Project-1: NoSQL Report

Reflection:

A. FindBusinessBasedOnCity():

As the name of the function suggests, the function task is to search the dataset to find all businesses present in the given city. This function will take three parameters, those are cityToSearch, saveLocation1, and collection.

Full function logic is attached below.

```
2
3  def FindBusinessBasedOnCity(cityToSearch, saveLocation1, collection):
4      #print(type(collection.fetch(0)))
5
6      result = collection.filter(lambda obj: cityToSearch in obj['full_address'].decode())
7
8      output= open(saveLocation1,"w+")
9
10     for r in result:
11         if cityToSearch in r['full_address'].decode():
12             name = r['name'].decode()
13             state = r['state'].decode()
14             city = r['city'].decode()
15             full_address = r['full_address'].decode()
16             output.write('%s$%s$%s$%s\n'%(name,full_address,city,state))
17
18     output.close()
19
20     pass
21
```

In order to understand the dataset dataset structure I have printed the first row of data (line 4). From which I found key names, value types, etc. Next, I have used the filter() function to get all rows where the "full address" contains the cityToSearch value (line 6). Filter function gives us a list of values matching the condition. Each item in the list is the data set row of the type dictionary. Finally, to write output to a given location (I.e saveLocation1), first will open the file using the open function with write mode (line 8). To write output, we will traverse through each item of the list "result", followed by extracting the value, decoding it and storing it in an equivalent object and then writing the value to opened file, where values are $ separated (from line 10 to line 16).

NOTE: 1.The given dataset values are in bytes, to run any operator we need to decode this value using the .decode() method.

B. FindBusinessBasedOnLocation():

In this function we need to find all businesses belonging to a specific category and falling within the max distance range from a given location. The function will take five parameters, these are categoriesToSearch, myLocation, maxDistance, saveLocation2 and Collection. Object “myLocation” is a list containing latitude and longitude value, using which we need to compute distance.

Screen-grab of code logic is shown below.

```
21
22  def FindBusinessBasedOnLocation(categoriesToSearch, myLocation, maxDistance, saveLocation2, collection):
23      from math import sin, cos, sqrt, atan2, radians
24
25      #get business based on categoriesToSearch
26      result = collection.filter(lambda obj: set(categoriesToSearch) <= set([x.decode() for x in obj['categories']]))
27
28      #open file to write data
29      output= open(saveLocation2,"w+")
30
31      R = 3959; # Aproximate Earth Radius in miles
32      lat1 = radians(myLocation[0])
33      lon1 = radians(myLocation[1])
34
35      for r in result:
36          lat2 = radians(r["latitude"])
37          lon2 = radians(r["longitude"])
38          dlat = lat2 - lat1
39          dlon = lon2 - lon1
40
41          a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
42          c = 2 * atan2(sqrt(a), sqrt(1 - a))
43          distance = R * c
44
45          if distance <= maxDistance:
46              name = r['name'].decode()
47              output.write('%s\n'%(name))
48
49      output.close()
50      pass
```

To get the fastest result, I have first minimised the dataset by filtering business on its categories. Filtering is achieved using the “filter()" function (line 26). Filter function returns result of all business where categories list contains “categoriesToSearch” value. Next, using a for loop we traverse through each filtered dataset row and extract business latitude and longitude values (line 36 and 37).

Substituting the latitudes and longitudes in the formula provided by project-1 instruction, we compute the distance between a given location and each business found after filtering (line 38 to 43). For final result, we check if the computed distance is less than or equal to maxDistance, if yes, we write the business name to the output file i.e saveLocation2 (line 45 to 47).

Lessons Learned:

1. Understand Jupiter interface, get a hand on and know how to create cell and execute them.
2. Learn how to handle NoSQL data and perform operations over NoSQL data.

Output:

A. Out put of FindBusinessBasedOnCity(cityToSearch, saveLocation1, 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

```

B. Out put of FindBusinessBasedOnLocation(categoriesToSearch, myLocation, maxDistance, saveLocation2, data) ->

```
VinciTorio's Restaurant

```

Result:

A. FindBusinessBasedOnCity(cityToSearch, saveLocation1, data) logic was successfully able to pass TestCase 1 from given two other test case.

TestCase 1: FindBusinessBasedOnCity('Scottsdale', 'output_city.txt', data)

```
Turf Direct$8350 E Evans Rd, Scottsdale, AZ 85260$Scottsdale$AZ
Sangria's$7700 E McCormick Pkwy, Scottsdale, AZ 85258$Scottsdale$AZ
3 Palms$7707 E McDowell Rd, Scottsdale, AZ 85257$Scottsdale$AZ
Bob's Bike Shop$1608 N Miller Rd, Scottsdale, AZ 85257$Scottsdale$AZ
Ronan & Tagart, PLC$8980 E Raintree Dr, Ste 120, Scottsdale, AZ 85260$Scottsdale$AZ

```

B. FindBusinessBasedOnLocation(categoriesToSearch, myLocation, maxDistance, saveLocation2, data) logic was successfully able pass Test Case 2 from given three other test cases.

TestCase 2: FindBusinessBasedOnLocation(['Bakeries'], [33.3482589, -111.9088346], 15, 'output_loc.txt', data)

Result:

```
Nothing Bundt Cakes
P.croissants

```