SI 206 Final Report

Rec Team: Meghan Quigley, Nicole Claerhout, and Lauren Belous
Link to repository: https://github.com/meghanq/S1206-final-project.git

1. The goals for your project (10 points)

Our goal of the project was to determine if there was a correlation between the number of rec areas around a city and the quality of life of that city. To do this, we outlined four main goals:

a. Gather all of the cities provided in the Quality of Life API and the average ratings for all given domains
    i. Calculate one overall average quality of life to represent that city
b. Gather all of the longitudes and latitudes of the cities provided in the Quality of Life API to use to look up recreation areas
    i. Use the longitude and latitude of each city to search for the federal recreation areas within a 50 mile radius
c. Count the number of recreational areas returned for each city search
    i. Provide visualizations to reflect our findings, including a bar chart comparing the rec area number to the quality of life average, and pie charts comparing higher ratings of quality of life among each other and lower ratings of quality of life among each other

2. The goals that were achieved (10 points)

We were able to achieve all of our main goals with only a few minor changes to our plan. We were able to find the counts of the number of rec areas within the 50 mile radius as originally planned. Additionally, we were able to collect all of the ratings provided by the Quality of Life API to calculate one overall average for each city. The only change we really made was within our visualizations, as we chose to also include a scatter plot. We thought it would be easier to visualize any possible correlations and represent our data. While doing so, we also decided to include a function that produces a correlation coefficient to numerically show any possible correlation. The results of our data analysis show that the correlation coefficient between these the average quality of life for a city and the number of rec areas a city has is weak as it was calculated to be 0.072.

3. The problems that you faced (10 points)

a. When gathering the longitudes and latitudes, the website we were using had the city names formatted differently than those provided in the Quality of Life API. Therefore, we had to make a dictionary with matching keys and after doing this, we discovered a few cities were missing so we had to do a separate search to account for these.

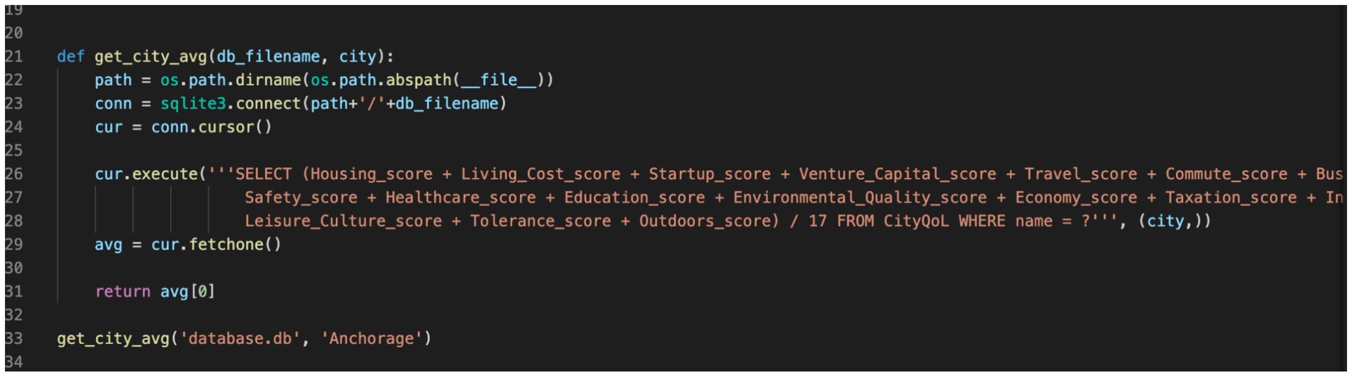
    i. This code, included in lat_long_new.py on lines 57 to 63, adds the missing cities to the database from a hardcoded list.

b. We also ran into merging errors when we were all working on the project at the same time, even within our own files. We mitigated this by copying our changes, deleting our files, recloning the git repository, and then inserting our changes if these errors came up.

c. When collecting the recreation area data, Washington D.C. returned 1025 recreation areas while all the other cities returned less than 20. Although this makes sense considering that the Recreation API only retrieved federal recreation areas, this skewed the data for our visualizations providing illegible depictions. Therefore, we accounted for this outlier by just removing it just from the scatter plot and bar chart.

