

CSE 511: DATA PROCESSING AT SCALE

PROJECT 1: INDIVIDUAL REPORT

(1) Reflection:

I used the google colab with jupyter notebook environment to run this project while importing all the necessary packages. I uploaded the necessary documents and then I started working on implementing functions. I imported the provided file and ran to check if everything else is working correctly or not. So, now the environment and code are ready to implement FindBusinessBasedOnCity, and FindBusinessBasedOnLocation.

The Distance Function :- This function calculates the distance between two geographical locations specified by their latitude and longitude using the Haversine formula. It takes 4 inputs **startLatitude**, **startLongitude** which are coordinates of the starting point **endLatitude**, **endLongitude** which are coordinates of the ending point. Haversine Formula is used over here. The formula calculates distances between two points on a sphere using their longitudes and latitudes. It is particularly useful in navigation.

In this method, since it is already given, I put Earth's radius as 3959 miles. The function converts latitude and longitude from degrees to radians. It calculates the changes in latitude (**deltaLat**) and longitude (**deltaLon**). Then I calculated the distance as the product of the Earth's radius and the central angle which is defined as per the Haversine Formula.

FindBusinessBasedOnCity:- This function is used to search for businesses within a certain distance from a user's location, based on specified categories, and I have also used this function to write the results to a file. This function takes 3 inputs, The first one is **targetCity**, this input is for the city for which businesses needs to be searched. The second input for this function is **outputFilePath1**, I have used this input to get the Path of the file where results will be stored. The last input is **dataCollection** this is a collection containing business records. The use of this function is to find a business based on the matched city name. It iterates through each record in dataCollection. If the city of a business matches targetCity, details of the business are appended


to cityBusinesses which is a list. I have concatenated business details into a string with "\$" as a separator as mentioned in the format. The resulting list of businesses is written to the specified file, each on a new line.

FindBusinessBasedOnLocation:- This function searches for businesses within a certain distance from a user's location, based on specified categories, and writes the results to a file. I have specified 5 inputs for this function. **searchCategories** this input is to find the list of categories to filter businesses. I have used this **userLocation** input to get the user's current latitude and longitude. **maxAllowedDistance:** This parameter is to find Maximum distance from the user's location. And lastly I have used **outputFilePath2** to save the search results. **dataCollection** is the same as mentioned in the previous function which is a collection containing business records. This function iterates through each business in dataCollection. And For each business, it checks if any of the searchCategories are in the business's categories. If a matching category is found, it calculates the distance to the business from the user's location using DistanceFunction. If this distance is within maxAllowedDistance, the business's name is added to nearbyBusinesses. It ensures that business names are unique and writes these names to the specified file, each on a new line.

So, I have implemented the three functions, FindBusinessBasedOnCity, FindBusinessBasedOnLocation and DistanceFunction and I have run on the sample database to get the output.

Here is the SS of code for the three functions:

(i) DistanceFunction:



```
# Graded Cell, PartID: o1fLK

import math
def DistanceFunction(startLatitude, startLongitude, endLatitude, endLongitude):
    earthRadius = 3959
    radStartLat = math.radians(startLatitude)
    radEndLat = math.radians(endLatitude)
    deltaLat = math.radians((endLatitude - startLatitude))
    deltaLon = math.radians((endLongitude - startLongitude))
    haversineFormula = math.sin(deltaLat / 2) * math.sin(deltaLat / 2) + math.cos(radStartLat) * math.cos(radEndLat) * math.sin(deltaLon / 2)
    centralAngle = 2 * math.atan2(math.sqrt(haversineFormula), math.sqrt(1 - haversineFormula))
    distance = earthRadius * centralAngle
    return distance
```

(ii) FindBusinessBasedOnCity function:

```
def FindBusinessBasedOnCity(targetCity, outputFile1, dataCollection):
    cityBusinesses = []

    for businessRecord in dataCollection:
        if(businessRecord['city'] == targetCity):
            cityBusinesses.append(businessRecord['name'] + "$" + businessRecord['full_address'] + "$" + businessRecord['city'] + "$" + businessRecord['latitude'] + "$" + businessRecord['longitude'])

    fileWriter = open(outputFile1, "a")
    for businessInfo in cityBusinesses:
        fileWriter.write(businessInfo + "\n")
    fileWriter.close()
```

(iii) FindBusinessBasedOnLocation function:

```
def FindBusinessBasedOnLocation(searchCategories, userLocation, maxAllowedDistance, outputFile2, dataCollection):
    nearbyBusinesses = []
    for businessRecord in dataCollection:
        for category in searchCategories:
            if (category in businessRecord['categories']):
                calculatedDistance = DistanceFunction(userLocation[0], userLocation[1], businessRecord['latitude'], businessRecord['longitude'])
                if(calculatedDistance <= maxAllowedDistance):
                    nearbyBusinesses.append(businessRecord['name'])
    uniqueBusinesses = list(set(nearbyBusinesses))
    fileWriter = open(outputFile2, "a")
    for businessName in uniqueBusinesses:
        fileWriter.write(businessName + "\n")
    fileWriter.close()
```

(2) Lessons Learned:

Database Query

I gained insights into diverse methods of data storage and the techniques for processing, querying, and managing different data types.

Geographical Calculation

I learned skills in computing the distance between two geographical coordinates, enhancing my understanding of spatial data.

