Naman Mathur and Nikitha Subramaniam
SI 206 Winter 2023
April 21, 2023

Final Project Report

Github link: https://github.com/namansm/SI206-Final-Project

Goals Planned

We initially planned to use the COVID19 API for covid case data around the world from March 2020 to March 2021 (including date of the case, longitude, latitude, and country name). We also planned on using the Oikoweather API for weather data, which would use the dates, longitudes, and latitudes obtained from the covid api to retrieve daily weather information (which includes average temp, total precipitation, and wind speed). We wanted to use the information from both these APIs to see if there was a correlation between the type of weather in a certain area and an increase/decrease in the number of covid cases. Our plan for determining the severity of weather is if the wind speed is at 65 kph, if the temperature is above 35 C or below 0 C, or if the total precipitation is above 48 mm per day.

Goals Achieved

We were able to achieve our goal of using both Covid and weather APIs to obtain information about type of weather (temperature, wind speed, and precipitation) in correlation to the number of covid cases. The countries we ended up picking were India and Switzerland since we wanted 2 different countries with very different weather as well a similar order of magnitude of population density. One main difference between our planned goal was the API we used for weather. We ended up using Meteostat instead of Oikoweather since it was simpler to use and allowed us more free API calls per month. As far as visualization and calculations, we were able to successfully plot a relationship between severe weather and overall number of covid cases (see below for graph).

Problems Faced

- The initial weather API we had planned to use was very confusing to use/interpret and also had a very limited amount of data that we could obtain monthly for free.
    - We ended up switching to another free weather API (Meteostat) that was easier to use and had more free monthly calls.

- Even using the new weather API, some locations would return nothing for the data (what would normally be a populated list of dictionaries would just be an empty list). We solved this problem by ignoring those anomalies, as most longitudes and latitudes came up correctly.
- Additionally for Meteostat, it would sometimes not return anything when we called the API, resulting in more time being utilized to obtain all the data we needed.

