Python 6

Total Marks

35.0

Pass Marks

20.0

Marks Obtained

NA

Status

NA

Report

---

Start Time :  18 Dec 2020 09:13

|

End Time :  31 Mar 2023 00:00

Question 1 :

During the COVID19 pandemic, the status of beds availability is to be tracked  
  
Create a class City with the below attributes:  
city\_id of type Number  
state\_name of type String  
city\_name of type String  
covidbeds of type Number  
avlblcovbeds of type Number  
ventilbeds of type Number  
avlblventilbeds of type Number  
  
  
Attribute description:  
city\_id represents Unique ID for the city  
state\_name represents the state name  
city\_name represents the city name  
  
covidbeds represents the total covid beds in the city  
avlblcovbeds represents the total available covid beds in the city  
ventilbeds represents the total ventilator beds in the city  
avlblventilbeds represents the total available ventilator beds in the city  
  
  
Create the \_\_init\_\_ method which takes all parameters in the above sequence. The method should set the value of attributes to parameter values .  
  
Create another class CovBedsAnalysis with the below attributes:  
analysis\_name of type String  
city\_list of type List having city objects  
  
Create the \_\_init\_\_ method which takes all parameters in the above sequence. The method should set the value of attributes to parameter values inside the method.  
  
Create another method inside the class with the name get\_StateWiseAvlblBedStats  
  
This method is used to find the state wise available covid beds and available ventilator beds and returns a list of tuples with State name,total available covid beds and total available ventilator beds for each state, sorted by state name.  
  
  
Note: A state contains multiple cities. Total number of available beds for a respective category (covid or ventilator beds) in a state is the sum of the available beds of all the cities in that state for the respective category(covid or ventilator). Refer testcase output for more clarity.  
  
  
Create another method with the name get\_CiitesWithMoreThanAvgOccupiedbeds, which takes state as argument and returns the dictionary with city as the key and tuple of occupied covid beds and occupied ventilator beds as value, where number of covid beds occupied and ventilator beds occupied are more than the state average for the respective category of beds .  
  
  
i.e. the City(cities) in the given state to be recorded/resulted( with the data mentioned), should satisfy the below conditions:  
  
Whose occupied covid beds count is more than the "average of Occupied covid beds of all the cities of the given state" and the respective City should also contain the Occupied ventilator beds count more than the "average of occupied ventilator beds of all cities of the given state".  
  
For more clarity , please refer the Sample test case Input and Output in below section  
  
If no city is found with the occupied beds more than state average as mentioned above, then return None and display ‘No city available' (Without quotes) in main function.  
  
  
Please note that the search operations(if any as per the requirement ..) should be case insensitive.  
  
Instructions to write main function:  
  
Instructions to write main section of the code  
a. You would require to write the main section completely, hence please follow the below instructions for the same.  
b. You would require to write the main program which is inline to the "sample input description section" mentioned below and to read the data in the same sequence.  
c. Create the respective objects(City and CovBedsAnalysis ) with the given sequence of arguments to fulfill the \_\_init\_\_ method requirement defined in the respective classes referring to the below instructions.  
  
i. Create a City object after reading the data related to it and add the object to the list of city objects which will be provided to the CovBedsAnalysis object while creation.  
This point repeats for the number of city objects(considered in the first line of input data) .  
  
ii. Create CovBedsAnalysis object by passing the CovBedsAnalysis name(you can hard-code any name you want) and List of city objects ( created as mentioned in above  
point# c.i ) as the arguments.  
d. Take a string value as input depicting the state which is passed to the get\_CiitesWithMoreThanAvgOccupiedbeds  
e. Call the method get\_StateWiseAvlblBedStats mentioned above from the main section.  
f. Display the State,total available covid beds and total available ventilator beds received from the method, with a single space in between as shown in sample testcase output,  
g. Call the method get\_CiitesWithMoreThanAvgOccupiedbeds mentioned above from the main section  
h. Display the city name, occupied covid beds and occupied ventilator beds with a single space in between as shown in the sample testcase output.  
I. If None is returned by the method get\_CiitesWithMoreThanAvgOccupiedbeds, display the message ‘No city available' (excluding the quotes).  
  
  
You can use/refer the below given sample input and output to verify your solution using ' Test against Custom Input ' option in Hackerrank.  
  
