FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING

Linux Lab - ENCS313

Python Project

Prepared by: Rawan Yassin 1182224 Alaa Zuhd 1180865
Instructor: Dr. Aziz Qaroush
Assistant: Eng. Aseel Awwad

Section: 2

BIRZEIT
August 26, 2020

1 Abstract

The aim of this project is to be more familiar with python programming by building a python program for a car rental company which do multiple tasks including the modification of dates format , trying to complete the missing information in the database ,and removing the duplicate entries in the database, also it allows the function of inquiring a person from the database using it's name or id , and inquiring a car using it's license number , also it allows the function of adding information for a new car rental into the completed database , and finally it's support the task of printing statistics about each car in the completed database.

Contents

2 Code Illustration and Comments

In this section, we will be considering our code in details, every case we have tried to study will be mentioned, and all the concepts implemented will be explained.

2.1 Main file

In the main of our program, we have imported the CarRental class which can be found in appendix A. The CarRental class has mainly the ten data fields for each entry as properties, and has a method that called printInfo, that returns the data. In our main, we have started by reading the CarRentalOld file and then we started processing the read data, by looping through the lines, in which each line represents an entry. If the line is not empty (the obj returned from searching for "[A-Za-z0-9-/;]" is not null), we will be splitting the line according to the semicolon, that is used to separate the data fields and creating an instance of the CarRental Class and add it to the list that is called list.
Then the function that is called "CompleteMissing" is called, and it will be illustrated in the next page, but calling it is necessary as we will be using out list for other tasks. The setOfCars is a set of the cars id, created by looping through the list after it has been processed by the CompleteMissing function, and its used will be illustrated later, and carsDict is a dictionary with car id as a key, and car make (cm), and year of make (year) as values. We have decided to make our main flexible by creating a loop that asks the user to choose between the four available tasks, and after finishing each task, he/she will be able to choose whether to try another task or terminate, and absolutely, if a wrong task number is entered, a message stating this will appear and the user can also choose between terminating or trying again.

2.2 Task1

This task has mainly six different functions: - code found in appendix D

1. modifyDate function, which takes a CarRental object as an argument and starts modifying the dates in that specific entry. There are three dates in each entry, the date of birth of the user, the start and end rental dates, and these dates, can be in the following three different formats:

   (a) Day Month(written in words) Year.

   (b) Day-Month(written as a digit)-Year.

   (c) Day/Month(written as a digit)/Year.

   If the date is in the first format, no modification is needed, if it is in the second or third, it will be modified to be like the first using this function. It uses the following concept: search using regular expression for a - or / followed by a number followed by a - or / again, if the search function returns a match, the old date will be replaced by a new date that has the month found from using the old date month as an index for the monthName list, and using this element of the monthName list (indexed using month in the old date format) to replace the old month, and replacing the - or / by spaces. (while keeping the day and year of each month, as they are not searched for and no modification is needed for them).
   This function returns either a 1 or 0, if any date in the entry is modified then the flag is set to 1 and is returned, if no date is modified, the function will return 0.

2. completeMissing function, which uses the car id (cl) as a primary key for completing empty data fields, and determining the validity of the entry. it starts by looping through the list " which is all the entries read from the CarRentalOld file". and if cl is not missing, we will be processing this entry, if any field is empty, we will be looping through the same list using j to search for the same cl, and if the same cl is found in an entry, then if the year, or cm field are missing then they can be filled as they are the same for different entries with same cl. If the start date is missing, and we are sure we are now comparing two entries with same cl, so if end date is found and the same for two entries, then the missing start date is absolutely the same for the two entries (the same car can not be rented for more than one person in the same time). Now, to fill the amount paid, we will be considering the two entries with same cl, but before recovering the missing rb, we will be checking that the start or end date "one of them at least" are the same for the two entries. After having all of the previous fields filled, we can start filling the personal fields. We start by filling the id of the person after checking the similarity of cl,sd,and ed, as if they are the same, these two entries are duplicates and can be used to complete each other. Then the name, date of birth, and mobile, can be completed,

using two entries with same car id and personal id. This is previously the implemented concept used in this function.

Note : tracing the comments in the code, helps tracing our logic.

3. printSummary function, prints the required summary.

4. calcNumOfDuplicates function, in this function we have encountered how many duplicates are there in the whole database , by the helping of flagList which will indicate if the entry have been encountered or not "to not encounter the same entry more than once" , so this function will loop through the whole list and check if this entry is marked as a duplicate of anther entry or if the car id is missing or the start and end dates are missing in that entry , if so then we can't know if there is a duplicate of this entry , so it will ignore this entry and continue in looping , else if the current entry is not marked as a duplicate of anther entry and the car id is exist and at least one of the dates "start or end date" is exist, then we can look into the duplicates of the current entry , so we enter a loop starting from the next entry of the current entry and check if this entry has been encounter in anther duplicates or not , and if the cars id are identical , and start or end date are identical , if so then it will be a duplicate and it will mark the entry obtained from the last loop as encountered "flagList[j]=1" , and it will increase the number of duplicate by 1 . When the loop ending then the sum variable will contains the number of duplicate and return it .

5. removeDuplicate function, if two entries have exactly the same data after they have been completed then they are duplicates and one can be removed.

6. compareDates function, in this function we were trying to solve the idea that maybe date1 = 1 May 2020 and date2 = 01 May 2020, then comparing them as string will yield to be not identical , so using date objects will solve the problem, and it is used in completeMissing function, when comparing start or end dates or birth dates for two entries.

2.3 Task2

In this task we aimed to get the inquiry code done, the inquiry can be done either using a personal information(name or id), or using a car's information. This task file has mainly three functions, the inquiryPerson function, which uses the help of searchId function, and an inquiryCar function.

The inquiryPerson function, asks the user to enter the name to search for, converts it to lower case, and when searching names are compared after converting them to lower case (so our search is case insensitive). And if the user enters a name with characters other than letters or spaces or underscores or dashes, the name will be considered invalid. If the entered name has passed this test, the searching process for this name would start, but as the name is not unique, a "listIds" list is used as an auxiliary list. If the entered name is matched with the name of any entry, the id of this name is added to the listIds list, but before adding it to the list (if for example, this name has rented a car more than once) this id is checked, if it does not exist in the list (listIds.count(list[i].getId()) == 0), the id will be added to the list. After that, the listIds elements are passed one by one to the searchId function, that searches for a match in ids to print the information. The searchId function, loops through the processed list(with duplicates removed and missing data recovered) to search for a match in ids, and has total that adds the amount paid for each different entry.

Note: if the user chooses to enter the id, it is checked that it is made of numbers only, otherwise it is considered invalid. The inquiryCar does the search after checking that the entered cl is made of digits and alphabetical letters only. The same approach for searching is used, and then when a match with cl is found in the list, the person's information are printed.

2.4 Task3

In this task we have implemented a simple system to let the customer to rent a new car , and this task has mainly three functions ,readDates function , validCars function , and addInfoToList function
readDates function asks the user to enter the start date and the end date to rent a car at these dates , and with the help of modifyDate function it checks that the dates are valid and logical, which means month are from 1 to 12 and the day is in the range of each month , also it checks if the dates are illogical like negative dates and so on , then it checks if the end date is more than the start date , and finally checks if the start date is at least equal to today's date . if any of the dates is not consistent with the above conditions it will print an error message and ask the user to try again or finish the task . else if there is no error then it can call validCars function

validCars function , this function will check for all valid cars available during the the entered dates , by using a set contains the valid cars in the completed database "setOfValidCars" , and then looping through the list of the completed database and check if the car is exist in the setOfValidCars , if so then it checks if there is no intersection between the start date and end date of that car and with entered dates , if so then it will do nothing , else it will remove that car from the setOfValidCars because it will be considered as not valid during these dates. and finally when this loop end the setOfValidCars will contains all valid cars to rent them . and it will print the basic information "cl,cm,and year" of all valid cars to the user , then it will call addInfoToList function .

addInfoToList function , this function will ask the user to enter the id of the valid cars printed from the last function , and since the car id (cl) contains letters so it will convert the entered cl into upper case to make the comparison case insensitive , then it will loops through the setOfValidCars to check if the entered cl is valid or not , if it's valid then it will ask the user to enter his/her personal information like "name,id,mobile number,DOB" and to enter the rental money needed to be paid "rb" , and for every entered input it will check if they are valid and logical , like the name should contains letters and some special characters "space,underscore,and dash" only, the id,mobile number,and rb should contains digits only , and it will use isDOBValid function to check the validity of the DOB like if the date is valid or not , and if the user is under 16 or not , if any thing is not consistent with our conditions then an error message will appear and ask the user to either try again or finish the task , after all of that if the whole entered information are valid then this rental will be added to the completed database , and task 3 will be finished .

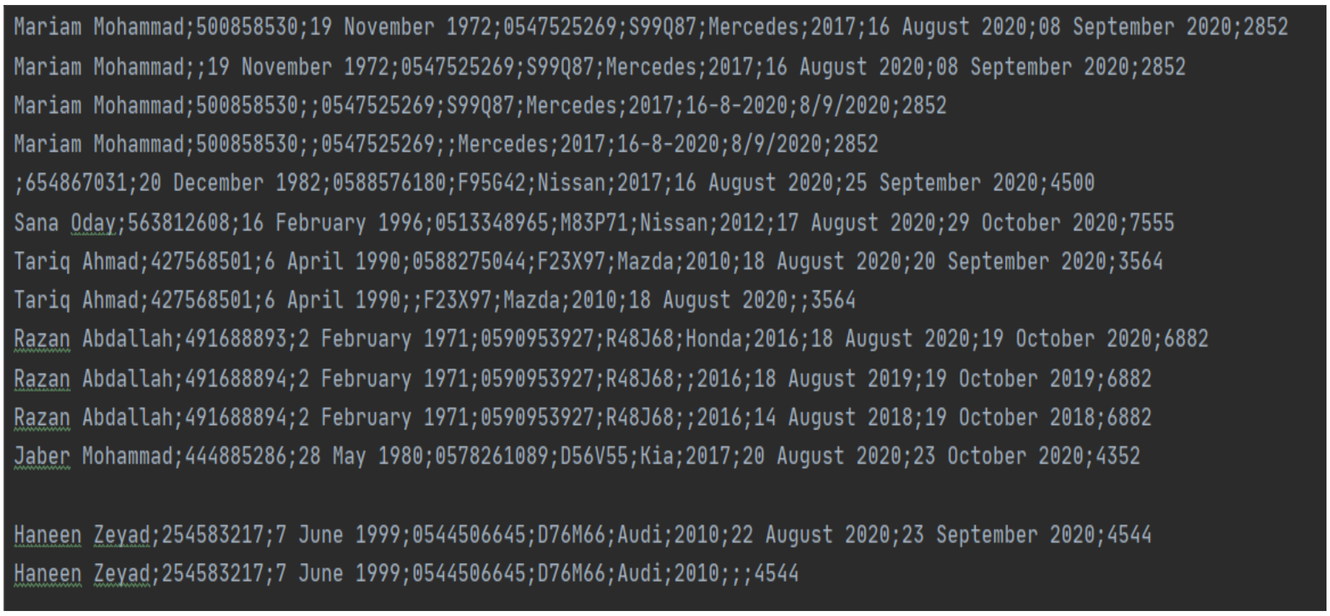
2.5 Task4

In this task, we were supposed to print the statistics about the cars we have read from file. As we have noticed in the main, we have created a setOfCars, which has the the cars id (cl) as its inputs. We have decided to use the set data structure, to make sure that each cl is entered once. And now to be able to calculated the required output, we have imported the datetime module, which will ease our work. We will be looping through the setOfCars elements, and for each element we will be calling the calcNumOfRentedAndRevenueDays function. This function, searched for a match of cl in the whole processed list (with duplicates removed and missing data recovered), and for each match, the start and end date are converted to a specific format using datetime, this format allowed us to subtract the end date from the start date and get the number of days this car was rented. And then the total paid and average can be calculated accordingly.

3 STUDY CASE 1

3.1 CarRentalOld.txt

In the following few pages, we will be considering this text file to read from.



```
Mariam Mohammad;500858530;19 November 1972;0547525269;S99Q87;Mercedes;2017;16 August 2020;08 September 2020;2852
Mariam Mohammad;;19 November 1972;0547525269;S99Q87;Mercedes;2017;16 August 2020;08 September 2020;2852
Mariam Mohammad;500858530;;0547525269;S99Q87;Mercedes;2017;16-8-2020;8/9/2020;2852
Mariam Mohammad;500858530;;0547525269;;Mercedes;2017;16-8-2020;8/9/2020;2852
;654867031;20 December 1982;0588576180;F95G42;Nissan;2017;16 August 2020;25 September 2020;4500
Sana Oday;563812608;16 February 1996;0513348965;M83P71;Nissan;2012;17 August 2020;29 October 2020;7555
Tariq Ahmad;427568501;6 April 1990;0588275044;F23X97;Mazda;2010;18 August 2020;20 September 2020;3564
Tariq Ahmad;427568501;6 April 1990;;F23X97;Mazda;2010;18 August 2020;;3564
Razan Abdallah;491688893;2 February 1971;0590953927;R48J68;Honda;2016;18 August 2020;19 October 2020;6882
Razan Abdallah;491688894;2 February 1971;0590953927;R48J68;;2016;18 August 2019;19 October 2019;6882
Razan Abdallah;491688894;2 February 1971;0590953927;R48J68;;2016;14 August 2018;19 October 2018;6882
Jaber Mohammad;444885286;28 May 1980;0578261089;D56V55;Kia;2017;20 August 2020;23 October 2020;4352

Haneen Zeyad;254583217;7 June 1999;0544506645;D76M66;Audi;2010;22 August 2020;23 September 2020;4544
Haneen Zeyad;254583217;7 June 1999;0544506645;D76M66;Audi;2010;;;4544
```

Figure 1: CarRentalOld.txt

Few Notes about this file:

(a) Mariam Mohammad is repeated 4 times, but the fourth time will not be considered since the car id (cl) is missing.

(b) The fifth entry has a missing name, and can not be filled because it does not have any field in common with other entries.

(c) Tariq Ahmad will be missing the mobile number and end date on the eighth entry but can be filled, as it is a duplicate of the seventh entry.

