DATABASE MANAGEMENT SYSTEM:

MINI PROJECT:

PROJECT TITLE:

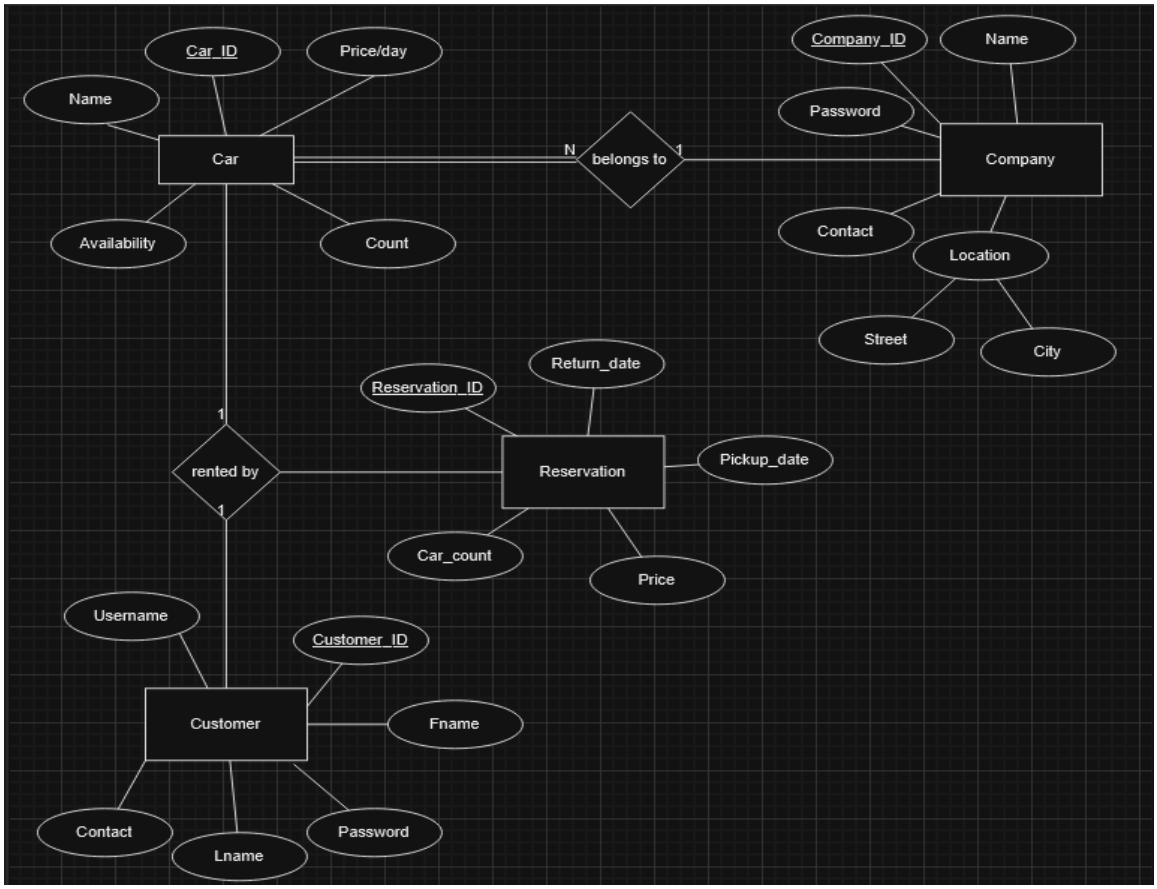
CAR RENTAL MANAGEMENT SYSTEM

TEAM MEMBERS:

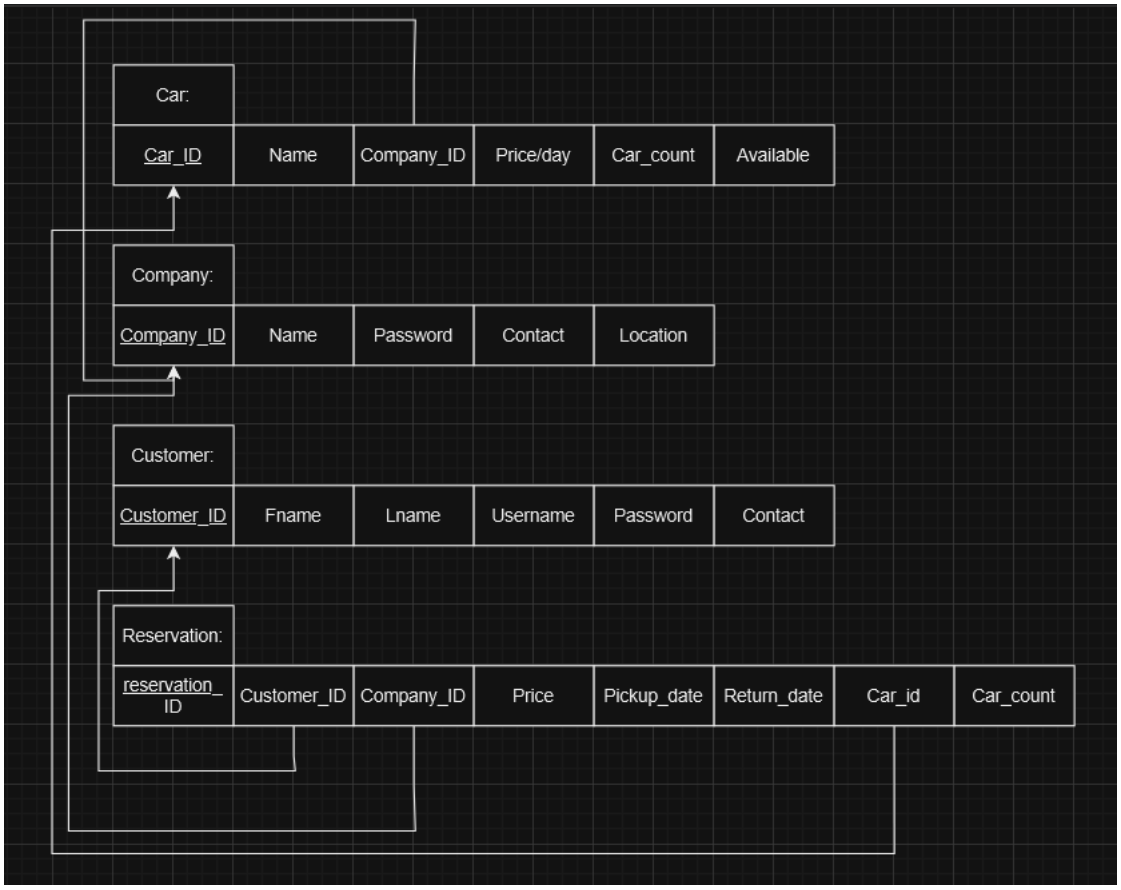
Suhas M: PES1UG21CS310

Lakshmeesh Bhat: PES1UG21CS298

ER DIAGRAM:


Car_ID
Price/day
Name
Car
N
belongs to
1
Company_ID
Name
Password
Company
Availability
Count
Contact
Location
Street
City
Return_date
Reservation_ID
1
rented by
1
Reservation
Pickup_date
Car_count
Price
Username
Customer_ID
Customer
Fname
Contact
Lname
Password

RELATIONAL SCHEMA:



USER REQUIREMENT SPECIFICATIONS:

1. Introduction:

The purpose of the project is to establish a platform for customers to rent cars from different companies for their needs.
The scope of the projects extends only to customers and companies who are the main entities involved in the rental transaction process. The scope is confined to users renting a car of their choice based on location and companies servicing the customers' needs. Detailed payment processing, multi-language services, inventory management beyond rentals, location tracking, license verification, user accounts for non customers and non companies, analytics is out of scope. The companies can only see the reservations made for their respective company. Physically servicing the customers is out of scope of the project.

2. Project Description:

The title of the project is Car Rental Management System. The project requires users and companies to register before using the platform. The users are allowed to see the available cars and can rent one at a time. The companies can see the reserved cars that belong to their company and service them.

The major project functionalities include:

- User and company registration and authentication
- User and company dashboards
- Reservation system
- Data storage using MySQL
- Error handling
- Search and filter
- Company statistics including necessary graphs

3. System features and function requirements:

System feature 1: User registration

- Entities involved -> Customers and companies
- Inputs -> Username, password, contact, location
- Process -> Entities provide their registration details in their respective registration pages. The system validates the details based on the constraints specified and the existing entities. The data is stored in their respective tables and are verified every time the entity logs into their account.

System feature 2: Car reservation

- Entities involved -> Customer, Car and Reservation
- Inputs required -> Customer name, rented car, reservation dates
- Process -> Customer search available cars based on location and book a car specifying the pickup and return dates. The details specified by the customers are validated by the system and are stored in the reservation table. The system updates the car's availability status and the company can visit their account to see the reservations made for their company.

