2017

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

BUSINESS ANALYTICS
Master of Science

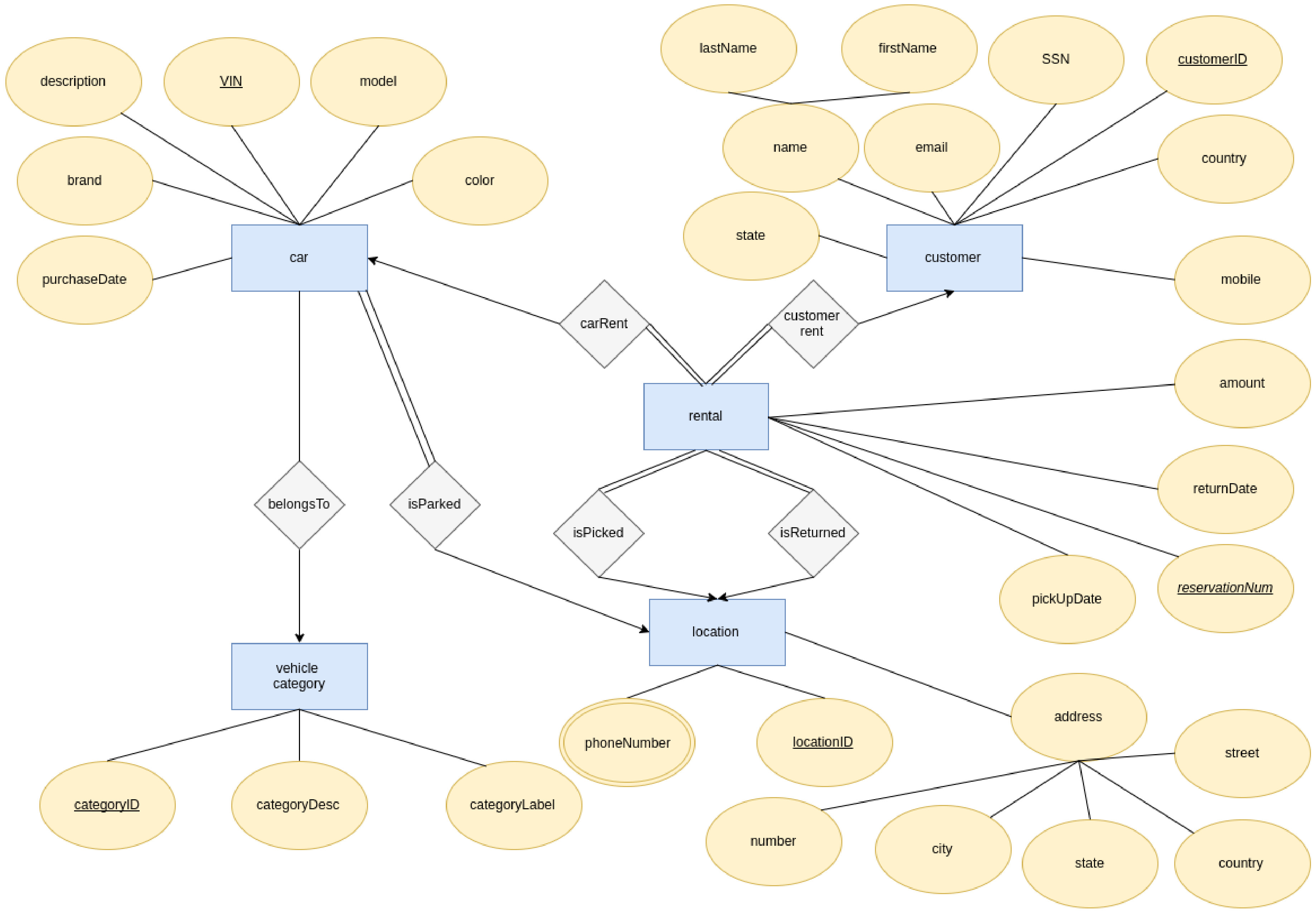
Data Management and Business Intelligence