(d) Razan Abdallah with ID 491688893 does not have duplicates.

(e) Razan Abdallah with ID 491688894 does has duplicates.

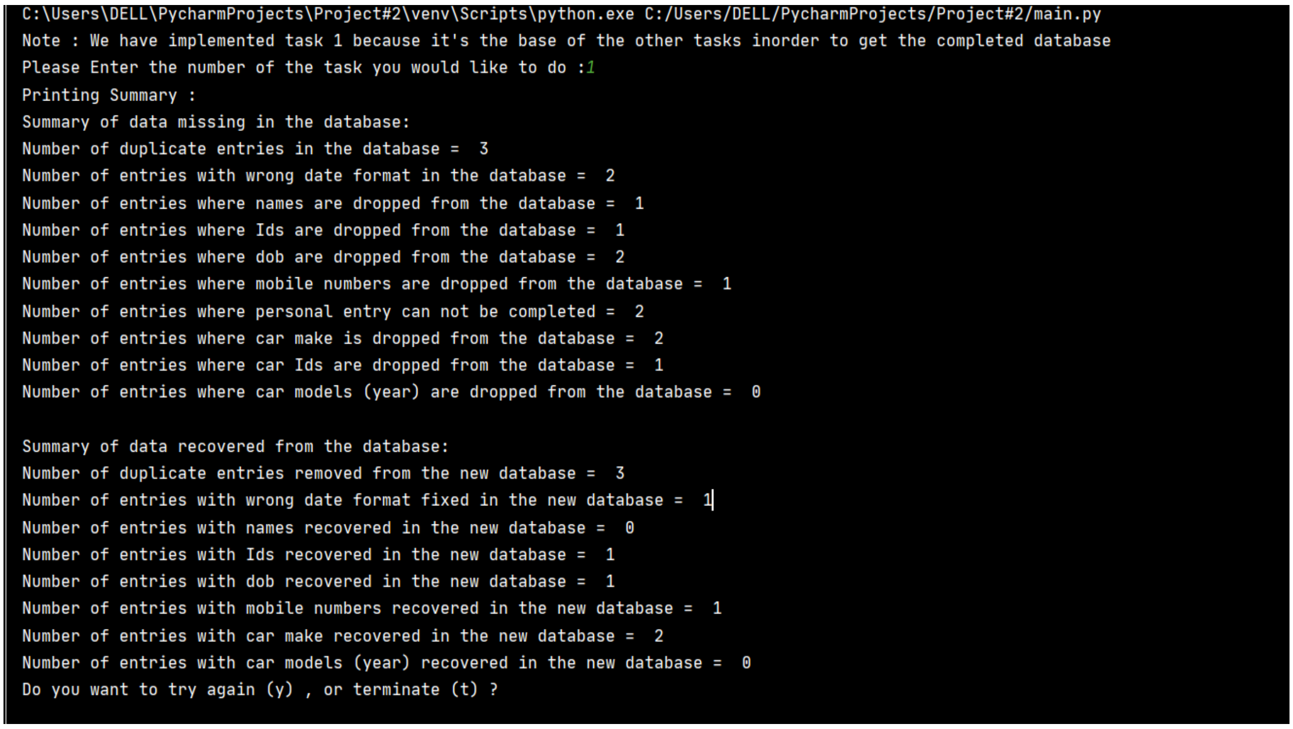
(f) An empty line is added to the file to see how our programs handles this case.

(g) In Haneen's second entry, the start and end dates are missing, and so this case will be discarded.

(h) More details will be given when studying the tasks separately.

3.2 Discussion

3.2.1 Task1



```
C:\Users\DELL\PycharmProjects\Project#2\venv\Scripts\python.exe C:/Users/DELL/PycharmProjects/Project#2/main.py
Note : We have implemented task 1 because it's the base of the other tasks inorder to get the completed database
Please Enter the number of the task you would like to do :1
Printing Summary :
Summary of data missing in the database:
Number of duplicate entries in the database =  3
Number of entries with wrong date format in the database =  2
Number of entries where names are dropped from the database =  1
Number of entries where Ids are dropped from the database =  1
Number of entries where dob are dropped from the database =  2
Number of entries where mobile numbers are dropped from the database =  1
Number of entries where personal entry can not be completed =  2
Number of entries where car make is dropped from the database =  2
Number of entries where car Ids are dropped from the database =  1
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 =  3
Number of entries with wrong date format fixed in the new database =  1
Number of entries with names recovered in the new database =  0
Number of entries with Ids recovered in the new database =  1
Number of entries with dob recovered in the new database =  1
Number of entries with mobile numbers recovered in the new database =  1
Number of entries with car make recovered in the new database =  2
Number of entries with car models (year) recovered in the new database =  0
Do you want to try again (y) , or terminate (t) ?
```

Figure 2: Task1 Output

Task 1 as mentioned above, does the processing for the data obtained from the file, it edits the date format, recovers the entries that are missing some fields, and finally remove duplicates.And to view the output of this task, this summary is printed on screen, when the user chooses to do task 1. Considering this output in more details:

(a) We have 3 duplicates, 2 of Mariam (not counting the fourth entry since it does not have a car id ,which we have considered our primary key to check validity of entry, and yes Tariq has a duplicate as well.

(b) We have two dates in wrong format in entries 3 and 4.

(c) We have one name dropped from the fifth entry.

(d) We have a personal id missing from Mariam's second entry but can be filled easily from the first entry, since it is a duplicate.

(e) We have date of birth dropped twice in Mariams entries.

(f) We have a mobile dropped from Tariq second's entry but that could be recovered from his first entry.(since it is a duplicate)

(g) We have two personal entries that can not be completed, the fourth enrty, which is Mariam's, as it is missing the car id. And the fifth entry,as name is missing but all other completed information do not match any data in the file.

(h) Razan Abdallah (id= 491688894) is missing the car make in its two entires, but this case is handled as the car id (cl) is known.

(i) One entry is missing the cl (Mariam's fourth entry).

(j) No entries are missing the year.

After doing the processing for data, we notice that:

(a) All completed duplicates are removed.

(b) Wrong data format is fixed in one entry, since the second is dropped from the completed entries, and transferred to the missing.

(c) The missing name in the fifth entry could not be recovered as this entry is not a duplicate nor it has any common information with other entries.

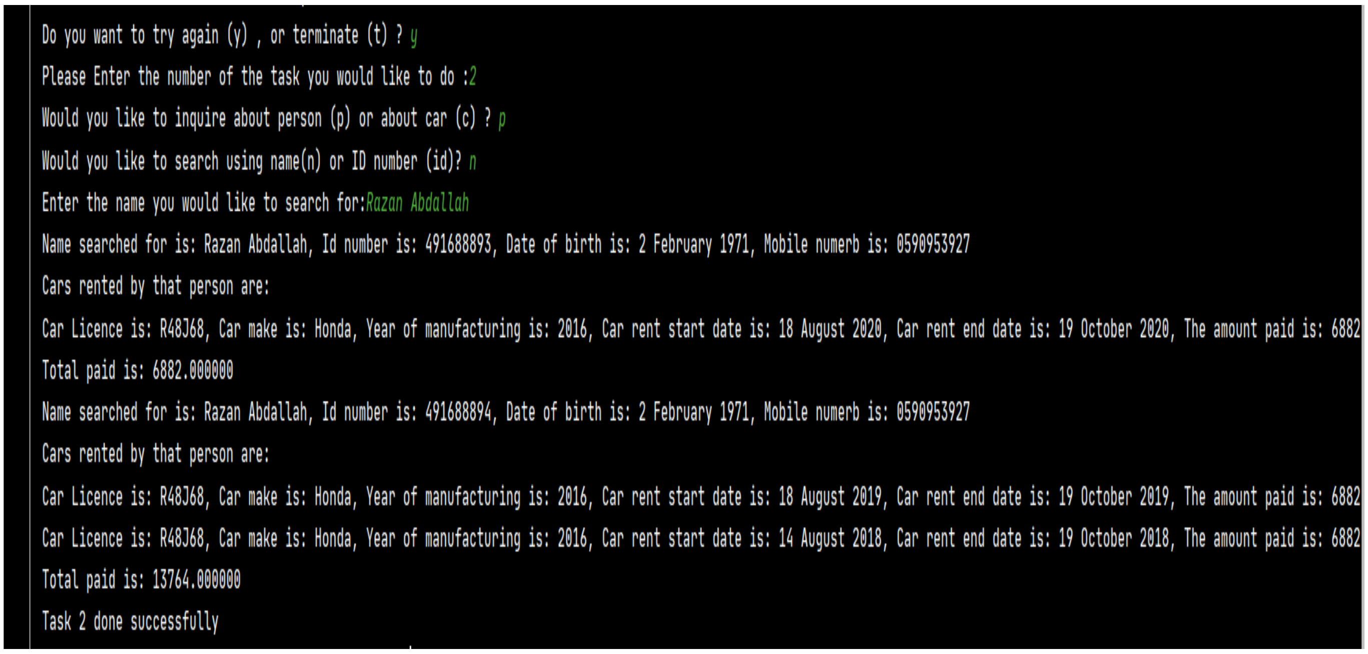
(d) The missing id in Mariam's fourth entry could be recovered easily.

(e) The date of birth of the fifth entry could not be completed, same reason as the name

(f) The mobile numbers and car make could be recovered.

3.2.2 Task2

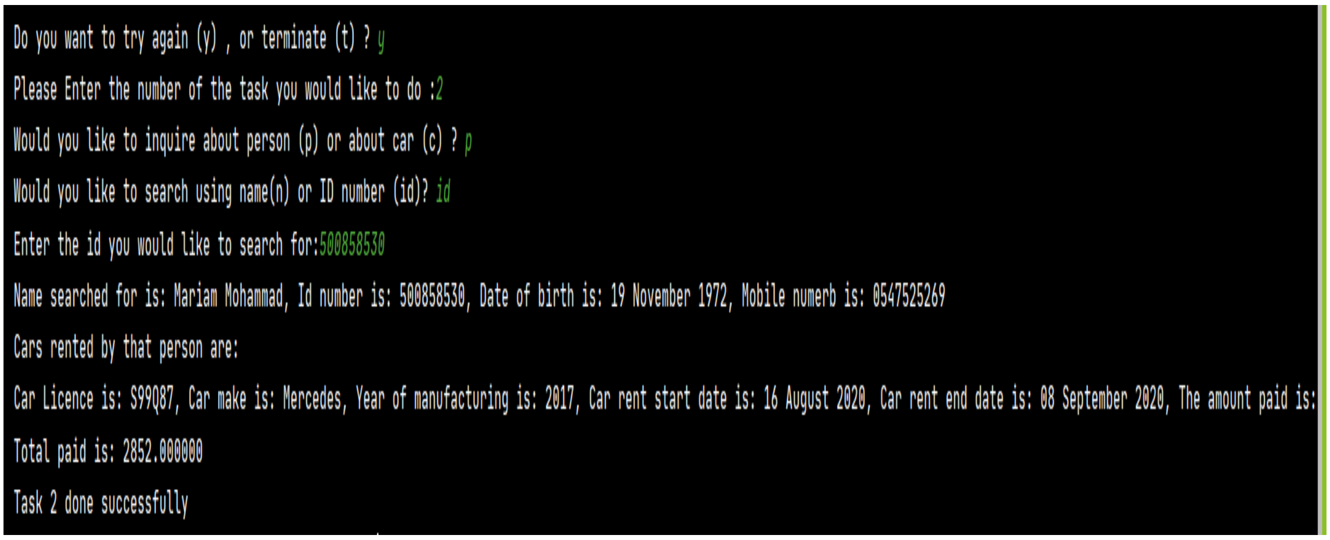
Now moving to task 2, after the program finishes task 1, it asks the user, whether he/she wants to try another task or terminate the program, we have chosen task 2 which does the inquiry, and then a message asking whether the inquiry will be using personal information or cars information appears, and the progrma works accordingly. In the first snapshot of this task, we have decided to the inquiry, about a person, using the name, but as name is not unique, the name searched for will be presented with the different ids, as follows:
Note: For Razan with id 491688893, there is only one entry, so the total paid equals the amount paid in that entry.
For Razan with id 491688894, there are two different rental dates, so the total paid equals the sum of the amount paid in each entry.



```
Do you want to try again (y) , or terminate (t) ? y
Please Enter the number of the task you would like to do :2
Would you like to inquire about person (p) or about car (c) ? p
Would you like to search using name(n) or ID number (id)? n
Enter the name you would like to search for:Razan Abdallah
Name searched for is: Razan Abdallah, Id number is: 491688893, Date of birth is: 2 February 1971, Mobile numerb is: 0590953927
Cars rented by that person are:
Car Licence is: R48J68, Car make is: Honda, Year of manufacturing is: 2016, Car rent start date is: 18 August 2020, Car rent end date is: 19 October 2020, The amount paid is: 6882
Total paid is: 6882.000000
Name searched for is: Razan Abdallah, Id number is: 491688894, Date of birth is: 2 February 1971, Mobile numerb is: 0590953927
Cars rented by that person are:
Car Licence is: R48J68, Car make is: Honda, Year of manufacturing is: 2016, Car rent start date is: 18 August 2019, Car rent end date is: 19 October 2019, The amount paid is: 6882
Car Licence is: R48J68, Car make is: Honda, Year of manufacturing is: 2016, Car rent start date is: 14 August 2018, Car rent end date is: 19 October 2018, The amount paid is: 6882
Total paid is: 13764.000000
Task 2 done successfully
```

