Data Management & Business Intelligence

ASSIGNMENT 1

ACADEMIC YEAR 2010-2021 (PART TIME)

KARPATHAKI ANDRIANI – P2822020
VRETTEAS STYLIANOS – P2822003

Περιεχόμενα

SECTION A.: Assignment

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.)
Deliverables (in one word document):
1. (10%) 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.
2. (10%) 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.
3. (60%) 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

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'

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

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

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

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

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

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

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

4. (20%) 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.

Deliverable 1: Entity-Relationship Diagram (ERD)

The Entity-Relational Diagram (ERD) contains five (5) entities, their attributes and four (4) relationships between them, where:

- *Entities* are signed with blue color
  Customers, rentals, cars, categories, location
- *Attributes* with green
- *Relationships* with pink
- *Primary keys* with purple

(see Figure 1.)

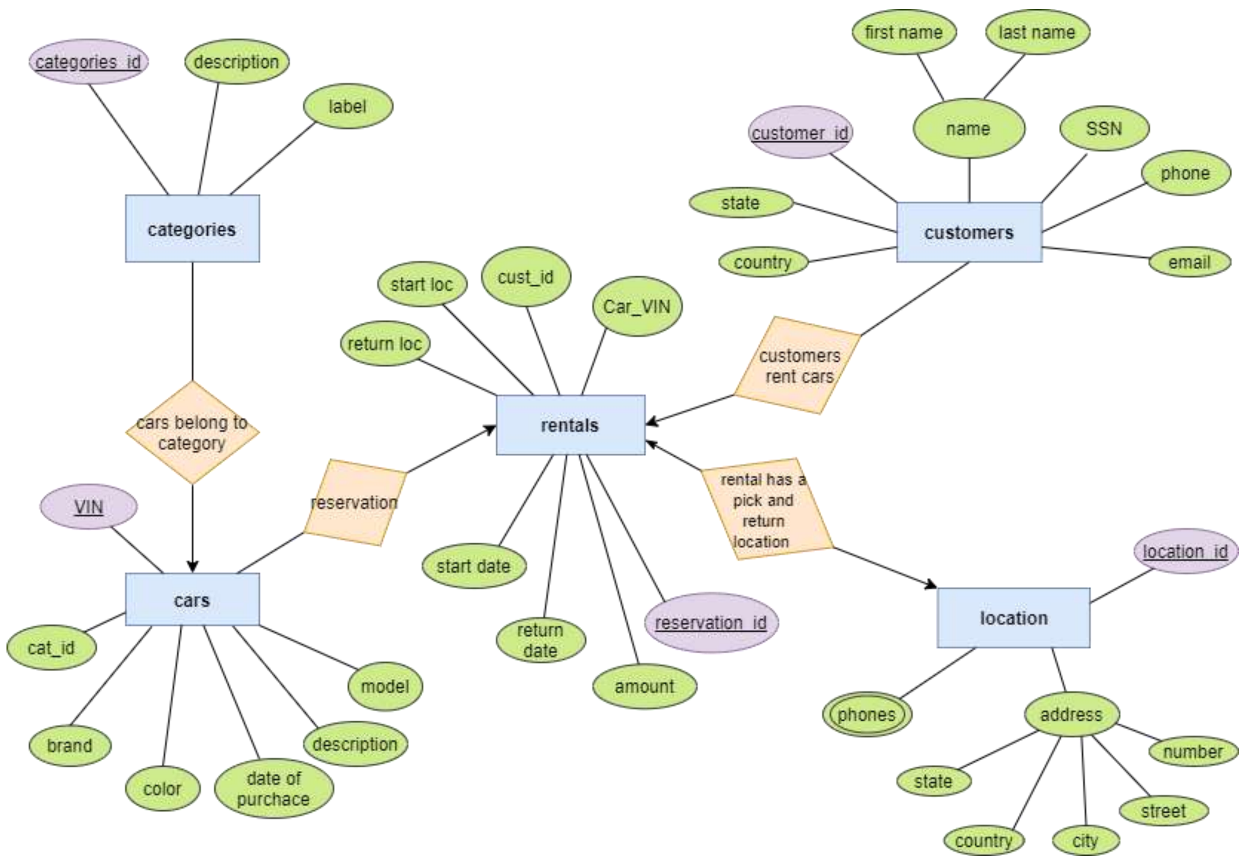


Figure 1.

