Erin Chai, Irene Chung, Katie Wu
Group Name: Erin/Katie/Irene Final Project
April 21, 2023

Final Project Report

Link to git repository: https://github.com/erinchai/SI206-Final-Project.git

1.The goals for your project (10 points)

Our original goal for this project was to see how COVID cases and weather affected city bike riders in two cities. We wanted to do this with the OpenWeather API, City Bikes, and Covid Tracking Project. We planned on collecting the number of daily city bike riders in selected regions as a proportion of the population in that city, the average temperature of each of those days per city, and the number of daily COVID cases in our selected cities. We then were going to calculate the weekly averages for each piece of data to plot as line graphs (this helps account for variants in bike riders depending on weekday). These would be 3 line graphs on one plot to represent one city, and we would do 3 cities. However, we couldn't find enough data for city bikes and covid data, so we ended up changing to see if humidity and temperature affects air quality level in two different cities. We ended up using 3 APIs (OpenWeather API, Weatherbit API, WorldWeather API) to gather information for temperature, humidity, and air quality index to see if temperature and humidity of two different locations have an effect on the air quality level. We decided to find the average weekly temperature between 01/01/2023 to 02/28/2023 between NewYork and Honolulu since they have very different climates.

2. The goals that were achieved (10 points)

We were able to successfully extract daily AQI, temperature, and humidity data from two cities (NYC and Honolulu) in Jan-Feb 2023. From this, we were able to first see how these measures compared individually, and if there was a correlation or not between air quality index and temperature. From our visualizations of both NYC and Honolulu, we were able to come to the conclusion they were not as correlated as we initially thought. This is likely due to many other factors that could affect the air quality index such as how industrialized a location is. In this particular example, NYC is a more industrialized city compared to Honolulu.

3. The problems that you faced (10 points)

We had many issues with finding APIs and websites that worked to gather data, so we had to change our plans many times. Initially we wanted to compare the number of city bikes and temperature to see if it affects the number of covid cases, but we couldn't really find any APIs with recent Covid-19 data. Additionally, the OpenWeather API didn't store historical weather

data from that far back. We then pivoted to working with humidity and air quality indexes, and also because there wasn't enough data for city bikes.

Another issue that we ran into was trying to run our data and gather the information from the API keys. We all had different ways of calling the data due to API limitations, so we had to figure out how to combine it together. Two of the API keys also had request limits, so we couldn't run it too often. We also had trouble combining the data into one database since we all created our own, and we then had to join it at the end.

4. Calculations from our Database

Text file for Temp:

temp.txt

```
Showing the average weekly temperatures for New York City from 1/1/23 - 2/28/23: [46, 38, 39, 40, 32, 41, 44, 39]
Showing the average weekly temperatures for New York City from 1/1/23 - 2/28/23: [74, 73, 77, 74, 73, 74, 73, 73]

```

Text file for AQI:

aqi.txt

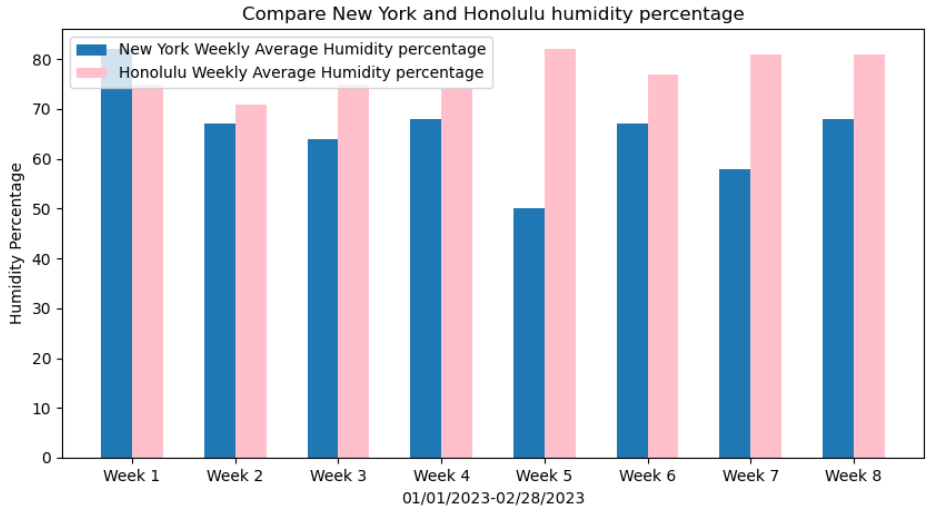
```
Showing the average weekly aqi for New York City from 1/1/23 - 2/28/23: [72, 50, 41, 41, 52, 57, 55, 46]
Showing the average weekly aqi for Honolulu from 1/1/23 - 2/28/23: [48, 38, 37, 35, 30, 38, 37, 38]