Figure 3: Task2 Output

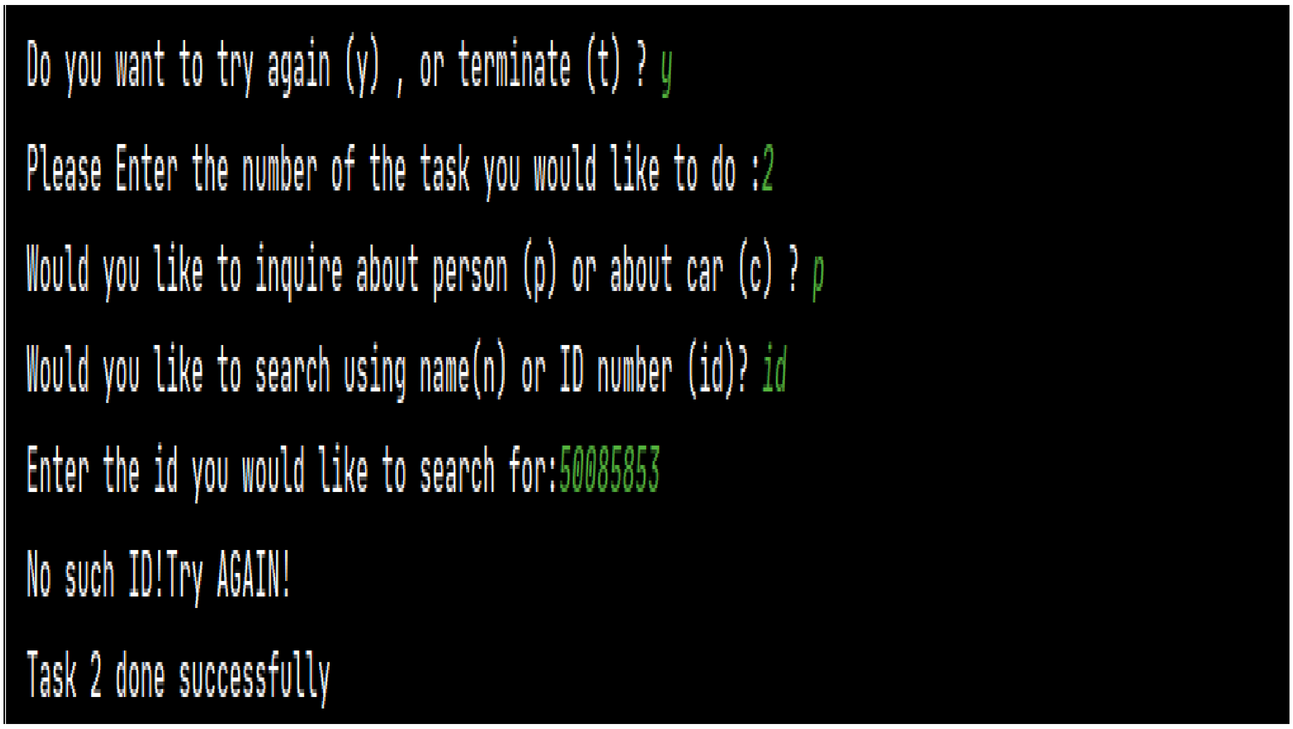
In the following case of task 2, we have decided to do the search using id number of a person



```
Do you want to try again (y) , or terminate (t) ? y
Please Enter the number of the task you would like to do :2
Would you like to inquire about person (p) or about car (c) ? p
Would you like to search using name(n) or ID number (id)? id
Enter the id you would like to search for:500858530
Name searched for is: Mariam Mohammad, Id number is: 500858530, Date of birth is: 19 November 1972, Mobile numerb is: 0547525269
Cars rented by that person are:
Car Licence is: S99Q87, Car make is: Mercedes, Year of manufacturing is: 2017, Car rent start date is: 16 August 2020, Car rent end date is: 08 September 2020, The amount paid is:
Total paid is: 2852.000000
Task 2 done successfully
```

Figure 4: Task2 Output

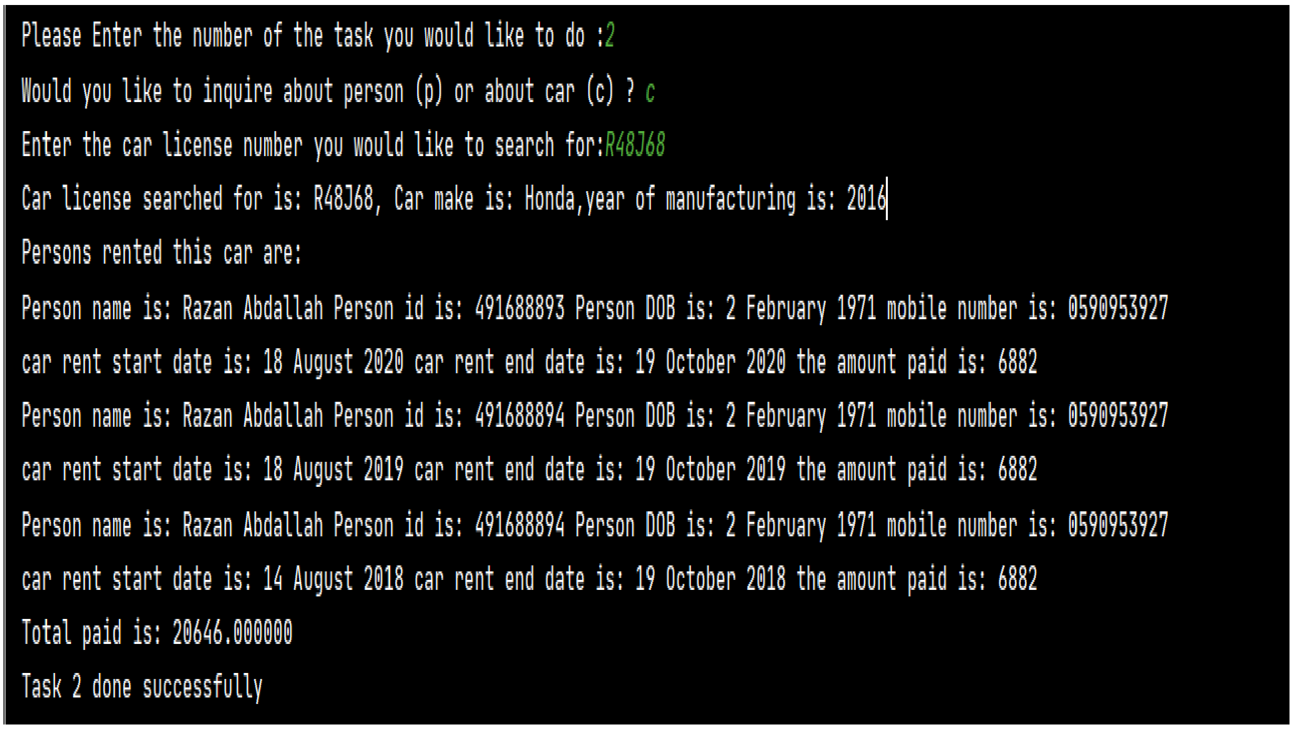
In this snapshot, we wanted to prove the effectiveness of our program, if an id with no entries is searched for, a message stating that no such id is found is presented.



```
Do you want to try again (y) , or terminate (t) ? y
Please Enter the number of the task you would like to do :2
Would you like to inquire about person (p) or about car (c) ? p
Would you like to search using name(n) or ID number (id)? id
Enter the id you would like to search for:50085853
No such ID!Try AGAIN!
Task 2 done successfully
```

Figure 5: Task2 Output

In this last snapshot of task 2, we have done the inquiry using cl as follows:

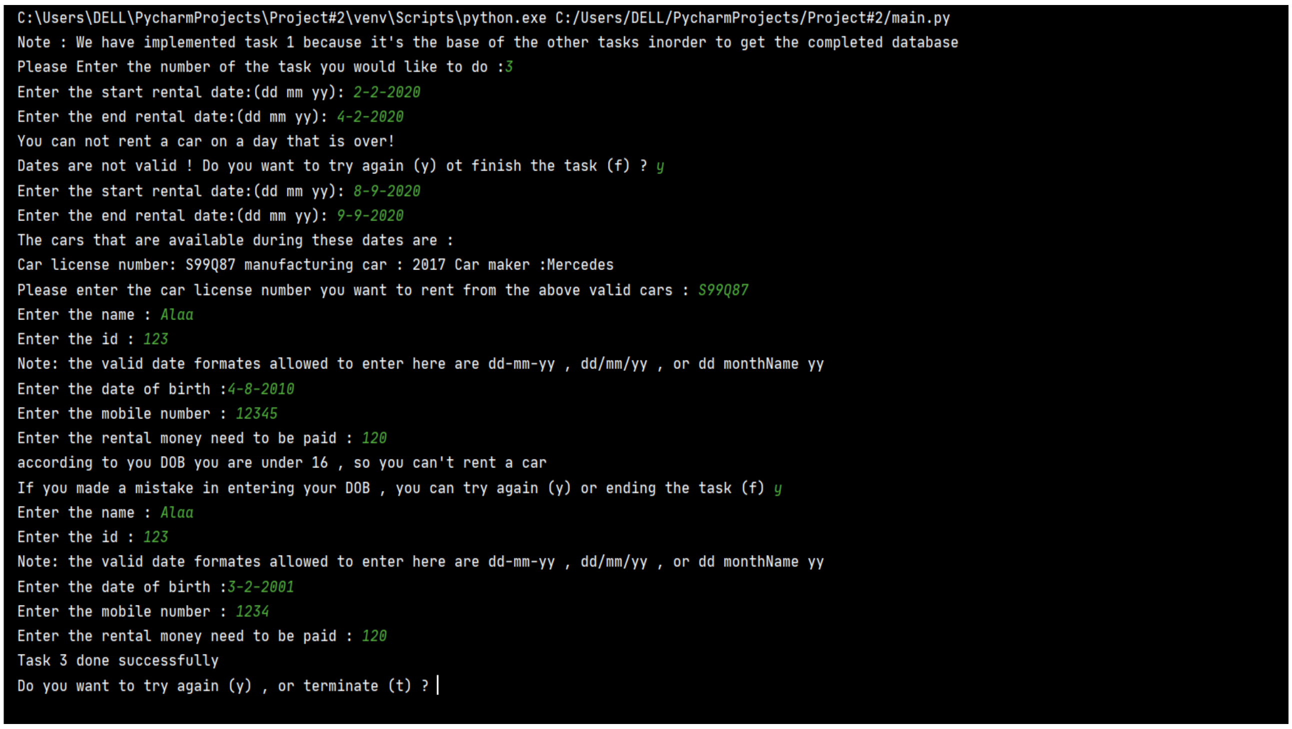


```
Please Enter the number of the task you would like to do :2
Would you like to inquire about person (p) or about car (c) ? c
Enter the car license number you would like to search for:R48J68
Car license searched for is: R48J68, Car make is: Honda,year of manufacturing is: 2016
Persons rented this car are:
Person name is: Razan Abdallah Person id is: 491688893 Person DOB is: 2 February 1971 mobile number is: 0590953927
car rent start date is: 18 August 2020 car rent end date is: 19 October 2020 the amount paid is: 6882
Person name is: Razan Abdallah Person id is: 491688894 Person DOB is: 2 February 1971 mobile number is: 0590953927
car rent start date is: 18 August 2019 car rent end date is: 19 October 2019 the amount paid is: 6882
Person name is: Razan Abdallah Person id is: 491688894 Person DOB is: 2 February 1971 mobile number is: 0590953927
car rent start date is: 14 August 2018 car rent end date is: 19 October 2018 the amount paid is: 6882
Total paid is: 20646.000000
Task 2 done successfully
```

Figure 6: Task2 Output

3.2.3 Task3

Now we have chosen to perform task 3, which asks the user mainly to enter two dates, to see all available cars during these dates, and then to allow the user to choose a car, and then filling the remaining information.
Our robust program, checks that the enter dates are valid and logical (as mentioned above in the illustration of task 3), and in the next snapshot, we have entered date that has already passed (comparing to today's date), this would be an illogical renting case. so the program will ask the user whether he/she wants to enter another dates, or finish this task. After entering logical dates, all available cars are presented, and then the user enters the cl he/she prefers, then, when filling all other information, many cases are checked and considered, for the following snapshot, we have entered a birth date for a 10 years old, and so the rent can not be done, so message stating that those under 16 can not rent cars "yet" is presented, while giving the user the chance to try filling the information again.

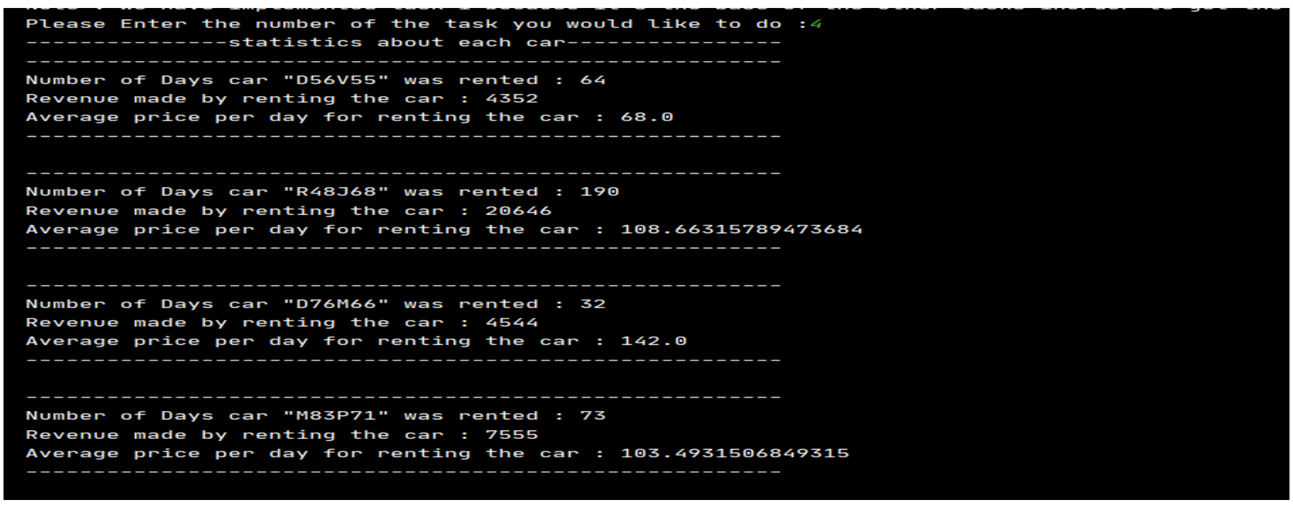


```
C:\Users\DELL\PycharmProjects\Project#2\venv\Scripts\python.exe C:/Users/DELL/PycharmProjects/Project#2/main.py
Note : We have implemented task 1 because it's the base of the other tasks inorder to get the completed database
Please Enter the number of the task you would like to do :3
Enter the start rental date:(dd mm yy): 2-2-2020
Enter the end rental date:(dd mm yy): 4-2-2020
You can not rent a car on a day that is over!
Dates are not valid ! Do you want to try again (y) ot finish the task (f) ? y
Enter the start rental date:(dd mm yy): 8-9-2020
Enter the end rental date:(dd mm yy): 9-9-2020
The cars that are available during these dates are :
Car license number: S99Q87 manufacturing car : 2017 Car maker :Mercedes
Please enter the car license number you want to rent from the above valid cars : S99Q87
Enter the name : Alaa
Enter the id : 123
Note: the valid date formates allowed to enter here are dd-mm-yy , dd/mm/yy , or dd monthName yy
Enter the date of birth :4-8-2010
Enter the mobile number : 12345
Enter the rental money need to be paid : 120
according to you DOB you are under 16 , so you can't rent a car
If you made a mistake in entering your DOB , you can try again (y) or ending the task (f) y
Enter the name : Alaa
Enter the id : 123
Note: the valid date formates allowed to enter here are dd-mm-yy , dd/mm/yy , or dd monthName yy
Enter the date of birth :3-2-2001
Enter the mobile number : 1234
Enter the rental money need to be paid : 120
Task 3 done successfully
Do you want to try again (y) , or terminate (t) ? |
```

