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:

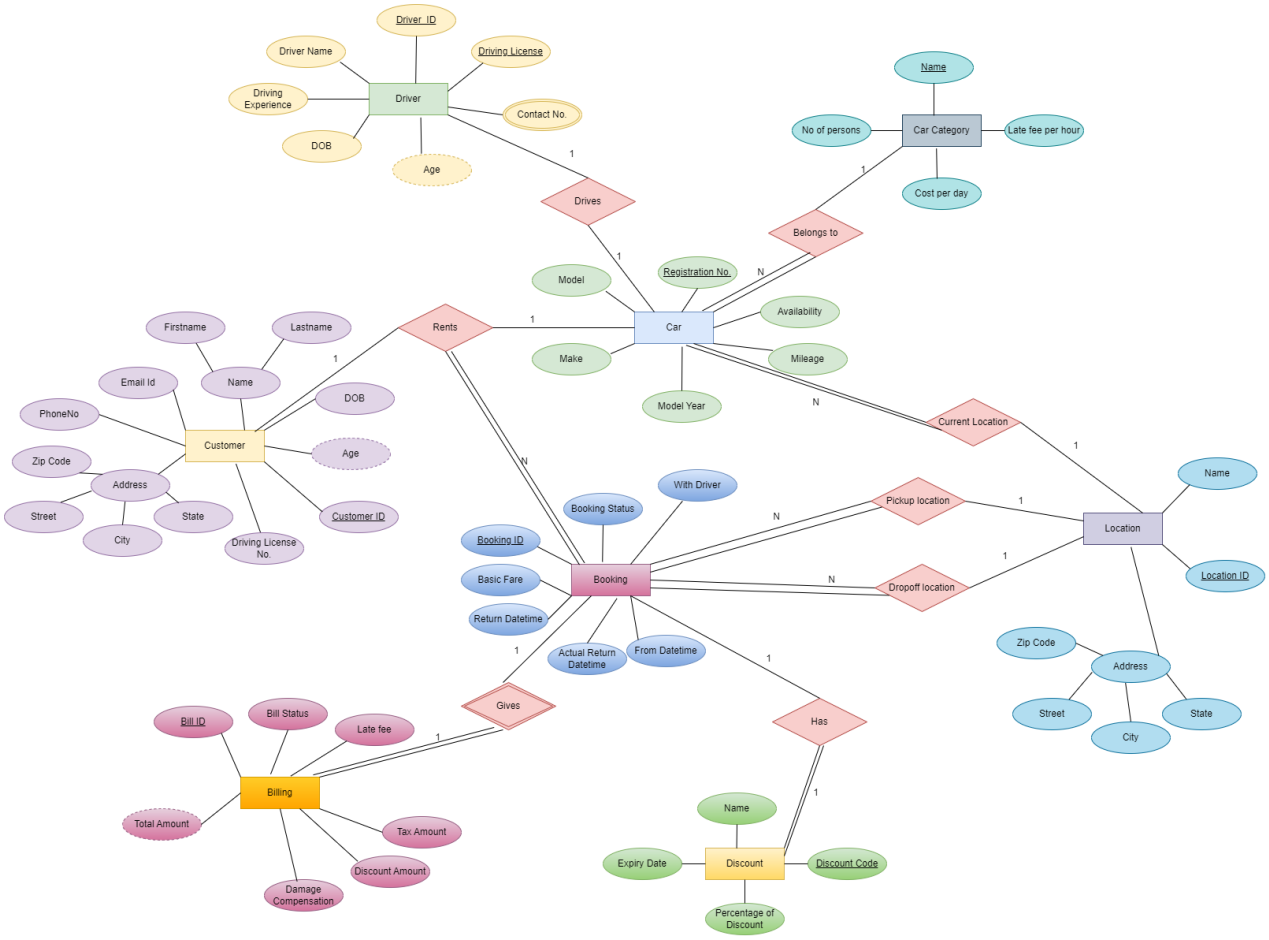
- Renting a car is a self-sustaining system which is the best opportunity for the people who cannot afford to buy the car in their family.
- Due to this system the people can borrow the car on rent for some time and do they compulsory work while paying the charges the rented car.
- Making a company available 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

Name: Y Srinivas

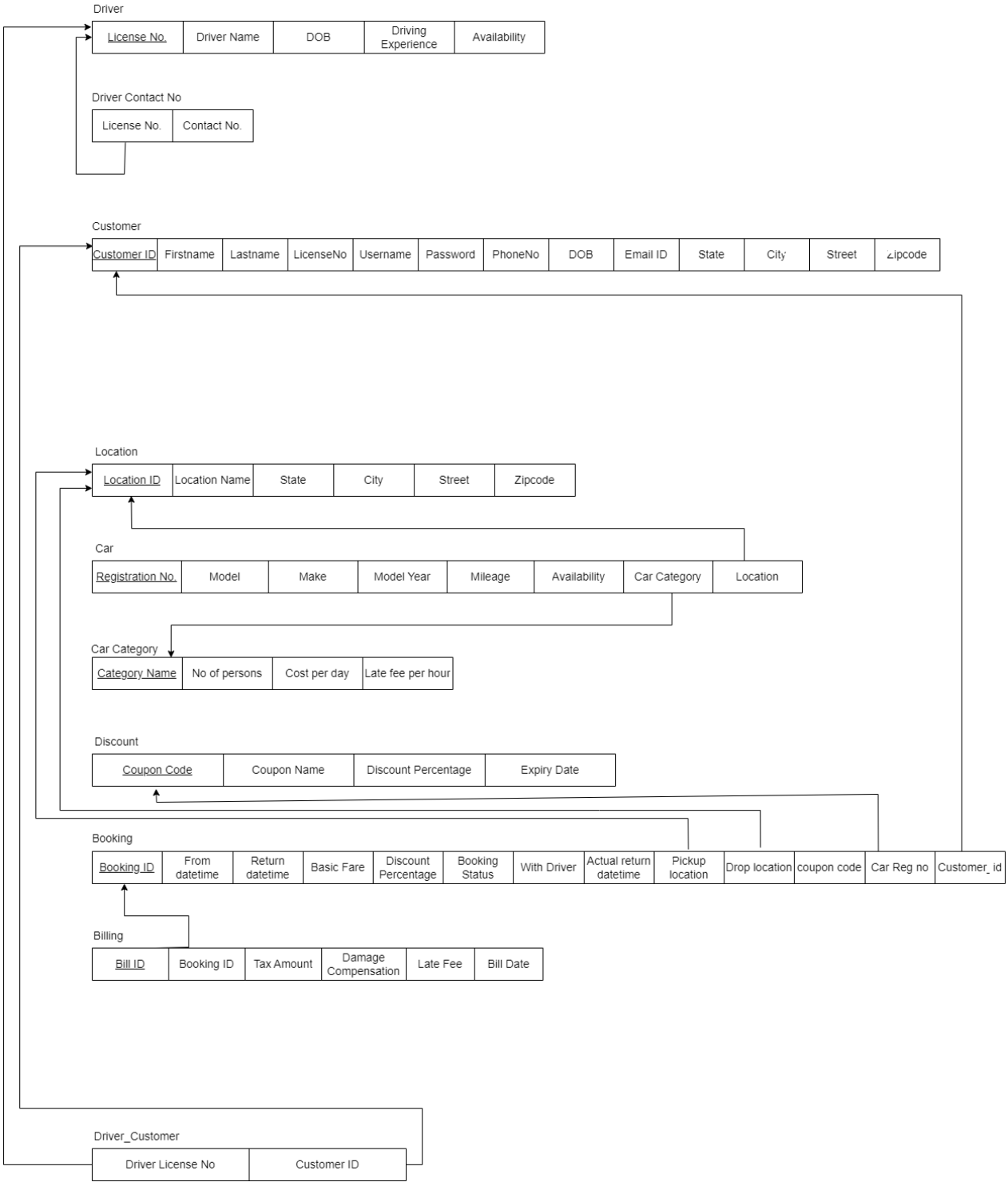
SRN: PES1UG20CS517

Mini Project Title: Car Rental Management System



Relational Schema

Car Rental Management System



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;
```

```sql
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)
);
```

```sql
-- 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;

	registration_no	model_name	make	model_year	mileage	available	category	car_location	category_name	no_persons	cost_per_day	late_fee_per_hour
0	GLZ237	COROLLA	TOYOTA	2016	5.0000	1	ECONOMY	40002	ECONOMY	5	30.0000	0.9000
1	KA50Z0	Vitara Brezza	Maruthi Suzuki	2017	9.0000	1	STANDARD SUV	40008	STANDARD SUV	5	40.0000	1.2000
2	GLS762	FOCUS	FORD	2014	12.0100	1	COMPACT	40001	COMPACT	6	390.0000	50.0000
3	HNX189	PRIUS	TOYOTA	2015	7.8000	1	COMPACT	40003	COMPACT	6	390.0000	50.0000

