MSc in Business Analytics
1st Assignment
Due Date: October 23rd 2016

Eva Giannatou
Email: eva_giannatou@yahoo.com

Contents

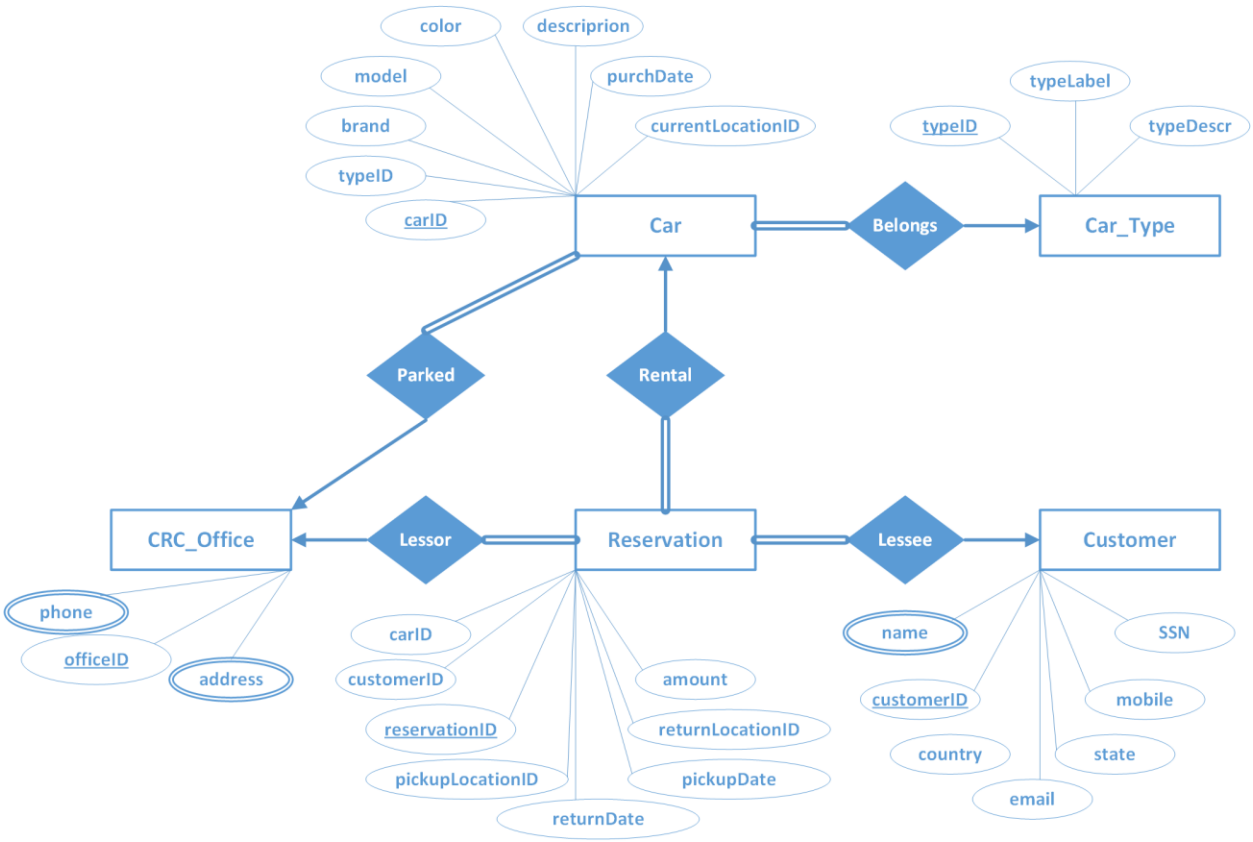
Description of the Problem

A car rental company (let's call it CRC) wants to develop a relational database to monitor customers, rentals, fleet and locations.

CRC's fleet consists of cars of different types. A car is described via a unique code (VIN), a description, color, brand, model, and date of purchase. A car may belong to one (exactly one) vehicle category (compact, economy, convertible, etc.). Each category is described by a unique ID, a label and a detailed description. CRC has several locations around the globe. Each location has a unique ID, an address (street, number, city, state, country) and one or more telephone numbers. CRC should also store in this database its customers. A customer is described by a unique ID, SSN, Name (First, Last), email, mobile phone number and lives in a state and country. Customers rent a car, which they pickup from a location and return it another location (not necessarily the same.) A rental is described by a unique reservation number, it has an amount and contains the pickup date and the return date.

