# NoSQL Project Report (UnQLite & Jupyter)

**Kiruthika Ponnan**
**ASUID: 1227400293**

---

## 1. Reflection

For this project, I was tasked with implementing two functions: FindBusinessBasedOnCity and FindBusinessBasedOnLocation. I also had to create a distance function and provide an explanation of both functions.

### A) FindBusinessBasedOnCity Function:

The FindBusinessBasedOnCity function searches a given UnQLite collection in order to find all the businesses that exist in a specific city. It then saves the retrieved information to a specified location.

To accomplish this, I followed these steps:

1. Iterate through each entry in the collection.
2. Check if the city of the current entry matches the cityToSearch parameter.
3. If there is a match, retrieve the full address, name, state, and city of the business.
4. Output is then saved in the following format: Name$FullAddress$City$State.
5. Append each business entry as a new line in the file located at saveLocation1.

Example code for the FindBusinessBasedOnCity function:

```
def FindBusinessBasedOnCity(cityToSearch, saveLocation1, collection):
    businesses = collection.filter(lambda b: b['city'] == cityToSearch)
    with open(saveLocation1, 'w') as file:
        for business in businesses:
            line = "{}${}${}${}\n".format(
                business['name'],
                business['full_address'],
                business['city'],
                business['state']
            )
            file.write(line)
    time.sleep(2)
```

# NoSQL Project Report (UnQLite & Jupyter)

## Kiruthika Ponnan
## ASUID: 1227400293


---

### B) FindBusinessBasedOnLocation Function:

The FindBusinessBasedOnLocation function searches a given UnQLite collection to locate the businesses (with the name provided) located within the given input location. It then saves the retrieved business names to a specified location.

To implement this function, I employed ***haversine*** formula to calculate distance in miles between the given location and each business in the collection. The distance is compared against the maxDistance parameter, and businesses within that range are considered.

Here are the steps I took to create the FindBusinessBasedOnLocation function:

1. Iterate through each entry in the collection.
2. Calculate the distance between the myLocation and the coordinates of the current business using the haversine formula.
3. Check if maxDistance >= distance calulated.
4. If the distance is within the specified range, save the business name to the file at saveLocation2.
5. Each business name is appended to the output file.

Example code for the FindBusinessBasedOnLocation function:

```
def FindBusinessBasedOnLocation(categoriesToSearch, myLocation, maxDistance, saveLocation2, collection):
    businesses = collection.all()
    nearby_businesses = []
    for business in businesses:
        categories_exists = set(business['categories']).intersection(set(categoriesToSearch))
        if categories_exists:
            distance = haversine(business['latitude'], business['longitude'], myLocation[0], myLocation[1])
            if distance <= maxDistance:
                nearby_businesses.append(business['name'])
    with open(saveLocation2, 'w') as file:
        for business_name in nearby_businesses:
            file.write("{}\n".format(business_name))
    time.sleep(2)
```

# NoSQL Project Report (UnQLite & Jupyter)

## Kiruthika Ponnan
## ASUID: 1227400293


---

### C) Distance Function (haversine formula):

```python
# Data Processing at Scale - Assignment 1 (NoSQL)
# @author  : Kiruthika Ponnan - ASUID: 1227400293

from math import radians, sin, cos, sqrt, atan2
from unqlite import UnQLite
import time

def haversine(lat1, lon1, lat2, lon2):
    R = 3959  # Radius of the Earth in miles

    dlat = radians(lat2 - lat1)
    dlon = radians(lon2 - lon1)

    a = sin(dlat / 2) ** 2 + cos(radians(lat1)) * cos(radians(lat2)) * sin(dlon / 2) ** 2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance
```

Haversine formula calculator (Distance Function), takes into account the curvature of the Earth when calculating distances, making it suitable for calculating distances between latitude and longitude coordinates.

Here's a breakdown of the function:

1. R = 3959 assigns the value 3959 to the variable `R`, representing the approximate radius of the Earth in miles. This value is used later in the calculation.

