Βάσεις Δεδομένων

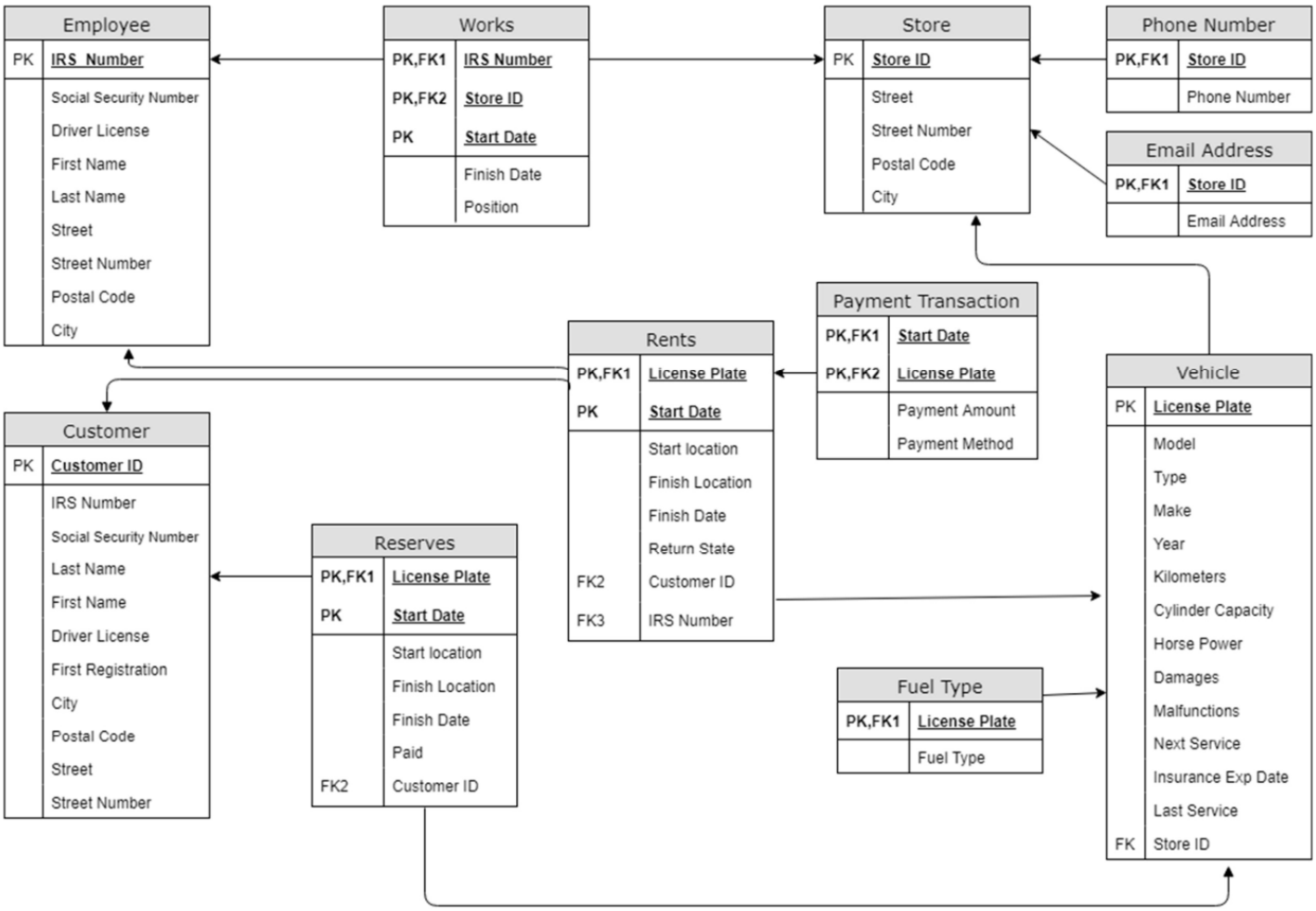
ΕΞΑΜΗΝΑΙΟ PROJECT

ΑΛΕΞΗΣ ΟΡΦΕΑΣ ΚΟΥΡΚΑΚΗΣ 03114014

ΒΑΣΙΛΕΙΟΣ ΠΑΝΑΓΙΩΤΟΠΟΥΛΟΣ 03114303

Εισαγωγή

Η παρούσα εργασία αφορά στην ανάπτυξη σε σχεσιακό σύστημα μιας βάσης δεδομένων που αφορά την επιχείρηση ενοικίασης οχημάτων. Το σχεσιακό μοντέλο όπως χρησιμοποιήθηκε απ' τη λύση της 1ης άσκησης είναι το παρακάτω:



Εργαλεία που χρησιμοποιήθηκαν

Java Swing - Eclipse

Ο σχεδιασμός του γραφικού περιβάλλοντος της εφαρμογής έγινε με τη γλώσσα Java και τη χρήση της βιβλιοθήκης Swing, η οποία παρέχει έτοιμες συναρτήσεις σχεδιασμού κουμπιών, πινάκων και άλλων χρήσιμων γραφικών στοιχείων και με αυτό τον τρόπο διευκολύνει τον προγραμματιστή. Επίσης, έγινε χρήση του IDE Eclipse το οποίο δίνει τη δυνατότητα καλύτερης διαχείρισης μεγάλων προγραμματιστικών project που αποτελούνται από πολλά αρχεία. Το συγκεκριμένο IDE περιλαμβάνει το plugin WindowBuilder που δίνει στον προγραμματιστή τη δυνατότητα, έως ένα βαθμό, να δημιουργήσει το γραφικό περιβάλλον της εφαρμογής με μεθόδους drag' n' drop, διευκολύνοντας λίγο το σχεδιασμό της εφαρμογής.

MySQL + Workbench

Η πλατφόρμα διαχείρισης και ανάπτυξης βάσεων δεδομένων MySQL επιλέχθηκε επειδή αποτελεί open-source dbms με ευρεία χρήση σε πραγματικές εφαρμογές και συνεπώς υπάρχει αρκετές πηγές πληροφοριών. Επιπλέον,έχει cross-platform λειτουργικότητα και σε συνδυασμό με το λογισμικό MySQL Workbench προσφέρει έναν αρκετά φιλικό τρόπο ανάπτυξης μιας βάσης δεδομένων προς το σχεδιαστή.

Προσιορισμός Προδιαγραφών – Περιορισμοί

Στις σχέσεις Customer και Employee τα πεδία Social Security Number και IRS Number τα θέσαμε Unique (αν και στον Employee είναι και Primary key το πεδίο IRS Number) γιατί θεωρήσαμε ότι δε γίνεται δύο άτομα να έχουν το ίδιο SSN ή IRSN. Επίσης, όλα τα primary keys είναι μοναδικά. Όσον αφορά τα foreign keys, χρησιμοποιήθηκε ON DELETE CASCADE ώστε κάθε φορά που διαγράφεται μια εγγραφή από table που το primary key της είναι foreign σε άλλη, να διαγράφεται και η αντίστοιχη καταχώριση στη δεύτερη.

Σχεδιασμός Βάσης

Η σχεδίαση της βάσης έγινε με βάση την προτεινόμενη λύση της 1ης Άσκησης. Τα primary keys χωρίς φυσική σημασία κάθε πίνακα(πχ Customer ID) παράγονται με αυτόματη αύξηση (auto increment),το οποίο εξασφαλίζει την ακεραιότητα οντότητας (entity integrity) για κάθε εγγραφή. Συγκεκριμένα, αυτό συμβαίνει στους εξής πίνακες: Customer, Store. Για τους υπόλοιπους πίνακες, η ακεραιότητα οντότητας εξασφαλίζεται μέσω των εξωτερικών κλειδιών των υπόλοιπων πινάκων πέρα του table Employee που έχει ως primary key το IRSN που θεωρήθηκε μοναδικό και δίνεται κατά την είσοδο δεδομένων. Όσον αφορά την αναφορική ακεραιότητα (referential integity), εξασφαλίζεται ως εξής: Εάν διαγραφεί μία εγγραφή που το PK της είναι FK σε ένα άλλο table, τότε αυτομάτως διαγράφεται και η εγγραφή που αφορά το ίδιο κλειδί στο άλλο table. Επίσης, εξασφαλίζεται και το domain intergrity με τους κατάλληλους ορισμούς τύπων δεδομένων για κάθε πεδίο, με κάποια επίσης constraints, όπως μη επιτρέποντας σε κάποια πεδία να πάρουν τιμή null. Τέλος, εξασφαλίζεται και η ακεραιότητα user defined, αφού κάθε οντότητα έχει δικά της πεδία, ανεξάρτητου και διαφορετικού τύπου.