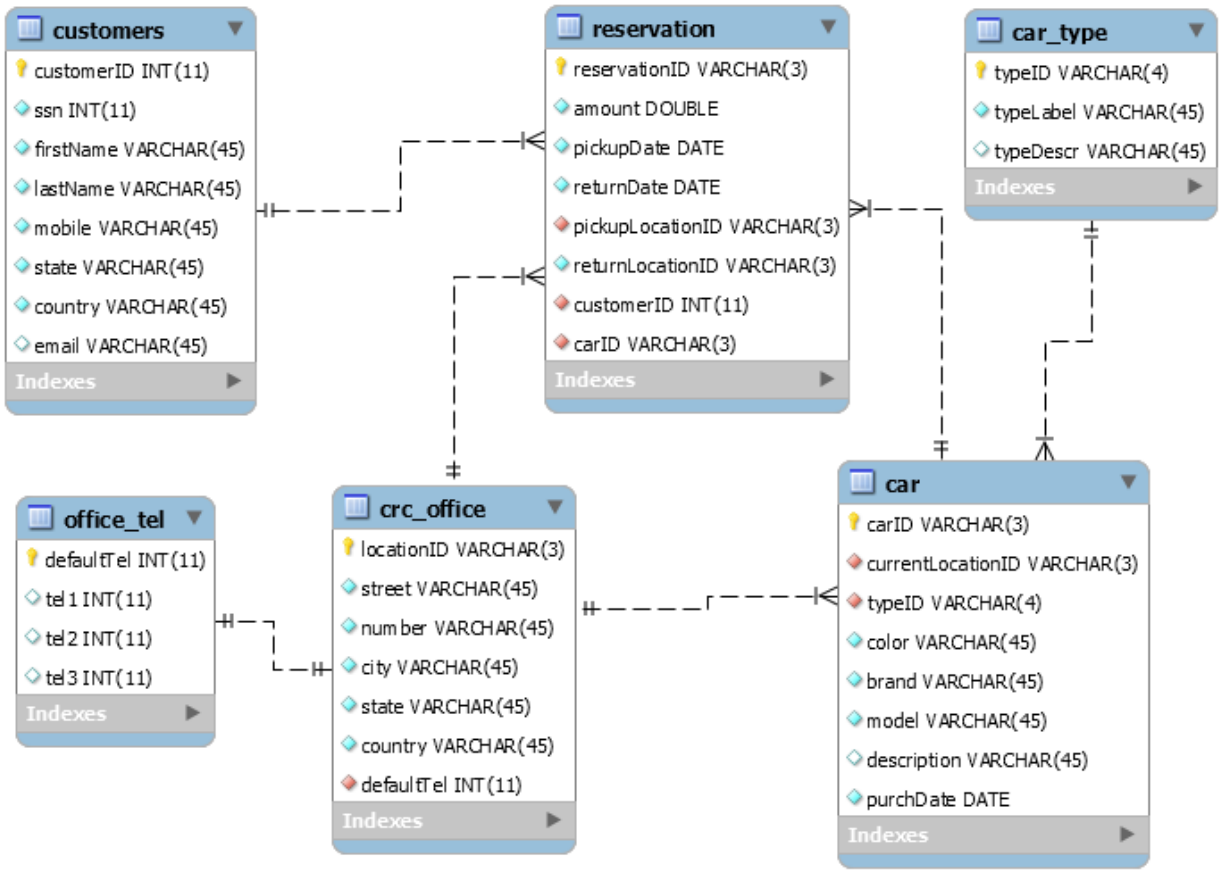
Entity-Relationship Diagram (ERD)

Use the Entity-Relationship Diagram (ERD) to model entities, relationships, attributes, cardinalities, and all necessary constraints. Use any tool you like to draw the ERD.



