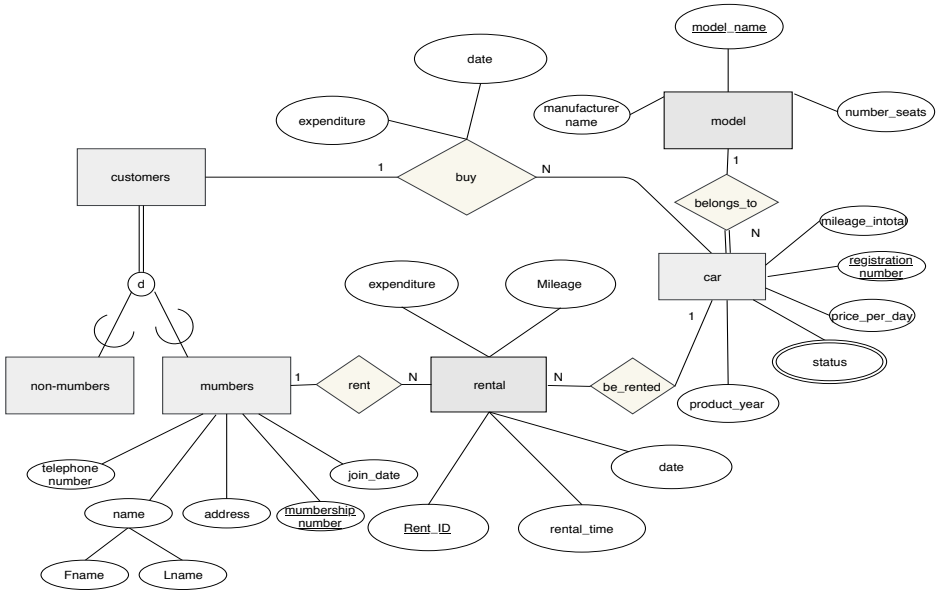
1.

There are two kinds of customers:mumbers and non-mumbers, So I created three entities and the former one is the supertype of the other two.
Then I created rental entity which presents every single record of rental, as only mumbers can rent cars, so mumbers have a relation with rental, as one customer can rent several cars but one car can not be rented by several customers, so it is a 1:N relationship
Then I created car entity, as the company will often buy several cars of the same model, Instead of adding new attributes to car entity, I created another entity called model which has a 1:N relationship to car to reduce redundancy
As in every single record of rental, it can rent one car at a time but one car can be rented many times, so rental has a N:1 relationship with car
All customers (no matter mumbers or not) can buy a car, so I create an relationship between customers and car, as one customer can buy several cars, but one car can only bought by a customer, so it is a 1:N relationship.
Apart from the attributes mentioned in the description, I add a new attribute in car called status which indicates the condition of a car :available_rent ,rented, available_sell,sold, so when some one wants to rent a car or buy a car, he can search the condition of the car based on the status attribute and check its availability. When a car is sold or rented, we only need to change the value of this attribute.
I also created date attributes in rental entity and buy relation, so we can find certain information based on the give date of selling or renting



2.

1. Mapping Regular Entity Types
Car: R_number->Product date,Price_per_day,Mileage_intotal
Model: Model_name->Manufacture_name,Number_seats
Mumbers: M_number-> Address, Fname, Lname, Tel_number, Join_date
Rental: Rent_ID-> Rental_time, Mileage,Rental_expenditure,rent_date
There are four strong entities in the ER diagram, so I created four relations,
Registration number is unique for a single car, so I let it to be the primary key of the car
If the model name is decided, the manufacture name and the number of seats are decided as well.
Mumbership number is unique for a single mumber(unlike name which may be seem among several mumbers), so I let it to be the primary key of the mumber
if we use foreign keys (Mumbership number and Registration number)as the primary key of rental, actually one mumber could rent the same car several times, so our primary key can not distinguish them. So I set a new attribute called Rent_ID which specifies a single record of rental and let it to be the primary key.
2. Mapping Weak Entity Types
No week entity in my design
3. Mapping 1:1 Relationships
No 1:1 relationship in my design
4. Mapping 1:N Relationships
Car has N:1 relationship with model, so add the primary key of model(model name) as a foreign key in car
Car has N:1 relationship with customer, so add the simple attributes of the relationship(buy expenditure,buy date) as foreign keys in car
Rental has N:1 relationship with car and mumbers, so add the primary key of car(registration number) and mumbers(membership number) as foreign keys in rental

Car: R_number->Product date,Price_per_day,Mileage_intotal, Model_name,
Buy_expenditure,Buy_date
Model: Model_name->Manufacture_name,Number_seats
Mumbers: M_number-> Address, Fname, Lname, Tel_number, Join_date
Rental: Rent_ID-> Rental_time, Mileage,Rental_expenditure, Rent_date,
R_number,M_number
5. Mapping M:N Relationships
No M:N Relationships in my design
6. Mapping Multivalued Attributes
status in car is a multivalued attribute, this attribute has four possible values: available_rent,rented,available_sell,sold, if we imagine the true condition of a car,

first, a car can only be rented, but after a period of time, it could be sold to every one, but at that time the car can also be rented, so the status has two value:available_sell,available_rent
Based on that, we have to create a new relation called status, we let the attribute itself(Status) and the primary key of the entity car(registration number) to be the primary key of the new relation

Car: R_number->Product date,Price_per_day,Mileage_intotal, Model_name,
Buy_expenditure,Buy_date
Model: Model_name->Manufacture_name,Number_seats
Mumbers: M_number-> Address, Fname, Lname, Tel_number, Join_date
Rental:Rent_ID->Rental_time,Mileage,Rental_expenditure,Rent_date,R_number,
M_number
Status: R_number,Status
7. Mapping N-ary Relationships
No N-ary Relationships in my design
8. Mapping Supertypes/Subtypes
Although in my design, I created subtype of customers called mumbers and non-mumbers, but only mumbers has some attributes, so we don't need to create new relation of subtypes.