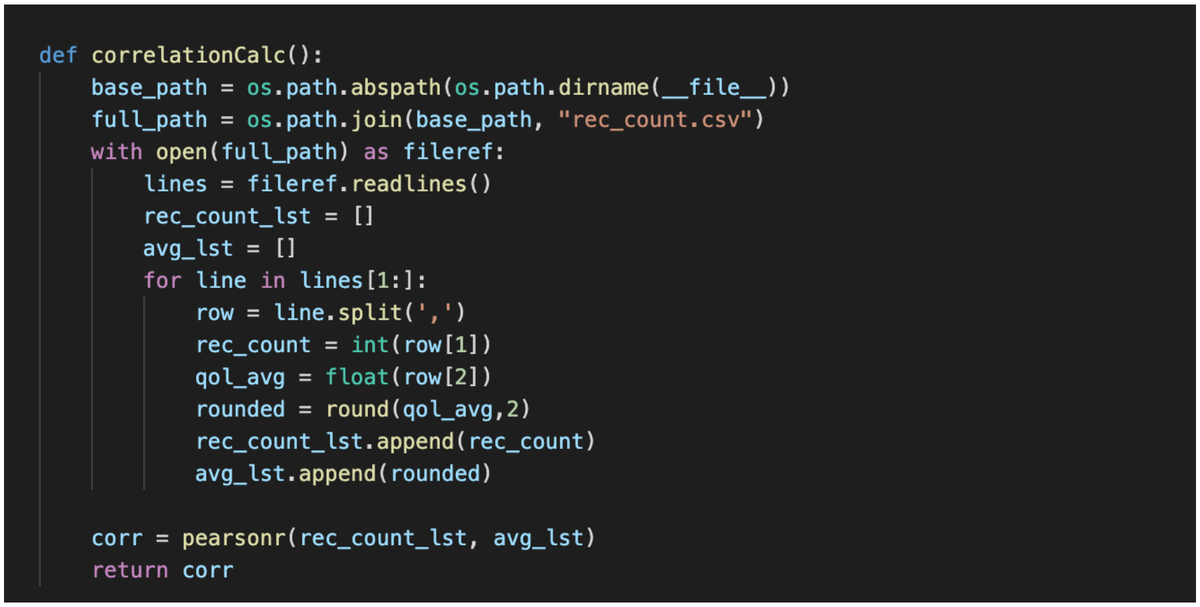
d. We also had problems adding the average quality of life calculations to the database column avgQoL because there were issues 1)calculating the averages and 2)inputting them into the database. Specifically, rrors occurred when we tried to put the averages into the database with the initial data we were pulling from the api. To mitigate this, we created a separate file (QoL_Calculations) to run the calculations and used the UPDATE command to add each average to the database after all the other data had already been added.