Entity Relational Diagram

Car rental company

Deliverable 2: Creation of Schema and tables

A. SCHEMA

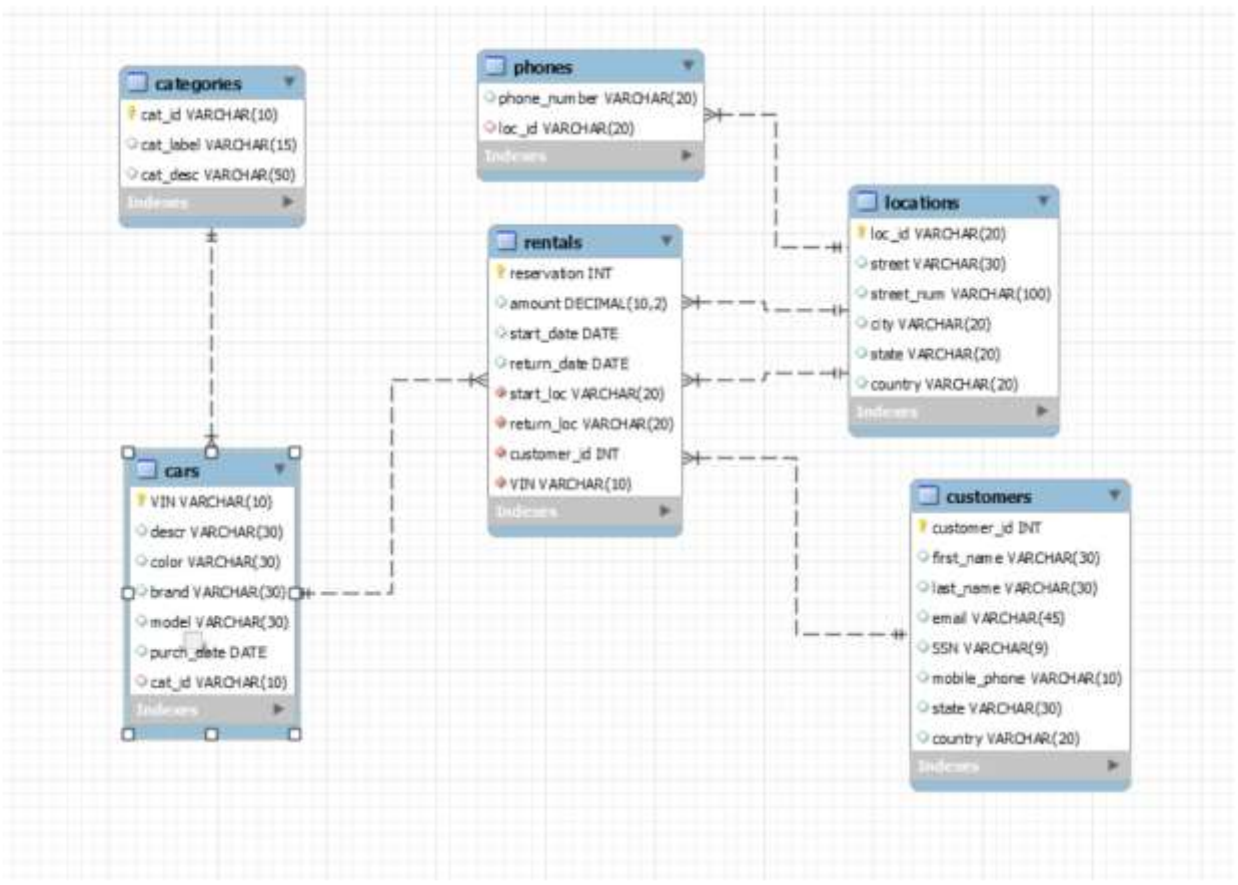


Figure 2
Relational Schema
Car rental company

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database using the below terms.

