



Bilkent University  
Department of Computer Engineering

# **CS353 Database Systems**

## ***Group 13 Project Design Report Car Rental System***

29.11.2021

### *Group Members:*

Akmuhammet Ashyralyev - 21801347

Berke Ceran - 21703920

Hakan Gülcü - 21702275

Sila Saraoglu - 21803313

Instructor: Özgür Ulusoy

Teaching Assistant: Mustafa Can Çavdar

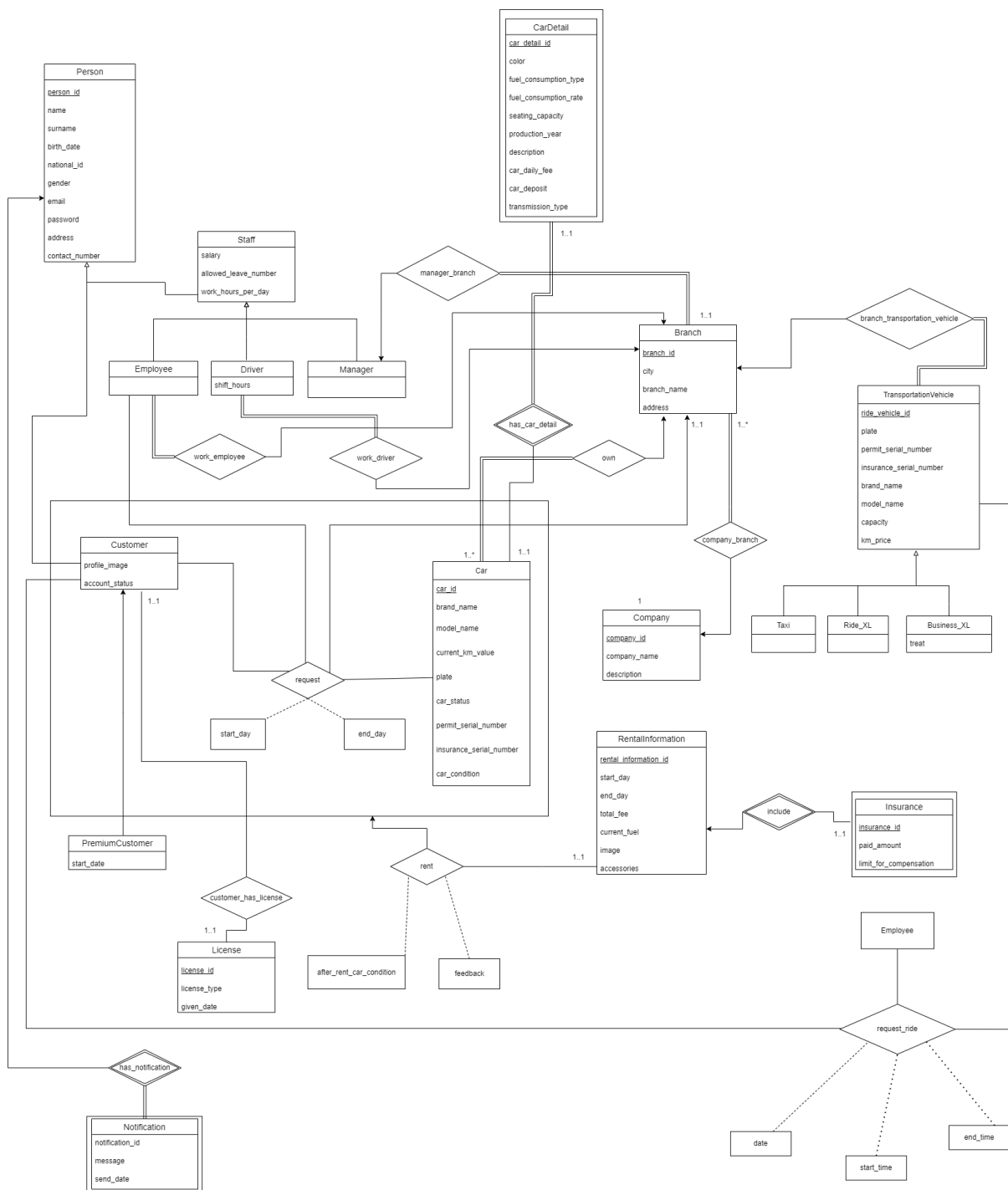
<https://berkeceran.github.io/CarRentalSystem/>

|  |           |
|--|-----------|
| <b>1. Revised E/R Model</b>                          | <b>2</b>  |
| <b>2. Relation Schemas</b>                           | <b>3</b>  |
| 2.1. Person  | 3         |
| 2.2. Staff   | 5         |
| 2.3. Employee  | 5         |
| 2.4. Driver  | 6         |
| 2.5. Manager   | 7         |
| 2.6. Customer  | 8         |
| 2.6. PremiumCustomer                                 | 8         |
| 2.7. Car   | 9         |
| 2.8. CarDetail                                       | 11        |
| 2.9. Branch  | 12        |
| 2.10. Company  | 13        |
| 2.11. TransportationVehicle                          | 13        |
| 2.12. Taxi   | 15        |
| 2.13. Ride_XL  | 15        |
| 2.14. Business_XL                                    | 16        |
| 2.15. License  | 17        |
| 2.16. RentalInformation                              | 17        |
| 2.17. Insurance                                      | 19        |
| 2.18. request  | 20        |
| 2.19. request_ride                                   | 21        |
| 2.20. Notification                                   | 22        |
| <b>3. Functional Dependencies</b>                    | <b>23</b> |
| <b>4. UI Design And Corresponding SQL Statements</b> | <b>23</b> |
| 4.1. Sign Up Page                                    | 23        |
| 4.2. Login Page                                      | 24        |
| 4.3. Customer Main Page                              | 26        |
| 4.4. Search Car Customer                             | 27        |
| 4.5. Customer Notifications                          | 31        |
| 4.6. Customer Current Documents Page                 | 32        |
| 4.7. Employee Main Page                              | 34        |
| 4.8. Employee Create Form Page                       | 36        |
| 4.9. Customer Profile Page                           | 38        |
| 4.10. Employee Profile Page                          | 39        |
| 4.11. Manager Main Page                              | 41        |
| 4.12. Create Employee Page                           | 42        |
| 4.13. Add Car Page                                   | 43        |
| <b>5. Implementation Plan</b>                        | <b>44</b> |

# 1. Revised E/R Model

## Changes

- The manager entity is added with its relation to the branch as manager\_branch.
- The driver entity (who is the driver in the branch that is employed for the transportation functionality) with its relation is added.
- The employee entity's name is changed to staff to indicate differences between a classic employee, a driver and a manager.
- For the driver entity, shift\_hours attribute is added to indicate if the driver is working during day or night shift.
- Transportation vehicle entity is added with its' specialization entities: taxi, ride XL, and business XL with its' new attribute treat which indicates that users who are using Business XL vehicles will have treats on their journey.
- The weak entities from the previous design of E-R diagram such as cars are corrected.
- The weak notification entity is added to the E-R diagram with its' relation between customer which is the messages given by the system to customers.
- A request\_ride relation is added between the customers, employees and transportation vehicles to indicate that customers can request rides for transportation vehicles from employees in the system within a specific date, beginning and end time (as the relation's attributes).
- PersonDetail weak entity class has been eliminated and its attributes are added to Person entity.
- For extra functionality of our system, requesting a transportation vehicle (a ride) is added for this functionality: the transportation vehicle entity and its' specialization entities such as Taxi added. Moreover for the extra functionality, the Driver entity is added.
- For each staff entity's specialization, a new relation such as manager\_branch is introduced. The underlying reason that introducing 3 relations for each specialization entity is that their multiplicities are different.