Figure 7: Task3 Output

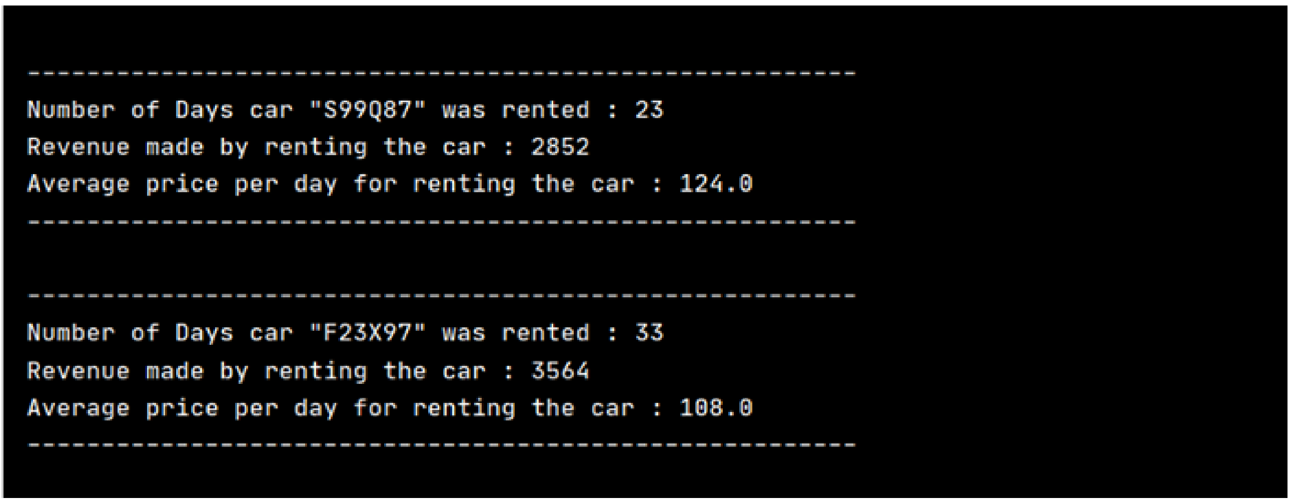
3.2.4 Task4

When choosing to perform task 4, all calculated statistics is printed.



```
Please Enter the number of the task you would like to do :4
---------------statistics about each car----------------
--------------------------------------------------------
Number of Days car "D56V55" was rented : 64
Revenue made by renting the car : 4352
Average price per day for renting the car : 68.0
--------------------------------------------------------

--------------------------------------------------------
Number of Days car "R48J68" was rented : 190
Revenue made by renting the car : 20646
Average price per day for renting the car : 108.66315789473684
--------------------------------------------------------

--------------------------------------------------------
Number of Days car "D76M66" was rented : 32
Revenue made by renting the car : 4544
Average price per day for renting the car : 142.0
--------------------------------------------------------

--------------------------------------------------------
Number of Days car "M83P71" was rented : 73
Revenue made by renting the car : 7555
Average price per day for renting the car : 103.4931506849315
--------------------------------------------------------
```

Figure 8: Task4 Output

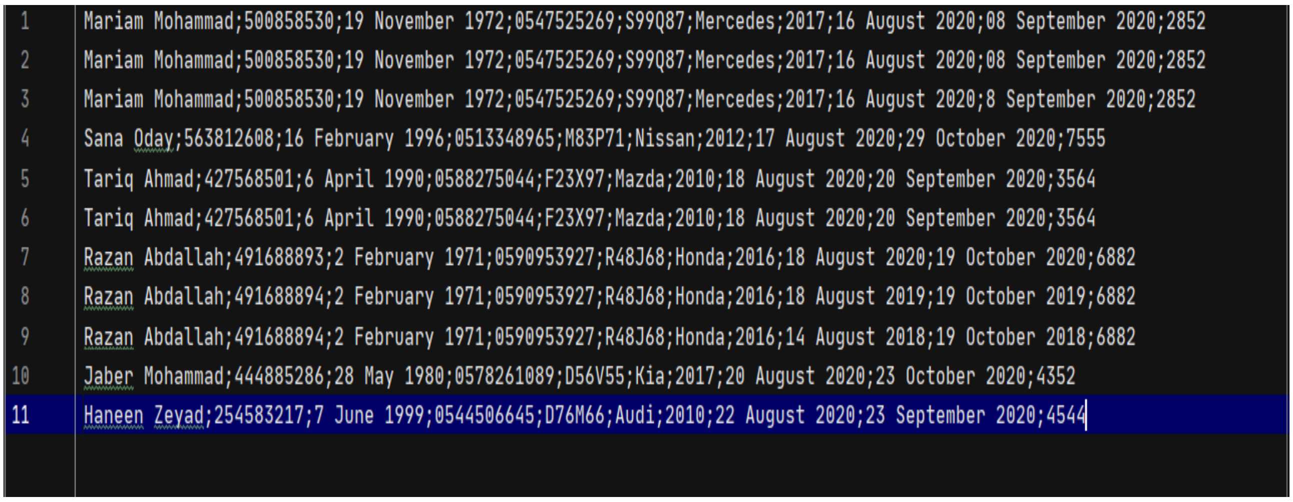


```
--------------------------------------------------------
Number of Days car "S99Q87" was rented : 23
Revenue made by renting the car : 2852
Average price per day for renting the car : 124.0
--------------------------------------------------------


--------------------------------------------------------
Number of Days car "F23X97" was rented : 33
Revenue made by renting the car : 3564
Average price per day for renting the car : 108.0
--------------------------------------------------------
```

Figure 9: Task4 Output

3.2.5 CarRentalCompleted file

Following is a snapshot of the CarRentalCompleted file that has the completed list printed on it (with duplicates), and we have taken this snapshot before performing task 3 and adding a new person, in case study 2, this case will be covered.

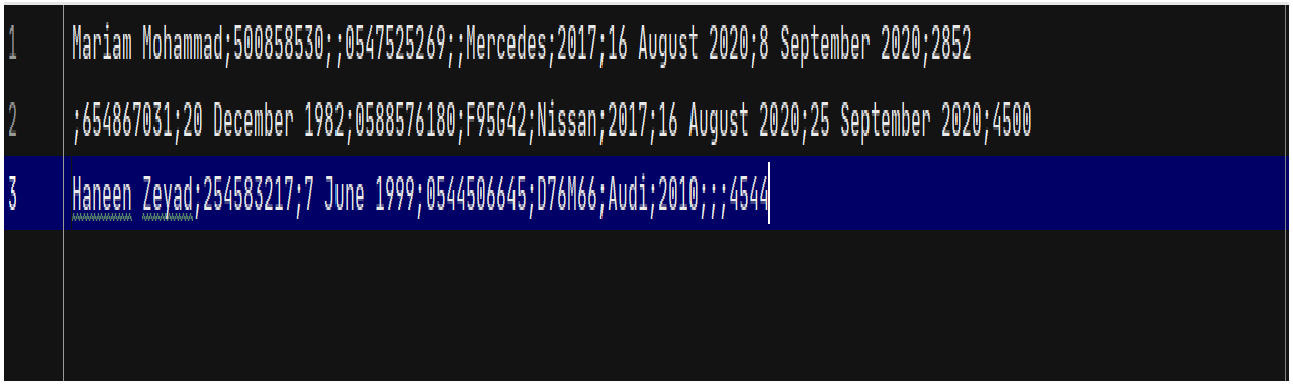


```
Mariam Mohammad;500858530;19 November 1972;0547525269;S99Q87;Mercedes;2017;16 August 2020;08 September 2020;2852
Mariam Mohammad;500858530;19 November 1972;0547525269;S99Q87;Mercedes;2017;16 August 2020;08 September 2020;2852
Mariam Mohammad;500858530;19 November 1972;0547525269;S99Q87;Mercedes;2017;16 August 2020;8 September 2020;2852
Sana Oday;563812608;16 February 1996;0513348965;M83P71;Nissan;2012;17 August 2020;29 October 2020;7555
Tariq Ahmad;427568501;6 April 1990;0588275044;F23X97;Mazda;2010;18 August 2020;20 September 2020;3564
Tariq Ahmad;427568501;6 April 1990;0588275044;F23X97;Mazda;2010;18 August 2020;20 September 2020;3564
Razan Abdallah;491688893;2 February 1971;0590953927;R48J68;Honda;2016;18 August 2020;19 October 2020;6882
Razan Abdallah;491688894;2 February 1971;0590953927;R48J68;Honda;2016;18 August 2019;19 October 2019;6882
Razan Abdallah;491688894;2 February 1971;0590953927;R48J68;Honda;2016;14 August 2018;19 October 2018;6882
Jaber Mohammad;444885286;28 May 1980;0578261089;D56V55;Kia;2017;20 August 2020;23 October 2020;4352
Haneen Zeyad;254583217;7 June 1999;0544506645;D76M66;Audi;2010;22 August 2020;23 September 2020;4544
```

Figure 10: file Output

3.2.6 CarRentalMissing file

Following is a snapshot of the CarRentalMissing file which has mainly the three cases that could not be recovered, one missing the cl, one does not have any data in common with other entries, and the last one is missing the start and end date, and so they can not be predicted (note this case is not calculated in the missing personal information in task 1 summary, as we have decided to consider start and end date as data related to the car not as personal).

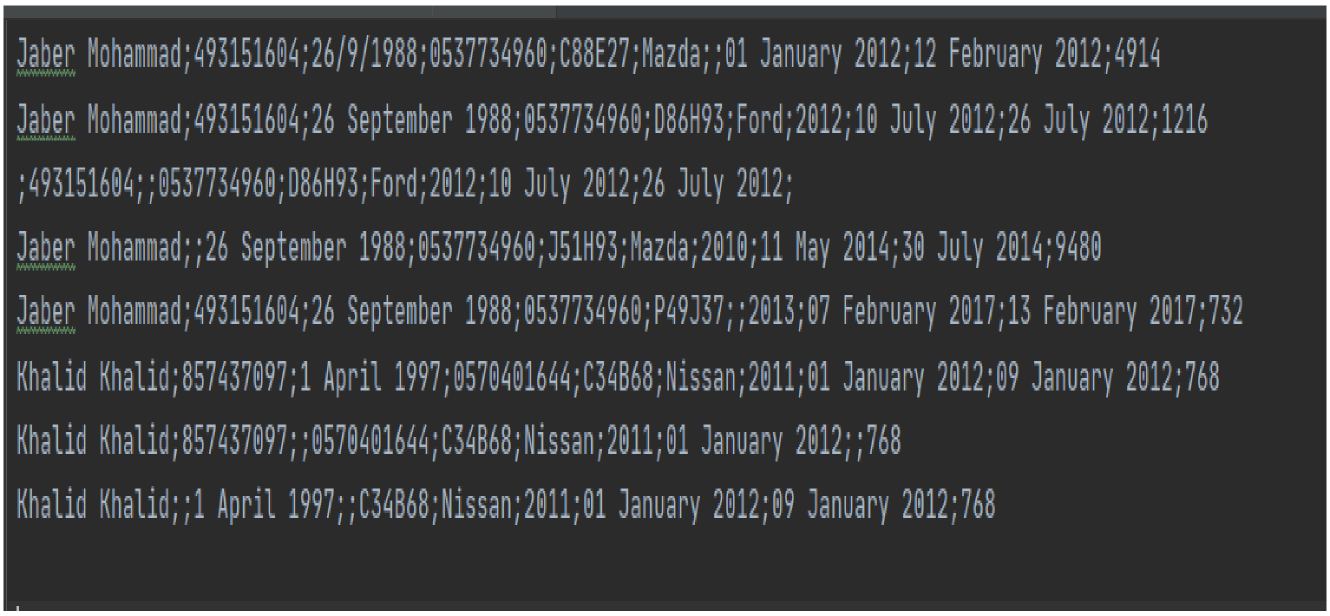


```
Mariam Mohammad;500858530;;0547525269;;Mercedes;2017;16 August 2020;8 September 2020;2852
;654867031;20 December 1982;0588576180;F95G42;Nissan;2017;16 August 2020;25 September 2020;4500
Haneen Zeyad;254583217;7 June 1999;0544506645;D76M66;Audi;2010;;;4544
```

Figure 11: file Output

4 STUDY CASE 2

4.1 CarRentalOld.txt



```
Jaber Mohammad;493151604;26/9/1988;0537734960;C88E27;Mazda;;01 January 2012;12 February 2012;4914
Jaber Mohammad;493151604;26 September 1988;0537734960;D86H93;Ford;2012;10 July 2012;26 July 2012;1216
;493151604;;0537734960;D86H93;Ford;2012;10 July 2012;26 July 2012;
Jaber Mohammad;;26 September 1988;0537734960;J51H93;Mazda;2010;11 May 2014;30 July 2014;9480
Jaber Mohammad;493151604;26 September 1988;0537734960;P49J37;;2013;07 February 2017;13 February 2017;732
Khalid Khalid;857437097;1 April 1997;0570401644;C34B68;Nissan;2011;01 January 2012;09 January 2012;768
Khalid Khalid;857437097;;0570401644;C34B68;Nissan;2011;01 January 2012;;768
Khalid Khalid;;1 April 1997;;C34B68;Nissan;2011;01 January 2012;09 January 2012;768
```

Figure 12: CarRentalOld.txt

In the following few pages, we will be considering this text file to read from. Few Notes about this file:

(a) The first entry with wrong birth date format, and a missing year.