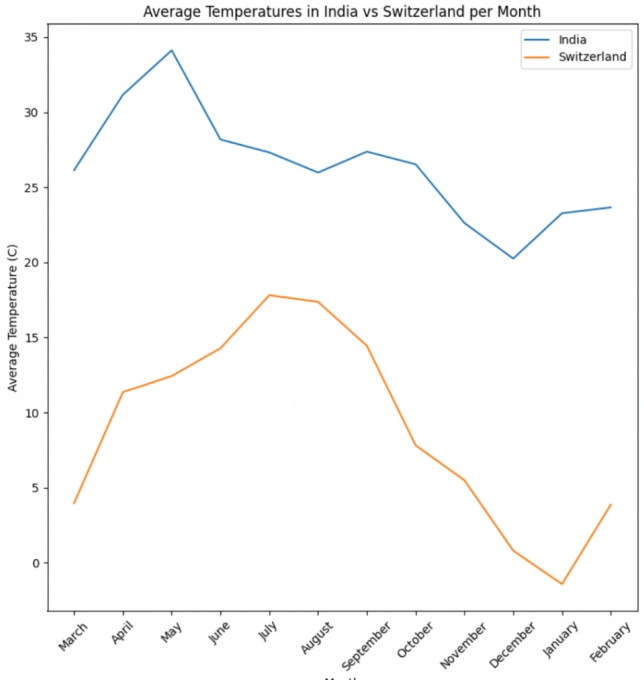
Calculations From Database

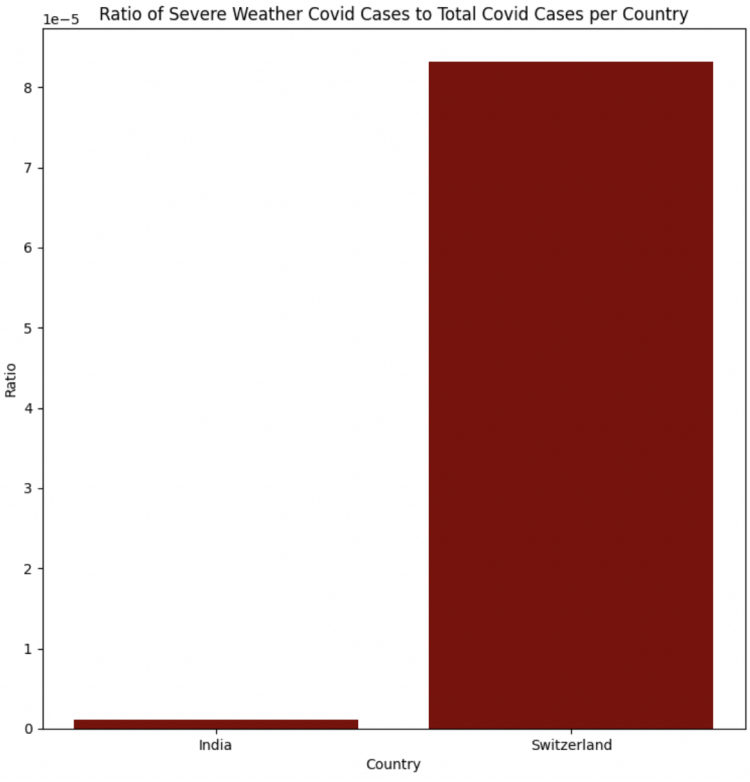
- We calculated the average temperature, wind speed, and rainfall for each country (India and Switzerland). We decided to write this data to a csv file and also plot the data for temperature specifically (see below). We used a SQL inner join for this to obtain the country name for easier visualization.
- output.csv

```
Country,Month,Average Daily Temperature (C),Average Daily Wind Speed (kph),Average Daily Rainfall (mm)
India,March,26.141935483870967,5.54516129032258,0.0
India,April,31.163333333333334,6.206666666666667,
India,May,34.11612903225806,7.53225806451613,3.0
India,June,28.18333333333333,7.309999999999998,5.42
India,July,27.31612903225807,5.3032258064516125,37.166666666666664
India,August,25.97741935483872,9.180645161290322,10.879999999999999
India,September,27.369999999999994,3.8066666666666675,7.938461538461538
India,October,26.525806451612905,3.5903225806451613,2.58
India,November,22.61666666666667,3.2800000000000002,
India,December,20.25161290322581,2.7387096774193544,
India,January,23.270967741935483,6.048387096774195,0.0033333333333333335
India,February,23.66428571428571,6.503571428571429,0.33571428571428574
India,March,28.251612903225798,6.509677419354839,0.2129032258064516
Switzerland,March,3.9699999999999998,4.960000000000002,2.9033333333333333
Switzerland,April,11.359999999999998,5.403333333333332,2.775862068965517
Switzerland,May,12.425806451612905,4.525806451612903,3.270000000000001
Switzerland,June,14.283333333333335,4.3100000000000005,4.620689655172414
Switzerland,July,17.806451612903224,4.573333333333333,5.163333333333335
Switzerland,August,17.367741935483874,4.166666666666666,5.7700000000000005
Switzerland,September,14.446666666666665,3.7689655172413805,3.003448275862069
Switzerland,October,7.819354838709677,4.370967741935484,5.138709677419356
Switzerland,November,5.493333333333333,3.540000000000001,14.593103448275867
Switzerland,December,0.8096774193548386,4.258064516129032,20.71612903225807
Switzerland,January,-1.4161290322580637,4.464516129032257,9.263333333333334
Switzerland,February,3.8749999999999996,5.521428571428572,3.896428571428572
Switzerland,March,4.126666666666667,4.91,5.866666666666666
```

- The second calculation we did was the count of severe weather instances as defined in the intro section above. We compared this to the total number of cases (depicted by the max amount for number of cases total).
    - **CALCULATED VALUES:** [('India', 14, 12221665), ('Switzerland', 50, 601124)]
        - Where the first value is the country, the second value is the number of severe weather instances for that country over the year, and the third value is the total number of covid cases for that country over the year

Visualizations Created





How to Run our Code

- To populate the database, first run covid_api.py as many times as needed to populate the "Covid" table.
- Once the Covid table is populated, run weather_api.py to populate the "weather" table with the dates and locations from "Covid".
    - NOTE: for weather API you must have a valid API key with Rapid-API
- To execute the calculations and create the visualizations, simply run calculations.py. It will output calculations to 'output.csv" and create visualizations for the data.

