

DATA PROCESSING AT SCALE - CSE 511

Submitted By: Aman Peshin (1225476655)

PROJECT 1 REPORT

Reflection:

Assignment 5 was a challenging task since it required us to work with NoSQL databases which I had never done before. My approach to the problem statement was to create unique solutions for the given tasks. I feel that the function “FindBusinessBasedOnLocation” was more challenging since it asked us to work with latitudes and longitudes to determine the distance between the two locations. One complex implementation was the approach taken to not write the name of the restaurants twice in our output file. Let’s say for example that the categories we were looking for were buffet and Italian and the categories that the restaurant served up were buffet, Italian and continental. The traditional approach of solving this task using loops would make us write this restaurant name twice! Once for every element in the categories list. The unique approach I have taken is by solving it through the built-in concept of sets in python. Taking the previous example if the categories we are searching are put in a set and on the other hand, the restaurant’s categories are in another set if we subtract those two sets we will end up with a number of elements between 0 and no of search categories. A value of zero would mean that all the categories are present in the restaurant and two would be the value when none match. We can then compare these values to make sure that the restaurant’s name is printed once.

I have tried to make the code as modular as possible like in the function where we write output to file. Here I have separated out the code for both query functions into this common snippet. Also, the challenge lay in trying to insert a \$ symbol as a delimiter. The complexity lay in doing it for one output from “Find based on city” function and not for “Find based on location”. There was a simple implementation for the problem, where the data which did not need a delimiter was sent as a single element in a list. What I take away from this assignment is that Non-relational databases are very powerful and solution-oriented albeit very rarely used.

```
[128] # Graded Cell, Part10: oifkK

def FindBusinessBasedOnCity(cityToSearch, saveLocation1, collection):
    output = []
    data = collection.all()
    for item in data:
        if item['city'] == cityToSearch:
            item_name = item['name']
            item_addr = item['full address']
            item_city = item['city']
            item_stat = item['state']
            dict_temp = [item_name, item_addr, item_city, item_stat]
            output.append(dict_temp)
    writeOutputToTxt(saveLocation1, output)

def FindBusinessBasedOnLocation(categoriesToSearch, myLocation, maxDistance, saveLocation2, collection):
    output = []
    lati, loni = myLocation
    data = collection.all()
    for item in data:
        if len(set(categoriesToSearch) - set(item['categories'])) < len(categoriesToSearch):
            lat2 = item['latitude']
            lon2 = item['longitude']
            distance = calcDistance(lat2, lon2, lati, loni)
            if distance <= maxDistance:
                output.append([item['name']])
    writeOutputToTxt(saveLocation2, output)
```

```
def calcDistance(lat2, lon2, lat1, lon1):
    R = 3959
    latitude_1_rad = math.radians(lat1)
    latitude_2_rad = math.radians(lat2)
    difference_latitude_rad = math.radians(lat2 - lat1)
    difference_longitude_rad = math.radians(lon2 - lon1)
    calc = math.sin(difference_latitude_rad / 2) * math.sin(difference_latitude_rad/2) + math.cos(latitude_1_rad) * math.cos(latitude_2_rad) * math.sin(difference_longitude_rad / 2)
    c = 2 * math.atan2(math.sqrt(calc), math.sqrt(1 - calc))
    return R * c
```

```
[126] def writeOutputToTxt(filename, output):
    file = open(filename, 'w')
    for item in output:
        file.write('$'.join(str(s) for s in item))
        file.write('\n')
    file.close()
```

DATA PROCESSING AT SCALE - CSE 511

Submitted By: Aman Peshin (1225476655)

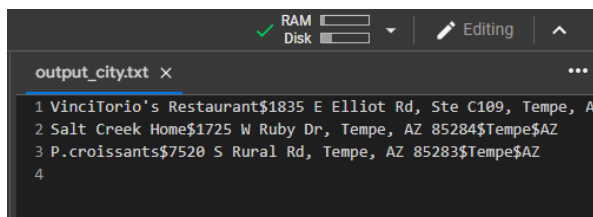
Lessons learned:

The IDE I used in this assignment are Jupyter notebook and google colab. This has been a great experience since I got to explore these two tools. I have learned about how the data is stored in a JSON/ key-value pair format in NoSQL which is very easily accessible once we iterate over all keys. The function which is used to calculate the distance between two points given by their latitude and longitude is complex and informative since such a function will help us understand graph databases in depth, especially the ones used in any web mapping application. Unqlite python package which is used in this task to read and store the NoSQL data is developed by a fellow python enthusiast and we can gain a lot of knowledge by looking at the distribution files and how he has integrated it with Visual Studio tools.

