**DBMS - Mini Project**

**Car Rental Management System**

**Submitted By: Name: Y Srinivas**

**SRN: PES1UG20CS517**

**V Semester Section I**

# Short Description and Scope of the Project

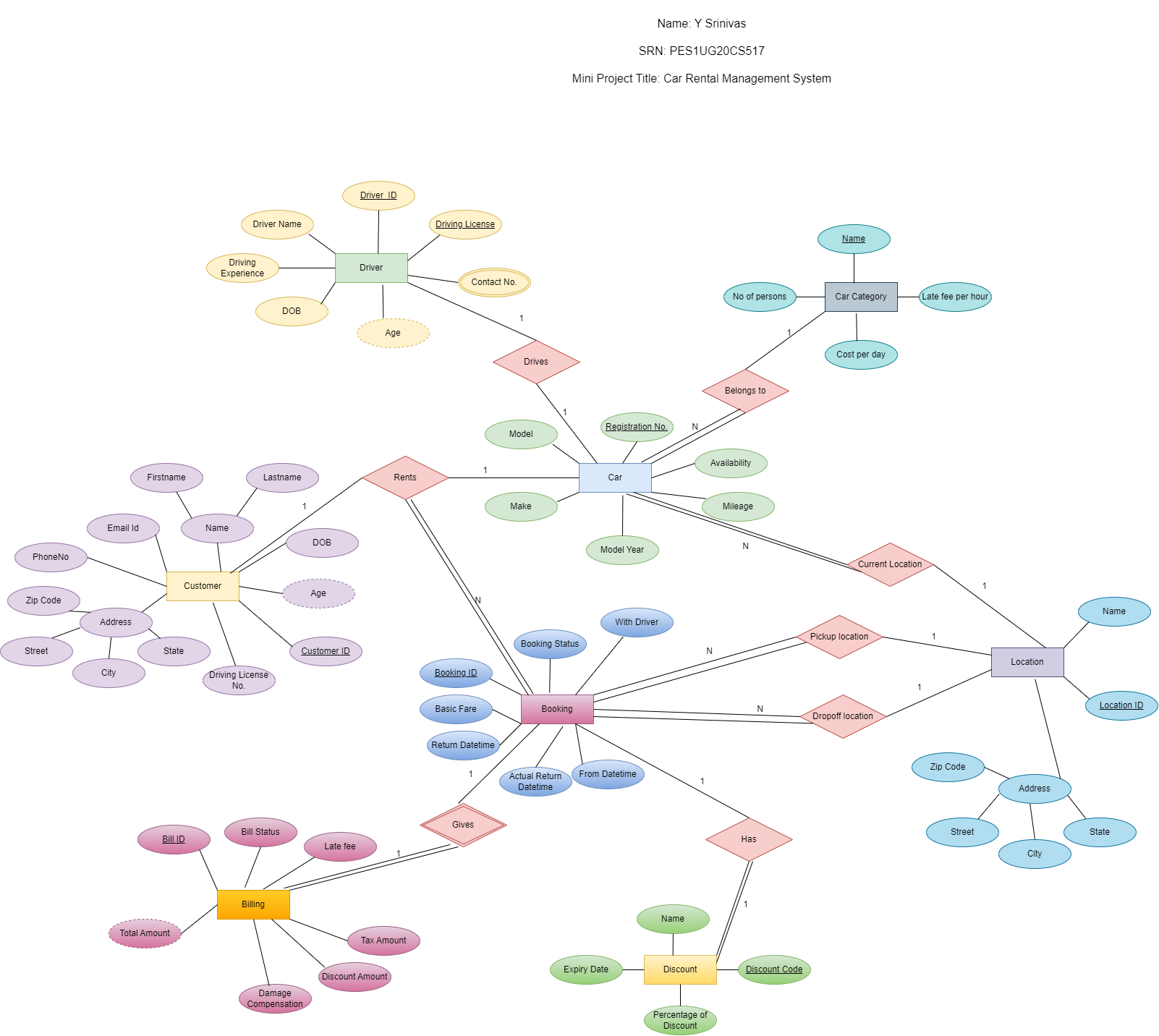
**Description**:

* Car rental management system is a utility which can be useful for a car rental agency and for people who rent cars, by providing solutions for managing cars, customers and the cars taken for rent by customers.
* Our system provides customer to have different pick-up and drop-off locations and will impose late fee if the rental car is returned beyond the return date and time.
* Customer has a choice to rent a car with or without the driver.
* If customer rents a car with the driver, then they will be charged driver fare based on the driver experience.
* If customer decides to rent a car without driver, then he has to provide his driving license number that is mandatory.
* Customer has given with a lot discount coupons each coupon has different discount percentage.
* After they return the Car, we are going to enter the actual return date. Based on all these details we are going to calculate the Amount to be paid by the customer using Functions and updating the payment details using procedure and Cursors.

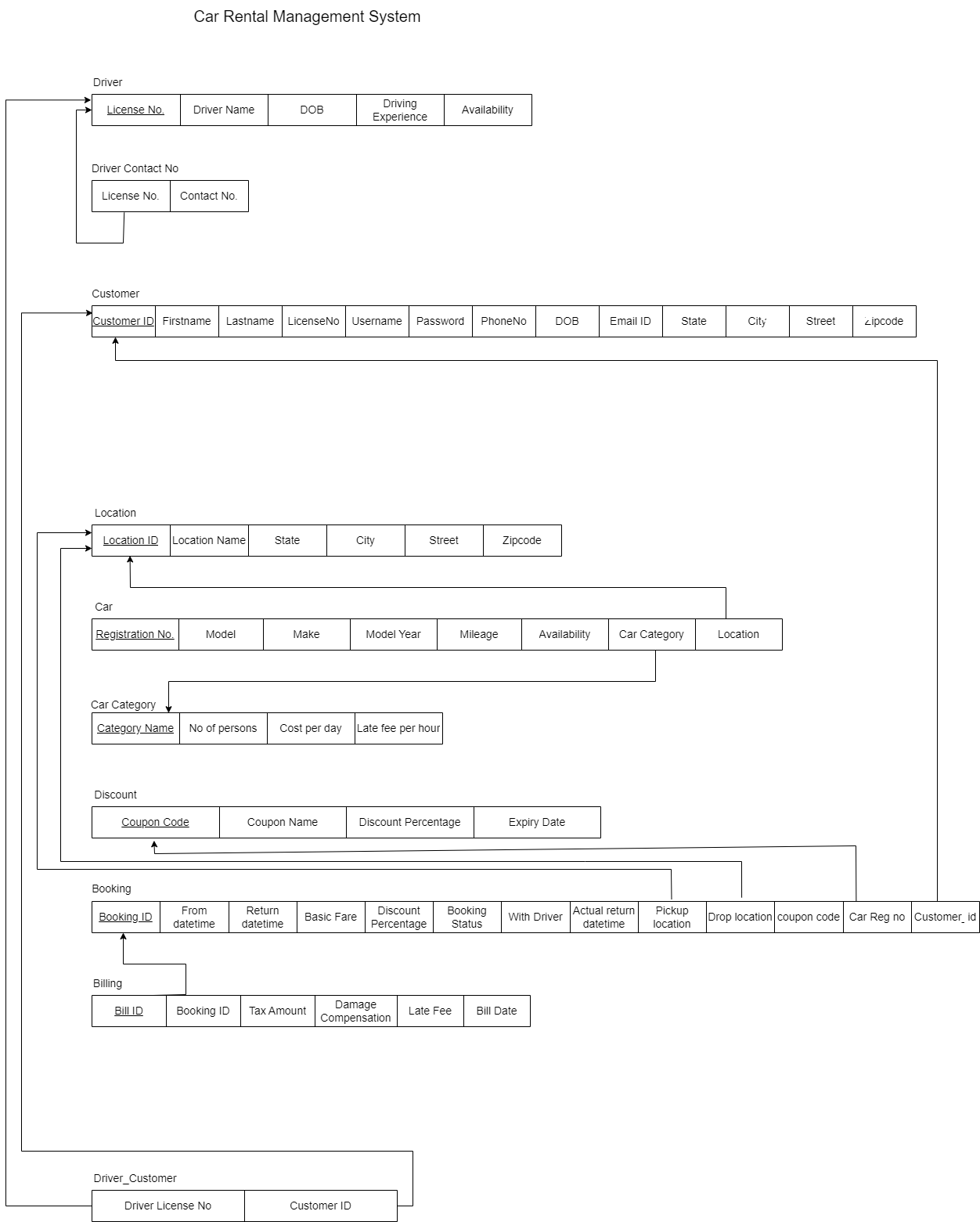
**Scope**:

* Renting a car is a self-sustaining system which is the best opportunity for the people who cannot afford to buy the car in their family.
* Due to this system the people can borrow the car on rent for some time and do they compulsory work while paying the charges the rented car.
* Making a company available to customers 24 hours a day, seven days a week.
* Online systems reduce the time it takes to rent a car and the costs of hiring people to input data into paper-based records.
* Having all the records in one place it is much easier for you to track your expenses and budget appropriately. This can help with financial planning and decision-making for the future of your business.

