SI 206 FINAL PROJECT REPORT:

File1: *Air Quality Index Data API:*

https://api.waqi.info/feed/here?token={API key}

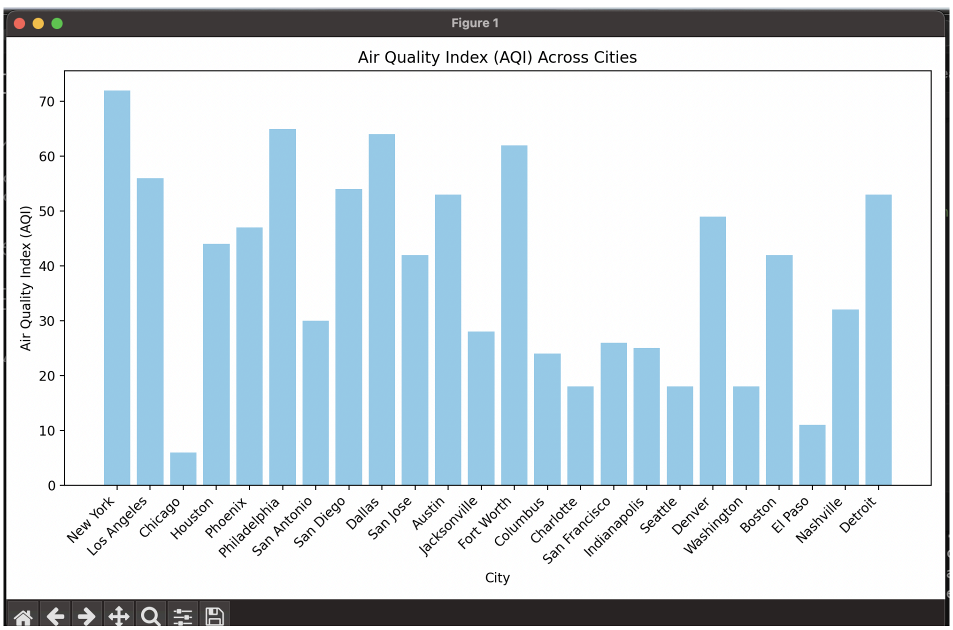
File 2: *OpenWeatherMap API:*

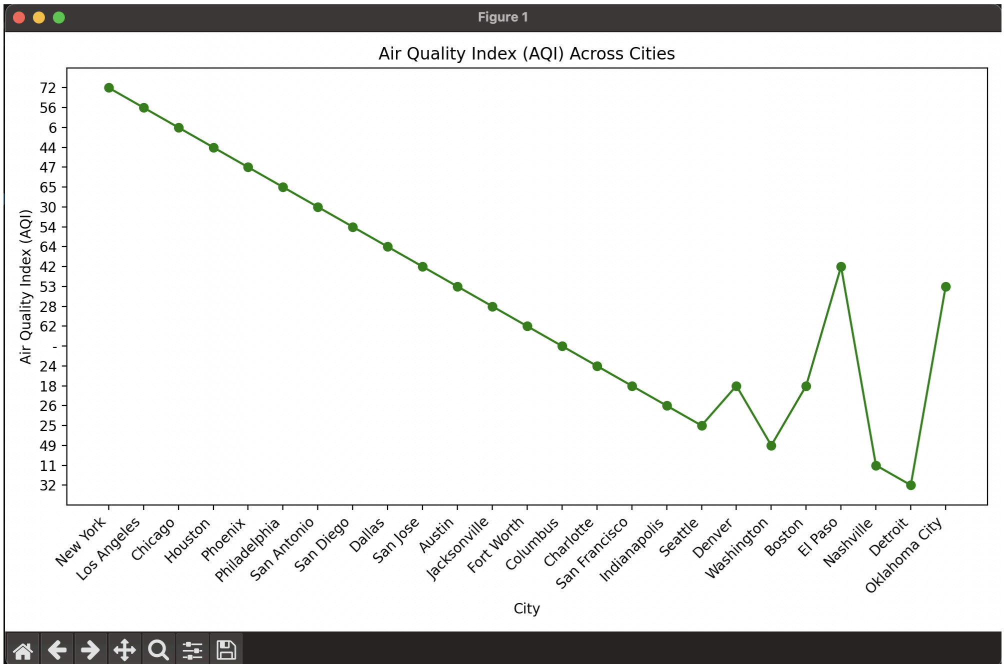
https://api.openweathermap.org/data/3.0/onecall?lat={lat}&lon={lon}&exclude={part}&appid={API key}