## 2. Relation Schemas

### 2.1. Person

**Model:**

Person(person\_id, name, surname, birth\_date, national\_id, gender, email, password, address, contact\_number)

**Candidate Keys:**

{(person\_id), (national\_id), (email)}

**Primary Key:**

(person\_id)

**Functional Dependencies:**

person\_id → name, surname, birth\_date, national\_id, gender, email, password, address, contact\_number

national\_id → person\_id, name, surname, birth\_date, gender, email, password, address, contact\_number

email → person\_id, name, surname, birth\_date, national\_id, gender, password, address, contact\_number

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Person(  
  person_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
  name      VARCHAR(20) NOT NULL,  
  surname   VARCHAR(25) NOT NULL,  
  birth_date DATE NOT NULL,  
  national_id VARCHAR(32) NOT NULL,  
  gender    VARCHAR(32),  
  email     VARCHAR(128) NOT NULL UNIQUE,  
  password  VARCHAR(21) NOT NULL,  
  address   VARCHAR(128) NOT NULL,  
  contact_number VARCHAR(20) NOT NULL) ENGINE = InnoDB;
```

## 2.2. Staff

### Model:

Staff(person\_id, salary, allowed\_leave\_number, work\_hours\_per\_day)

Foreign Key: person\_id references Person(person\_id)

### Candidate Keys:

{{person\_id}}

### Primary Key:

(person\_id)

### Functional Dependencies:

person\_id → salary, allowed\_leave\_number, work\_hours\_per\_day

### Normal Form:

BCNF

### Table Definition:

```
CREATE TABLE Staff(  
  person_id INT NOT NULL PRIMARY KEY,  
  salary      FLOAT NOT NULL,  
  allowed_leave_number INT,  
  work_hours_per_day INT,  
  FOREIGN KEY (person_id) REFERENCES Person(person_id)) ENGINE  
= InnoDB;
```

## 2.3. Employee

### Model:

Employee(employee\_id, branch\_id)

Foreign Key: employee\_id references Person(person\_id)

Foreign Key: branch\_id references Branch(branch\_id)

### Candidate Keys:

{{employee\_id}}

**Primary Key:**

(employee\_id)

**Functional Dependencies:**

employee\_id  $\rightarrow$  branch\_id

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Employee(  
  employee_id INT NOT NULL PRIMARY KEY,  
  branch_id INT NOT NULL,  
  FOREIGN KEY (employee_id) REFERENCES Person(person_id),  
  FOREIGN KEY (branch_id) REFERENCES Branch(branch_id)) ENGINE  
= InnoDB;
```

## 2.4. Driver

**Model:**

Driver(person\_id, shift\_hours, branch\_id)

Foreign Key: person\_id references Person(person\_id)

Foreign Key: branch\_id references Branch(branch\_id)

**Candidate Keys:**

{{(person\_id)}}

**Primary Key:**

(person\_id)

**Functional Dependencies:**

person\_id  $\rightarrow$  shift\_hours, branch\_id

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Driver(  
  person_id INT NOT NULL PRIMARY KEY,  
  shift_hours VARCHAR(15),  
  branch_id INT NOT NULL  
  FOREIGN KEY (person_id) REFERENCES Person(person_id),  
  FOREIGN KEY (branch_id) REFERENCES Branch(branch_id))) ENGINE  
= InnoDB;
```

## 2.5. Manager

**Model:**

Manager(manager\_id)

Foreign Key: manager\_id references Person(person\_id)

**Candidate Keys:**

{{manager\_id}}

**Primary Key:**

(manager\_id)

**Functional Dependencies:**

None

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Manager(  
  manager_id INT NOT NULL PRIMARY KEY,
```



FOREIGN KEY (person\_id) REFERENCES Person(person\_id)) ENGINE = InnoDB;

## 2.6. Customer

### **Model:**

Customer(customer\_id, profile\_image, account\_status, license\_id)

Foreign Key: customer\_id references Person(person\_id)

Foreign Key: license\_id references License(license\_id)

### **Candidate Keys:**

{{customer\_id}}

### **Primary Key:**

(customer\_id)

### **Functional Dependencies:**

customer\_id → profile\_image, account\_status, license\_id

### **Normal Form:**

BCNF

### **Table Definition:**

```
CREATE TABLE Customer(  
  customer_id          INT NOT NULL PRIMARY KEY,  
  profile_image VARCHAR(128),  
  account_status VARCHAR(10),  
  license_id  INT NOT NULL UNIQUE,  
  FOREIGN KEY (license_id) REFERENCES License(license_id),  
  FOREIGN KEY (customer_id) REFERENCES Person(person_id))  
ENGINE = InnoDB;
```

## 2.6. PremiumCustomer

### **Model:**

PremiumCustomer(person\_id, start\_date)

Foreign Key: person\_id references Person(person\_id)

**Candidate Keys:**

{{person\_id}}

**Primary Key:**

(person\_id)

**Functional Dependencies:**

person\_id → start\_date

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE PremiumCustomer(  
  person_id INT NOT NULL PRIMARY KEY,  
  start_date DATE,  
  FOREIGN KEY (person_id) REFERENCES Person(person_id)) ENGINE  
= InnoDB;
```

## 2.7. Car

**Model:**

Car(car\_id, brand\_name, model\_name, current\_km\_value, plate,  
car\_status, permit\_serial\_number, insurance\_serial\_number,  
car\_condition, branch\_id)

Foreign Key: branch\_id references Branch(branch\_id)

**Candidate Keys:**

{{car\_id}, {plate}, {permit\_serial\_number}, {insurance\_serial\_number}}

**Primary Key:**

(car\_id)

**Functional Dependencies:**

car\_id → brand\_name, model\_name, current\_km\_value, plate,  
car\_status, permit\_serial\_number, insurance\_serial\_number,  
car\_condition, branch\_id

plate → car\_id, brand\_name, model\_name, current\_km\_value,  
car\_status, permit\_serial\_number, insurance\_serial\_number,  
car\_condition, branch\_id

permit\_serial\_number → car\_id, brand\_name, model\_name,  
current\_km\_value, plate, car\_status, insurance\_serial\_number,  
car\_condition, branch\_id

insurance\_serial\_number → car\_id, brand\_name, model\_name,  
current\_km\_value, plate, car\_status, permit\_serial\_number,  
car\_condition, branch\_id

### **Normal Form:**

BCNF

### **Table Definition:**

CREATE TABLE Car(

car\_id INT NOT NULL AUTO\_INCREMENT PRIMARY  
KEY,

brand\_name VARCHAR(20) NOT NULL,

model\_name VARCHAR(20) NOT NULL,

current\_km\_value INT NOT NULL,

plate VARCHAR(20) NOT NULL UNIQUE,

car\_status VARCHAR(20),

permit\_serial\_number VARCHAR(10) NOT NULL UNIQUE,

insurance\_serial\_number VARCHAR(10) NOT NULL UNIQUE,

car\_condition VARCHAR(20),

branch\_id INT NOT NULL,

FOREIGN KEY (branch\_id) REFERENCES Branch(branch\_id)) ENGINE  
= InnoDB;

## 2.8. CarDetail

### Model:

CarDetail(car\_id, car\_detail\_id, color, fuel\_consumption\_type, fuel\_consumption\_rate, seating\_capacity, production\_year, description, car\_daily\_fee, car\_deposit, transmission\_type)

Foreign Key: car\_id references Car(car\_id)

### Candidate Keys:

{(car\_id, car\_detail\_id)}

### Primary Key:

(car\_id, car\_detail\_id)

### Functional Dependencies:

car\_id, car\_detail\_id → color, fuel\_consumption\_type, fuel\_consumption\_rate, seating\_capacity, production\_year, description, car\_daily\_fee, car\_deposit, transmission\_type

### Normal Form:

BCNF

### Table Definition:

```
CREATE TABLE CarDetail(  
  car_id          INT NOT NULL,  
  car_detail_id   INT NOT NULL AUTO_INCREMENT,  
  color           VARCHAR(10),  
  fuel_consumption_type VARCHAR(10),  
  fuel_consumption_rate VARCHAR(10),  
  seating_capacity INT,  
  production_year VARCHAR(10),  
  description     VARCHAR(128),  
  car_daily_fee   INT NOT NULL,  
  car_deposit     INT NOT NULL,  
  transmission_type VARCHAR(20),
```

PRIMARY KEY (car\_id, car\_detail\_id),

FOREIGN KEY (car\_id) REFERENCES Car(car\_id))ENGINE = InnoDB;

## 2.9. Branch

### **Model:**

Branch(branch\_id, city, branch\_name, address, manager\_id, company\_id)

Foreign Key: manager\_id references Manager(person\_id)

Foreign Key: company\_id references Company(company\_id)

### **Candidate Keys:**

{{branch\_id}}

### **Primary Key:**

(branch\_id)

### **Functional Dependencies:**

branch\_id → city, branch\_name, address, manager\_id, company\_id

### **Normal Form:**

BCNF

### **Table Definition:**

```
CREATE TABLE Branch(  
branch_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
city      VARCHAR(20),  
branch_name VARCHAR(20),  
address   VARCHAR(128),  
manager_id INT NOT NULL UNIQUE,  
company_id INT NOT NULL,  
FOREIGN KEY (manager_id) REFERENCES Manager(person_id),  
FOREIGN KEY (company_id) REFERENCES Company(company_id)
```

) ENGINE = InnoDB;

## 2.10. Company

### **Model:**

Company(company\_id, company\_name, description)

### **Candidate Keys:**

{(company\_id), (company\_name)}

### **Primary Key:**

(company\_id)

### **Functional Dependencies:**

company\_id → company\_name, description

company\_name → company\_id, description

### **Normal Form:**

BCNF

### **Table Definition:**

```
CREATE TABLE Company(  
  company_id      INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
  company_name    VARCHAR(20) NOT NULL UNIQUE,  
  description     VARCHAR(128)) ENGINE = InnoDB;
```

## 2.11. TransportationVehicle

### **Model:**

TransportationVehicle(ride\_vehicle\_id, plate, permit\_serial\_number, insurance\_serial\_number, brand\_name, model\_name, capacity, km\_price, branch\_id)

Foreign Key: branch\_id references Branch(branch\_id)

### **Candidate Keys:**

{(ride\_vehicle\_id), (plate), (permit\_serial\_number),  
(insurance\_serial\_number)}

**Primary Key:**

(ride\_vehicle\_id)

**Functional Dependencies:**

ride\_vehicle\_id → plate, permit\_serial\_number,  
insurance\_serial\_number, brand\_name, model\_name, capacity,  
km\_price, branch\_id

plate → ride\_vehicle\_id, permit\_serial\_number,  
insurance\_serial\_number, brand\_name, model\_name, capacity,  
km\_price, branch\_id

permit\_serial\_number → ride\_vehicle\_id, plate,  
insurance\_serial\_number, brand\_name, model\_name, capacity,  
km\_price, branch\_id

insurance\_serial\_number → ride\_vehicle\_id, plate,  
permit\_serial\_number, brand\_name, model\_name, capacity, km\_price,  
branch\_id

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE TransportationVehicle(  
ride_vehicle_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
plate VARCHAR(20) NOT NULL UNIQUE,  
permit_serial_number VARCHAR(20) NOT NULL UNIQUE,  
insurance_serial_number VARCHAR(20) NOT NULL UNIQUE,  
brand_name VARCHAR(20),  
model_name VARCHAR(20),  
capacity INT,  
km_price INT,  
branch_id INT NOT NULL UNIQUE,
```

```
FOREIGN KEY(branch_id) REFERENCES Branch(branch_id)
) ENGINE = InnoDB;
```

## 2.12. Taxi

### **Model:**

Taxi(ride\_vehicle\_id)

Foreign Key: ride\_vehicle\_id references to TransportationVehicle

### **Candidate Keys:**

{{ride\_vehicle\_id}}

### **Primary Key:**

(ride\_vehicle\_id)

### **Functional Dependencies:**

None

### **Normal Form:**

BCNF

### **Table Definition:**

```
CREATE TABLE Taxi(
ride_vehicle_id  INT NOT NULL PRIMARY KEY,
FOREIGN KEY (ride_vehicle_id) REFERENCES
TransportationVehicle(ride_vehicle_id)
) ENGINE = InnoDB;
```

## 2.13. Ride\_XL

### **Model:**

Ride\_XL(ride\_vehicle\_id)

Foreign Key: ride\_vehicle\_id references to TransportationVehicle

### **Candidate Keys:**

{{ride\_vehicle\_id}}

### **Primary Key:**



(ride\_vehicle\_id)

**Functional Dependencies:**

None

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Ride_XL(  
ride_vehicle_id INT NOT NULL PRIMARY KEY,  
FOREIGN KEY (ride_vehicle_id) REFERENCES  
TransportationVehicle(ride_vehicle_id)  
) ENGINE = InnoDB;
```

## 2.14. Business\_XL

**Model:**

Business\_XL(ride\_vehicle\_id, treat)

Foreign Key: ride\_vehicle\_id references to TransportationVehicle

**Candidate Keys:**

{(ride\_vehicle\_id)}

**Primary Key:**

(ride\_vehicle\_id)

**Functional Dependencies:**

ride\_vehicle\_id → treat

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Business_XL(  
ride_vehicle_id INT NOT NULL PRIMARY KEY,  
treat VARCHAR(20),
```

```
FOREIGN KEY (ride_vehicle_id) REFERENCES  
TransportationVehicle(ride_vehicle_id)  
) ENGINE = InnoDB;
```

## 2.15. License

### **Model:**

License(license\_id, license\_type, given\_date)

### **Candidate Keys:**

{{license\_id}}

### **Primary Key:**

(license\_id)

### **Functional Dependencies:**

license\_id → license\_type, given\_date

### **Normal Form:**

BCNF

### **Table Definition:**

```
CREATE TABLE License(  
license_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
license_type VARCHAR(5) NOT NULL,  
given_date DATE NOT NULL  
) ENGINE = InnoDB;
```

## 2.16. RentalInformation

### **Model:**

RentalInformation(rental\_information\_id, start\_day, end\_day, total\_fee,  
current\_fuel, image, accessories, car\_id, customer\_id, employee\_id,  
after\_car\_rent\_condition, feedback)

