Project Report 1 - NoSQL

Student Name: Mehran Tajbakhsh

Email: mtajbakh@asu.edu

Submission Date: 13th Nov,2022

Class Name and Term: CSE511 Fall 2022

Project Overview

In this project, I did transformations and actions on a NoSQL data records using Python. In order to do this job I used a NoSQL database (sample.db) that allowed me to find businesses in the provided city or location. I implemented my code in the Jupyter Notebook and tested my code with several test cases that were given to me.

I. Reflection

In this project I was assigned with implementing two functions: FindBusinessBasedOnCity and FindBusinessBasedOnLocation. The function FindBusinnessBasedOnCity gets three parameters: cityToSearch, saveLocation1, and collection.
This function searches in the input data that's given in the 'collection' to find the city that's given in the 'cityToSearch' and printout the result(s) in the 'saveLocation1'.
The function FindBusinessBasedOnTheLocation gets 5 parameters as follow:

- categoriesToSearch - specify the categories of the businesses
- myLocation - My current location as a set of latitude and longitude
- maxDistance – Specify maximum distance to search businesses from the current location
- saveLocation2 – Name of the output file
- collection – Input database file

Then search for all the businesses in the radius of maxDistance from the current location based on the categories that specified. This Function uses the haversin algorithm to compute the greate-circle distance between two locations on a sphere given their Longitude and Latitude.

In the following images I show my code with details comment that explain how the codes work:

```
# Graded Cell, PartID: o1flK

from unqlite import UnQLite
import math   # Built-in module in the Python 3 standard library that provides standard mathematical
              # constants and functions that we used in DistanceFunction to implement haversin algorithm

db = UnQLite('sample.db')    # sample data that's given to us
data = db.collection('data') # variable data is a handler to access to input data

def DistanceFunction(latitude2, longitude2, latitude1, longitude1):
    # We use haversin algorithm to find the great-circle distance between two points on a sphere given
    # their longitudes and latitudes.
    # haversin(d/r) = haversin(φ1 - φ2) + cos(φ1)*cos(φ2)haversin(λ2-λ1)
    # a = sin²(ΔlatDifference/2) + cos(latitude1).cos(latitude2).sin²(ΔlonDifference/2)
    # c = 2*atan2(√a, √(1-a))
    # d = R*c
    R=3959 # Radiuds of earth (miles)
    latDifference1=math.radians(latitude1) # φ1
    latDifference2=math.radians(latitude2) # φ2
    delta_latDifference=math.radians(latDifference2-latDifference1) # Δφ
    delta_longitude=math.radians(longitude2-longitude1) # Δλ
    a = math.sin(delta_latDifference/2) ** 2 + math.cos(latDifference1)*math.cos(latDifference2) * math.sin(delta_longitude)**2
    c=2*math.atan2(math.sqrt(a),math.sqrt(1-a))
    d=R*c
    return d
```

```
def FindBusinessBasedOnCity(cityToSearch,saveLocation1,collection):
    fout=open(saveLocation1,'w')  # Open output file for writing and use saveLocation1 variable as a handler to access to it
    for item in collection.all(): # Search in input database
        if item['city'].lower() == cityToSearch.lower():  # Find records with the matching city
            fout.write(item['name']+'$'+ item['full_address']+'$' + item['city']+'$' + item['state']+'\n')
            # Print every matched record in a new line in the format that's given to us
    fout.close()
```

```
def FindBusinessBasedOnLocation(categoriesToSearch,myLocation,maxDistance, saveLocation2, collection):
    # myLocation is a list variable and we use to pass our current location as latitude and longitude.
    latitude1 = myLocation[0]  # Access to the first item in the list list and save that  into the latitude1 variable.
    longitude1 = myLocation[1]  # Access to the second item in the list and save that into the longitude1 variable.
    search_cat = set([i.lower() for i in categoriesToSearch])   # create a set of categories that's given as input
                                                                # to search and name it as search_cat
    fout = open(saveLocation2,'w')  # Open output file for writing and use saveLocation1 variable as a handler to access to it

    for item in collection.all():   # Search in input database
        latitude2 = item['latitude']   # Extract value of the latitude key in the current record into the variable latitude2
        longitude2 = item['longitude']  # Extract value of the longitude key in the current record into the variable longitude2
        cat_list = set([i.lower() for i in item['categories']])  # Extract value of the categories key in the current
                                                                 # record and create a set with name cat_list
        # find similarity between search_cat set(categories that's given to us) and cat_list set(categories key in the database)
        # and compute the distance between our current location and the target business. if the category of the business matched
        # and it's distance less that the maxDistance that's given to us as input then write name of the business in a new line
        # at the output file.
        if (len(search_cat.intersection(cat_list)) > 0 and  \
            DistanceFunction(latitude2, longitude2, latitude1, longitude1)<=maxDistance):
            fout.write(item['name']+'\n') # Write name of the business in a newline
    fout.close()
```