File 3: *Weather rapidAPI by ninjas:*

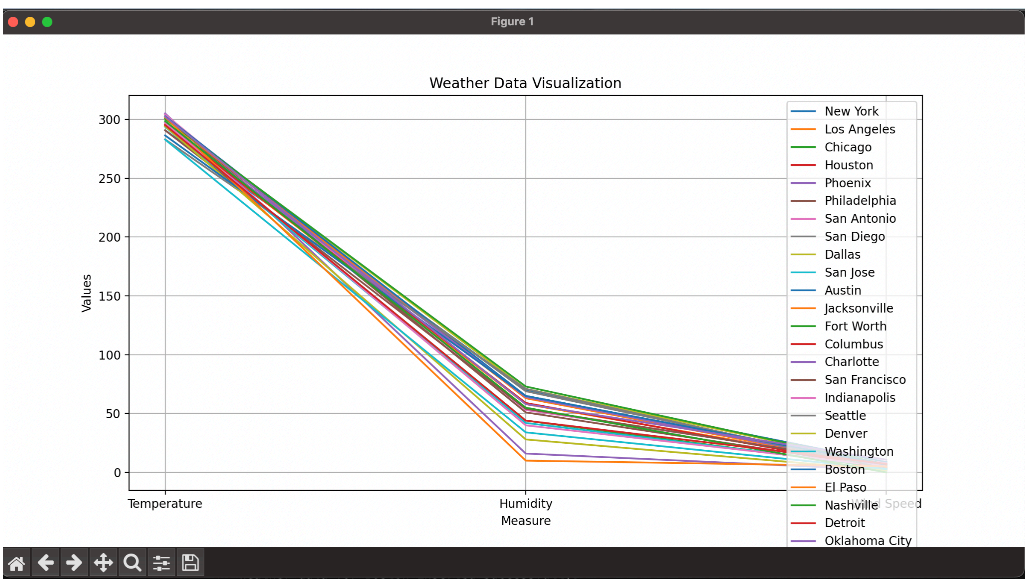
https://weather-by-api-ninjas.p.rapidapi.com/v1/weather

File 1 visualizations:

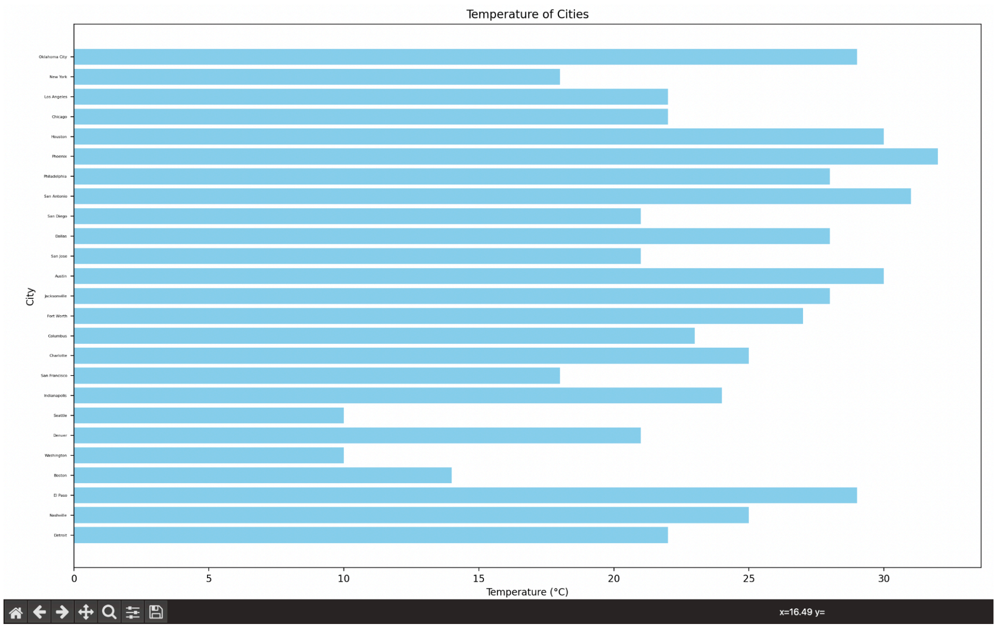


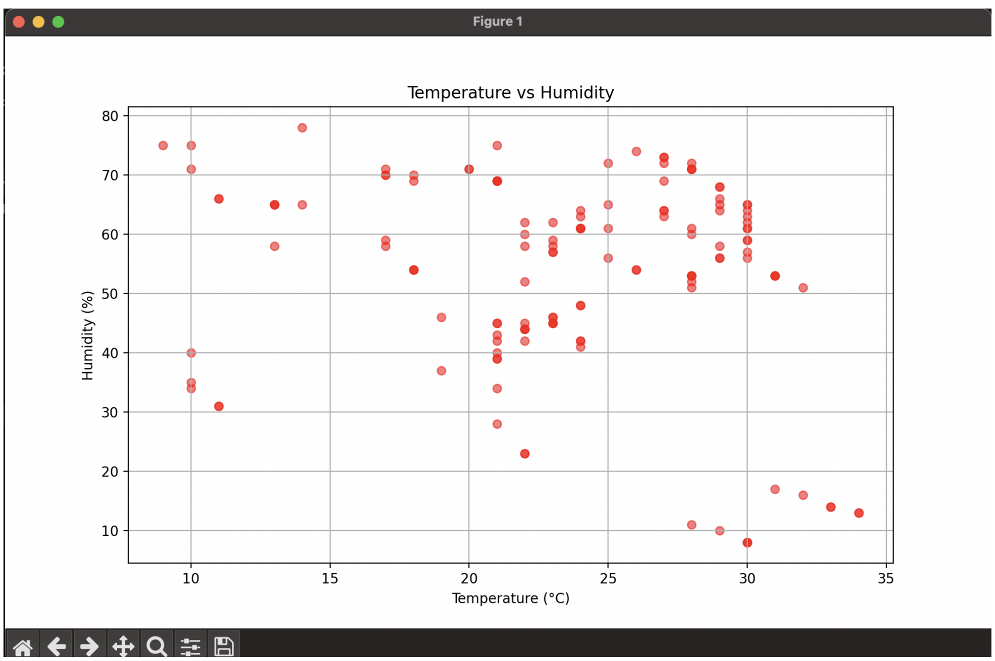


File 2 visualization:



File 3 visualizations:





Calculations:

calculations > main

```python
import sqlite3

# Function to perform calculations and database join
def process_data():
    conn = sqlite3.connect('weather_data.db')
    c = conn.cursor()

    # Perform calculations from air_quality_data table
    c.execute("SELECT city, AVG(aqi) FROM air_quality_data GROUP BY city")
    average_aqi_data = c.fetchall()

    # Perform calculations from weather_data table
    c.execute("SELECT city, AVG(feels_like) FROM weather_data GROUP BY city")
    average_feelslike_data = c.fetchall()


    # Perform calculations from weather table
    conn = sqlite3.connect('weather_data.db')
    c.execute("SELECT city, AVG(wind_speed) FROM weather GROUP BY city")
    average_weather_data = c.fetchall()


    # Perform database join and additional calculations
    c.execute('''SELECT a.city, a.avg_aqi, b.avg_wind_speed
                 FROM (SELECT city, AVG(aqi) AS avg_aqi FROM air_quality_data GROUP BY city) AS a
                 JOIN (SELECT city, AVG(wind_speed) AS avg_wind_speed
                       FROM weather GROUP BY city) AS b
                 ON a.city = b.city''')
    joined_data = c.fetchall()

    conn.close()
    return average_aqi_data, average_weather_data, average_feelslike_data, joined_data