Triggers

Τα triggers είναι αντικείμενα στη βάση δεδομένων που είναι συνδεδεμένα με έναν πίνακα. Είναι όπως ένα διαδικαστικό κομμάτι κώδικα, με τη διαφορά ότι ενεργοποιείται μόνο όταν γίνεται κάποιο INSERT, UPDATE ή DELETE στον πίνακα με τον οποίο είναι συνδεδεμένο. Στον προγραμματιστή, εκτός από την δυνατότητα ορισμού της ενέργειας που θα εκτελεστεί, προσφέρεται η επιλογή της σειράς μεταξύ ενέργειας του trigger και INSERT/UPDATE/DELETE.

Trigger 1

Σχετίζεται άμεσα με τη δυνατότητα ενημέρωσης της όψης beforeView1 και κατευθύνει την είσοδο νέων στοιχείων στο κεντρικό κατάστημα (Store ID = 1).

```
DELIMITER //

create trigger beforeView1 before insert on Vehicle
for each row
begin
    if new.`Store ID` = 0  then set new.`Store ID` = 1 ; end if;
end //

DELIMITER ;

/* view beforeView1 updatable (and contains foreign key) ==>
/* trigger for updating view1 (derived from Vehicle table ,
and Vehicle contains foreign key StoreID) ---> we cannot insert StoreID
that does not exist in Store so we put pre-agreed StoreID=1 with null
values in the fields of type Datetime --> so we know it came from an insertion in view1  */
```

Trigger 2

Σχετίζεται με τον αριθμό αμαξιών καθώς πρόκειται για επιχείριση με βλέψεις πελατών πολυτελείας, και έτσι δε διαχειρίζεται ακόμα πάνω από 40 οχήματα, για λόγους διασφάλισης ποιότητας υπηρεσιών.

```
/* luxury buiseness provision ==>  not too many , but quality cars  , so keep track of
the number of Vehicles we have available for renting/reserving
   ( do not order too many cars / quality car - quantity of cars tradeoff */

DELIMITER//

create trigger VehicleLimit before insert on Vehicle
for each row
begin
    if( select COUNT(*) from Vehicle) >= 40 then
    signal sqlstate '45000' set message_text = " CarRent company does not manage/accept more than 40 Vehicles";
    end if ;
END//

DELIMITER ;
```

Views

Ένα view είναι ένας εικονικός πίνακας που προκύπτει από το αποτέλεσμα ενός ερωτήματος sql. Μπορούμε να χρησιμοποιήσουμε μία όψη σαν να είναι ένας πίνακας (για εντολές select). Η όψη δεν απαιτείται να είναι αποθηκευμένη σε φυσική μορφή και το ερώτημα στο οποίο βασίζεται πραγματοποιείται μία φορά.

View 1

Η πρώτη όψη είναι ενημερώσιμη γιατί αναφέρεται σε ένα τραπέζι και περιέχει και το primary key του. Εμφανίζει την πινακίδα, το μοντέλο, τη μάρκα, τον τύπο, τα κυβικά και την ιπποδύναμη κάθε οχήματος.

```
/* view1 updatable */
create view Vehicle-View1 as
    select `License Plate`, Model, Make, Type, `Cylinder Capacity`, `Horse Power`
    from Vehicle
```

View 2

Η δεύτερη όψη δεν είναι ενημερώσιμη καθώς παρουσιάζει στοιχεία από πολλούς πίνακες. Εμφανίζει τα στοιχεία επικοινωνίας κάθε καταστήματος: Πόλη, Τηλέφωνο και Email.

```
/* view 2 non updatable */
create view StoreContactInfo as
select s.`City` , `Phone Number`,
    (select Email_Address from `Email Address` where `Email Address`.`Store ID` = p.`Store ID`)
    from Store s
    inner join `Phone Number` p
    on s.`Store ID` = p.`Store ID` ;
```

DDL κατασκευή της Βάσης

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema CarRent
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema CarRent
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `CarRent` DEFAULT CHARACTER SET utf8 ;
USE `CarRent` ;
```