2. dlat = radians(lat2 - lat1) calculates the difference in latitude between the two points and converts it from degrees to radians. The `radians` function is used to perform this conversion.

3. dlon = radians(lon2 - lon1) calculates the difference in longitude between the two points and converts it from degrees to radians.

4. The following line calculates the Haversine formula:

```python
a = sin(dlat / 2) ** 2 + cos(radians(lat1)) * cos(radians(lat2)) * sin(dlon / 2) ** 2
```

**NoSQL Project Report (UnQLite & Jupyter)**
**Kiruthika Ponnan**
**ASUID: 1227400293**

---

5. `c = 2 * atan2(sqrt(a), sqrt(1 - a))` uses the `atan2` function to calculate the arctangent of the square root of `a` divided by the square root of `1 - a`. The result is multiplied by 2 to get the central angle between the two points.

6. distance = R * c calculates the distance by multiplying the Earth's radius by the central angle.

7. The calculated distance is returned as the output of the function.

## 2. Lessons Learned

a) **Working with UnQLite:** I learned how to use UnQLite, a NoSQL database, to store and retrieve data efficiently. Understanding the concepts and functionality of such databases is beneficial when dealing with large datasets.

**b) Working with Jupyter Notebooks:** I learned how to setup jupyter notebooks and install all the necessary tools used to fetch and manipulate the data from a given data source.

**c) File Manipulation:** Implementing file read and write operations allowed me to save and retrieve data in different formats. This knowledge can be applied to various scenarios that involve working with files and data persistence.

**d) Geographic Distance Calculation:** By utilizing the haversine formula, I enhanced my understanding of calculating distances between geographical locations on the Earth's surface. This knowledge can be extended to other applications requiring distance calculations in real-world scenarios.

**e) Code Modularity:** Developing reusable functions helped improve code organization and maintainability. Separating different functionalities into distinct functions promotes code reusability and simplifies testing and debugging.

## 3. Output Screenshots

A) The following screenshot shows

- sample.db file loaded into Jupyter notebook (for local dev only)
- FindBusinessBasedOnCity called with params

# NoSQL Project Report (UnQLite & Jupyter)

**Kiruthika Ponnan**
**ASUID: 1227400293**

---

- FindBusinessBasedOnLocation called with params

```
print("Loading database...")
data = LoadDB('sample.db')
# for business in data.all():
#     print(business)
print("Database loaded successfully!")

FindBusinessBasedOnCity('Tempe', 'output_city.txt', data)
FindBusinessBasedOnLocation(
    ['Food', 'Specialty Food'],
    [33.3482589, -111.9088346], 30,
    'output_loc.txt',
    data
)
```

B) FindBusinessBasedOnCity output file

output_city.txt

databases > unQLite > python > output_city.txt

```
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$1950 S Country Club Dr, Mesa, AZ 85210$Mesa$AZ
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
```

C) FindBusinessBasedOnLocation output file

# NoSQL Project Report (UnQLite & Jupyter)
Kiruthika Ponnan
ASUID: 1227400293

---

output_city.txt | output_loc.txt

databases > unQLite > python > output_loc.txt

```
1  Nothing Bundt Cakes
2  Olive Creations
3  The Seafood Market
4  P.croissants
5
```

## 4. Results of the code passing test cases locally

solution.ipynb

databases > unQLite > python > solution.ipynb > # Tests

Code | Markdown | Run All | Clear All Outputs | Restart | Variables | Outline | Python 3 (ipykernel)

```
try:
    FindBusinessBasedOnLocation(
        ['Bakeries'], [33.3482589, -111.9088346], 15, '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.")

true_results = ['Nothing Bundt Cakes', 'Olive Creations',
                'P.croissants', 'The Seafood Market']
try:
    FindBusinessBasedOnLocation(['Food', 'Specialty Food'], [
                                33.3482589, -111.9088346], 30, '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) != 4:
    print("The FindBusinessBasedOnLocation function does not find the correct number of results, should be only 4.")
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.")
```

[48] ✓ 10.2s Python

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