```

average_aqi_data.txt

```
('Austin', 63.0)
('Boston', 21.0)
('Charlotte', 31.0)
('Chicago', 2.0)
('Columbus', 0.0)
('Dallas', 58.0)
('Denver', 26.0)
('Detroit', 38.0)
('El Paso', 52.0)
('Fort Worth', 58.0)
('Houston', 58.0)
('Indianapolis', 28.0)
('Jacksonville', 35.0)
('Los Angeles', 60.0)
('Nashville', 14.0)
('New York', 58.0)
('Oklahoma City', 63.0)
('Philadelphia', 71.0)
('Phoenix', 43.0)
('San Antonio', 30.0)
('San Diego', 51.0)
('San Francisco', 26.0)
('San Jose', 46.0)
('Seattle', 25.0)
('Washington', 40.0)
```

average_feelslike_data.txt

```
('Charlotte', 31.0)
('Chicago', 2.0)
('Columbus', 0.0)
('Dallas', 58.0)
('Denver', 26.0)
('Detroit', 38.0)
('El Paso', 52.0)
('Fort Worth', 58.0)
('Houston', 58.0)
('Indianapolis', 28.0)
('Jacksonville', 35.0)
('Los Angeles', 60.0)
('Nashville', 14.0)
('New York', 58.0)
('Oklahoma City', 63.0)
('Philadelphia', 71.0)
('Phoenix', 43.0)
('San Antonio', 30.0)
('San Diego', 51.0)
('San Francisco', 26.0)
('San Jose', 46.0)
('Seattle', 25.0)
('Washington', 40.0)
```

average_windspeed_data.txt

```
('Albuquerque', 4.702857142857142)
('Anaheim', 3.527142857142857)
('Anchorage', 4.482857142857142)
('Arlington', 5.732857142857142)
('Atlanta', 3.4528571428571433)
('Aurora', 3.3057142857142856)
('Austin', 4.69375)
('Bakersfield', 4.192857142857144)
('Baltimore', 5.512857142857143)
('Baton Rouge', 0.68)
('Boise', 1.0571428571428572)
('Boston', 5.951874999999999)
('Buffalo', 3.527142857142857)
('Chandler', 2.382857142857143)
('Charlotte', 5.4625)
('Chesapeake', 6.2442857142857155)
('Chicago', 4.402777777777779)
('Chula Vista', 3.898571428571429)
('Cincinnati', 3.914285714285715)
('Cleveland', 5.067142857142857)
('Colorado Springs', 8.381428571428572)
('Columbus', 3.9225000000000001)
('Corpus Christi', 7.792857142857143)
('Dallas', 6.074375000000001)
('Denver', 1.8456249999999994)
```

Plan, Problems, and Work Timeline:

In our original project plan we had projected our goal as collecting weather data from two external weather sources: National Weather Service and Open Weather Map. We planned to calculate the average, maximum, and minimum temperature for 100 cities over a selected time period. From there our plan was to create three separate visualization charts that coincide with the output of our various functions calculating specific weather data.

After turning in our original plan, is where we ran into some of our initial problems: the first being that in order to get full points, since we have three members in our group we needed to pull information from *three different* APIs/Websites. So, we decided to choose our third source: Weather by API Ninjas. The reason behind this decision lies within our second problem.

Many weather APIs have requests that require latitude and longitude coordinates for specific weather stations. Using our predefined list of 100 US cities would require us seeking out 100 pairs of coordinates on our own time, which seemed counterproductive. To address this issue we decided to change the National Weather Service source to Air Quality Index API which collected its data without using latitudinal and longitudinal coordinates. The three weather APIs we ended up using allow requests simply by the city name in string form, which made iterating through our list of cities a lot simpler.

In our final project in File 1, we use the Air Quality API to create a database of 100 cities with information about the Station Name for that city, AQI (Air Quality Index) number, and the dominant pollutant of that specified city. We also included a data visualization of some major cities from our database and their according AQI numbers.

