

# Project 1-Individual Report

**BY: Shreyas Tirukoilur Parvatharajan**

**ASU ID:1231832006**

# REFLECTION

## Code:

```
import math

def to_radians(degrees):
    return degrees * (math.pi / 180)

def FindBusinessBasedOnCity(cityToSearch, saveLocation, collection):
    businesses = collection.filter(lambda business: business['city'].lower() == cityToSearch.lower())
    with open(saveLocation, 'w') as file:
        for business in businesses:
            file.write(f"{business['name']}${business['full_address']}${business['city']}${business['state']}\n")

def haversine_distance(lat1, lon1, lat2, lon2):
    R = 3959
    φ1, φ2 = map(to_radians, [lat1, lat2])
    Δφ = to_radians(lat2 - lat1)
    Δλ = to_radians(lon2 - lon1)
    a = math.sin(Δφ / 2) ** 2 + math.cos(φ1) * math.cos(φ2) * math.sin(Δλ / 2) ** 2
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    return R * c

def is_within_distance(business, myLocation, maxDistance):
    business_lat, business_lon = business['latitude'], business['longitude']
    distance = haversine_distance(myLocation[0], myLocation[1], business_lat, business_lon)
    return distance <= maxDistance

def FindBusinessBasedOnLocation(categoriesToSearch, myLocation, maxDistance, saveLocation, collection):
    businesses = collection.filter(
        lambda business: any(category in business['categories'] for category in categoriesToSearch) and
        is_within_distance(business, myLocation, maxDistance)
    )
    with open(saveLocation, 'w') as file:
        for business in businesses:
            file.write(f"{business['name']}\n")
```

## Explanation:

### Step 1: Importing the `math` module

```
import math
```

This line imports the `math` module, which provides mathematical functions and constants.

### Step 2: Defining the `to\_radians` function

```
def to_radians(degrees):
    return degrees * (math.pi / 180)
```

This function takes an angle in degrees as input and returns the equivalent angle in radians. It's used for converting latitude and longitude values from degrees to radians.

### Step 3: Defining the `FindBusinessBasedOnCity` function

```
def FindBusinessBasedOnCity(cityToSearch, saveLocation, collection):
    businesses = collection.filter(lambda business: business['city'].lower() == cityToSearch.lower())
    with open(saveLocation, 'w') as file:
        for business in businesses:
            file.write(f"{business['name']} ${business['full_address']} ${business['city']} ${business['state']}\n")
```

This function takes three arguments: `cityToSearch`, `saveLocation`, and `collection`. It filters the businesses in the collection based on the provided city (`cityToSearch`) and writes the details of each business (name, full address, city, and state) to a file specified by `saveLocation`.

### Step 4: Defining the `haversine\_distance` function

```
def haversine_distance(lat1, lon1, lat2, lon2):
    R = 3959
    φ1, φ2 = map(to_radians, [lat1, lat2])
    Δφ = to_radians(lat2 - lat1)
    Δλ = to_radians(lon2 - lon1)
    a = math.sin(Δφ / 2) ** 2 + math.cos(φ1) * math.cos(φ2) * math.sin(Δλ / 2) ** 2
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    return R * c
```

This function calculates the Haversine distance between two points on the Earth's surface given their latitude and longitude coordinates. It's used to calculate the distance between the user's location and each business location.

### Step 5: Defining the `is\_within\_distance` function

```
def is_within_distance(business, myLocation, maxDistance):
    business_lat, business_lon = business['latitude'], business['longitude']
    distance = haversine_distance(myLocation[0], myLocation[1], business_lat, business_lon)
    return distance <= maxDistance
```

This function checks if a business is within a specified distance (`maxDistance`) from a given location (`myLocation`). It calculates the distance between the business location and the specified location using the `haversine\_distance` function and returns `True` if the distance is less than or equal to `maxDistance`, indicating that the business is within range.

## Step 6: Defining the `FindBusinessBasedOnLocation` function

```
def FindBusinessBasedOnLocation(categoriesToSearch, myLocation, maxDistance, saveLocation, collection):
    businesses = collection.filter(
        lambda business: any(category in business['categories'] for category in categoriesToSearch) and
        is_within_distance(business, myLocation, maxDistance)
    )
    with open(saveLocation, 'w') as file:
        for business in businesses:
            file.write(f"{business['name']}\n")
```

