MSc in Business Analytics - Part time

1st Assignment

Data Management & Business Intelligence

Professor: Mr. Chatziantoniou Damianos

Students: Arseniou Evangelia (p2822026)

Ligkou Sotiria (p2822023)

CONTENTS

Description of the Case

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 also keeps data about 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 cars. A car rental has a unique reservation number, an amount (the value of the rental), the pickup and the return date. The car is picked up from a location and returned to another location (not necessarily the same).

Entity-Relationship Diagram

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.

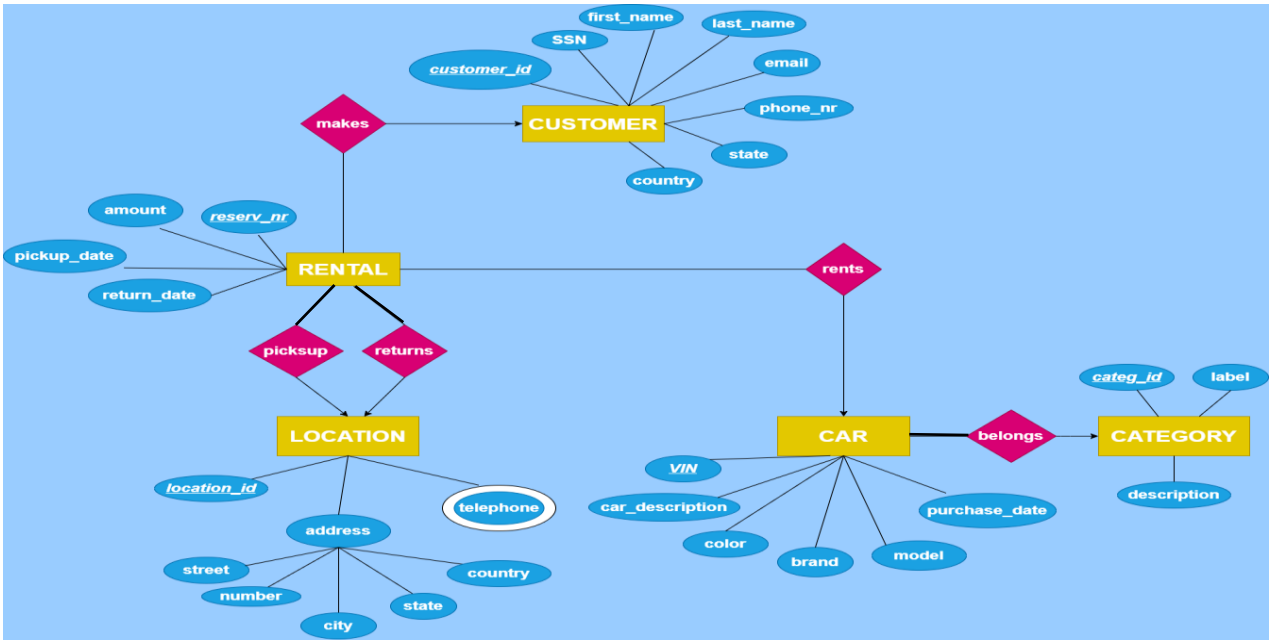

Diagram 1. Entity – Relationship Diagram for Car Rental Company

In order to draw the ERD it was used Draw.io tools in https://app.diagrams.net/ .

Based on "Diagram 1":

- Entity Sets are represented by yellow color (Customer, Rental, Location, Car, Category)
- Relationship sets are represented by pink color (makes, pickup, returns, rents, belongs)
- Attributes are represented by blue color (city, amount, brand, color etc.)
  - Multivalued attributes are represented by double ellipses with white and blue color (telephone)
- Primary keys are underlined (reserv_nr, location_id, VIN, categ_id, customer_id).

Relational Schema

Create the relational schema in MySQL/SQLServer and insert a few records into the tables to test your queries below. You will have to hand in the CREATE TABLE statements.