ASSIGNMENT 1

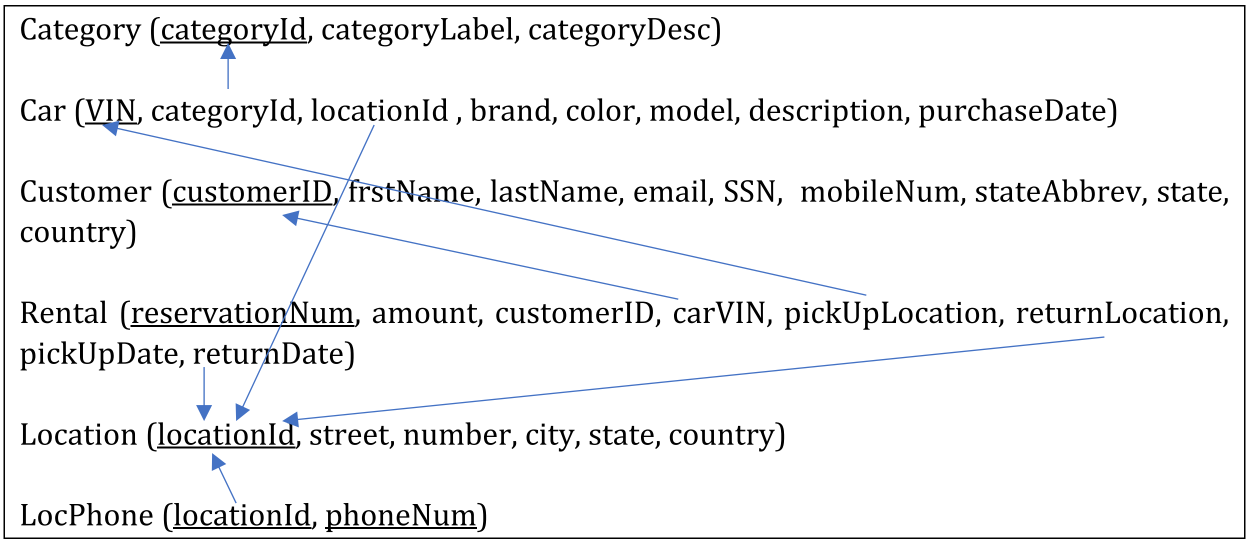
DIMITRIOS GERASIMOS ANASTASATOS
MEROPI MARIA ARGYROPOULOU
AGLAIA TSOPELAKOU

Contents

1. **Question 1)**
   Use the Entity-Relationship Diagram (ERD) to model entities, relationships, attributes, cardinalities, and constraints.
   a) Entity-Relationship Diagram



b) Relational Schema/Tables


Category (categoryId, categoryLabel, categoryDesc)

Car (VIN, categoryId, locationId , brand, color, model, description, purchaseDate)

Customer (customerID, frstName, lastName, email, SSN, mobileNum, stateAbbrev, state, country)

Rental (reservationNum, amount, customerID, carVIN, pickUpLocation, returnLocation, pickUpDate, returnDate)

Location (locationId, street, number, city, state, country)

LocPhone (locationId, phoneNum)

2. **Question 2)**
   Create the relational schema in MySQL/SQLServer and insert a few records into the tables to test your queries below.
   Below the relational schema created in MySQL after executing the CREATE TABLE statements.

