

Project 1 Individual Report

1. Reflection

There are two sub-problems in this project:

Sub-Problem A: FindBusinessBasedOnCity(cityToSearch, saveLocation1, collection)

For this function, we were given data collection, an output file 'saveLocation1,' and a city name 'cityToSearch' as input arguments. Using these, I had to do the following:

1. Find businesses within that city.
2. Append the result to the output file in the mentioned format.

I approached this sub-problem by including 2 main conditions, i.e., type is 'business' and city is cityToSearch. After this, I opened the output file in w+ mode and wrote in it in the given format.

Code snippet of Sub-Problem A:

```
def FindBusinessBasedOnCity(cityToSearch, saveLocation1, collection):
    all_business=[]
    for c in collection.all():
        if c['type']=='business' and c['city']==cityToSearch:
            all_business.append(c)

    with open(saveLocation1, 'w+') as file:
        for c in all_business:
            file.write(c['name']+'$'+c['full_address'].replace("\n", ", ")+'$'+c['city']+'$'+c['state']+'\n')
```

Sub-Problem B: FindBusinessBasedOnLocation(categoriesToSearch, myLocation, maxDistance, saveLocation2, collection)

For the 2nd function, we had 5 input arguments: a kind of businesses to search for, current location of the user as 'myLocation,' a threshold distance, an output file, and the data collection. With the help of these, I had to work on the following:

1. Find businesses from the collection which includes the input categories.
2. Check if the distance between the business location and the user location is less than the threshold distance.
3. Append the name of the business to the output file.

To solve this sub-problem, firstly, I checked if the type of business exists in the input categories. Then, I calculated the distance according to the given distance formula and if it was less than the threshold distance, wrote the business name on the output file with w+ mode.

Code snippet of Sub-Problem B:

1. Calculating Distance:

```
def distance(lat2, long2, lat1, long1):
    #radius of earth
    R=3959
    phi1=radians(lat1)
    phi2=radians(lat2)
    delphi=radians(lat2-lat1)
    dellam=radians(long2-long1)
    a=sin(delphi/2)*sin(delphi/2) + cos(phi1)*cos(phi2)*sin(dellam/2)*sin(dellam/2)
    c=2 * atan2(sqrt(a), sqrt(1 - a))
    return R*c
```

2. FindBusinessBasedOnLocation:

```
def FindBusinessBasedOnLocation(categoriesToSearch, myLocation, maxDistance, saveLocation2, collection):
    all_business=[]
    for c in collection.all():
        for cat in c['categories']:
            if cat in categoriesToSearch and distance(c['latitude'],c['longitude'],myLocation[0],myLocation[1])<=maxDistance:
                all_business.append(c)
                break

    with open(saveLocation2, 'w+') as file:
        for c in all_business:
            file.write(c['name']+'\n')
```

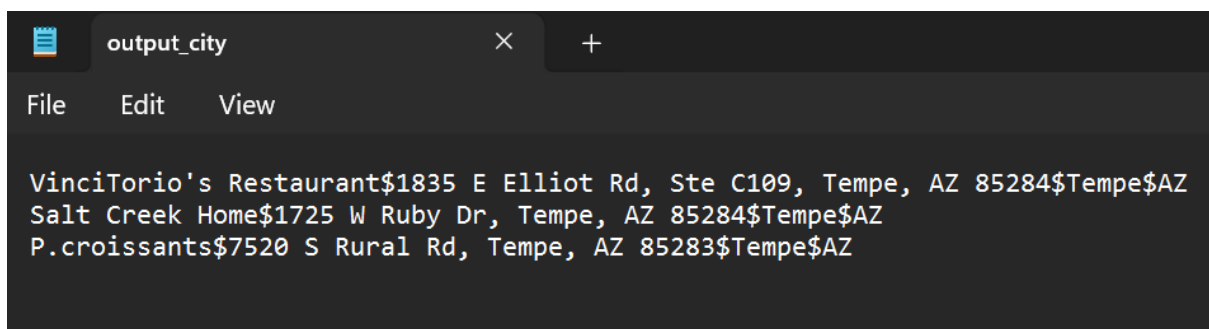
2. Lessons Learned

This project taught me how to query with a NoSQL type of database. This kind of querying is way different than SQL. I learned how to use data from a database using the Python library unqlite. This project also taught me how to use Jupyter notebooks.