4. Your file that contains the calculations from the data in the database (10 points)



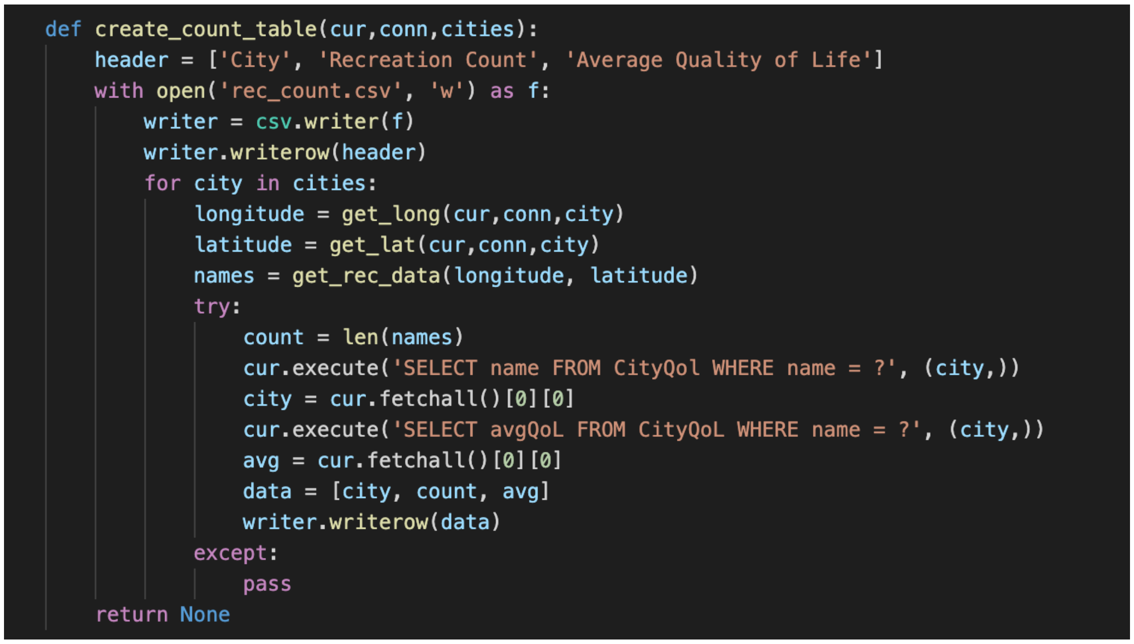
```

def get_city_avg(db_filename, city):
    path = os.path.dirname(os.path.abspath(__file__))
    conn = sqlite3.connect(path+'/'+db_filename)
    cur = conn.cursor()

    cur.execute('''SELECT (Housing_score + Living_Cost_score + Startup_score + Venture_Capital_score + Travel_score + Commute_score + Bus
                Safety_score + Healthcare_score + Education_score + Environmental_Quality_score + Economy_score + Taxation_score + In
                Leisure_Culture_score + Tolerance_score + Outdoors_score) / 17 FROM CityQoL WHERE name = ?''', (city,))
    avg = cur.fetchone()

    return avg[0]

get_city_avg('database.db', 'Anchorage')

```

The above code calculates the average of all of the quality of life scores for a city.



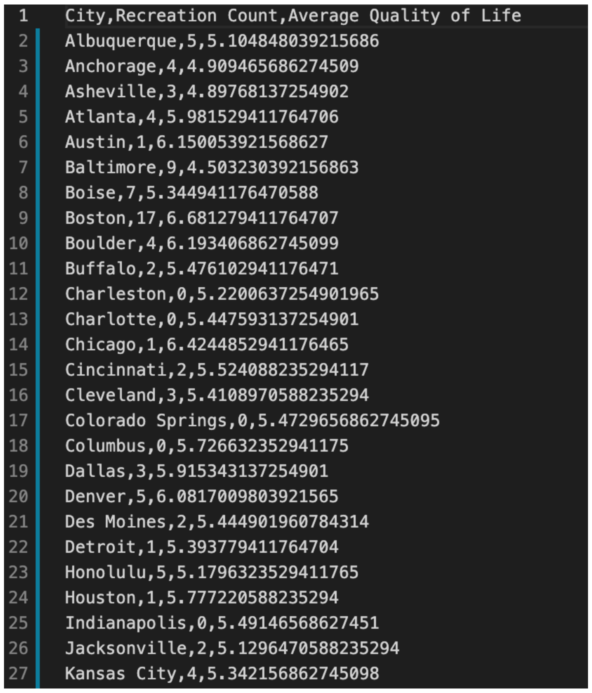
```python
def correlationCalc():
    base_path = os.path.abspath(os.path.dirname(__file__))
    full_path = os.path.join(base_path, "rec_count.csv")
    with open(full_path) as fileref:
        lines = fileref.readlines()
        rec_count_lst = []
        avg_lst = []
        for line in lines[1:]:
            row = line.split(',')
            rec_count = int(row[1])
            qol_avg = float(row[2])
            rounded = round(qol_avg,2)
            rec_count_lst.append(rec_count)
            avg_lst.append(rounded)

    corr = pearsonr(rec_count_lst, avg_lst)
    return corr
```

The above code finds the correlation between the recreation counts and the average quality of life scores for all cities