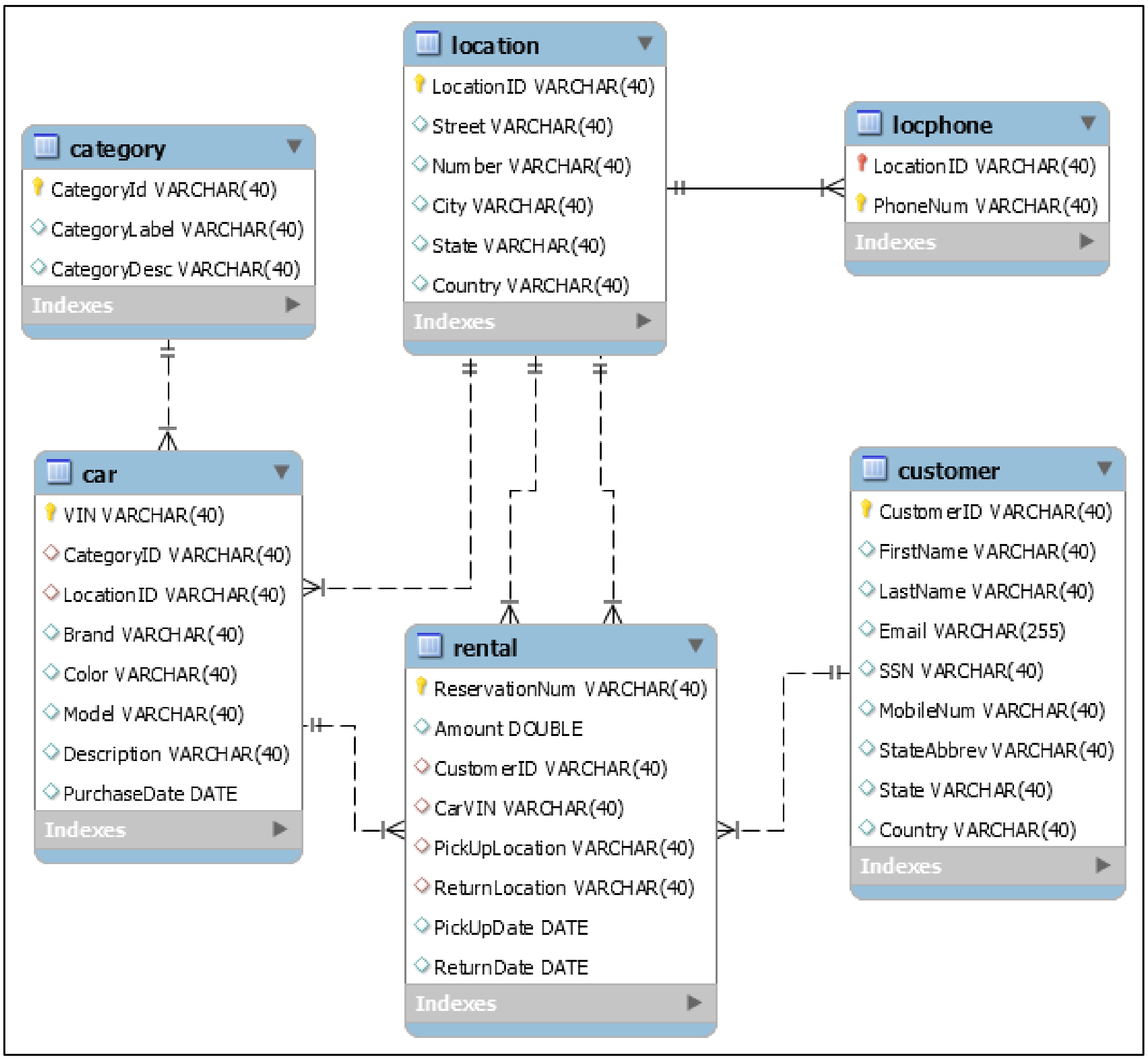


Figure 1: Relational Schema

<u>Create table statements:</u>
```
CREATE TABLE Category(
    CategoryId VARCHAR(40),
    CategoryLabel VARCHAR(40),
    CategoryDesc VARCHAR(40),
    PRIMARY KEY (CategoryID)
)

CREATE TABLE Location(
    LocationID VARCHAR(40),
    Street VARCHAR(40),
    Number VARCHAR(40),
    City VARCHAR(40),
    State VARCHAR(40),
    Country VARCHAR(40),
    PRIMARY KEY (LocationID)
)
```

```
CREATE TABLE Car (
    VIN VARCHAR(40),
    CategoryID VARCHAR(40),
    LocationID VARCHAR(40),
    Brand VARCHAR(40),
    Color VARCHAR(40),
    Model VARCHAR(40),
    Description VARCHAR(40),
    PurchaseDate DATE,
    PRIMARY KEY (VIN),
    FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID),
    FOREIGN KEY (LocationID) REFERENCES Location(LocationID)
)

CREATE TABLE Customer(
    CustomerID VARCHAR(40),
    FirstName VARCHAR(40),
    LastName VARCHAR(40),
    Email VARCHAR(255),
    SSN VARCHAR(40),
    MobileNum VARCHAR(40),
    StateAbbrev VARCHAR(40),
    State VARCHAR(40),
    Country VARCHAR(40),
    PRIMARY KEY (CustomerID)
)

CREATE TABLE LocPhone(
    LocationID VARCHAR(40),
    PhoneNum VARCHAR(40),
    CONSTRAINT LocPhoneID PRIMARY KEY (LocationID, PhoneNum),
    FOREIGN KEY (LocationID) REFERENCES Location (LocationID)
)

CREATE Table Rental (
    ReservationNum VARCHAR(40),
    Amount DOUBLE,
    CustomerID VARCHAR(40),
    CarVIN VARCHAR(40),
    PickUpLocation VARCHAR(40),
    ReturnLocation VARCHAR(40),
    PickUpDate DATE,
    ReturnDate DATE,
    PRIMARY KEY(ReservationNum),
    FOREIGN KEY(CarVIN) REFERENCES Car(VIN),
    FOREIGN KEY(CustomerID) REFERENCES Customer(CustomerID),
    FOREIGN KEY(PickUpLocation) REFERENCES Location(LocationID),
    FOREIGN KEY(ReturnLocation) REFERENCES Location(LocationID))
```

