Faculty of Engineering & Technology
Electrical & Computer Engineering Department

ENCS313

Python Project Report

Prepared By :

Tareq Shannak 1181404

Abdalrahman Mansour 1182955

Date: 30/8/2020

Objectives

- Gain Programming Skills Using Python.
- Know How Much OOP makes the programmer more comfortable when deals with data base.
- Take benefits from libraries (like: datatime), and dealing with modules.

Table of Contents

CarRental

In the first of the report we will talk about file CarRental which contains many classes that help us in each task. In figure1 the UML of the CarRental file.

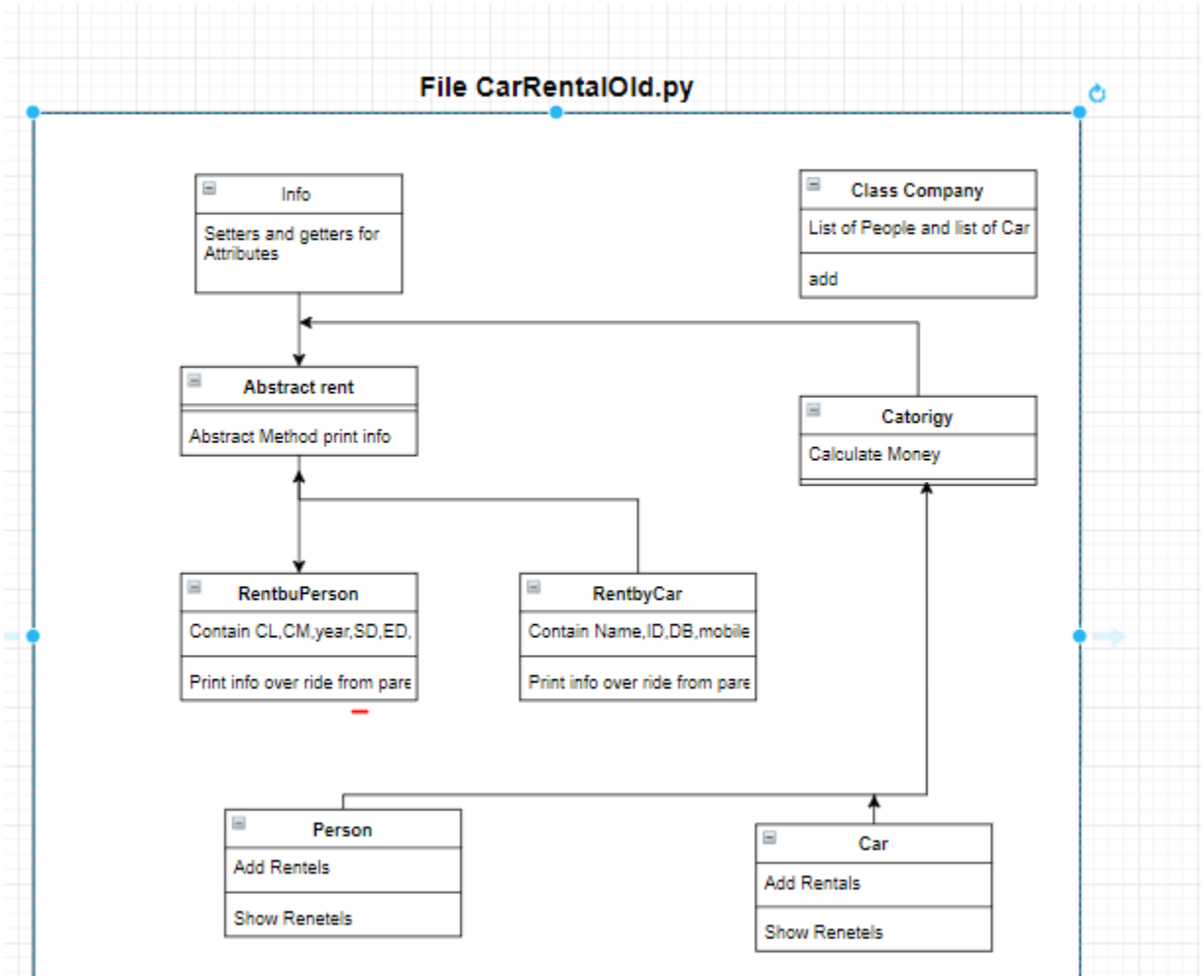


Figure 1

The CarRental contains company class and info class .the info class I the parent of the abstractrent class and catorigy class. The class abstractrent is the have two subclasses rentbyperson and rentbycar. The catorigy class is parent of two class car and person.

In figure 2 first part of code of the CarRental.

```
class Company:
    nDuplicates = 0
    nDatechange = 0
    nDropnames = 0
    nDropids = 0
    nDropdob = 0
    nDropmobiles = 0
    nDroppentry = 0
    nDropcmake = 0
    nDropcids = 0
    nDropcyears = 0

    def __init__(self):
        self.__listOfPeople = []
        self.__listOfCars = []

    def __del__(self):
        print('DataBase Will Be Removed!\n')

    def getListOfPeople(self):
        return self.__listOfPeople

    def getListOfCars(self):
        return self.__listOfCars
```

Figure 2

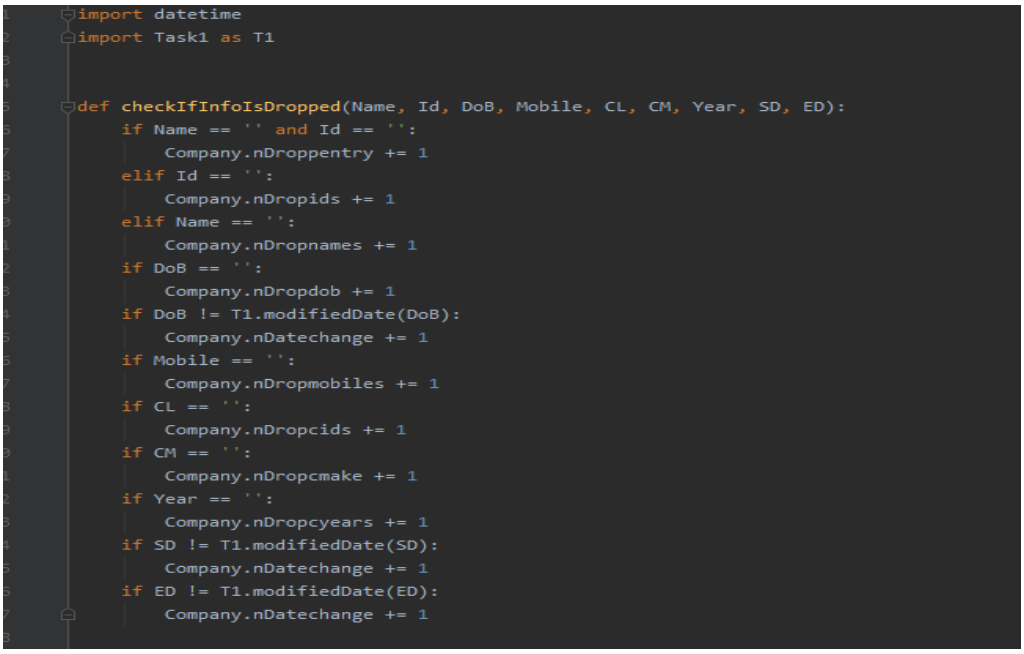
Company class: we defined variables and initialize it zero to counter the number of errors in the data (nDuplicates ……. nDropcyears). The constructer is containing two empty lists which will fill when we read the data from the file. We implemented getters and setters method to get the private lists. We implement the Del method which is remove the data base (self-object) .the most important method on the class company is the add method because it is add data to the list and fill the missing data. In figure 3 in the next page show the first part of the add method.

```
def add(self, Name='', Id='', DoB='', Mobile='', CL='', CM='', Year='', SD='', ED='', RB=''):
    flag = 1
    if ((Id.isdigit() or len(Id) == 0) and (Mobile.isdigit() or len(Mobile) == 0) and (
            CL.isalnum() or len(CL) == 0) and (CM.isalpha() or len(CM) == 0) and (
            Year.isdigit() or len(Year) == 0) and (RB.isdigit() or len(RB) == 0)):
        checkIfInfoIsDropped(Name, Id, DoB, Mobile, CL, CM, Year, SD, ED)
        new = {"Name": Name, "Id": Id, "DoB": DoB, "Mobile": Mobile, "CL": CL, "CM": CM, "Year": Year,
               "SD": SD, "ED": ED, "RB": RB}
        for obj in self.getListOfPeople():
            for rent in obj.getlistOfRents():
                if CL == '':
                    break
                if rent.getCL() == CL:
                    T1.fixCarInfo(rent, new)
            if Id == '':
                break
            if obj.getId() == new["Id"]:
                T1.fixPersonInfo(obj, new)
                flag = 0
                obj.addRental(new["CL"], new["CM"], new["Year"], new["SD"], new["ED"], new["RB"])
                break
        if flag:
            self.getListOfPeople().append(
                Person(new["Name"], new["Id"], new["DoB"], new["Mobile"], new["CL"], new["CM"],
                       new["Year"], new["SD"], new["ED"], new["RB"]))
        flag = 1
        for obj in self.getListOfCars():
```