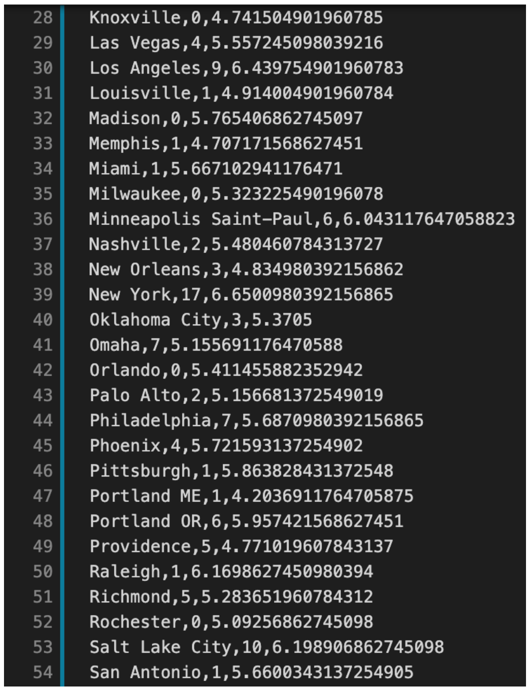


```python
def create_count_table(cur,conn,cities):
    header = ['City', 'Recreation Count', 'Average Quality of Life']
    with open('rec_count.csv', 'w') as f:
        writer = csv.writer(f)
        writer.writerow(header)
        for city in cities:
            longitude = get_long(cur,conn,city)
            latitude = get_lat(cur,conn,city)
            names = get_rec_data(longitude, latitude)
            try:
                count = len(names)
                cur.execute('SELECT name FROM CityQol WHERE name = ?', (city,))
                city = cur.fetchall()[0][0]
                cur.execute('SELECT avgQoL FROM CityQoL WHERE name = ?', (city,))
                avg = cur.fetchall()[0][0]
                data = [city, count, avg]
                writer.writerow(data)
            except:
                pass
    return None
```

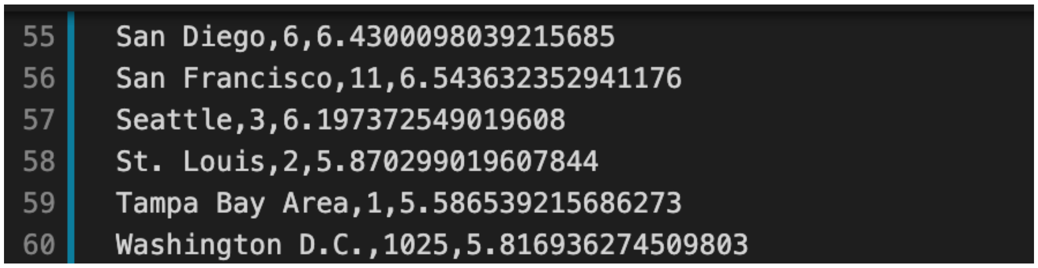
The above code writes the file that holds the calculations for the average quality of life and inserts the count of recreation areas. The screenshots below are what it contains.



```
City,Recreation Count,Average Quality of Life
Albuquerque,5,5.104848039215686
Anchorage,4,4.909465686274509
Asheville,3,4.89768137254902
Atlanta,4,5.981529411764706
Austin,1,6.150053921568627
Baltimore,9,4.503230392156863
Boise,7,5.344941176470588
Boston,17,6.681279411764707
Boulder,4,6.193406862745099
Buffalo,2,5.476102941176471
Charleston,0,5.2200637254901965
Charlotte,0,5.447593137254901
Chicago,1,6.4244852941176465
Cincinnati,2,5.524088235294117
Cleveland,3,5.4108970588235294
Colorado Springs,0,5.4729656862745095
Columbus,0,5.726632352941175
Dallas,3,5.915343137254901
Denver,5,6.0817009803921565
Des Moines,2,5.444901960784314
Detroit,1,5.393779411764704
Honolulu,5,5.1796323529411765
Houston,1,5.777220588235294
Indianapolis,0,5.49146568627451
Jacksonville,2,5.1296470588235294
Kansas City,4,5.342156862745098
```