**ER Diagram**



# Relational Schema



**DDL statements - Building the database**

-- Driver's information

create table driver\_info(

    dl\_number char(16) not null,

    constraint pk\_driver\_dl primary key(dl\_number),

    driver\_name varchar(35) not null,

    driver\_dob date not null,

    driving\_experience int not null,

    available boolean default true

);

alter table driver\_info

add constraint check\_experience check(driving\_experience >= 2);

-- Drivers Contact numbers

create table driver\_contacts(

    dl\_number char(16) not null,

    constraint fk\_driver\_dl foreign key(dl\_number) references driver\_info(dl\_number),

    phone\_no char(10)

);

alter table driver\_contacts

modify phone\_no char(10) not null;

alter table driver\_contacts

add constraint check\_phone check(char\_length(phone\_no) = 10);

-- Customer's information

create table customer\_info(

    customer\_id int not null auto\_increment,

    dl\_number char(16) unique,

    firstname varchar(35) not null,

    lastname varchar(35) not null,

    phone\_no char(10) not null,

    username varchar(35) not null unique,

    password varchar(50) not null,

    dob date not null,

    email varchar(35),

    state varchar(35) not null,

    city varchar(35) not null,

    street varchar(35) not null,

    zipcode char(5) not null,

    constraint pk\_customer\_id primary key(customer\_id),

    constraint check\_phone\_customer check(char\_length(phone\_no) = 10)

);

alter table customer\_info auto\_increment = 20000;

create table locations(

    location\_id int not null auto\_increment,

    location\_name varchar(50) not null,

    state varchar(35) not null,

    city varchar(35) not null,

    street varchar(35) not null,

    zipcode int(5) not null,

    constraint pk\_location\_id primary key(location\_id)

);

alter table locations auto\_increment = 40000;

alter table locations

modify location\_name varchar(50) not null unique;

-- Car Category

create table car\_category(

    category\_name varchar(35) not null,

    no\_persons int not null,

    cost\_per\_day double not null,

    late\_fee\_per\_hour double not null,

    constraint pk\_category\_name primary key(category\_name)

);

-- Car details

create table car\_details(

    registration\_no char(6) not null,

    model\_name varchar(35) not null,

    make varchar(35) not null,

    model\_year int(4) not null,

    mileage double not null,

    available boolean default true,

    category varchar(35) not null,

    car\_location int not null,

    constraint pk\_car\_reg primary key(registration\_no),

    constraint fk\_car\_category foreign key(category) references car\_category(category\_name),

    constraint fk\_car\_location foreign key(car\_location) references locations(location\_id)

);

-- Discount

create table discount(

    coupon\_code char(4) not null,

    coupon\_name varchar(35) not null,

    discount\_percentage double not null,

    expiry\_date date not null,

    constraint pk\_discount\_coupon primary key(coupon\_code)

);

-- Booking Details

create table booking\_details(

    booking\_id int not null auto\_increment,

    from\_date date not null,

    return\_date date not null,

    basic\_fare double not null, -- no. of days \* price per day of car - discount amt

    discount\_percentage double default 0, -- from discount percentage

    booking\_status boolean default false,

    with\_driver boolean default false,

    actual\_return\_date date not null,

    pickup\_location int not null,

    drop\_location int not null,

    coupon\_code char(4),

    car\_reg\_no char(6) not null,

    customer\_id int not null,

    booking\_date date default current\_date(),

    constraint pk\_booking\_id primary key(booking\_id),

    constraint fk\_booking\_coupon foreign key(coupon\_code) references discount(coupon\_code),

    constraint fk\_booking\_car foreign key(car\_reg\_no) references car\_details(registration\_no),

    constraint fk\_booking\_pickup foreign key(pickup\_location) references locations(location\_id),

    constraint fk\_booking\_pickdrop foreign key(drop\_location) references locations(location\_id),

    constraint fk\_booking\_customer foreign key(customer\_id) references customer\_info(customer\_id)

);

alter table booking\_details auto\_increment=50000;

-- Billing Details

create table billing\_details(

    bill\_id int not null auto\_increment,

    booking\_id int not null,

    tax\_amount double default 20.0,

    damage\_compensation double default 0,

    late\_fee double default 0, -- if actual return date > return date

    bill\_date date default current\_date(),

    constraint pk\_bill\_id primary key(bill\_id),

    constraint fk\_bill\_booking foreign key(booking\_id) references booking\_details(booking\_id)

);

alter table billing\_details auto\_increment = 60000;

-- Customer who has booked with driver those detils will be stored here

create table booking\_with\_driver(

    id int not null auto\_increment,

    driver\_dl char(16) not null,

    customer\_id int not null,

    booking\_id int not null,

    booking\_date date default current\_date(),

    constraint pk\_driver\_customer\_id primary key(id),

    constraint fk\_booking\_driver\_customer foreign key(driver\_dl) references driver\_info(dl\_number),

    constraint fk\_booking\_customer\_driver foreign key(customer\_id) references customer\_info(customer\_id),

    constraint fk\_booking\_id foreign key(booking\_id) references booking\_details(booking\_id)

);

alter table booking\_with\_driver auto\_increment = 70000;

# Populating the Database

-- driver's data

insert into driver\_info values

    ( "HR-0619830034761", "Liam", "1975-01-07", 15, true),

    ( "HR-0619850034771", "Noah", "1970-03-17", 8, false),

    ("HR-0619880034781", "Oliver", "1985-05-20", 10, true),

    ( "HR-0619890034791", "Elijah", "1995-07-11", 6, false),

    ( "HR-0619820034661", "James", "1972-09-18", 20, true);

-- driver's contacts

insert into driver\_contacts values

    ("HR-0619830034761", 9982641789),

    ("HR-0619830034761", 9927593732),

    ("HR-0619850034771", 8535919898),

    ("HR-0619880034781", 9972104143),

    ("HR-0619890034791", 6363212645),

    ("HR-0619890034791", 7676676566),

    ("HR-0619820034661", 9591858426);

-- Customers info

insert into customer\_info( dl\_number, firstname, lastname, phone\_no, username, password, dob, email, state, city, street, zipcode ) values

    ( "HR-0719830034891", "Adler", "Anderson", 8322335022, "alderanderson", "1234@", "2000-02-04", "alderanderson@gmail.com", "Alabama", "Montgomery", "Main Street.", 40202 ),

    ( "HR-0719830034892", "Seth", "Ivan", 7926870547, "sethivan", "abcd@", "2002-03-05", "sethivan@gmail.com", "Alaska", "Juneau", "2nd Street.", 40203 ),

    ( "HR-0719830034893", "Riley", "Gilbert", 9822334254, "rileygilbert", "wxyz@", "1975-04-06", "rileygilbert@gmail.com", "Arizona", "Phoenix", "7th Street.", 40204),

    ( null, "Jorge", "Dan", 9841310497, "jorgedan", "1234@", "1980-05-07", null, "Arkansas", "Little Rock", "3rd Street.", 40205 ),

    ( "HR-0719830034895", "Brian", "Roberto", 9998958055, "brianroberto", "abcd@", "2003-06-08", "brianroberto@gmail.com", "California", "Sacramento", "1st Street.", 40207 ),

    ( null, "Ramon", "Miles", 7759228501, "ramonmiles", "wxyz@", "2005-07-09", "ramonmiles@gmail.com", "Alabama", "Montgomery", "Main Street", 40202),

    ("HR-0719830034897","Liam","Nathaniel",8212415127,"liamnathaniel","1234@","1960-08-10","liamnathaniel@gmail.com","Alabama","Montgomery","Main Street",40202);

-- Locations

insert into locations(location\_name, state, city, street, zipcode) VALUES

    ( 'Private AIRPORT', "Alabama", "Montgomery", "Main Street.", 40202 ),

    ( 'DALLAS LOVE FIELD AIRPORT', "Alabama", "Montgomery", "Main Street.", 40202 ),

    ( 'LOS ANGELES INTL AIRPORT', "Alaska", "Juneau", "2nd Street.", 40203 ),

    ( 'DALLAS/ FORT WORTH INTL AIRPORT', "Alabama", "Montgomery", "Main Street.", 40202 ),

    ( 'WEST HOUSTON AIRPORT', "Arizona", "Phoenix", "7th Street.", 40204),

    ( 'WASHINGTON DULLES INTL AIRPORT', "California", "Sacramento", "1st Street.", 40207);

-- car category

insert into car\_category values

    ('ECONOMY', 5, 30, 0.9),

    ('COMPACT', 5, 32, 0.96),

    ('MID SIZE', 5, 35, 1.05),

    ('STANDARD', 5, 38, 1.14),

    ('FULL SIZE', 5, 40, 1.2),

    ('LUXURY CAR', 5, 75, 2.25),

    ('MID SIZE SUV', 5, 36, 1.08),

    ('STANDARD SUV', 5, 40, 1.2),

    ('FULL SIZE SUV', 8, 60, 1.8),

    ('MINI VAN', 7, 70, 2.1);

-- Car Details

insert into car\_details( registration\_no, model\_name, make, model\_year, mileage, category, car\_location) values

    ( 'AB1234', 'CIVIC', 'HONDA', 2014, 8, 'ECONOMY', 40000),

    ( 'SD4567', 'FIESTA', 'FORD', 2015, 6, 'ECONOMY', 40001),

    ( 'GLZ2376', 'COROLLA', 'TOYOTA', 2016, 5.000, 'ECONOMY', 40002),

    ( 'WER3245', 'ACCENT', 'HYUNDAI', 2014, 12.356, 'ECONOMY', 40003),

    ( 'HJK1234', 'CIVIC', 'HONDA', 2015, 20.145, 'ECONOMY', 40004),

    ( 'GLS7625', 'FOCUS', 'FORD', 2014, 12.01, 'COMPACT', 40001),

    ( 'FKD8202', 'GOLF', 'VOLKSWAGAN', 2016, 11.5, 'COMPACT', 40002),

    ( 'HNX1890', 'PRIUS', 'TOYOTA', 2015, 7.8, 'COMPACT', 40003),

    ( 'KJS1983', 'PRIUS', 'TOYOTA', 2014, 9.5, 'COMPACT', 40004),

    ( 'SDL9356', 'FOCUS', 'FORD', 2016, 10, 'COMPACT', 40003),

    ( 'OTY7293', 'CRUZE', 'CHEVROLET', 2016, 14, 'MID SIZE', 40002);

-- discount details

insert into discount( coupon\_code, coupon\_name, expiry\_date, discount\_percentage) values

    ( 'D678', 'IBM CORPORATE', '2023-01-25', 25),

    ( 'D234', 'CTS CORPORATE', '2024-09-02', 20),

    ( 'D109', 'WEEKLY RENTALS', '2022-11-09', 25),

    ( 'D972', 'ONE WAY SPECIAL', '2023-12-15', 20),

    ( 'D297', 'UPGRADE SPECIAL', '2025-02-18', 20),

    ( 'D756', 'HOLIDAY SPECIAL', '2021-10-29', 10);

**Join Queries**

1. Show all the car details and category to which car belongs to where car available for booking.

Query:

select \* from car\_details inner join car\_category on category = category\_name where available = true order by cost\_per\_day;



1. Show car details along with its current location.

Query:

select \* from car\_details inner join locations on car\_location = location\_id;



