Github repository: https://github.com/aryanagg/si206-final.git

Our original goal was to determine the link between air pollution, animal populations, and endangered species using different APIs to get the data for the 3 items mentioned above. However, in the end, we explored the relationship between COVID data, COVID deaths, and country population.

Some problems that we faced include our original APIs having some trouble. For example, the API we were going to use to explore data regarding endangered species migrated to a new URL and required us to request an API key. However, there was no email given for us to contact regarding the possibility of receiving an API key. As a result, we ended up needing to switch to a different API. Another issue we encountered was the fact that the COVID-19 deaths API had multiple records for each country. This made it harder to keep track of which deaths correlated to which country. Also, some countries didn't have regions associated with them which made storing and keeping track of the data challenging. Our solution to this was to record each observation in our table, group the data by country, and calculate the count of COVID deaths per country from there for the visualizations.

Calculation File:

Since the COVID API has multiple records per country, the calculation we did is to get the total count of COVID deaths per country.

Screenshots:
Code (from deaths scatterplot.py):

```
def main():
    db = 'final_data.db'
    query = """
        SELECT covid_deaths.country_region AS country,
               SUM(covid_deaths.deaths) AS total_deaths,
               pollution_data.pollution_2023
        FROM covid_deaths
        JOIN pollution_data ON covid_deaths.country_region = pollution_data.country
        GROUP BY covid_deaths.country_region, pollution_data.pollution_2023;
    """
```

CSV file with the calculated total number of COVID deaths per country:

1	Country	Total Number of COVID Deaths
2	Afghanistan	7896
3	Albania	3598
4	Angola	1933
5	Argentina	130472
6	Armenia	8727
7	Bahrain	1553
8	Bangladesh	29445
9	Benin	163
10	Bhutan	21
11	Bolivia	22365
12	Bosnia and Herzegovina	16280
13	Botswana	2801
14	Bulgaria	38228
15	Burundi	38
16	Cambodia	3056
17	Cameroon	1965

Instructions for running the code:

To create the table of air pollution level by country with 100 rows, run the pollution_data.py file 4 times. Each time the file is run, it adds 25 rows to the table. To create the Country and Population tables, the former of which has each country's name and the integer key it corresponds to, and the latter of which has the integer key and the

population for that key, run the tables.py file 4 times. Each time the file is run, it adds 25 rows to the tables. To join these tables, run combine_tables.py. In order to create the table of COVID death records, run the COVID Deaths.py file 5 times. It adds 25 rows at a time the first 4 times it's run and then inserts all remaining rows at once after that.

To create the scatterplot of number of COVID deaths vs air pollution level by country and generate the text file with calculations, run the deaths scatterplot.py file. To create the scatterplot of number of COVID deaths vs population by country and generate the text file with calculations, run the population_vs_deaths_scatter.py file. To create the barplot of the top 10 countries with the highest air pollution levels, run the barchart.py file.

Code documentation:

COVID Deaths.py

- get_data(): Gets the COVID-19 data from the specified API and outputs the "raw data" as a list of dictionaries if the API call is successful.
- get_current_row_count(cursor): Takes in a database cursor, executes a SQL query to count the number of rows in the covid_deaths table, and outputs the count (int).
- insert_rows(data, chunk_size): Takes in the data to be inserted (list of dictionaries) and the number of rows to insert at once (int). Creates the covid_deaths table if it doesn't exist and inserts data in chunks of 25 rows until the table has 100 rows, then inserts all remaining data in one batch.
- main(): Calls the get_data and insert_rows() functions.