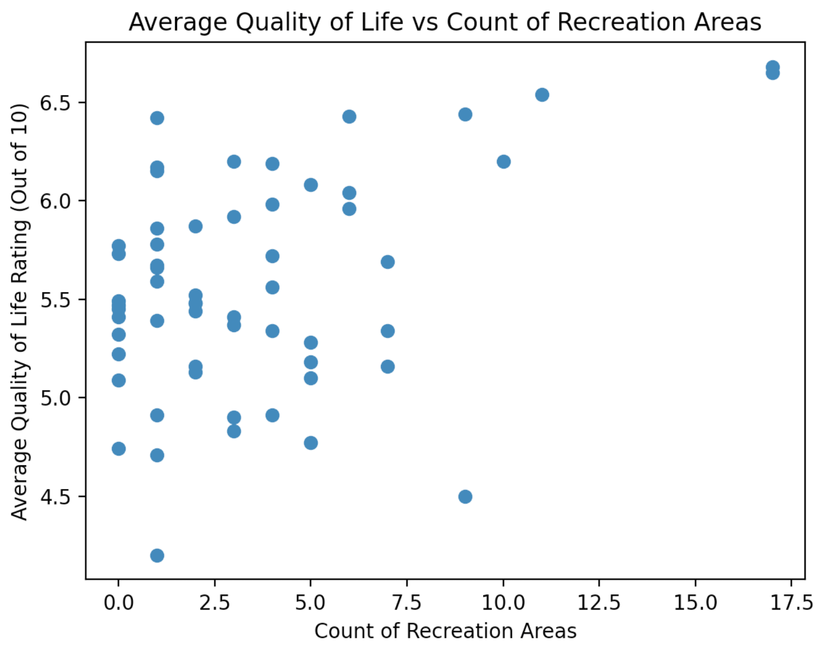


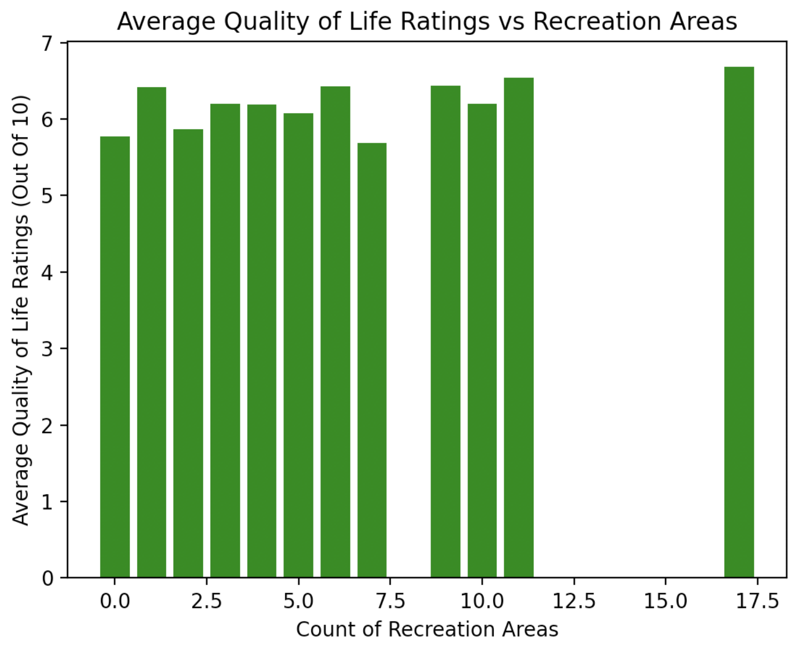
```
Knoxville,0,4.741504901960785
Las Vegas,4,5.557245098039216
Los Angeles,9,6.439754901960783
Louisville,1,4.914004901960784
Madison,0,5.765406862745097
Memphis,1,4.707171568627451
Miami,1,5.667102941176471
Milwaukee,0,5.323225490196078
Minneapolis Saint-Paul,6,6.043117647058823
Nashville,2,5.480460784313727
New Orleans,3,4.834980392156862
New York,17,6.6500980392156865
Oklahoma City,3,5.3705
Omaha,7,5.155691176470588
Orlando,0,5.411455882352942
Palo Alto,2,5.156681372549019
Philadelphia,7,5.6870980392156865
Phoenix,4,5.721593137254902
Pittsburgh,1,5.863828431372548
Portland ME,1,4.2036911764705875
Portland OR,6,5.957421568627451
Providence,5,4.771019607843137
Raleigh,1,6.1698627450980394
Richmond,5,5.283651960784312
Rochester,0,5.09256862745098
Salt Lake City,10,6.198906862745098
San Antonio,1,5.6600343137254905
```