Foreign Key: car\_id references Car(car\_id)

Foreign Key: customer\_id references Customer(person\_id)

Foreign Key: employee\_id references Employee(person\_id)

**Candidate Keys:**

{{rental\_information\_id}}

**Primary Key:**

(rental\_information\_id)

**Functional Dependencies:**

rental\_information\_id → start\_day, end\_day, total\_fee, current\_fuel,  
image, accessories, car\_id, customer\_id, employee\_id,  
after\_car\_rent\_condition, feedback

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE RentalInformation(  
rental_information_id INT NOT NULL AUTO_INCREMENT PRIMARY  
KEY,  
start_day DATE NOT NULL,  
end_day DATE NOT NULL,  
total_fee FLOAT,  
current_fuel FLOAT,  
image VARCHAR (128),  
accessories VARCHAR (32),  
car_id INT NOT NULL,  
employee_id INT NOT NULL,  
customer_id INT NOT NULL,  
after_car_rent_condition VARCHAR (128),  
feedback VARCHAR (128),
```

```
FOREIGN KEY (car_id) REFERENCES Car(car_id),  
FOREIGN KEY (employee_id) REFERENCES Employee(person_id),  
FOREIGN KEY (customer_id) REFERENCES Customer(person_id)  
) ENGINE = InnoDB;
```

## 2.17. Insurance

### **Model:**

Insurance(insurance\_id, paid\_amount, limit\_for\_compensation,  
rental\_information\_id)

Foreign Key: rental\_information\_id references  
RentalInformation(rental\_information\_id)

### **Candidate Keys:**

{(insurance\_id)}

### **Primary Key:**

(insurance\_id)

### **Functional Dependencies:**

insurance\_id → paid\_amount, limit\_for\_compensation,  
rental\_information\_id

### **Normal Form:**

BCNF

### **Table Definition:**

```
CREATE TABLE Insurance (  
insurance_id    INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
paid_amount    INT,  
limit_for_compensation INT NOT NULL,  
rental_information_id INT NOT NULL UNIQUE,  
FOREIGN KEY (rental_information_id) REFERENCES  
RentalInformation(rental_information_id)
```

) ENGINE = InnoDB;

## 2.18. request

### **Model:**

request(employee\_id, customer\_id, car\_id, branch\_id, start\_day, end\_day)

Foreign Key: employee\_id references Employee(person\_id)

Foreign Key: customer\_id references Customer(person\_id)

Foreign Key: car\_id references Car(car\_id)

Foreign Key: branch\_id references Branch(branch\_id)

### **Candidate Keys:**

{(employee\_id, customer\_id, car\_id, branch\_id)}

### **Primary Key:**

(employee\_id, customer\_id, car\_id, branch\_id)

### **Functional Dependencies:**

employee\_id, customer\_id, car\_id, branch\_id → start\_day, end\_day

### **Normal Form:**

BCNF

### **Table Definition:**

```
CREATE TABLE request (  
  employee_id    INT NOT NULL,  
  customer_id    INT NOT NULL,  
  car_id         INT NOT NULL,  
  branch_id     INT NOT NULL,  
  start_day      DATE NOT NULL,  
  end_day        DATE NOT NULL,
```

```

PRIMARY KEY (employee_id, customer_id, car_id),
FOREIGN KEY (employee_id) REFERENCES Employee(person_id),
FOREIGN KEY (customer_id) REFERENCES Customer(person_id),
FOREIGN KEY (car_id) REFERENCES Car(car_id),
FOREIGN KEY (branch_id) REFERENCES Branch(branch_id)
) ENGINE = InnoDB;

```

## 2.19. request\_ride

### Model:

```
request_ride(employee_id, customer_id, ride_vehicle_id, date,
start_time, end_time)
```

Foreign Key: employee\_id references Employee(person\_id)

Foreign Key: customer\_id references Customer(person\_id)

Foreign Key: ride\_vehicle\_id references Car(car\_id)

### Candidate Keys:

```
{(employee_id, customer_id, ride_vehicle_id)}
```

### Primary Key:

```
(employee_id, customer_id, ride_vehicle_id)
```

### Functional Dependencies:

```
employee_id, customer_id, ride__vehicle_id → date, start_time,
end_time
```

### Normal Form:

BCNF

### Table Definition:

```

CREATE TABLE request_ride (
employee_id      INT NOT NULL,
customer_id      INT NOT NULL,
ride_vehicle_id  INT NOT NULL,

```

```

date          DATE NOT NULL,
start_time    TIME NOT NULL,
end_time      TIME,
PRIMARY KEY (employee_id, customer_id, ride_vehicle_id),
FOREIGN KEY (employee_id) REFERENCES Employee(person_id),
FOREIGN KEY (customer_id) REFERENCES Customer(person_id),
FOREIGN KEY (ride_vehicle_id) REFERENCES
TransportationVehicle(ride_vehicle_id)
) ENGINE = InnoDB;

```

## 2.20. Notification

### **Model:**

Notification(person\_id, notification\_id, message, send\_date)

Foreign Key: person\_id references Person(person\_id)

### **Candidate Keys:**

{{person\_id, notification\_id}}

### **Primary Key:**

(person\_id, notification\_id)

### **Functional Dependencies:**

person\_id, notification\_id → message, send\_date

### **Normal Form:**

BCNF

### **Table Definition:**

```

CREATE TABLE Notification (
person_id      INT NOT NULL ,
notification_id INT NOT NULL AUTO_INCREMENT,
message        VARCHAR(32),
send_date      DATE,

```