3. **Question 3)**
Write SQL code and test it to your data for the following queries:

- We used indicatively the following statement to insert the csv files with dummy data to test the SQL queries:

```
LOAD DATA LOCAL INFILE 'Assignment_1_Customers.csv' into table
Customer fields terminated by ',' enclosed by '' lines terminated by
'\n' (CustomerID, FirstName, LastName, Email, SSN, MobileNum,
StateAbbrev, State, Country)
```

which is equivalent to the following insert statement:

```
INSERT INTO
Customer(customerId,firstName,lastName,email,SSN,mobileNum,stateAbbre
v,state,country)
VALUES ("1","Pate","Berisford","pberisford0@vimeo.com","160-66-
7934","723-510-6026","","","Syria");
```

Below are all the excel files with the dummy data that we inserted and the respective statements in sql *(“insert values from excel.sql”)*.

location.csv locationphone.csv Rental.csv Customers_dummy.csv Car.csv

categories.csv

insert_values_from_excel.sql

- DB dump file:

Dump20171024.sql

- Queries answers:

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

```
SELECT ReservationNum,PickupLocation
FROM Rental
WHERE pickupDate='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 distinct c.firstName, c.lastName, c.mobileNum
FROM Customer AS c,Rental AS r,Category AS t,Car
WHERE c.CustomerID = r.CustomerID AND r.carVIN = Car.VIN
        AND Car.CategoryID = t.CategoryID AND t.CategoryLabel =
'luxury'
order by firstname desc;
```

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

```
SELECT PickUpLocation, sum(Rental.Amount)
FROM Rental
GROUP BY PickUpLocation
```

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

```
SELECT Car.CategoryID,
EXTRACT(Month from r.pickupDate) AS Month_,EXTRACT(Year from
r.PickUpDate) AS Year_
,sum(r.Amount) as TOTAL_AMOUNT_OF_RENTALS
FROM Rental as r, Car
WHERE r.carVIN=Car.VIN
GROUP BY Car.CategoryID,EXTRACT(Month from r.PickUpDate),EXTRACT(Year
from r.PickUpDate)
order by EXTRACT(Year from r.PickUpDate) asc, EXTRACT( Month from
r.PickUpDate) asc
```

NOTE/CLARIFICATION: We grouped also by Year as our data contain information from different years.

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

```
CREATE view State AS
SELECT l.state,t.CategoryLabel,count(reservationNum) AS Rentals
FROM Rental as r, Location as l, car as c, category as t
WHERE r.pickupLocation =l.locationId AND r.carVIN=c.VIN AND
c.categoryId=t.categoryId
GROUP BY l.state, t.CategoryLabel

SELECT state, CategoryLabel
FROM State AS e1
WHERE e1.Rentals = (SELECT MAX(e2.Rentals)
                    FROM State AS e2
                    WHERE e2.state = e1.state)
```

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

```
CREATE view transpose as
SELECT l.state,count(r.reservationNum) as Rentals
FROM rental as r, Location as l
WHERE r.pickupLocation=l.locationId AND
(l.state="NY" OR l.state="NJ" OR l.state="CA")
AND EXTRACT(year from r.pickupDate)="2015"
AND EXTRACT(month from r.pickupDate)="5"
GROUP BY l.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;
```

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

```
SELECT YEAR(r.pickupdate) AS yr, MONTH(r.pickupdate) AS month,
COUNT(r.reservationNum) AS counter
FROM rental as r
WHERE year(r.pickupdate)= '2015' AND
      r.amount>(SELECT avg(a.amount)
                FROM rental as a
                WHERE month(r.pickupdate)=month(a.pickupdate)
                and year(r.pickupdate)=year(a.pickupdate))
 GROUP BY month(r.pickupdate)
```

NOTE/CLARIFICATION: If there are no rentals with amount greater than the average, SQL will not show the result. This is due to the use of COUNT() and AVG() over GROUP BY Statement.