- Entities are represented with rectangles. An entity is an object where you want to store information. (rentals, customers, locations, cars, categories)
- Attributes of entities are represented with ovals. A key attribute is the unique element – characteristic of each entity. Examples (SSN, TIN, etc.) Furthermore, multivalued attributes are represented by double ovals, these are attributes that can have more than one value, e.g. telephones, *"Each location has one or more telephone numbers"* (we can store these attributes in a different table in our database).
- Relationships between the entities are shown with actions which are represented by diamond shapes. Here we have to check about cardinalities and ordinalities. These are terms that are used to show the maximum - minimum number of relationships between entities and are easily interpreted with "one to many", "many to one", "zero to many" etc. quotes.

  <u>Specifically, in CRC database.</u>
- Cars – Categories is a "many to one" relationship. *"A car may belong to one (exactly one) vehicle category".*
- Customers – Rentals is "one to many" but not vice versa relationship, a customer can have many entries of rentals, but a rental is unique to only one customer. "*Customers rent cars*"
- Rentals – Locations is "many to many" relationship, due to the fact that a rental consists of a pickup location (start_loc) and a return location. "*The car is picked up from a location and returned to another location (not exactly the same)"*

ER diagram is transformed into the relational schema.

The relational schema is an outline of how data is organized in the database. It also specifies better which columns communicate with Primary Key or Foreign key relationships.

B. TABLE STATEMENTS

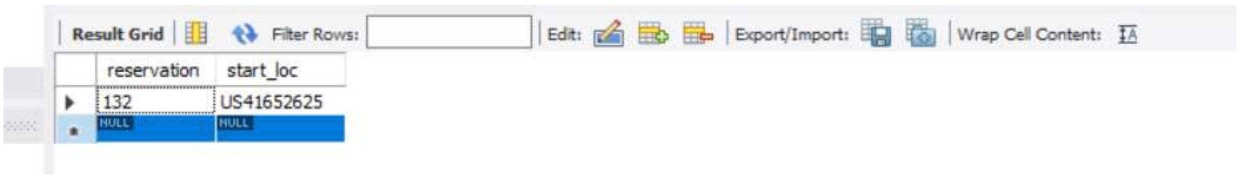
```
CREATE DATABASE IF NOT EXISTS CRC;
USE CRC;

CREATE TABLE IF NOT EXISTS customers
(
customer_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
first_name VARCHAR(30),
last_name VARCHAR(30),
email VARCHAR(45),
SSN VARCHAR(9),
mobile_phone VARCHAR(10),
state VARCHAR(30),
country VARCHAR(20)
);

CREATE TABLE IF NOT EXISTS locations
(
loc_id VARCHAR(20) NOT NULL PRIMARY KEY,
street VARCHAR(30),
street_num VARCHAR(100),
city VARCHAR(20),
state VARCHAR(20),
country VARCHAR(20)
);

CREATE TABLE IF NOT EXISTS categories
(
cat_id VARCHAR(10) NOT NULL PRIMARY KEY,
cat_label VARCHAR(15),
cat_desc VARCHAR(50)
);

CREATE TABLE IF NOT EXISTS phones
(
phone_number VARCHAR(20),
loc_id VARCHAR(20),
FOREIGN KEY(loc_id) REFERENCES
locations(loc_id)
);

CREATE TABLE IF NOT EXISTS cars
(
VIN VARCHAR(10) NOT NULL PRIMARY KEY,
descr VARCHAR(30),
color VARCHAR(30),
brand VARCHAR(30),
model VARCHAR(30),
purch_date DATE,
cat_id VARCHAR(10),
FOREIGN KEY (cat_id) REFERENCES
categories(cat_id)
);

CREATE TABLE IF NOT EXISTS rentals
(
reservation INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
amount NUMERIC(10,2),
start_date DATE,
return_date DATE,
start_loc VARCHAR(20) NOT NULL,
return_loc VARCHAR(20) NOT NULL,
customer_id INT NOT NULL,
VIN VARCHAR(10) NOT NULL,
FOREIGN KEY(start_loc) REFERENCES
locations(loc_id),
FOREIGN KEY(return_loc) REFERENCES
locations(loc_id),
FOREIGN KEY(customer_id) REFERENCES
customers(customer_id),
FOREIGN KEY(VIN) REFERENCES
cars(VIN)
);
```

Deliverable 3: Queries

Note: Data entries were inserted through the Data Import wizard tool of MySQL workbench. Data generated from Microsoft excel with the use of random and lookup functions.