Database Diagram

Use SQL Workbench to create the tables and insert a few records into the tables to test your queries below. You will have to hand in the CREATE TABLE statements.



Foreign Keys

Car table:

- Car.currentLocationID references crc_office.locationID
- Car.typeID references car_type.typeID

Crc_office table:

- Crc_office.defaultTel references office_tel.defaultTel

Reservation table:

- Reservation.pickupLocationID references crc_office.locationID
- Reservation.customerID references customers.customerID
- Reservation.carID references car.carID

CREATE TABLE statements

Table 1 - car_type

```
'CREATE TABLE `car_type` (
  `typeID` varchar(4) NOT NULL,
  `typeLabel` varchar(45) NOT NULL,
  `typeDescr` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`typeID`),
  UNIQUE KEY `typeID_UNIQUE` (`typeID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8'
```

Table 2 - car

```
'CREATE TABLE `car` (
  `carID` varchar(3) NOT NULL,
  `currentLocationID` varchar(3) NOT NULL,
  `typeID` varchar(4) NOT NULL,
  `color` varchar(45) NOT NULL,
  `brand` varchar(45) NOT NULL,
  `model` varchar(45) NOT NULL,
  `description` varchar(45) DEFAULT NULL,
  `purchDate` date NOT NULL,
 PRIMARY KEY (`carID`),
  UNIQUE KEY `typeID_UNIQUE` (`typeID`),
  UNIQUE KEY `currentLocationID_UNIQUE` (`currentLocationID`),
  UNIQUE KEY `carID_UNIQUE` (`carID`),
  CONSTRAINT `currentLocationID` FOREIGN KEY (`currentLocationID`) REFERENCES
`crc_office` (`locationID`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `locationID` FOREIGN KEY (`currentLocationID`) REFERENCES `crc_office`
(`locationID`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `typeID` FOREIGN KEY (`typeID`) REFERENCES `car_type` (`typeID`) ON
DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8'
```

Table 3 - office_tel

```
'CREATE TABLE `office_tel` (
  `defaultTel` int(11) NOT NULL,
  `tel1` int(11) DEFAULT NULL,
  `tel2` int(11) DEFAULT NULL,
  `tel3` int(11) DEFAULT NULL,
  PRIMARY KEY (`defaultTel`),
  UNIQUE KEY `defaultTel_UNIQUE` (`defaultTel`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8'
```

Table 4 - crc_office

```
'CREATE TABLE `crc_office` (
  `locationID` varchar(3) NOT NULL,
  `defaultTel` int(11) NOT NULL,
  `street` varchar(45) NOT NULL,
  `number` varchar(45) NOT NULL,
  `city` varchar(45) NOT NULL,
  `state` varchar(45) NOT NULL,
  `country` varchar(45) NOT NULL,
  PRIMARY KEY (`locationID`),
  UNIQUE KEY `locationID_UNIQUE` (`locationID`),
  UNIQUE KEY `defaultTel_UNIQUE` (`defaultTel`),
  KEY `defaultTel_idx` (`defaultTel`),
  CONSTRAINT `defaultTel` FOREIGN KEY (`defaultTel`) REFERENCES `office_tel`
(`defaultTel`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8'
```

Table 5 - customers

```
'CREATE TABLE `customers` (
  `customerID` int(11) NOT NULL,
  `ssn` int(11) NOT NULL,
  `firstName` varchar(45) NOT NULL,
  `lastName` varchar(45) NOT NULL,
  `mobile` varchar(45) NOT NULL,
  `state` varchar(45) NOT NULL,
  `country` varchar(45) NOT NULL,
  `email` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`customerID`),
  UNIQUE KEY `ssn_UNIQUE` (`ssn`),
  UNIQUE KEY `customerID_UNIQUE` (`customerID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8'
```