2. Show car details along with its current location.
   Query:
   select * from car_details inner join locations on car_location = location_id;

	registration_no	model_name	make	model_year	mileage	available	category	car_location	location_id	location_name	state	city	street	zipcode
0	AB1234	CIVIC	HONDA	2014	8.0000	0	ECONOMY	40000	40000	DALLAS LOVE FIELD AIRPORT	Alabama	Montgomery	Main Street.	40202
1	FKD820	GOLF	VOLKSWAGAN	2016	11.5000	0	COMPACT	40002	40002	DALLAS/ FORT WORTH INTL AIRPORT	Alabama	Montgomery	Main Street.	40202
2	GLS762	FOCUS	FORD	2014	12.0100	1	COMPACT	40001	40001	LOS ANGELES INTL AIRPORT	Alaska	Juneau	2nd Street.	40203
3	GLZ237	COROLLA	TOYOTA	2016	5.0000	1	ECONOMY	40002	40002	DALLAS/ FORT WORTH INTL AIRPORT	Alabama	Montgomery	Main Street.	40202
4	HNX189	PRIUS	TOYOTA	2015	7.8000	1	COMPACT	40003	40003	WEST HOUSTON AIRPORT	Arizona	Phoenix	7th Street.	40204
5	KA50Z0	Vitara Brezza	Maruthi Suzuki	2017	9.0000	1	STANDARD SUV	40008	40008	Shivakote	Karnataka	Banglore	Manjunatha Restaurant Road	56008
6	SD4567	FIESTA	FORD	2015	6.0000	0	ECONOMY	40001	40001	LOS ANGELES INTL AIRPORT	Alaska	Juneau	2nd Street.	40203
7	SDL935	FOCUS	FORD	2016	10.0000	0	COMPACT	40003	40003	WEST HOUSTON AIRPORT	Arizona	Phoenix	7th Street.	40204

3. 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;

Enter MySQL 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;
```

Execute Query

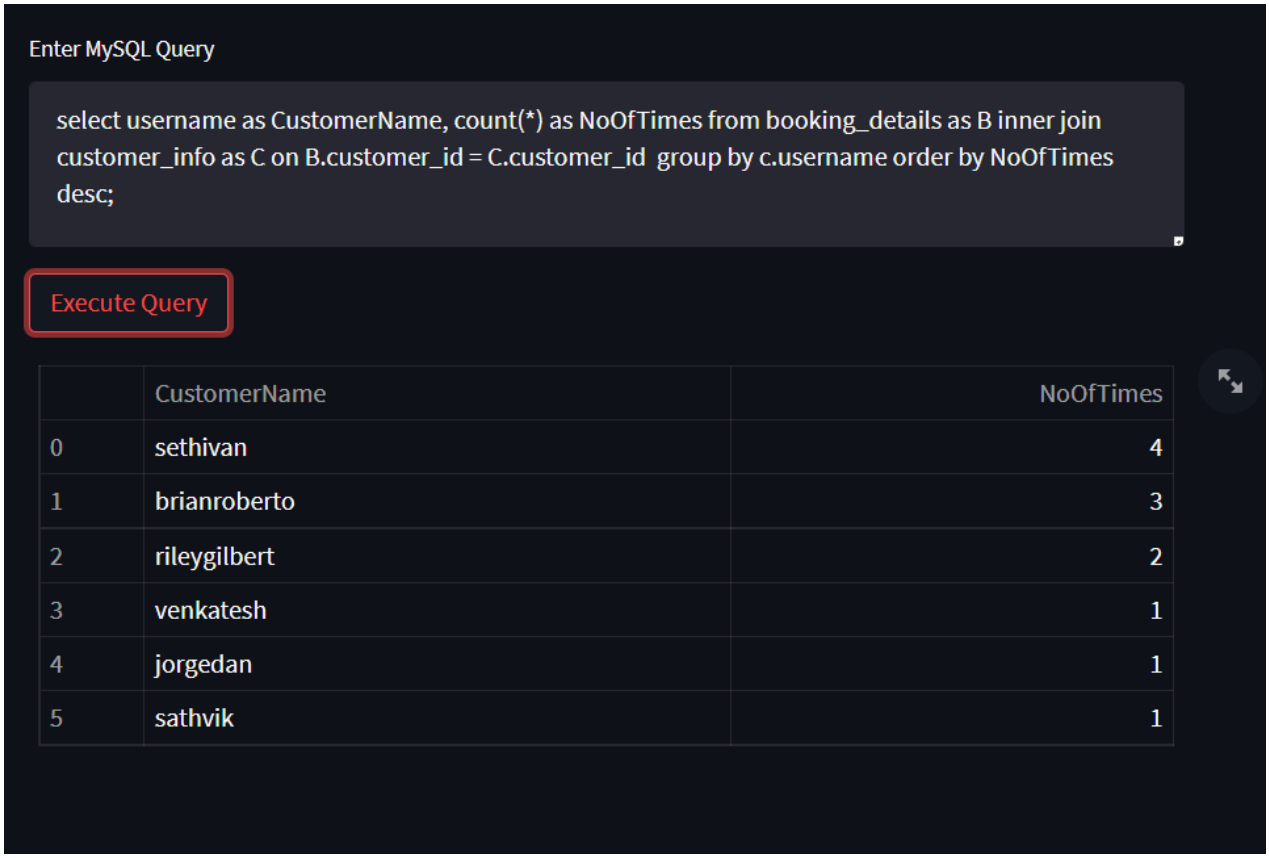
	Firstname	Lastname	model_name
0	Riley	Gilbert	COROLLA
1	Riley	Gilbert	FIESTA
2	Jorge	Dan	FOCUS
3	Brian	Roberto	PRIUS
4	Sathvik	A V	CIVIC
5	Seth	Ivan	Vitara Brezza
6	Seth	Ivan	COROLLA
7	Venkatesh	B R	Vitara Brezza
8	Seth	Ivan	COROLLA
9	Brian	Roberto	COROLLA
10	Brian	Roberto	COROLLA
11	Seth	Ivan	CIVIC

4. 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;

	registration_no	model_name	make	model_year	mileage	available	category	car_location	category_name	no_persons	cost_per_day	late_fee_per_hour
0	AB1234	CIVIC	HONDA	2014	8.0000	0	ECONOMY	40000	ECONOMY	5	30.0000	0.9000
1	GLZ237	COROLLA	TOYOTA	2016	5.0000	1	ECONOMY	40002	ECONOMY	5	30.0000	0.9000
2	SD4567	FIESTA	FORD	2015	6.0000	0	ECONOMY	40001	ECONOMY	5	30.0000	0.9000
3	KA50Z0	Vitara Brezza	Maruthi Suzuki	2017	9.0000	1	STANDARD SUV	40008	STANDARD SUV	5	40.0000	1.2000
4	SDL935	FOCUS	FORD	2016	10.0000	0	COMPACT	40003	COMPACT	6	390.0000	50.0000
5	FKD820	GOLF	VOLKSWAGAN	2016	11.5000	0	COMPACT	40002	COMPACT	6	390.0000	50.0000
6	HNX189	PRIUS	TOYOTA	2015	7.8000	1	COMPACT	40003	COMPACT	6	390.0000	50.0000
7	GLS762	FOCUS	FORD	2014	12.0100	1	COMPACT	40001	COMPACT	6	390.0000	50.0000

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;

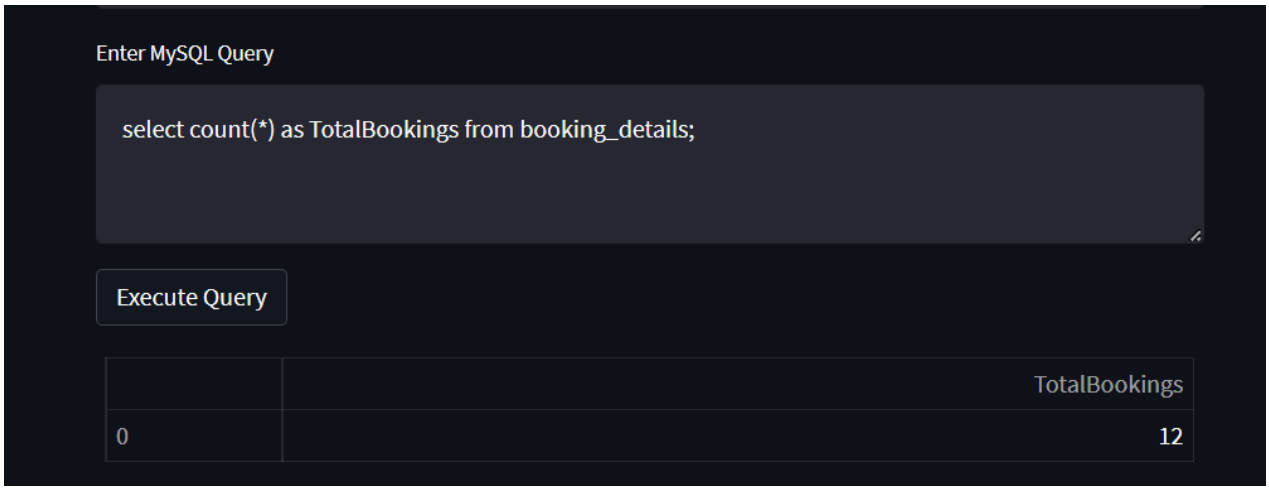


	CustomerName	NoOfTimes
0	sethivan	4
1	brianroberto	3
2	rileygilbert	2
3	venkatesh	1
4	jorgedan	1
5	sathvik	1

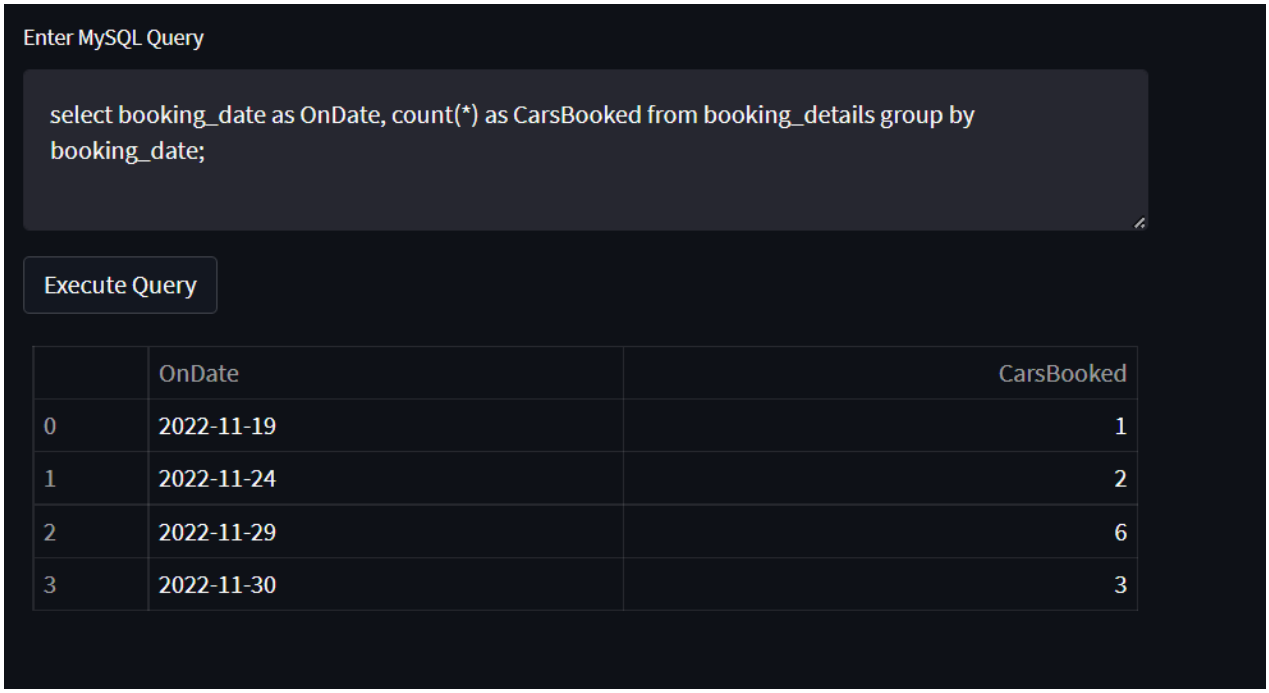
2. Show total number of bookings done till today.

   Query:

   select count(*) as TotalBookings from booking_details;



3. Show number of cars booked on particular date.
   Query:
   select booking_date as OnDate, count(*) as CarsBooked from booking_details group by booking_date;



	OnDate	CarsBooked
0	2022-11-19	1
1	2022-11-24	2
2	2022-11-29	6
3	2022-11-30	3

4. Show number of cars belong to each category.
   Query:
   select category, count(*) as NoOfCars from car_details group by category;

Enter MySQL Query