```
PRIMARY KEY (person_id, notification_id),  
FOREIGN KEY (person_id) REFERENCES Person(person_id)  
) ENGINE = InnoDB;
```

All relational schemas in our system are in BCNF by having trivial functional dependencies thus all our relational schemas also in 3NF form.

### 3. Functional Dependencies

The functional dependencies are given with its' normalization forms in the previous section.

## 4. UI Design And Corresponding SQL Statements

**NOTE :** The @ sign used below is for the values taken from the system, and the values in the database are written unsigned.

### 4.1. Sign Up Page

Sign Up

Sign Up / Login

Name\*  
Please enter name

Surname\*  
Please enter surname

Email\*  
Email

Birth Date\*  
Birt Date

National ID\*  
National ID

Password  
Password

Verify Password  
Password

Address\*  
Address

Sing Up



The login and sign up is located in single but accessed by the switch as it is shown above. If the user uses a switch, then the appropriate page components will show up. Only customers can create accounts since employees will have a predefined account (defined by manager).

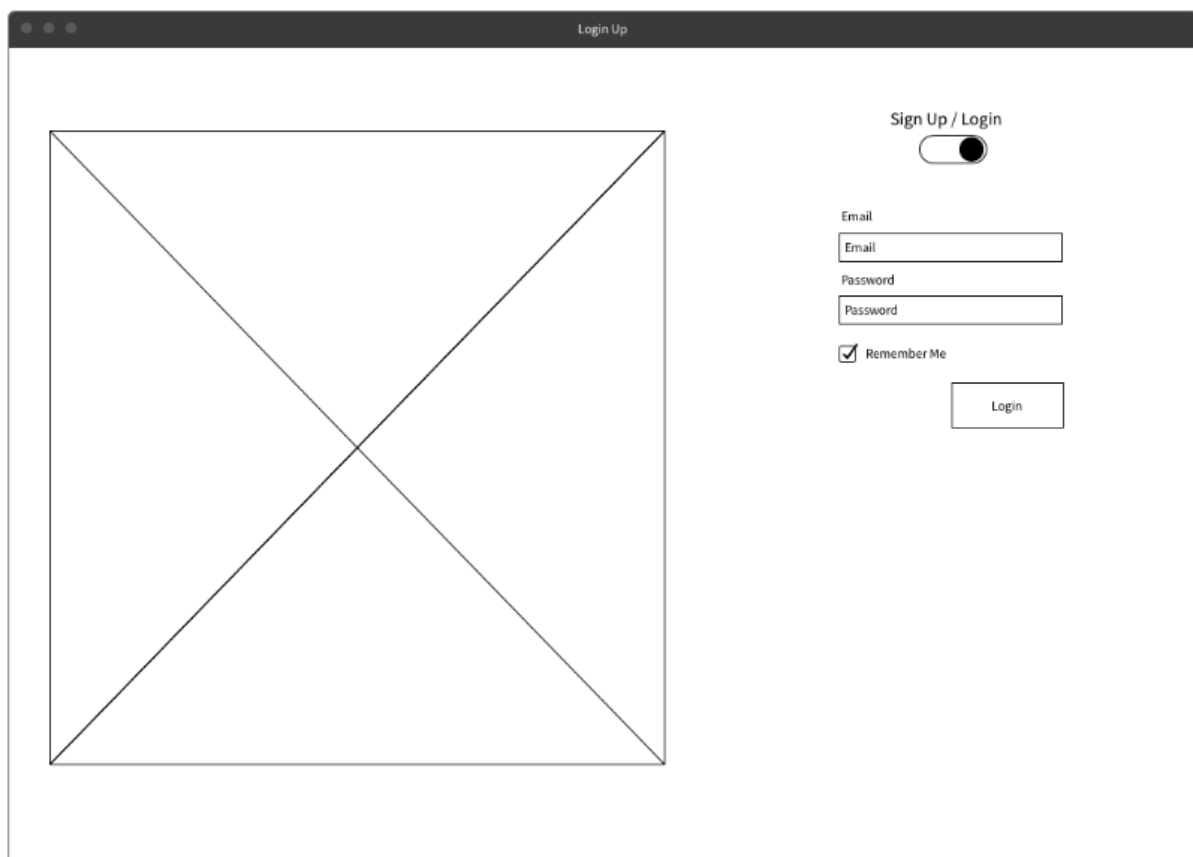
Check whether user exists

```
SELECT person_id
FROM person
WHERE email = @email;
```

Registering person

```
INSERT INTO person (name, surname, birth_date, national_id, gender, email, password,
address)
VALUES(@name, @surname, @birth_date, @national_id, NULL, @email, @password,
@address);
```

## 4.2. Login Page



The screenshot shows a web browser window with the title "Login Up". The main content area is divided into two sections. On the left, there is a large square placeholder with a diagonal "X" drawn across it. On the right, there is a "Sign Up / Login" toggle switch, currently set to "Login". Below the toggle, there are two input fields: "Email" and "Password". Below these fields is a "Remember Me" checkbox, which is checked. At the bottom right of the form is a "Login" button.

The Login Page is the same for all types of users. After the validation, different types of users will be directed to the different types of main pages.

Login Statements for Customer

```
SELECT *  
FROM Customer  
WHERE email = @email AND password = @password;
```

Login Statements for Employee

```
SELECT *  
FROM Employee  
WHERE email = @email AND password = @password;
```

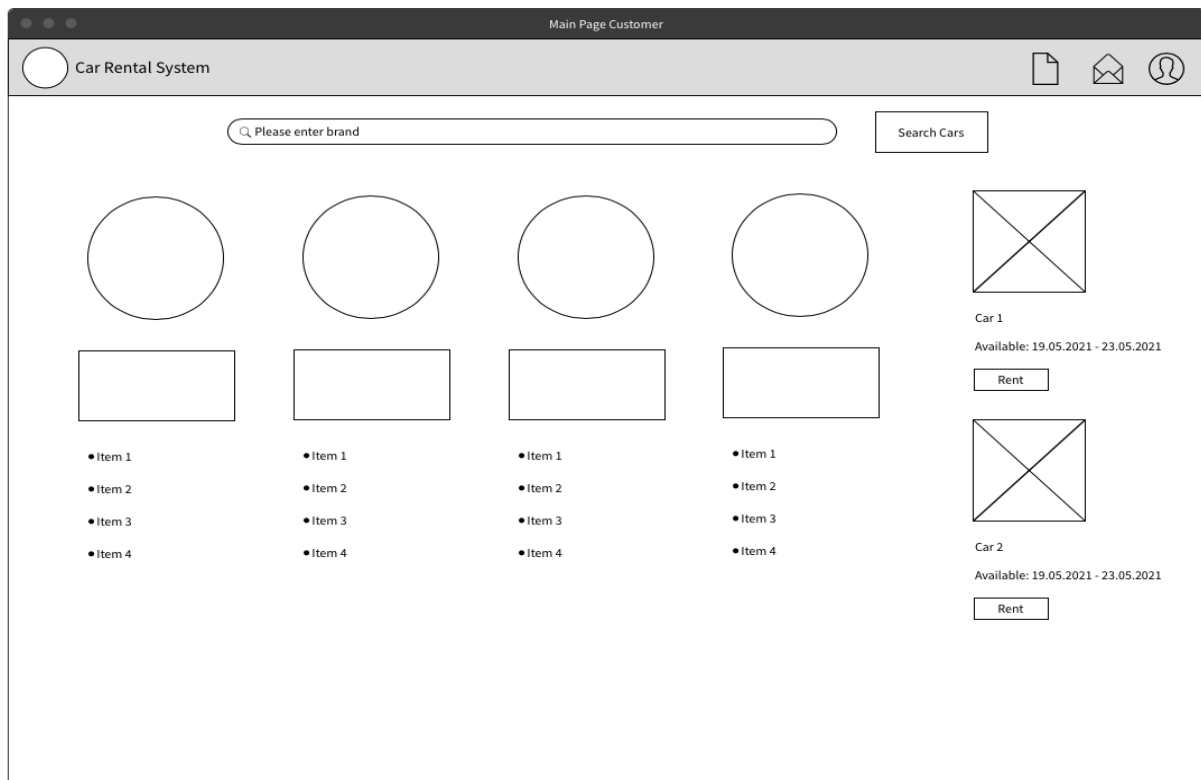
Login Statements for Driver

```
SELECT *  
FROM Driver  
WHERE email = @email AND password = @password;
```

Login Statements for Manager

```
SELECT *  
FROM Manager  
WHERE email = @email AND password = @password;
```

### 4.3. Customer Main Page



The main page of Customer consists of 3 parts. First part is the search bar. Users can type brand, model or color, so filtered outputs (cars) will be displayed to him. The search page is discussed in the following sections. The second part is the company's advertisements. This part displays the cars that are grouped by the brand. And the last part is also advertisements but it is targeted at the customer by displaying cars that are available.

SQL statement for the search bar

```
SELECT *  
FROM Car  
WHERE brand_model = @search_bar OR model_name = @search_bar OR color =  
@search_bar;
```

SQL statement for the advertisement