Problem Solving

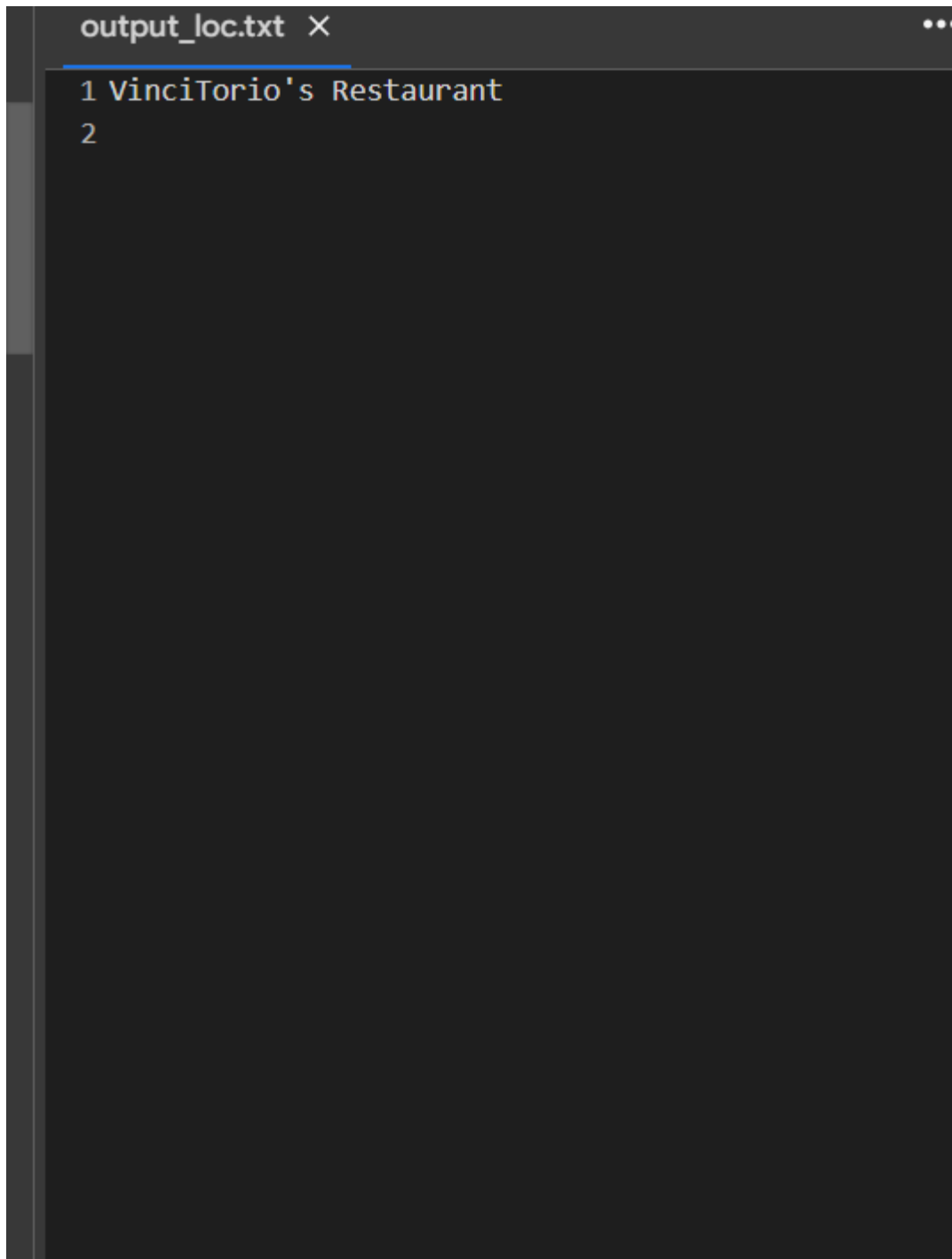
I also Developed proficiency in handling and modifying data within UnQLite databases, showcasing my ability to address data management challenges effectively.

Jupyter Notebook

I gained more proficiency in my Python programming skills, particularly in utilizing Jupyter Notebooks, a tool that's very good for writing, testing, and sharing code in an interactive environment.

(3) Output:

The screenshots of output files generated by the provided code.



A screenshot of a text editor window with a dark background. The title bar at the top shows 'output_loc.txt' followed by a close button icon. The editor area contains two lines of text: '1 VinciTorio's Restaurant' on the first line and '2' on the second line. The text is in a light-colored, monospaced font.

```
output_loc.txt X
1 VinciTorio's Restaurant
2
```

```

output_city.txt X
1 VinciTorio's Restaurant$1835 E Elliot Rd, Ste C109, Tempe, AZ 85284$Tempe$AZ
2 Salt Creek Home$1725 W Ruby Dr, Tempe, AZ 85284$Tempe$AZ
3 P.croissants$7520 S Rural Rd, Tempe, AZ 85283$Tempe$AZ
4

```

(4) Result:

The screenshots of output from the code of given test cases.

```

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 m

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

[6] # @title
true_results = ["VinciTorio's Restaurant$1835 E Elliot Rd, Ste C109, Tempe, AZ 85284$Tempe$AZ", "P.croissants$7520 S Rural Rd, Tempe, AZ :

```

```

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, :

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!

```