```
select category, count(*) as NoOfCars from car_details group by category;
```

Execute Query

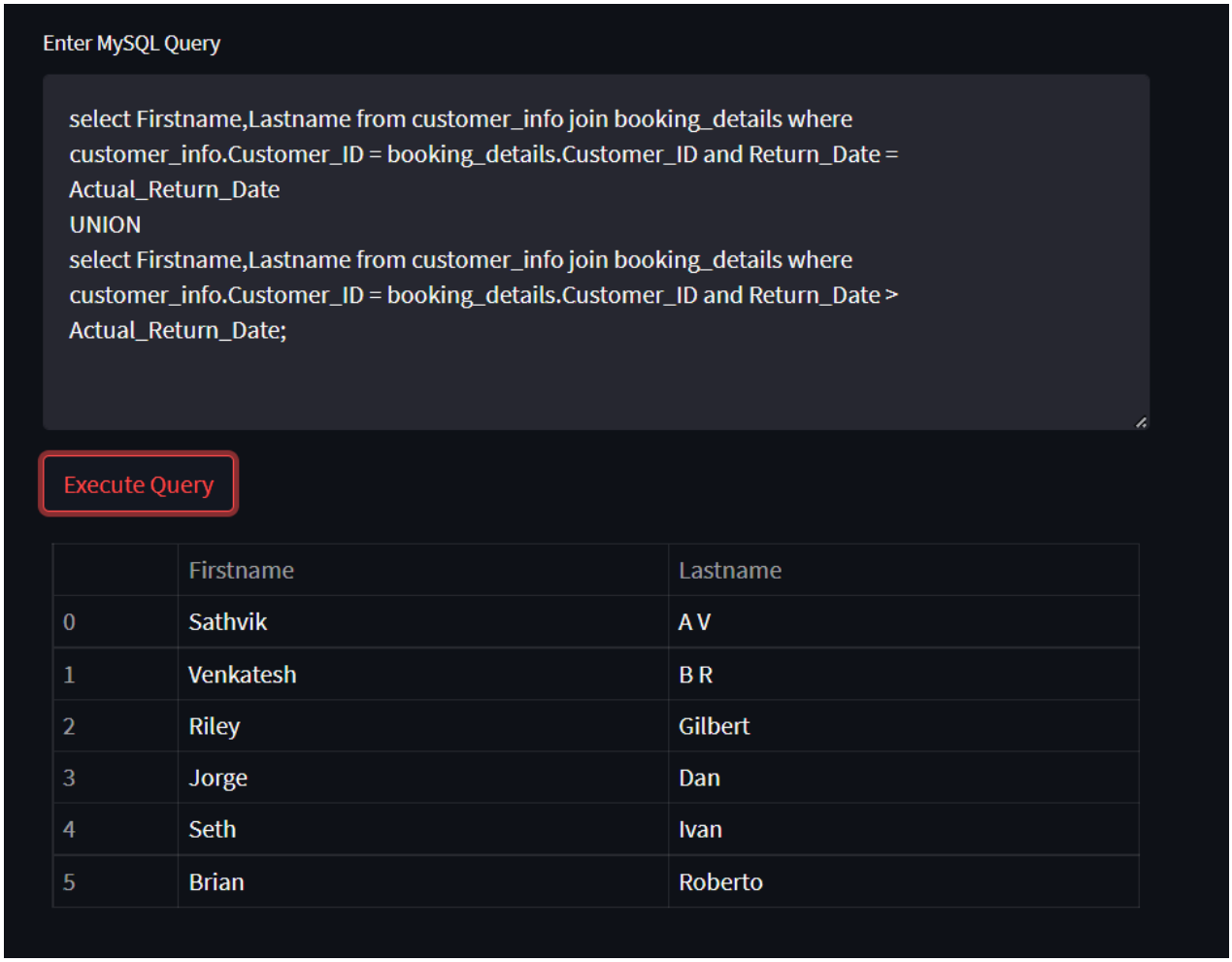
	category	NoOfCars
0	COMPACT	4
1	ECONOMY	3
2	STANDARD SUV	1

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;
   ```