deaths scatterplot.py

- get_data(db, query): Takes in the name of the SQLite database file (string) and the SQL query (string). Connects to the database, runs the query, and outputs the data as a Pandas dataframe.
- write_to_csv(df, filename): Takes in the Pandas dataframe and the name of the CSV file (string). Writes the country and total_deaths columns from the DataFrame into a CSV file with a header row.
- create_scatter_plot(df): Takes in the Pandas dataframe, creates/displays the scatterplot of number of covid deaths vs air pollution level by country, and saves it as a png
- main(): Contains the database name and SQL query. Calls get_data(), write_to_csv(), and create_scatter_plot()

pollution_data.py

- fillDatabase(url): Creates a beautiful soup object and uses the passed in url (string) to scrape the page for pollution data. It connects to and fills the database 25 entries at a time by executing INSERT batch_size number of times which is set to 25
- main(): Calls fillDatabase() with the url for the wikipedia page with pollution statistics

population_vs_deaths_scatter.py

- get_data(db, query): Takes in a database name (string) and SQL query (string) and outputs the data that will be used in the visualization (Pandas dataframe)
- write_to_csv(df, filename): Takes in the data returned from the get_data() function (Pandas dataframe) and a filename (string). It writes the given name and data from the get_data() function to a csv file.
- create_scatter_plot(df): Takes in the data returned from the get_data() function (Pandas dataframe), creates/displays a scatterplot of number of covid deaths vs population, and saves it as a png file.
- main(): Contains the SQL query and database name. Then it calls the above functions to create the visualization.

tables.py

- setup_database(): Creates the Country and Population tables
- fetch_and_store_country_data(limit, offset): Takes in the limit of rows to store per run (int) and the offset/starting index (int). Gets data from the API and stores it in the tables
- fetch_incremental_data(): Gets the data in chunks of 25, tracks the progress using a FetchTracker table, and updates the offset after each fetch.
- main(): Calls fetch_incremental_data()