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.

```
CREATE view view1_2014 as
SELECT sum(r.amount) as amount_2014, EXTRACT(month from r.pickupDate)
as month_2014
FROM rental as r
WHERE EXTRACT(year from r.pickupDate)='2014'
GROUP BY EXTRACT(month from r.pickupDate)

CREATE view view1_2015 as
SELECT sum(r.amount) as amount_2015, EXTRACT(month from r.pickupDate)
as month_2015
FROM rental as r
WHERE EXTRACT(year from r.pickupDate)='2015'
GROUP BY EXTRACT(month from r.pickupDate)

select month_2014, month_2015, round(((view1_2015.amount_2015-
view1_2014.amount_2014)/view1_2014.amount_2014)*100,2) as percentage
from view1_2014, view1_2015
where month_2014=month_2015
```

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 mnth, p-t as prevMonths, t as thisMonth, n-p as nextMonth
FROM
      (SELECT year(r.pickupdate) AS yr, month(r.pickupdate) AS
              mnth, sum(r.amount) as t,  (SELECT sum(amount)
      FROM rental
                WHERE year(pickupdate)='2015') as n,
               (SELECT sum(a.amount) as p
                FROM rental as a
                WHERE month(a.pickupdate)<=mnth
   AND year(a.pickupdate)=yr
   GROUP BY mnth ASC)as p
       FROM rental as r
       WHERE year(r.pickupdate)='2015'
   GROUP BY month (r.pickupdate) ASC)as t
```

4. **Question 4)**
   You are given a csv file called "temp.csv" (comma delimited). Using the programming language of your choice, open the file, connect to the database, and populate the table storing customers in your schema (insert). The file is in the format SSN, First Name, Last Name, mobile phone number, email, ID, state, country.

   Below follows the R code for:
   - ✓ retrieving the customer data from Assignment_1_Customers.csv,
   - ✓ connecting to the database and,
   - ✓ populating the table "*Customer*" with this data.

```
library(RMySQL)

## Loading required package: DBI

#Load csv file to R
customer_csv <-read.csv(file="/Customers/Assignment_1_Customers.csv",
header=TRUE,sep=",",stringsAsFactors = FALSE)

#Connect to the DataBase
mydb <- dbConnect(MySQL(),user='root',password='admin2017',
                  dbname='dmbi_assignment1',host='127.0.0.1')
rs1 <- dbSendQuery(mydb, "SELECT * FROM customer")
data1 <- dbFetch(rs1,n=-1)

#make the csv's column names identical to the table's column names on
the
#DataBase
names(customer_csv)<-names(data1)

#Populate the table from csv
dbWriteTable(mydb, name="customer", value=customer_csv, overwrite=FAL
SE, append= TRUE,row.names=FALSE)
## [1] TRUE

#Check if the table is filled in
rs2 <- dbSendQuery(mydb, "SELECT * FROM customer")
data2 <- dbFetch(rs2,n=-1)
head(data2)

##   CustomerID FirstName  LastName                      Email
SSN
## 1          1      Pate Berisford      pberisford0@vimeo.com 160-66
-7934
## 2         10   Roxanna  Claussen    rclaussen9@edublogs.org 854-91
-0733
## 3        100   Coretta Jendrusch    cjendrusch2r@utexas.edu 181-90
-5311
## 4       1000   Dimitry   Patzelt dpatzeltrr@theguardian.com 366-36
-3079
## 5        101       Tad   Skerman     tskerman2s@youtube.com 599-74
-2983
```

```
## 6        102     Audry   McLaren      amclaren2t@nature.com 239-01
-4801
##      MobileNum StateAbbrev State Country
## 1 723-510-6026                     Syria
## 2 232-449-0621                     China
## 3 732-918-6034                   Bolivia
## 4 398-223-4189                     China
## 5 932-444-9756                     Libya
## 6 840-747-9892                     China
```