	Firstname	Lastname
0	Sathvik	A V
1	Venkatesh	B R
2	Riley	Gilbert
3	Jorge	Dan
4	Seth	Ivan
5	Brian	Roberto

2. **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;
```

Enter MySQL 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;
```

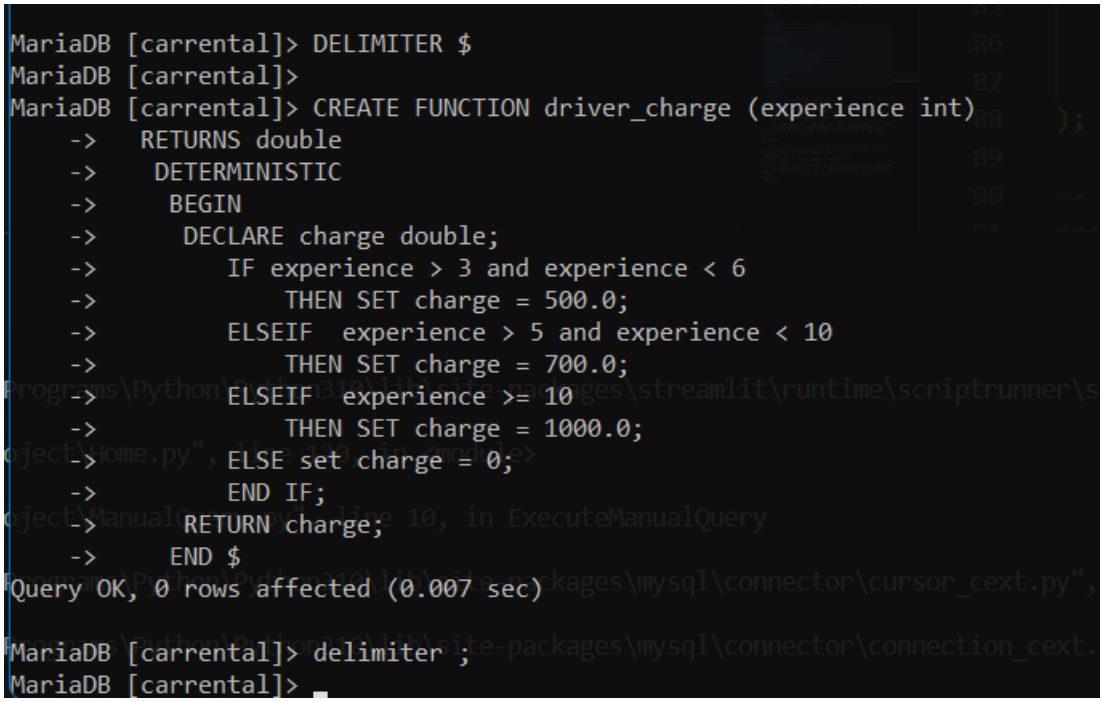
Execute Query

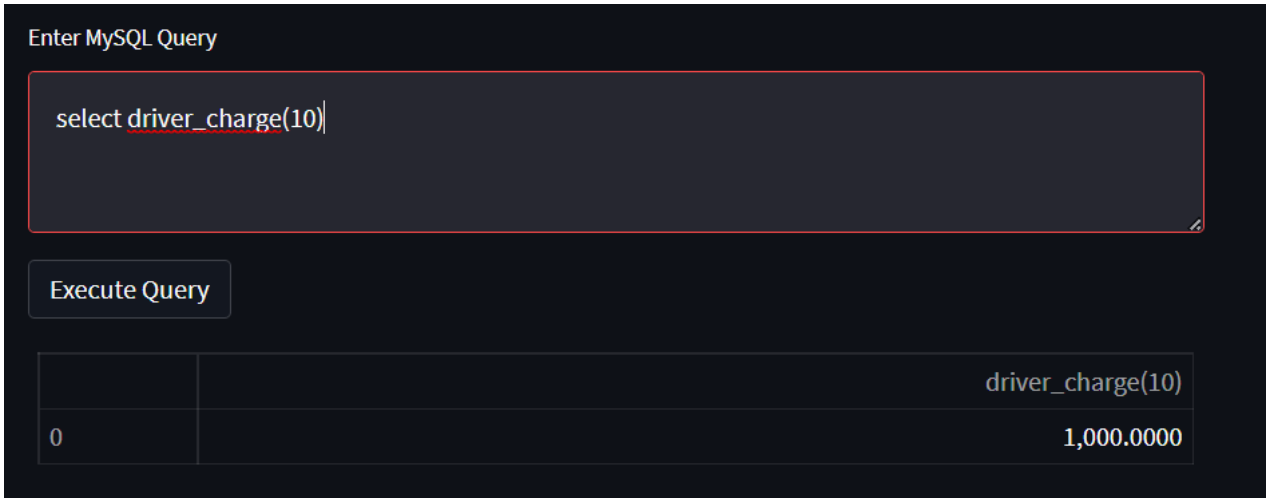
	Make	Model_name	Model_Year
0	TOYOTA	COROLLA	2016
1	TOYOTA	PRIUS	2015
2	Maruthi Suzuki	Vitara Brezza	2017
3	FORD	FIESTA	2015

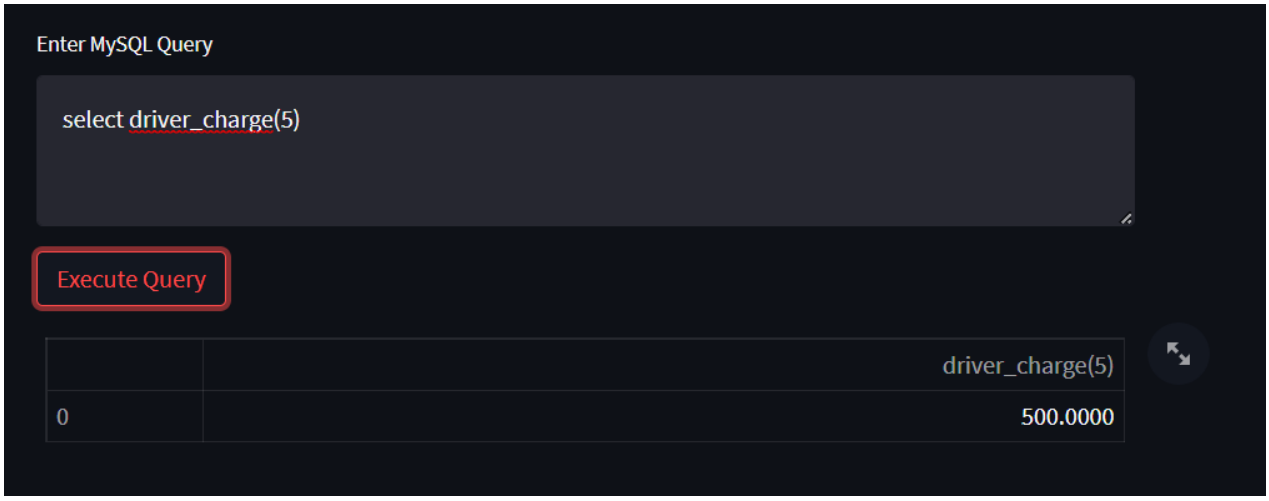
Functions and Procedures

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 ;

```
MariaDB [carrental]> drop procedure if exists get_total_amount;
Query OK, 0 rows affected (0.012 sec)

MariaDB [carrental]> delimiter $
MariaDB [carrental]>
MariaDB [carrental]> 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 $
Query OK, 0 rows affected (0.010 sec)

MariaDB [carrental]>
MariaDB [carrental]> delimiter ;
MariaDB [carrental]>
```

```
MariaDB [carrental]> call get_total_amount(50024, @M);
Query OK, 2 rows affected (0.005 sec)

MariaDB [carrental]> select @M;
+-------+
| @M    |
+-------+
| 219.7 |
+-------+
1 row in set (0.000 sec)

MariaDB [carrental]>
```

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 ;
```

```
MariaDB [carrental]> delimiter ;
MariaDB [carrental]> drop trigger if exists after_driver_customer;
Query OK, 0 rows affected (0.013 sec)

MariaDB [carrental]> delimiter $$
MariaDB [carrental]> create trigger after_driver_customer
    -> after insert
    -> on booking_with_driver for each row
    -> begin
    ->
    -> update driver_info set available = false where dl_number = new.driver_dl;
    ->
    -> end $$
Query OK, 0 rows affected (0.015 sec)

MariaDB [carrental]> delimiter ;
MariaDB [carrental]>
```

Initial available drivers.