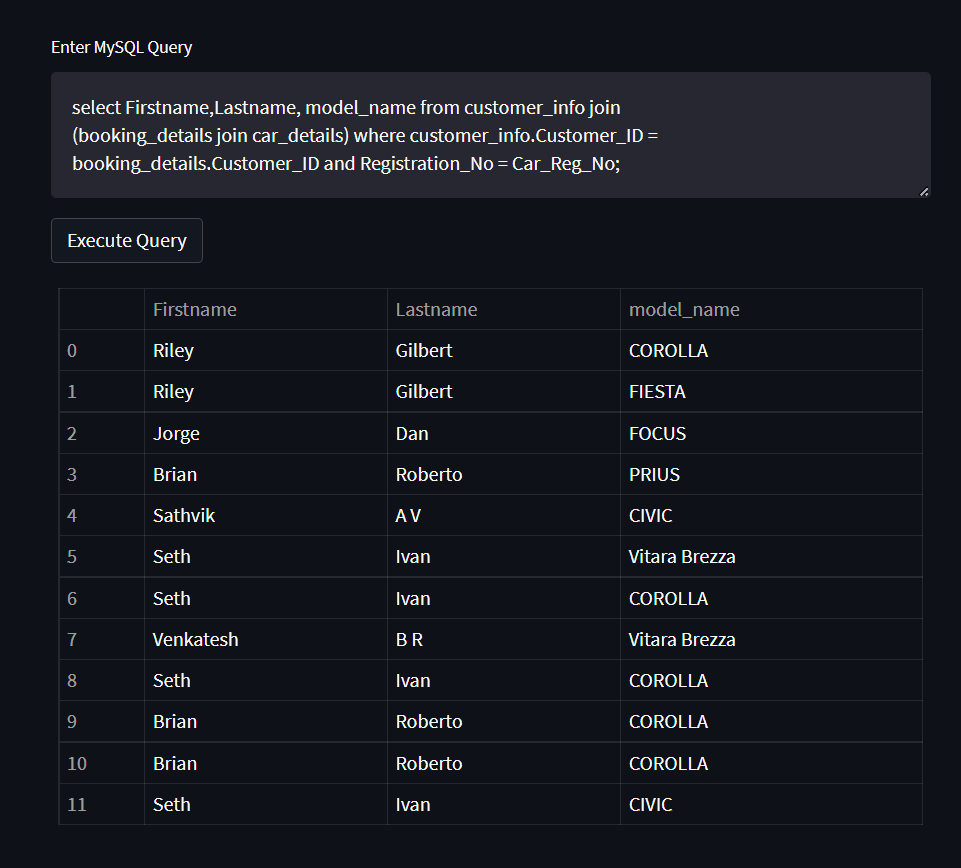
1. Show cars booked by a customer.

Query:

select Firstname,Lastname, model\_name from customer\_info join

(booking\_details join car\_details) where customer\_info.Customer\_ID =

booking\_details.Customer\_ID and Registration\_No = Car\_Reg\_No;



1. Display Car Details along with along with the category to which it belongs, No of persons it can hold, Cost per day.

Query:

select \* from car\_details inner join car\_category on category = category\_name order by cost\_per\_day;

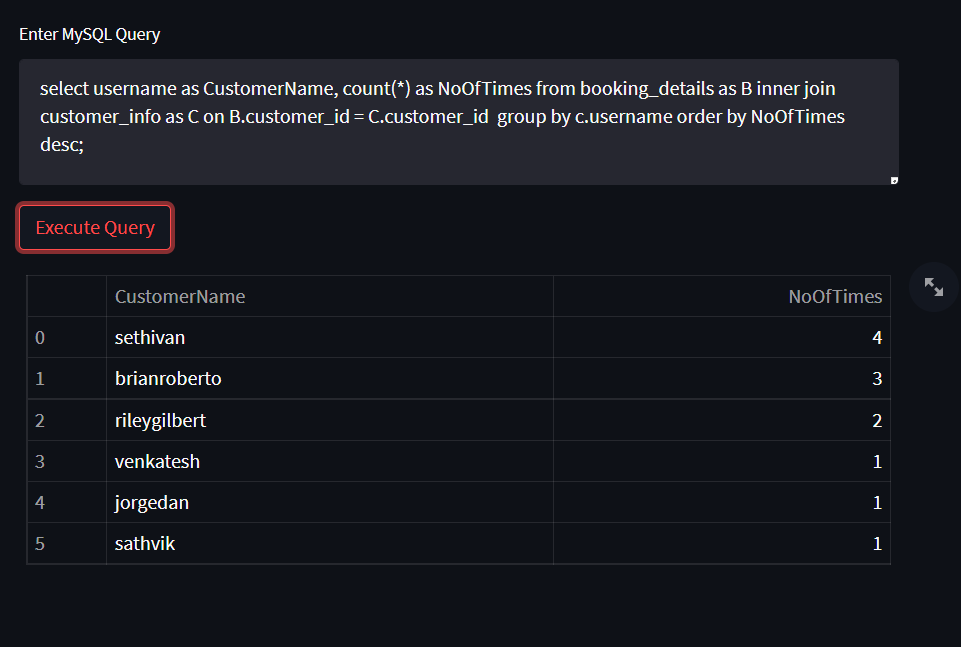


# Aggregate Functions

1. Show number of bookings made by each customer.

Query:

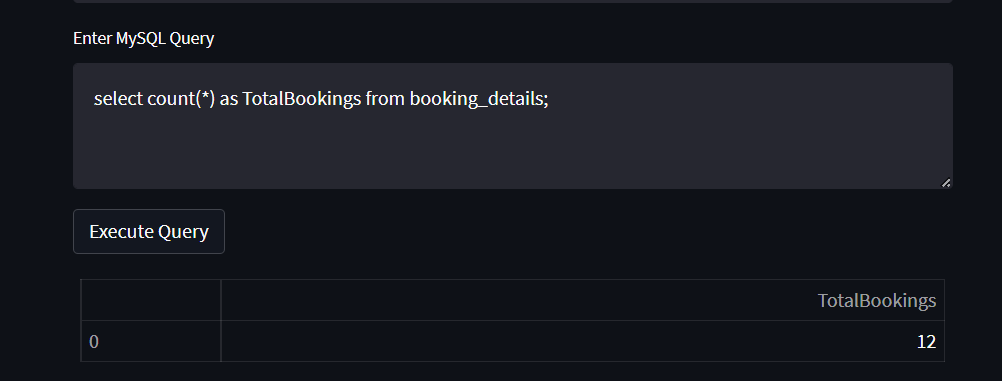
select username as CustomerName, count(\*) as NoOfTimes from booking\_details as B inner join customer\_info as C on B.customer\_id = C.customer\_id  group by c.username order by NoOfTimes desc;



1. Show total number of bookings done till today.

Query:

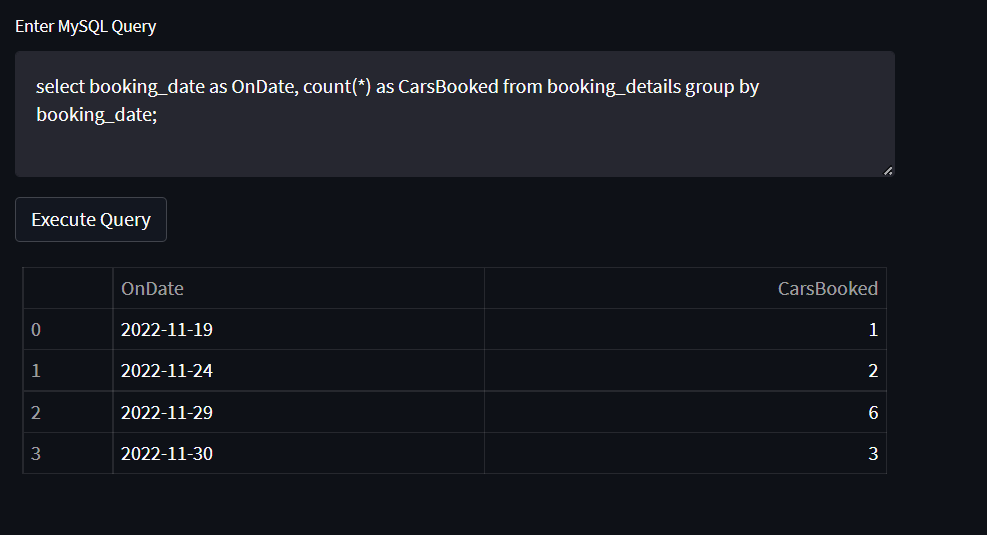
select count(\*) as TotalBookings from booking\_details;



1. Show number of cars booked on particular date.

Query:

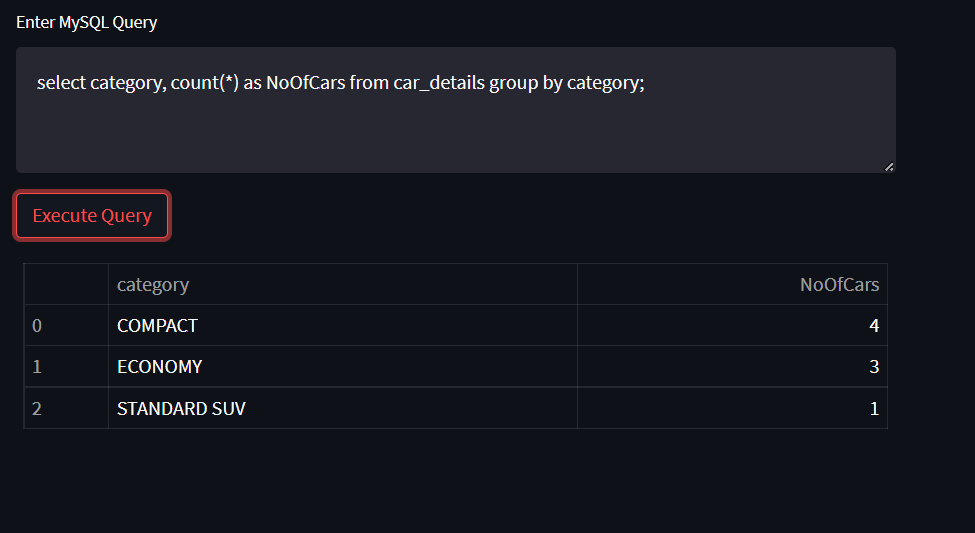
select booking\_date as OnDate, count(\*) as CarsBooked from booking\_details group by booking\_date;



