Modern Data Management & Business Intelligence

Assignment #1

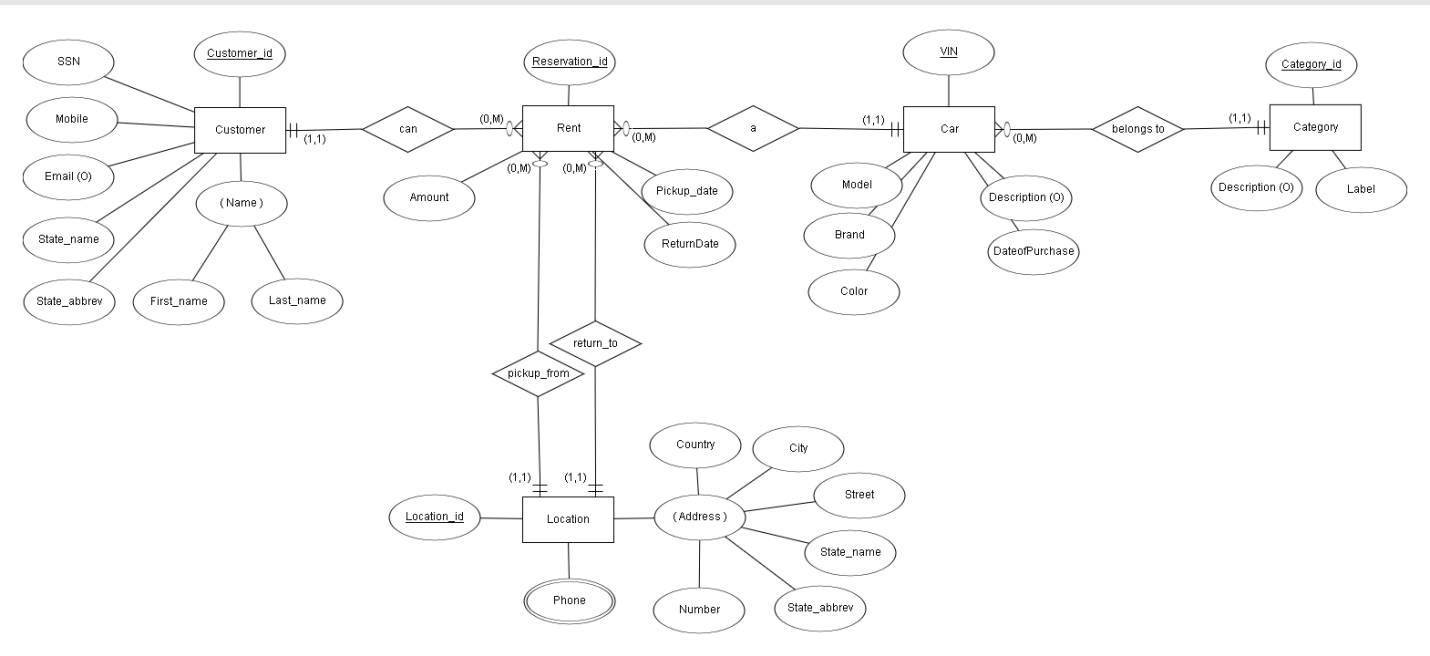
Team

Tsoulkas Giorgos
Papachristou Giorgos

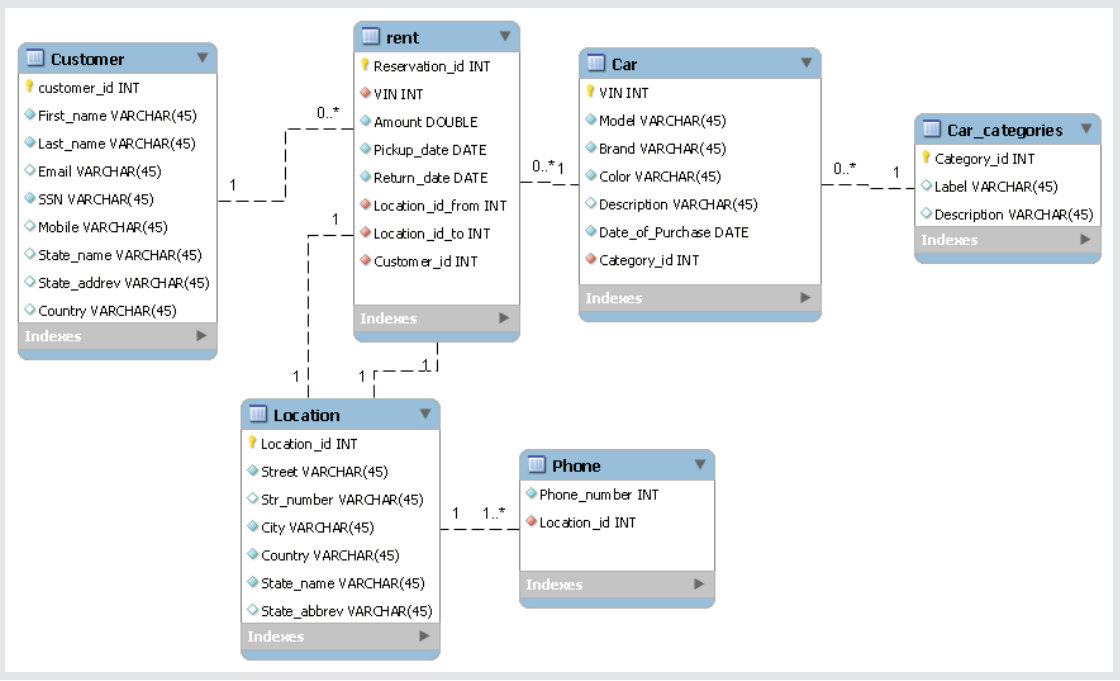
Report

Question-1

We created the **ERD** diagram with ERDPLUS online editor (You will find a png with high resolution in (**Question-1 folder**)



Also we have exported **EER** diagram from Workbench



Question-2

We have the Schema as a total and also each table query in (**Question-2 folder**)

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

-- -----------------------------------------------------
-- Table `mydb`.`Car_categories`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Car_categories` (
  `Category_id` INT NOT NULL AUTO_INCREMENT,
  `Label` VARCHAR(45) NULL,
  `Description` VARCHAR(45) NULL,
  UNIQUE INDEX `Label_UNIQUE` (`Label` ASC),
  UNIQUE INDEX `Category_id_UNIQUE` (`Category_id` ASC),
  PRIMARY KEY (`Category_id`))


-- -----------------------------------------------------