Figure 3

Add received the attributes name ,id , DoB , mobile ,CL , CM ,Year ,SD ,ED and RB . first we set the flag to one which will used it to append the data to the list of people if the flag is equal one . First we check the data is true (ID is digit without any character , the car made is alphabet for all attributes) .

If the data is true then we call the method in the file CarRental CheckInfoIsDropped to counter the number of errors in the data for each type .show figure 3-2 the implementation of method CheckInfoIsDropped.



```
import datetime
import Task1 as T1


def checkIfInfoIsDropped(Name, Id, DoB, Mobile, CL, CM, Year, SD, ED):
    if Name == '' and Id == '':
        Company.nDroppentry += 1
    elif Id == '':
        Company.nDropids += 1
    elif Name == '':
        Company.nDropnames += 1
    if DoB == '':
        Company.nDropdob += 1
    if DoB != T1.modifiedDate(DoB):
        Company.nDatechange += 1
    if Mobile == '':
        Company.nDropmobiles += 1
    if CL == '':
        Company.nDropcids += 1
    if CM == '':
        Company.nDropcmake += 1
    if Year == '':
        Company.nDropcyears += 1
    if SD != T1.modifiedDate(SD):
        Company.nDatechange += 1
    if ED != T1.modifiedDate(ED):
        Company.nDatechange += 1
```

Figure 4

The method is countered the errors but remember that sometimes more than one error in the same object for example some object has missing the name and the date is in the wrong format because of change of date error format we don't use else if (**elif**) in check dates .

We return to the explain of class company. after that we defined a dictionary with keys are the name of the attributes and the value for each key is the value that passed throw the method add .after that there are two loops : first loop is to pass throw the list of the people who is rented any car .the second loop is to pass throw the list of cars rented .

```
        for obj in self.getListOfCars():
            for rent in obj.getlistOfRents():
                if Id == '':
                    break
                if rent.getId() == Id:
                    T1.fixPersonInfo(rent, new)
            if CL == '':
                break
            if obj.getCL() == new["CL"]:
                T1.fixCarInfo(obj, new)
                flag = 0
                obj.addRental(new["Name"], new["Id"], new["DoB"], new["Mobile"], new["SD"], new["ED"], new["RB"])
                break
        if flag:
            self.getListOfCars().append(Car(new["Name"], new["Id"], new["DoB"], new["Mobile"], new["CL"], new["CM"],
                                            new["Year"], new["SD"], new["ED"], new["RB"]))
    else:
        print("There is at least one wrong input!")
```

Figure 5

Figure 5 is from add-Function in Company Class, this function checks if the information has an Id or not, if it has no Id then it will be added as person object. If there is an id and it is defined in old person object, then check if it need information like mobile, DoB and Name either the new rent or the old one. The same idea is with Car Object.

```python
class Info:
    """Define a New Car Rental"""

    def setName(self, Name):
        self.__Name = Name

    def getName(self):
        return self.__Name

    def setId(self, Id):
        self.__Id = Id

    def getId(self):
        return self.__Id

    def setDoB(self, DoB):
        self.__DoB = T1.modifiedDate(DoB)

    def getDoB(self):
        return self.__DoB

    def setMobile(self, Mobile):
        self.__Mobile = Mobile

    def getMobile(self):
        return self.__Mobile
```

Figure 6

Class info : figure 6 show the class .this class is to get information about the object like name of the person , ID ,DoB and other attributes using getters method ,note getters method is inherit to subclass but when get the DoB and other Dates we return the correct form using the method modifiedDate in the class Task1 using import as shown in the figure 6.

```python
import datetime
import Task1 as T1
```

Figure 7

Class abstractRent: this is an abstract class contain the method printInfo which must be implemented in their subclasses(RentByPerson and RentByCar) .In figure7 show the class abstractRent .

```
class abstractRent(Info):
    def printInfo(self):
        raise NotImplementedError("Subclass must implement abstract method")
```

Figure 8

Class RentByPerson: show figure8 to know the code.

```
class RentByPerson(abstractRent):

    def __init__(self, CL='', CM='', Year='', SD='', ED='', RB=''):
        self.setCL(CL)
        self.setCM(CM)
        self.setYear(Year)
        self.setSD(SD)
        self.setED(ED)
        self.setRB(RB)

    def printInfo(self):
        print(
            "  CL: " + self.getCL() + ", CM: " + self.getCM() + ", Year: " + self.getYear() + ", SD: " + self.getSD() +
            ", ED: " + self.getED() + ", RB: " + self.getRB())
```

Figure 9

This is help us in task3 to add information about new cars and print t description about it .Notice that the method printInfo that implement in this class is an override for the method in abstractRent .

Class Rent by Car : the same thing as the previous class (Class RentByPerson) but the different between them in attributes which is for person description (Name , ID , DoB ,mobile..etc) .the code in figure 9 in the next page .

```
class RentByCar(abstractRent):
    def __init__(self, Name='', Id='', DoB='', Mobile='', SD='', ED='', RB=''):
        self.setName(Name)
        self.setId(Id)
        self.setDoB(DoB)
        self.setMobile(Mobile)
        self.setSD(SD)
        self.setED(ED)
        self.setRB(RB)

    def printInfo(self):
        print(
            "  Name: " + self.getName() + ", Id: " + self.getId() + ", Date Of Birth: " + self.getDoB() + ", Mobile: " +
            self.getMobile() + ", SD: " + self.getSD() + ", ED: " + self.getED() + ", RB: " + self.getRB())
```

Figure 10

Class catorigy: this is an subclass for class info in this class we implement an abstract method which is must be in their child and we handled the exception not implemented, also it contains the method calculate money to calculate the total paid of rents car for the object (for the person) . Show the figure 10 to know the code.

```
class abstractCategory(Info):
    def getlistOfRents(self):
        raise NotImplementedError("Subclass must implement abstract method")

    def addRental(self):
        raise NotImplementedError("Subclass must implement abstract method")

    def showRentals(self):
        for obj in self.getlistOfRents():
            obj.printInfo()

    def printInfo(self):
        raise NotImplementedError("Subclass must implement abstract method")

    def calculateMoney(self):
        summation = 0
        for rent in self.getlistOfRents():
            summation += int(rent.getRB())
        return summation
```