Table 6 - reservation

```
'CREATE TABLE `reservation` (
  `reservationID` varchar(3) NOT NULL,
  `amount` double NOT NULL,
  `pickupDate` date NOT NULL,
  `returnDate` date NOT NULL,
  `pickupLocationID` varchar(3) NOT NULL,
  `returnLocationID` varchar(3) NOT NULL,
  `customerID` int(11) NOT NULL,
  `carID` varchar(3) NOT NULL,
  PRIMARY KEY (`reservationID`),
  UNIQUE KEY `reservationID_UNIQUE` (`reservationID`),
  KEY `pickupLocationID_idx` (`pickupLocationID`),
  KEY `customerID_idx` (`customerID`),
  KEY `carID_idx` (`carID`),
  CONSTRAINT `carID` FOREIGN KEY (`carID`) REFERENCES `car` (`carID`) ON DELETE NO
ACTION ON UPDATE NO ACTION,
  CONSTRAINT `customerID` FOREIGN KEY (`customerID`) REFERENCES `customers`
(`customerID`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `pickupLocationID` FOREIGN KEY (`pickupLocationID`) REFERENCES
`crc_office` (`locationID`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8'
```

Queries

Write SQL code and test it to your data for the following queries

Query A

a. Show the reservation number and the location ID of all rentals in 5/20/2015

```
SELECT reservationID, pickupLocationID
FROM reservation
WHERE pickupDate = '2015-05-01';
```

Query B

b. Show the first and the last name and the mobile phone number of those customers that have rented a car in the category that has label = 'luxury'

```
SELECT DISTINCT c.firstName, c.lastName, c.mobile
FROM customers AS c, reservation AS r, car_type AS t, car
WHERE c.customerID=r.customerID AND r.carID=car.carID AND car.typeID=t.typeID AND
t.typeLabel='Luxury';
```

Query C

c. Show the total amount of rentals per location ID (pick up)

```
SELECT pickupLocationID, COUNT(carID)
FROM reservation
GROUP BY pickupLocationID;
```

Query D

d. Show the total amount of rentals per car's category ID and month

```
select car.typeID,  extract(year from r.pickupDate) "Year",
      extract(month from r.pickupDate) "Month", count(r.carID) "No. of Cars"
from  reservation as r, car
where r.carID=car.carID
group by car.typeID, year, month
order by car.typeID ASC, year ASC, month ASC;
```

Query E

e. For each rental's state (pick up) show the top renting category

```
create view State as
select o.state, t.typeLabel, count(reservationID) as rentals
from reservation as r, CRC_office as o, car as c, car_type as t
where r.pickupLocationID=o.locationID and r.carID=c.carID and c.typeID=t.typeID
group by o.state, c.typeID
order by o.state, rentals DESC;

select state, typeLabel , max(rentals)
from State
group by state;
```

Query F

f. Show how many rentals there were in May 2015 in 'NY', 'NJ' and 'CA' (in three columns)

```
create view transpose as
select o.state, count(r.reservationID) as rentals
from reservation as r, CRC_office as o
where r.pickupLocationID=o.locationID and
      (o.state="NY" or o.state="NJ" or o.state="CA") and
      extract(year from r.pickupDate)=2015 and
      extract(month from r.pickupDate)=5
group by o.state;

select
  sum(if(state = 'NY', rentals, 0)) AS 'NY',
  sum(if(state = 'NJ', rentals, 0)) AS 'NJ',
  sum(if(state = 'CA', rentals, 0)) AS 'CA'
       from transpose;
```

Query G

g. For each month of 2015, count how many rentals had amount greater than this month's average rental amount

Amount greater than **this month's** average rental amount

```
select year(a.pickupdate) as yr, month(a.pickupDate) as mnth,
      count(a.reservationID) as counter
from reservation as a
where year(a.pickupDate)=2015 and a.amount >
      (select avg(b.amount)
      from reservation as b
      where month(a.pickupDate)=month(b.pickupDate) and
      year(a.pickupDate)=year(b.pickupDate)
      group by month(b.pickupDate))
group by month(a.pickupDate);
```

Amount greater than **every month's** average rental amount

```
select year(pickupdate) as yr, month(pickupDate) as mnth,
      count(reservationID) as counter
from reservation
where amount>all
      (select avg(amount)
      from reservation
      group by month(pickupDate))
      and reservationID in
            (select reservationID
            from reservation
            where year(pickupDate)=2015 )
group by month(pickupDate);
```

