Project - 1 Report

Nigar Sultana

ID:1221371489

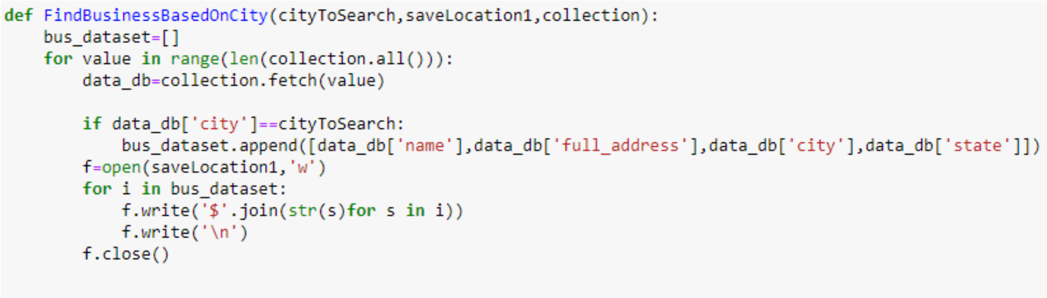
Reflection

At the outset of the project, my initial step was to understand the pseudo-code. After that, I found that UnQlite functions as both a Key-Value store and Document-store database, shedding light on the underlying data storage mechanism. With the additional investigation, I found an improved approach to iterate through all the key-value pairs within the collection. These findings were very important in the development of the project's functions.

For the completion of the project, the following function was implemented.

- FindBusinessBasedOnCity(cityToSearch, saveLocation1, collection): In this function, a list called bus_dataset was declared to store the name, full address, city, and state of cities specified in the cityToSearch input parameter. I performed a "for loop" to iterate through the stored list of key-value in the collection. I placed the condition that if the city from the iteration is equal to the cityToSearch parameter, the name, full address, city, and state corresponding to the town are appended to bus_dataset. Finally, I iterated over the bus dataset and saved the entries in saveLocation1 as specified.

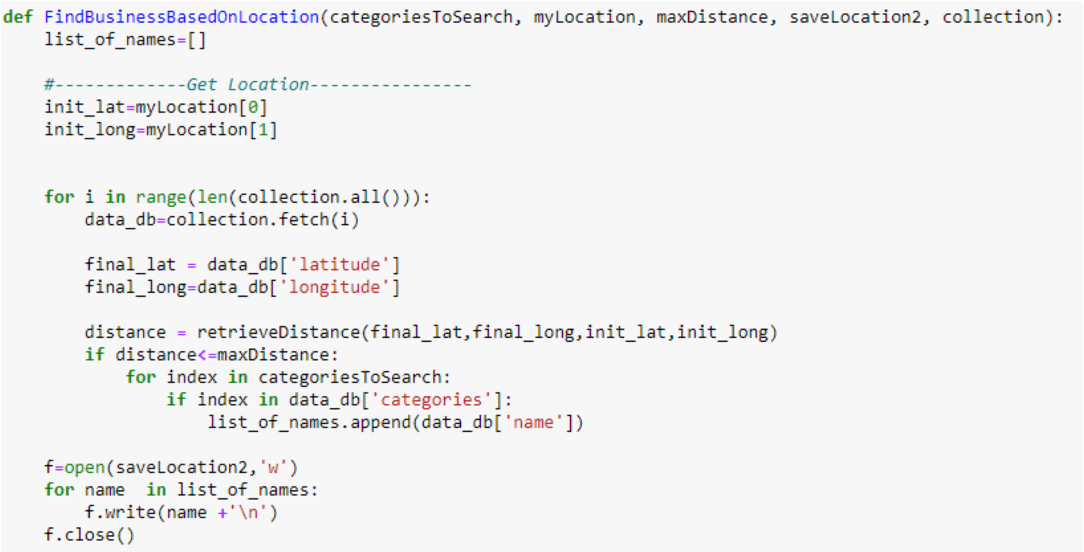
  The figure below shows the screenshot of the FindBusinessBasedOnCity function implementation.



```
def FindBusinessBasedOnCity(cityToSearch,saveLocation1,collection):
    bus_dataset=[]
    for value in range(len(collection.all())):
        data_db=collection.fetch(value)

        if data_db['city']==cityToSearch:
            bus_dataset.append([data_db['name'],data_db['full_address'],data_db['city'],data_db['state']])
        f=open(saveLocation1,'w')
        for i in bus_dataset:
            f.write('$'.join(str(s)for s in i))
            f.write('\n')
        f.close()
```