1. Show number of cars belong to each category.

Query:

select category, count(\*) as NoOfCars from car\_details group by category;



# Set Operations

1. Display the Customer First name and Last Name who have returned the Car on or before the Return Date.

Query**:**

select Firstname,Lastname from customer\_info join booking\_details where

customer\_info.Customer\_ID = booking\_details.Customer\_ID and Return\_Date =

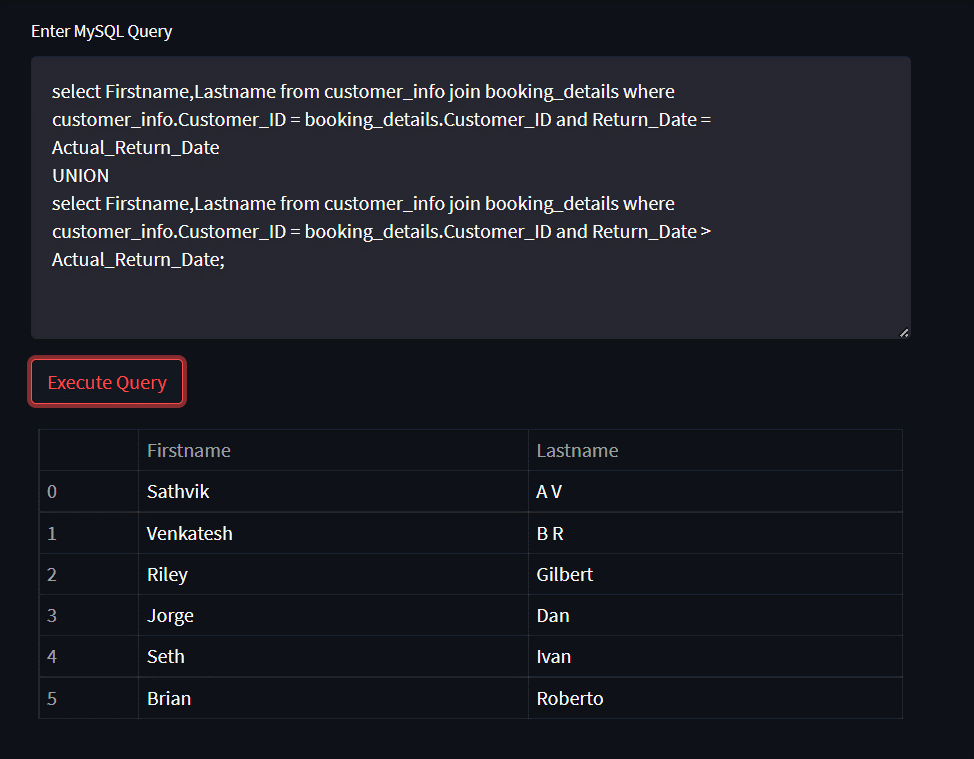
Actual\_Return\_Date

UNION

select Firstname,Lastname from customer\_info join booking\_details where

customer\_info.Customer\_ID = booking\_details.Customer\_ID and Return\_Date >

Actual\_Return\_Date;



1. **Display the Car that were Booked and the Model\_Year >2014.**

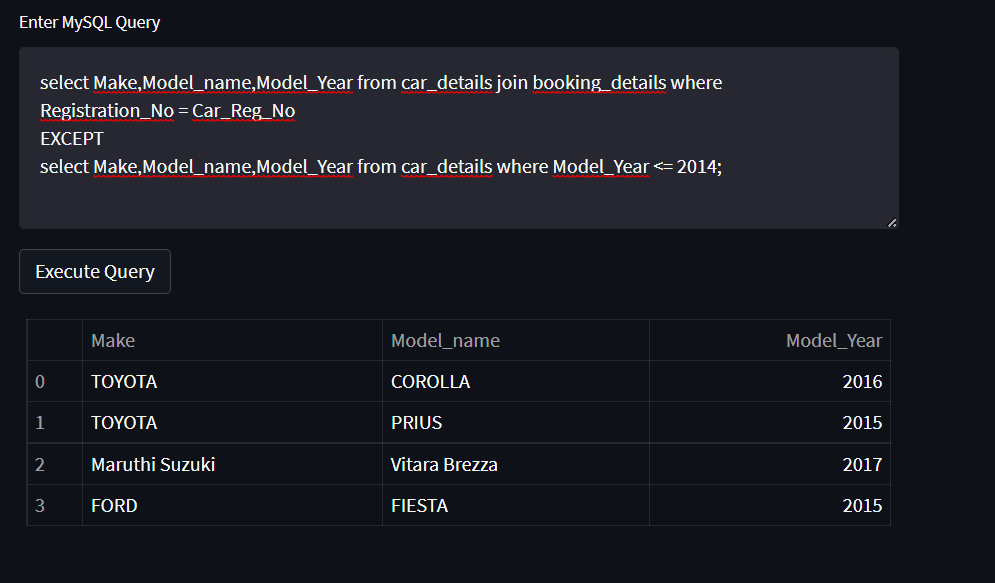
**Query:**

select Make,Model\_name,Model\_Year from car\_details join booking\_details where

Registration\_No = Car\_Reg\_No

EXCEPT

select Make,Model\_name,Model\_Year from car\_details where Model\_Year <= 2014;



# Functions and Procedures

**Function to calculate driver fare based on his experience.**

drop function if exists driver\_charge;

DELIMITER $

CREATE FUNCTION driver\_charge (experience int)

  RETURNS double

   DETERMINISTIC

    BEGIN

     DECLARE charge double;

        IF experience > 3 and experience < 6

            THEN SET charge = 500.0;

        ELSEIF  experience > 5 and experience < 10

            THEN SET charge = 700.0;

        ELSEIF  experience >= 10

            THEN SET charge = 1000.0;

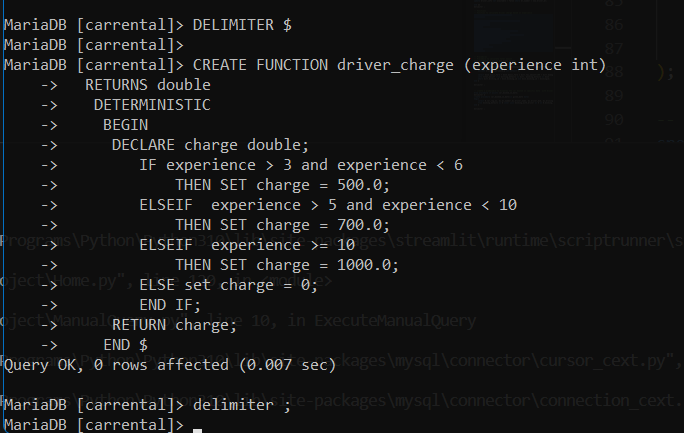
        ELSE set charge = 0;

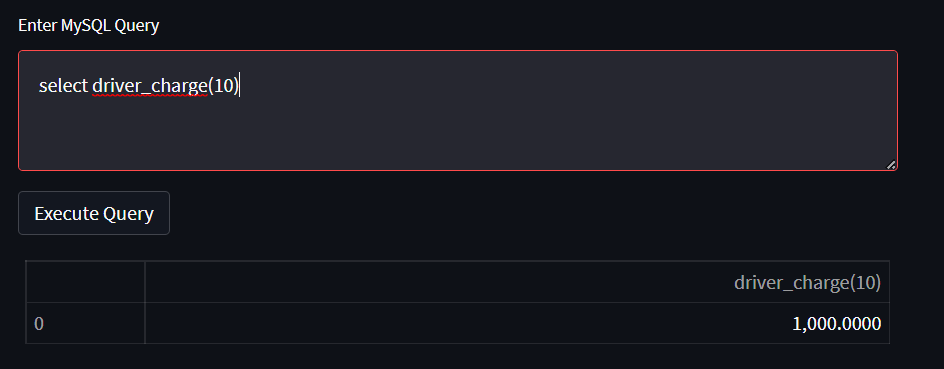
        END IF;

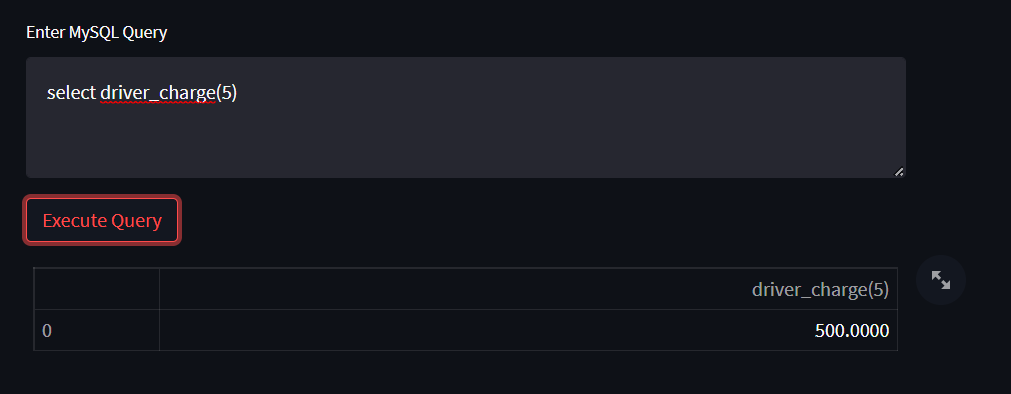
     RETURN charge;

    END $

delimiter ;







**Procedure to calculate total amount of a booking with including tax amount and driver fare.**

drop procedure if exists get\_total\_amount;

delimiter $

create Procedure get\_total\_amount(IN bookingid INT, OUT total\_amount double)

begin

    DECLARE driver\_experience\_p int default 0;