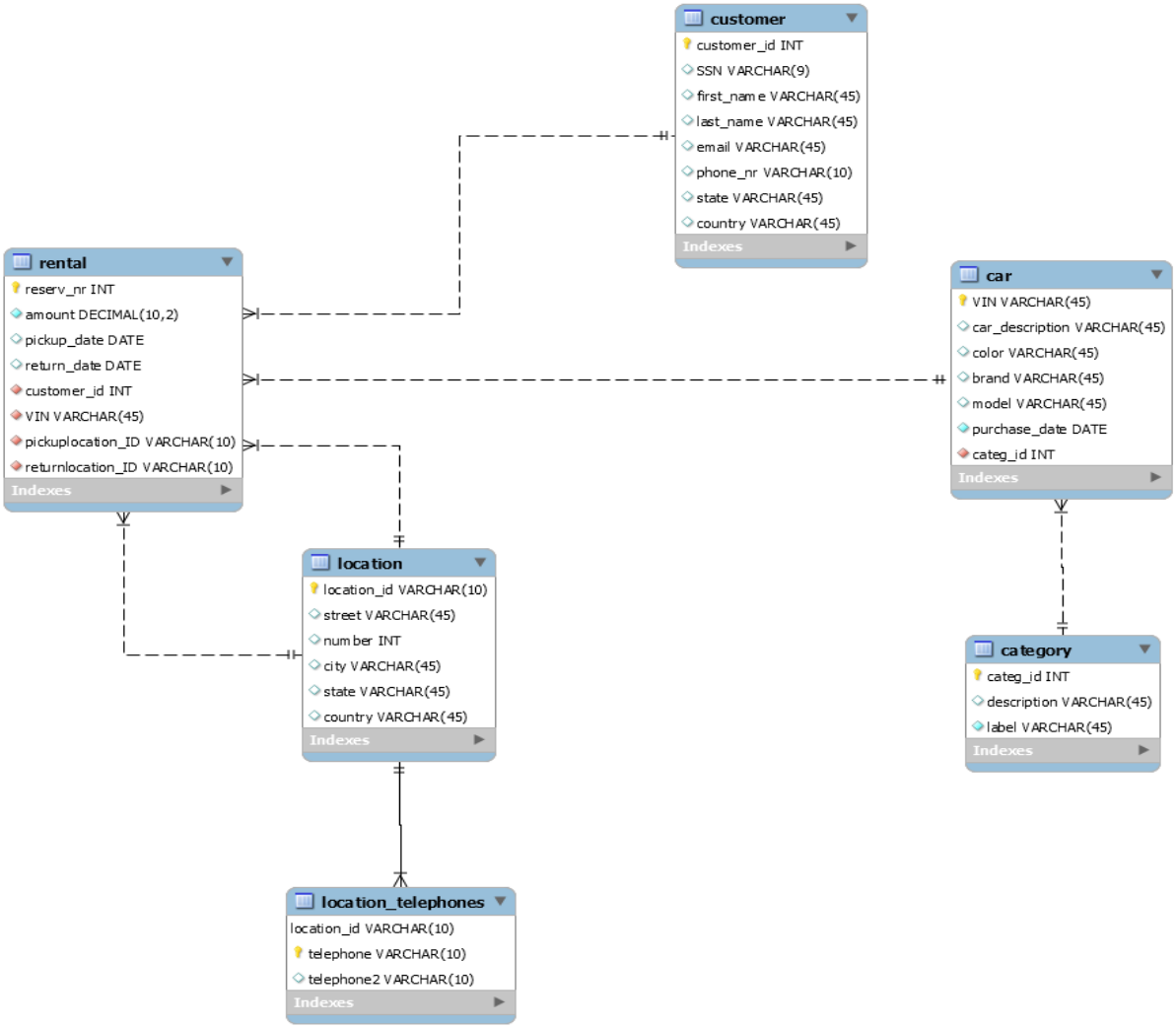

Diagram 2. Relational Schema for Car Rental Company

As we could see from "Diagram 2" there are some foreign keys which referred to:

Rental Table

- rental.customer_id -> customer.customer_id
- rental.VIN -> car.VIN
- rental.pickuplocation_id -> location.location_id
- rental.returnlocation_id -> location.location_id

Car Table

- car.categ_id -> category.categ_id

Location Telephones Table

- location_telephones.location_id -> location.location_id

Create Table Statements

```
CREATE TABLE category (
  categ_id int NOT NULL,
  description VARCHAR(300),
  label VARCHAR(45) NOT NULL,
  PRIMARY KEY (`categ_id`);
```