What I have learned is that Tools like Jupyter and Colab are extremely powerful in tasks where visualizations are a priority and having assignments, where we can work with them, would be the cherry on top of the cake.

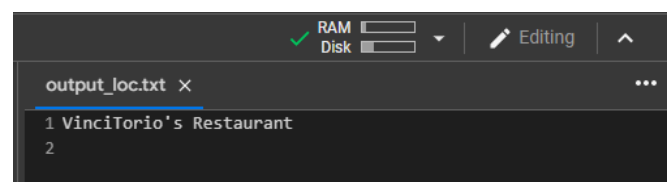
OUTPUT: Embedded tests

FindBusinessBasedOnCity:



```
output_city.txt x
1 VinciTorio's Restaurant$1835 E Elliot Rd, Ste C109, Tempe, A
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
```

FindBusinessBasedOnLocation:

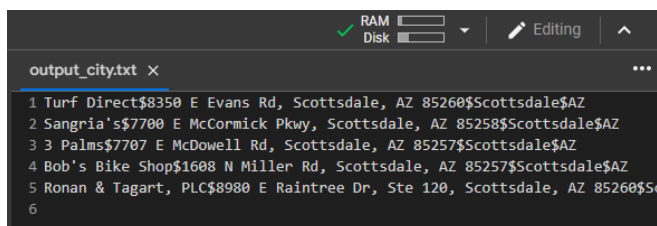


```
output_loc.txt x
1 VinciTorio's Restaurant
2
```

OUTPUT: Additional tests

FindBusinessBasedOnCity:

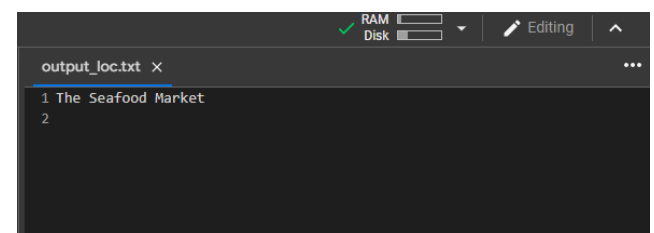
Additional test 1:



```
output_city.txt x
1 Turf Direct$8350 E Evans Rd, Scottsdale, AZ 85260$Scottsdale$AZ
2 Sangria's$7700 E McCormick Pkwy, Scottsdale, AZ 85258$Scottsdale$AZ
3 3 Palms$7707 E McDowell Rd, Scottsdale, AZ 85257$Scottsdale$AZ
4 Bob's Bike Shop$1608 N Miller Rd, Scottsdale, AZ 85257$Scottsdale$AZ
5 Ronan & Tagart, PLC$8980 E Raintree Dr, Ste 120, Scottsdale, AZ 85260$Sc
6
```

FindBusinessBasedOnLocation:

Additional test 1:

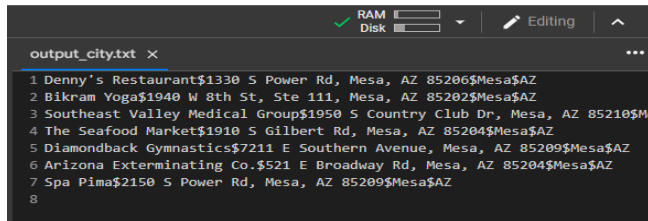


```
output_loc.txt x
1 The Seafood Market
2
```