```

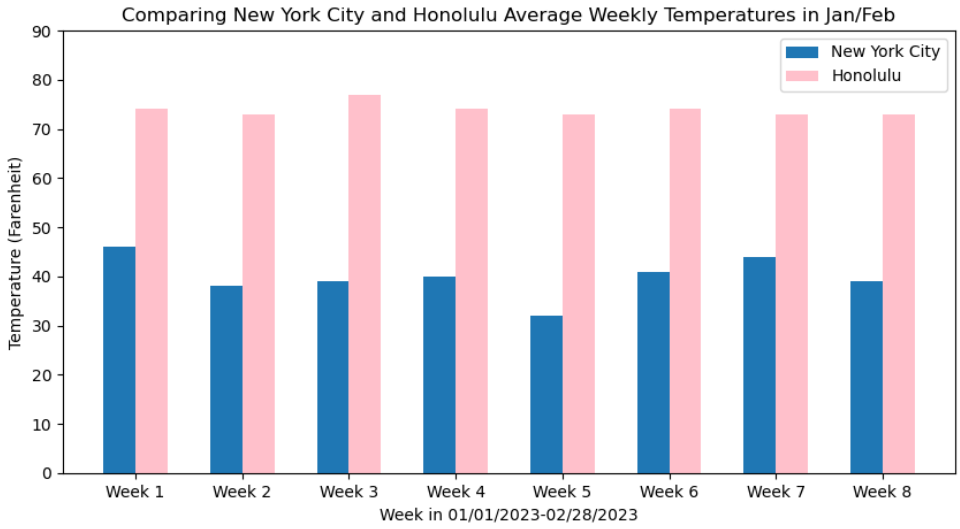
Test file for Humidity:

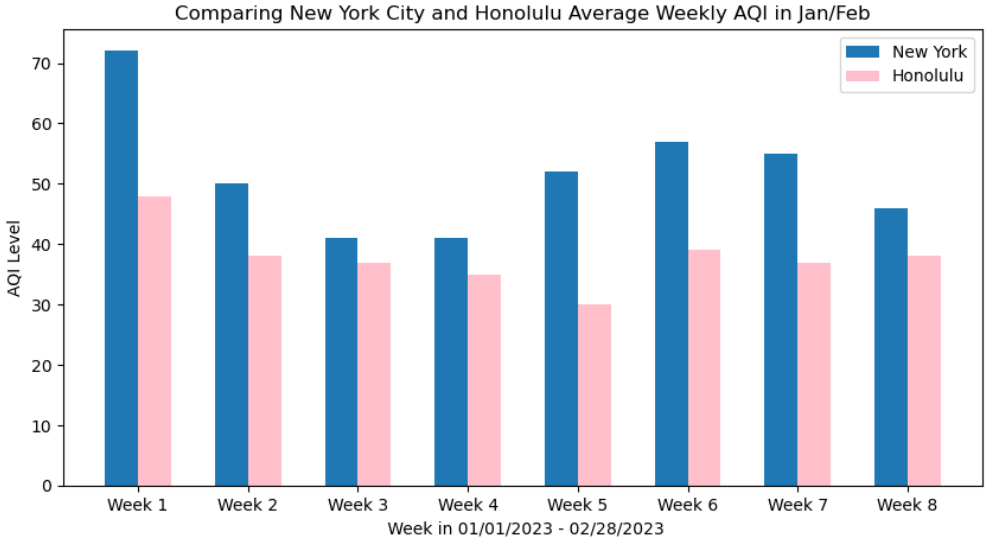
humidity.txt

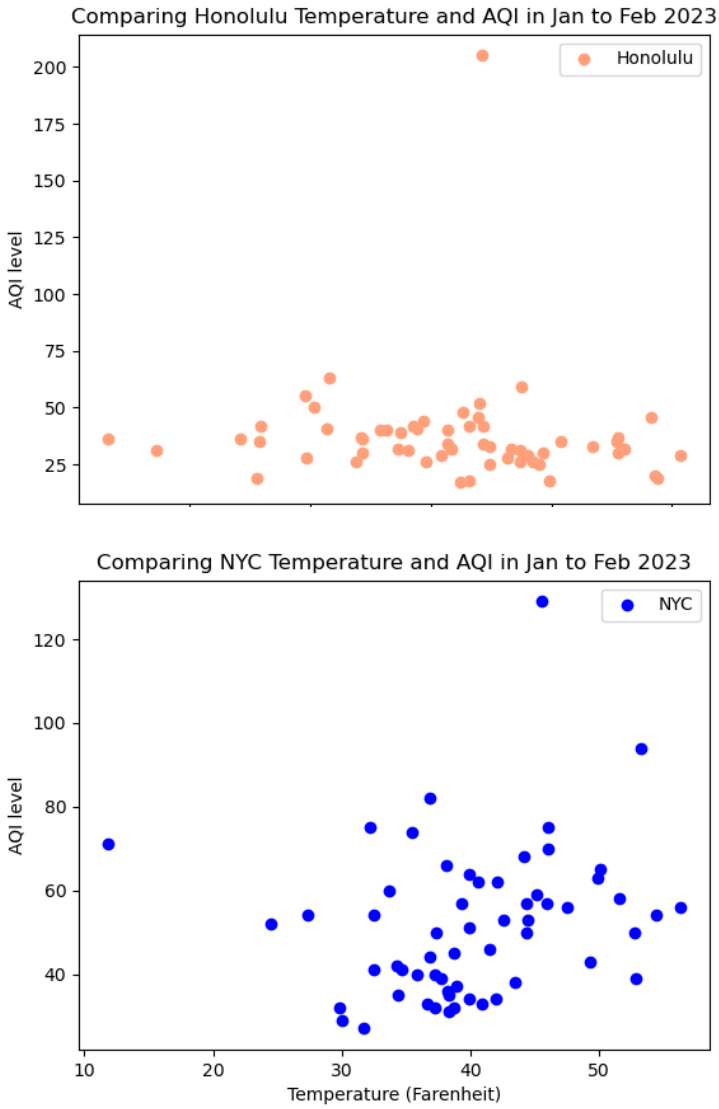
```
Showing the average weekly humidity percentage for New York City from 1/1/23 - 2/28/23: [82, 67, 63, 68, 50, 67, 58, 68]
Showing the average weekly humidity percentage for Honolulu from 1/1/23 - 2/28/23: [75, 71, 75, 74, 82, 77, 81, 81]

```

5. Visualizations Created




Comparing New York City and Honolulu Average Weekly Temperatures in Jan/Feb
New York City
Honolulu
Temperature (Farenheit)
0
10
20
30
40
50
60
70
80
90
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week in 01/01/2023-02/28/2023


Comparing New York City and Honolulu Average Weekly AQI in Jan/Feb
New York
Honolulu
AQI Level
0
10
20
30
40
50
60
70
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week in 01/01/2023 - 02/28/2023


Comparing Honolulu Temperature and AQI in Jan to Feb 2023
Honolulu
AQI level
200
175
150
125
100
75
50
25
Comparing NYC Temperature and AQI in Jan to Feb 2023
NYC
AQI level
120
100
80
60
40
10
20
30
40
50
Temperature (Farenheit)

6. Instructions for running your code

1. Air_quality.py
   Run this file first as it has the master date and city table. Press play 6 times. Each time you press play, less than 25 lines will populate the database.The lines will populate as 21, 21, 17, 21, 21, 17 because the API will only allow a max of 21 data taken at each request. The main function will also create the master_city table where New+York will be assigned an id of 1, and Honolulu will be 2. Lastly, it will also create the master_date table where it runs from 2023-01-01 to 2023-02-28 and assign each date from 1-59. The table will populate by 25, 25, 9. After you run it more than 6 times, it's just going to return None

2. Db2.py
   Press play 5 times. Each time you press play, 25 lines will populate the database, except the last run, which only populates 18 lines to get us to a total of 118 lines in the temperature table. It will first populate all of New York's Jan/Feb data followed by Honolulu's.