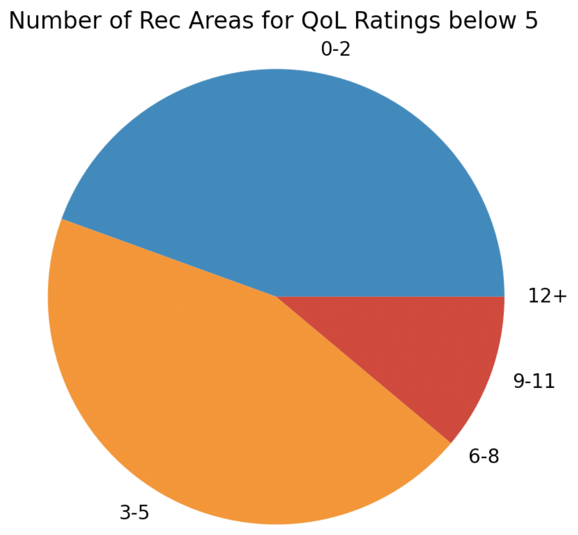


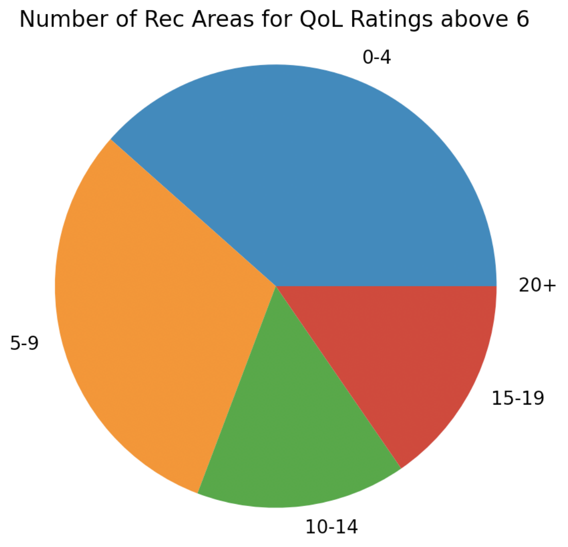
```
San Diego,6,6.4300098039215685
San Francisco,11,6.543632352941176
Seattle,3,6.197372549019608
St. Louis,2,5.870299019607844
Tampa Bay Area,1,5.586539215686273
Washington D.C.,1025,5.816936274509803
```

5. The visualization that you created (i.e. screenshot or image file) (10 points)


Average Quality of Life vs Count of Recreation Areas
Average Quality of Life Rating (Out of 10)
Count of Recreation Areas
4.5
5.0
5.5
6.0
6.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Average Quality of Life Ratings vs Recreation Areas
Average Quality of Life Ratings (Out Of 10)
Count of Recreation Areas
0
1
2
3
4
5
6
7
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5





6. Instructions for running your code (10 points)

Steps:

1. Run quality_of_life.py 4 times
2. Run QoL_calculations once
3. Run lat_long_new.py twice
4. Run rec-areas.py once
5. Then run plot.py once

7. Documentation for each function that you wrote. This includes the input and output for each function (20 points)

Lat_long_new.py

- `find_lat_long(city)` – this function was my original attempt at pulling the latitude and longitude for a specific city using Beautiful Soup from the website. Although this functions output did not get directly written into the database, It helped me write the next listed function
    - Input: `'Herrin'`
    - Output: (37.805065, -89.038292)

