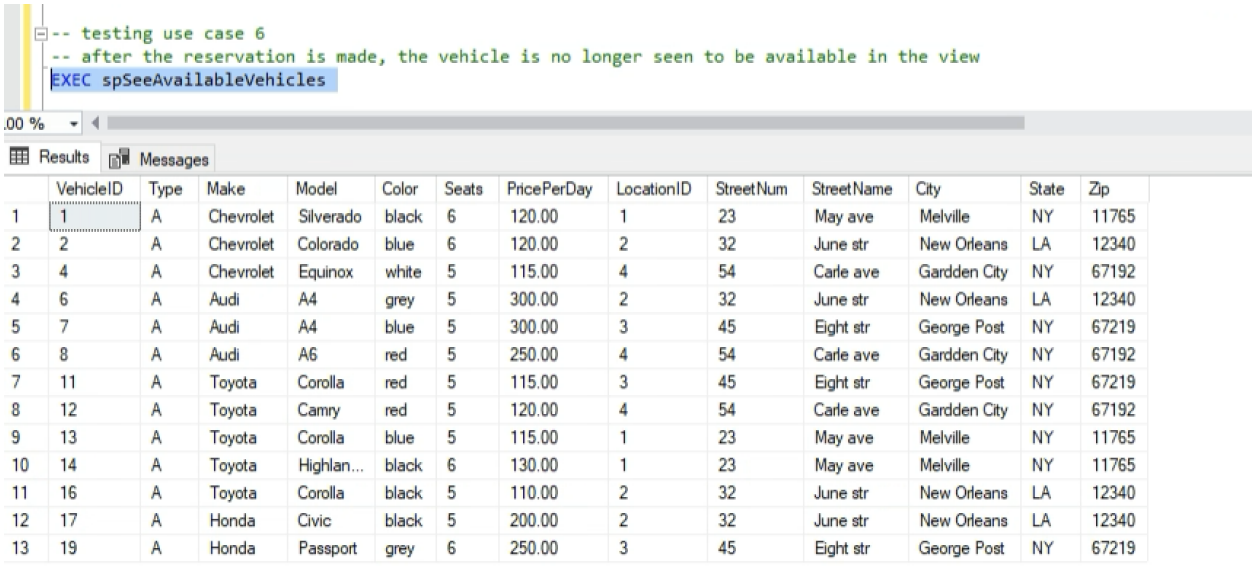


Fig. 2-15 - if the customer successfully creates a reservation, it tells the user that the reservation was made.



Testing Use Case 6: Vehicle Taken Status

Fig. 2-16: After a reservation was made, the status of that vehicle is changed to 'T' meaning 'taken', as is the case with the vehicle with ID 5 from the previous example, and will thus not be shown in the stored procedure that displays the available vehicles and their information



```
-- testing use case 6
-- after the reservation is made, the vehicle is no longer seen to be available in the view
EXEC spSeeAvailableVehicles
```

.00 %

Results | Messages

	VehicleID	Type	Make	Model	Color	Seats	PricePerDay	LocationID	StreetNum	StreetName	City	State	Zip
1	1	A	Chevrolet	Silverado	black	6	120.00	1	23	May ave	Melville	NY	11765
2	2	A	Chevrolet	Colorado	blue	6	120.00	2	32	June str	New Orleans	LA	12340
3	4	A	Chevrolet	Equinox	white	5	115.00	4	54	Carle ave	Gardden City	NY	67192
4	6	A	Audi	A4	grey	5	300.00	2	32	June str	New Orleans	LA	12340
5	7	A	Audi	A4	blue	5	300.00	3	45	Eight str	George Post	NY	67219
6	8	A	Audi	A6	red	5	250.00	4	54	Carle ave	Gardden City	NY	67192
7	11	A	Toyota	Corolla	red	5	115.00	3	45	Eight str	George Post	NY	67219
8	12	A	Toyota	Camry	red	5	120.00	4	54	Carle ave	Gardden City	NY	67192
9	13	A	Toyota	Corolla	blue	5	115.00	1	23	May ave	Melville	NY	11765
10	14	A	Toyota	Highlan...	black	6	130.00	1	23	May ave	Melville	NY	11765
11	16	A	Toyota	Corolla	black	5	110.00	2	32	June str	New Orleans	LA	12340
12	17	A	Honda	Civic	black	5	200.00	2	32	June str	New Orleans	LA	12340
13	19	A	Honda	Passport	grey	6	250.00	3	45	Eight str	George Post	NY	67219