Documentation for Functions Written

- **File Name:** covid_api.py
    - **def open_database(db_name):**
        - Input:
            - *db_name,* name of database file
        - Output:
            - *cur*, a cursor object to traverse database and *conn*, connects python code to database
        - Purpose: Opens database file and creates cursor and connection objects so we can connect to database and traverse it later in the program
    - **def main():**
        - Input: None
        - Output: None
        - Purpose: Drives program, calls all functions to run program in order
    - **def load_api_data():**
        - Input: None
        - Output:
            - *d,* a dictionary of the decoded JSON
        - Purpose: Returns a dictionary representation of the decoded JSON. If the search is unsuccessful, the function prints "Exception!" and returns None.
    - **def make_covid_table(data, cur, conn):**
        - Input:
            - *data*, the necessary API data loaded into a dictionary
            - *cur,* cursor object to traverse database
            - *conn,* connects python code to databases
        - Output: None
        - Purpose: Uses data from API to create table of values in database

- **File Name:** weather_api.py
    - **def open_database(db_name):**

        - Input:
            - *db_name,* name of database file
        - Output:
            - *cur*, a cursor object to traverse database and *conn*, connects python code to database
        - Purpose: Opens database file and creates cursor and connection objects so we can connect to database and traverse it later in the program
    - **def get_covid_data(cur, conn):**
        - Input:
            - *cur,* a cursor object to traverse the database
            - *conn,* establishes connection between python code and database
        - Output:
            - *d*, a list of dictionaries containing information about temperature in each country based on latitude and longitude
        - Purpose: Retrieves data from covid table in database and formats it into a list of dictionary
    - **def load_weather_data(data, cur, conn):**
        - Input:
            - *data*, the list of dictionaries returned from get_covid_data()
            - *cur,* a cursor object to traverse the database
            - *conn,* establishes connection between python code and database
        - Output:
            - *No output*
    - **def main():**
        - Input: None
        - Output: None
        - Purpose: Drives program, calls all functions to run program in order
- **File Name:** calculations.py
    - **def open_database(db_name):**
        - Input:
            - *db_name,* name of database file
        - Output:
            - *cur*, a cursor object to traverse database and *conn*, connects python code to database
        - Purpose: Opens database file and creates cursor and connection objects so we can connect to database and traverse it later in the program
    - **def calc_and_write_averages(outfile, cur, conn)**
        - Input:
            - *outfile*, the name of the csv file that the data should be written to
            - *cur,* a cursor object to traverse the database

        - *conn,* establishes connection between python code and database
    - Output:
        - *data*, a list of tuples that contains (country, month, average daily temp, average daily wind speed, average daily rain). The reason for this is to facilitate visualization.
    - Purpose: Calculates daily weather averages for each month for each country. The function then writes this data to a csv file to be accessed later.
- **def plot_month_vs_temps(filename):**
    - Input:
        - *filename,* the name of the csv file of weather data for each month
    - Output: None
    - Purpose: Plots each country's monthly temperature as a line graph to compare the two countries' weather patterns over time. Saves graph to a png file.
- **def calc_severe(cur, conn)**
    - Input:
        - *cur,* a cursor object to traverse the database
        - *conn,* establishes connection between python code and database
    - Output:
        - *data*, a list of tuples that contains (country, # of severe weather instances, # of covid cases in the year).
    - Purpose: Calculates severe weather instances and returns tuples to easily identify the number of covid cases in severe weather out of total covid cases per country. The reason for this function is to facilitate visualization.
- **def plot_severe_vs_total_cases(data):**
    - Input:
        - *data,* a list of tuples specifying each country's covid case counts in instances of severe weather versus overall
    - Output: None
    - Purpose: Calculates ratio of cases in severe weather versus cases overall, and plots ratio as a bar graph for both India and Switzerland. Saves graph to a png file.
- **def main():**
    - Input: None
    - Output: None
    - Purpose: Drives program, calls all functions to run program in order

Resources Used

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
4/13/2023	Finding free APIs that we could use for the project	https://github.com/public-apis/public-apis	Yes, we found both the Oikoweather API and the Covid-19 API
4/16/2023	Finding out information on determining the severity of weather	NOAA, USGS (for rainfall), and National Geographic	Yes, we found adequate methods of determining whether a day had severe weather or not
4/16/2023	Utilizing documentation for Oikoweather API	https://docs.oikolab.com/#1-introduction	Yes, we learned how to use the Oikoweather API but did not end up using it
4/16/2023	Utilizing documentation for Covid-19 API	https://documenter.gettpostman.com/view/10808728/SzS8rjbc#intro	Yes, we learned how to use the Covid-19 API
4/16/2023	Utilizing documentation for Meteostat API	https://dev.meteostat.net/api/#sign-up	Yes, we learned how to sign up for the Meteostat API using rapid-keys and how to use the API
4/21/2023	Documentation for matplotlib.pylot for functions such as figure size, legend, xticks.	https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html	Yes, we found the correct code for those functions and successfully implemented them