(b) The second entry with no missing fields, and not a duplicate of the first.

(c) The third entry with missing name,date of birth, amount paid, but can be recovered from the second entry as it is a duplicate.

(d) The fourth entry with missing id, but that can not be recovered, as the cl in this entry is unique and not common with others Jaber Mohammed.

(e) The fifth entry is Jaber Mohammed, with another new cl, and a missing cm, which can not be recovered in this case, as P49J37 (cl) is not mentioned in any other entries. However, if cl was mentioned even if for a person other than Jaber Mohammed, the cm, and year could be filled, as in Razan Abdallah in the first study case.

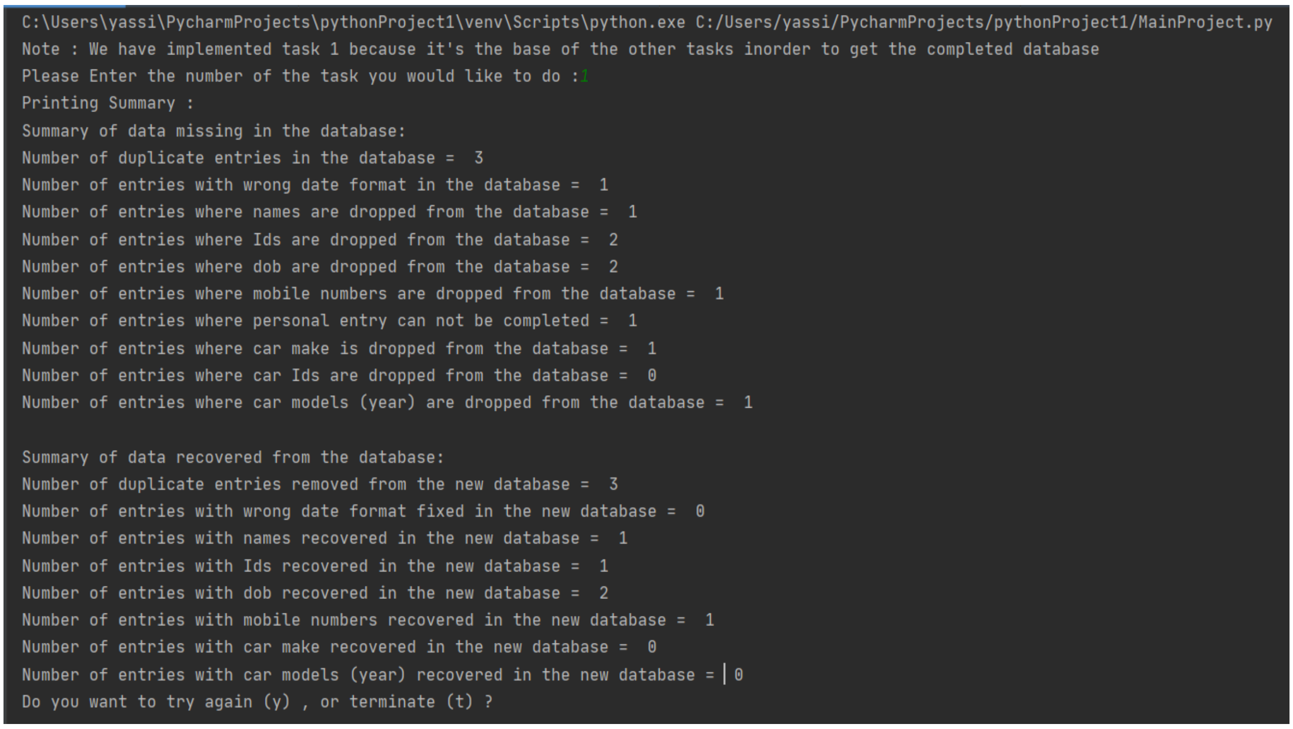
(f) The sixth entry hasn't any missing fields so it's ready .

(g) the seventh entry with missing date of birth (DOB) , but it can be recovered using the sixth entry because it's belong to the same person , and this entry also have another missing fields which is the end date (ed) and can be recovered in the same way as the DOB , because simply it's a duplicates of the sixth entry .

(h) the eighth entry has a missing field which is the id number , but it can be recovered using the sixth entry because it's a duplicate of this entry

4.2 Discussion

4.2.1 Task1



```
C:\Users\yassi\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/yassi/PycharmProjects/pythonProject1/MainProject.py
Note : We have implemented task 1 because it's the base of the other tasks inorder to get the completed database
Please Enter the number of the task you would like to do :1
Printing Summary :
Summary of data missing in the database:
Number of duplicate entries in the database =  3
Number of entries with wrong date format in the database =  1
Number of entries where names are dropped from the database =  1
Number of entries where Ids are dropped from the database =  2
Number of entries where dob are dropped from the database =  2
Number of entries where mobile numbers are dropped from the database =  1
Number of entries where personal entry can not be completed =  1
Number of entries where car make is dropped from the database =  1
Number of entries where car Ids are dropped from the database =  0
Number of entries where car models (year) are dropped from the database =  1

Summary of data recovered from the database:
Number of duplicate entries removed from the new database =  3
Number of entries with wrong date format fixed in the new database =  0
Number of entries with names recovered in the new database =  1
Number of entries with Ids recovered in the new database =  1
Number of entries with dob recovered in the new database =  2
Number of entries with mobile numbers recovered in the new database =  1
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
Do you want to try again (y) , or terminate (t) ?
```

Figure 13: Task1 Output

Task 1 as mentioned above, does the processing for the data obtained from the file,it edits the date format, recovers the entries that are missing some fields, and finally remove duplicates.And to view the output of this task, this summary is printed onscreen, when the user chooses to do task 1. Considering this output in more details:

(a) We have 3 duplicates, 1 of Jaber Mohammad (third entry is a duplicate of the second entry),and 2 of Khalid Khalid have a duplicate as well.

(b) We have one date in wrong format in entry 1 .

(c) We have one name dropped from the database which is the third .

(d) We have two personal ids missing from the database in the third and fourth entry .

(e) We have two entries with dropped date of birth "the third entry and the seventh entry" .

(f) We have a mobile number dropped from the database which is in the eighth entry , but it can be recovered since there is a duplicate of this entry .

(g) We have one entry where the personal information can't be completed which is the fourth entry , because the missing field is the id number , and this field to be recovered need to have a duplicate of the whole entry, but here there is no duplicate .

(h) we have one entry with missing car make (cm) , which is the fifth entry in the database .

(i) We have no entry with missing id .

(j) We have just one entry with missing car model (year) , which is the first entry .

After doing the processing for the data , We notice that :

(a) All completed duplicates are removed .

(b) the entry with the wrong date format is not completed so the number of completed entry with fixed date format is 0 .

(c) the missing name in the third entry was recovered because there is a duplicate of this entry , and the name in its duplicate is exist .

(d) only the missing id in the eighth entry was recovered using it's duplicate , but the missing id in the fourth entry couldn't be recovered .

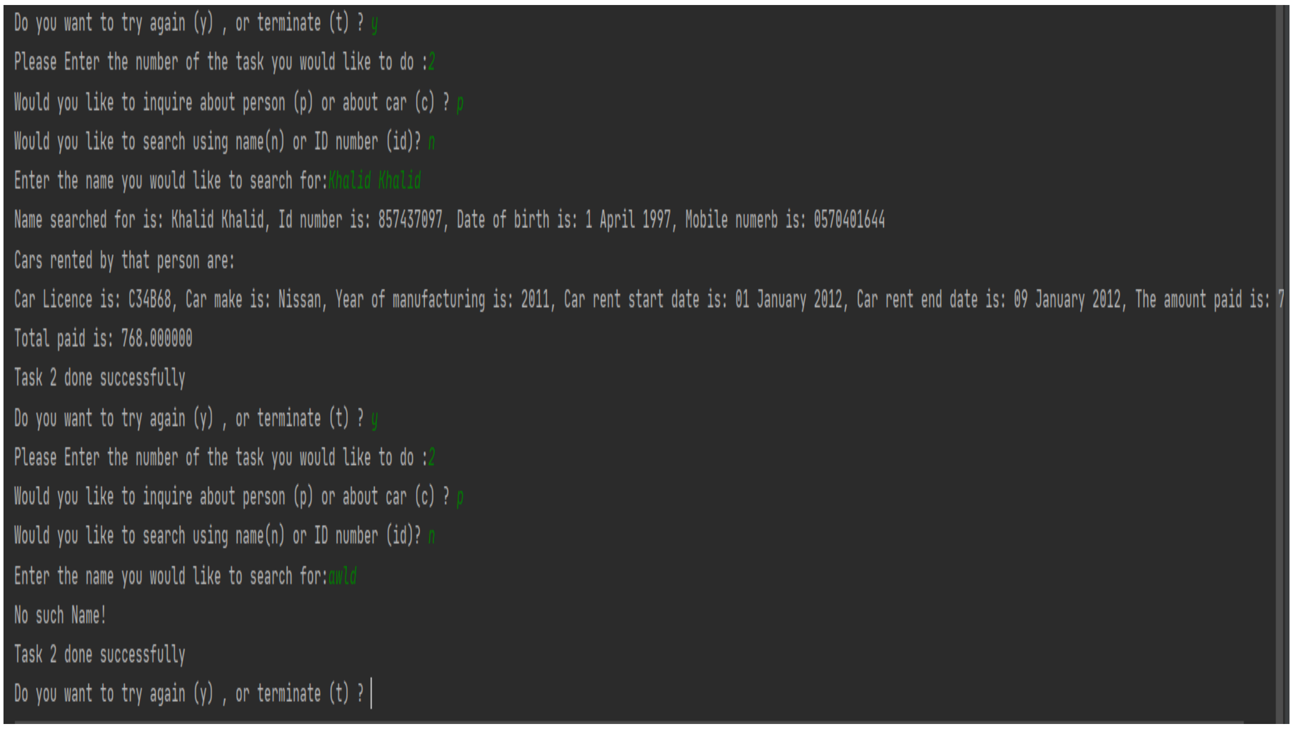
(e) the missing date of birth in the third and seventh entry were recovered using their personal id .

(f) the missing mobile number in the eighth entry was recovered because of the duplicate .

(g) the missing (cm) in the first entry couldn't be recovered because there is no entry with the same car id to get the car make from .

4.2.2 Task2

Now moving to task 2 , after the program finishes task 1 , it asks the user , whether he/she wants to try another task or terminate the program , we have chosen to try another task , and chosen task 2 which dose the inquiry , and then a message asking whether the inquiry will be using personal information or cars information , so we have chosen to inquire using personal information using it's name which is "Khalid Khalid" , then the program print the basic information of "Khalid Khalid" and then print all the cars rented by "Khalid Khalid" , and then we have chosen to try again and chosen task 2 again , but now we have tried to inquire about a person that is not exist in the completed database so the output was "No such person!" .

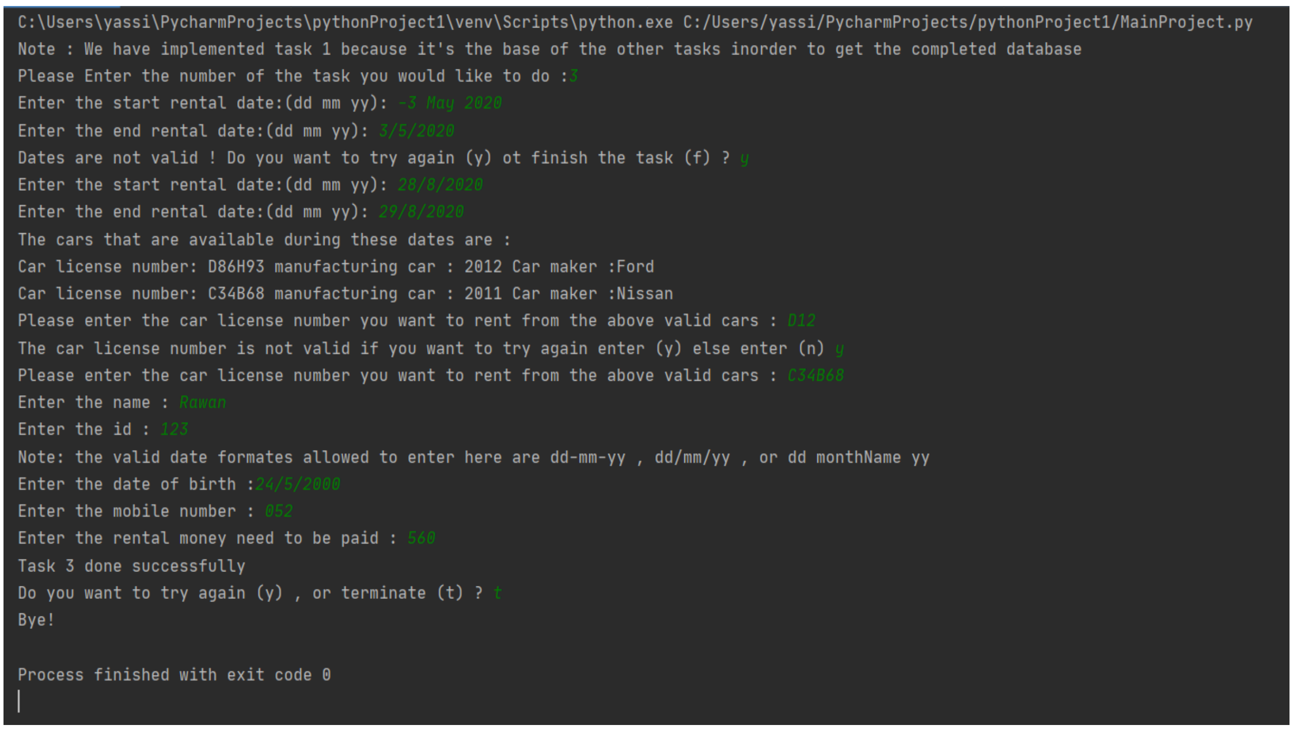


```
Do you want to try again (y) , or terminate (t) ? y
Please Enter the number of the task you would like to do :2
Would you like to inquire about person (p) or about car (c) ? p
Would you like to search using name(n) or ID number (id)? n
Enter the name you would like to search for:Khalid Khalid
Name searched for is: Khalid Khalid, Id number is: 857437097, Date of birth is: 1 April 1997, Mobile numerb is: 0570401644
Cars rented by that person are:
Car Licence is: C34B68, Car make is: Nissan, Year of manufacturing is: 2011, Car rent start date is: 01 January 2012, Car rent end date is: 09 January 2012, The amount paid is: 7
Total paid is: 768.000000
Task 2 done successfully
Do you want to try again (y) , or terminate (t) ? y
Please Enter the number of the task you would like to do :2
Would you like to inquire about person (p) or about car (c) ? p
Would you like to search using name(n) or ID number (id)? n
Enter the name you would like to search for:awld
No such Name!
Task 2 done successfully
Do you want to try again (y) , or terminate (t) ?
```

Figure 14: Task2 Output

4.2.3 Task3

Now we have chosen to perform task 3, which asks the user mainly to enter two dates,to see all available cars during these dates, and then to allow the user to choose a car,and then filling the remaining information. First our program check if the entered dates are valid and logical or not , and in the above case we have entered the start date with illogical value which is in negative , so the program realize that error and show an error message for the user , and ask if we want to try again , and we have chosen to try again , and this time we have entered valid dates , then the program print the car information for all valid cars in those dates , then we have tried to enter a car id which is not exist in the above cars list , so the program detects this error and print a message error , and we have tried again and this time we have enter a valid id , and then we have entered all the required information to rent the car , and finally task 3 done successfully and the new renting was added to the completed database .

