SI 206 REPORT

A. The goals for your project including what APIs/websites you planned to work with and what data you planned to gather (10 points)

Our goal for this project was to analyze COVID-19 case trends and vaccination efforts across the United States. We planned to work with two APIs:

1. The COVID Tracking Project API: https://api.covidtracking.com/v1/states/daily.json- to gather daily cases and hospitalizations by state
2. The CDC Vaccination API: https://data.cdc.gov/resource/unsk-b7fc.json - to obtain vaccination data by state over time

We aimed to:

- Store the data in a normalized SQLite database with joined tables
- Calculate average daily positive cases and average vaccinations by state
- Generate visualizations and a heatmap

B. **The goals that were achieved including what APIs/websites you actually worked with and what data you did gather (10 points)**

We successfully:

- Used both the COVID Tracking Project API and the CDC Vaccination API
- Created a single database (covid_data.db) with three joined tables: covid_data, vaccination_data, and state_metadata.
- Collected over 100 rows from each API (only inserting 25 rows at a time)
- Performed calculations using SQL joining to compute average daily positive case increases and average vaccine doses administered per state
- Exported these metrics into a average_by_state.txt file
- Created evolving heat map

C. The problems that you faced (10 points)

- Date Format Conflicts: The COVID API used YYYYMMDD while the CDC API used YYYY-MM-DDTHH:MM:SS. We had to parse and standardize these formats to allow proper joins.
- Matching States: Ensuring accurate state_id foreign key relationships required a new state_metadata table
- Visualization Troubles: Getting folium to work was not easy. It took lots of time and effort to figure out what was going wrong with the time series visualization. Making the map visualization data come across aesthetically pleasing and accurate.

D. The calculations from the data in the database (i.e. a screenshot) (10 points)

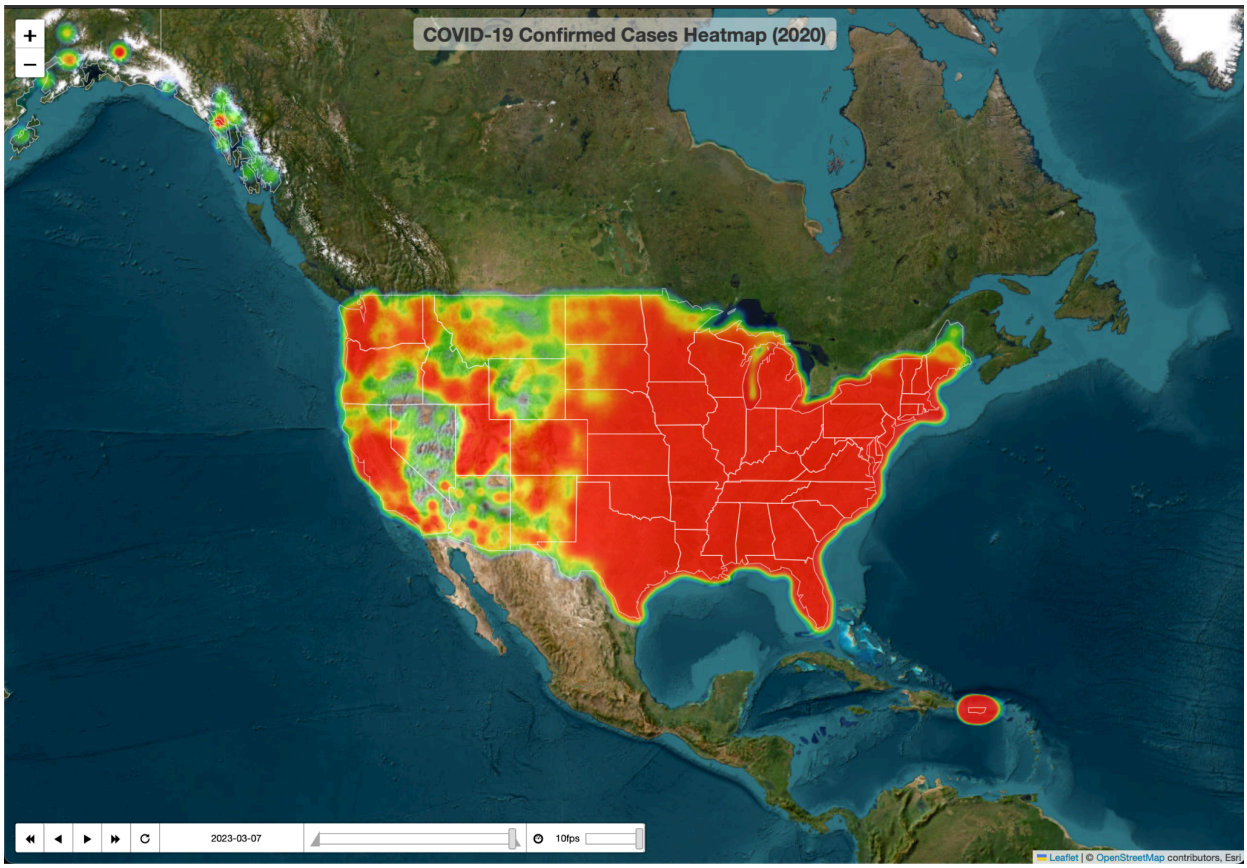
This file was generated by joining the COVID and vaccination tables on both state_id and date, and computing the averages using SQL aggregation functions.