-- Table `mydb`.`Location`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Location` (
  `Location_id` INT NOT NULL,
  `Street` VARCHAR(45) NOT NULL,
  `Str_number` VARCHAR(45) NULL,
  `City` VARCHAR(45) NOT NULL,
  `Country` VARCHAR(45) NOT NULL,
  `State_name` VARCHAR(45) NOT NULL,
  `State_abbrev` VARCHAR(45) NULL,
  PRIMARY KEY (`Location_id`),
  UNIQUE INDEX `Location_id_UNIQUE` (`Location_id` ASC))


 -- -----------------------------------------------------
 -- Table `mydb`.`Car`
 -- -----------------------------------------------------
 CREATE TABLE IF NOT EXISTS `mydb`.`Car` (
   `VIN` INT NOT NULL,
   `Model` VARCHAR(45) NOT NULL,
   `Brand` VARCHAR(45) NOT NULL,
   `Color` VARCHAR(45) NOT NULL,
   `Description` VARCHAR(45) NULL,
   `Date_of_Purchase` DATE NOT NULL,
   `Category_id` INT NOT NULL,
   PRIMARY KEY (`VIN`),
   CONSTRAINT `Category_id`
     FOREIGN KEY (`Category_id`)
     REFERENCES `mydb`.`Car_categories` (`Category_id`)
     ON DELETE NO ACTION
     ON UPDATE NO ACTION,
```