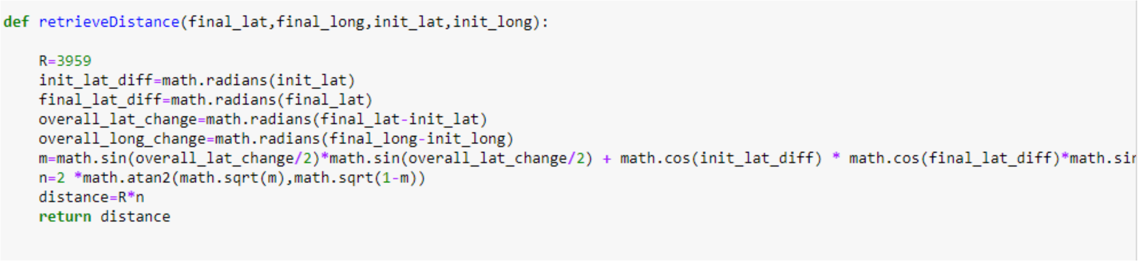
- FindBusinessBasedOnLocation(categoriesToSearch,myLocation,maxDistance,saveLocation2, collection): This function takes in five input parameters, namely categoriesToSearch, myLocation, maxDistance, saveLocation2, and collection. In the first part of my code, a list called list_of_names was declared, and the function of this list is to store the names of cities within a certain distance specified by the maxDistance parameter. Next, I am retrieving the longitude and latitude of the current location and saving it to a variable called init_long and init_lat. Then I looped through the list of items contained in the collection and retrieved the longitude and latitude. After this, I am calling a distance function called retrieveDistance to calculate the distance from given latitudes and longitudes. If the distance retrieved is less than or equal to the max distance, the city's name in data_db is appended to the list_of_names. Finally, I am saving the names containing the list_of_names to saveLoacation2.

The figure below shows the screenshot of the FindBusinessBasedOnLocation function implementation.



```
def FindBusinessBasedOnLocation(categoriesToSearch, myLocation, maxDistance, saveLocation2, collection):
    list_of_names=[]

    #-------------Get Location----------------
    init_lat=myLocation[0]
    init_long=myLocation[1]


    for i in range(len(collection.all())):
        data_db=collection.fetch(i)

        final_lat = data_db['latitude']
        final_long=data_db['longitude']

        distance = retrieveDistance(final_lat,final_long,init_lat,init_long)
        if distance<=maxDistance:
            for index in categoriesToSearch:
                if index in data_db['categories']:
                    list_of_names.append(data_db['name'])

    f=open(saveLocation2,'w')
    for name  in list_of_names:
        f.write(name +'\n')
    f.close()
```

- retrieveDistance(final_lat,final_long,init_lat,init_long): The retrieveDistance function is a distance function that takes pairs of latitude and longitude and calculates the distance using the formula d=r*c. The figure below shows the screenshot of the retrieveDistance function implementation.



```
def retrieveDistance(final_lat,final_long,init_lat,init_long):

    R=3959
    init_lat_diff=math.radians(init_lat)
    final_lat_diff=math.radians(final_lat)
    overall_lat_change=math.radians(final_lat-init_lat)
    overall_long_change=math.radians(final_long-init_long)
    m=math.sin(overall_lat_change/2)*math.sin(overall_lat_change/2) + math.cos(init_lat_diff) * math.cos(final_lat_diff)*math.sin
    n=2 *math.atan2(math.sqrt(m),math.sqrt(1-m))
    distance=R*n
    return distance
```

Lessons **Learned:**

- Even though I had some prior experience in Jupyter Notebook, using this tool for this project gave me invaluable experience in using it for cloud environments and simultaneously executing multiple terminals.

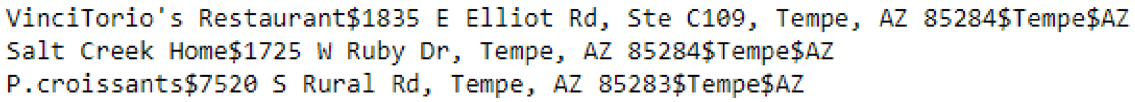
- By reviewing the course materials, I acquired an in-depth understanding of UnQlite, realizing its enhanced versatility in executing queries and retrieving information.

- I recognized the significance of embracing novel concepts of unstructured database management systems and cloud-based programming, as they can enhance programming skills and broaden the range of application scenarios.

Output:

Output_city.txt output for the test case(3rd cell) for the FindBusinessBasedOnCity function:

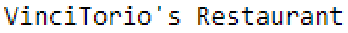
FindBusinessBasedOnCity('Tempe', 'output_city.txt', data)



```
VinciTorio's Restaurant$1835 E Elliot Rd, Ste C109, Tempe, AZ 85284$Tempe$AZ
Salt Creek Home$1725 W Ruby Dr, Tempe, AZ 85284$Tempe$AZ
P.croissants$7520 S Rural Rd, Tempe, AZ 85283$Tempe$AZ
```

Output_loc.txt output for the second test case(4th cell) for the FindBusinessOnLocation function:

FindBusinessBasedOnLocation (['Buffets'], [33.3482589, -111.9088346], 10, 'output_loc.txt', data)



```
VinciTorio's Restaurant
```

Result:

The FindBusinessBasedOnCity function in my code contains the following:

- Filtering the result to exclude entries unrelated to the specified city provided as a parameter.
- Combining all the Dollar values extracted and decoded from UnQlite, part of the filtered values, into a single string.
- Finally, my code generates output with accurate numerical values.

Similarly, the FindBusinessBasedOnLocation function in my code covers the following aspects:

- Effectively storing the correct latitude and longitude values for the given locations in their respective variables.
- Accurately calculating the numerical distance for each restaurant within the location provided.
- Including only the restaurants with the maximum calculated distance in the resulting output.
- Preventing the repetition of a restaurant in the results by utilizing an "if statement" to compare the current and previous latitude and longitude values.
- Avoiding the duplication of writing the same restaurant into the text file.