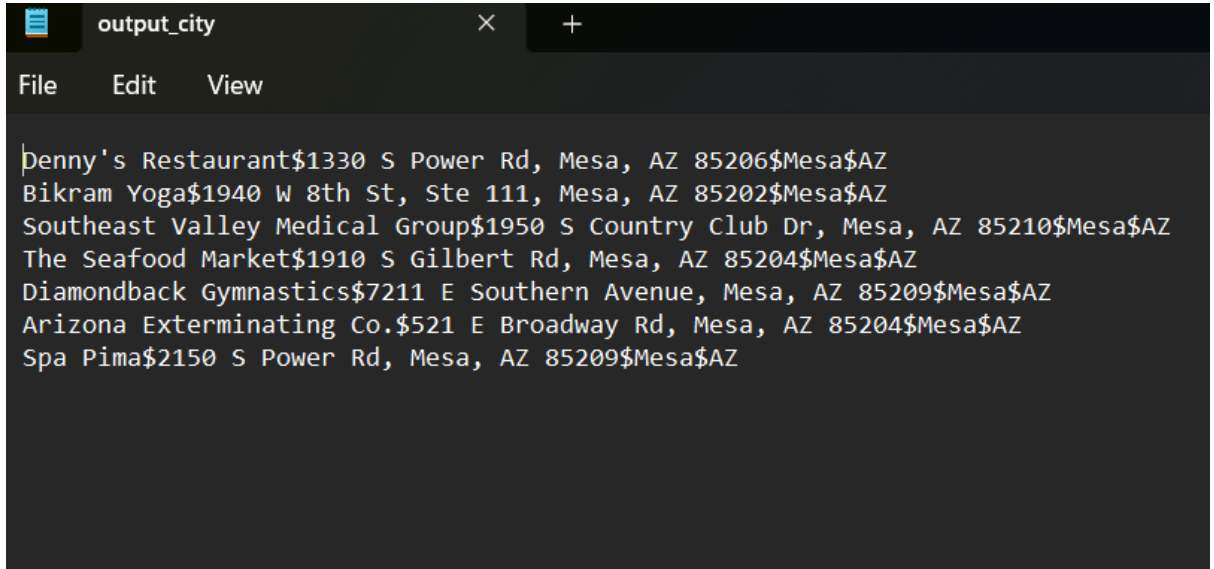
This function takes five arguments: `categoriesToSearch`, `myLocation`, `maxDistance`, `saveLocation`, and `collection`. It filters the businesses in the collection based on the provided categories (`categoriesToSearch`) and their proximity to the specified location (`myLocation`). It then writes the names of the filtered businesses to a file specified by `saveLocation`.

## LESSONS LEARNED

1. **Database Querying and Filtering:** Utilizing database querying to filter and retrieve data based on specific criteria (e.g., finding businesses in a particular city).
2. **Algorithm Implementation:** Implementing the Haversine formula to calculate distances between geographic coordinates, which is important for geospatial data processing.
3. **Data Manipulation:** Performing data manipulation tasks such as filtering, transforming, and writing data to files in a structured format. This includes handling different data types and structures effectively.
4. **File Handling:** Working with files to read input data, write output data, and manage file operations.
5. **Testing Functions:** Making sure that functions work correctly by testing them and verifying their accuracy.
6. **Error Handling:** Implementing error handling mechanisms, such as handling file not found errors or type errors, to ensure reliability of the code.

# OUTPUT

1: output\_city

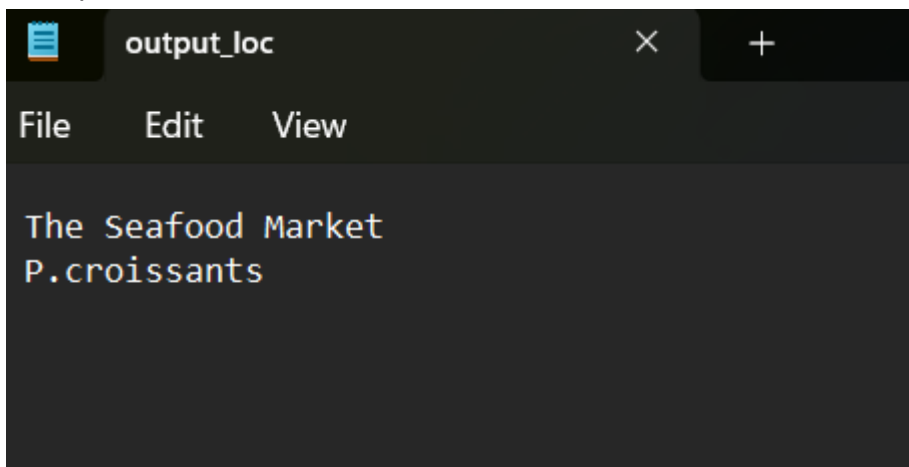


```

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

```

2:output\_loc



```

The Seafood Market
P.croissants

```

# RESULT

## Test case 1:

```
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!

## Test case 2:

```
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.")
```

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

### Test case 3:

```
# Test case for the FindBusinessBasedOnCity function
true_results = [
    '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',
    "Sangria's$7700 E McCormick Pkwy, Scottsdale, AZ 85258$Scottsdale$AZ",
    'Turf Direct$8350 E Evans Rd, Scottsdale, AZ 85260$Scottsdale$AZ'
]

try:
    FindBusinessBasedOnCity('Scottsdale', 'output_city.txt', data)
    with open('output_city.txt', 'r') as opf:
        lines = opf.readlines()
        lines = [line.strip() for line in lines]
        if sorted(lines) != sorted(true_results):
            print("The FindBusinessBasedOnCity function's output is incorrect.")
        else:
            print("Correct! The FindBusinessBasedOnCity 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!")
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!")
except FileNotFoundError as e:
    print("The FindBusinessBasedOnCity function does not write data to the correct location.")
```

Correct! The FindBusinessBasedOnCity 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!

### Test case 4:

```
# Test case for the FindBusinessBasedOnCity function
true_results = [
    'Arizona Exterminating Co.$521 E Broadway Rd, Mesa, AZ 85204$Mesa$AZ',
    'Bikram Yoga$1940 W 8th St, Ste 111, Mesa, AZ 85202$Mesa$AZ',
    "Denny's Restaurant$1330 S Power Rd, Mesa, AZ 85206$Mesa$AZ",
    'Diamondback Gymnastics$7211 E Southern Avenue, Mesa, AZ 85209$Mesa$AZ',
    'Southeast Valley Medical Group$1950 S Country Club Dr, Mesa, AZ 85210$Mesa$AZ',
    'Spa Pima$2150 S Power Rd, Mesa, AZ 85209$Mesa$AZ',
    'The Seafood Market$1910 S Gilbert Rd, Mesa, AZ 85204$Mesa$AZ'
]

try:
    FindBusinessBasedOnCity('Mesa', 'output_city.txt', data)
    with open('output_city.txt', 'r') as opf:
        lines = opf.readlines()
        lines = [line.strip() for line in lines]
        if len(lines) != 7:
            print("The FindBusinessBasedOnCity function does not find the correct number of results, should be 7.")
        if sorted(lines) == sorted(true_results):
            print("Correct! Your 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("The FindBusinessBasedOnCity function's output is incorrect.")
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!")
except FileNotFoundError as e:
    print("The FindBusinessBasedOnCity function does not write data to the correct location.")
```

Correct! Your 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!

## Test case 5:

```
# Test case for the FindBusinessBasedOnLocation function
true_results = ['The Seafood Market']

try:
    FindBusinessBasedOnLocation(['Specialty Food'], [33.3482589, -111.9088346], 10, 'output_loc.txt', data)
    with open('output_loc.txt', 'r') as opf:
        lines = opf.readlines()
        lines = [line.strip() for line in lines]
        if len(lines) != 1:
            print("The FindBusinessBasedOnLocation function does not find the correct number of results, should be only 1.")
        else:
            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.")
            else:
                print("The output is incorrect for FindBusinessBasedOnLocation function.")
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(e)
    print("The FindBusinessBasedOnLocation function is supposed to accept five arguments. Yours does not!")
except FileNotFoundError as e:
    print("The FindBusinessBasedOnLocation function does not write data to the correct location.")
```

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

## Test case 6:

```
# Test case for the FindBusinessBasedOnLocation function
true_results = ['P.croissants']

try:
    FindBusinessBasedOnLocation(['Bakeries'], [33.3482589, -111.9088346], 10, 'output_loc.txt', data)
    with open('output_loc.txt', 'r') as opf:
        lines = opf.readlines()
        lines = [line.strip() for line in lines]
        if len(lines) != 1:
            print("The FindBusinessBasedOnLocation function does not find the correct number of results, should be only 1.")
        else:
            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.")
            else:
                print("The output is incorrect for FindBusinessBasedOnLocation function.")
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(e)
    print("The FindBusinessBasedOnLocation function is supposed to accept five arguments. Yours does not!")
except FileNotFoundError as e:
    print("The FindBusinessBasedOnLocation function does not write data to the correct location.")
```

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

## Test case 7:

```
# Test case for the FindBusinessBasedOnLocation function with multiple results
true_results = ['The Seafood Market', 'P.croissants']

try:
    FindBusinessBasedOnLocation(['Food', 'Specialty Food'], [33.3482589, -111.9088346], 10, 'output_loc.txt', data)
    with open('output_loc.txt', 'r') as opf:
        lines = opf.readlines()
        lines = [line.strip() for line in lines]
        if len(lines) != 2:
            print("The FindBusinessBasedOnLocation function does not find the correct number of results, should be 2.")
        else:
            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.")
            else:
                print("The output is incorrect for FindBusinessBasedOnLocation function.")
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(e)
    print("The FindBusinessBasedOnLocation function is supposed to accept five arguments. Yours does not!")
except FileNotFoundError as e:
    print("The FindBusinessBasedOnLocation function does not write data to the correct location.")
```

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