```
CREATE TABLE IF NOT EXISTS `CarRent`.`Employee` (
  `IRS Number` INT NOT NULL,
  `Social Security Number` INT NOT NULL,
  `Driver License` VARCHAR(45) NOT NULL,
  `First Name` VARCHAR(45) NOT NULL,
  `Last Name` VARCHAR(45) NOT NULL,
  `Street` VARCHAR(45) NOT NULL,
  `Street Number` INT NOT NULL,
  `Postal Code` INT NOT NULL,
  `City` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`IRS Number`),
  UNIQUE INDEX `Social Security Number_UNIQUE` (`Social Security Number` ASC),
  UNIQUE INDEX `Driver License_UNIQUE` (`Driver License` ASC),
  UNIQUE INDEX `IRS Number_UNIQUE` (`IRS Number` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CarRent`.`Customer`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CarRent`.`Customer` ;

CREATE TABLE IF NOT EXISTS `CarRent`.`Customer` (
  `Customer ID` INT NOT NULL AUTO_INCREMENT,
  `IRS Number` INT NOT NULL,
  `Social Security Number` INT NOT NULL,
  `Last Name` VARCHAR(45) NOT NULL,
  `First Name` VARCHAR(45) NOT NULL,
  `Driver License` VARCHAR(45) NOT NULL,
  `First Registration` DATETIME NULL,
  `City` VARCHAR(45) NOT NULL,
  `Postal Code` INT NOT NULL,
  `Street` VARCHAR(45) NOT NULL,
  `Street Number` INT NOT NULL,
  PRIMARY KEY (`Customer ID`),
  UNIQUE INDEX `IRS Number_UNIQUE` (`IRS Number` ASC),
  UNIQUE INDEX `Social Security Number_UNIQUE` (`Social Security Number` ASC),
  UNIQUE INDEX `Driver License_UNIQUE` (`Driver License` ASC),
  UNIQUE INDEX `Customer ID_UNIQUE` (`Customer ID` ASC))
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `CarRent`.`Store`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CarRent`.`Store` ;

CREATE TABLE IF NOT EXISTS `CarRent`.`Store` (
  `Store ID` INT NOT NULL AUTO_INCREMENT,
  `Street` VARCHAR(45) NOT NULL,
  `Street Number` INT NOT NULL,
  `Postal Code` INT NOT NULL,
  `City` VARCHAR(45) NOT NULL,
  UNIQUE INDEX `Store ID_UNIQUE` (`Store ID` ASC),
  PRIMARY KEY (`Store ID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CarRent`.`Vehicle`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CarRent`.`Vehicle` ;

CREATE TABLE IF NOT EXISTS `CarRent`.`Vehicle` (
  `License Plate` VARCHAR(10) NOT NULL,
  `Model` VARCHAR(20) NOT NULL,
  `Type` VARCHAR(20) NOT NULL,
  `Make` VARCHAR(20) NOT NULL,
  `Year` INT NOT NULL DEFAULT 0,
  `Kilometers` INT NOT NULL DEFAULT 0,
  `Cylinder Capacity` INT NOT NULL,
  `Horse Power` INT NOT NULL,
  `Damages` VARCHAR(35) NOT NULL DEFAULT 'N/A',
  `Malfunctions` VARCHAR(35) NOT NULL DEFAULT 'N/A',
  `Next Service` DATE NULL,
  `Insurance Exp Date` DATE NULL,
  `Last Service` DATE NULL,
  `Store ID` INT NOT NULL DEFAULT 0,
  PRIMARY KEY (`License Plate`),
  UNIQUE INDEX `License Plate_UNIQUE` (`License Plate` ASC),
  INDEX `Stode ID_idx` (`Store ID` ASC),
  CONSTRAINT `Store ID Vehicle`
    FOREIGN KEY (`Store ID`)
    REFERENCES `CarRent`.`Store` (`Store ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `CarRent`.`Phone Number`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CarRent`.`Phone Number` ;

CREATE TABLE IF NOT EXISTS `CarRent`.`Phone Number` (
  `Store ID` INT NOT NULL,
  `Phone Number` INT NOT NULL,
  UNIQUE INDEX `Store ID_UNIQUE` (`Store ID` ASC),
  PRIMARY KEY (`Store ID`),
  CONSTRAINT `Store ID Phone`
    FOREIGN KEY (`Store ID`)
    REFERENCES `CarRent`.`Store` (`Store ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CarRent`.`Email Address`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CarRent`.`Email Address` ;