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

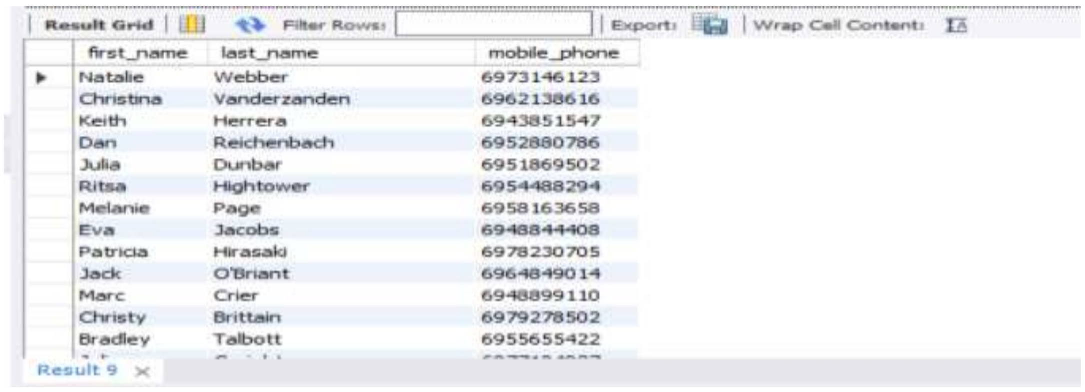
```
SELECT rentals.reservation, rentals.start_loc
FROM rentals
WHERE start_date = '2015-5-20';
```



reservation	start_loc
132	US41652625
NULL	NULL

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'

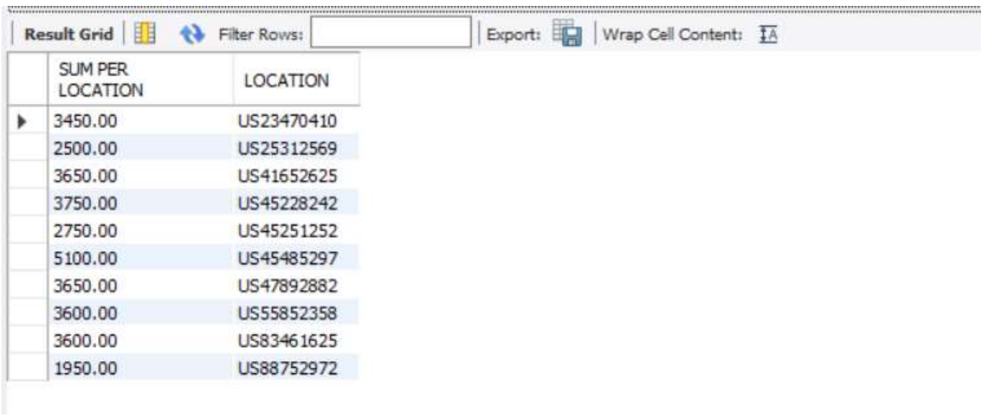
```
SELECT first_name, last_name, mobile_phone
FROM customers as cust,
     rentals as r,
     cars,
     categories as cat
WHERE cust.customer_id = r.customer_id
    AND r.VIN = cars.VIN
    AND cars.cat_id = cat.cat_id
    AND cat.cat_label = 'luxury';
```



first_name	last_name	mobile_phone
Natalie	Webber	6973146123
Christina	Vanderzanden	6962138616
Keith	Herrera	6943851547
Dan	Reichenbach	6952880786
Julia	Dunbar	6951869502
Ritsa	Hightower	6954488294
Melanie	Page	6958163658
Eva	Jacobs	6948844408
Patricia	Hirasaki	6978230705
Jack	O'Briant	6964849014
Marc	Crier	6948899110
Christy	Brittain	6979278502
Bradley	Talbott	6955655422