Figure 11

Class Person : In figure 11 show the class person .

```
class Person(abstractCategory):
    """Who rents the car"""

    def getlistOfRents(self):
        return self.__listOfRents

    def addRental(self, CL='', CM='', Year='', SD='', ED='', RB=''):
        if T1.isDuplicatedPerson(self, CL, CM, Year, SD, ED, RB):
            Company.nDuplicates += 1
            return
        self.getlistOfRents().append(RentByPerson(CL, CM, Year, SD, ED, RB))

    def __init__(self, Name='', Id='', DoB='', Mobile='', CL='', CM='', Year='', SD='', ED='', RB=''):
        self.setName(Name)
        self.setId(Id)
        self.setDoB(DoB)
        self.setMobile(Mobile)
        self.__listOfRents = []
        self.addRental(CL, CM, Year, SD, ED, RB)

    def printInfo(self):
        print(
            "* Name: " + self.getName() + ", ID: " + self.getId() + ", Date Of Birth: " + self.getDoB() + ", Mobile: " +
            self.getMobile() + " *")
```

Figure 12

The constructer takes the all attributes and use setters in class info to set new object. in the contractor we use the instance method add rental to add the object to the list of rents but we check if it is an object with the same parameters using method is Duplicated Person in the T file if it is Duplicate we add the duplicated counter .else we add to the list of rents . There is the print info method which is print information for the object (about the person not car).

Class Car: the constructer is the same as the constructer in the person class

But the different we use to check data in the add rental method the duplicated car method from Task 1 file to check if the object is exist previously else add to the rental list .the print info method it prints details about the car not person .Figure 12 in the next page show the class .

```
248
249 class Car(abstractCategory):
250     """Which be rented"""
251
252     def getlistOfRents(self):
253         return self.__listOfRents
254
255     def addRental(self, Name='', Id='', DoB='', Mobile='', SD='', ED='', RB=''):
256         if T1.isDuplicatedCar(self, Name, Id, DoB, Mobile, SD, ED, RB):
257             return
258         self.getlistOfRents().append(RentByCar(Name, Id, DoB, Mobile, SD, ED, RB))
259
260     def __init__(self, Name='', Id='', DoB='', Mobile='', CL='', CM='', Year='', SD='', ED='', RB=''):
261         self.setCL(CL)
262         self.setCM(CM)
263         self.setYear(Year)
264         self.__listOfRents = []
265         self.addRental(Name, Id, DoB, Mobile, SD, ED, RB)
266
267     def printInfo(self):
268         print(
269             "* CL: " + self.getCL() + ", CM: " + self.getCM() + ", Year: " + self.getYear() + " *")
270
271
272
```

Figure 13

Task1

The first part of the code Task1 is in figure 14 below.

```
import datetime
import re
import Task2
import Task3
import Task4
import CarRental


def modifiedDate(date):
    if date != '':
        dateFields = list(re.split('[-/]', date))
        if len(dateFields) == 1:
            return date
        x = datetime.datetime(int(dateFields[2]), int(dateFields[1]), int(dateFields[0]))
        date = x.strftime("%d %B %Y")
    return date


def fixCarInfo(rent, newRent):
    if rent.getCM() == '':
        rent.setCM(newRent["CM"])
    elif newRent["CM"] == '':
        newRent["CM"] = rent.getCM()
    if rent.getYear() == '':
        rent.setYear(newRent["Year"])
    elif newRent["Year"] == '':
        newRent["Year"] = rent.getYear()
```

Figure 14

The first method is to modified date, first must check if date is missing data or not before change data then we put the split data from (– or /) in list we check if the length of the list is one this means that the date is in the standard form so we return the date else the length of the list is 3 so we use class date to convert the list to date then use **strftime** method to convert to the standard date.

The second method in figure 14 and the Clarification of it in figure 15.

```
def fixCarInfo(rent, newRent):
    if rent.getCM() == '':
        rent.setCM(newRent["CM"])
    elif newRent["CM"] == '':
        newRent["CM"] = rent.getCM()
    if rent.getYear() == '':
        rent.setYear(newRent["Year"])
    elif newRent["Year"] == '':
        newRent["Year"] = rent.getYear()
```

Figure 15

This method is to fill missing data of car and it take two parameters object and dictionary if the CM and Year is missing from one then fill it from the other.

```
def fixPersonInfo(rent, newRent):
    if rent.getName() == '':
        rent.setName(newRent["Name"])
    elif newRent["Name"] == '':
        newRent["Name"] = rent.getName()
    if rent.getDoB() == '':
        rent.setDoB(newRent["DoB"])
    elif newRent["DoB"] == '':
        newRent["DoB"] = rent.getDoB()
    if rent.getMobile == '':
        rent.setMobile(newRent["Mobile"])
    elif newRent["Mobile"] == '':
        newRent["Mobile"] = rent.getMobile()


def writeOnFile(fo, obj, rent):
    try:
        fo.write(
            obj.getName() + ";" + obj.getId() + ";" + obj.getDoB() + ";" + obj.getMobile() + ";" + rent.getCL() + ";" +
            rent.getCM() + ";" + rent.getYear() + ";" + rent.getSD() + ";" + rent.getED() + ";" + rent.getRB() + "\n")
    except IOError:
        print("Error: can't write data on the file")
```

Figure 16

In figure 16 the second part of the Task1. FixPersonInfo methods to complete information about the person as method FixCarInfo takes two parameter dictionary and object it fill name and DoB and mobile from each other.

The method WriteOnFile is take three parameter file to write on it ,object from class person handle information about person (Name ,ID ,DoB ,mobile) , and object from class Car handle the information about the car and available date of rented for the person we write to file using write method of file .

We handle the IOerror exception if we can't write to the file.

In figure 17 below, method to print the missing data in file and the complete data in other file.

```
52
53
54  def moveDataBase(company):
55      try:
56          foCompleted = open("CarRentalCompleted.txt", "w")
57          foMissing = open("CarRentalMissing.txt", "w")
58          for obj in company.getListOfPeople():
59              if obj.getName() == '' or obj.getId() == '' or obj.getDoB() == '' or obj.getMobile() == '':
60                  for rent in obj.getlistOfRents():
61                      writeOnFile(foMissing, obj, rent)
62              else:
63                  for rent in obj.getlistOfRents():
64                      if (
65                              rent.getCL() == '' or rent.getCM() == '' or rent.getYear() == '' or rent.getSD() == '' or
66                              rent.getED() == '' or rent.getRB() == ''):
67                          writeOnFile(foMissing, obj, rent)
68                      else:
69                          writeOnFile(foCompleted, obj, rent)
70          foMissing.close()
71          foCompleted.close()
72      except IOError:
73          print("Error: can't write data or find file")
74
```

Figure 17

This method takes object from Class Company; first we open two files to print missing and complete data. Then we pass throw the list of people and check if any attributes of the person not car is empty then pass throw all cars that rents from this object and print it in the missing file. on the other hand we pass throw the cars that rented from the object and check if any information of car is empty then write in missing file else write in complete file because

neither the information about the person(Name, ID, DoB……etc) nor the information about the car (CM, CL, Year….etc) is empty .

We handle the IOerror exception if they don't able to write in any file.