DATA PROCESSING AT SCALE - CSE 511

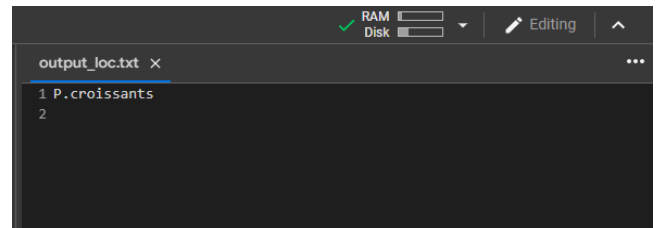
Submitted By: Aman Peshin (1225476655)

Additional test 2:



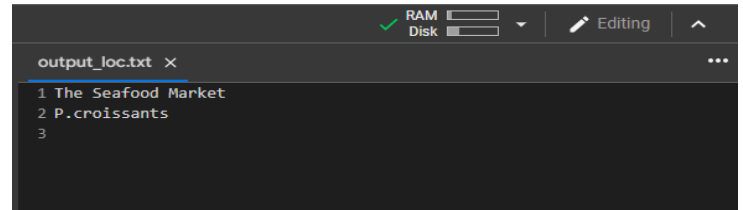
```
output_city.txt X
1 Denny's Restaurant$1330 S Power Rd, Mesa, AZ 85206$Mesa$AZ
2 Bikram Yoga$1940 W 8th St, Ste 111, Mesa, AZ 85202$Mesa$AZ
3 Southeast Valley Medical Group$1950 S Country Club Dr, Mesa, AZ 85210$Me
4 The Seafood Market$1910 S Gilbert Rd, Mesa, AZ 85204$Mesa$AZ
5 Diamondback Gymnastics$7211 E Southern Avenue, Mesa, AZ 85209$Mesa$AZ
6 Arizona Exterminating Co.$521 E Broadway Rd, Mesa, AZ 85204$Mesa$AZ
7 Spa Pima$2150 S Power Rd, Mesa, AZ 85209$Mesa$AZ
8
```

Additional test 2:



```
output_loc.txt X
1 P.croissants
2
```

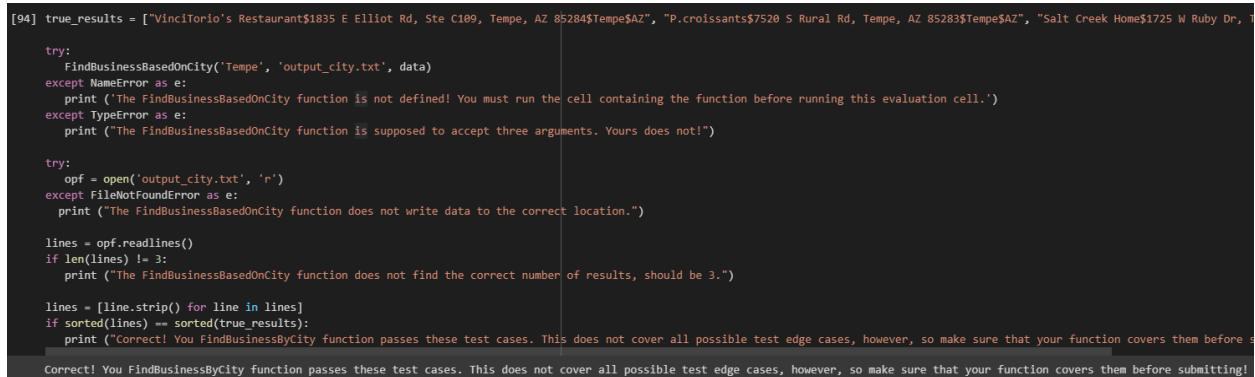
Additional test 3:



```
output_loc.txt X
1 The Seafood Market
2 P.croissants
3
```