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

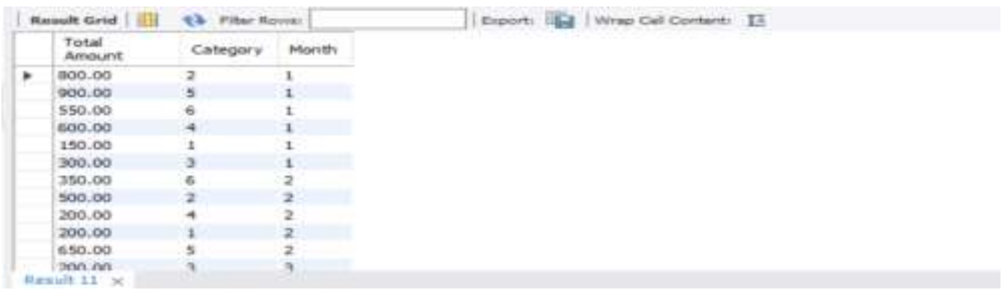
```
SELECT SUM(amount) AS 'SUM PER LOCATION',
       start_loc AS 'LOCATION'
FROM rentals
GROUP BY start_loc;
```



SUM PER LOCATION	LOCATION
3450.00	US23470410
2500.00	US25312569
3650.00	US41652625
3750.00	US45228242
2750.00	US45251252
5100.00	US45485297
3650.00	US47892882
3600.00	US55852358
3600.00	US83461625
1950.00	US88752972

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

```
SELECT SUM(r.amount) as 'Total Amount',
       cars.cat_id as 'Category',
       MONTH(r.start_date) as 'Month'
FROM rentals as r,
     cars
WHERE r.VIN = cars.VIN
GROUP BY cars.cat_id,
         MONTH(r.start_date)
ORDER BY MONTH(r.start_date);
```

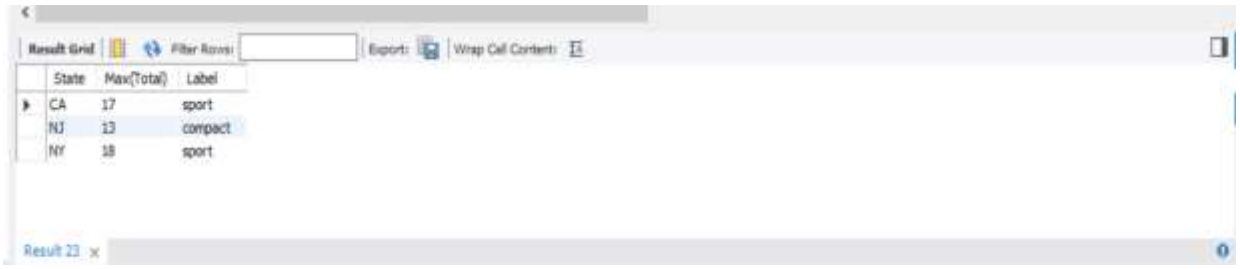


Total Amount	Category	Month
800.00	2	1
900.00	5	1
550.00	6	1
600.00	4	1
150.00	1	1
300.00	3	1
350.00	6	2
500.00	2	2
200.00	4	2
200.00	1	2
650.00	5	2
300.00	3	3

Result 11

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

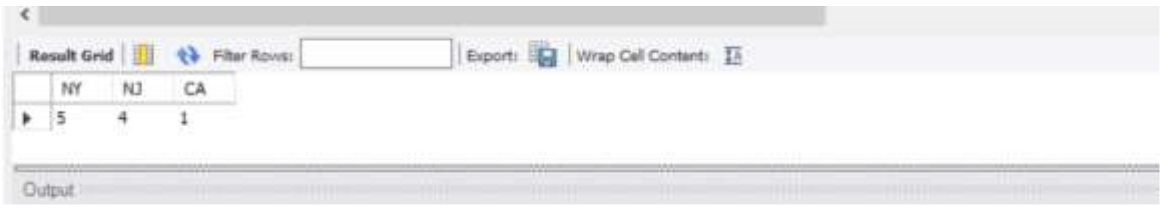
```
SELECT State, Max(Total), Label
FROM(
SELECT l.state as "State", count(cars.cat_id) as " Total",
    cat.cat_label as "Label"
FROM locations l,
    cars,
    rentals r,
    categories cat
WHERE r.start_loc = l.loc_id AND cars.VIN = r.VIN
    AND cat.cat_id = cars.cat_id
GROUP BY State, Label
ORDER BY State, Total DESC) as taske_table
GROUP BY State;
```