The final part of Task1 is in figure 18 below.

```
75
76  def isDuplicatedPerson(person, CL, CM, Year, SD, ED, RB):
77      for rent in person.getlistOfRents():
78          if (rent.getCL() == CL and rent.getCM() == CM and rent.getYear() == Year and rent.getSD() == SD and
79                  rent.getED() == ED and rent.getRB() == RB):
80              return 1
81      return 0
82
83
84  def isDuplicatedCar(car, Name, Id, DoB, Mobile, SD, ED, RB):
85      for rent in car.getlistOfRents():
86          if (
87                  rent.getName() == Name and rent.getId() == Id and rent.getDoB() == DoB and rent.getMobile() ==
88                  Mobile and rent.getSD() == SD and rent.getED() == ED and rent.getRB() == RB):
89              return 1
90      return 0
```

Figure 18

The method isDuplicatedPerson take object from class person and CL ,CM ,Year , SD, ED and RB then we pass throw that rented of this person and compare if information of any cars rented by this person is the same with the passed parameter then return one else return zero . The other method is isDuplicatedCar which is opposite of the isDuplicatedPerson because it takes object of type car and parameter like Name, ID, DoB, mobile, SD, ED and RB. Then it pass throw all person that rented this car and check if it has the same information then return 1 else return 0.

Task2

The first method is in figure 19 below.

```
import Task1
import Task3
import Task4
import CarRental
def inquiryPerson(company):
    mode = input("a. Inquiry Using Name\tb. Inquiry Using ID\n")
    if mode == 'a':
        wantedName = input("Write the name:\n")
        wantedId = ''
    elif mode == 'b':
        wantedName = ''
        wantedId = input("Write the id:\n")
    else:
        print("Wrong Input!\n")
        return
    flag = 1
    for person in company.getListOfPeople():
        if (person.getName().lower() == wantedName.lower() and mode == 'a') or (
                person.getId() == wantedId and mode == 'b'):
            flag = 0
            person.printInfo()
            person.showRentals()
            print("   The Amount Paid By This Person For Renting Cars: " + str(person.calculateMoney()))
            break
    if flag:
        print("No Person With This Name/ID!\n")
```

Figure 19

The method inquiry Person take an object from class company ,first set flag one ,then we asked the user to choose between inquiry using name "a" or ID "b" then we check the choice if it is "a" then ask user to input the name if the choice is "b" then ask user to input the id . after that we will pass throw the list of people of the object company and check if the choice is "a" and the same name or it chose "b" and the id is the same with the object then print information about this person and the car rentals and the amount that paid from this person notice that flag is equal zero and repeat this operation until end the list. If flag is one then there is no person or ID so we use flag to know if there are a person with this name or ID or not have any of them.

The second method of task2 is in figure20 below.

```
def inquiryCar(company):
    cl = input("Write the CL:\n")
    flag = 1
    for car in company.getListOfCars():
        if car.getCL().lower() == cl.lower():
            flag = 0
            car.printInfo()
            car.showRentals()
            print("   The Revenue Made By Renting This Car: " + str(car.calculateMoney()))
            break
    if flag:
        print("No Car with this CL!\n")
```

Figure 20

The inquiry car method take object from class company this method is to inquiry using CL so we asked user to enter the CL then we set flag to one to know if there is car with the entered CL or not . we pass throw the list of cars of the object and checked the CL from list is equal to the entered CL or not if equal then print the information of the car and the revenue from renting this car using method calculate money , also clear flag to zero and break because there is no two cars with same CL . then check if the flag is set this mean there is no car with the entered CL .

Task3

In figure 21 below show the first part of the first method.

```
def isAvailable(self, WantedSD, WantedED):
    months = {"January": 1, "February": 2, "March": 3, "April": 4, "May": 5, "June": 6, "July": 7, "August": 8,
              "September": 9, "October": 10, "November": 11, "December": 12}
    WantedSDcols = T1.modifiedDate(WantedSD).split(" ")
    WantedEDcols = T1.modifiedDate(WantedED).split(" ")
    for rent in self.getlistOfRents():
        SDcols = rent.getSD().split(" ")
        EDcols = rent.getED().split(" ")
        if (datetime.date(int(EDcols[2]), months[EDcols[1]], int(EDcols[0])) >=
            datetime.date(int(WantedSDcols[2]), months[WantedSDcols[1]], int(WantedSDcols[0])) >=
            datetime.date(int(SDcols[2]), months[SDcols[1]], int(SDcols[0])) or \
                (datetime.date(int(EDcols[2]), months[EDcols[1]], int(EDcols[0])) >=
                 (datetime.date(int(WantedEDcols[2]), months[WantedEDcols[1]], int(WantedEDcols[0])) >=
                 datetime.date(int(SDcols[2]), months[SDcols[1]], int(SDcols[0]))):
            return 0
    return 1
```

Figure 21

The method take object and two string (SD, ED). First we defined the dictionary months to know the each month in integer. We want to convert the SD, ED to the standard form using modifiedDate in task 1 because we take them from user and split the spaces . Then we pass throw the list of the car rents from object self and get SE, ED and split it from spaces then we compare the date that the user entered of the self-object (Car specification) if it is available to rented or not if by return 0 or 1.

The second method in task3 in figure 22 below .

```
import Task1 as T1
import Task2
import Task4
import CarRental
import datetime
def newCarRental(company):
    dictionary = {'Name: ': '', 'Id: ': '', 'Date Of Birth: ': '', 'Mobile: ': '', 'Car Rent Start Date: ': '',
                  'Car Rent End Date: ': ''}
    listOfAvailableCars = []
    for element in dictionary.keys():
        dictionary[element] = input(element)
    for car in company.getListOfCars():
        if isAvailable(car,dictionary["Car Rent Start Date: "], dictionary["Car Rent End Date: "]):
            listOfAvailableCars.append(car)
    i = 0
```

Figure 22

The method takes object of company. First we defined a dictionary with keys is the same name of the attributes (Name, ID…..etc). we defined list of the available cars to add the available car to it .we pass throw the keys of dictionary to fill it with proper data .then we pass throw the list of the cars and check if any of cars is available and add it to the list of available car .the second part of the code in figure 22 below.

```
i = 0
for car in listOfAvailableCars:
    i += 1
    print("*" + str(i), end="")
    car.printInfo()
if i == 0:
    print("No Available Cars During These Days, Sorry!\n")
    return
WantedCar = input("Enter The Number Of Car That You Want: ")
WantedCar = int(WantedCar) - 1
RB = input("The Amount Paid For This Rental: ")
company.add(dictionary["Name: "], dictionary["Id: "], dictionary["Date Of Birth: "], dictionary["Mobile: "],
            listOfAvailableCars[WantedCar].getCL(), listOfAvailableCars[WantedCar].getCM(),
            listOfAvailableCars[WantedCar].getYear(), dictionary["Car Rent Start Date: "],
            dictionary["Car Rent End Date: "],
```

Figure 23

We set i to zero then pass throw the list of available car and print the information of the cars then we check if i equal to zero this mean there is no car available so we end the operation . Else we asked user to inter the number of car that he want and the amount RB then we add a new object with its information (name, ID, DoB, mobile, CL,CM, year ,SD…etc) using dictionary and list of the car at the index that he want .

```
T1.moveDataBase(company)
print("The Car Rental Was Added!")
```

This code is to add object to data base and print the car rental was added .

Task4

Figure 24 show the code of task 4

```
import CarRental


def carStatistics(company):
    for car in company.getListOfCars():
        car.printInfo()
        print("   Number of days the car was rented: " + str(calculateRentedDays(car)) +
              "\n The Revenue Made By Renting This Car: " + str(car.calculateMoney()) +
              "\nAverage price per day for renting each car: " + str(car.calculateMoney() / calculateRentedDays(car)))


def calculateRentedDays(self):
    months = {"January": 1, "February": 2, "March": 3, "April": 4, "May": 5, "June": 6, "July": 7, "August": 8,
              "September": 9, "October": 10, "November": 11, "December": 12}
    summation = 0
    for rent in self.getlistOfRents():
        SDcols = rent.getSD().split(" ")
        EDcols = rent.getED().split(" ")
        summation += int((
                                datetime.date(int(EDcols[2]), months[EDcols[1]], int(EDcols[0])) -
                                datetime.date(int(SDcols[2]), months[SDcols[1]], int(SDcols[0]))).days)
        del SDcols
        del EDcols
    return summation
```

Figure 24

The carStatitics method is to print the number of days of the car that rented using method calculateRentedDays the revenue made by renting the car and the average price by divided the calculated money over the number of days. We pass throw each car in the list of cars and print the information of this car.