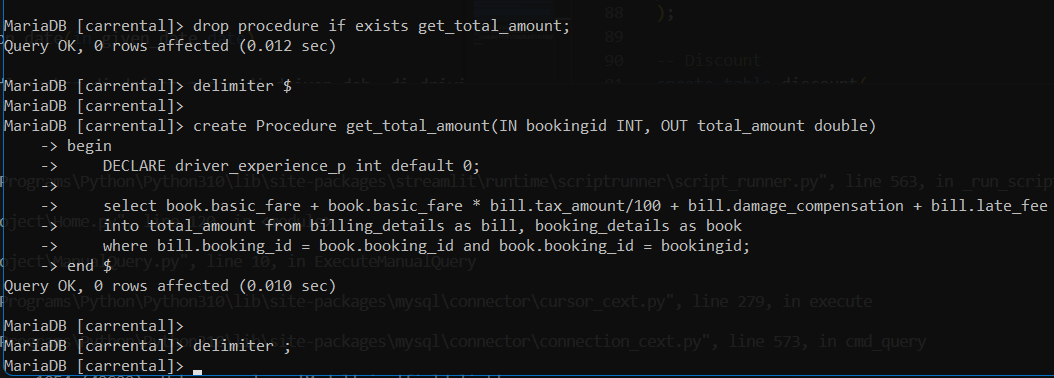
    select book.basic\_fare + book.basic\_fare \* bill.tax\_amount/100 + bill.damage\_compensation + bill.late\_fee

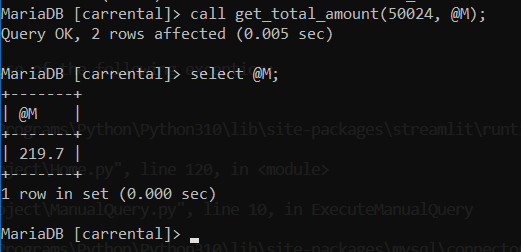
    into total\_amount from billing\_details as bill, booking\_details as book

    where bill.booking\_id = book.booking\_id and book.booking\_id = bookingid;

end $

delimiter ;





# Triggers and Cursors

1. Set the availability of a particular driver to false when that car is booked.

drop trigger if exists after\_driver\_customer;

delimiter $$

create trigger after\_driver\_customer

after insert

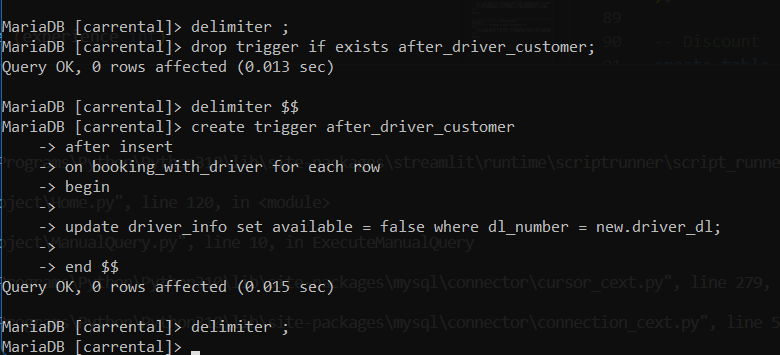
on booking\_with\_driver for each row

begin

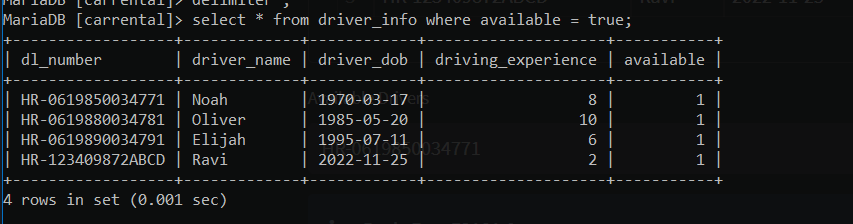
update driver\_info set available = false where dl\_number = new.driver\_dl;

end $$

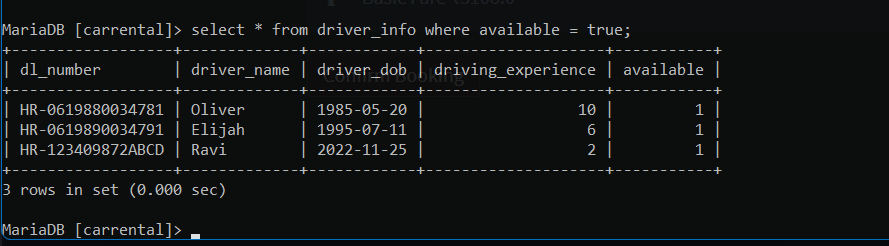
delimiter ;



Initial available drivers.



After Booking with driver.



A cursor for looping over the usernames of customers, and a NOT FOUND handler:

drop procedure if exists createUsernameList;

DELIMITER $$

CREATE PROCEDURE createUsernameList (

    INOUT usernameList varchar(4000)

)

BEGIN

    DECLARE finished INTEGER DEFAULT 0;

    DECLARE username varchar(100) DEFAULT "";

    -- declare cursor for employee email

    DEClARE curUsername

        CURSOR FOR

            SELECT username FROM customer\_info;

    -- declare NOT FOUND handler

    DECLARE CONTINUE HANDLER

        FOR NOT FOUND SET finished = 1;

    OPEN curUsername;

    getUsername: LOOP

        FETCH curUsername INTO username;

        IF finished = 1 THEN

            LEAVE getUsername;

        END IF;

        -- build email list

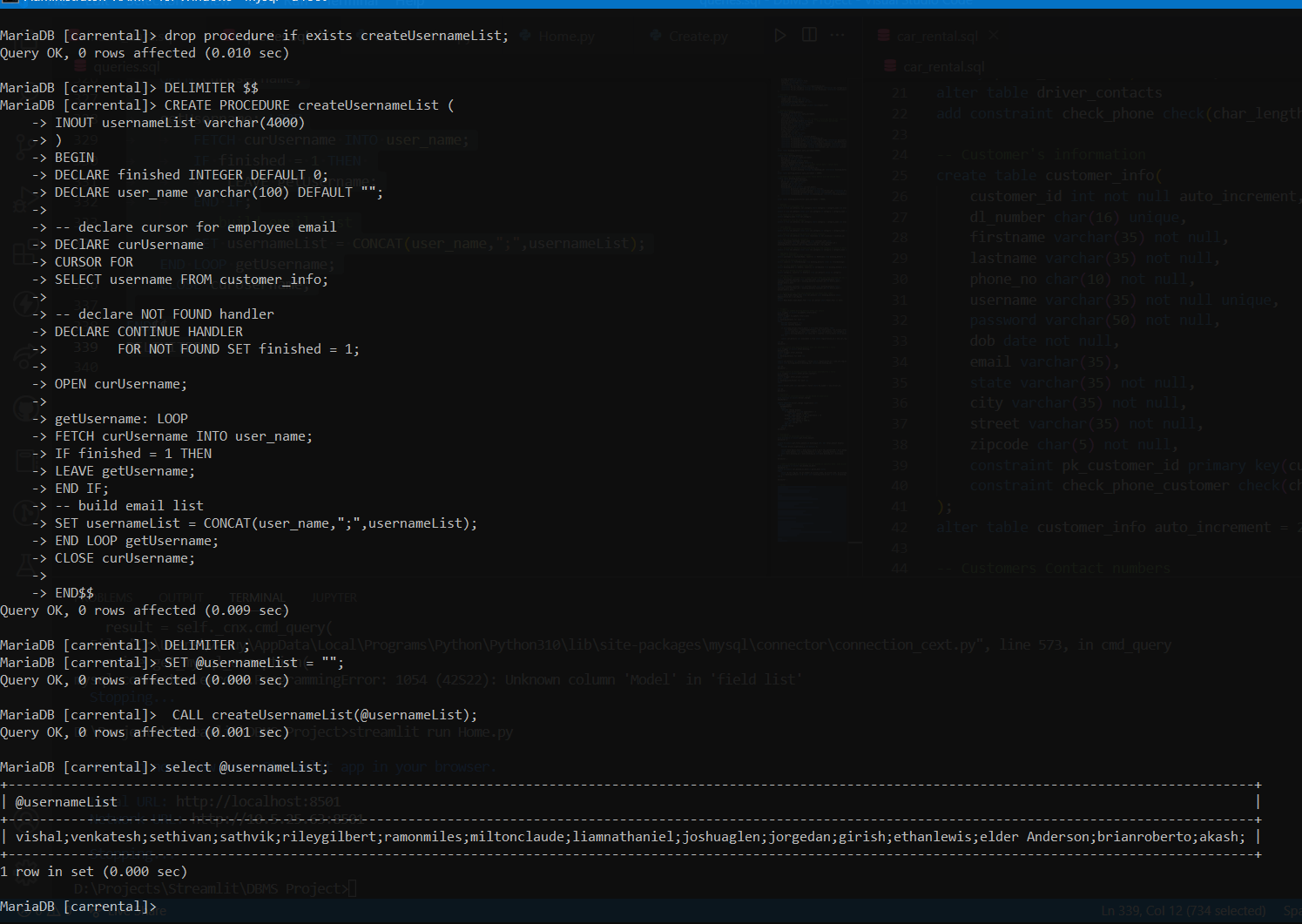
        SET usernameList = CONCAT(username,";",usernameList);

    END LOOP getUsername;

    CLOSE curUsername;

END$$

DELIMITER ;



Modification Given during demo:

create a prodecudre to dispalay the car booked on specific date  with driver details.