- `database_creation(db_filename, website_url)` – This function scrapes the given URL and writes the latitude and longitude coordinates of the cities included on the website into the database 'database.db'. To achieve this, I inspected the website and used Beautiful Soup. I added the tuples i was
    - Input: `'database.db'`, `'https://www.latlong.net/category/cities-236-15.html'`
    - Output: None (data went directly to the database)

Rec-areas.py

- `setUpDatabase(db_name)` – this function creates our database and returns the cursor and connection objects which is used to update the database
    - Input: `'database.db'`
    - Output: `cur, conn`

- `get_cities(cur,conn)` – this function uses the cursor and connection objects to select all of the city names from the CityQoL and returns a list of all of the needed cities
    - Input: `cur, conn`
    - Output: `city_list`

- `get_lat(cur,conn,name)` – this function uses the cursor and connection objects along with the name of the city to select the latitude value of that city from the cityInformation table in the database. It returns the latitude value if it is included in the CityQoL table and "Invalid city name" if it is not.
    - Input: `cur,conn,name`
    - Output: latitude of the city or "Invalid city name"

- `get_long(cur,conn,name)` – this function uses the cursor and connection objects along with the name of the city to select the longitude value of that city from the cityInformation table in the database. It returns the longitude value if it is included in the CityQoL table and "Invalid city name" if it is not.

  - Input: `cur,conn,name`
  - Output: latitude of the city or "Invalid city name"

- `get_rec_data(longitude, latitude, radius=50.0, limit=25)` – this function takes in the longitude and latitude values that were returned by the previous two functions and uses the default parameters for a radius of 50 miles and a limit of 25 recreation areas returned. If the longitude and latitude are valid, it will retrieve all of the recreation areas within a 50 mile radius of that city and return a list of all of the recreation area names. If not, it will print invalid city if the longitude and latitude are invalid or an exception if the retrieval did not work and in both cases, it will return None.
  - Input: `longitude, latitude, radius=50.0, limit=2`
  - Output: names of recreation areas or None

- `createRecIdTable(cur,conn,cities)` – this function takes in the cursor and connection objects along with the list of cities returned from get_cities. It will retrieve the longitude, latitude and names of rec areas for each city in the list to create an id table for each rec area returned.
  - Input: `longitude, latitude, radius=50.0, limit=2`
  - Output: None (creates recId table)

- `create_rec_table(cur,conn,cities)` – this function takes in the cursor and connection objects along with the list of cities returned from get_cities. It will retrieve the longitude, latitude and names of rec areas for each city in the list and insert them into the recAreas table based on the rec id created in the previous function (so no rec areas are duplicate strings if they overlap).
  - Input: `cur,conn,cities`
  - Output: None (creates recAreas table)

- `create_count_table(cur,conn,cities)` – this function takes in the cursor and connection objects along with the list of cities returned from get_cities. It creates a CSV file that contains the city name, the count of recreation areas within a 50 mile radius, and the average quality of life for that city retrieved from the tables.
  - Input: `cur,conn,cities`
  - Output: None (creates CSV file)

- `main()` – this function calls the setDatabase and get_cities functions to use their return values in the parameters for the functions that create the recId and recAreas tables. It also calls on the function that writes the CSV file and prints done so we know all of the data has been retrieved and placed in the appropriate places.

- Input: None
- Output: None

plot.py

- `def create_scatter_plot()` – this function retrieves the recreation count and average quality of life for each city in the CSV file (excluding outliers) and uses it to produce a scatter plot

- `def create_bar_chart()`– this function retrieves the recreation count and average quality of life for each city in the CSV file (excluding outliers) and uses it to produce a bar chart

- `def correlationCalc()` – this function retrieves the recreation count and average quality of life for each city in the CSV file (excluding outliers) and uses it to calculate the correlation coefficient of the data
    - Input: None
    - Output: `the correlation coefficient of the data in 'rec_count.csv'`

- `def createPieChart1()` – this function retrieves the recreation count and average quality of life for each city in the CSV file (excluding outliers) that has an average quality of life greater than 6 and uses it to produce a pie chart that is grouped by the number of recreation areas