Final version:
Car: R_number->Product date,Price_per_day,Status,Mileage_intotal, Model_name,
Buy_expenditure, Buy_date
Model: Model_name->Manufacture_name,Number_seats
Mumbers: M_number-> Address, Fname, Lname, Tel_number, Join_date
Rental:Rent_ID->Rental_time,Mileage,Rental_expenditure,Rent_date,R_number,
M_number
Status: R_number,Status

3.

```
CREATE TABLE mumbers(
    MumbershipNo   INT(8)   NOT NULL   PRIMARY KEY   UNIQUE,
    Address        CHAR(30) ,
    FName          CHAR(15),
    LName          CHAR(15),
    TelNumber      CHAR(11),
    JoinDate       DATE
);

CREATE TABLE model(
    ModelName   CHAR(20)   NOT NULL   PRIMARY KEY     UNIQUE,
```

```
    ManufactureName    CHAR(15),
    NumberSeats    INT(3)
);

CREATE TABLE car(
    RegistrationNumber    INT(10)    NOT NULL PRIMARY KEY    UNIQUE,
    ProductDate    DATE,
    PricePerDay    INT(5),
    MileageInTotal    DECIMAL(10,3),
    ModelName    CHAR(20),
    BuyExpenditure    INT(8),
    BuyDate    DATE,
    FOREIGN KEY(ModelName) REFERENCES model(ModelName)
     ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE status(
    RegistrationNumber    INT(10)    NOT NULL,
    Status ENUM('available_rent' ,'rented','available_sell','sold')    NOT NULL,
    FOREIGN KEY(RegistrationNumber)    REFERENCES
    car(RegistrationNumber) ON UPDATE CASCADE ON DELETE CASCADE,
    PRIMARY KEY(RegistrationNumber, Status)
);
CREATE TABLE rental(
    RentID    INT(10)    NOT NULL    PRIMARY KEY    UNIQUE,
    RentalTime    INT(3),
    Mileage    DECIMAL(10,3),
    RentalExpenditure    INT(7),
    RentDate    DATE,
    RegistrationNumber    INT(10),
    FOREIGN KEY(RegistrationNumber)    REFERENCES
    car(RegistrationNumber) ON UPDATE CASCADE ON DELETE CASCADE,
    MumbershipNo    INT(8),
    FOREIGN KEY(MumbershipNo)    REFERENCES
    mumbers (MumbershipNo) ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.

• How much money did the company earn from selling cars in 2015?

```
SELECT SUM(BuyExpenditure) FROM car WHERE BuyDate LIKE '2015%';
```

• Which rental car has driven the furthest distance in total?

```
SELECT RegistrationNumber FROM car WHERE MileageInTotal = (SELECT
MAX(MileageInTotal) FROM car);
```

• Which member has driven rented cars the furthest distance in total?

SELECT DISTINCT FName,LName FROM mumbers INNER JOIN rental USING(MumbershipNo) where MumbershipNo= (select MumbershipNo FROM rental GROUP BY RentID ORDER BY(SUM(Mileage)) DESC LIMIT 1);

• Which day(s) did the company have the most rentals?

SELECT RentDate,COUNT(*) from rental GROUP BY RentDate HAVING COUNT(*) = (SELECT COUNT(RentID) from rental GROUP BY RentDate ORDER BY(COUNT(RentID)) DESC LIMIT 1);

• How many cars of each model does the company have available for rent?

SELECT COUNT(RegistrationNumber), ModelName FROM car inner join Model using(ModelName) inner join status using(RegistrationNumber) where Status = 'available_rent' GROUP BY ModelName;

• Who was the 10th member to join? (Hint: the LIMIT clause may be useful here)

SELECT FName,LName, MumbershipNo FROM mumbers ORDER BY JoinDate ASC LIMIT 9,1;

• What car is the cheapest to rent?

SELECT RegistrationNumber,ModelName FROM car where PricePerDay = (select min(PricePerDay) from car);

• How old is the company's oldest car? (Hint: MySQL's date and time functions may be useful here)

SELECT RegistrationNumber,ModelName, TIMESTAMPDIFF(YEAR,ProductDate,NOW()) FROM car where ProductDate =(select MIN(ProductDate) from car);

• Which cars have never been rented?

SELECT RegistrationNumber FROM rental right JOIN car USING(RegistrationNumber) WHERE RentID is null;

• A customer wants to rent a car that can hold at least 4 people, is less than 2 years old and costs less than 1,000 RMB per day to rent. What suitable cars are available?

select RegistrationNumber,ModelName from car inner join model using(ModelName) where NumberSeats >= 4 && PricePerDay < 1000 && TIMESTAMPDIFF(DAY,ProductDate,NOW()) < 365*2；