**Query**:

drop procedure if exists car\_booked\_on\_date;

delimiter $

create procedure car\_booked\_on\_date(in given\_date date)

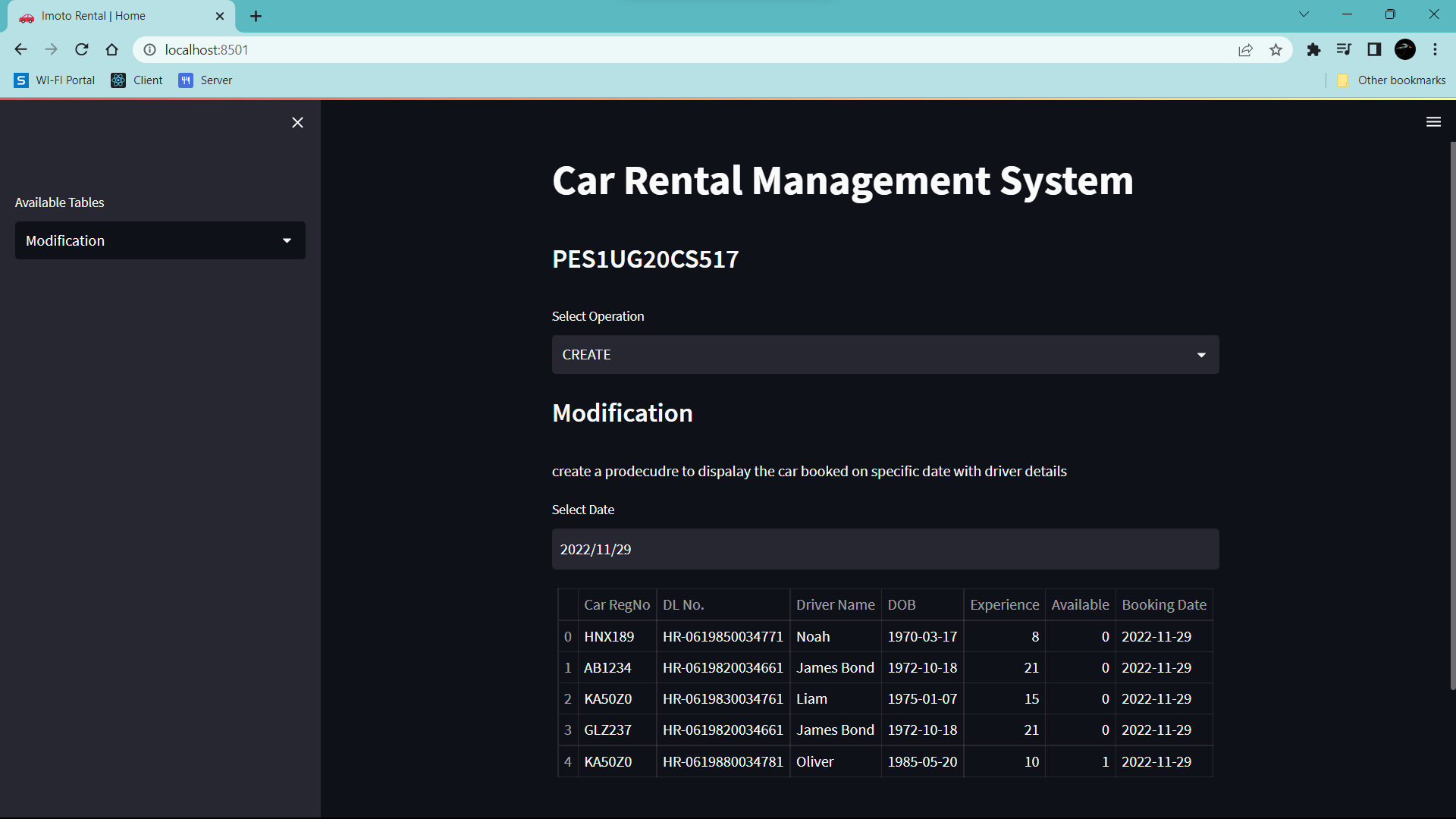
begin

    select B.car\_reg\_no, di.dl\_number,di.driver\_name, di.driver\_dob, di.driving\_experience, di.available, b.booking\_date

    from booking\_details as B inner join booking\_with\_driver as D on B.booking\_id = D.booking\_id inner join driver\_info as di on D.driver\_dl = di.dl\_number where b.booking\_date = given\_date;

end $

delimiter ;