State	Max(Total)	Label
CA	17	sport
NJ	13	compact
NY	18	sport

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

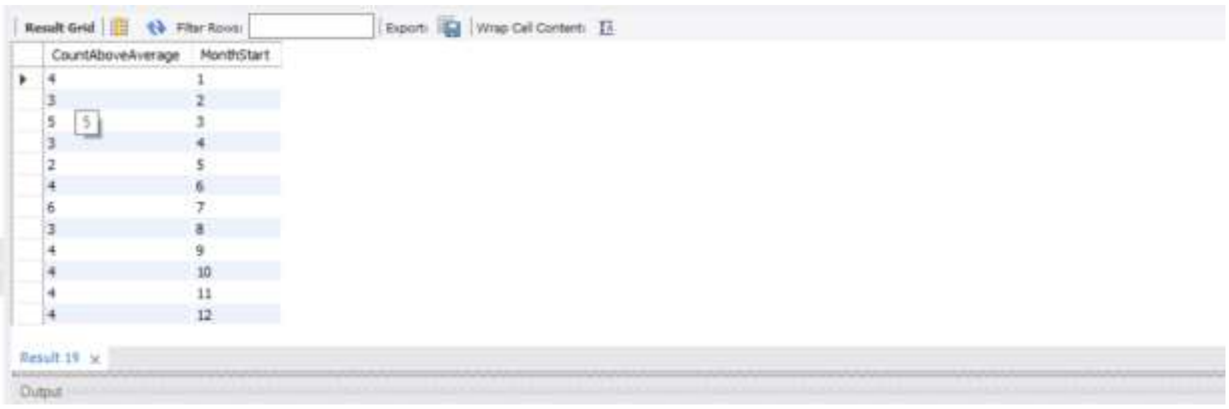
```
SELECT SUM(NY) AS NY, SUM(NJ) AS NJ, SUM(CA) AS CA
FROM (
SELECT
(CASE WHEN LOC.STATE = 'NY' THEN 1 ELSE 0 END )AS 'NY',
(CASE WHEN LOC.STATE = 'NJ' THEN 1 ELSE 0 END  )AS 'NJ',
(CASE WHEN LOC.STATE = 'CA' THEN 1 ELSE 0 END )AS 'CA'
FROM locations loc
INNER JOIN rentals r ON r.start_loc = loc.loc_id
WHERE MONTH(r.start_date) = 05 AND YEAR(r.start_date) = 2015
AND loc.state IN ('NY','NJ','CA')) taskf;
```