```
MariaDB [carrental]> select * from driver_info where available = true;
+------------------+-------------+------------+--------------------+-----------+
| dl_number        | driver_name | driver_dob | driving_experience | available |
+------------------+-------------+------------+--------------------+-----------+
| HR-0619850034771 | Noah        | 1970-03-17 |                  8 |         1 |
| HR-0619880034781 | Oliver      | 1985-05-20 |                 10 |         1 |
| HR-0619890034791 | Elijah      | 1995-07-11 |                  6 |         1 |
| HR-123409872ABCD | Ravi        | 2022-11-25 |                  2 |         1 |
+------------------+-------------+------------+--------------------+-----------+
4 rows in set (0.001 sec)
```

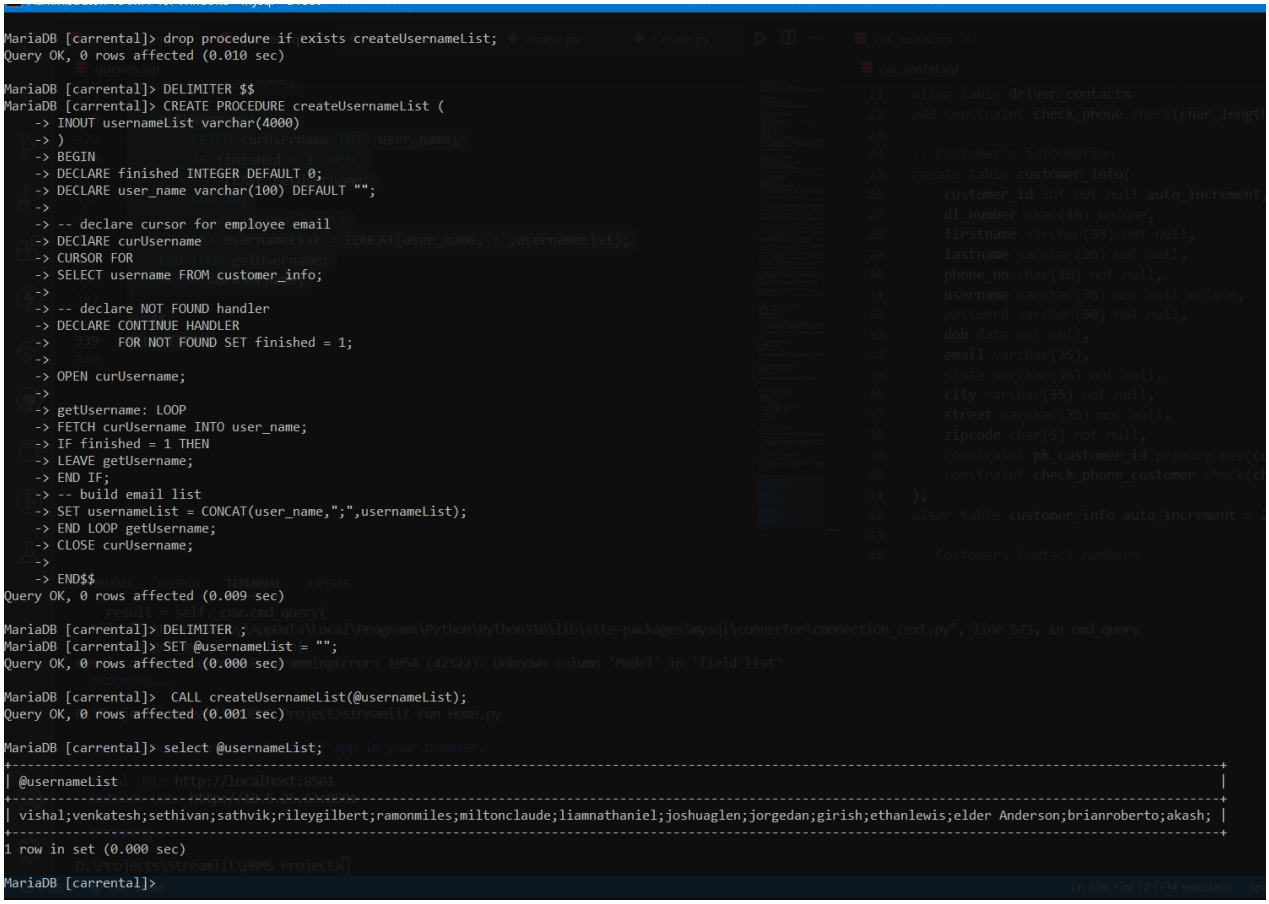
After Booking with driver.

```
MariaDB [carrental]> select * from driver_info where available = true;
+------------------+-------------+------------+--------------------+-----------+
| dl_number        | driver_name | driver_dob | driving_experience | available |
+------------------+-------------+------------+--------------------+-----------+
| HR-0619880034781 | Oliver      | 1985-05-20 |                 10 |         1 |
| HR-0619890034791 | Elijah      | 1995-07-11 |                  6 |         1 |
| HR-123409872ABCD | Ravi        | 2022-11-25 |                  2 |         1 |
+------------------+-------------+------------+--------------------+-----------+
3 rows in set (0.000 sec)

MariaDB [carrental]>
```

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

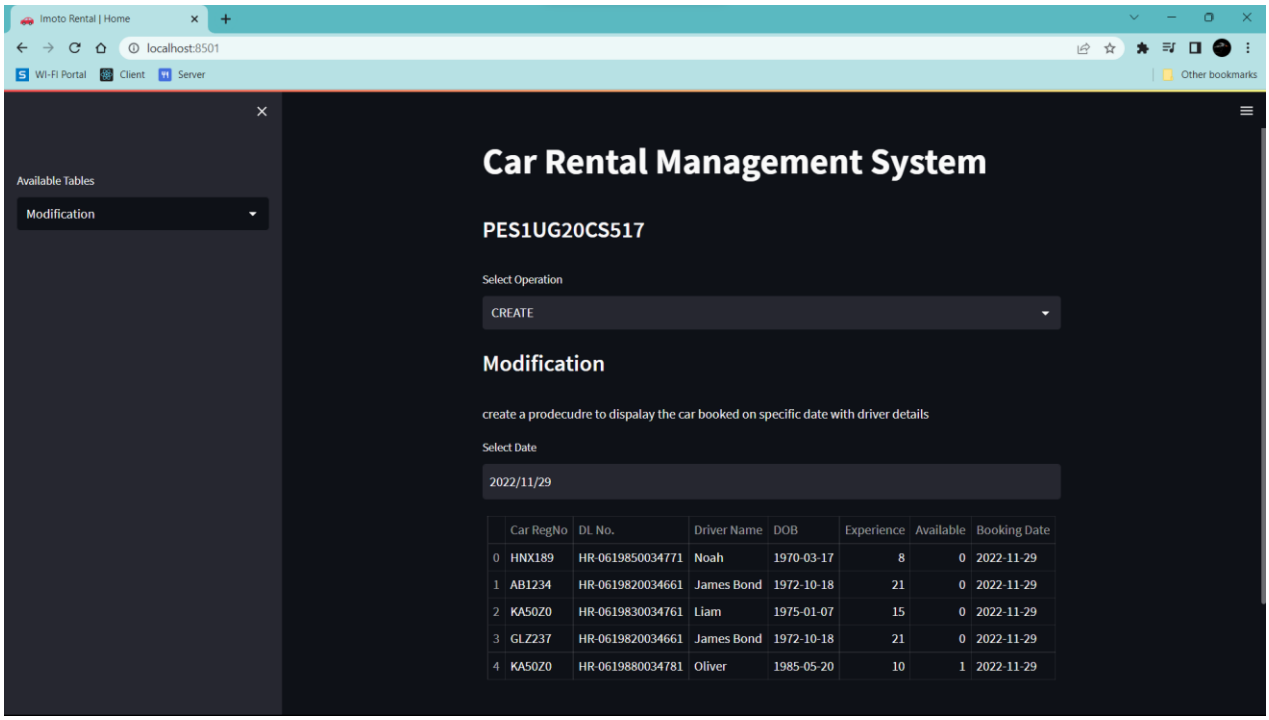
```
drop procedure if exists createUsernameList;
DELIMITER $$
CREATE PROCEDURE createUsernameList (
    INOUT usernameList varchar(4000)
)
BEGIN
    DECLARE finished INTEGER DEFAULT 0;
    DECLARE username varchar(100) DEFAULT "";