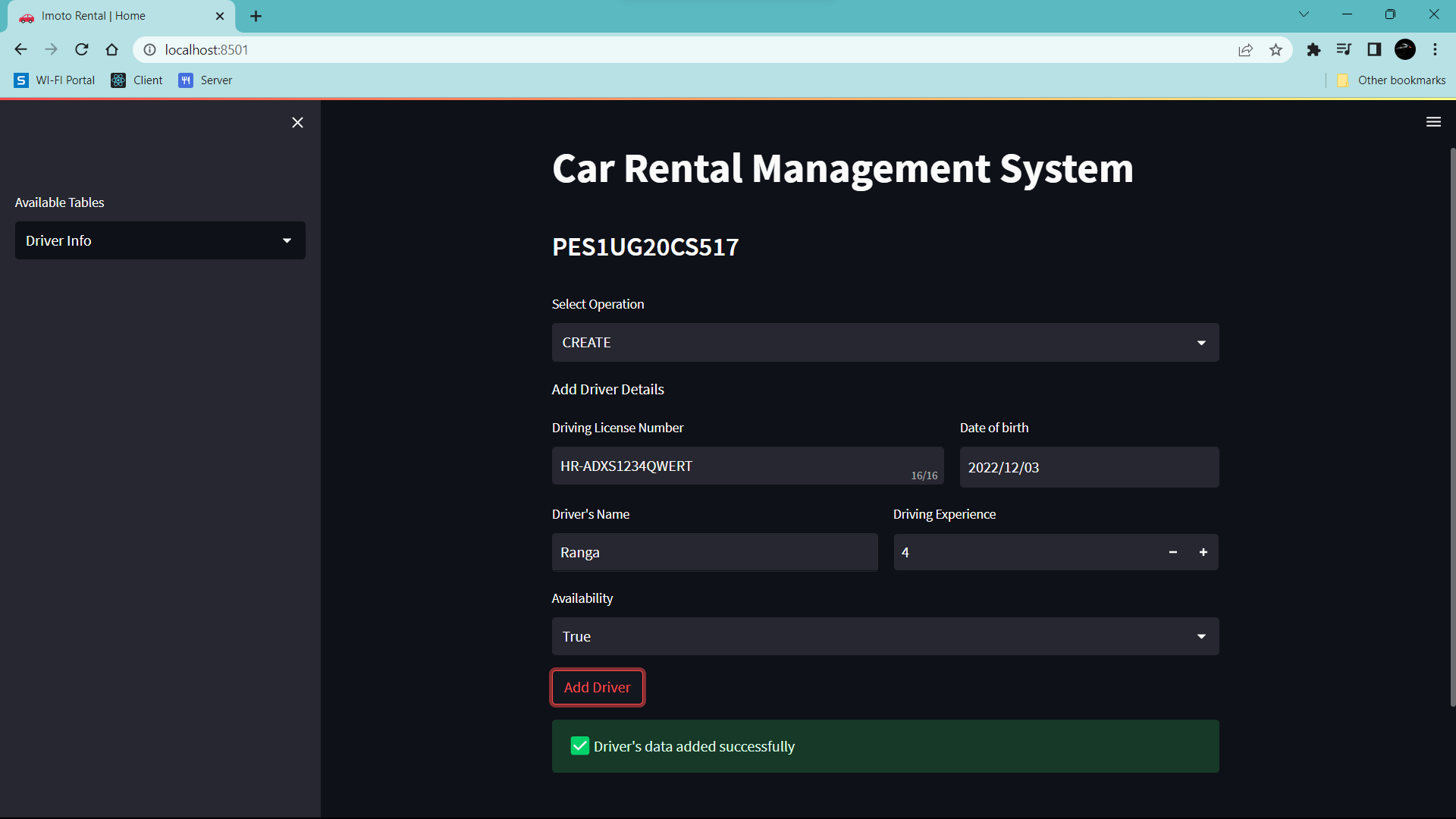


# Developing a Frontend

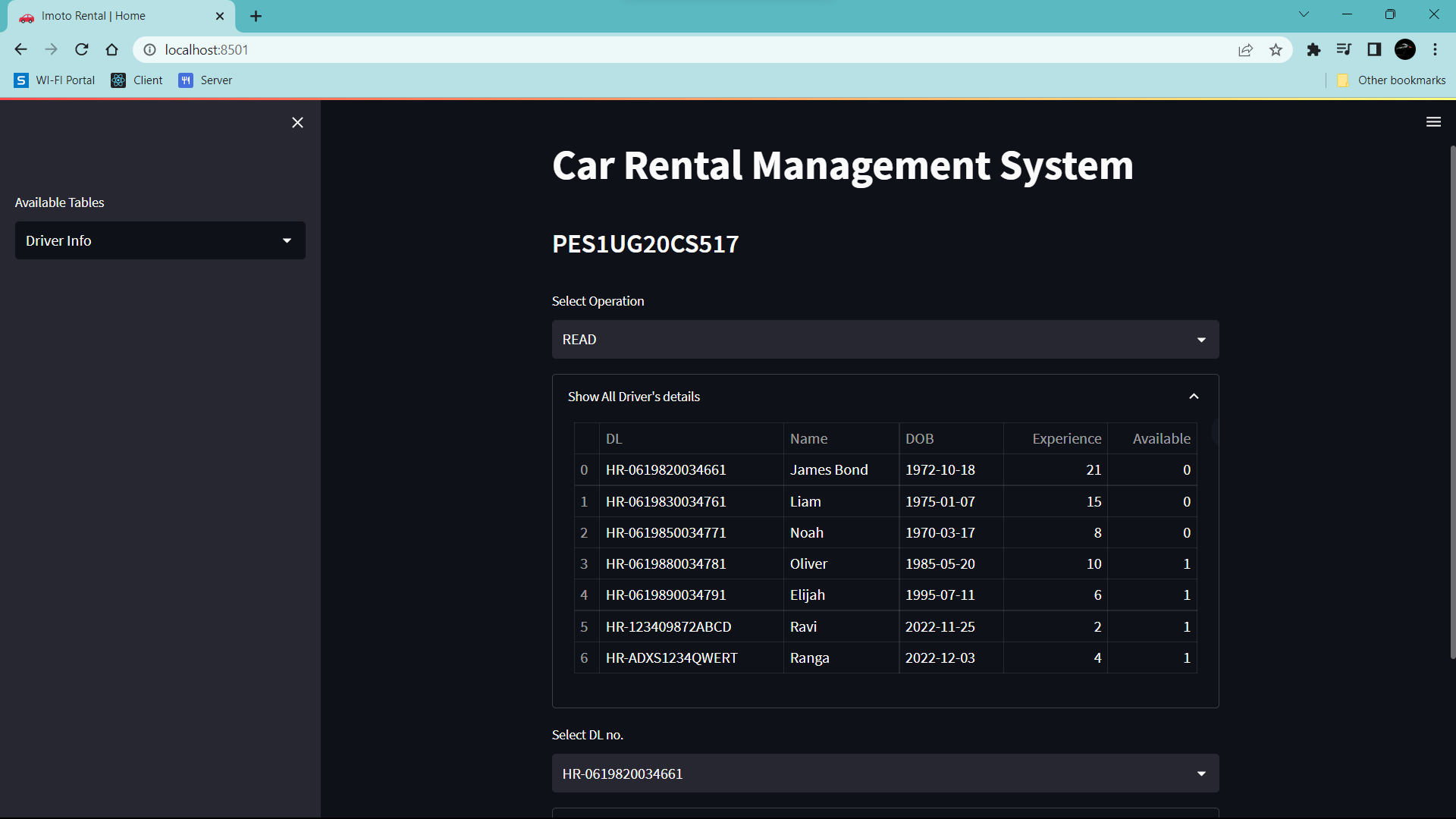
The frontend should support

1. Addition, Modification and Deletion of records from any chosen table

**Create**:

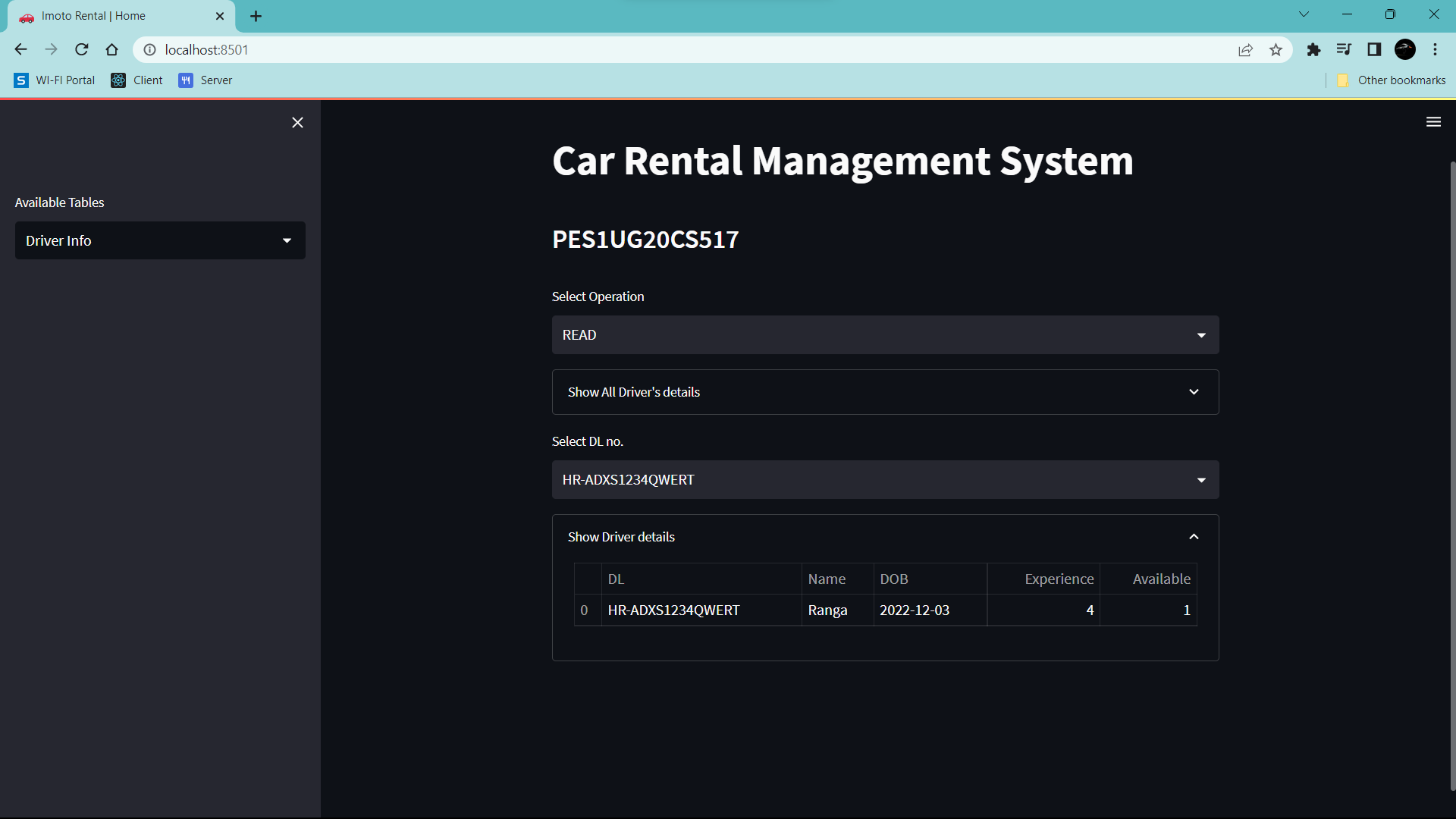


**Read:**

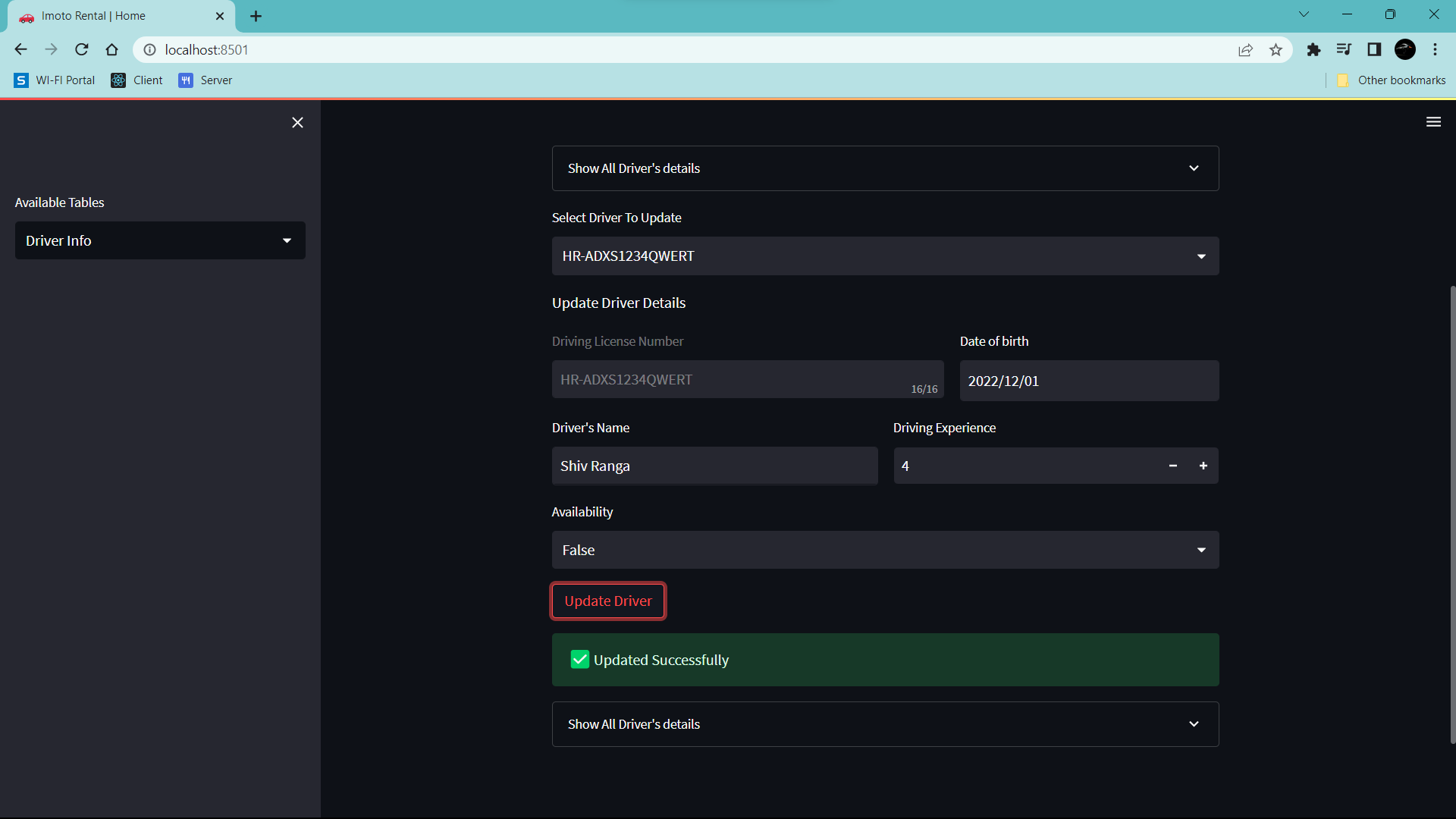