```
str(data2)
```

```
## 'data.frame':    1000 obs. of  9 variables:
##  $ CustomerID : chr  "1" "10" "100" "1000" ...
##  $ FirstName  : chr  "Pate" "Roxanna" "Coretta" "Dimitry" ...
##  $ LastName   : chr  "Berisford" "Claussen" "Jendrusch" "Patzelt"
...
##  $ Email      : chr  "pberisford0@vimeo.com" "rclaussen9@edublogs.
org" "cjendrusch2r@utexas.edu" "dpatzeltrr@theguardian.com" ...
##  $ SSN        : chr  "160-66-7934" "854-91-0733" "181-90-5311" "36
6-36-3079" ...
##  $ MobileNum  : chr  "723-510-6026" "232-449-0621" "732-918-6034"
"398-223-4189" ...
##  $ StateAbbrev: chr  "" "" "" "" ...
##  $ State      : chr  "" "" "" "" ...
##  $ Country    : chr  "Syria" "China" "Bolivia" "China" ...
```

```
dbClearResult(dbListResults(mydb)[[1]])
```

```
## [1] TRUE
```

```
dbDisconnect(mydb)
```

```
## [1] TRUE
```

5. **Question 5)**
   Using the programming language of your choice, connect to the database and implement query (i)– without using GROUP BY SQL statements, you are only allowed to use SELECT-FROM-WHERE. Best implementation gets a bonus ☺.

   Below the R code follows for connecting to the database and implementing query *(i).*

```
library(RMySQL)

## Loading required package: DBI

mydb <- dbConnect(MySQL(),user='root',password='admin2017',dbname='
dmbi_assignment1',host='127.0.0.1')
l<-c()
start1<-Sys.time()
for (i in 1:12){
  rs1<-dbSendQuery(mydb, paste0("

SELECT  SUM(CASE WHEN month(r1.pickupDate)<'", i,"' THEN amount ELS
E 0 END) as Previous,
        SUM(CASE WHEN month(r1.pickupDate)='", i,"' THEN amount ELS
E 0 END) as Current,
        SUM(CASE WHEN month(r1.pickupDate)>'", i,"' THEN amount ELS
E 0 END) as Following
FROM Rental as r1
WHERE year(r1.pickupDate)='2015'"))
  l<-append(l,dbFetch(rs1,n=-1))
}
end1<- Sys.time()
time1<-end1-start1
m<-matrix(unlist(unname(l)),ncol=3, byrow = TRUE)
colnames(m)<-c("Previous","Current","Following")
rownames(m)<-1:12
m
##    Previous Current Following
## 1         0    1554     16695
## 2      1554     970     15725
## 3      2524    1335     14390
## 4      3859     865     13525
## 5      4724    5498      8027
## 6     10222     574      7453
## 7     10796    1695      5758
## 8     12491    1132      4626
## 9     13623    1475      3151
## 10    15098     869      2282
## 11    15967    1320       962
## 12    17287     962         0
dbClearResult(dbListResults(mydb)[[1]])

## [1] TRUE

dbDisconnect(mydb)