	NY	NJ	CA
▶	5	4	1

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

```
CREATE VIEW T1 AS
SELECT ROUND(AVG(amount),2) as aver, MonthStart
FROM    (SELECT reservation, MONTH(start_date) as MonthStart, amount
            FROM rentals
            WHERE YEAR(start_date) = '2015'
            GROUP BY MONTH(start_date), reservation
            ORDER BY MONTH(start_date)) as loc
GROUP BY MonthStart;


CREATE VIEW T2 AS
SELECT reservation, MONTH(start_date) as MonthStart, amount
            FROM rentals
            WHERE YEAR(start_date) = '2015'
            GROUP BY MONTH(start_date), reservation
            ORDER BY MONTH(start_date);

SELECT count(amount) as CountAboveAverage, T1.MonthStart
FROM T1,T2
WHERE T1.MonthStart = T2.MonthStart AND amount > aver
GROUP BY T1.MonthStart;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

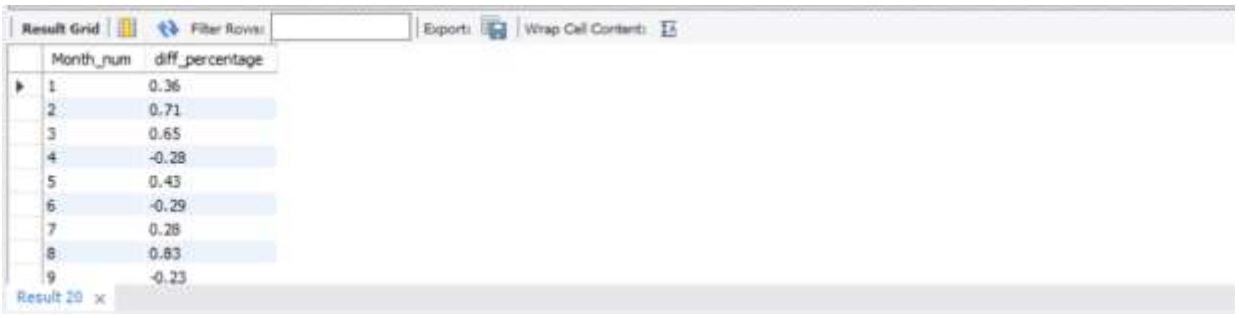
CountAboveAverage	MonthStart
4	1
3	2
5	3
3	4
2	5
4	6
6	7
3	8
4	9
4	10
4	11
4	12

Result 19 ×

Output

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 MONTH(start_date) AS Month_num, ROUND(((SUM(amount)-
t2.sum_2014)/t2.sum_2014),2) AS diff_percentage
FROM
rentals t1
LEFT JOIN
(SELECT MONTH(start_date) Month_num, ROUND(SUM(amount),2) as sum_2014
FROM rentals
WHERE YEAR(start_date) = "2014"
GROUP BY MONTH(start_date)
ORDER BY MONTH(start_date)) t2
ON MONTH(t1.start_date) = t2.Month_num
WHERE YEAR(t1.start_date) = '2015'
GROUP BY MONTH(t1.start_date)
ORDER BY MONTH(t1.start_date);
```



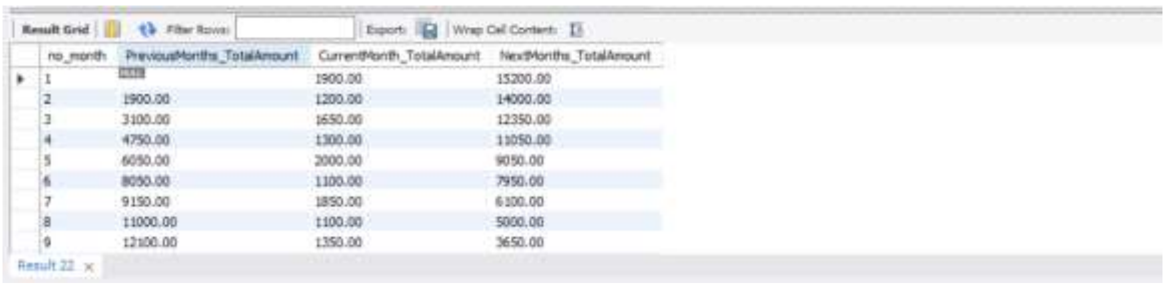
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Month_num	diff_percentage
1	0.36
2	0.71
3	0.65
4	-0.28
5	0.43
6	-0.29
7	0.28
8	0.83
9	-0.23

Result 20 ×

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

```
CREATE VIEW task_i AS
    SELECT
        MONTH(start_date) AS no_month, SUM(amount) AS test_amount
    FROM
        rentals
    WHERE
        YEAR(start_date) = '2015'
    GROUP BY MONTH(start_date)
    ORDER BY MONTH(start_date);

SELECT
    t1.no_month,
    (SELECT
            SUM(test_amount) AS sum_amount
        FROM
            task_i
        WHERE
            no_month < t1.no_month) AS PreviousMonths_TotalAmount,
    t1.test_amount AS CurrentMonth_TotalAmount,
    (SELECT
            SUM(test_amount) AS sum_amount
        FROM
            task_i
        WHERE
            no_month > t1.no_month) AS NextMonths_TotalAmount
FROM
    task_i t1;
```