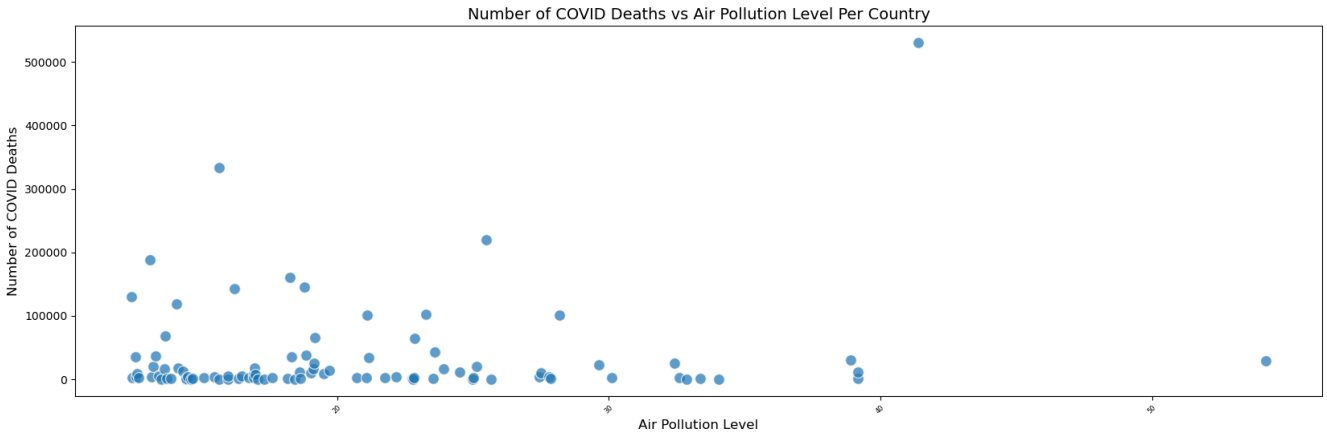
barchart.py

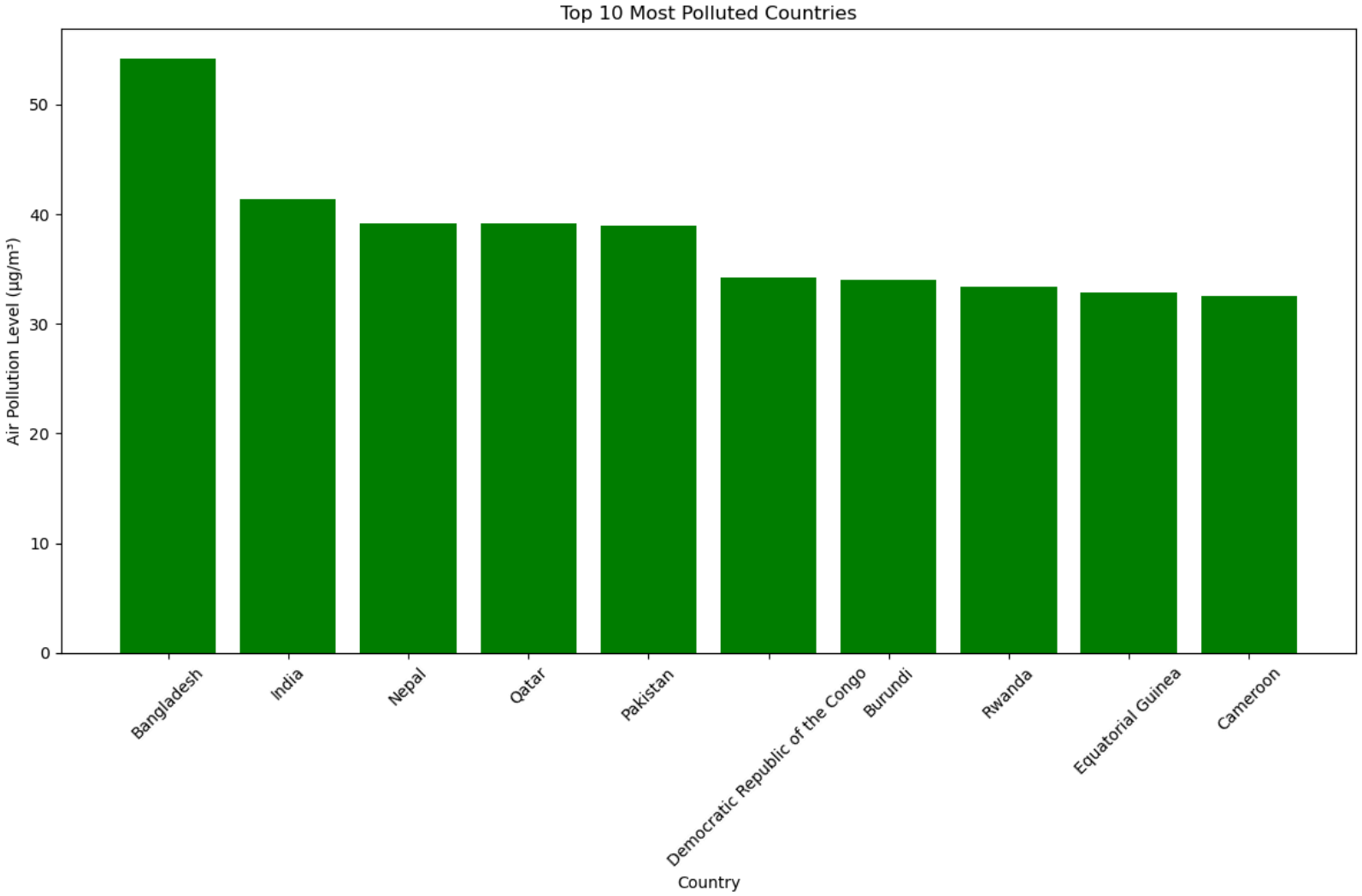
- get_data(db, query): Takes in the name of the SQLite database file (string) and the SQL query (string). Connects to the database, runs the query, and outputs the data as a Pandas dataframe
- create_pollution_bar_chart(df): Takes in the Pandas dataframe, creates/displays the bar chart of top 10 most polluted countries, and saves it as a png.
- main(): Contains the database name and SQL query. Calls get_data() and create_pollution_bar_chart()