```sql
-- -----------------------------------------------------
-- Table `mydb`.`Customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Customer` (
  `customer_id` INT NOT NULL AUTO_INCREMENT,
  `First_name` VARCHAR(45) NOT NULL,
  `Last_name` VARCHAR(45) NOT NULL,
  `Email` VARCHAR(45) NULL,
  `SSN` VARCHAR(45) NOT NULL,
  `Mobile` INT NULL,
  `State_name` VARCHAR(45) NULL,
  `State_addrev` VARCHAR(45) NULL,
  `Country` VARCHAR(45) NULL,
  PRIMARY KEY (`customer_id`),
  UNIQUE INDEX `SSN_UNIQUE` (`SSN` ASC),
  UNIQUE INDEX `customer_id_UNIQUE` (`customer_id` ASC))

-- -----------------------------------------------------
-- Table `mydb`.`rent`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`rent` (
  `Reservation_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `VIN` INT NOT NULL,
  `Amount` DOUBLE NOT NULL,
  `Pickup_date` DATE NOT NULL,
  `Return_date` DATE NOT NULL,
  `Location_id_from` INT NOT NULL,
  `Location_id_to` INT NOT NULL,
  `Customer_id` INT NOT NULL,
  PRIMARY KEY (`Reservation_id`),
  UNIQUE INDEX `Reservation_id_UNIQUE` (`Reservation_id` ASC),
  CONSTRAINT `Customer_id`
    FOREIGN KEY (`Customer_id`)
    REFERENCES `mydb`.`Customer` (`customer_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `VIN`
    FOREIGN KEY (`VIN`)
    REFERENCES `mydb`.`Car` (`VIN`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Location_id_to`
    FOREIGN KEY (`Location_id_to`)
    REFERENCES `mydb`.`Location` (`Location_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Location_id_from`
    FOREIGN KEY (`Location_id_from`)
    REFERENCES `mydb`.`Location` (`Location_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

```sql
-- -----------------------------------------------------
-- Table `mydb`.`Phone`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Phone` (
  `Phone_number` INT NOT NULL,
  `Location_id` INT NOT NULL,
  UNIQUE INDEX `Phone_number_UNIQUE` (`Phone_number` ASC),
  CONSTRAINT `loc_id`
    FOREIGN KEY (`Location_id`)
    REFERENCES `mydb`.`Location` (`Location_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

Question-3

We have all the SQL queries in (**Question-3 folder**), and below a preview of them

- Don't forget in queries with VIEWS to RUN seperate the CREATE VIEW and after run the SELECT.... queries

a)

```sql
SELECT reservation_id, location_id_from
FROM rent
WHERE pickup_date ='2015/05/20'
```

b)

```sql
SELECT First_name, Last_name, mobile
FROM customer, rent , car, car_categories
WHERE car_categories.label='luxury'
AND car.Category_id = car_categories.Category_id
AND rent.vin=car.vin
AND customer.customer_id=rent.Customer_id
```

c)

```sql
SELECT MONTH(rent.pickup_date) as Month, car.category_id as Category,
COUNT(rent.Reservation_id) as TOTAL FROM rent, car
WHERE rent.vin=car.vin
GROUP BY MONTH(rent.pickup_date), car.category_id
ORDER BY MONTH(rent.pickup_date)
```

d)

```sql
/*Firstly you run the create view and after the SELECT*/
CREATE VIEW V1 AS (SELECT location.State_abbrev, car.Category_id,
COUNT(*) AS SUMA FROM rent, car, location
WHERE rent.Location_id_from=location.location_id  and rent.vin=car.vin
GROUP BY location.State_abbrev, car.Category_id)


SELECT State_abbrev, Category_id FROM V1 AS E1
WHERE SUMA >= ALL( SELECT SUMA FROM V1 AS E2
                   WHERE E1.Category_id <> E2.Category_id
                   AND   E1.state_abbrev = E2.state_abbrev)
```

e)