For File 2, we used the Open Weather Map API to fetch weather data from the API and create a database and visualization of temperature, humidity, and wind speed of the specified city. We then created a line graph visualization of 25 of those cities and their according temperature, humidity, and wind speed.

In File 3, we use Weather API by Ninjas to fetch weather data from the API and create a database that we can use for statistical comparison. We created a third database with the temperature, feels like, humidity, and wind speed to hopefully discover the most accurate measurements across various cities.

In our Calculation file we calculate the average AQI, average feels like temperature, average temperature, humidity, and windpeed across all three tables and calculated that into three separate file named average_weather_data, average_feelslike_data, and average_aqi_data.

Additionally, another big problem we encountered was our connection to updating the main branch within our repository. Only Emily’s code was able to correctly update, push/pull, and collaborate on the main repository. So we all had to take turns on Emily’s computer and work together to time out all of our pushes and commits. Thankfully, we were able to solve this issue with some strong communication and planning and our repository should be up to date.

Instructions for Running the Code:

Running our code to see the 3 separate data visualizations is a relatively simple process on the user end of things. Each API and its respective visualization are each contained in a separate file for organization, named file1, file2, and file3, respectively. If you want to see the data visualization for our first API (air quality), you simply open our repository and run file1.

Our calculations from the collected data are stored in our final file, file4. Upon running file4, the user can see the results of our data calculations clearly printed.

Function Documentation:

File1:

- fetch_data_from_api(api_key, city_name):
  - INPUT:
    - api_key (str): the API key required for accessing the air quality data API
    - city_name (str): The name of the city whose air quality data you are trying to fetch
  - OUTPUT:
    - Returns a dictionary containing air quality data for the specified city if the API call is successful. Otherwise returns None.
- create_database_and_insert_data(api_key, city_name):
  - INPUT:
    - api_key (str): the API key required for accessing the air quality data API
    - city_name (str): The name of the city whose air quality data you are trying to fetch
  - OUTPUT:
    - None
  - DESCRIPTION:
    - Fetches air quality data for the specified city using the fetch_data_from_api function and inserts it into an SQLite database named air_quality.db. It creates a new table named air_quality_data if it doesn't exist already.
- plot_air_quality(cities, aqi_values):
  - INPUT:
    - Cities (list of strings): a list of city names for which air quality data is available
    - Aqi_values (list of int): A list of corresponding AQI (Air Quality Index) values for each city.
  - OUTPUT:
    - None
  - DESCRIPTION:
    - Plots a bar chart showing the Air Quality Index (AQI) across different cities. The x-axis represents the cities, and the y-axis represents the AQI values. It skips non-numeric AQI values while plotting.
- main()
  - INPUT: none
  - OUTPUT: none
  - DESCRIPTION:

        - The main function of the program. It initializes necessary variables, fetches air quality data for 25 cities using the create_database_and_insert_data function, retrieves the inserted data from the SQLite database, and then plots the air quality data using the plot_air_quality function

File2:

- fetch_weather_data(api_key, city_name):
    - INPUT:
        - api_key(str): The API key required for accessing the OpenWeatherMap API
        - City_name (str): The name of the city for which weather data needs to be fetched.
    - OUTPUT:
        - Returns a dictionary containing weather data for the specified city if the API call is successful. If the API call fails or no data is available, it returns None.
- create_database_and_insert_data(api_key, city_name):
    - INPUT:
        - api_key (str): The API key required for accessing the OpenWeatherMap API.
        - city_name (str): The name of the city for which weather data needs to be fetched and inserted into the database.
    - OUTPUT:
        - None
    - DESCRIPTION:
        - Fetches weather data for the specified city using the fetch_weather_data function and inserts it into an SQLite database named weather_data.db. It creates a new table named weather if it doesn't exist already.
- Plot_weather_data(city_data):
    - INPUT:
        - City_data (dict): A dictionary containing weather data for multiple cities. The keys are city names, and the values are lists containing temperature, humidity, and wind speed data
    - OUTPUT: None
    - DESCRIPTION: Plots a line graph to visualize weather data (temperature, humidity, and wind speed) for each city in the city_data dictionary. Each city's data is represented by a separate line on the graph.
- main()
    - INPUT: none
    - OUTPUT: none
    - DESCRIPTION: The main function of the program. It initializes necessary variables, fetches weather data for the first 25 cities using the fetch_weather_data function, inserts the data into the SQLite database using the

create_database_and_insert_weather function, retrieves the inserted data, and then plots the weather data using the plot_weather_data function.