Category Table

Customer Table

```
CREATE TABLE customer (
  customer_id int NOT NULL,
  SSN VARCHAR(9),
  first_name VARCHAR(45),
  last_name VARCHAR(45),
  email VARCHAR(45),
  phone_nr VARCHAR(10),
  state VARCHAR(45),
  country VARCHAR(45),
  PRIMARY KEY (`customer_id`);
```

```
CREATE TABLE location (
  location_id VARCHAR(10) NOT NULL,
  street VARCHAR(45),
  number INT,
  city VARCHAR(45),
  state VARCHAR(45),
  country VARCHAR(45),
  PRIMARY KEY (location_id);
```

Location Table

Location_telephones Table

```
CREATE TABLE location_telephones (
location_id VARCHAR(10) NOT NULL,
telephone VARCHAR(10) NOT NULL,
telephone2 VARCHAR(10),
CONSTRAINT LT_PK PRIMARY KEY(location_id, telephone),
CONSTRAINT LT_FK FOREIGN KEY(location_id) REFERENCES location(location_id);
```

```
CREATE TABLE car (
  VIN VARCHAR(45) NOT NULL,
  car_description VARCHAR(300),
  color VARCHAR(45),
  brand VARCHAR(45),
  model VARCHAR(45),
  purchase_date DATE NOT NULL,
  categ_id INT NOT NULL,
  PRIMARY KEY (VIN),
  FOREIGN KEY (categ_id) REFERENCES category(categ_id));
```


```
CREATE TABLE rental (
  reserv_nr INT NOT NULL,
  amount DECIMAL(10,2) NOT NULL,
  pickup_date DATE,
  return_date DATE,
  customer_id INT NOT NULL,
  VIN VARCHAR(45) NOT NULL,
  pickuplocation_ID VARCHAR(10) NOT NULL,
  returnlocation_ID VARCHAR(10) NOT NULL,
  PRIMARY KEY (reserv_nr),
  FOREIGN KEY (pickuplocation_ID) REFERENCES location (location_id),
  FOREIGN KEY (returnlocation_ID) REFERENCES location (location_id),
  FOREIGN KEY (customer_id) REFERENCES customer (customer_id),
  FOREIGN KEY (VIN) REFERENCES car(VIN);
```

SQL QUERIES

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

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

```
SELECT reserv_nr, pickuplocation_id
FROM rental
WHERE pickup_date = '2015-05-20';
```

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

```
SELECT first_name "First Name", last_name "Last Name",
        phone_nr "phone numbers of customers in category luxury"
FROM customer as c, rental as r, car, category
WHERE c.customer_id=r.customer_id and r.VIN=car.VIN and
        car.categ_id=category.categ_id and category.label='luxury';
```

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

```
SELECT pickuplocation_id "location ID", round(sum(amount),2) "Total amount of rentals"
FROM rental
GROUP BY pickuplocation_id;
```

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

```
SELECT car.categ_id "Category ID",  extract(year from r.pickup_date) "Year",
        extract(month from r.pickup_date) "Month", round(sum(amount),2) "Total amount of rentals"
FROM  rental as r, car
WHERE r.VIN=car.VIN
GROUP BY car.categ_id, year, month
ORDER BY car.categ_id ASC, year ASC, month ASC;
```

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

```
DROP VIEW IF EXISTS V1;

CREATE VIEW V1 AS
SELECT state, label, count(reserv_nr) AS rentals
FROM location, category, rental, car
WHERE rental.pickuplocation_id=location.location_id AND rental.VIN=car.VIN
        AND category.categ_id=car.categ_id
GROUP BY location.state, category.categ_id
ORDER BY location.state, rentals DESC;

SELECT state, label , max(rentals) "Total Rentals for top renting category"
FROM V1
GROUP BY state;
```