Query H

h. For each month of 2015, show the percentage change of the total amount of rentals over the total amount of rentals of the same month of 2014

```
select year(t.pickupdate) as yr, month(t.pickupdate) as mnth,
      round((t.amount- l.amount) * 100 /  t.amount, 2) as percent
from      reservation as l
inner join reservation as t
on month(t.pickupdate) = month(l.pickupdate) and
      year(l.pickupdate) = year(t.pickupdate) - 1
where year(t.pickupdate)=2015;
```

Query I

i. For each month of 2015, show in three columns: the total rentals' amount of the previous months, the total rentals' amount of this month and the total rentals' amount of the following months

```
SELECT cumTi-ti as prevMonths, ti as thisMonth, cumulativeTotal-cumTi as nextMonths
FROM
      (SELECT
      year(b.pickupdate) AS yr, month(b.pickupdate) AS mnth,
      sum(b.amount) AS ti,
            (SELECT sum(amount)
            from reservation
            where year(pickupdate)=2015) as cumulativeTotal,
                        (SELECT
                        sum(a.amount) as cumTi
                        FROM reservation as a
                        WHERE month(a.pickupdate) <= mnth and
                        year(a.pickupdate)=yr
                        GROUP BY mnth ASC
                        )as cumTi
                  FROM reservation AS b
            WHERE year(b.pickupdate)=2015
            GROUP BY month(b.pickupdate) ASC
      ) as tiandTi;
```

Connect java to MySQL

Using the programming language of your choice, connect to the database and implement query (i) above – without using GROUP BY SQL statements

```
import java.sql.*;
//assignment_1
//Eva Giannatou

public class Assignment_1
{
  public static void main(String[] args)
  {
    try
    {
      //create mysql database connection
      String myDriver = "org.gjt.mm.mysql.Driver";
      String myUrl = "jdbc:mysql://localhost/sql_assignment1";
      Class.forName(myDriver);
      Connection conn = DriverManager.getConnection(myUrl, "root", "1684");

//the SQL query
//inner subquery-> I calculated the cumulative function(cumTi) of the total amount
//for each month(ti)

//mid subquery-> I created a table combining the current month, the total amount for
//this month(ti) and the cumulative amount (cumTi) for this month

//outer query-> I use the cumulative function (cumTi) and  the total amount for each
//month(ti) in order
//to calculate the total amount of the previous, the current and the next months
//ex cumTi-ti as prevMonths, ti as thisMonth, cumulative(for month=12)-cumTi as
//nextMonths

String query =
"select cumTi-ti as prevMonths, ti as thisMonth, cumulativeTotal-cumTi as nextMonths  from "+
"(SELECT year(b.pickupdate) AS yr, month(b.pickupdate) AS mnth, sum(b.amount) AS ti, "+
" (select sum(amount) from reservation where year(pickupdate)=2015) as cumulativeTotal, "+
"(SELECT sum(a.amount) as cumTi "+
"FROM reservation as a "+
"WHERE month(a.pickupdate) <= mnth and year(a.pickupdate)=yr "+
")as cumTi "+
"FROM reservation AS b "+
"where year(b.pickupdate)=2015 and month(b.pickupdate)=?"+
") as tiandTi ";

//create the java prepared statement
PreparedStatement st = null;

//create the resultselt
ResultSet rs = null;

// execute the query, and get a java resultset
st = conn.prepareStatement(query);
```

```
//print table title
System.out.format("%15s%15s%15s\n",
   "Previous Months", "This Month", "Next Months");
   System.out.format("%15s%15s%15s\n",
   "===============", "==========", "===========");


//loop mnth from 1 to 12
for (int mnth=1;mnth<13;mnth++){

// replacing the first ? with mnth
st.setInt(1, mnth);

      // execute query
      rs = st.executeQuery();

      while (rs.next())
      {

             int prev = rs.getInt("prevMonths");
             int curr = rs.getInt("thisMonth");
             int next = rs.getInt("nextMonths");


        // print the results
        System.out.format("%15d%15d%15d\n", prev, curr, next);

      } //end of while
  } //end of for

      st.close();
    }
    catch (Exception e)
    {
      System.err.println("Got an exception! ");
      System.err.println(e.getMessage());
    }
  }
}
```