    -- 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 ;



```
MariaDB [carrental]> drop procedure if exists createUsernameList;
Query OK, 0 rows affected (0.010 sec)

MariaDB [carrental]> DELIMITER $$
MariaDB [carrental]> CREATE PROCEDURE createUsernameList (
    -> INOUT usernameList varchar(4000)
    -> )
    -> BEGIN
    -> DECLARE finished INTEGER DEFAULT 0;
    -> DECLARE user_name varchar(100) DEFAULT "";
    ->
    -> -- declare cursor for employee email
    -> DECLARE curUsername
    -> CURSOR FOR
    -> SELECT username FROM customer_info;
    ->
    -> -- declare NOT FOUND handler
    -> DECLARE CONTINUE HANDLER
    ->     FOR NOT FOUND SET finished = 1;
    ->
    -> OPEN curUsername;
    ->
    -> getUsername: LOOP
    -> FETCH curUsername INTO user_name;
    -> IF finished = 1 THEN
    -> LEAVE getUsername;
    -> END IF;
    -> -- build email list
    -> SET usernameList = CONCAT(user_name,";",usernameList);
    -> END LOOP getUsername;
    -> CLOSE curUsername;
    ->
    -> END$$
Query OK, 0 rows affected (0.009 sec)

MariaDB [carrental]> DELIMITER ;
MariaDB [carrental]> SET @usernameList = "";
Query OK, 0 rows affected (0.000 sec)

MariaDB [carrental]>  CALL createUsernameList(@usernameList);
Query OK, 0 rows affected (0.001 sec)

MariaDB [carrental]> select @usernameList;
+-----------------------
```

delimiter ;

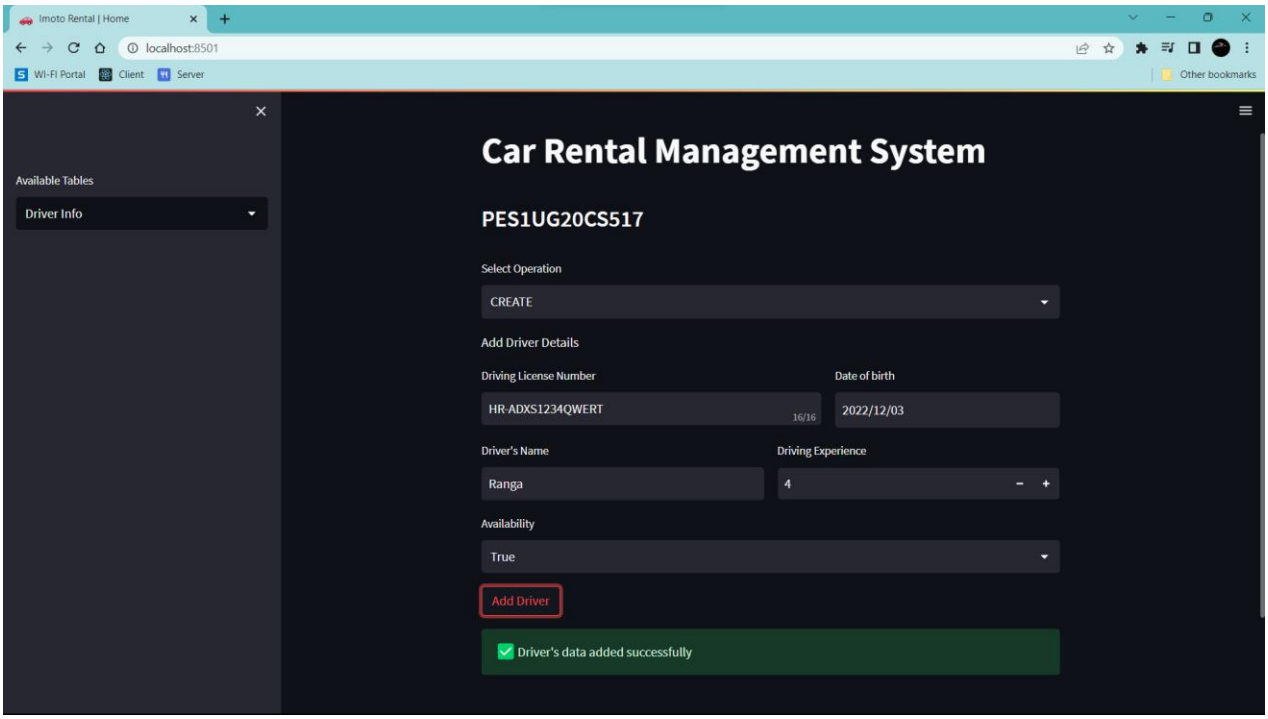

Imoto Rental | Home
localhost:8501
Wi-Fi Portal
Client
Server
Other bookmarks
Available Tables
Modification
Car Rental Management System
PES1UG20CS517
Select Operation
CREATE
Modification
create a prodecudre to dispalay the car booked on specific date with driver details
Select Date
2022/11/29
Car RegNo DL No. Driver Name DOB Experience Available Booking Date
0 HNX189 HR-0619850034771 Noah 1970-03-17 8 0 2022-11-29
1 AB1234 HR-0619820034661 James Bond 1972-10-18 21 0 2022-11-29
2 KA50Z0 HR-0619830034761 Liam 1975-01-07 15 0 2022-11-29
3 GLZ237 HR-0619820034661 James Bond 1972-10-18 21 0 2022-11-29
4 KA50Z0 HR-0619880034781 Oliver 1985-05-20 10 1 2022-11-29

Developing a Frontend

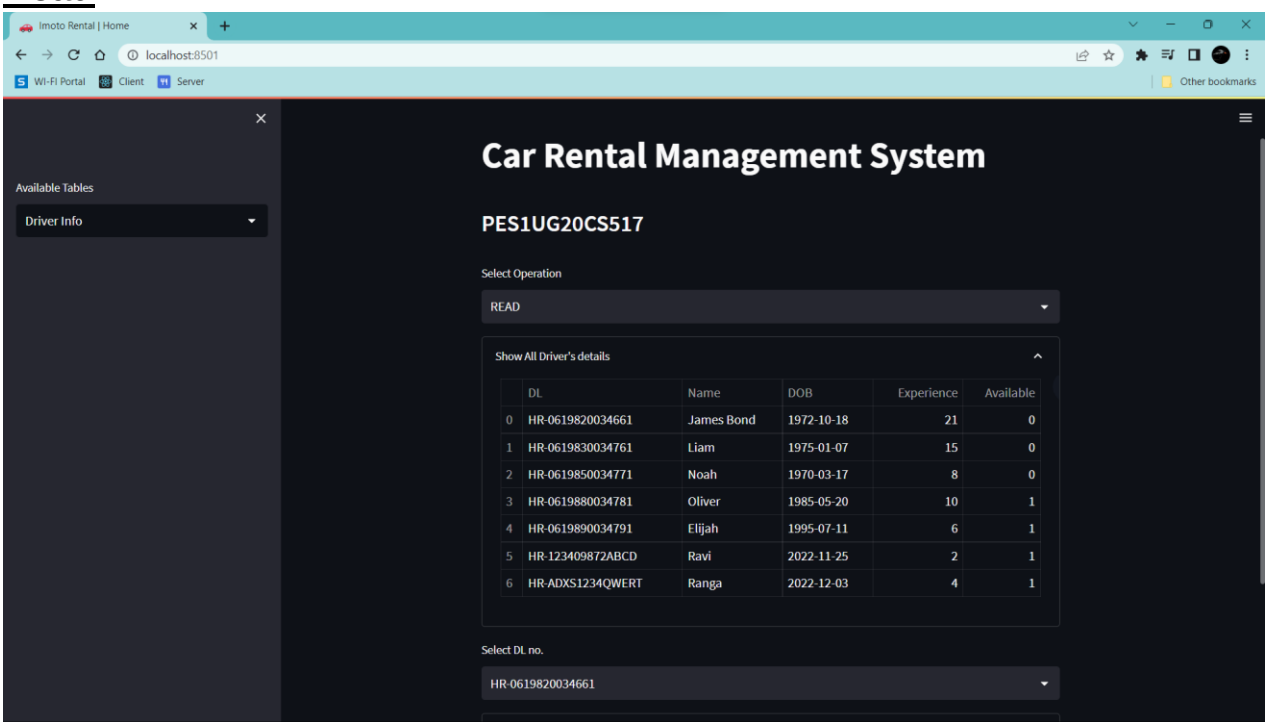
The frontend should support

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

Create:

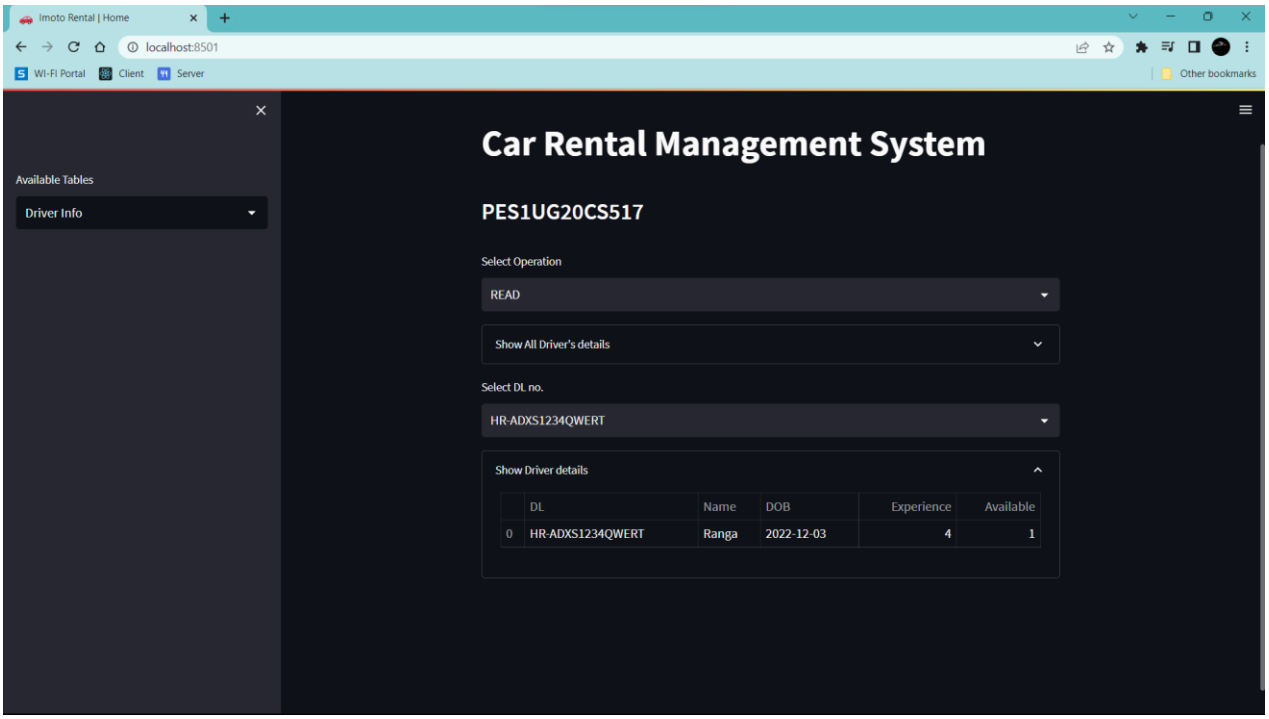


Read:

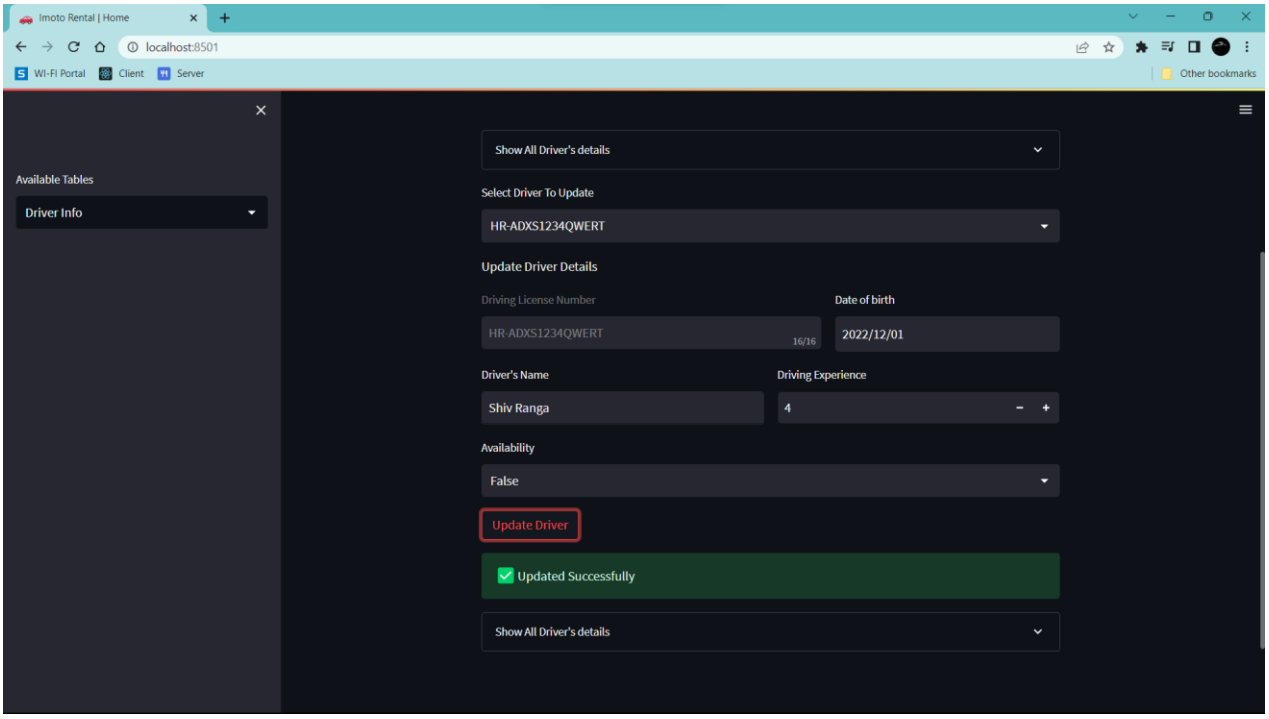


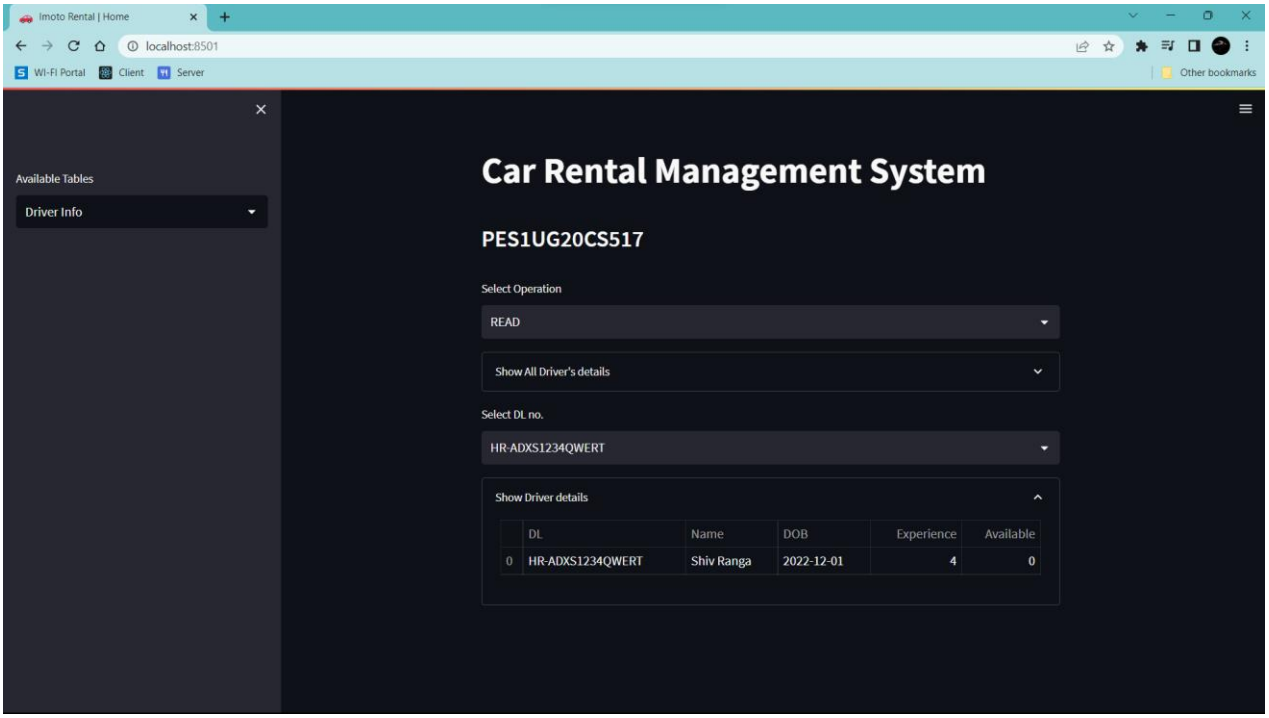
Update:

Before Update:

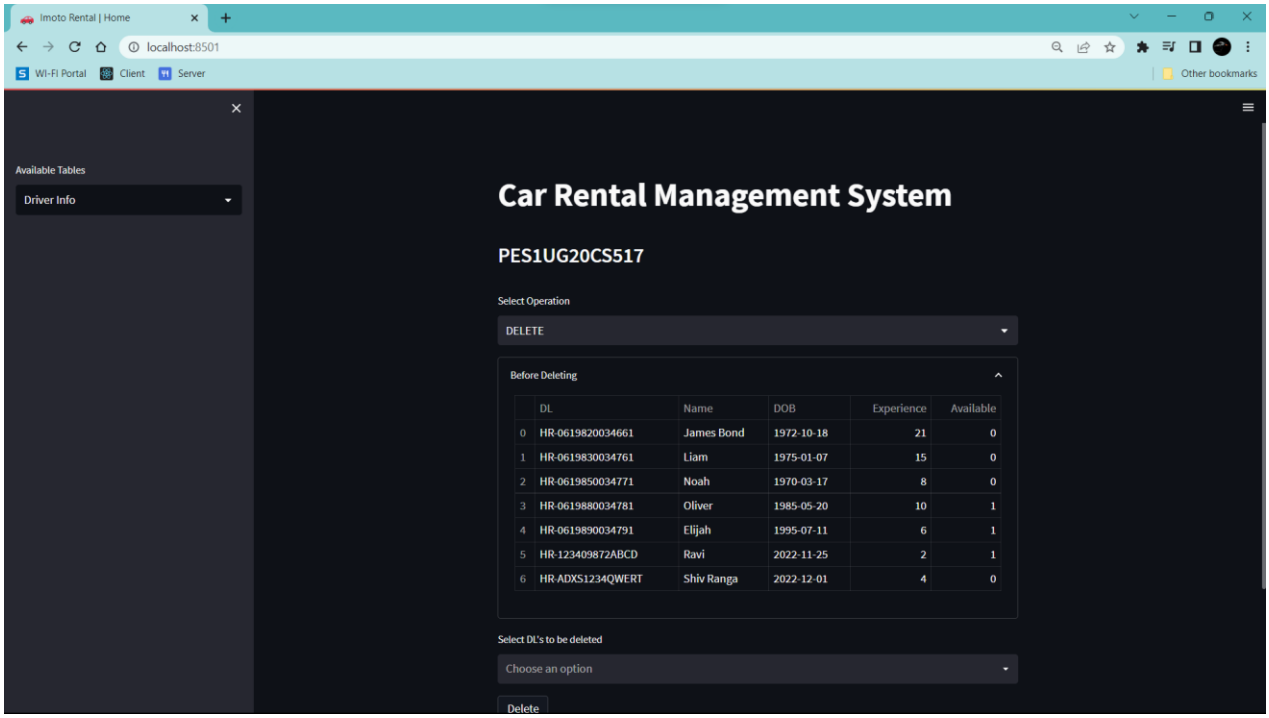


After Update:

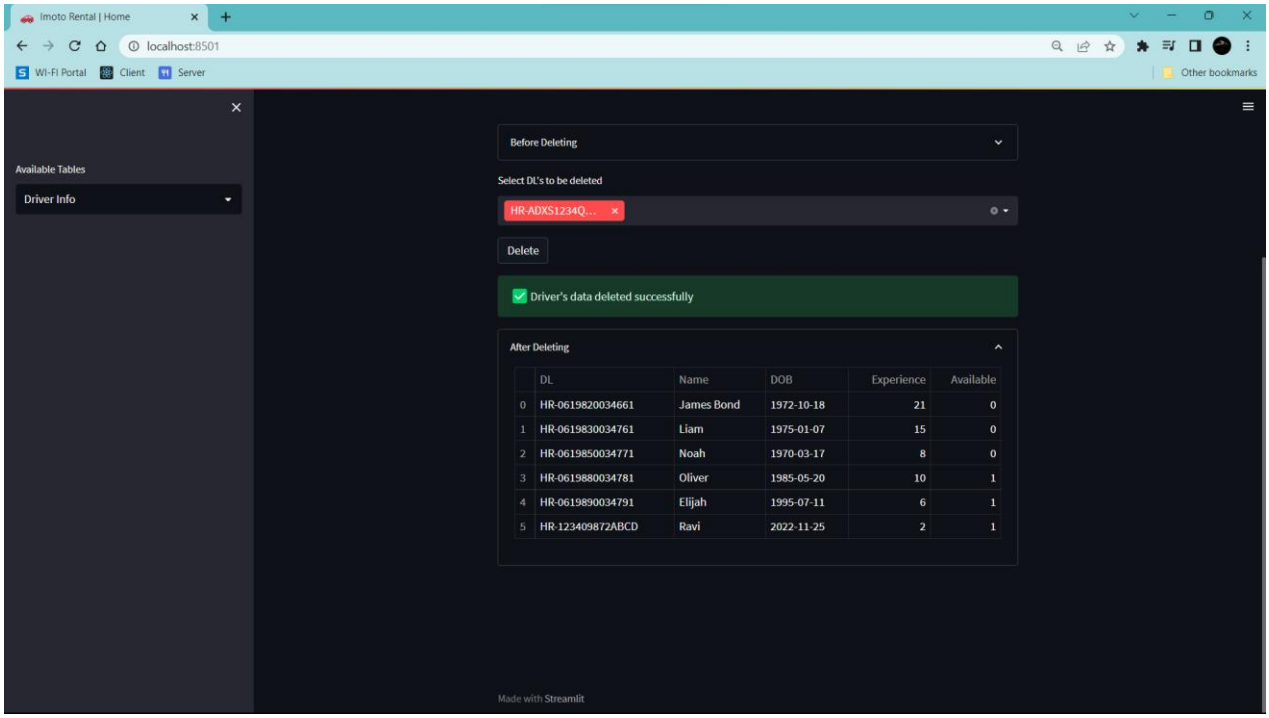
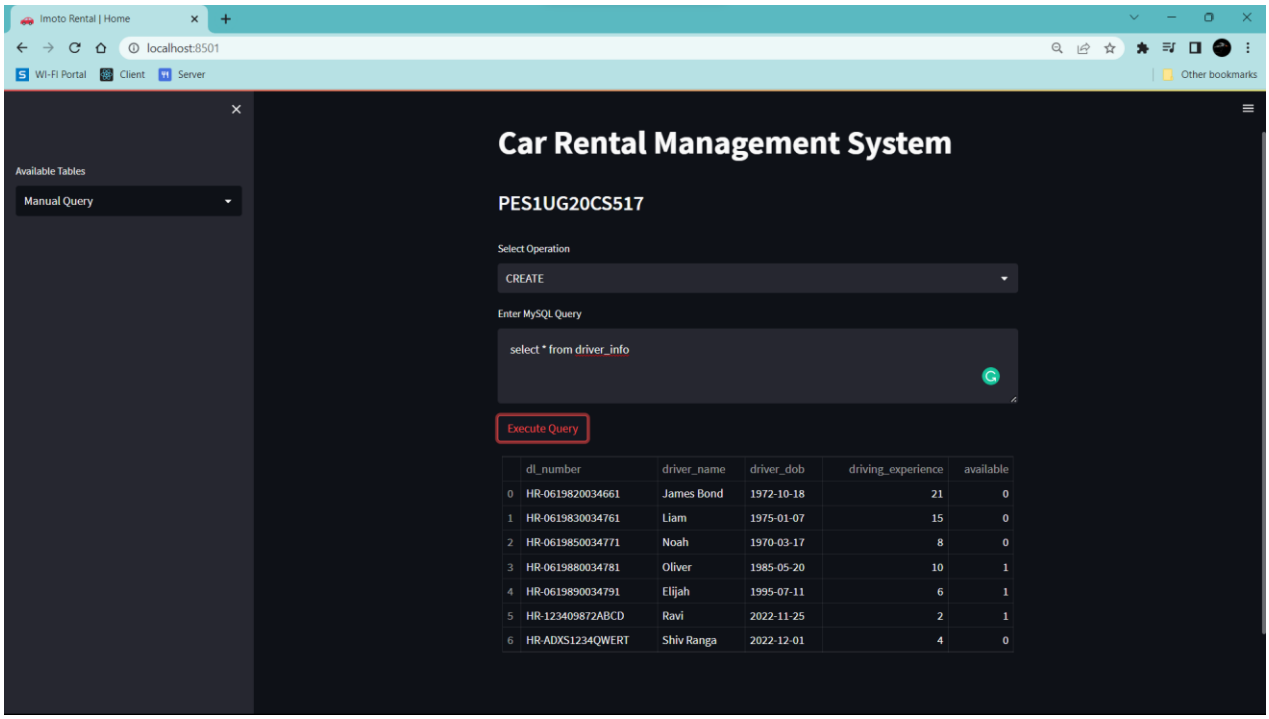




Before Delete:



After Delete:



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