CREATE TABLE IF NOT EXISTS `CarRent`.`Email Address` (
  `Store ID` INT NOT NULL,
  `Email_Address` VARCHAR(35) NOT NULL,
  UNIQUE INDEX `Store ID_UNIQUE` (`Store ID` ASC),
  PRIMARY KEY (`Store ID`),
  CONSTRAINT `Store ID Email Address`
    FOREIGN KEY (`Store ID`)
    REFERENCES `CarRent`.`Store` (`Store ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
-- Table `CarRent`.`Fuel Type`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CarRent`.`Fuel Type` ;

CREATE TABLE IF NOT EXISTS `CarRent`.`Fuel Type` (
  `License Plate` VARCHAR(10) NOT NULL,
  `Fuel Type` VARCHAR(15) NOT NULL,
  UNIQUE INDEX `License Plate_UNIQUE` (`License Plate` ASC),
  PRIMARY KEY (`License Plate`),
  CONSTRAINT `License Plate Fuel Type`
    FOREIGN KEY (`License Plate`)
    REFERENCES `CarRent`.`Vehicle` (`License Plate`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CarRent`.`Reserves`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CarRent`.`Reserves` ;

CREATE TABLE IF NOT EXISTS `CarRent`.`Reserves` (
  `License Plate` VARCHAR(10) NOT NULL,
  `Start Date` DATE NOT NULL,
  `Start Location` VARCHAR(25) NOT NULL,
  `Finish Location` VARCHAR(25) NOT NULL,
  `Finish Date` VARCHAR(25) NOT NULL,
  `Paid` TINYINT(1) NOT NULL,
  `Customer ID` INT NOT NULL,
  PRIMARY KEY (`License Plate`, `Start Date`),
  UNIQUE INDEX `License Plate_UNIQUE` (`License Plate` ASC),
  UNIQUE INDEX `Start Date_UNIQUE` (`Start Date` ASC),
  INDEX `Customer ID_idx` (`Customer ID` ASC),
  CONSTRAINT `Customer ID Reserves`
    FOREIGN KEY (`Customer ID`)
    REFERENCES `CarRent`.`Customer` (`Customer ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `License Plate Reserves`
    FOREIGN KEY (`License Plate`)
    REFERENCES `CarRent`.`Vehicle` (`License Plate`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `CarRent`.`Works`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CarRent`.`Works` ;

CREATE TABLE IF NOT EXISTS `CarRent`.`Works` (
  `IRS Number` INT NOT NULL,
  `Store ID` INT NOT NULL,
  `Start Date` DATE NOT NULL,
  `Finish Date` DATE NULL,
  `Position` VARCHAR(25) NULL,
  UNIQUE INDEX `IRS Number_UNIQUE` (`IRS Number` ASC),
  PRIMARY KEY (`IRS Number`, `Store ID`, `Start Date`),
  UNIQUE INDEX `Start Date_UNIQUE` (`Start Date` ASC),
  CONSTRAINT `IRS Number Works`
    FOREIGN KEY (`IRS Number`)
    REFERENCES `CarRent`.`Employee` (`IRS Number`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `Store ID Works`
    FOREIGN KEY (`Store ID`)
    REFERENCES `CarRent`.`Store` (`Store ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CarRent`.`Rents`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CarRent`.`Rents` ;

CREATE TABLE IF NOT EXISTS `CarRent`.`Rents` (
  `License Plate` VARCHAR(10) NOT NULL,
  `Start Date` DATE NOT NULL,
  `Start Location` VARCHAR(25) NOT NULL,
  `Finish Location` VARCHAR(25) NOT NULL,
  `Finish Date` DATE NOT NULL,
  `Return State` VARCHAR(25) NOT NULL,
  `Customer ID` INT NOT NULL,
  `IRS Number` INT NOT NULL,
  UNIQUE INDEX `License Plate_UNIQUE` (`License Plate` ASC),
  PRIMARY KEY (`License Plate`, `Start Date`),
  UNIQUE INDEX `Start Date_UNIQUE` (`Start Date` ASC),
  INDEX `Customer ID_idx` (`Customer ID` ASC),
  INDEX `IRS Number_idx` (`IRS Number` ASC),
  CONSTRAINT `License Plate Rents`
    FOREIGN KEY (`License Plate`)
    REFERENCES `CarRent`.`Vehicle` (`License Plate`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `Customer ID  Rents`
    FOREIGN KEY (`Customer ID`)
    REFERENCES `CarRent`.`Customer` (`Customer ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `IRS Number  Rents`
    FOREIGN KEY (`IRS Number`)
    REFERENCES `CarRent`.`Employee` (`IRS Number`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CarRent`.`Payment Transaction`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CarRent`.`Payment Transaction` ;