System feature 3: Car listing and management

- Entities involved -> Car and Company
- Inputs required -> Company authentication, Car details
- Process -> Companies can log into their accounts and manage their car listings. They can remove or add new cars to their listings. Every time they add a new car, they must provide the availability and price for the added car. Customers will consider these details before booking a car. Companies can also delete cars from their listings only if it is not reserved at that point of time.

System feature 4: Reservation management

- Entities involved -> Customer, Reservation
- Inputs required -> Customer authentication, Reservation management actions (view, cancel, add)
- Process -> Customers, after logging into their respective accounts, can perform any of the reservation actions such as viewing their current reservations, making a new reservation and cancelling an already existing reservation. The system validates their credentials and performs actions on the reservation table based on the customer's actions.

System feature 5: Company statistics

- Entities involved -> Reservations, Cars
- Process -> This function makes use of aggregate SQL functions like SUM, AVG, COUNT to obtain some basic statistical conclusions about the reservations for a particular company. Pie chart and Bar graphs are included to show the car availability and price of each car respectively.

QUERIES EXECUTION

1. User/roles creation and varied privileges:

```
--User roles and privileges
create user 'customers'@'localhost' identified by 'customers123';
create user 'companies'@'localhost' identified by 'companies123';
grant select, insert, delete on carrent.customers to 'customers'@'localhost';
grant select, update on carrent.cars to 'customers'@'localhost';
grant select, insert on carrent.reservations to 'customers'@'localhost';
grant execute on procedure carrent.sp_UpdateCarAvailability TO 'customers'@'localhost';
grant select, insert, delete on carrent.companies to 'companies'@'localhost';
grant select, insert, delete, update on carrent.cars to 'companies'@'localhost';
grant select on carrent.reservations to 'companies'@'localhost';
```

```
mysql> show grants for 'customers'@'localhost';
+-----------------------------------------------------------------------------------------------+
| Grants for customers@localhost                                                                |
+-----------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `customers`@`localhost`                                                 |
| GRANT SELECT, UPDATE ON `carrent`.`cars` TO `customers`@`localhost`                           |
| GRANT SELECT, INSERT, DELETE ON `carrent`.`customers` TO `customers`@`localhost`              |
| GRANT SELECT, INSERT ON `carrent`.`reservations` TO `customers`@`localhost`                   |
| GRANT EXECUTE ON PROCEDURE `carrent`.`sp_updatecaravailability` TO `customers`@`localhost`    |
+-----------------------------------------------------------------------------------------------+
5 rows in set (0.01 sec)

mysql> show grants for 'companies'@'localhost';
+-----------------------------------------------------------------------------------------+
| Grants for companies@localhost                                                          |
+-----------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `companies`@`localhost`                                           |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `carrent`.`cars` TO `companies`@`localhost`     |
| GRANT SELECT, INSERT, DELETE ON `carrent`.`companies` TO `companies`@`localhost`        |
| GRANT SELECT ON `carrent`.`reservations` TO `companies`@`localhost`                     |
+-----------------------------------------------------------------------------------------+
4 rows in set (0.00 sec)
```

2. Triggers

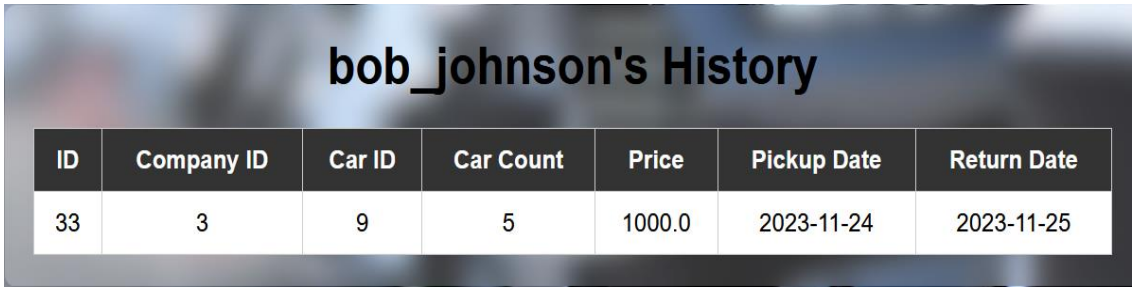
```
-- Trigger to update available cars for a company when a customer who has reserved cars has deleted the account
DELIMITER //
CREATE TRIGGER before_delete_customer
BEFORE DELETE ON customers
FOR EACH ROW
BEGIN
    DECLARE car_id_var INT;
    DECLARE car_count_var INT;

    SELECT car_id, car_count
    INTO car_id_var, car_count_var
    FROM reservations       You, 7 days ago • added a procedure and a trigger
    WHERE customer_id = OLD.customer_id
    LIMIT 1;

    IF car_id_var IS NOT NULL THEN
        UPDATE cars
        SET available = available + car_count_var
        WHERE car_id = car_id_var;
    END IF;
END;
//
DELIMITER ;
```