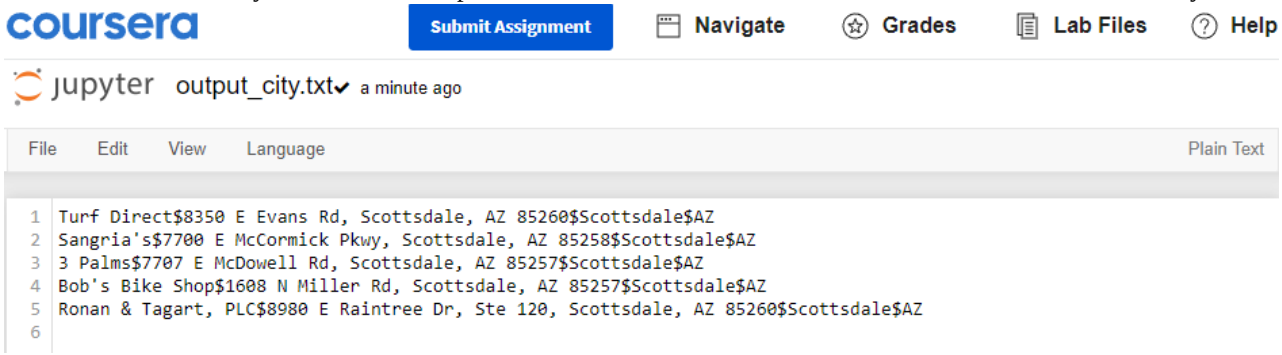
II. LESSON LEARNED

I can summarize the lessons that I learned in this project as follows:

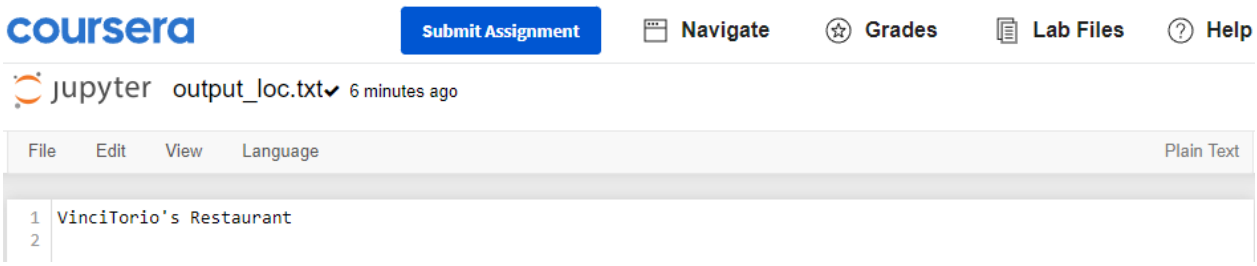
- How to use unqlite library to process nosql database in python
- How to access and process the spatial data in the NoSQL database with Python
- How to compute greate-circle distance between two locations on the map (earth) based on the latitude/longitude coordinates using Haversin algorithm in Python

III. OUTPUT

Below Figure shows the screenshot of the content the output_city.txt file that I used to save output of the FindBusinessBaseOnTheCity function. The output file shows that there are 5 businesses located at the Scottsdale city.



Below Figure shows the screenshot of the content the output_loc.txt file that I used to save output of the FindBusinessBaseOnTheLocation function. The output file shows that there is 1 buffet restaurant found in the 10 miles from our current location.



IV. RESULT

The output of the function FindBusinessBasedOnTheCity when I pass the test case:

```
true_results =['3 Palms$7707 E McDowell Rd, Scottsdale, AZ 85257$Scottsdale$AZ', "Bob's Bike Shop$1608

try:
    FindBusinessBasedOnCity('Scottsdale', 'output_city.txt', data)
except NameError as e:
    print ('The FindBusinessBasedOnCity function is not defined! You must run the cell containing the f
except TypeError as e:
    print(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) != 5:
    print ("The FindBusinessBasedOnCity function does not find the correct number of results, should be
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 p
```

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

The output message shows that my program passed the test case successfully. The FindBusinessBasedOnTheCity function filters all records based on the target city (Scottsdale) and prints the results in the output file (output_city.txt). I followed the syntax that were given to me as true_results and save the output in the output_city.txt file with following format:
(Name of business +$+Full Address+$+City+$+State)
The output of the function FindBusinessBasedOnTheLocation when I pass the test case:

```
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 t
except TypeError as e:
    print ("The FindBusinessBasedOnLocation function is supposed to accept five arguments. Yours does r

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, shoul

if lines[0].strip() == true_results[0]:
    print ("Correct! Your FindBusinessBasedOnLocation function passes these test cases. This does not c
```

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

The output message shows that my program passed the test case successfully. The FindBusinessBasedOnTheLocation function filters all records based on the type of business and distance from the current location that specified as part of the input and prints the results in the output file (output_loc.txt). I followed the syntax that were given to me as true_results and save the output in the output_loc.txt file with following format:
(Name of business)

TECHNOLOGY REQUIREMENTS

- Python 3.5.x version
- cython
- unqlite

REFERENCES

1. http://www.movable-type.co.uk/scripts/latlong.html