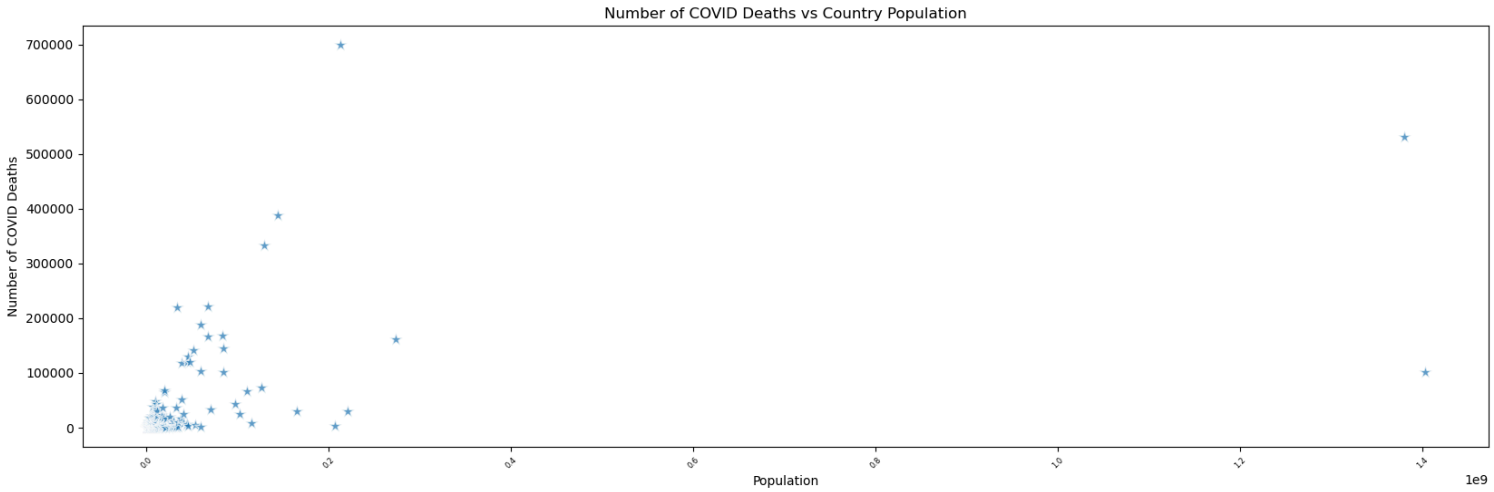
combine_tables.py

- There are no functions but the file creates a connection to the database and creates a new table that joins the table for Population data and Country name based on the common country id in the respective database tables

Visualizations:




Top 10 Most Polluted Countries
Air Pollution Level (µg/m³)
0
10
20
30
40
50
Bangladesh
India
Nepal
Qatar
Pakistan
Democratic Republic of the Congo
Burundi
Rwanda
Equatorial Guinea
Cameroon
Country


Number of COVID Deaths vs Country Population
Number of COVID Deaths
0
100000
200000
300000
400000
500000
600000
700000
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Population
1e9

Documentation:

Date	Issue Description	Location of resource	Result (did it solve the issue)
12/9/2024	Need to find a way to count the number of rows in a table	https://www.w3schools.com/sql/sql_count.asp	Yes - we were able to use this SQL query to count the rows
12/11/2024	Need to find a function to convert results of a SQL query to a Pandas dataframe	https://pandas.pydata.org/docs/reference/api/pandas.read_sql.html	Yes - we were able to use read_sql()
12/11/2024	Need to get values from a Pandas dataframe to write to a csv file	https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.values.html	Yes - I was able to use .values() to get the values, and then use .tolist() to convert them to a list
12/12/2024	Wanted to customize elements of my plot	https://matplotlib.org/stable/index.html	Yes, I was able to customize my plot

REPORT	**100**	
Goals	_____ 20	-10 if missing original goals -10 if missing achieved goals
Problems that you faced	_____ 10	-10 if missing
Include the calculation file	_____ 10	-10 if not included physically in the report
Include the visualizations created	_____ 10	-10 if no visualizations physically included -5 if some but not all visualizations included
Instructions for running the code	_____ 10	-5 if unclear -5 if instructions didn't work
Code documentation (explain what each function does including describing its input and output)	_____ 20	-5 for each missing function explanation (up to -20)
Documentation of resources used	_____ 20	-10 for unclear resource documentation (not following the format) -20 for no documentation