average_by_state.txt

```
State | Avg Positive Increase | Avg Vaccinations Administered
------------------------------------------------------------
AK | 47.00 | 888762.33
AL | 581.00 | 6299658.33
AR | 354.00 | 4872445.50
AZ | 1782.00 | 14639152.50
CA | 4309.00 | 59034472.67
CO | 1193.67 | 13025897.50
CT | 276.67 | 9036867.50
DC | 135.00 | 2136508.33
DE | 237.00 | 1627912.00
FL | 4829.00 | 38799956.50
GA | 2019.67 | 17118032.50
HI | 77.67 | 2356050.67
IA | 331.00 | 6126596.33
ID | 214.33 | 2892598.00
IL | 1691.67 | 26824394.33
IN | 938.67 | 7475711.33
KS | 250.67 | 5383450.00
KY | 756.33 | 7505544.33
LA | 419.33 | 6261614.00
MA | 1682.00 | 12123992.67
MD | 851.33 | 14704828.00
ME | 189.00 | 2365195.00
MI | 1161.00 | 14072078.00
MN | 879.00 | 12821131.00
MO | 369.00 | 10500721.00
MS | 475.67 | 4305879.50
MT | 127.67 | 1869148.33
NC | 1373.33 | 19421292.33
ND | 69.00 | 988475.00
NE | 351.00 | 3448286.75
NH | 227.67 | 2981369.50
NJ | 3313.33 | 17934144.00
NM | 255.33 | 3564759.25
NV | 373.67 | 5588042.00
NY | 7797.33 | 45141876.33
OH | 1330.33 | 14080644.33
OK | 785.33 | 6768664.67
OR | 264.00 | 9390626.00
PA | 2401.33 | 18410482.00
RI | 351.33 | 2670640.50
SC | 1434.00 | 8663454.00
```

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

Created animated heatmap: gen_map.py generates a heatmap using Folium, which animates COVID case spread over time.




```python
clipped_data = [
    [[point[0], point[1], max(0, point[2])] for point in time_step]
    for time_step in data
]

# Step 2: logarithmic transformation (ideally will bring out large changes in data)
transformed_data = [
    [[point[0], point[1], math.log1p(point[2])] for point in time_step]
    for time_step in clipped_data
]

# Step 3: Global Max
global_max_transformed = max(
    max(point[2] for point in time_step)
    for time_step in transformed_data if any(point[2] > 0 for point in time_step)
) or 1  # Avoid division by zero

# Step 4: Normalization
normalized_data = [
    [[point[0], point[1], point[2] / global_max_transformed] for point in time_step]
    for time_step in transformed_data
]

curated_data = [
    [[point[0], point[1], point[2] if point[2] > threshold else 0] for point in time_step]
    for time_step in normalized_data
]
```

Data transformation process for gen_map.py

F. Instructions for running your code (10 points)

The instructions are as follows:

- load_covid_data.py: sets up the sqlite3 database for interaction. It populated covid_data.db by pulling from both APIs and inserting up to 25 new rows per run.

- avg_by_state.py: uses the sqlite3 database and the joined tables to create an average_by_state.txt file that contains the calculated average positive increase and the average vaccines administered

- Time_series_to_sql.py: Loads time series case data (from CSV) into the time_series_data table, applying a skip interval.

- gen_map.py: running this will ask you for a skip value ("You want every x-day of data." Makes the script run faster. I don't recommend going above 20). After it finishes, a new file will be created "heatmap_with_time.html". It should open in safari automatically. If not, simply open the file with the browser of your choice.

- csv_manipulation.py should not be run. It was used in the development stage and is now deprecated.

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

- **Function**: **insert_covid_data (row,cur)** in the load_covid_data.py file
- Purpose: inserts one COVID data row into the covid_data table, if it doesn't already exist
    - Inputs:
        - Row: dictionary from the COVID tracking project (ex. a single states data for a day
        - Cur: SQLite cursor
    - Outputs:
        - Inserts data into covid_data (no return values)
- **Function**: **insert_vaccination_data (row,cur)** in the load_covid_data.py file
- Purpose: inserts one vaccination data row into the vaccination_data, linking to state_meta
    - Inputs:

        - Row: dictionary from CDC Vaccination API
        - Cur: SQLite cursor
    - Outputs:
        - Inserts data into vaccination_data (no return values)
- **Function**: **generate_avg_report()** in the avg_by_state.py
- Purpose: Calculates and writes average positive cases and average administered vaccinations per state to a .txt file.
    - Inputs:
        - None since it operates
    - Outputs:
        - Creates a text file: average_by_state.txt
- **Function: get_time_series_dict(data, skip_value)** in time_series_to_sql.py
- Purpose: Converts CSV data into a structured format ready for animation (e.g., list of [lon, lat, intensity] by time step).
    - Inputs:
        - data: dataset from read_csv()
        - skip_value: how often to skip time steps to control animation length
    - Outputs:
    - Returns a list of time steps where each time step is a list of [lon, lat, intensity]

H. 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?)
4/12/25	Couldn't get the visualization to use time series data	Documentation	Half way
4/13/25	A static visualization showed up but no evolving visualization	ChatGPT	No
4/14/25	^	Documentation and Grok	Yes