```
SELECT brand_name FROM Car Grouped By brand_name;  
  
SELECT * FROM Car WHERE brand_name = @brand_name;
```

These queries will be called one after another. Firstly, the brand names will be retrieved from the database and according to those names, the cars will be searched and retrieved (First 4

cars will be displayed in front end. If there is less than 4 cars, the web page will display the number of cars that brand exists in database)

SQL statement for the suggesting car

```
SELECT * FROM Car WHERE car_status = "available";
```

#### 4.4. Search Car Customer

The screenshot shows a web application titled "Search Car Customer" with a "Car Rental System" header. It features a search bar and a "Rent / Transporter" toggle. The left sidebar contains filters for Branch (Istanbul), Brand (Toyota, Volkswagen, Mercedes, BMW), Date (12 May 2016), Start (12 May 2016), End (12 May 2016), Price (From 200, Up To 1000), and Type (Taxi, Ride\_XL, Business\_XL). The main area displays four placeholder cards, each with fields for Car Brand, Car Model, Capacity, Fee, Available dates, and Type, and a "Call" button.

The search for the rents or transporters will be done by retrieving all the cars and filtering according to the user inputs. Since for every filter, a different sql query needs to be implemented, this is a more efficient method to do.

The query that retrieves all the branches

```
SELECT * FROM Branch;
```

The query that retrieves all brand names

*SELECT brand\_name FROM Car GROUP BY brand\_name;*

The query for the rental cars

*SELECT \*  
FROM Car NATURAL JOIN CarDetail  
WHERE brand\_model = @search\_bar OR model\_name = @search\_bar OR color =  
@search\_bar AND branch\_id=@branch\_id;*

The query for the Businexx\_XL

*SELECT \*  
FROM TransportationVehicle NATURAL JOIN Businexx\_XL  
WHERE brand\_model = @search\_bar OR model\_name = @search\_bar OR color =  
@search\_bar AND branch\_id=@branch\_id;*

The query for the transporters cars

*SELECT \*  
FROM TransportationVehicle  
WHERE brand\_model = @search\_bar OR model\_name = @search\_bar OR color =  
@search\_bar AND branch\_id=@branch\_id;*

The query that inserts the request\_ride if user calls transporter

*INSERT INTO request\_ride(employee\_id, customer\_id, ride\_vehicle\_id, date, start\_time,  
end\_time)  
VALUES(@employee\_id, @customer\_id, @ride\_vehicle\_id, @date, @start\_time,  
@end\_time);*

The employee id will be retrieved from the following query

*SELECT e.id  
FROM (        SELECT COUNT(customer\_id) as process\_count, employee\_id as id  
              FROM request  
              GROUP BY employee\_id) as e  
ORDER BY e.process\_count ASC  
LIMIT 1;*

The branch\_id will be taken from the branch's row that is retrieved from the first query. Brand names will also be taken from the second query.

The UI design given above illustrates how the branches will be taken as an input from the user if he/she rents a car. If the user calls for the transporter, then this popup will not be displayed.

Search Car Customer

Car Rental System

Branch

Istanbul

Brand

☒ Toyota

☐ Volkswagen

☐ Mercedes

☒ BMW

Color

☒ Red

☐ Blue

☐ White

☒ Black

Date

From

12 May 2016

To

12 May 2016

Price

From

200

Up To

1000

Car Brand

Car Model

Color

Fee

Km

Available dates

Rent

Please enter information

Branch That Car Will Be Returned:

Next

Rent / Transporter

☒

Search Car Customer

Car Rental System

Branch

Istanbul

Brand

☒ Toyota

☐ Volkswagen

☐ Mercedes

☒ BMW

Color

☒ Red

☐ Blue

☐ White

☒ Black

Date

From

12 May 2016

To

12 May 2016

Price

From

200

Up To

1000

Car Brand

Car Model

Color

Fee

Km

Available dates

Rent

Thank you

Your application successfully submitted. Please wait for the approval. We will notify you as soon as possible.

Close

Rent / Transporter

☒

Retrieval of branches and brand names will be retrieved with the same query given above.  
The colors will be assigned automatically and be independent from the backend.

The query that inserts the rental information

```
INSERT INTO request (employee_id, customer_id, branch_id, start_day, end_day)
VALUES(@employee_id, @customer_id, @branch_id, @start_day, @end_day);
```

The query that will retrieve the employee id that has the least processes ongoing.

```
SELECT e.id
FROM (      SELECT COUNT(customer_id) as process_count, employee_id as id
            FROM request
            GROUP BY employee_id) as e
ORDER BY e.process_count ASC
LIMIT 1;
```

The user will be informed as given above related to the successful rental form submission.

## 4.5. Customer Notifications

The screenshot shows a web application titled "Car Rental System" with a "Search Car Customer" header. The interface includes a search bar and several filter sections: "Branch" (Istanbul), "Brand" (Toyota, Volkswagen, Mercedes, BMW), "Color" (Red, Blue, White, Black), "Date" (From: 12 May 2016, To: 12 May 2016), and "Price" (From: 200, Up To: 1000). Each filter section has a corresponding "Rent" button. A notification box in the top right corner, indicated by a red arrow, contains the following text: "Your application is approved.", "You should return the car soon.", and "Your application is denied."

The user will also be able to see the notifications from the notifications part. All the notification sending will be done by the system automatically (inserting new notification by implementing triggers).

The query for retrieval of notifications.

```
SELECT *
```

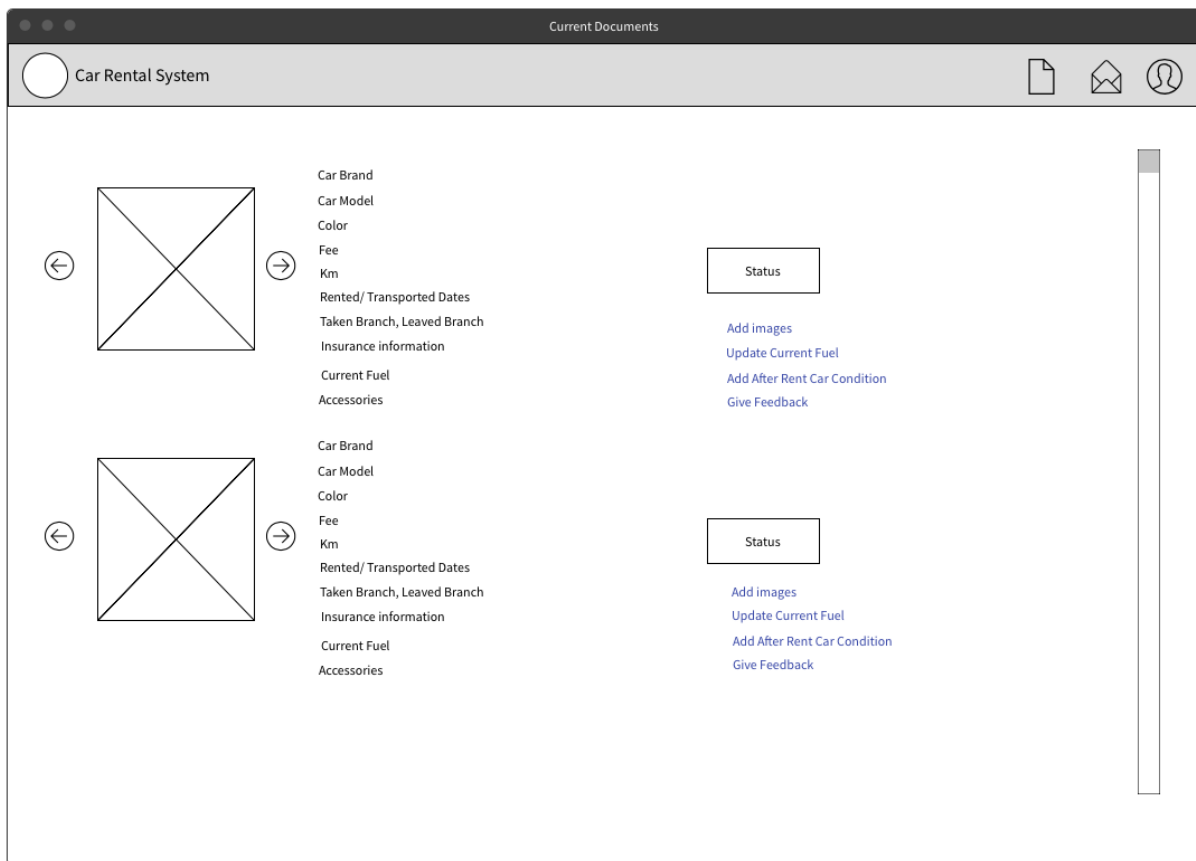
```
FROM Notification
```