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

```
DROP VIEW IF EXISTS may2015;

CREATE VIEW may2015 AS
SELECT state, count(reserv_nr) AS rentals
FROM location, rental
WHERE rental.pickuplocation_id=location.location_id AND (location.state="NY" OR
        location.state="NJ" OR location.state="CA") AND extract(YEAR FROM
        rental.pickup_date)='2015' AND extract(MONTH FROM rental.pickup_date)='5'
GROUP BY location.state;

SELECT
  max(case when state = 'CA' then rentals end) AS CA,
  max(case when state = 'NY' then rentals end) AS NY,
  max(case when state = 'NJ' then rentals end) AS NJ
FROM may2015;
```

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

```
SELECT count(reserv_nr) AS rentals, month(pickup_date) AS month, year(pickup_date) AS year
FROM rental AS r1
WHERE year(r1.pickup_date)=2015 AND r1.amount > (SELECT avg(r2.amount)
                                                 FROM rental AS r2
                                                 WHERE
                                                     month(r1.pickup_date)=month(r2.pickup_date)
                                                     and year(r1.pickup_date)=year(r2.pickup_date)
                                                 GROUP BY month(r2.pickup_date))
GROUP BY month(r1.pickup_date);
```

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(pickup_date), month(pickup_date) AS month,
        ROUND((sum(amount)- r2.sum2014) * 100 / sum(amount), 2) AS percentage
FROM rental AS r1
INNER JOIN (SELECT month(pickup_date) month, round(sum(AMOUNT),2) AS sum2014
            FROM rental
            WHERE year(pickup_date)="2014"
            GROUP BY MONTH(pickup_date)
            ORDER BY month(pickup_date) ASC) as r2
ON month(r1.pickup_date)= r2.month
WHERE year(r1.pickup_date)="2015"
GROUP BY MONTH(r1.pickup_date)
ORDER BY month(r1.pickup_date) ASC;
```

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

```
DROP VIEW IF EXISTS V2;

CREATE VIEW V2 AS
SELECT MONTH(pickup_date) AS current_month, SUM(amount) AS current_amount
FROM rental
WHERE YEAR(pickup_date) = '2015'
GROUP BY MONTH(pickup_date)
ORDER BY MONTH(pickup_date);

SELECT i.current_month AS month_number,
    (SELECT SUM(current_amount) AS sum_amount
     FROM V2
     WHERE current_month < i.current_month) AS Previous_Month, i.current_amount AS Current_Month,
            (SELECT SUM(current_amount) AS sum_amount
             FROM V2
             WHERE current_month > i.current_month) AS Next_Month
FROM V2 i
ORDER BY month_number asc;
```

Connect R to SQL Server

Using the programming language of your choice, connect to the database and implement query (i) above – ***without using GROUP BY SQL statements***, i.e. you are only allowed to use SELECT…FROM…WHERE.

- We are going to use R language for this question. At first, we have to install the package RMySQL.

```
install.packages('RMySQL')
library(RMySQL)
```

- In order to connect R to SQL we use the command dbConnect.

```
mydb<-dbConnect(MySQL(),user='root',password='*******',dbname='company',host='127.0.0.1')
```

- Every time we want to create a query in R and send it back to SQL we use dbSendQuery and the commands in quotation marks (““). By using fetch, it is possible to see the data frame that have been created from the query and with “n=-1” we have the opportunity to retrieve all pending records.

```
c<-dbSendQuery(mydb,"DROP VIEW IF EXISTS V3")
d<-fetch(c,n=-1)

c<-dbSendQuery(mydb,"CREATE VIEW V3 AS
                        SELECT distinct MONTH(pickup_date) AS current_month,
                                SUM(amount) OVER (PARTITION BY month(pickup_date)) AS
                                current_amount
                        FROM rental
                        WHERE YEAR(pickup_date) = '2015'
                        ORDER BY MONTH(pickup_date)")

c<-dbSendQuery(mydb,"SELECT i.current_month AS month_number,
                            (SELECT SUM(current_amount) AS sum_amount
                             FROM V3
                             WHERE current_month < i.current_month) AS Previous_Month,
                                    i.current_amount AS Current_Month,
                                    (SELECT SUM(current_amount) AS sum_amount
                                      FROM V3
                                      WHERE current_month > i.current_month) AS Next_Month
FROM V3 i
ORDER BY month_number asc")

d<-fetch(c,n=-1)
d
```