```sql
/*Run seperate this view */
CREATE VIEW v( mon, avgmonth) AS SELECT MONTH(rent.pickup_date), avg(rent.amount)
FROM rent
WHERE YEAR(rent.pickup_date)=2015
GROUP BY MONTH(rent.pickup_date)

/*First create the view and after run this SELECT*/
SELECT MONTH(rent.pickup_date), count(rent.reservation_id) as Counter FROM rent, v
WHERE rent.amount>v.avgmonth
AND MONTH(rent.pickup_date)=v.mon
```

f)

```sql
CREATE VIEW NJ AS (SELECT MONTH(rent.pickup_date) as M, YEAR(rent.pickup_date) as Y, location.state_abbrev, count(*)as count_1 FROM rent, location
WHERE rent.location_id_from=location_id AND location.State_abbrev='NJ'
AND MONTH(rent.pickup_date)=5 AND YEAR(rent.pickup_date)
GROUP BY location.state_abbrev)

CREATE VIEW CA AS (SELECT MONTH(rent.pickup_date) as M, YEAR(rent.pickup_date) as Y, location.state_abbrev, count(*)as count_1 FROM rent, location
WHERE rent.location_id_from=location_id AND location.State_abbrev='CA'
AND MONTH(rent.pickup_date)=5 AND YEAR(rent.pickup_date)
GROUP BY location.state_abbrev)

CREATE VIEW NY AS (SELECT MONTH(rent.pickup_date) as M, YEAR(rent.pickup_date) as Y, location.state_abbrev, count(*)as count_1 FROM rent, location
WHERE rent.location_id_from=location_id AND location.State_abbrev='NY'
AND MONTH(rent.pickup_date)=5 AND YEAR(rent.pickup_date)
GROUP BY location.state_abbrev)

SELECT NJ.count_1 as'NJ', NY.count_1 as 'NY', CA.count_1 as 'CA' FROM NJ
LEFT JOIN NY ON NJ.M=NY.M AND NJ.Y=NY.Y
LEFT JOIN CA ON CA.M=NY.M AND CA.Y=NY.Y
```

g)

```sql
/*Run seperate this view */
CREATE VIEW v( mon, avgmonth) AS SELECT MONTH(rent.pickup_date), avg(rent.amount)
FROM rent
WHERE YEAR(rent.pickup_date)=2015
GROUP BY MONTH(rent.pickup_date)

/*First create the view and after run this SELECT*/
SELECT MONTH(rent.pickup_date), count(rent.reservation_id) as Counter FROM rent, v
WHERE rent.amount>v.avgmonth
AND MONTH(rent.pickup_date)=v.mon
```

h)

```sql
/*Run seperate these two views */

CREATE VIEW V2015( mon, count2015) AS SELECT MONTH(rent.pickup_date), count(rent.Reservation_id)
FROM rent
WHERE YEAR(rent.pickup_date)=2015
GROUP BY MONTH(rent.pickup_date)

CREATE VIEW V2014( mon, count2014) AS SELECT MONTH(rent.pickup_date), count(rent.Reservation_id)
FROM rent
WHERE YEAR(rent.pickup_date)=2014
GROUP BY MONTH(rent.pickup_date)

/*IMPORTANT: WE PRESENT ONLY THE MONTHS THAT HAD RENT IN BOTH YEARS*/

/*First create the 2 views and after run this SELECT*/
SELECT months, ((count2015-count2014)/count2014*100) as percentage from

/*Due to join the v2015.mon is equal with v2014.mon so we used one of the, */
(SELECT distinct  v2015.mon as months,  count2014, count2015 FROM v2014
   INNER JOIN v2015 ON v2014.mon=v2015.mon) as a
```

i)

```sql
/*Run seperate this  view */
CREATE VIEW V1 AS SELECT MONTH(RENT.PICKUP_DATE) AS M1, SUM(RENT.AMOUNT) AS S1 FROM RENT
WHERE YEAR(RENT.PICKUP_DATE)=2015
GROUP BY MONTH(RENT.PICKUP_DATE)