CREATE TABLE IF NOT EXISTS `CarRent`.`Payment Transaction` (
  `Start Date` DATE NOT NULL,
  `License Plate` VARCHAR(10) NOT NULL,
  `Payment Amount` INT NOT NULL,
  `Payment Method` VARCHAR(25) NOT NULL,
  PRIMARY KEY (`Start Date`, `License Plate`),
  UNIQUE INDEX `Start Date_UNIQUE` (`Start Date` ASC),
  UNIQUE INDEX `License Plate_UNIQUE` (`License Plate` ASC),
  CONSTRAINT `Start Date Payment Transactions`
    FOREIGN KEY (`Start Date`)
    REFERENCES `CarRent`.`Rents` (`Start Date`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `License Plate  Payment Transactions`
    FOREIGN KEY (`License Plate`)
    REFERENCES `CarRent`.`Rents` (`License Plate`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

SQL QUERIES

```
/*FIRST QUERY */
/* Employees grouped by Last Name with Street Number > input */

select  `First Name`,`Last Name`,`Street Number` from Employee group by `Last Name` having `Street Number` > ? order by `Street Number` desc;

/* SECOND QUERY  */

/*  show all License Plate and Make(Brand of the vehicle) where Make = the input Make(e.g. AUDI) given */
select `License Plate`, Make from Vehicle where Make like ?;

/* THIRD QUERY */

/* show stores in descending order (by number Vehicles each store has) */

select Store.`Store ID` , Store.`City` , count(Vehicle.`Store ID`)
from Store
right join Vehicle on Store.`Store ID` =  Vehicle.`Store ID`
group by Store.`Store ID`
order by count(Vehicle.`Store ID`) desc ;

/* FOURTH QUERY  */
/* shows each employee's irs number and position  and in what city is the store that he works */

select e.`IRS Number` , City , Position ,
    (select City from Store where Store.`Store ID` = w.`Store ID`)
from Employee  e
inner join Works w
    on e.`IRS Number` = w.`IRS Number`  ;

/* FIFTH QUERY */
/* shows License Plate and Kilometers from the most used Vehicles (the ones with kilometers above average KMs*/

select distinct `License Plate`, Kilometers
from Vehicle
where Kilometers > (select avg(Kilometers) from Vehicle);
/* sixth query   */
/* Customers ordered by First Registration (datetime) */

select c.`First Name`, c.`Last Name`, c.`First Registration`
from Customer c
order by c.`First Registration`;

/*  seventh query */
/* show customers reserved during 2018 summer order by their reserved finish location */

select c.`First Name`, c.`Last Name`, r.`Finish Location`
from Customer c RIGHT JOIN Reserves r ON c.`Customer ID` = r.`Customer ID`
where r.`Start Date` between '2018-06-01' and '2018-08-31'
order by r.`Finish Location`;


/*  8th query  */
/* Rent Store location with the most total paid amount*/

select r.`Start Location`, sum(p.`Payment Amount`)
from Rents as r, `Payment Transaction` as p
where r.`Start Date` = p.`Start Date`
group by r.`Start Location`
order by sum(p.`Payment Amount`) Desc;

/*  9th query  */
/* Show Vehicles' Damages and order by Avg Kilometers*/

select Damages, COUNT(Damages), AVG(Kilometers)
    from Vehicle
    group by Damages
order by AVG(Kilometers) Desc;

/* 10th query  */
/*show Vehicles with fuel type of input*/
select v.`License Plate` from vehicle v
where v.`License Plate` in
(select f.`License Plate` from `Fuel Type`  f where f.`Fuel Type` = ? );
```