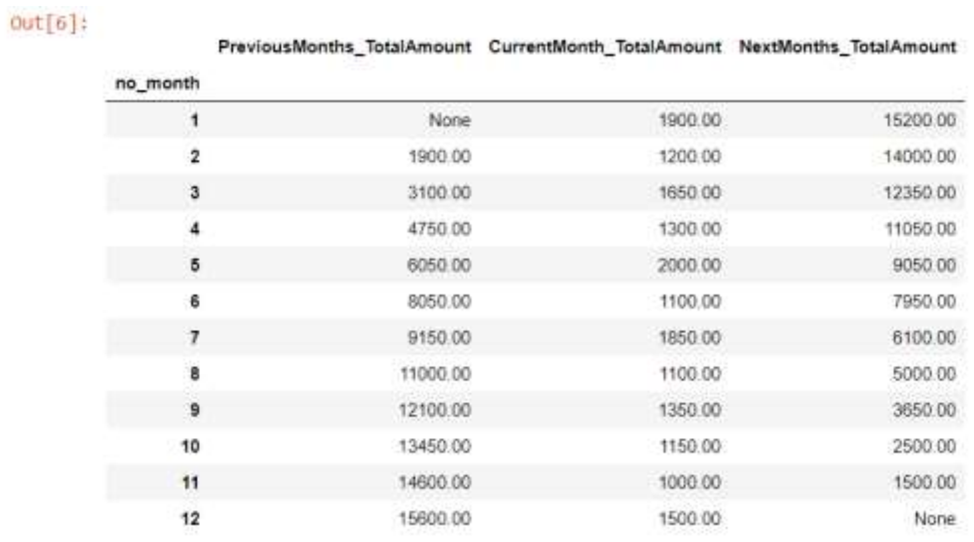
no_month	PreviousMonths_TotalAmount	CurrentMonth_TotalAmount	NextMonths_TotalAmount
1	NULL	1900.00	15200.00
2	1900.00	1200.00	14000.00
3	3100.00	1650.00	12350.00
4	4750.00	1300.00	11050.00
5	6050.00	2000.00	9050.00
6	8050.00	1100.00	7950.00
7	9150.00	1850.00	6100.00
8	11000.00	1100.00	5000.00
9	12100.00	1350.00	3650.00

Deliverable 4: Connect to Database through Python

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.

```
# Connection with the database
import mysql.connector
mydb = mysql.connector.connect(  host = "localhost",
                                 user = "root",
                                 passwd = "@businessanalytics",
                                 database = "crc")
mycursor = mydb.cursor()

# Query execution
mycursor.execute("""
SELECT
    t1.no_month,
    (SELECT
            SUM(test_amount) AS sum_amount
        FROM
            task_i
        WHERE
            no_month < t1.no_month) AS PreviousMonths_TotalAmount,
    t1.test_amount AS CurrentMonth_TotalAmount,
    (SELECT
            SUM(test_amount) AS sum_amount
        FROM
            task_i
        WHERE
            no_month > t1.no_month) AS NextMonths_TotalAmount
FROM
    task_i t1;""")
```

```python
# Data manipulation
result = []
for i in mycursor :
    result.append(i)

# Import decimal module in order to get rounded arithmetic
from decimal import Decimal
result_fixed = []
for row in result :
       result_fixed.append(list(map(str, list(row))))



# Returns a list of tuples describing the columns in a result set
columns = mycursor.description

# Transformation to list
list_columns=[]
for i in columns:
    y = list(i)
    list_columns.append(y)

names_col = []
for i in range((len(list_columns))):
    names_col.append(list_columns[i][0])

# Use of Pandas tool for dataframe creation
import pandas as pd

data = pd.DataFrame(result_fixed, columns = names_col)
data.set_index(names_col[0])
```



Out[6]:

no_month	PreviousMonths_TotalAmount	CurrentMonth_TotalAmount	NextMonths_TotalAmount
1	None	1900.00	15200.00
2	1900.00	1200.00	14000.00
3	3100.00	1650.00	12350.00
4	4750.00	1300.00	11050.00
5	6050.00	2000.00	9050.00
6	8050.00	1100.00	7950.00
7	9150.00	1850.00	6100.00
8	11000.00	1100.00	5000.00
9	12100.00	1350.00	3650.00
10	13450.00	1150.00	2500.00
11	14600.00	1000.00	1500.00
12	15600.00	1500.00	None

Figure 3.

Python connection to MySQL

Table created using pandas tool