3. Humidity.py
   Press play 6 times. Each time you press play, 19 items or 20 items would be added to the database. It goes in the order (19,20,20,19,20,20) It would first populate New York's January and February data followed by Honolulu's. New York would be assigned an id of 1 in the city_id column and Honolulu would be assigned an id of 2. The date_id column would be assigned numbers from 1-59 chronologically and repeated for Honolulu's date data.

4. Calculations.py
   Run this file once. It will calculate the weekly averages of AQI, temperature, and humidity for Jan-Feb 2023 in New York City and Honolulu. It does this by getting the associated rows for that city from the database. It writes this data to three separate text files.

5. Visualizations.py
   Run this file once. It uses the data from calculations to create visualizations comparing each of these 3 weekly measures between the two cities. It also creates a visualization comparing the temperature and air quality index as scatterplots for each city.

7. Documentation for each function that you wrote. This includes describing the input and output for each function

Db2.py:

1. get_city_temps(lat, long):

Inputs: latitude and longitude of the city you want to find temperatures for
Outputs: a dictionary of daily temperatures from 1/1/2023 to 2/25/2023, with dates being keys and temperature in Fahrenheit as values
Purpose: This function grabs data from the OpenWeatherAPI, and cleans it into a JSON with only the information we want (daily temperatures for 1/1/23-2/28/23).

2. open_db(db_name):
   Inputs: name of the database you want to create
   Outputs: cursor and connect to be used when adding to the database
   Purpose: Setup to create/add to database

3. make_weather_table(data, cur, conn, start):
   Inputs: the daily temperature data from the two cities as a list, where the first 59 elements are all the daily NYC temperatures from 1/1/23-2/28/23, and the second 59 are the daily Honolulu temperatures from 1/1/23-2/28/23. Also takes in cursor and connection, as well as the start value (what row the database is on).
   Outputs: adds 25 rows to the database (except last run it will add the last 18), no outputs in vscode
   Purpose: Populate a table with the relevant information and including a city_id and date_id to later join with other tables

Air_quality.py

1. open_db(db_name):
   Inputs: name of the database you want to create
   Outputs: cursor and connect to be used when adding to the database
   Purpose: Setup to create/add to database

2. get_aqi_data(cur, con):
   Inputs: cursor and connection to make changes to database
   Outputs: None to python file, but in SQLite Browser it creates an aqi_level2 table with a corresponding city_id, date_id, and AQI level as an integer. It populates the table in increments of 21, 21, 17, 21, 21, 17. The first 59 are for New York, and the second half of the 59 are for Honolulu, creating a total of 118 pieces of data.

3. date_table(cur, con, row)
   Inputs: cursor and connection to make changes to database
   Outputs: none to python file, but in SQLite Browser it creates a master_date table with an integer and the corresponding date as a string. It populates the table in 25, 25, 9, for a total of 59 dates. It stops after 59 days, and each date is assigned to an integer from 1-59.
   Purpose: Avoid duplicate string data

4. city_table(cur, con):
   Inputs: cursor and connection to make changes to database

Outputs: none to python file, but in SQLite Browser it creates a master_city table with an integer and the corresponding city (New+York and Honolulu) as a string. New+York is 1 and Honolulu is 2.

5. main():
   Input: None
   Outputs: none to python file, but in SQLite Browser it creates a aqi_level2 table with an integer from 1-118 with a unique date and city. It combines all the three functions together, and it calls the date_table and city_table first, and then the get_aqi_data table. Since it creates the date and city table first, the get_aqi_data function is able to get the aqi level while also accessing the master_date and master_city table to assign the correct city_id and date_id into the table. The counter for the master_date table is initiated here to make sure that we aren't adding more than 25 dates at a time.