The calculateRentedDays method take object and calculate the number of days .first we define a months as dictionary to know the value of the month as integer. then we pass throw all person who is rented this type of the car .we save the date SD and ED as string with spilt spaces from it then add to summation the difference start date and end date after change it to integer .

We delete the strings in each loop to avoid take large area of memory.

Main

This is the first part of the code

```
import Task1 as T1
import Task2 as T2
import Task3 as T3
import Task4 as T4
import CarRental as C
import re


def readFile(company):
    try:
        fo = open("CarRentalOld.txt", "r")
        Line_exists = fo.readline()  # Read 1 line
        flag = 1  # to know if there is information in the file or no
        while Line_exists:
            Line_exists = re.sub(r'\n', "", Line_exists)  # remove \n
            fields = list(Line_exists.split(";"))
            flag = 0
            if len(fields) != 10:
                print("Wrong Details in the file!")
                company.__del__()
                break
            company.add(fields[0], fields[1], fields[2], fields[3], fields[4], fields[5], fields[6], fields[7],
                        fields[8],
                        fields[9])
            Line_exists = fo.readline()
```

Figure 25

We defined a method take one object of company to read data from file called CarRentalOld then we pass throw all lines that read and spilt "\n" then spilt the semi colon and save the result as list of fields then we check if the length of the data not equal 10.Then the data is wrong else we add the fields to the company. We repeated this operation until reach the end of the file. .we set flag to one to check if the file has any data or not. We convert the flag to zero if the file has any data. We handle the IOerror exception if the file not founded.

We check if the flags equal one then the file is empty .look to the last 6 lines in figure 26 below.

The second part of the main code in figure 25 .

```
            Line_exists = fo.readline()
        if flag:
            print("No Information in this file!\n")
        fo.close()
    except IOError:
        print("Error: can't find file or read data")


def readOption():
    return input("Please Enter Number Of Option To Execute:\n"
                 "1. Inquiry About A Person.\n"
                 "2. Inquiry About A Car\n"
                 "3. Add A New Car Rental.\n"
                 "4. Show Statistics About Each Car.\n"
                 "5. Exit Program.\n")


def printMissingSummary(company):
    print("           Summary of data missing in the database:"
          "\nNumber of duplicate entries in the database = " + str(company.nDuplicates) +
          "\nNumber of entries with wrong date format in the database = " + str(company.nDatechange) +
          "\nNumber of entries where names are dropped from the database = " + str(company.nDropnames) +
          "\nNumber of entries where Ids are dropped from the database = " + str(company.nDropids) +
          "\nNumber of entries where dob are dropped from the database = " + str(company.nDropdob) +
          "\nNumber of entries where mobile numbers are dropped from the database = " + str(company.nDropmobiles) +
          "\nNumber of entries where personal entry can not be completed = " + str(company.nDroppentry) +
```

Figure 26

The method read file is to ask user to choose what he want and return his choice . The method print missing summary take object company and it prints the number of Duplicates and drops and so on .

```
def printCompletedSummary(company):
    print("           Summary of data recovered from the database:"
          "\nNumber of duplicate entries removed from the new database = " + str(company.nDuplicates) +
          "\nNumber of entries with wrong date format fixed in the new database = " + str(company.nDatechange) +
          "\nNumber of entries with names recovered in the new database = " + str(company.nDropnames) +
          "\nNumber of entries with Ids recovered in the new database = " + str(company.nDropids) +
          "\nNumber of entries with dob recovered in the new database = " + str(company.nDropdob) +
          "\nNumber of entries with mobile numbers recovered in the new database = " + str(company.nDropmobiles) +
          "\nNumber of entries with car make recovered in the new database = " + str(company.nDropcmake) +
          "\nNumber of entries with car models (year) recovered in the new database = " + str(
        company.nDropcyears) + "\n")
```

Figure 27

In figure 27 above methods to print summery of data .it takes object of company to use it to print the summery.

```
71      ################ MAIN ################
72      try:
73          company118 = C.Company()
74          readFile(company118)
75          printMissingSummary(company118)
76          printCompletedSummary(company118)
77          T1.moveDataBase(company118)
78          while 1:
79              Option = readOption()
80              if Option == '1':
81                  T2.inquiryPerson(company118)
82              elif Option == '2':
83                  T2.inquiryCar(company118)
84              elif Option == '3':
85                  T3.newCarRental(company118)
86              elif Option == '4':
87                  T4.carStatistics(company118)
88              elif Option == '5':
89                  break
90              else:
91                  print("Wrong Input!\n")
92      except Exception:
93          print("There Is At Least One Error!")
94
```

Figure 28

We defined object of company then we call the method readFile to read file then call the method print missing summary to printed it .after that we called print completed summary to print summary of recovered data then we called moveDataBase method from task one to print the data complete in complete data file and missing data in missing data file .

Then we entered infinite loop and call readOption method to read option and check what the user want and repeat it until user chose the choice exit.

Results

carRentalOld0

For the file carRentalOld0 the summery is in figure 29 below .

```
           Summary of data missing in the database:
Number of duplicate entries in the database = 0
Number of entries with wrong date format in the database = 0
Number of entries where names are dropped from the database = 0
Number of entries where Ids are dropped from the database = 0
Number of entries where dob are dropped from the database = 0
Number of entries where mobile numbers are dropped from the database = 0
Number of entries where personal entry can not be completed = 0
Number of entries where car make is dropped from the database = 0
Number of entries where car Ids are dropped from the database = 0
Number of entries where car models (year) are dropped from the database = 0

           Summary of data recovered from the database:
Number of duplicate entries removed from the new database = 0
Number of entries with wrong date format fixed in the new database = 0
Number of entries with names recovered in the new database = 0
Number of entries with Ids recovered in the new database = 0
Number of entries with dob recovered in the new database = 0
Number of entries with mobile numbers recovered in the new database = 0
Number of entries with car make recovered in the new database = 0
Number of entries with car models (year) recovered in the new database = 0
```

Figure 29

The result for inquiry using name for the file CarRentalOld0 is in figure 30, 31 in the next page.