- Cars in company id=3 before user deletion:

Cars in tokyo

Company ID	Company Name	Company Contact	Company Location	Car ID	Car Name	Price per Day	Available
3	toyota	87654	tokyo	3	Compact B	45.0	1
3	toyota	87654	tokyo	9	Ford mustang	200.0	16
3	toyota	87654	tokyo	10	Creta	80.0	2
3	toyota	87654	tokyo	11	Eco Sport	75.0	3

- Reservation for car_id=9:



bob_johnson's History

ID	Company ID	Car ID	Car Count	Price	Pickup Date	Return Date
33	3	9	5	1000.0	2023-11-24	2023-11-25

- Cars after user deletion:

Your Car Collection

Car ID	Name	Price per Day	Count	Available
3	Compact B	45.0	20	1
9	Ford mustang	200.0	26	21
10	Creta	80.0	4	2
11	Eco Sport	75.0	6	3

- Reservations after user deletion:

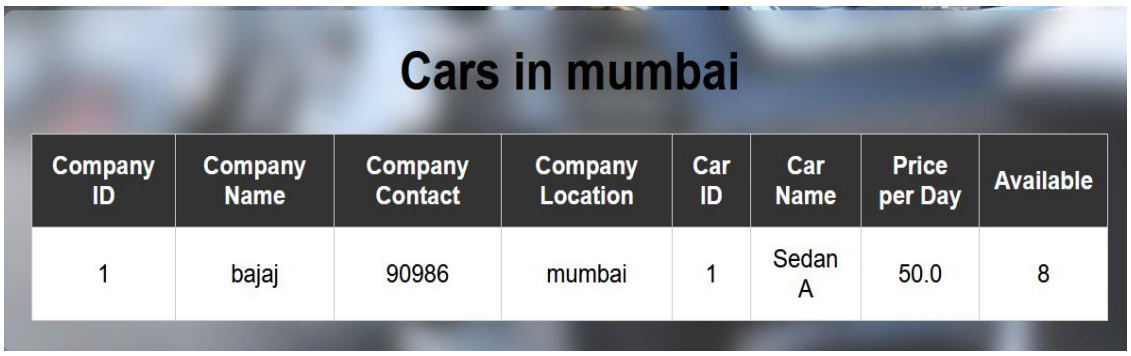
Reservations for You:

ID	Customer ID	Car ID	Car Count	Price	Pickup Date	Return Date	Contact

3. Procedures

```
--Procedure to update car_availability when cars are booked by customers
DELIMITER //

CREATE PROCEDURE sp_UpdateCarAvailability(IN p_car_id INT, IN p_count INT)
BEGIN
    UPDATE cars SET available = available - p_count WHERE car_id = p_car_id;
END //

DELIMITER ;
```

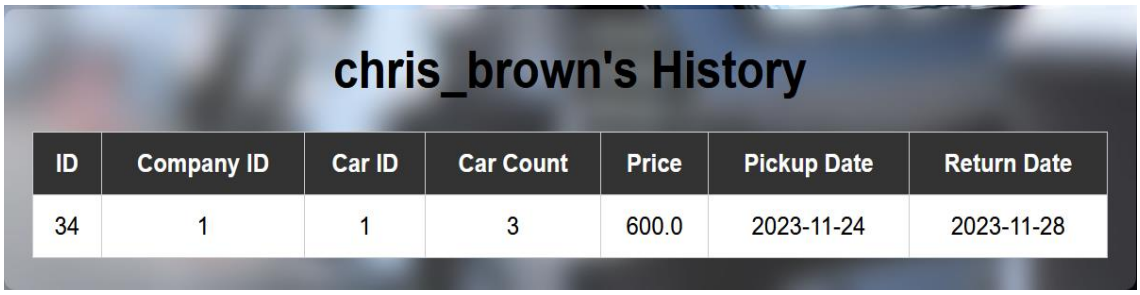
- Cars before reserving:



Cars in mumbai

Company ID	Company Name	Company Contact	Company Location	Car ID	Car Name	Price per Day	Available
1	bajaj	90986	mumbai	1	Sedan A	50.0	8

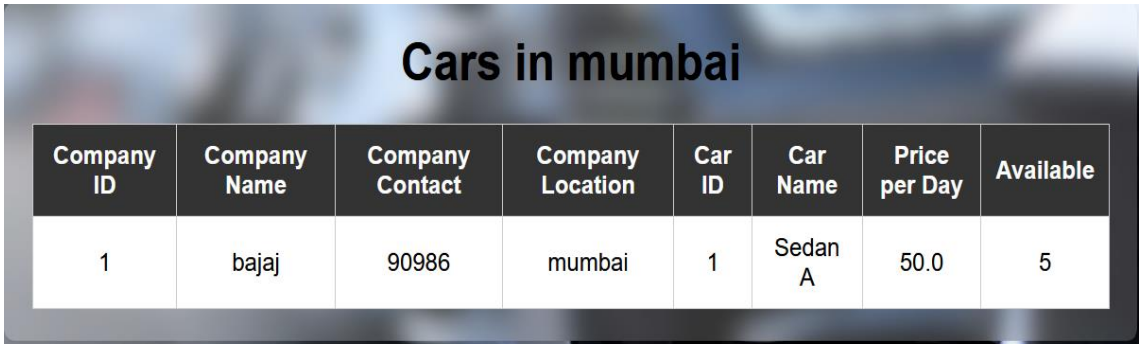
- Reserve a car:



chris_brown's History

ID	Company ID	Car ID	Car Count	Price	Pickup Date	Return Date
34	1	1	3	600.0	2023-11-24	2023-11-28

- Cars after reserving (change in availability):



Cars in mumbai

Company ID	Company Name	Company Contact	Company Location	Car ID	Car Name	Price per Day	Available
1	bajaj	90986	mumbai	1	Sedan A	50.0	5

4. Create operations:

Create database and tables (customers and companies.)

```
CREATE DATABASE if not exists carrent;
USE carrent;

create table customers
(customer_id int primary key auto_increment,
fname varchar(20) not null,
lname varchar(20) not null,
username varchar(20) not null,
password varchar(100) not null,
contact int not null
);

create table companies
(company_id int primary key auto_increment,
name varchar(20) not null,
password varchar(100) not null,
contact int not null,
location varchar(30) not null
);
```

Create table Cars.

```
create table cars
(car_id int primary key auto_increment,
name varchar(20) not null,
company_id int not null,
price_per_day float not null,
car_count int not null,
available int not null,
key fkcars1 (company_id),
constraint fkcars1 foreign key (company_id) references companies(company_id) on delete cascade,
constraint chk_price_per_day check ((price_per_day>0)),
constraint chk_count check((car_count>0)),
constraint chk_available check((available<=car_count))
);
```

Create table reservations:

```
create table reservations
(id int primary key auto_increment,
customer_id int not null,
company_id int not null,
price float not null,
pickup_date date not null,
return_date date not null,
car_id int not null,
car_count int not null,
key fkreservations1 (customer_id),
key fkreservations2 (company_id),
key fkreservations3 (car_id),
constraint fkreservations1 foreign key (customer_id) references customers(customer_id) on delete cascade,
constraint fkreservations2 foreign key (company_id) references companies(company_id) on delete cascade,
constraint fkreservations3 foreign key (car_id) references cars(car_id) on delete cascade,
constraint chk_car_count check((car_count>0))
);
```

Tables in SQL:

```
mysql> show tables;
+-------------------+
| Tables_in_carrent |
+-------------------+
| cars              |
| companies         |
| customers         |
| reservations      |
+-------------------+
4 rows in set (0.04 sec)
```

