With the rise of online sources of data and the demands of online traffic, various popular applications have built their data storage through NoSQL databases. The report describes the process of programming software to filter, and write to a file, businesses using the business's city, location, or category.

<u>Reflection</u>

In order to filter businesses based off a specified city, I wrote FindBusinessBasedOnCity. The function FindBusinessBasedOnCity is split into two parts: filtering the specified NoSQL collection based off of city names, and then writing the filtered cities to a specified file. In order to parse the NoSQL document, I used UnQLite filter function. The filter function selects which items should be included from a collection by using a function or lambda to return true if the item should be included and false if the item should not be included. To filter for businesses that should be selected because they are in the specified city, I set the lambda to only select businesses that have a 'city' JSON attribute equal to the specified city. In order to record the filtered businesses, I opened a file location to write to by the user, and loop through each business to record relevant information. For this function, the user needs to know the name of the business, the full address of the business, the city of the business, and the state the business is located in. For each row of the file, the businesses are stored in the format: "Name$FullAddress$City$State".

The second task of this software was to report businesses if they are within a specified maximum distance away and fit one of a specified list of categories. This task required the creation of two functions: the main method, FindBusinessBasedOnLocation, and a distance calculation helper function, distanceFunction. The method FindBusinessBasedOnLocation is split into the same two parts as FindBusinessBasedOnCity: the filtering step and the writing step, but the filtering step for this task is more complicated.

Due to users expecting to find businesses in dynamic locations, the businesses NoSQL collection does not store the distance from users as an attribute but instead stores their coordinates. These coordinates must be used to calculate the distance from the user's specified coordinates. The formulas used for distance calculation from coordinates cannot fit on one line, so I wrote the function distanceFunction to return the distance from the business's coordinates to the user's coordinates. When programming distanceFunction, I assumed that the user would not need precise enough data to factor in hills and bumpiness that is not captured using a coordinate projection, so I utilized the haversine formula to calculate the distances between two pairs of coordinates. The haversine formula code, which was provided by the professors as part of this project, converts coordinate pair to radians and uses those coordinates in the radians with sine, cosine, and atan2 functions to calculate their distance apart. This distance value is checked against a maxDistance values specified by the user so only businesses that are closer than the maxDistance are included in the filter.

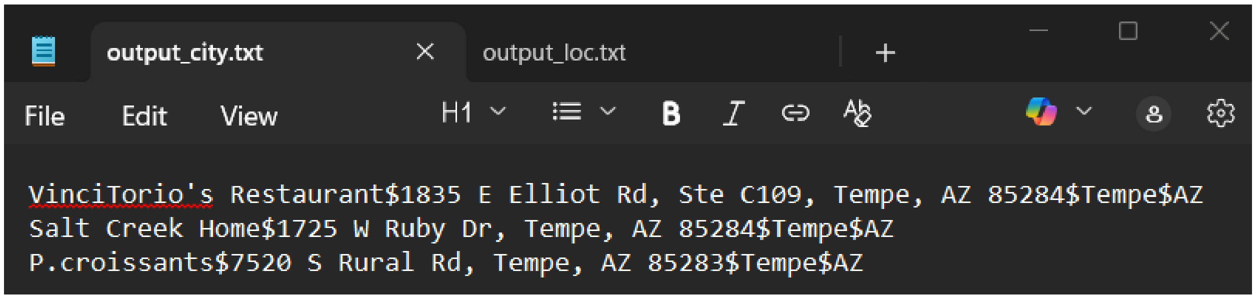
Once the distance from the business is calculated using distanceFunction, the business is checked if it fits under a list of categories provided by the user. The task of checking if a business's list of categories share a value with the specified list of categories can be accomplished in one line. By converting the business's list of categories to a set, I was able to run the intersection function between the two lists, and then only include a business if the shared set between the business's categories and the specified categories is not empty. Using the filter method discussed in the previous task, I set the filter to only include businesses that pass both conditions.

Once the list of businesses are filtered down to businesses that pass both conditions, I follow the same format as the previous task. However, the output written to the specified files follows the format: "Name" instead of "Name$FullAddress$City$State".

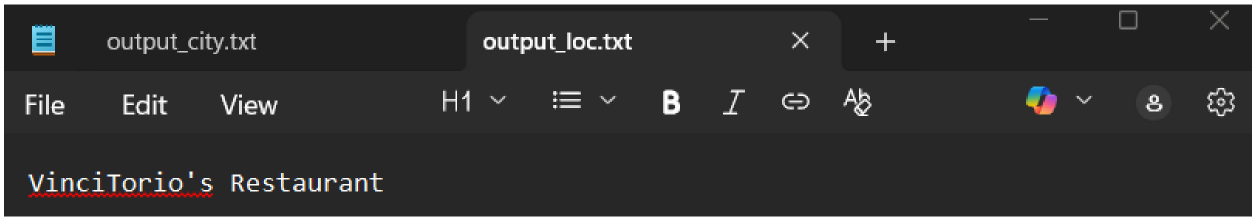
Lessons Learned

Through the process of programming this software, I learned how to process geolocation data, and furthered my understanding of processing NoSQL data. Through my previous work of processing data from the Nobel Prize API, I have learned how to process NoSQL data, as their API sent out the data using JSON. This process taught me how to navigate the various attributes in a JSON object, but I had utilized different libraries to process that data. UnQLite, a library to better process NoSQL data in Python, made the process of filtering data down to a small list of elements much easier than using for loops to cycle through each data point or running the filtering though Pandas Dataframe objects. Through this filtering step, I gained more experience of using the lambda function in python, because most of the datasets I would have used could have been done lazily through using for loops to return the filtered values. This project provided my first instance of creating a distance function to calculate 3D distance. The haversine formula is an elegant form of calculating distances without having to factor in projections of the Earth or Earth's topology.

Output of The Two Functions



```
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
```



```
VinciTorio's Restaurant
```

Results

The FindBusinessBasedOnCity and FindBusinessBasedOnLocation functions were tested using a sample business NoSQL database provided as part of the project. The test for FindBusinessBasedOnCity searched for businesses in the city of Tempe and wrote to output_city.txt three businesses: VinciTorio's Restaurant, Salt Creek Home, and P.croissants. The test for FindBusinessBasedOnLocation wrote to output_loc.txt only the business VinciTorio’s Resturant after searching for businesses that were categorized with the category “Buffets” and was 10 miles from the coordinates 33.3482589 lattitude, -111.9088346 longitude.