****

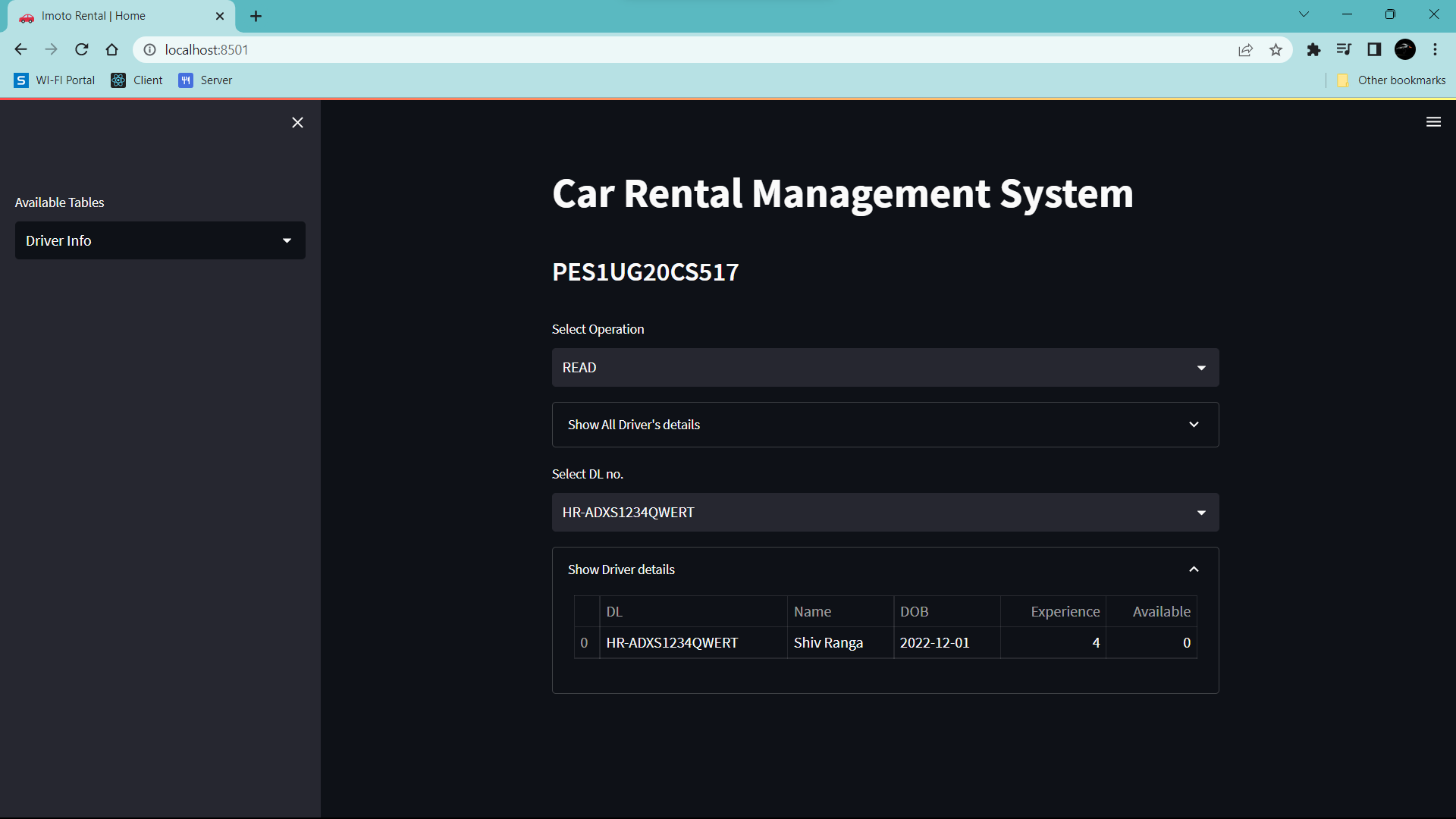
**Update:**

**Before Update:**

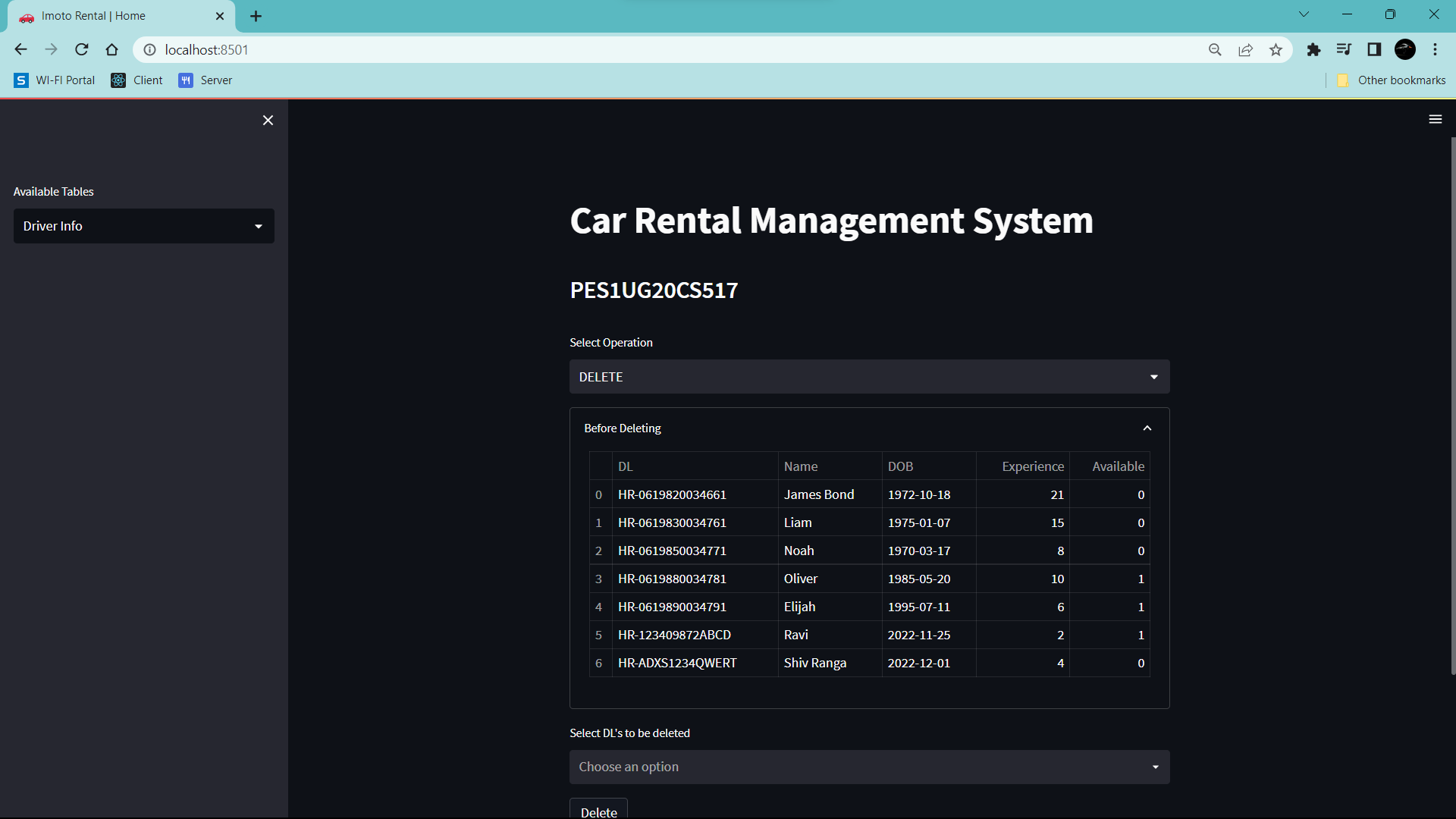
****

**After Update:**

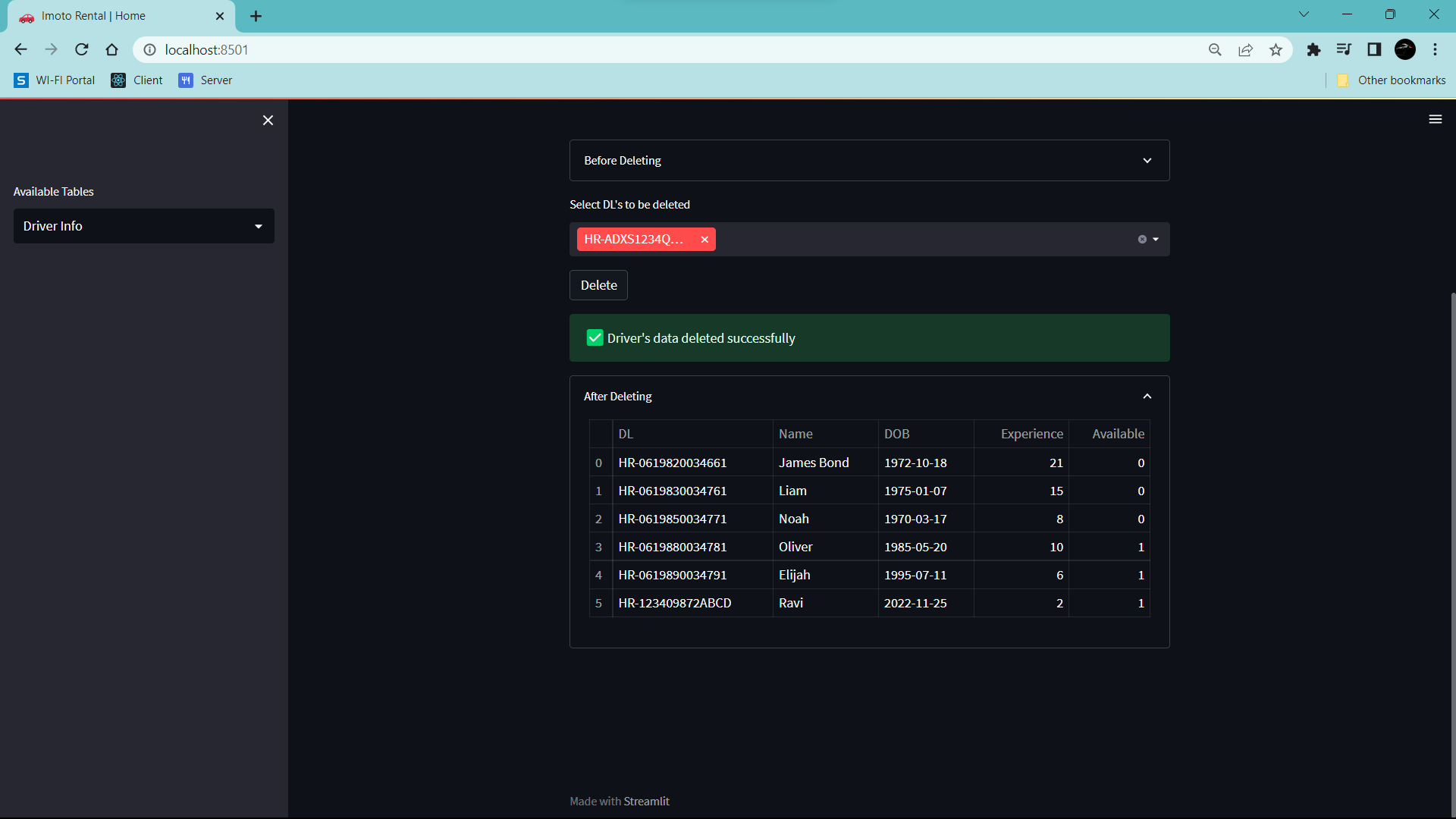
****

****

Before Delete:



After Delete:



1. There should be an window to accept and run any SQL statement and display the result