5. Read operations:

```
mysql> select * from cars where car_id=4;
+--------+--------------+------------+--------------+-----------+-----------+
| car_id | name         | company_id | price_per_day | car_count | available |
+--------+--------------+------------+--------------+-----------+-----------+
|      4 | Convertible C |          4 |           90 |         8 |         6 |
+--------+--------------+------------+--------------+-----------+-----------+
1 row in set (0.00 sec)

mysql> select * from customers;
+-------------+-------+----------+--------------+--------------+-----------+
| customer_id | fname | lname    | username     | password     | contact   |
+-------------+-------+----------+--------------+--------------+-----------+
|           5 | Eva   | Williams | eva_williams | mysecretpass | 111222333 |
|           6 | Chris | Brown    | chris_brown  | brownpass    | 999888777 |
|           7 | Megan | Taylor   | megan_taylor | taylorpass   | 444555666 |
|           8 | Ryan  | Miller   | ryan_miller  | millerpass   | 777888999 |
+-------------+-------+----------+--------------+--------------+-----------+
4 rows in set (0.00 sec)
```

```
mysql> select * from companies;
+------------+-----------+-------------+---------+-----------+
| company_id | name      | password    | contact | location  |
+------------+-----------+-------------+---------+-----------+
|          1 | bajaj     | jabab       |   90986 | mumbai    |
|          2 | ford      | pass123     |   98765 | detroit   |
|          3 | toyota    | securepass  |   87654 | tokyo     |
|          4 | honda     | hondapass   |   76543 | osaka     |
|          5 | audi      | audipass    |   54321 | ingolstadt |
|          6 | mercedes  | mercedespass |  43210 | stuttgart |
|          7 | volkswagen | vwpass     |   32109 | wolfsburg |
|          8 | bmw       | bmwpass     |   21098 | munich    |
+------------+-----------+-------------+---------+-----------+
8 rows in set (0.00 sec)

mysql> select * from reservations;
+----+-------------+------------+-------+-------------+-------------+--------+-----------+
| id | customer_id | company_id | price | pickup_date | return_date | car_id | car_count |
+----+-------------+------------+-------+-------------+-------------+--------+-----------+
| 34 |           6 |          1 |   600 | 2023-11-24  | 2023-11-28  |      1 |         3 |
+----+-------------+------------+-------+-------------+-------------+--------+-----------+
1 row in set (0.00 sec)
```

6. Update operations

Inserting values into the tables

```sql
--Values for companies page
INSERT INTO companies (company_id, name, password, contact, location)
VALUES (1, 'bajaj', 'jabab', 90986, 'mumbai');
INSERT INTO companies (company_id, name, password, contact, location)
VALUES (2, 'ford', 'pass123', 98765, 'detroit');
INSERT INTO companies (company_id, name, password, contact, location)
VALUES (3, 'toyota', 'securepass', 87654, 'tokyo');
INSERT INTO companies (company_id, name, password, contact, location)
VALUES (4, 'honda', 'hondapass', 76543, 'osaka');
```

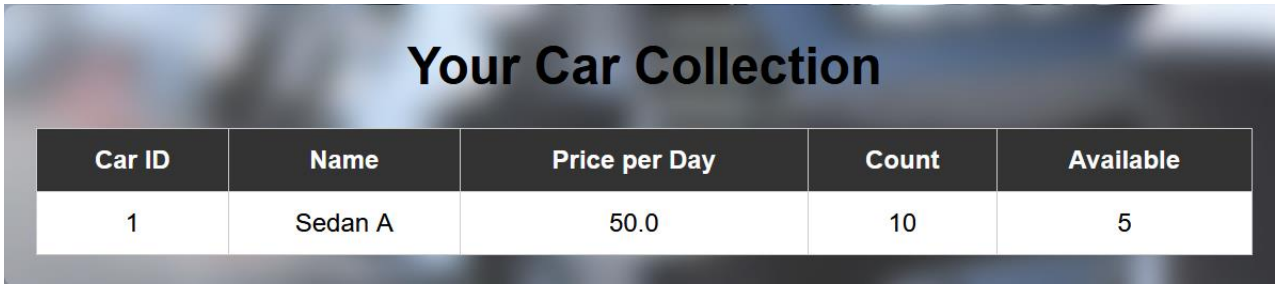
```sql
--Values for customers page
INSERT INTO customers (customer_id, fname, lname, username, password, contact)
VALUES (1, 'likith', 'mc', 'likith', 'lik', 12345);
INSERT INTO customers (customer_id, fname, lname, username, password, contact)
VALUES (2, 'John', 'Doe', 'john_doe', 'password123', 123456789);
INSERT INTO customers (customer_id, fname, lname, username, password, contact)
VALUES (3, 'Alice', 'Smith', 'alice_smith', 'securepass', 987654321);
INSERT INTO customers (customer_id, fname, lname, username, password, contact)
VALUES (4, 'Bob', 'Johnson', 'bob_johnson', 'pass1234', 555666777);
```

```sql
--Values for cars page
INSERT INTO cars (car_id, name, company_id, price_per_day, car_count, available)
VALUES (1, 'Sedan A', 1, 50.0, 10, 8);
INSERT INTO cars (car_id, name, company_id, price_per_day, car_count, available)
VALUES (2, 'SUV X', 2, 70.0, 15, 12);
INSERT INTO cars (car_id, name, company_id, price_per_day, car_count, available)
VALUES (3, 'Compact B', 3, 45.0, 20, 18);
INSERT INTO cars (car_id, name, company_id, price_per_day, car_count, available)
VALUES (4, 'Convertible C', 4, 90.0, 8, 6);
```