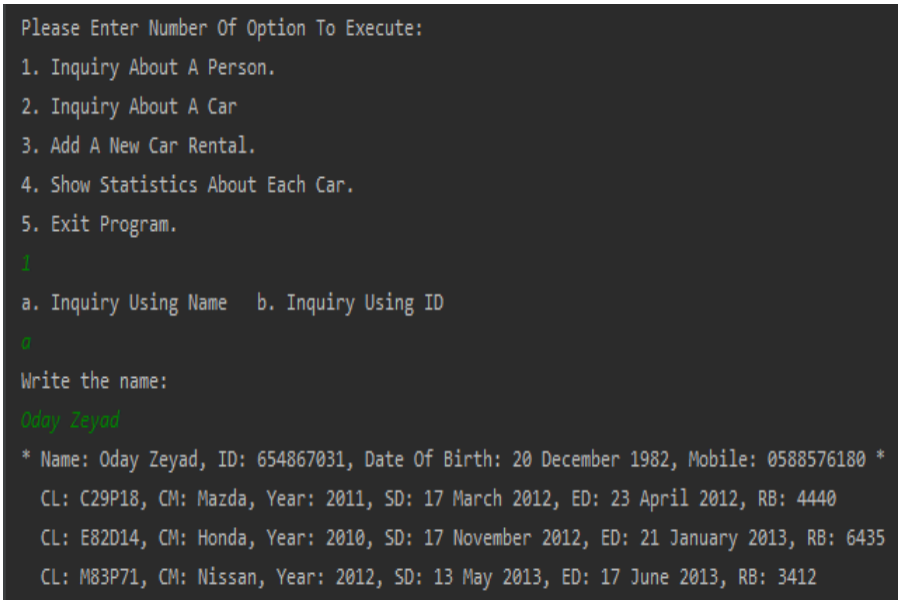


Figure 30

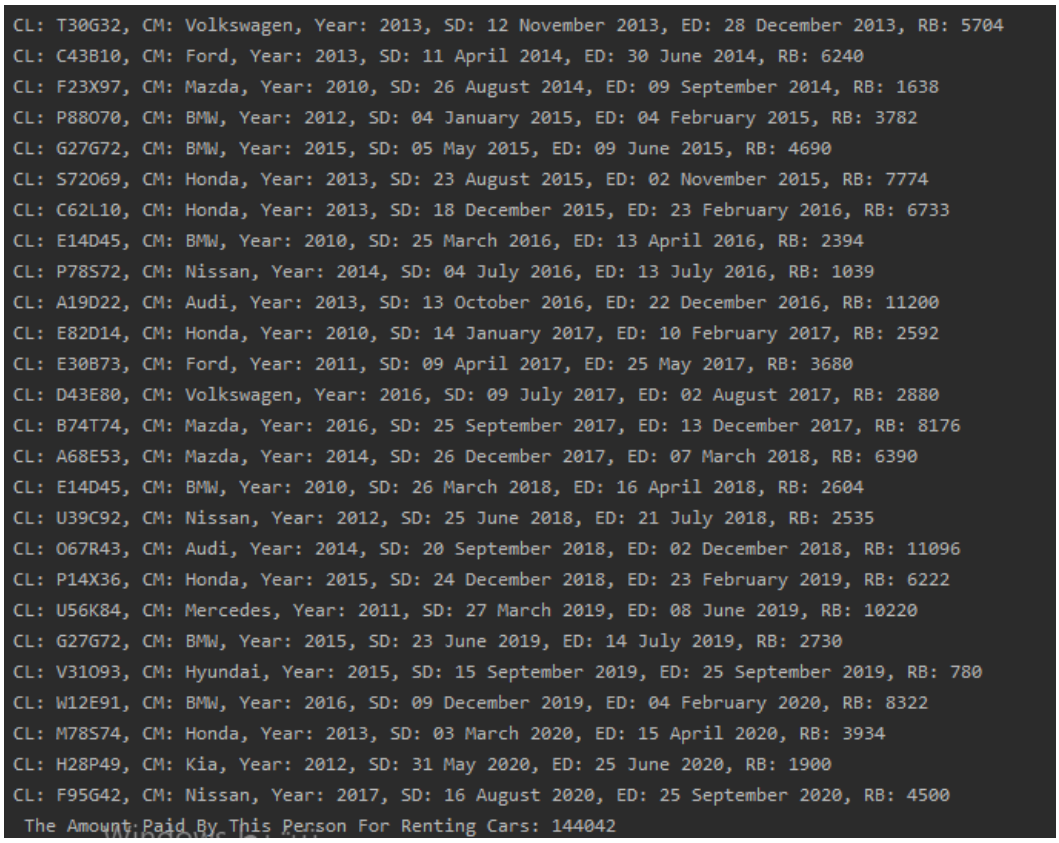


Figure 31

The result of inquiry using ID in figure 32, 33 in the next page .

```
Please Enter Number Of Option To Execute:
1. Inquiry About A Person.
2. Inquiry About A Car
3. Add A New Car Rental.
4. Show Statistics About Each Car.
5. Exit Program.
1
a. Inquiry Using Name   b. Inquiry Using ID
b
Write the id:
654867031
* Name: Oday Zeyad, ID: 654867031, Date Of Birth: 20 December 1982, Mobile: 0588576180 *
  CL: C29P18, CM: Mazda, Year: 2011, SD: 17 March 2012, ED: 23 April 2012, RB: 4440
  CL: E82D14, CM: Honda, Year: 2010, SD: 17 November 2012, ED: 21 January 2013, RB: 6435
  CL: M83P71, CM: Nissan, Year: 2012, SD: 13 May 2013, ED: 17 June 2013, RB: 3412
  CL: T30G32, CM: Volkswagen, Year: 2013, SD: 12 November 2013, ED: 28 December 2013, RB: 5704
  CL: C43B10, CM: Ford, Year: 2013, SD: 11 April 2014, ED: 30 June 2014, RB: 6240
  CL: F23X97, CM: Mazda, Year: 2010, SD: 26 August 2014, ED: 09 September 2014, RB: 1638
```

Figure 32

```
  CL: P88O70, CM: BMW, Year: 2012, SD: 04 January 2015, ED: 04 February 2015, RB: 3782
  CL: G27G72, CM: BMW, Year: 2015, SD: 05 May 2015, ED: 09 June 2015, RB: 4690
  CL: S72O69, CM: Honda, Year: 2013, SD: 23 August 2015, ED: 02 November 2015, RB: 7774
  CL: C62L10, CM: Honda, Year: 2013, SD: 18 December 2015, ED: 23 February 2016, RB: 6733
  CL: E14D45, CM: BMW, Year: 2010, SD: 25 March 2016, ED: 13 April 2016, RB: 2394
  CL: P78S72, CM: Nissan, Year: 2014, SD: 04 July 2016, ED: 13 July 2016, RB: 1039
  CL: A19D22, CM: Audi, Year: 2013, SD: 13 October 2016, ED: 22 December 2016, RB: 11200
  CL: E82D14, CM: Honda, Year: 2010, SD: 14 January 2017, ED: 10 February 2017, RB: 2592
  CL: E30B73, CM: Ford, Year: 2011, SD: 09 April 2017, ED: 25 May 2017, RB: 3680
  CL: D43E80, CM: Volkswagen, Year: 2016, SD: 09 July 2017, ED: 02 August 2017, RB: 2880
  CL: B74T74, CM: Mazda, Year: 2016, SD: 25 September 2017, ED: 13 December 2017, RB: 8176
  CL: A68E53, CM: Mazda, Year: 2014, SD: 26 December 2017, ED: 07 March 2018, RB: 6390
  CL: E14D45, CM: BMW, Year: 2010, SD: 26 March 2018, ED: 16 April 2018, RB: 2604
  CL: U39C92, CM: Nissan, Year: 2012, SD: 25 June 2018, ED: 21 July 2018, RB: 2535
  CL: O67R43, CM: Audi, Year: 2014, SD: 20 September 2018, ED: 02 December 2018, RB: 11096
  CL: P14X36, CM: Honda, Year: 2015, SD: 24 December 2018, ED: 23 February 2019, RB: 6222
  CL: U56K84, CM: Mercedes, Year: 2011, SD: 27 March 2019, ED: 08 June 2019, RB: 10220
  CL: G27G72, CM: BMW, Year: 2015, SD: 23 June 2019, ED: 14 July 2019, RB: 2730
  CL: V31O93, CM: Hyundai, Year: 2015, SD: 15 September 2019, ED: 25 September 2019, RB: 780
  CL: W12E91, CM: BMW, Year: 2016, SD: 09 December 2019, ED: 04 February 2020, RB: 8322
  CL: M78S74, CM: Honda, Year: 2013, SD: 03 March 2020, ED: 15 April 2020, RB: 3934
  CL: H28P49, CM: Kia, Year: 2012, SD: 31 May 2020, ED: 25 June 2020, RB: 1900
  CL: F95G42, CM: Nissan, Year: 2017, SD: 16 August 2020, ED: 25 September 2020, RB: 4500
   The Amount Paid By This Person For Renting Cars: 144042
Please Enter Number Of Option To Execute:
```

Figure 33

The inquiry using CL is in the figure 34,35 below .