```
WHERE person_id = @person_id
```

```
ORDER BY send_date ASC;
```



## 4.6. Customer Current Documents Page



This is the document page where customers can see the rental information (previous rentals as well as in process and ongoing).

The query for the rental information is

```
SELECT *  
FROM RentalInformation NATURAL JOIN Car NATURAL JOIN CarDetail  
WHERE customer_id=@customer_id;
```

```
SELECT *  
FROM request NATURAL JOIN Car NATURAL JOIN CarDetail  
WHERE customer_id=@customer_id;
```

The query for the that adds images

```
UPDATE RentalInformation  
SET image=@image  
WHERE rental_information_id=@rental_information_id;
```

The query for the that adds feedback

```
UPDATE RentalInformation  
SET feedback=@feedback  
WHERE rental_information_id=@rental_information_id;
```

The query for the that adds current fuel

```
UPDATE RentalInformation  
SET current_fuel=@current_fuel  
WHERE rental_information_id=@rental_information_id;
```

The query for the that adds after\_car\_rent\_condition

```
UPDATE RentalInformation  
SET after_car_rent_condition=@after_car_rent_condition  
WHERE rental_information_id=@rental_information_id;
```

The queries given 4 for update RentalInformation are only available if the rental request is approved. If the process is ongoing, then those links will not appear.

The information about image, current\_fuel, car\_condition and feedback will be taken from the popups. We did not include those popups since it would have taken a lot of space.

The customer\_id will be taken from the Session that is saved in login.

## 4.7. Employee Main Page

The screenshot shows the 'Main Page Employee' interface for the 'Car Rental System'. At the top, there is a search bar with the placeholder text 'Please enter brand' and a 'Create Form' button. Below the search bar, there are two identical request cards. Each card contains two placeholder images (represented by boxes with an 'X'), a list of fields: 'Customer name, Customer Type', 'Car brand, model', 'Rent Date', 'Customer National ID', 'Customer License', 'Customer point', 'Contact Number', 'Taken From - Will be Returned To', and 'Rental Type'. To the right of each card are 'Decline' and 'Approve' buttons.

This is the main employee page where employees can approve or decline a car rental request. For the request, the information in customer and car needs to be retrieved in respect to the requests.

The query for the rental requests is (start\_day and end\_day shown as Rent Date):

```
SELECT *  
FROM request  
WHERE employee_id = @person_id;
```

The query for getting the taken from and will be returned to branch ids (The taken from branch is the employee's branch and the returned to branch is the branch\_id which is gotten from the above request query's branch\_id):

```
SELECT branch_id, branch_name  
FROM Employee NATURAL JOIN Branch  
WHERE employee_id = @person_id;
```

The above query gets the taken from branch id with its name.

```
SELECT branch_name
FROM branch
WHERE branch_id = @branch_id;
```

The above query is used to get branch\_name from the branch\_id in the first query to find the returned branch.

The query for getting customer information whose request is shown (each customer\_id is gotten from the above SQL query):

```
SELECT name, surname, national_id, contact_number
FROM Person
WHERE person_id = @customer_id;
```

For getting the customer type if it is premium or not, the customer\_ids from PremiumCustomer will be gotten and the program in the front-end will check whether the customer\_id is in that list or not:

```
SELECT person_id
FROM PremiumCustomer
```

The query about getting car information whose request is shown (each car\_id is gotten from the first SQL query):

```
SELECT brand_name, model_name
FROM Car
WHERE car_id = @car_id;
```

The query about getting the license information of the customer whose id is gotten from the first query:

```
SELECT *
FROM License NATURAL JOIN Customer
WHERE customer_id = @customer_id;
```

For the requests that the employee declines there is nothing to be done except deleting the request from the request query.

```
DELETE FROM request WHERE employee_id = @employee_id AND customer_id =  
@customer_id AND car_id = @car_id AND branch_id = @branch_id;
```

For the request that is accepted it needs to be deleted from the request table and added to the rent table.

```
DELETE FROM request WHERE employee_id = @employee_id AND customer_id =  
@customer_id AND car_id = @car_id AND branch_id = @branch_id;
```

```
INSERT INTO RentalInformation(rental_information_id, start_day, end_day, total_fee,  
current_fuel, image, accessories, car_id, customer_id, employee_id,  
after_car_rent_condition) VALUES(@start_day, @end_day, @total_fee, @current_fuel,  
@image, @accessories, @car_id, @customer_id, @employee_id, NULL, NULL);
```

The variables in RentalInformation are obtained from the variables in the request table and other queries.

## 4.8. Employee Create Form Page

The screenshot shows a web application window titled "Create Form Page Employee" for a "Car Rental System". The form is organized into two columns. The left column contains input fields for "Name", "Surname", and "Email". The right column contains fields for "Start Date" and "End Day" (both pre-filled with "12 May 2016" and featuring calendar icons), "Rental Type", "Car" (with a "Choose Car" button), "Given Branch", "Returned Branch", "Insurance" (with "Yes" and "No" radio buttons, where "No" is selected), "Images", and "Accessories". A "RENT CAR" button is located at the bottom right of the form area.

This is the creating form page of employees where employees can create a form for rental car requests to someone who wants to come to the branch and rent a car.

Car Rental System

Create Form Page Employee

Name:

Surname:

Email:

Start Date:

End Day:

Car:

**Warning**

Please enter dates before selecting car. By the specific dates, the system will show you the available cars.

This is the same page above where giving a pop-up message about selecting dates for renting a car. This page does not do any SQL queries since it shows a warning message.

Car Rental System

Create Form Page Employee

Name:

Surname:

**Choose Car**

☒ ☐

Car Brand  
Car Model  
Color  
Fee  
Km  
Available dates

Car Brand  
Car Model  
Color  
Fee  
Km  
Available dates

This is the same page above where giving a pop-up message about selecting available cars on the selected date for renting a car.