  
Input format for Custom Testing:  
  
a.The 1st input taken in the main section is the number of city objects to be added to the list of cities.  
b.The next set of inputs are the values for the attribtes: city\_id,state\_name,city\_name, covidbeds,avlblcovbeds,ventilbeds,avlblventilbeds respectively for each city taken one after other and is repeated for number of objects given in the first line of testcase input  
c.The last line of input refers the state name which is passed to the function: get\_CiitesWithMoreThanAvgOccupiedbeds  
  
  
Sample Testcase1:  
  
Input:  
5  
101  
Delhi  
Delhi  
100000  
20000  
5000  
2000  
102  
Telangana  
Hyderabad  
200000  
40000  
1000  
500  
103  
Telangana  
Warangal  
100000  
30000  
2000  
1000  
104  
AndhraPradesh  
Vijayawada  
800000  
300000  
30000  
2500  
105  
AndhraPradesh  
Vizag  
500000  
100000  
6000  
1000  
AndhraPradesh  
  
  
Output:  
  
AndhraPradesh 400000 3500  
Delhi 20000 2000  
Telangana 70000 1500  
Vijayawada 500000 27500  
  
  
  
Sample TestCase2:  
  
Input:  
  
6  
101  
Delhi  
Delhi  
100000  
20000  
8000  
3000  
102  
Telangana  
Hyderabad  
200000  
100000  
12000  
6000  
103  
Telangana  
Warangal  
100000  
50000  
1000  
500  
104  
AndhraPradesh  
Vijayawada  
800000  
400000  
7500  
3750  
105  
AndhraPradesh  
Vizag  
500000  
100000  
11000  
8500  
106  
Maharashtra  
Mumbai  
1000000  
0  
12500  
0  
maharashtra  
  
  
Output:  
  
AndhraPradesh 500000 12250  
Delhi 20000 3000  
Maharashtra 0 0  
Telangana 150000 6500  
No city available

class City:

def \_\_init\_\_(self,city\_id,state\_name,city\_name,covidbeds,avlblcovbeds,ventilbeds,avlblventilbeds):

self.city\_id = city\_id # represents Unique ID for the city

self.state\_name = state\_name # represents the state name

self.city\_name = city\_name # represents the city name

self.covidbeds = covidbeds # represents the total covid beds in the city

self.avlblcovbeds = avlblcovbeds # represents the total available covid beds in the city

self.ventilbeds = ventilbeds # represents the total ventilator beds in the city

self.avlblventilbeds = avlblventilbeds # represents the total available ventilator beds in the city

def \_\_repr\_\_(self):

return self.city\_name

class CovBedsAnalysis:

def \_\_init\_\_(self,analysis\_name,city\_list):

self.analysis\_name = analysis\_name

self.city\_list = city\_list

def get\_StateWiseAvlblBedStats(self):

states = {}

for city in self.city\_list:

lower\_state\_list = [s.lower() for s in list(states.keys())]

if city.state\_name.lower() in lower\_state\_list:

state = states[city.state\_name]

state[0] += city.covidbeds

state[1] += city.ventilbeds

else:

states[city.state\_name] = [city.covidbeds, city.ventilbeds]

res = []

for k,v in states.items():

res.append((k, v[0], v[1]))

res.sort(key=lambda x: x[0])

return res

def get\_CiitesWithMoreThanAvgOccupiedbeds(self, state):

states = {}

for city in self.city\_list:

if city.state\_name in states.keys():

states[city.state\_name].append(city)

else:

states[city.state\_name] = [city]

state = [s for s in states.keys() if s.lower()==state.lower()][0]

city\_list = states[state]

avg\_occcovbeds=0

avg\_occvntbeds=0

for city in city\_list:

avg\_occcovbeds += city.covidbeds - city.avlblcovbeds

avg\_occvntbeds += city.ventilbeds - city.avlblventilbeds

avg\_occcovbeds /= len(city\_list)

avg\_occvntbeds /= len(city\_list)

for city in city\_list:

occcovbeds = city.covidbeds - city.avlblcovbeds

occvntbeds = city.ventilbeds - city.avlblventilbeds

if occcovbeds > avg\_occcovbeds and occvntbeds > avg\_occvntbeds:

res[city] = (occcovbeds, occvntbeds)

if len(res) == 0:

print('No city available')

return None

return res

if \_\_name\_\_ == "\_\_main\_\_":

# inputs

n = int(input())

city\_list = []

for \_ in range(n):

city\_id = int(input())

state\_name = input()

city\_name = input()

covidbeds = int(input())

avlblcovbeds = int(input())

ventilbeds = int(input())

avlblventilbeds = int(input())

city\_list.append(City(city\_id,state\_name,city\_name,covidbeds,avlblcovbeds,ventilbeds,avlblventilbeds))

state\_name = input()

# logic

covBedsAnalysis = CovBedsAnalysis("Test Analysis", city\_list)

res = covBedsAnalysis.get\_StateWiseAvlblBedStats()

for s in res:

print(\*s)

res = covBedsAnalysis.get\_CiitesWithMoreThanAvgOccupiedbeds(state\_name)

print(res)