File3:

- fetch_weather_data(city):
  - INPUT:
    - City (str): The name of the city for which weather data needs to be fetched.
  - OUTPUT:
    - Returns a dictionary containing weather data for the specified city if the API call is successful. If the API call fails or no data is available, it returns None.
- create_database_and_insert_data(cities):
  - INPUT:
    - Cities (list of strings): A list of city names for which weather data needs to be fetched and inserted into the database
  - OUTPUT: None
  - DESCRIPTION:
    - Fetches weather data for the specified cities using the fetch_weather_data function and inserts it into an SQLite database named weather_data.db. It creates a new table named weather_data if it doesn't exist already.
- print_weather_data()
  - INPUT: none
  - OUTPUT: none
  - DESCRIPTION:
    - Prints all the weather data stored in the weather_data table of the SQLite database.
- print_and_visualize_weather_data():
  - INPUT: none
  - OUTPUT: none
  - DESCRIPTION:
    - Prints the city names and their corresponding temperatures stored in the weather_data table of the SQLite database. Additionally, it plots a horizontal bar chart to visualize the temperatures of cities
- main():
  - INPUT: none
  - OUTPUT: none
  - DESCRIPTION:
    - The main function of the program. It fetches weather data for 100 cities from the cities_list and inserts them into the SQLite database. After inserting each batch of 25 cities, it prints and visualizes the weather data using the print_and_visualize_weather_data function. Finally, it exits the program after fetching data for 100 cities.

Calculations file:

- process_data():
    - DESCRIPTION: This function performs calculations and database joins using data from the 'weather_data.db' SQLite database
    - INPUTS: none
    - OUTPUTS: 4 lists of tuples:
        - average_aqi_data: Contains tuples with city names and their corresponding average AQI (Air Quality Index) values.
        - average_weather_data: Contains tuples with city names and their corresponding average wind speeds.
        - average_feelslike_data: Contains tuples with city names and their corresponding average "feels like" temperatures.
        - joined_data: Contains tuples with city names, average AQI values, and average wind speeds for cities where data is available in both air quality and weather tables.
- write_to_text_file(data, filename)
    - DESCRIPTION: This function writes the calculated data to a text file
    - INPUTS:
        - data: List of tuples containing the calculated data.
        - filename: Name of the text file to which the data will be written.
    - OUTPUTS: No explicit output, but data is written to a text file
- main():
    - DESCRIPTION: The main function of the program. It orchestrates the processing of data and writing to text files
    - INPUTS: None
    - OUTPUT: No explicit output, but prints message saying data has been processed and written to text files

Resources Used:

Date	Issue Description	Location of Resource	Result
4/28/2024	When working with File 2, I was trying to create the visualization my original plan was to create a stacked bar chart where	Chat GPT	Attempted to use chat GPT to help with spacing issues, but eventually decided to scrap the whole bar chart

	one bar was wind speed and the other humidity, but the spacing of the bars was not correct and additionally my x-axis was iterating through the list of 100 cities 25 times so nothing was evenly spaced or clearly marked.		visualization in general and instead do a line graph plotting all thre weather measurements for File 2 and its according city.
4/30/2024	Originally we accidentally created two databases, instead of one database with three tables, and we could not seem to figure out how to join all three tables into one database	ChatGPT	Used chat to debug our code and turns out all we needed to change was one line of code and we merged tables into one database.
4/26/2024	When creating our data visualizations we had trouble with fixing how it looked, spacing, xticks, fontsize, etc.	ChatGPT	Asked chat how we could correctly and aesthetically add three pieces of information per one city. Turns out we needed to create a for loop to our visualization function.