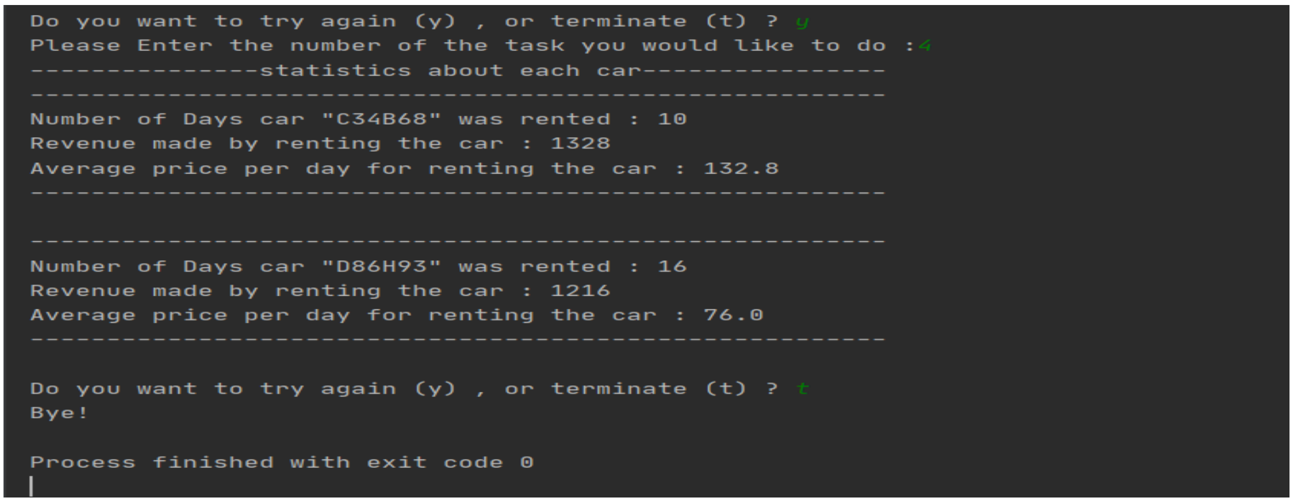


```
C:\Users\yassi\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/yassi/PycharmProjects/pythonProject1/MainProject.py
Note : We have implemented task 1 because it's the base of the other tasks inorder to get the completed database
Please Enter the number of the task you would like to do :3
Enter the start rental date:(dd mm yy): -3 May 2020
Enter the end rental date:(dd mm yy): 3/5/2020
Dates are not valid ! Do you want to try again (y) ot finish the task (f) ? y
Enter the start rental date:(dd mm yy): 28/8/2020
Enter the end rental date:(dd mm yy): 29/8/2020
The cars that are available during these dates are :
Car license number: D86H93 manufacturing car : 2012 Car maker :Ford
Car license number: C34B68 manufacturing car : 2011 Car maker :Nissan
Please enter the car license number you want to rent from the above valid cars : D12
The car license number is not valid if you want to try again enter (y) else enter (n) y
Please enter the car license number you want to rent from the above valid cars : C34B68
Enter the name : Rawan
Enter the id : 123
Note: the valid date formates allowed to enter here are dd-mm-yy , dd/mm/yy , or dd monthName yy
Enter the date of birth :24/5/2000
Enter the mobile number : 052
Enter the rental money need to be paid : 560
Task 3 done successfully
Do you want to try again (y) , or terminate (t) ? t
Bye!

Process finished with exit code 0
```

Figure 15: Task3 Output

4.2.4 Task4

When choosing to perform task 4 , all calculated statistics is printed

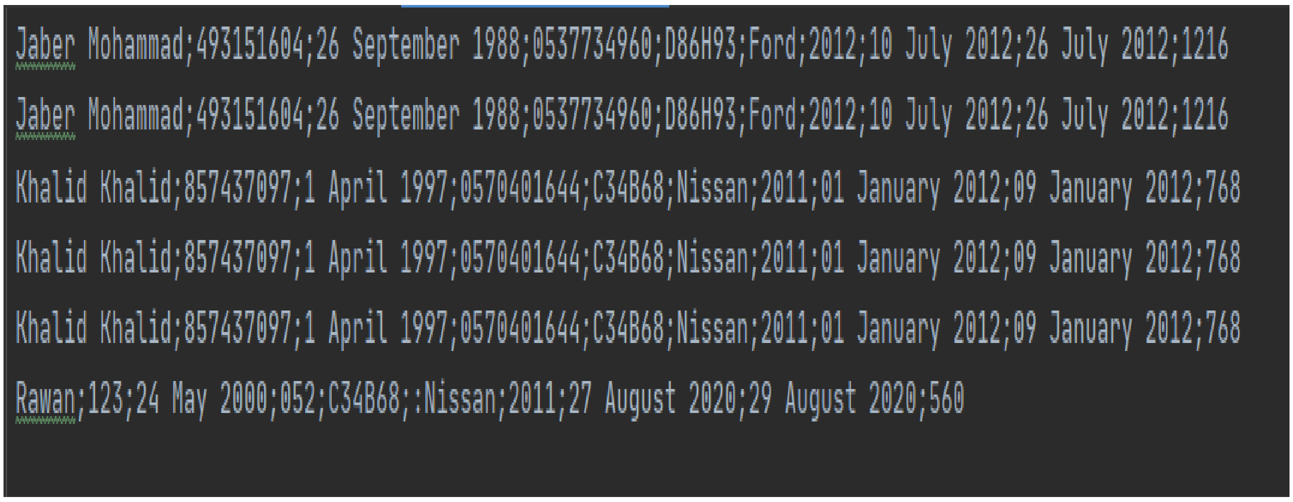


```
Do you want to try again (y) , or terminate (t) ? y
Please Enter the number of the task you would like to do :4
---------------statistics about each car----------------
---------------------------------------------------------
Number of Days car "C34B68" was rented : 10
Revenue made by renting the car : 1328
Average price per day for renting the car : 132.8
---------------------------------------------------------

---------------------------------------------------------
Number of Days car "D86H93" was rented : 16
Revenue made by renting the car : 1216
Average price per day for renting the car : 76.0
---------------------------------------------------------

Do you want to try again (y) , or terminate (t) ? t
Bye!

Process finished with exit code 0
```

Figure 16: Task4 Output

4.2.5 CarRentalCompleted file

Following is a snapshot of the CarRentalCompleted file that has the completed list- printed on it (with duplicates), and here the new renting that happened in task 3 is added to the completed database .

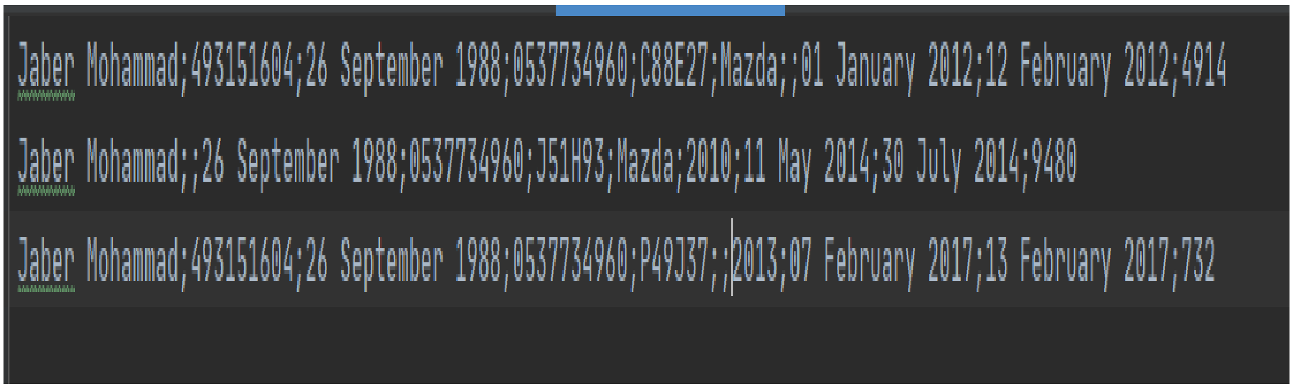


```
Jaber Mohammad;493151604;26 September 1988;0537734960;D86H93;Ford;2012;10 July 2012;26 July 2012;1216
Jaber Mohammad;493151604;26 September 1988;0537734960;D86H93;Ford;2012;10 July 2012;26 July 2012;1216
Khalid Khalid;857437097;1 April 1997;0570401644;C34B68;Nissan;2011;01 January 2012;09 January 2012;768
Khalid Khalid;857437097;1 April 1997;0570401644;C34B68;Nissan;2011;01 January 2012;09 January 2012;768
Khalid Khalid;857437097;1 April 1997;0570401644;C34B68;Nissan;2011;01 January 2012;09 January 2012;768
Rawan;123;24 May 2000;052;C34B68;:Nissan;2011;27 August 2020;29 August 2020;560
```

Figure 17: file Output

4.2.6 CarRentalMissing file

Following is a snapshot of the CarRentalMissing file which has mainly the three cases that could not be recovered, one missing the year (car model) because there is no entry with the same car cl to get the car year from , one missing the id because there is no duplicate of the entry to get the id from, and the last one is missing the car make (cm) .



```
Jaber Mohammad;493151604;26 September 1988;0537734960;C88E27;Mazda;;01 January 2012;12 February 2012;4914
Jaber Mohammad;;26 September 1988;0537734960;J51H93;Mazda;2010;11 May 2014;30 July 2014;9480
Jaber Mohammad;493151604;26 September 1988;0537734960;P49J37;;2013;07 February 2017;13 February 2017;732
```

Figure 18: file Output

5 Conclusion and Future Work

In this project we have learned a lot about python programming , also we have succeeded in implementing a simple system for a car rental company , and tried to include as much as possible cases to deal with any unexpected error from the user . As a future plan , this project encourage us to dive more and more in python programming because simply it's the language of the future , and it can be used for a wide range of fields like AI , Machine learning , web development , and so on .

6 REFERENCES

1. https://www.tutorialspoint.com/python/python_date_time.htm
   [Accessed on 26/Aug/2020 at 1:00 pm]

7 Appendices

7.1 Appendix A - CarRental Class

```
class CarRental:
    #Constructor and attributes
    def __init__(self, name, id, dob, mobile,cl, cm, year, sd, ed, rb):
        self.__name = name
        self.__id = id
        self.__dob = dob
        self.__mobile = mobile
        self.__cl = cl
        self.__cm = cm
        self.__year = year
        self.__sd = sd
        self.__ed = ed
        self.__rb = rb
    #Getters
    def getName(self):
        return self.__name
    def getId(self):
        return self.__id
    def getDob(self):
        return self.__dob
    def getMobile(self):
        return self.__mobile
    def getCl(self):
        return self.__cl
    def getCm(self):
        return self.__cm
    def getYear(self):
        return self.__year
    def getSd(self):
        return self.__sd
    def getEd(self):
        return self.__ed
    def getRb(self):
        return self.__rb
    #Setters
    def setName(self, name):
        self.__name = name
    def setId(self, id):
        self.__id = id
    def setDob(self, dob):
        self.__dob = dob
    def setMobile(self, mobile):
        self.__mobile = mobile
    def setCl(self, cl):
        self.__cl = cl
    def setCm(self, cm):
        self.__cm= cm
    def setYear(self, year):
        self.__year = year
    def setSd(self, sd):
        self.__sd = sd
    def setEd(self, ed):
        self.__ed = ed
    def setRb(self, rb):
        self.__rb = rb
    #function to print the attributes of the instances of the class
    def printInfo(self):
        return (self.__name + " " + self.__id + " " + self.__dob + " " +
            self.__mobile + " " + self.__cl + " " + self.__cm + " " +
            self.__year + " " + self.__sd + " " + self.__ed + " " + self.__rb)

    def printFormatedInfo(self):
        return (self.__name + ";" + self.__id + ";" + self.__dob + ";" +
            self.__mobile + ";" + self.__cl +
                ";" + self.__cm + ";" + self.__year + ";" + self.__sd + ";" +
                    self.__ed + ";" + self.__rb)
```

