

NoSQL Project Report

Reflection

In this project, the goal was to implement functions to query a NoSQL database, specifically UnQLite, to find businesses based on certain criteria. The project required creating functions that could search for businesses by city and by location within a given distance.

I approached the project by first understanding the requirements and then planning the structure of the functions needed. I implemented two functions: 'FindBusinessBasedOnCity' and 'FindBusinessBasedOnLocation'. The 'DistanceFunction' was implemented to support the FindBusinessBasedOnLocation function.

The design process involved understanding the requirements, planning the function structure, and implementing the functions with careful consideration of edge cases and data integrity. Below is a detailed explanation of each function and the design choices made.

- FindBusinessBasedOnCity Function

The 'FindBusinessBasedOnCity' function searches for businesses in a specified city and writes the results to a file. It ensures case-insensitive matching for city names and avoids duplicates by using a set. The function iterates over each business in the collection, checking if the city matches the 'cityToSearch' (case-insensitive). Then, the matching business details are formatted and written to the specified file.

- FindBusinessBasedOnLocation Function

The 'FindBusinessBasedOnLocation' function searches for businesses within a specified distance from a given location and writes the results to a file. It filters businesses based on the provided categories and ensures that only unique business names are included in the results. First, latitude and longitude of the provided location ('myLocation') are converted to floats to ensure accurate calculations. A set 'my_set' is used to store unique business names, ensuring that no duplicates are written to the file. The file specified by saveLocation2 is opened for writing. The with statement ensures that the file is properly closed after writing. The function iterates over each business in the collection. For each business, it retrieves the latitude and longitude and calculates the distance from the provided location using the 'DistanceFunction'. If the calculated distance is less than or equal to 'maxDistance', the business is considered for further filtering based on categories. The function iterates over the categories specified in 'categoriesToSearch'. If any of these categories are present in the business's categories, the business is considered for inclusion in the output. Before writing the business name to the file, the function checks if the name is already in the set 'my_set'. If not, it adds the name to the set and writes it to the file.

The 'DistanceFunction' was written to support the 'FindBusinessBasedOnLocation' function. It calculates the distance between two geographic coordinates using the Haversine formula. This formula accounts for the Earth's curvature, providing an accurate distance measurement between two points. The 'to_radians' function converts degrees to radians, which is necessary for trigonometric calculations in the Haversine formula. The formula itself calculates the great-circle distance between two points on the Earth's surface.

Lessons Learned

- Understanding NoSQL Databases

This project provided hands-on experience with UnQLite, an embedded NoSQL database. I learned how to perform basic CRUD operations and understand the nuances of NoSQL data structures compared to traditional SQL databases.

- Geospatial Calculations

Implementing the Haversine formula to calculate distances between geographic coordinates was a practical application of mathematical concepts. This is particularly useful in applications involving location-based services.

- Data Handling and Processing

Ensuring data consistency and handling edge cases became clear, such as ensuring all necessary fields are present and performing case-insensitive searches. Using sets to maintain uniqueness in results was also a key design.

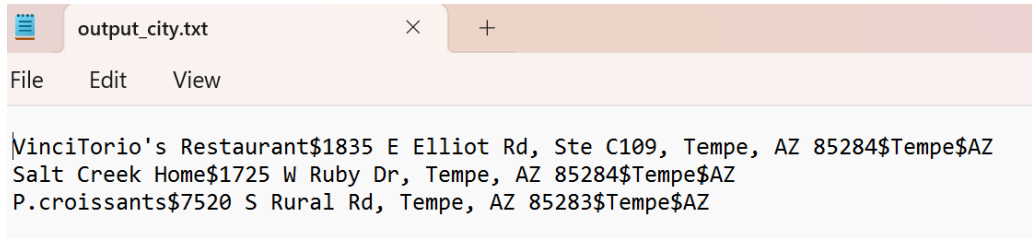
- File I/O in Python

Writing results to files in a specified format reinforced my understanding of file I/O operations in Python. This is a crucial skill for data processing tasks.

Output

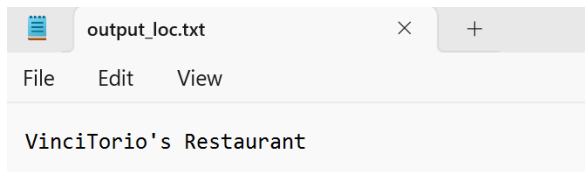
Name	Status	Date modified	Type	Size
.ipynb_checkpoints	✓	6/16/2024 9:30 PM	File folder	
Guo_Fangjie_CSE511_NoSQL Project Report	↻	6/18/2024 7:32 PM	Microsoft Word D...	26 KB
output_city	↻	6/18/2024 7:33 PM	Text Document	1 KB
output_loc	↻	6/18/2024 7:33 PM	Text Document	1 KB
project1	✓	6/17/2024 9:48 PM	Jupyter Source File	16 KB
sample	✓	5/14/2024 9:14 PM	DB File	56 KB

- output_city.txt (FindBusinessBasedOnCity Function output)



```
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
```

- output_loc.txt (FindBusinessBasedOnLocation Function output)



```
VinciTorio's Restaurant
```

Result

```
[217]: 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$1725 W Ruby Dr, Tempe, AZ 85284$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 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!
```

```
[218]: 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 covers them 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.
```