```
SELECT *
FROM Car
WHERE car_id IN (SELECT car_id
                  FROM RentalInformation
                  WHERE @start_day > end_day)
UNION ALL car_id IN (SELECT car_id
                     FROM RentalInformation
                     WHERE @start_day < start_day AND @end_day < start_day)
UNION ALL car_id NOT IN (SELECT car_id
                        FROM RentalInformation);
```

## 4.9. Customer Profile Page

The screenshot displays the 'Profile Page Customer' interface. The header includes the 'Car Rental System' logo and navigation icons. The main content area is divided into three sections:

- Profile Management:** Includes a profile picture placeholder with 'Remove Image' and 'Change Image' options, a 'Customer Type' dropdown, and a 'Change Password' button.
- Personal Information Form:** Contains input fields for Name (Ali), Surname (Yilmaz), Email (test@carrental.tr), Birth Day (12 May 2016), Gender, Licence Type, Licence Given Date, Contact Number, and Address. An 'Edit/Save' button is located at the top right of this section.
- Previous Rental Information:** Displays two rental records. Each record includes a placeholder for a rental image and a list of details: Car Brand, Car Model, Color, Total Fee, Rented/Transported Dates, Car Rental Condition, Feedback, and Rental Type.

This is the profile page of the customer where the customer can change or edit his/her datas in the web application. Also, on this page, customers can see his/her previous rental information.

The query that retrieves all information that belongs to the customer.

```
SELECT *
FROM Customer
WHERE person_id = @person_id;
```

The query that updates information that belongs to the customer.

```
UPDATE Customer INNER JOIN License
SET name = @name, surname = @surname, email = @email, birth_date = @birth_date,
gender = @gender, license_type = @license_type, given_date = @license_given_date,
contact_number = @contact_number, address = @address
WHERE email = @email;
```

## 4.10. Employee Profile Page

This is the profile page where the employee can see his information and change his password. Also, he can see the rental information that they used to accept.

The red part that stores shift\_hour is only visible if the profile is being displayed belonging to the driver.

The green part is visible only for the manager and person to himself/herself that profile belongs.

The sql statement that retrieves information if the profile belongs to *Driver*.



```
SELECT *  
FROM Person NATURAL JOIN Driver,  
WHERE person_id=@person_id;
```

The query that retrieves all information that belongs to the *Employee*.

```
SELECT *  
FROM Person NATURAL JOIN Employee,  
WHERE person_id=@person_id;
```

The query that retrieves all information that belongs to the *Manager*.

```
SELECT *  
FROM Person NATURAL JOIN Manager,  
WHERE person_id=@person_id;
```

The query that updates personal information of the user.

```
UPDATE Person  
SET name=@name, surname=@surname, birth_date=@birth_date,  
national_id=@national_id, national_id=@gender, national_id=@emai, address=@address,  
contact_number=@contact_number)  
WHERE person_id=@person_id;
```

The query that updates the password of Person

```
UPDATE Person  
SET password=@password  
WHERE person_id=@person_id;
```

The query that updates the password of the Driver

```
UPDATE Driver  
SET shift_hours=@shift_hours  
WHERE person_id=@person_id;
```

The query that updates password of Staff..

```
UPDATE Staff  
SET salary=@salary, allowed_leave_number=@allowed_leave_number,  
work_hours_per_day=@work_hours_per_day  
WHERE person_id=@person_id;
```

This is the profile page of the employee where the employee can change or edit his/her datas in the web application. Also, on this page, an employee can see his/her previous rental information.

## 4.11. Manager Main Page

This page will display all the employers and cars that belong to the branch.

The query that retrieves branch\_id  
**SELECT** branch\_id  
**FROM** branch  
**WHERE** manager\_id=@manager\_id;

The query that retrieves all cars that belong to the manager's branch  
**SELECT** \*  
**FROM** Car  
**WHERE** branch\_id=@branch\_id;

The query that retrieves all employees that belong to the manager's branch  
**SELECT** \*  
**FROM** Employee  
**WHERE** branch\_id=@branch\_id;

The query that retrieves all drivers that belong to the manager's branch

```
SELECT *
FROM Driver
WHERE branch_id=@branch_id;
```

## 4.12. Create Employee Page

This is the adding employees to the branch page where the manager can add a specific employee to the branch.

SQL that checks existence of the person in the system according to the national idl number

```
SELECT *
FROM Person
WHERE national_id=@national_id;
```

SQL statement that inserts new employee to the brand

```
INSERT INTO Person(name, surname, birth_date, national_id, gender, email, password,
address, contact_number)
VALUES (@name, @surname, @birth_date, @national_id, @gender, @email, @password,
@address, @contact_number);
```

The query that retrieves the person\_id inserted

```
SELECT person_id
FROM Person
ORDER BY person_id DESC
```

*LIMIT 1;*

The query that inserts staff.

```
INSERT INTO Staff(person_id, salary, allowed_leave_number, work_hours_per_day)  
VALUES(@person_id, @salary, @allowed_leave_number, @work_hours_per_day);
```

The query that inserts the driver if toggle is on.

```
INSERT INTO Driver(person_id, shift_hours, branch_id)  
VALUES(@person_id, @shift_hours, @branch_id);
```

The query that inserts the employee if toggle is on.

```
INSERT INTO Employee(person_id, shift_hours, branch_id)  
VALUES(@person_id, @shift_hours, @branch_id);
```

## 4.13. Add Car Page

The screenshot shows a web application window titled "Add Car Page Manager". The header bar includes a "Car Rental System" logo and navigation icons. The main content area is divided into two sections. On the left, there is a placeholder for a car image with a square box containing an 'X' and the text "Remove Image | Change Image". On the right, there are two columns of input fields for car details. The first column includes fields for Brand, Model, Color, Current km, Plate, Fee, Permit Serial Number, Insurance Serial Number, and Car Condition. The second column includes fields for Fuel Consumption Type, Fuel Consumption Rate, Seating Capacity, Production Year, Description, Deposit, and Transmission Type. An "Add Car" button is positioned at the bottom right of the form.

This is the adding car to the branch page where the manager can add a specific car to the branch.

SQL that checks existence of the car in the system according to the permit serial number or plate

```
SELECT *  
FROM car  
WHERE plate=@plate OR permit_serial_number=@permit_serial_number;
```

SQL statement that inserts new car to the brand

```
INSERT INTO Car(brand_name, model_name, current_km_value, plate, car_status,  
permit_serial_number, insurance_serial_number, car_condition, branch_id)  
VALUES (@brand_name, @model_name, @current_km_value, @plate, @car_status,  
@permit_serial_number, @insurance_serial_number, @car_condition, @branch_id);
```

The query that retrieves the car\_id inserted

```
SELECT car_id  
FROM Car  
ORDER BY car_id DESC  
LIMIT 1;
```

The query that inserts car details.

```
INSERT INTO CarDetail(car_id, car_detail_id, color, fuel_consumption_type,  
fuel_consumption_rate, seating_capacity, production_year, description, car_daily_fee,  
car_deposit, transmission_type)  
VALUES(car_id, @color, fuel_consumption_type, @fuel_consumption_rate,  
@seating_capacity, @production_year, @description, @car_daily_fee, @car_deposit,  
@transmission_type);
```

## 5. Implementation Plan

We are planning to use the React JS framework of JavaScript for the frontend and we will use Dijkstra MySQL as the database. For the backend, we will use PHP, and also we are planning to use RESTful API for the connections of the frontend and backend.