7.2 Appendix B - Car Class

```python
class Car:
    #Constructor and attributes
    def __init__(self,cl, cm, year):
        self.__cl = cl
        self.__cm = cm
        self.__year = year
    #Getters
    def getCl(self):
        return self.__cl
    def getCm(self):
        return self.__cm
    def getYear(self):
        return self.__year
    #Setters
    def setCl(self, cl):
        self.__cl = cl
    def setCm(self, cm):
        self.__cm= cm
    def setYear(self, year):
        self.__year = year
    #function to print the attributes of the instances of the class
    def printInfo(self):
        print(self.__cl + " " + self.__cm + " " + self.__year )
```

7.3 Appendix C - main

```
#reading all data from the file
i=0
with open('CarRentalInfo.txt')as f:
    info=f.read()

#splitting the read file according to the new line
infoList=info.split("\n")
#these are the lists to hold the information read from file seperately
name=[]
id=[]
dob=[]
mobile=[]
cl=[]
cm=[]
year=[]
sd=[]
ed=[]
rb=[]
#putting the data in each list seperately
for i in range(len(infoList)-1):
    #print(infoList[i])
    data=infoList[i].split(";")
    #print(data)
    name.append(data[0])
    id.append(data[1])
    dob.append(data[2])
    mobile.append(data[3])
    cl.append(data[4])
    cm.append(data[5])
    year.append(data[6])
    sd.append(data[7])
    ed.append(data[8])
    rb.append(data[9])


choice=input("Would you like to inquire about a person(p) or about a car using
    cl(cl)")
#if the search is done for a person:
if(choice=='p'):
    choice2=input("Would you like to search using Name(n) or ID number (id)")
```

```
#if the required search is done using the name
if(choice2=='n'):
  inputName=input("Enter the name of the person:")
  flag=0
  #total is the sum of the paid money by the person
  total=0
  for i in range(len(name)):
      if(inputName==name[i]):
          if(total==0):
              print("ID is:"+id[i])
              print("Date of Birth is:"+dob[i])
              print("Mobile is:"+mobile[i])
              print("The rented cars by this person are:")
          print(cm[i])
          t=int(rb[i])
          total+=t
          flag=1
  if(flag==1):
      print("Total Payment is: %f"%total)
  else:
        print("No such Name!")
#if the search is done using id:
elif(choice2=='id'):
  inputId = input("Enter the ID of the person:")
  flag = 0
  total = 0
  for i in range(len(id)):
      if (inputId == id[i]):
          if (total == 0):
              print("Name is:" + name[i])
              print("Date of Birth is:" + dob[i])
              print("Mobile is:" + mobile[i])
              print("The rented cars by this person are:")
          print(cm[i])
          t = int(rb[i])
          total += t
          flag = 1
  if (flag == 1):
      print("Total Payment is: %f" % total)
  else:
      print("No such ID number!")
else:
     print("Please Try again")
#if the search is done for the car using CL
elif(choice=='cl'):
    inputCL=input("Enter the CL:")
    total=0
    flag=0
    for i in range(len(cl)):
        if(inputCL==cl[i]):
            flag=1
            if(total==0):
              print("Car's information: ")
              print("Car make: "+cm[i])
              print("Year of manufacturing: "+year[i])
              print("-------")
              print("People who have rented this car are:")
            print("Name: "+name[i])
            print("Id: "+id[i])
            print("Starting from: "+sd[i])
            print("Till: "+ed[i])
            print("-------")
            t=int(rb[i])
            total+=t
    if(flag==1):
        print("Revenue is: %f"%total)
    else:
        print("No Such Car!")

else:
   print("WRONG choice! BYE!!")
f.close()
```

7.4 Appendix D - Task1

```
import CarRental
import datetime
import re

monthName = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
    'August', 'September', 'October',
            'November', 'December']


#modifying date format function
def modifyDate(entry):

    # We will be modifying:
    # 1)The date of birth of each object
    flag = 0
    object = re.search(r"[-/][0-9]*[-/]", entry.getDob(),re.M | re.S) # if the
        date has the format of "/digit/" or "-digit-" it should be modified
    if object:
        replaceString = monthName[int(object.group()[1:len(
            object.group()) - 1]) - 1] # the replace string is the monthName
                element indexed by the digit
        entry.setDob(re.sub(r"[-/][0-9]*[-/]", " " + replaceString + " ",
            entry.getDob()))
        flag = 1
    # 2)The start rental date of each object
    object = re.search(r"[-/][0-9]*[-/]", entry.getSd(), re.M | re.S)
    if object:
        replaceString = monthName[int(object.group()[1:len(object.group()) -
            1]) - 1]
        entry.setSd(re.sub(r"[-/][0-9]*[-/]", " " + replaceString + " ",
            entry.getSd()))
        flag= 1
    # 3)The end rental date of each object
    object = re.search(r"[-/][0-9]*[-/]", entry.getEd(), re.M | re.S)
    if object:
        replaceString = monthName[int(object.group()[1:len(object.group()) -
            1]) - 1]
        entry.setEd(re.sub(r"[-/][0-9]*[-/]", " " + replaceString + " ",
            entry.getEd()))
        flag = 1
    return flag

# function to try and complete the missing information
def completeMissing(list):
    missingList=[] # this is a list to put the elements which can't be completed
    completeList=[] # this is a list to put the elements which are completed
    printList1 = [0,0,0,0,0,0,0,0,0,0] #this list will be counting data for
        printing summary for completed/recovered entries
    printList2 = [0,0,0,0,0,0,0,0,0,0] #this list will be counting data for
        printing summary for uncompleted entries(either missing cl, or missing
        more than a field so that it can not be filled
    for i in list:
        countList = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0] #this list will be couting
            the data for printing summary for each entry seperately
        flag = modifyDate(i)  #if any date in the entry is modified, flag is set
            to one
        countList[1] = flag  #countList[1] represents if entry is with wrong
            date format
        if(i.getCl()!=""):   #if cl is missing, then this entry can not be
            completed
            if (i.getId() == "" or i.getName() == "" or i.getDob() == "" or
                i.getMobile() == "" or i.getCm() == "" or
                    i.getYear() == "" or i.getSd() == "" or i.getEd() == "" or
                        i.getRb() == ""):
                for j in list: #start searching the list, to fill the
                    incompleted entry
                    if (i.getCl() == j.getCl()):
                        # get the missing cm from the j.getCm is it's exist
                        if (i.getCm() == ""):
                            countList[7] = 1 #countList[7] states whether a
                                missing cm or not
                            if(j.getCm() != ""):
                              i.setCm(j.getCm())
                        # get the missing year from the j.getYear is it's exist
                        if (i.getYear() == ""):
                            countList[9] = 1 #countList[9] states whether the
                                year is missing in this entry or not
                            if(j.getYear() != ""):
                              i.setYear(j.getYear())
                        if (i.getSd() == "" and j.getSd() != "" and i.getEd() !=
                            "" and compareDates(i.getEd(),j.getEd())==1): #trying
                            to fill the start date,
                        #if missing start date, then need to check that end date
                            is the same,
                        #if both dates are missing, then entry can not be
                            completed
                            i.setSd(j.getSd())
                        if (i.getEd() == "" and j.getEd() != "" and i.getSd() !=
                            "" and compareDates(i.getSd(),j.getSd())==1): #trying
                            to fill the end date
                        #if missing end date, then start date should be checked
                            before completing entry
                            i.setEd(j.getEd())
                        # get the missing rb from the j.getRb is it's exist
                        if (i.getRb() == "" and j.getRb() != "" and ( (i.getEd()
                            != "" and compareDates(i.getEd(),j.getEd())==1)
                            or (i.getSd() != "" and
                                compareDates(i.getSd(),j.getSd())==1) ) ):
                        #before completing the Rb entry, need to check the start
                            and end date, to make sure we are dealing with same
                            entries
                        #note: no two cars with same Cl and same start date and
                            end date are not duplicates
                            i.setRb(j.getRb())
                        if (i.getId() == ""):
                            countList[3] = 1 #countList[3] states wether id is
                                missing in the given entry
                            #if missing and before filling it, need to check that
                                the cl and start date and end date are the same
                                for this person
                            if(j.getId() != "" and i.getCl() == j.getCl() and
                                i.getEd() != "" and
                                compareDates(i.getEd(),j.getEd())==1
                                and i.getSd() != "" and
                                    compareDates(i.getSd(),j.getSd())==1):
                                i.setId(j.getId())
                        if (i.getName() == ""):
                            countList[2] = 1 #countList[2] states wether name is
                                missing in the given entry
                            #if name is missing, check same id ( as cl is already
                                checked) before filling,smae concept is used for
                                Dob and mobile
                            if(j.getName() != "" and (i.getId()==j.getId())):
                              i.setName(j.getName())
```

```
                    # get the missing id from the j.getId if it's exist
                if (i.getDob() == ""):
                    countList[4] = 1 #countList[4] states wether Dob is
                        missing in the given entry
                    if(j.getDob() != "" and (i.getId()==j.getId())):
                       i.setDob(j.getDob())
                    # get the missing mobile from the j.getMobile if it's
                        exist
                if (i.getMobile() == ""):
                    countList[5] = 1 #countList[5] states wether id is
                        missing in the given entry
                    if(j.getMobile() != "" and (i.getId() == j.getId())):
                      i.setMobile(j.getMobile())


else:
    if (i.getCm() == ""):
        countList[7] = 1
    if (i.getYear() == ""):
        countList[9] = 1
    if (i.getId() == ""):
        countList[3] = 1
    if (i.getName() == ""):
        countList[2] = 1
    if (i.getDob() == ""):
        countList[4] = 1
    if (i.getMobile() == ""):
        countList[5] = 1
    countList[8] = 1 #countList[8] states wether cl is missing in the
        given entry

# now if there are still missing fields then move that element with
    missing fields into a list called missingList
if (i.getCl() == "" or i.getId() == "" or i.getName() == "" or
    i.getDob() == "" or i.getMobile() == "" or i.getCm() == "" or
    i.getYear() == ""
    or i.getSd() == "" or i.getEd() == "" or i.getRb() == ""):
    if(i.getId() == "" or i.getName() == "" or i.getDob() == "" or
        i.getMobile() == ""):
       countList[6] = 1 #countList[6] states wether the personal
           information in this entry can't be recovered
    missingList.append(i)
```

```
            for k in range(0,10):
                printList2[k] += countList[k]
        # if the current entry dosn't have any missing fields then mve it into
            a list called completeList
        else:
            completeList.append(i)
            for k in range(0, 10):
                printList1[k] += countList[k]

    printList2[0] = calcNumOfDuplicates(list)

    #moving all not comlete entries to a new text file
        called(CarRentalMissing.txt)
    f=open("CarRentalMissing.txt","w")
    for i in missingList:
        f.write(str(i.printFormatedInfo()))
        f.write("\n")
    f.close()
    # moving all comlete entries to a new text file
        called(CarRentalCompleted.txt
    f=open("CarRentalCompleted.txt","w")
    for i in completeList:
        f.write(str(i.printFormatedInfo()))
        f.write("\n")
    f.close()

    l = removeDuplicate(completeList)
    completeList = l[0]
    printList1[0] = l[1]
    #printSummary(printList1, printList2)

    return [completeList,printList1,printList2]

def printSummary(printList1,printList2):
    #printing summary before recovering, requires printing the summation of
        printList1[i]+printList2[i] in most cases
    print("Summary of data missing in the database:")
    print("Number of duplicate entries in the database = " , printList2[0] )
    print("Number of entries with wrong date format in the database = " ,
        (printList1[1] + printList2[1]))
    print("Number of entries where names are dropped from the database = "
        ,(printList1[2] + printList2[2]))
    print("Number of entries where Ids are dropped from the database = " ,
        (printList1[3] + printList2[3]))
    print("Number of entries where dob are dropped from the database = ",
        (printList1[4] + printList2[4]))
    print("Number of entries where mobile numbers are dropped from the database
        = " , (printList1[5] + printList2[5]))
    print("Number of entries where personal entry can not be completed = " ,
        printList2[6])
    print("Number of entries where car make is dropped from the database = " ,
        (printList1[7] + printList2[7]))
    print("Number of entries where car Ids are dropped from the database = ",
        (printList2[8]))
    print("Number of entries where car models (year) are dropped from the
        database = " , (printList1[9] + printList2[9]))

    print("")

    #printing summary after recovery, requires printing the printList1 elements
    print("Summary of data recovered from the database:")
    print("Number of duplicate entries removed from the new database = " ,
        printList1[0])
    print("Number of entries with wrong date format fixed in the new database =
        " , printList1[1])
    print("Number of entries with names recovered in the new database = " ,
        printList1[2])
    print("Number of entries with Ids recovered in the new database = " ,
        printList1[3])
    print("Number of entries with dob recovered in the new database = " ,
        printList1[4])
    print("Number of entries with mobile numbers recovered in the new database
        = " , printList1[5])
    print("Number of entries with car make recovered in the new database = " ,
        printList1[7])
    print("Number of entries with car models (year) recovered in the new
        database = " , printList1[9])



#This function will encounter the number of duplicates in the whole database
def calcNumOfDuplicates(list):

    # this list will be a flag to help as know weather this entry is
```

```
        encountered as a duplicates or not
    # so we have initialized it with zeros
    flagList = []
    for i in range(len(list)):
        flagList.append(0)
    # sum variable will contains the number of duplicates in the whole database
    sum = 0

    for i in range(len(list)):
        # if this entry is considered as a duplicate of anther entry or it's
            #cl is not exist or bot the start and end dates are not exist
        # then we can't know if it have a duplicates or not so continue looping
        if(flagList[i]==0 and list[i].getCl()=="" or (list[i].getSd()=="" and
            list[i].getEd()=="") ):
            continue
        # this loop will check the duplicate of the current entry starting from
            the next of this current entry
        for j in range(i+1,len(list)):
            # if this entry is not a duplicate of another entry , and the cars
                ids are identical and some of the start or end dates
            # are identical and not empty then this entry is a duplicate of the
                current entry
            if(flagList[j]==0 and list[i].getCl()==list[j].getCl() and
            ((list[i].getSd()!="" and list[i].getSd()==list[j].getSd()) or
                (list[i].getEd()!="" and list[i].getEd()==list[j].getEd())) ):
                sum += 1
                flagList[j] = 1


    return sum


# function to remove the duplicates from the completed database
def removeDuplicate(list):
    i=0
    sum=0
    while (i < len(list)): # loops through the list
        j=i+1
        # loops through the list starting from the first element in the list
            after the element obtained from the first loop
        while (j < len(list)):
            #if the ids are idenetical then cheack the other field and if they
```

```
               are all identical then remove the second element
            if (list[i].getCl() == list[j].getCl() and
                compareDates(list[i].getSd(),list[j].getSd())==1 and
                compareDates(list[i].getEd(),list[j].getEd())==1 and
                list[i].getName().lower() == list[j].getName().lower() and
                    compareDates(list[j].getDob(),list[i].getDob()) ==1
                 and list[i].getMobile() == list[j].getMobile() and
                     list[i].getCm().lower() == list[j].getCm().lower()
                 and list[i].getYear() == list[j].getYear() and list[i].getRb()
                     == list[j].getRb()):
                    sum += 1
                    list.pop(j)
                    j-=1
            j+=1
        i+=1

    return [list , sum ]


def compareDates(date1, date2):
    # if the date has the format of "/digit/" or "-digit-" it should be modified
    object1 = re.search(r"[-/][0-9]*[-/]", date1, re.M | re.S)
    if object1:
        # the replace string is the monthName element indexed by the digit
        replaceString = monthName[int(object1.group()[1:len(object1.group()) -
            1]) - 1]
        date1 = re.sub(r"[-/][0-9]*[-/]", " " + replaceString + " ", date1)

    object2 = re.search(r"[-/][0-9]*[-/]", date2, re.M | re.S)
    if object2:
        # the replace string is the monthName element indexed by the digit
        replaceString = monthName[int(object2.group()[1:len(object2.group()) -
            1]) - 1]
        date2 = re.sub(r"[-/][0-9]*[-/]", " " + replaceString + " ", date2)
    date1 = datetime.datetime.strptime(date1,"%d %B %Y")
    # we have created a date object to comaper the dates , because maybe date1
        = 1 May 2020 and date2 = 01 May 2020
    # then comparing them as string will yield to be not identical , so using
        date objects will solve the problem
    date2 = datetime.datetime.strptime(date2,"%d %B %Y")

    if (date1 == date2):
        return 1
    else:
        return 0
```