Figure 34

```
Name: Sana Oday, Id: 563812608, Date Of Birth: 16 February 1996, Mobile: 0513348965, SD: 25 June 2018, ED: 12 July 2018, RB: 1555
Name: Awwad Ahmad, Id: 156531979, Date Of Birth: 19 June 1995, Mobile: 0560416234, SD: 13 July 2018, ED: 16 September 2018, RB: 7507
Name: Mohammad Khalaf, Id: 198809439, Date Of Birth: 21 September 1991, Mobile: 0552026946, SD: 17 September 2018, ED: 26 September 2018, RB: 1066
Name: Razan Alzeer, Id: 723836531, Date Of Birth: 5 June 1991, Mobile: 0580773329, SD: 27 September 2018, ED: 01 October 2018, RB: 468
Name: Salam Awwad, Id: 642020136, Date Of Birth: 2 May 1983, Mobile: 0522564842, SD: 03 October 2018, ED: 18 December 2018, RB: 8208
Name: Salma Ahmad, Id: 235693108, Date Of Birth: 15 April 1978, Mobile: 0574379865, SD: 19 December 2018, ED: 29 December 2018, RB: 1050
Name: Tariq Suleiman, Id: 982995830, Date Of Birth: 21 April 2002, Mobile: 0543200262, SD: 30 December 2018, ED: 14 February 2019, RB: 4968
Name: Suleiman Tariq, Id: 376439121, Date Of Birth: 15 June 1971, Mobile: 0527427585, SD: 15 February 2019, ED: 24 February 2019, RB: 999
Name: Salma Alzeer, Id: 975430181, Date Of Birth: 2 December 1976, Mobile: 0534845661, SD: 25 February 2019, ED: 09 May 2019, RB: 8541
Name: Oday Jaber, Id: 758351856, Date Of Birth: 5 November 1983, Mobile: 0520258356, SD: 10 May 2019, ED: 17 July 2019, RB: 6630
Name: Haneen Amer, Id: 487713364, Date Of Birth: 21 August 1986, Mobile: 0533634407, SD: 18 July 2019, ED: 24 July 2019, RB: 558
Name: Abdallah Mohammad, Id: 155136685, Date Of Birth: 28 December 1971, Mobile: 0571656244, SD: 26 July 2019, ED: 21 August 2019, RB: 2574
Name: Nassar Khalid, Id: 176576674, Date Of Birth: 21 April 1979, Mobile: 0547707719, SD: 22 August 2019, ED: 10 October 2019, RB: 5806
Name: Salam Amer, Id: 716314111, Date Of Birth: 9 August 1993, Mobile: 0583435628, SD: 11 October 2019, ED: 23 October 2019, RB: 1368
Name: Haneen Abdul_Kreem, Id: 683250081, Date Of Birth: 6 September 1982, Mobile: 0574017067, SD: 26 October 2019, ED: 10 November 2019, RB: 1732
Name: Salam Alzeer, Id: 688723006, Date Of Birth: 9 May 1985, Mobile: 0577446062, SD: 11 November 2019, ED: 22 November 2019, RB: 990
Name: Nassar Abdallah, Id: 661116484, Date Of Birth: 24 April 1974, Mobile: 0534566789, SD: 23 November 2019, ED: 31 December 2019, RB: 3420
Name: Mohammad Jaber, Id: 359205924, Date Of Birth: 4 March 1995, Mobile: 0585662568, SD: 02 January 2020, ED: 31 January 2020, RB: 2871
Name: Salem Khalaf, Id: 604242054, Date Of Birth: 29 February 1990, Mobile: 0564698661, SD: 02 February 2020, ED: 04 March 2020, RB: 3627
Name: Tariq Abdul_Kreem, Id: 836160020, Date Of Birth: 21 December 1986, Mobile: 0578114974, SD: 05 March 2020, ED: 08 April 2020, RB: 3570
Name: Abdallah Mohammad, Id: 155136685, Date Of Birth: 28 December 1971, Mobile: 0571656244, SD: 09 April 2020, ED: 20 April 2020, RB: 1287
Name: Tariq Oday, Id: 633254534, Date Of Birth: 25 June 1986, Mobile: 0532207666, SD: 22 April 2020, ED: 28 June 2020, RB: 7437
Name: Salam Amer, Id: 716314111, Date Of Birth: 9 August 1993, Mobile: 0583435628, SD: 29 June 2020, ED: 20 July 2020, RB: 2047
Name: Amer Abdullah, Id: 623593535, Date Of Birth: 6 September 1970, Mobile: 0518201205, SD: 22 July 2020, ED: 15 August 2020, RB: 2304
Name: Oday Zeyad, Id: 654867031, Date Of Birth: 20 December 1982, Mobile: 0588576180, SD: 16 August 2020, ED: 25 September 2020, RB: 4500
 The Revenue Made By Renting This Car: 137275
```

Figure 35

Figure 36,37 is print the available cars and input data for person to rented car

```
Please Enter Number Of Option To Execute:
1. Inquiry About A Person.
2. Inquiry About A Car
3. Add A New Car Rental.
4. Show Statistics About Each Car.
5. Exit Program.
3
Name: Abood
Id: 4056433
Date Of Birth: 12/12/2020
Mobile: 056895213
Car Rent Start Date: 25/9/2020
Car Rent End Date: 25/10/2020
*1* CL: Z92R21, CM: Audi, Year: 2011 *
*2* CL: O46E85, CM: Ford, Year: 2012 *
*3* CL: G56L50, CM: Nissan, Year: 2012 *
*4* CL: D27G81, CM: Mercedes, Year: 2012 *
*5* CL: S52Q75, CM: Mazda, Year: 2012 *
*6* CL: O48N10, CM: Honda, Year: 2010 *
*7* CL: E82D14, CM: Honda, Year: 2010 *
*8* CL: V23P96, CM: Volkswagen, Year: 2011 *
*9* CL: O15X13, CM: Volkswagen, Year: 2012 *
*10* CL: P88O70, CM: BMW, Year: 2012 *
*11* CL: M38V49, CM: Ford, Year: 2011 *
*12* CL: K81C96, CM: BMW, Year: 2010 *
*13* CL: F23X97, CM: Mazda, Year: 2010 *
*14* CL: U39C92, CM: Nissan, Year: 2012 *
```