RESULT: Embedded tests

FindBusinessBasedOnCity:



```
[94] 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", "Salt Creek Home$1725 W Ruby Dr, T

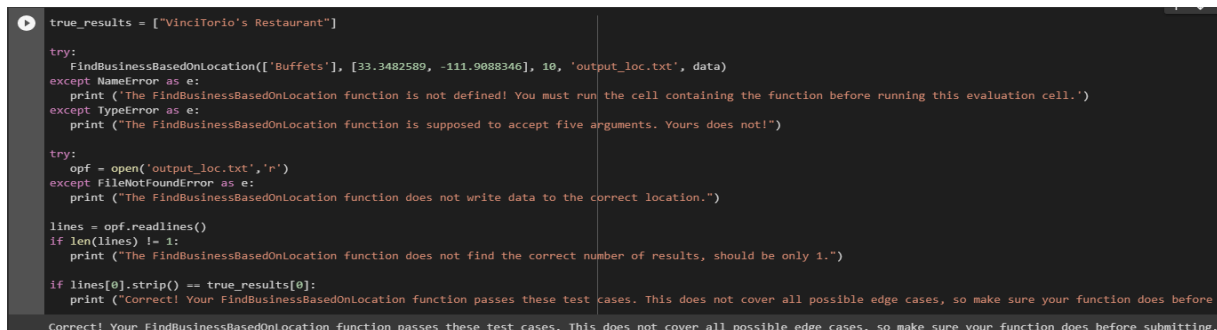
try:
    FindBusinessBasedOnCity('Tempe', 'output_city.txt', data)
except NameError as e:
    print ('The FindBusinessBasedOnCity function is not defined! You must run the cell containing the function before running this evaluation cell.')
except TypeError as e:
    print ("The FindBusinessBasedOnCity 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 that your function covers them before submitting!")
else:
    print ("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!")
```

FindBusinessBasedOnLocation:



```
▶ 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 function does before submitting!")
else:
    print ("Correct! Your FindBusinessBasedOnLocation function passes these test cases. This does not cover all possible edge cases, so make sure your function does before submitting.")
```

RESULT: Additional tests

FindBusinessBasedOnCity:

DATA PROCESSING AT SCALE - CSE 511

Submitted By: Aman Peshin (1225476655)

```
# Additional T1: FindBusinessBasedOnCity:
true_results = ['3 Palms$7707 E McDowell Rd, Scottsdale, AZ 85257$Scottsdale$AZ', 'Bob's Bike Shop$1008 N Miller Rd, Scottsdale, AZ 85257$Scottsdale$AZ', 'Ronan & Tagart, PLC$8980 E Raintree',
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 function before running this evaluation cell.')
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) != 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 that your function covers them before submitting!')

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!
```

```
# Additional T2: FindBusinessBasedOnCity:
true_results = ['Arizona Exterminating Co.$521 E Broadway Rd, Mesa, AZ 85204$Mesa$AZ', 'Bikram Yoga$1940 N 8th St, Ste 111, Mesa, AZ 85202$Mesa$AZ', 'Denny's Restaurant$1330 S Power Rd, Mesa,
try:
    FindBusinessBasedOnCity('Mesa', '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(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 that your function covers them before submitting!')

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!
```

FindBusinessBasedOnLocation:

```
#Additional T1: FindBusinessBasedOnLocation:
true_results = ['The Seafood Market']
try:
    FindBusinessBasedOnLocation(['Specialty Food'], [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.')
lines = [line.strip() for line in lines]
if sorted(lines) == sorted(true_results):
    print('Correct! Your FindBusinessBasedOnLocation function passes these test cases. This does not cover all possible edge cases, so make sure your function does before submitting.')

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

```
#Additional T2: FindBusinessBasedOnLocation:
true_results = ['P.croissants']
try:
    FindBusinessBasedOnLocation(['Bakeries'], [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.')
lines = [line.strip() for line in lines]
if sorted(lines) == sorted(true_results):
    print('Correct! Your FindBusinessBasedOnLocation function passes these test cases. This does not cover all possible edge cases, so make sure your function does before submitting.')

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

```
# Additional T3: FindBusinessBasedOnLocation:
true_results = ['The Seafood Market', 'P.croissants']
try:
    FindBusinessBasedOnLocation(['Food', 'Specialty Food'], [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) != 2:
    print('The FindBusinessBasedOnLocation function does not find the correct number of results, should be only 2.')
lines = [line.strip() for line in lines]
if sorted(lines) == sorted(true_results):
    print('Correct! Your FindBusinessBasedOnLocation function passes these test cases. This does not cover all possible edge cases, so make sure your function does before submitting.')

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