## [1] TRUE
```

We ended up in this implementation, after comparing the execution times (shown below indicatively) among the following codes:

Table 1: Implementation codes for Question 5

	Solution Description	Code	Sys.time()

Code1 (final solution)	**For loop &1 query**	library(RMySQL)
		#library(tictoc)
		mydb <-
		dbConnect(MySQL(),user='root',password='DBAdmin.1908',dbname='DMBI_Assignment1',host='127.0.0.1')
		l<-c()
		start1<-Sys.time()
		for (i in 1:12){
		  rs1<-dbSendQuery(mydb, paste0(" SELECT SUM(CASE WHEN month(r1.pickupDate)<'", i,"' THEN amount ELSE 0 END) as Previous,
		                    SUM(CASE WHEN month(r1.pickupDate)='", i,"' THEN amount ELSE 0 END) as Current,
		                    SUM(CASE WHEN month(r1.pickupDate)>'", i,"' THEN amount ELSE 0 END) as Following
		                    FROM Rental as r1
		                    WHERE year(r1.pickupDate)='2015'"))
		  l<-append(l,dbFetch(rs1,n=-1))
		}
		end1<- Sys.time()
		time1<-end1-start1
		m<-matrix(unlist(unname(l)),ncol=3, byrow = TRUE)
		colnames(m)<-c("Previous","Current","Following")
		rownames(m)<-1:12
		dbClearResult(dbListResults(mydb)[[1]])
		dbDisconnect(mydb)
			0,1432s

Code2	**lapply & 1 query**	rm(list=ls())
		library(RMySQL)
		#library(tictoc)
		mydb <-
		dbConnect(MySQL(),user='root',password='admin2017',dbname='dmbi_assignment1',host='127.0.0.1')
		l<-c()
		start4<-Sys.time()
		list<-lapply(1:12, function(i){
		  rs1<-dbSendQuery(mydb, paste0(" SELECT SUM(CASE WHEN month(r1.pickupDate)<'", i,"' THEN amount ELSE 0 END) as Previous,
		                    SUM(CASE WHEN month(r1.pickupDate)='", i,"' THEN amount ELSE 0 END) as Current,
		                    SUM(CASE WHEN month(r1.pickupDate)>'", i,"' THEN amount ELSE 0 END) as Following
		                    FROM Rental as r1
		                    WHERE year(r1.pickupDate)='2015'"))
		        l<-append(l,dbFetch(rs1,n=-1))
		  return (list(l))
		})
		end4<- Sys.time()
		time4<-end4-start4
		time4
		m<-matrix(unlist(unname(list)),ncol=3, byrow = TRUE)
		colnames(m)<-c("Previous","Current","Following")
		rownames(m)<-1:12
		m
		dbClearResult(dbListResults(mydb)[[1]])
		dbDisconnect(mydb)
			0,1501s

Code3	**For loop &2 queries& calculations**	library(RMySQL)
		mydb <-
		dbConnect(MySQL(),user='root',password='admin2017',dbname='dmbi_assignment1',host='127.0.0.1')
			0,2566s

	Solution Description	Code	Sys.time()

		sumCurrent<-c()
		sumPrevious<-c()
		sumFollowing<-c()
		start4<- Sys.time()
		rs2<-dbSendQuery(mydb, paste0("SELECT sum(amount) FROM Rental WHERE year(pickupdate)='2015'"))
		total_amount<-dbFetch(rs2,n=-1)
		for (i in 1:12){
		  rs1<-dbSendQuery(mydb, paste0("SELECT sum(amount) as Current FROM Rental as r1
		  where year(r1.pickupDate)='2015' and month(r1.pickupDate)='", i,"' "))
		  sumCurrent<-append(sumCurrent, dbFetch(rs1,n=-1))
		  if (i==1) sumPrevious<-c(0) else sumPrevious<-append(sumPrevious,sumCurrent[[i-1]]+sumPrevious[[i-1]])
		  total_amount<-total_amount-sumCurrent[[i]]
		  sumFollowing<-append(sumFollowing,total_amount)
		}
		end4<- Sys.time()
		time4<- end4-start4
		time4
		l<-list(unname(sumPrevious),unname(sumCurrent),unname(sumFollowing))
		m1<-matrix(unlist(l),ncol=3, byrow = FALSE)
		colnames(m1)<-c("Previous","Current","Following")
		rownames(m1)<-1:12
		m1
		dbClearResult(dbListResults(mydb)[[1]])
		dbDisconnect(mydb)	

Code4	**Lapply & 3 queries**	rm(list=ls())
		library(RMySQL)
		#Connect to the DataBase
		mydb <- dbConnect(MySQL(),user='root',password='admin2017',
		            dbname='dmbi_Assignment1',host='127.0.0.1')
		sumCurrent<-c()
		sumPrevious<-c()
		sumFollowing<-c()
		start3<- Sys.time()
		list<-lapply(1:12, function(x){
		  rs1<-dbSendQuery(mydb, paste0("SELECT sum(amount) as Current FROM Rental as r1 WHERE year(r1.pickupDate)='2015' and month(r1.pickupDate)='", x,"'"))
		  sumCurrent<-append(sumCurrent, dbFetch(rs1,n=-1))
		  rs2<-dbSendQuery(mydb, paste0("SELECT sum(amount) as Previous FROM Rental as r1
		                  where year(r1.pickupDate)='2015' and month(r1.pickupDate)<'", x,"'"))
		  sumPrevious<-append(sumPrevious,dbFetch(rs2,n=-1))
		  rs3<-dbSendQuery(mydb, paste0("SELECT sum(amount) as Following FROM Rental as r1
		                  where year(r1.pickupDate)='2015' and month(r1.pickupDate)>'", x,"'"))
		  sumFollowing<-append(sumFollowing,dbFetch(rs3,n=-1))
		  list(sumPrevious,sumCurrent,sumFollowing)})
		end3<- Sys.time()
		time3<- end3-start3
		time3
		m<-matrix(unlist(list),ncol=3, byrow = TRUE)
		colnames(m)<-c("Previous","Current","Following")
		m
		dbClearResult(dbListResults(mydb)[[1]])
		dbDisconnect(mydb)
			0.4901s