/*After the view run this query */
SELECT v2.Months, v2.Previous_Month, v2.This_Month, SUM(NX.S1) AS Next_Month FROM

(SELECT  V1.M1 AS Months ,SUM(PR.S1) AS Previous_Month, v1.s1 AS This_Month FROM V1
    LEFT JOIN V1 AS PR ON V1.M1>PR.M1
    GROUP BY Months
) AS v2

LEFT JOIN V1 AS NX ON V2.months<NX.M1
GROUP BY Months
```

Question-4

You will find the code (in R language) & temp.csv in (**Question-4 folder**)

- Don't forget to add your own credentials in order to connect to db

```r
#######################ATTENTION: WE HAVE REMOVED THE LOCAL PATH & PASSWORD#####################

library(MySQL)

#Get connection


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

# Upload File

rs<- dbSendQuery(mydb, "LOAD DATA LOCAL INFILE 'C:/#####/temp.csv.csv'
                 INTO TABLE customer FIELDS TERMINATED BY ','
                 LINES TERMINATED BY '\n' IGNORE 1 LINES")


data<- fetch(rs, n=-1)
```

Question-5

You will find the code (in R language) in (**Question-5 folder**)

```
#######################################     Library     #######################################
library("RMySQL")

####################################### Make a Connection #######################################
mydb<-dbConnect(MySQL(), user='root', password='####', dbname='mydb',host='127.0.0.1')

#######################################   First Query   #######################################
rs<- dbSendQuery(mydb, "SELECT MONTH(RENT.PICKUP_DATE) AS M1, RENT.AMOUNT AS S1
                        FROM RENT
                        WHERE YEAR(RENT.PICKUP_DATE)=2015")
t<-fetch(rs, n=-1)

# Order the data
t<-t[order(t$M1),]



################################    A Function Instead of 'Group by'  #########################

f<- function(x){

  #Initialization
  suma<-numeric(length(x$M1))
  suma[x$M1[1]]<-x$S1[1]
  names(suma)[1]<-x$M1[1]

  for(i in c(1:length(x$M1)))
  {
   if(!is.na(x$M1[i+1]))
    {
      if(x$M1[i]==x$M1[i+1])
        {
          suma[x$M1[i]]<-suma[x$M1[i]]+x$S1[i+1]
          names(suma)[x$M1[i]]<-x$M1[i]
        }
      else
        {
          suma[x$M1[i+1]]<-x$S1[i+1]
          names(suma)[x$M1[i+1]]<-x$M1[i+1]
        }

    }
   }
  suma<-suma[suma!=0]
  suma<-data.frame(suma,names(suma))
  names(suma)[1]<-paste("S1")
  names(suma)[2]<-paste("M1")
  return(suma)
}
```

```
# Create the table V1 ##################################################################
v1<-f(t)

# Create sum of NEXT months ############################################################
next_month<-c()
for(i in c(1:length(v1$M1)))
{
  next_month[i]<-sum(v1$S1[(i+1):12], na.rm=TRUE)
}

# Create sum of PREVIOUS months ########################################################
previous_month<-c()
for(i in c(1:length(v1$M1)))
{
  previous_month[i]<-sum(v1$S1[(1):i-1], na.rm=TRUE)
}

# Final table ##########################################################################
table<-cbind(v1,previous_month,next_month)
print(table[,c(2,3,1,4)])  #Re-order the columns in order to be more easy to read
```

Below we have the results of **query (i)** and **question-5**

- Don't forget to add your own credentials in order to connect to db
- Run the script line by line

This is the result table of the query (i)

Months	Previous_Month	This_Month	Next_Month
1	NULL	10	318
2	10	12	306
5	22	191	115
8	213	46	69
9	259	26	43
11	285	25	18
12	310	18	NULL

The result of the script has the same result with the query (i)

```
   M1 previous_month  S1 next_month
1   1              0  10        318
2   2             10  12        306
5   5             22 191        115
8   8            213  46         69
9   9            259  26         43
11 11            285  25         18
12 12            310  18          0
```