Humidity.py

1. open_db(db_name):
   Inputs: name of the database you want to create
   Outputs: cursor and connect to be used when adding to the database
   Purpose: Setup to create/add to database

2. get_humidity_data(cur, con):
   Inputs: cursor and connection to make changes to database
   Outputs: None to python file, but in SQLite Browser it creates an humidity_pct2 table with a corresponding city_id, date_id, and Humidity percentage as an integer. It populates the table in increments of 19, 20, 20, 19, 20, 20. The first 59 are for New York, and the second half of the 59 are for Honolulu, creating a total of 118 pieces of data. Once there are 118 items in the database, I put a try and except to return None.

3. main()
   Inputs: None
   Outputs: It first connects to database with open_db(db_name). The main function returns None to the Python file, but in SQLite Browser it creates and populates humidity_pct2 table with integer from 1-118 with a unique date and city.

Calculations.py

1. calc_wkly_avg_temp(cur, conn, city):
   Inputs: the cursor and connection, and the name of the city as a string
   Outputs: prints out the average weekly temperature for 8 weeks in 1/1/23-2/28/23 for the city as a list in temp.txt, also returns this list
   Purpose: make calculations that are used in visualizations

2. calc_wkly_avg_aqi(cur, conn, city):
   Inputs: the cursor and connection, and the name of the city as a string

Outputs: prints out the average weekly aqi for 8 weeks in 1/1/23-2/28/23 for the city as a list in aqi.txt, also returns this list
Purpose: make calculations that are used in visualizations

3. calc_wkly_avg_humidity(cur, conn, city):
   Inputs: the cursor and connection, and the name of the city as a string
   Outputs: prints out the average weekly humidity for 8 weeks in 1/1/23-2/28/23 for the city as a list in humidity.txt, also returns this list
   Purpose: make calculations that are used in visualizations

Visualizations.py

1. compare_temp_graph():
   Inputs: none
   Outputs: creates double bar chart comparing the 2 city's temperatures by week

2. compare_aqi_graph():
   Inputs: none
   Outputs: creates double bar chart comparing the 2 city's aqi by week

3. compare_humidity_graph():
   Inputs: none
   Outputs: creates double bar chart comparing the 2 city's humidity by week

4. compare_temp_aqi_nyc(cur, con):
   Inputs: cursor and connection to make changes to database
   Outputs: uses JOIN to grab data from the temperature table and aqi_level2 table for NYC and then creates a scatter plot visualization of the relationship between the two

5. compare_temp_aqi_honolulu(cur, con):
   Inputs: cursor and connection to make changes to database
   Outputs: uses JOIN to grab data from the temperature table and aqi_level2 table for NYC and then creates a scatter plot visualization of the relationship between the two

8. You must also clearly document all resources you used. In the following form:

- **Date, Issue Description, Location of Resource, Result**

Date	Issue Description	Location of Resource	Result
4/11	API change	http://api.citybik.es/v2/	It didn't work because there wasn't enough data for the US. It was less than 100 so we

			couldn't use it.
4/11	API change due to plan changes. We wanted to do COVID, city bikes, and temperature, however we couldn't find a usable COVID api.	api.covid19tracker.ca	It didn't work because we needed US city data but this API only included Canada's.
4/12	Couldn't figure out how to store data in a way that would be easiest to access when creating the database (JSON vs list, what item should be the key, etc)	Office Hours	Had a JSON with the date as a key and the cities with their temperatures as inner dicts, but changed to a list in order to get both cities data in one object in chronological orcer
4/12	To avoid using duplicate string data for the date, I looked into the datetime object for a column in the database	https://docs.python.org/3/c-api/datetime.html	I had lots of trouble getting it to print the correct date and couldn't debug it. So ultimately we ended up just creating a separate date table where each unique date was associated with an integer and used this instead.
4/15	I didn't know how to create a comparison bar graph.	Stack overflow	It was helpful to see how other people did it.
4/16	Do not know the syntax of drop table	Stackoverflow	I wanted to drop one of my tables but I don't know how the syntax worked. Stack Overflow was useful for informing the syntax. I then dropped the table in DB Browser.