Figure 36

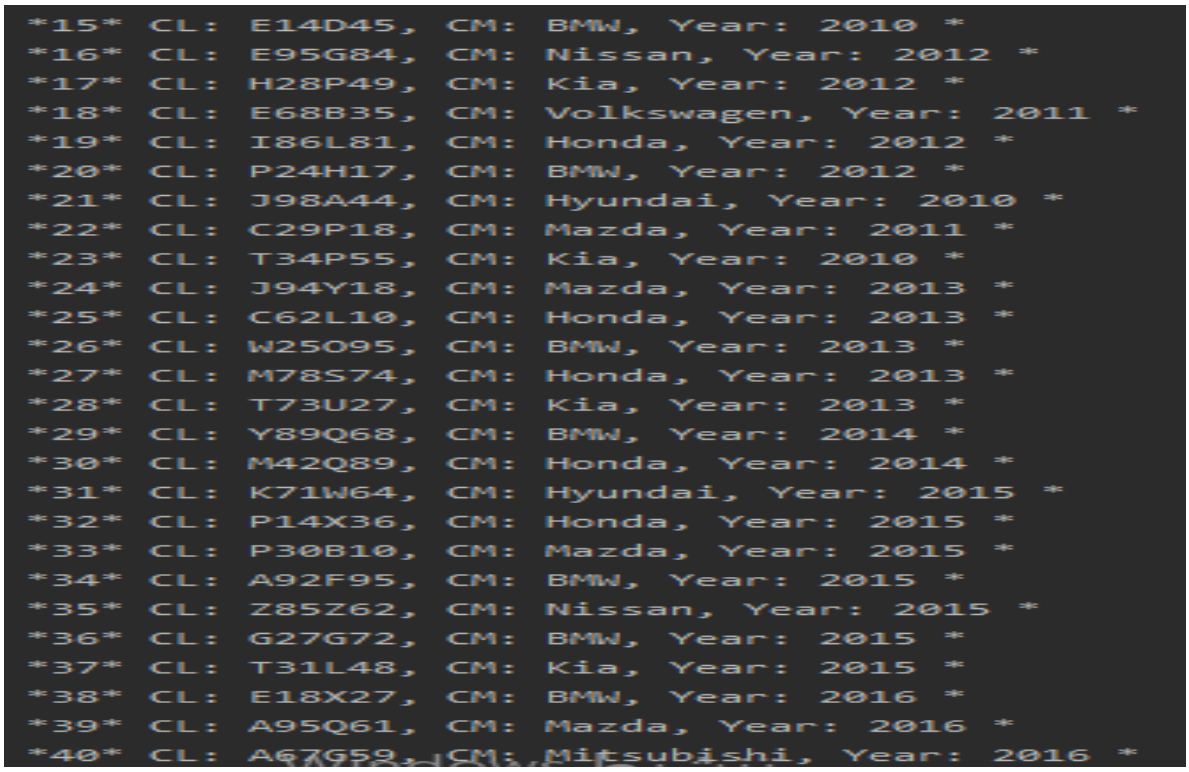


Figure 37

In figure 38 below show the statistics for each car.

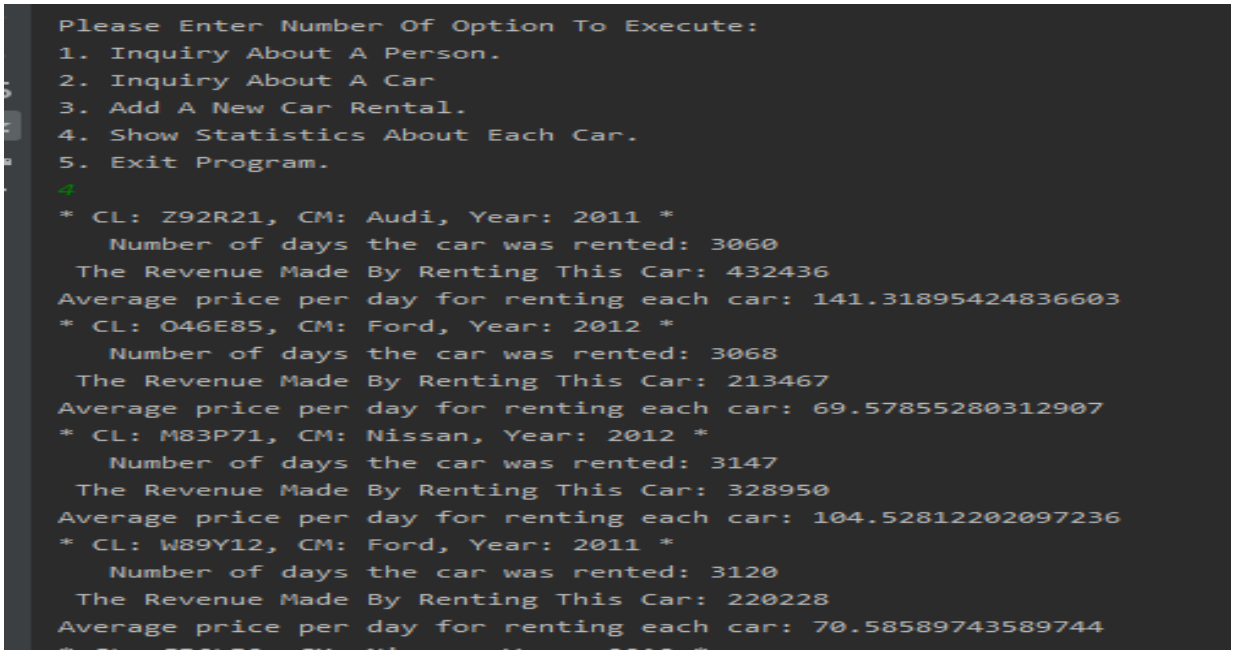


Figure 38

carRentalOld3

The summery of this the 3 file is in figure 39 below.

```
           Summary of data missing in the database:
Number of duplicate entries in the database = 0
Number of entries with wrong date format in the database = 0
Number of entries where names are dropped from the database = 100
Number of entries where Ids are dropped from the database = 0
Number of entries where dob are dropped from the database = 0
Number of entries where mobile numbers are dropped from the database = 0
Number of entries where personal entry can not be completed = 0
Number of entries where car make is dropped from the database = 0
Number of entries where car Ids are dropped from the database = 0
Number of entries where car models (year) are dropped from the database = 0

           Summary of data recovered from the database:
Number of duplicate entries removed from the new database = 0
Number of entries with wrong date format fixed in the new database = 0
Number of entries with names recovered in the new database = 100
Number of entries with Ids recovered in the new database = 0
Number of entries with dob recovered in the new database = 0
Number of entries with mobile numbers recovered in the new database = 0
Number of entries with car make recovered in the new database = 0
Number of entries with car models (year) recovered in the new database = 0
```

Figure 39

carRentalOld4

The summery of this the 4 file is in figure 40 below.

```
           Summary of data missing in the database:
Number of duplicate entries in the database = 0
Number of entries with wrong date format in the database = 0
Number of entries where names are dropped from the database = 0
Number of entries where Ids are dropped from the database = 100
Number of entries where dob are dropped from the database = 0
Number of entries where mobile numbers are dropped from the database = 0
Number of entries where personal entry can not be completed = 0
Number of entries where car make is dropped from the database = 0
Number of entries where car Ids are dropped from the database = 0
Number of entries where car models (year) are dropped from the database = 0

           Summary of data recovered from the database:
Number of duplicate entries removed from the new database = 0
Number of entries with wrong date format fixed in the new database = 0
Number of entries with names recovered in the new database = 0
Number of entries with Ids recovered in the new database = 100
Number of entries with dob recovered in the new database = 0
Number of entries with mobile numbers recovered in the new database = 0
Number of entries with car make recovered in the new database = 0
Number of entries with car models (year) recovered in the new database = 0
```

Figure 40

carRentalOld13

The summery of this the 13 file is in figure 41 below.

```
           Summary of data missing in the database:
Number of duplicate entries in the database = 30
Number of entries with wrong date format in the database = 29
Number of entries where names are dropped from the database = 29
Number of entries where Ids are dropped from the database = 30
Number of entries where dob are dropped from the database = 30
Number of entries where mobile numbers are dropped from the database = 60
Number of entries where personal entry can not be completed = 30
Number of entries where car make is dropped from the database = 30
Number of entries where car Ids are dropped from the database = 31
Number of entries where car models (year) are dropped from the database = 30

           Summary of data recovered from the database:
Number of duplicate entries removed from the new database = 30
Number of entries with wrong date format fixed in the new database = 29
Number of entries with names recovered in the new database = 29
Number of entries with Ids recovered in the new database = 30
Number of entries with dob recovered in the new database = 30
Number of entries with mobile numbers recovered in the new database = 60
Number of entries with car make recovered in the new database = 30
Number of entries with car models (year) recovered in the new database = 30
```

Figure 41