Update car count if cars are added to the company:

```python
car_id=existing_car[0]
car_count=existing_car[1]+int(limit)
available=existing_car[2]+int(limit)
query="update cars set price_per_day=%s, car_count=%s, available=%s where car_id=%s and company_id=%s"
data=(price_per_day,car_count,available,car_id,company_id)
cursor.execute(query,data)
connection.commit()
cursor.close()      You, last month • function to update db with adding and removing of…
connection.close()
return "updated info successfully"
```

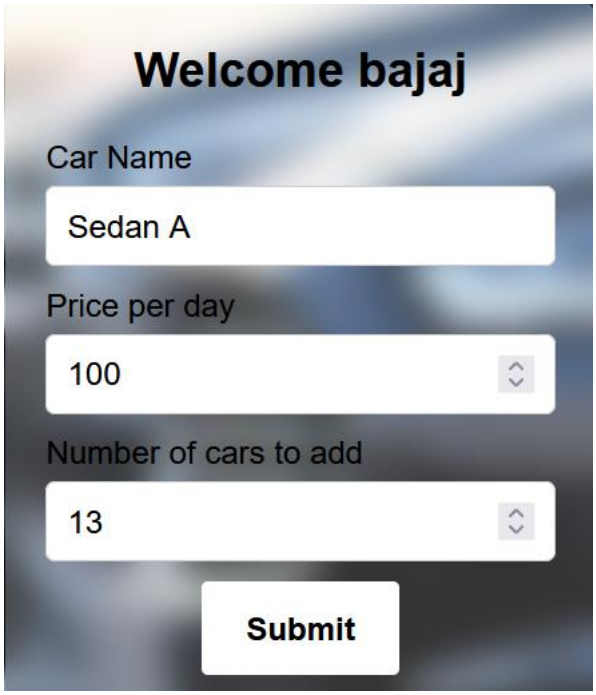
Before adding cars:



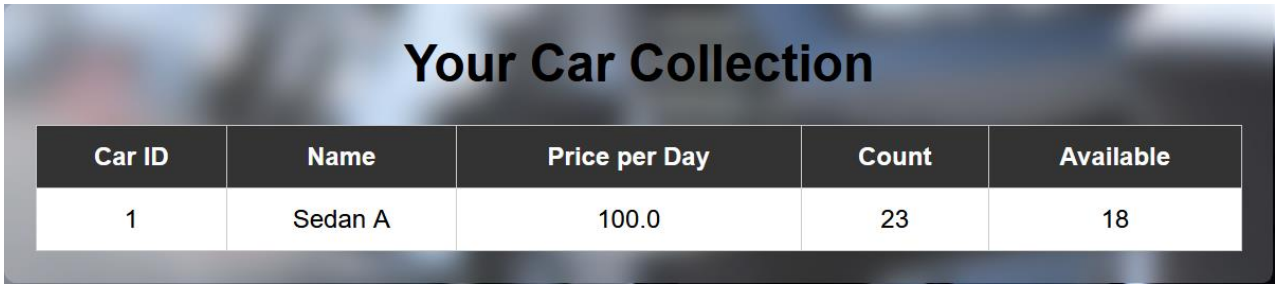
Your Car Collection

Car ID	Name	Price per Day	Count	Available
1	Sedan A	50.0	10	5

Adding cars to the company:



After adding cars to the company



Your Car Collection

Car ID	Name	Price per Day	Count	Available
1	Sedan A	100.0	23	18

7. Delete operations:

Customer deletion code snippet

```python
@app.route('/delete_customer/<int:customer_id>',methods=['GET'])
def deleteaccount(customer_id):
    connection=create_connection_customers()
    cursor=connection.cursor()
    query="delete from customers where customer_id=%s"
    data=(customer_id,)
    cursor.execute(query,data)
    connection.commit()
    cursor.close()
    connection.close()
    return redirect(url_for('home'))
```