- `def createPieChart2()` – this function retrieves the recreation count and average quality of life for each city in the CSV file (excluding outliers) that has an average quality of life less than 5 and uses it to produce a pie chart that is grouped by the number of recreation areas

quality_of_life.py

- `get_city_data(city)`– this function calls on the teleport quality of life API, requests the data for a single city, formats it in json, and then returns the relevant portion of the data (the quality of scores under the dictionary 'Categories')
    - Input: `a city name`
    - Output: `a list of dictionaries of each quality of life score for a city (the dictionary contains the score title and the score)`

- `createCityQOLTable(city_dict, db_filename)` - this function takes in a database file and creates the cur and conn objects for that database. It then creates the table CityQoL and its columns, and then selects 19 city names at a time by selecting the count number of the last city on the table and then adding 1 to access the next city. It gets the data for that city using the function above, accesses a score and assigns it to its variable name, and then inserts that data into the table. When this is run four times, the score data (excluding the average) for all 59 cities will be in the database.
    - Input: `a dictionary of city names, a database file name`
    - Output: `the CityQoL table complete with data for 19/20 columns (excludes data avgqol which will be added in a later function)`

- `main()` - this function includes the dictionary of city names we are using with the keys being the full, grammatically correct names and the values being the names formatted to be used in making a call to the api. It also calls the `createQoLCityTable` function and prints the word ‘finished’ to give a visual marker that the code is done running and can be run again.
    - Input: None
    - Output: None

QOL_calculations.py

- `createCityIdTable(city_dict, db_filename)` - this function takes in a database file and creates the cur and conn objects for that database. It then creates the table Cities that contains a column of city names and a column of corresponding ids
    - Input: `a dictionary of city names, a database file name`
    - Output: `a table of city names and their corresponding ids`

- `get_city_avg(db_filename, city)` - this function takes in a database file and creates the cur and conn objects for that database. It then selects the 17 quality of life scores for the (input) city from the database. It then calculates and returns the ovierall average quality of life score for a city.
    - Input: `a database file name, a city name`
    - Output: `the average quality of life score for that city`

- `addAvgQoL(db_filename, city_dict)` - this function takes in a database file and creates the cur and conn objects for that database. It then retrieves the city names from the keys of the input dictionary and loops through those names, calculating the average quality of life score for a city with the function above and then updating the database to add the average to the avgQoL in the database.
    - Input: `a database file name, a dictionary of city names`

- Output: `None`

- `main()` - this function includes the dictionary of city names we are using with the keys being the full, grammatically correct names and the values being the names formatted to be used in making a call to the api. It also calls the `createCityIdTable` function and `addAvgQoL`
    - Input: None
    - Output: None

8. You must also clearly document all resources you used. The documentation should be of the following form (20 points)

Date	**Issue Description**	**Location of Resource**	**Result: did it solve the issue?**
April 3	The original data scraping website did not have enough cities	Simple Google search for a more useful site	Yes. Finding this new site that had more of the cities we wanted to work with allowed Meghan to continue with her code.
April 9	Didn't know how to use the authentication key for the Recreation API	Stack Overflow/Professor Ericson	Yes. We were able to add it to our request as a key-value pair attached to the headers attribute
April 12	Forgot how to use the csv writer to create and write to a csv file	Stack Overflow	Yes. It provided the necessary information to allow us to write our csv file.
April 12	Realized some of the needed cities were still missing from the database	Amanda Hardy (discussion)	Yes. Amanda showed me how to create a list with the missing info and incorporate it into the database
April 15	Did not know how to create a scatter plot and reviewed how to create bar and pie charts	Matplotlib	Yes. This site provided all of the documentation to create the visualizations we needed
April 19	Accidently used DROP TABLE in my code	Amanda Hardy (discussion)	Yes. Amanda showed me how to add unique cities to the database without dropping my table each time.
April 19	Had to use an SQL method	Amanda Hardy (discussion)	Yes. We figured out how

	to calculate the averages rather than a python method (like a for loop)	and Stack Overflow	best to add up the quality of life scores of a city across a row rather than use an SQL command like AVG which finds the average across a column.