I have learned how to calculate distances given 2 spatial data points, i.e., their longitudes and latitudes. This project gave me an idea of how big corporations could go about calculating the shortest distance between a given source and destination. It also provided insight into how these corporations suggest users in Google and Apple Maps regarding the nearest restaurants, gas stations, hotels, supermarkets, etc.

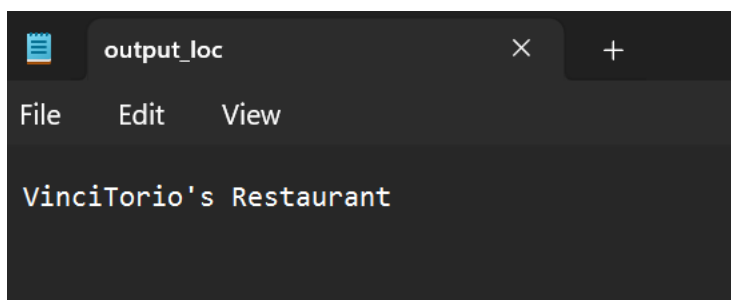
3. Output

Sub-Problem A:



```
output_city
File Edit View
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
```

Sub-Problem B:



```
output_loc
File Edit View
VinciTorio's Restaurant
```

4. Results

Sub-Problem A:

```
[4]: true_results = ["VinciTorio's Restaurant$1835 E Elliot Rd, Ste C109, Tempe, AZ 85284$Tempe$AZ", "P.croissants$7520 S Rural Rd, Tempe, AZ 85283$Tempe$AZ",  
  
try:  
    FindBusinessBasedOnCity('Tempe', 'output_city.txt', data)  
except NameError as e:  
    print ('The FindBusinessBasedOnCity function is not defined! You must run the cell containing the function before running this evaluation cell.')  
except TypeError as e:  
    print ("The FindBusinessBasedOnCity function is supposed to accept three arguments. Yours does not!")  
  
try:  
    opf = open('output_city.txt', 'r')  
except FileNotFoundError as e:  
    print ("The FindBusinessBasedOnCity function does not write data to the correct location.")  
  
lines = opf.readlines()  
if len(lines) != 3:  
    print ("The FindBusinessBasedOnCity function does not find the correct number of results, should be 3.")  
  
lines = [line.strip() for line in lines]  
if sorted(lines) == sorted(true_results):  
    print ("Correct! You FindBusinessByCity function passes these test cases. This does not cover all possible test edge cases, however, so make sure tha
```

Correct! You FindBusinessByCity function passes these test cases. This does not cover all possible test edge cases, however, so make sure that your function covers them before submitting!

Sub-Problem B:

```
[5]: true_results = ["VinciTorio's Restaurant"]  
  
try:  
    FindBusinessBasedOnLocation(['Buffets'], [33.3482589, -111.9088346], 10, 'output_loc.txt', data)  
except NameError as e:  
    print ('The FindBusinessBasedOnLocation function is not defined! You must run the cell containing the function before running this evaluation cell.')  
except TypeError as e:  
    print ("The FindBusinessBasedOnLocation function is supposed to accept five arguments. Yours does not!")  
  
try:  
    opf = open('output_loc.txt', 'r')  
except FileNotFoundError as e:  
    print ("The FindBusinessBasedOnLocation function does not write data to the correct location.")  
  
lines = opf.readlines()  
if len(lines) != 1:  
    print ("The FindBusinessBasedOnLocation function does not find the correct number of results, should be only 1.")  
  
if lines[0].strip() == true_results[0]:  
    print ("Correct! Your FindBusinessBasedOnLocation function passes these test cases. This does not cover all possible edge cases, so make sure your fu
```

Correct! Your FindBusinessBasedOnLocation function passes these test cases. This does not cover all possible edge cases, so make sure your function does before submitting.