Before deleting:

```
mysql> select * from customers;
+-------------+-------+--------+--------------+-----------+-----------+
| customer_id | fname | lname  | username     | password  | contact   |
+-------------+-------+--------+--------------+-----------+-----------+
|           6 | Chris | Brown  | chris_brown  | brownpass | 999888777 |
|           7 | Megan | Taylor | megan_taylor | taylorpass| 444555666 |
|           8 | Ryan  | Miller | ryan_miller  | millerpass| 777888999 |
+-------------+-------+--------+--------------+-----------+-----------+
3 rows in set (0.00 sec)
```

After deleting:

```
mysql> select * from customers;
+-------------+-------+--------+--------------+-----------+-----------+
| customer_id | fname | lname  | username     | password  | contact   |
+-------------+-------+--------+--------------+-----------+-----------+
|           7 | Megan | Taylor | megan_taylor | taylorpass| 444555666 |
|           8 | Ryan  | Miller | ryan_miller  | millerpass| 777888999 |
+-------------+-------+--------+--------------+-----------+-----------+
2 rows in set (0.00 sec)
```

8. Join queries:

Used to view available cars for the customer given the location, joining cars and company tables.

```
@app.route('/availablecars/<location>',methods=['GET','POST'])
def viewcars(location):
    connection=create_sql_connection()
    cursor=connection.cursor()
    query="""select c.company_id, c.name, c.contact, c.location,
    cars.car_id, cars.name, cars.price_per_day, cars.available
    from companies as c
    inner join cars on cars.company_id=c.company_id
    where substring_index(c.location,',',-1) = %s;"""
    cursor.execute(query,(location,))
    results=cursor.fetchall()
    cursor.close()
    connection.close()
    return render_template("searchcars.html", results=results, location=location)
```

Cars in tokyo

Company ID	Company Name	Company Contact	Company Location	Car ID	Car Name	Price per Day	Available
3	toyota	87654	tokyo	3	Compact B	45.0	1
3	toyota	87654	tokyo	9	Ford mustang	200.0	21
3	toyota	87654	tokyo	10	Creta	80.0	2
3	toyota	87654	tokyo	11	Eco Sport	75.0	3

Used to find reservation statistic for a company joining cars and reservations tables.

```
query="""
select cars.name, count(reservations.id) as rental_count
from cars left join reservations on cars.car_id=reservations.car_id
where cars.company_id=%s
group by cars.name
order by rental_count desc
limit 1
"""
cursor.execute(query,data)
most_rented=cursor.fetchone()
```

Used to find all reservations for a company joining reservations and customers tables.

```
@app.route('/company/bookings/<int:company_id>',methods=['GET'])
def bookings(company_id):
    connection=create_sql_connection()
    cursor=connection.cursor()
    query="""SELECT r.*, c.contact AS customer_contact FROM reservations r
    JOIN customers c ON r.customer_id = c.customer_id WHERE r.company_id = %s"""
    data=(company_id,)
```

ID	Customer ID	Car ID	Car Count	Price	Pickup Date	Return Date	Contact
38	8	11	2	300.0	2023-11-30	2023-12-02	777888999
37	8	10	1	400.0	2023-11-23	2023-11-28	777888999
36	7	9	8	3200.0	2023-11-25	2023-11-27	444555666
35	7	3	1	90.0	2023-11-24	2023-11-26	444555666

9. Aggregate queries:

Queries for company statistics.

```
query="""
select count(distinct(cars.car_id)),
avg(price_per_day),
sum(available) from cars where company_id=%s"""
data=(company_id,)
cursor.execute(query,data)
statistics=cursor.fetchone()

query="""
select sum(price)
from reservations
where company_id=%s
"""
data=(company_id,)
cursor.execute(query,data)
revenue=cursor.fetchone()
```

```
mysql> select count(distinct(cars.car_id)),
    ->         avg(price_per_day),
    ->         sum(available) from cars where company_id=3;
+------------------------------+--------------------+----------------+
| count(distinct(cars.car_id)) | avg(price_per_day) | sum(available) |
+------------------------------+--------------------+----------------+
|                            4 |                100 |             15 |
+------------------------------+--------------------+----------------+
1 row in set (0.00 sec)

mysql>  select sum(price)
    ->         from reservations
    ->         where company_id=3;
+------------+
| sum(price) |
+------------+
|       3990 |
+------------+
1 row in set (0.00 sec)
```