7.5 Appendix E - Task2

```
import re
import CarRental


def inquiryPerson (list):

    choice=input("Would you like to search using name(n) or ID number (id)? ")
    if(choice=='n'): # if the user want to search using the name
        listIds=[] # this list will contains the ids of every person whose name
            is identical to inputName
        inputName=input("Enter the name you would like to search for:")
        inputName=inputName.lower() # convert the inputName into lower case ,
            beacse names in lower case or upper case are the same
        #valid characters in inputName are ( A-Z , a-z , - , _ ,"space" )
        obj = re.search(r"[^A-Za_z \-_]",inputName,re.M|re.I)
        if(obj): # if obj is not null , that's means the inputName contains
            invalid characters
            print("You can not enter a name with invalid characters!")
        else: # else the inoutName is valid
            for i in range(len(list)):
                if (inputName == list[i].getName().lower()): # if they are equal
                    then add that element into the listIds
                    if (listIds.count(list[i].getId()) == 0):
                        listIds.append(list[i].getId())
            if (len(listIds) == 0): # if the length of the listIds is = 0 then
                there is no person with the entered name
                print("No such Name!")
            for i in range(len(listIds)):
                searchId(listIds[i], list)
    elif(choice=='id'): # if the user want to search using the id
        inputId = input("Enter the id you would like to search for:")
        if(inputId.isnumeric()):# checking if the id is consists of digits only
           searchId(inputId,list)
        else: # the entered id is not valid , because it have to consist digits
            only
           print("You can not enter an id with characters other than digits!")
    else:# the entered choice is invalid
        print("Wrong Choice!")
```

```
# search a person using it's id
def searchId(inputId,list):

    total = 0
    flag = 0
    # loops through the lsit
    for i in range(len(list)):
        # if the id is identical to inputId then print person information
        if (inputId == list[i].getId()):
            flag = 1
            # if total is 0 then that's means we need to first print the basic
                inforamtion of the person
            if (total == 0):
                print("Name searched for is: " + list[i].getName() + ", Id
                    number is: " + list[
                    i].getId() + ", Date of birth is: " + list[i].getDob() + ",
                        Mobile numerb is: " + list[
                          i].getMobile())
                print("Cars rented by that person are:")
            total += int(list[i].getRb())
            # now print the other information
            print("Car Licence is: " + list[i].getCl() + ", Car make is: " +
                list[
                i].getCm() + ", Year of manufacturing is: " + list[i].getYear()
                    + ", Car rent start date is: " +
                  list[i].getSd() + ", Car rent end date is: " + list[i].getEd()
                      + ", The amount paid is: " + list[
                      i].getRb())
    # printing the total amount paid by this person for renting the cars
    if (flag == 1):
        print("Total paid is: %f " % total)
    # there is no person with the specified id
    else:
        print("No such ID!Try AGAIN!")


# searching about car using it's cl number
def inquiryCar(list):

    inputCl = input("Enter the car license number you would like to search
        for:")
```

```
if(inputCl.isalnum()):
    total = 0
    flag = 0
    for i in range(len(list)): # loops through the list

        # if the cl of the car is identical to the inputCl then print the
            information
        if (inputCl == list[i].getCl()):
            flag = 1
            # if total is zero then print the basic iformation (the basic
                information will be printed once )
            if (total == 0):
                print("Car license searched for is: " + list[i].getCl() + ",
                    Car make is: " + list[
                    i].getCm() + ",year of manufacturing is: " +
                        list[i].getYear())
                print("Persons rented this car are:")

            total += int(list[i].getRb()) # calculating the revenue mase by
                renting this car

            # printint the other information
            print("Person name is: " + list[i].getName() + " Person id is: "
                + list[i].getId() + " Person DOB is: " +
                  list[i].getDob() + " mobile number is: " +
                      list[i].getMobile() + "\ncar rent start date is: " +
                  list[i].getSd() + " car rent end date is: " +
                      list[i].getEd() + " the amount paid is: " + list[
                      i].getRb())
    if (flag == 1):
        print("Total paid is: %f " % total)
    else:
        print("No such Car!Try AGAIN!")
else:
    print("the entered license number is not valid , it should contains
        letters and digits only ")
```

7.6 Appendix F - Task3

```
import CarRental
import Car
import datetime
import re

def ValidCars(list,setOfCars,carsDict,SD,ED):
    setOfValidCars = setOfCars

    sd = datetime.datetime.strptime(SD, "%d %B %Y")
    ed = datetime.datetime.strptime(ED, "%d %B %Y")

    for i in range(len(list)):
        # if the car from list is in setOfValidCars then cheack the dates to
            know if it’s will still valid or not
        if (list[i].getCl() in setOfValidCars):
            # if there is any intersection between dates then remove the car
                from the setOfValidCars
            currentSd = datetime.datetime.strptime(list[i].getSd(), "%d %B %Y")
            currentEd = datetime.datetime.strptime(list[i].getEd(), "%d %B %Y")
            if (sd == currentSd or (sd > currentSd and sd < currentEd)
                    or ed == currentEd or (ed < currentEd and ed > currentSd)):
                # this car is not available
                # remove this car from setOfValidCars
                setOfValidCars.remove(list[i].getCl())

    # if there isn’t any car avaialbe at these date
    if (len(setOfValidCars) == 0):
        print("There isn’t any car available at these dates (" + sd + "-->" +
            ed + ") ")
    else:
        print("The cars that are available during these dates are :")

        for i in setOfValidCars:
            print("Car license number: " + i + " " + carsDict[i])
    addInfoToList(list, setOfValidCars, carsDict, SD, ED)


def readDates(list,setOfCars,carsDict):
    flag1=0
    flag2 = 0
    while True :
        inputSd = input("Enter the start rental date:(dd mm yy): ")
        inputSd = modifyDate(inputSd)
        inputEd = input("Enter the end rental date:(dd mm yy): ")
        inputEd = modifyDate(inputEd)
        inputSd = modifyDate(inputSd)
        inputEd = modifyDate(inputEd)
        try :
            sd = datetime.datetime.strptime(inputSd, "%d %B %Y")
            ed = datetime.datetime.strptime(inputEd, "%d %B %Y")
            if(ed < sd):
                print("end date must be more than the start date !")
                raise Exception
            today = datetime.datetime.today()
            if (today.date() > sd.date()):
                print("You can not rent a car on a day that is over!")
                raise Exception
            break

        except Exception:
            choice = input("Dates are not valid ! Do you want to try again (y)
                ot finish the task (f) ? ")
            if(choice == "y" or choice == "Y"):
                continue
            else:
                return;

        #if we would like to add more cases
    ValidCars(list,setOfCars,carsDict,inputSd,inputEd)

#here we use this function to modify all entered dates in differed formats to
    the desired format
def modifyDate(date):
    monthName = ['January', 'February', 'March', 'April', 'May', 'June',
        'July', 'August', 'September', 'October', 'November', 'December']
    object = re.search(r"[-/][0-9]*[-/]", date, re.M | re.S) # if the date has
        the format of "/digit/" or "-digit-" it should be modified
    if object:
        replaceString = monthName[int(object.group()[1:len(object.group()) -
            1]) - 1] # the replace string is the monthName element indexed by
            the digit
        date=re.sub(r"[-/][0-9]*[-/]", " " + replaceString + " ", date)
    return date


def addInfoToList(list,setOfValidCars,carsDict,sd,ed):
    flag=0
    while True :
      cl = input("Please enter the car license number you want to rent from the
          above valid cars : ")
      cl = cl.upper()
      for i in setOfValidCars:
          if (cl == i):
              flag=0
              # add the information
              while(True):
                name = input("Enter the name : ")
                id = input("Enter the id : ")
                print("Note: the valid date formates allowed to enter here are
                    dd-mm-yy , dd/mm/yy , or dd monthName yy")
                dob = input("Enter the date of birth :")
                dob = modifyDate(dob)
                mobile = input("Enter the mobile number : ")
                rb = input("Enter the rental money need to be paid : ")
                splitString = carsDict[cl].split(" ")
                year = splitString[3]
                cm = splitString[6]
                obj = re.search(r"[^A-Za-z \-_]",name,re.M|re.I)
                if (obj):
                    print("name is not valid , name should contains letters , - ,
                        _ , or space ")
                    choice = input("Do you want to try again (y) or finish this
                        task(f) ? ")
                elif(not id.isnumeric()):
                    print("Id is not valid , id should contains digits only !")
                    choice = input("Do you want to try again (y) or finish this
                        task(f) ? ")
                elif(not mobile.isnumeric()):
                    print("mobile number is not valid , it should contains digits
                        only !")
                    choice = input("Do you want to try again (y) or finish this
                        task(f) ? ")
                elif(not rb.isnumeric()):
                    print("money paid is not valid , it should contains digits
                        only !")
                    choice = input("Do you want to try again (y) or finish this
                        task(f) ? ")
                else:
                    choice = isDOBValid(dob)
                    flag = choice # will be 1 if there is no error in the date of
                        birth
                    if(flag==1):
                       break;

                if(choice == "y" or choice == "Y"):
                    continue
                else:
                  return ;
              if(flag):
                  obj = CarRental.CarRental(name, id, dob, mobile, cl, cm, year,
                      sd, ed, rb)
                  list.append(obj)
                  file = open("CarRentalCompleted.txt","a")
                  file.write(obj.printFormatedInfo())
                  file.write("\n")
                  break
      if(flag==0):
          choice = input("The car license number is not valid if you want to
              try again enter (y) else enter (n) ")
          if (choice == "y" or choice == "Y"):
              continue
          else:
              break
      else:
          break



def isDOBValid(birthDate):
    try:

        birthDate = datetime.datetime.strptime(birthDate, "%d %B %Y")

        currentDate = datetime.date.today()
        DateBecfor16Years = datetime.date(currentDate.year - 16,
            currentDate.month, currentDate.day)

        # to cheack that the age of the user is valid to rent a car
        if (DateBecfor16Years < birthDate.date()):
            print("according to you DOB you are under 16 , so you can't rent a
                car ")
            choice = input("If you made a mistake in entering your DOB , you can
                try again (y) or ending the task (f) ")
            return choice

    except Exception:
        choice = input("Yor date not valid")
        return choice

    # summary if the return value is "Y" or "y" then that is mean there is an
        error in the date formate and the user want to try again
    # if the return valus is 1 that means there is'nt any error in the
        dateFormate
    # else means there is an error in the dateFormate and the user choice to
        finish the task .
    return 1
```

7.7 Appendix G - Task4

```
import datetime

# function to print the number of days a car was rented , the revenue made by
    each car , and the average price per day for renting each car
def printStatisticsAboutCars(list,setOfCars):

    print("---------------statistics about each car----------------")
    # loops through all cars in setOfCars ( sedOfCars contains the cl number of
        each car in the completed database )
    for i in setOfCars:
        # calcNumOfRentedDays will return a list the first element in it will
            be the number of days the car was rented ,
        # and the second element is the revenue made by the car .
        l = calcNumOfRentedAndRevenueDays(list,i)

        # printing the statistics of the car
        print("--------------------------------------------------------")
        print("Number of Days car \""+ i + "\" was rented : " + str(l[0]) )
        print("Revenue made by renting the car : " + str(l[1]))
        # the average price per day for renting a car = Revenue made by the car
            / number of rented days
        print("Average price per day for renting the car : " +
            str(float(l[1])/l[0]))
        print("--------------------------------------------------------\n")

# this function take the list that contains the completed database , and the cl
    of the car we want to search about
# then it calculates how many days the given car was rented , and the revenue
    made by this car .
def calcNumOfRentedAndRevenueDays(list,car):

    num = 0
    revenue = 0
    # loops through the list
    for i in list:

        if(i.getCl() == car):
            # convert the dates from this format (dayNum nonthName yearNum) to
                this format (yearNum-monthNum-dayNum)
            sd = datetime.datetime.strptime(i.getSd(), "%d %B %Y")
            ed = datetime.datetime.strptime(i.getEd(), "%d %B %Y")
            # (ed-sd).days will return how many days are between this two dates
            days = (ed-sd).days
            num += days
            revenue += int(i.getRb())

    # return a list contains the number of days the car was rented , and the
        revenue made by renting this car
    return [num,revenue]
```