Hazardous Waste in Texas

Angelica Guerrero, Deanna Sanchez, Ernesto Garcia, Juan Segovia, Kalan Suneson

Where did we get our data?

Our data was captured from the United States Environmental Protection Agency Website.
(The EPA is a federal agency that protects the environment and human health)

EPA United States Environmental Protection Agency

Toxic Release Inventory

- The TRI is a public database that tracks the release, transfer, and management of toxic chemicals by certain facilities in the united states.
- Facilities that manufacture, process, or use toxic chemicals above certain levels must report to TRI.
- The TRI includes information on over 767 chemicals and 33 chemical categories. These chemicals are considered toxic because they may cause cancer, other chronic health effects, or significant adverse environmental effects

Prepping the waste data?

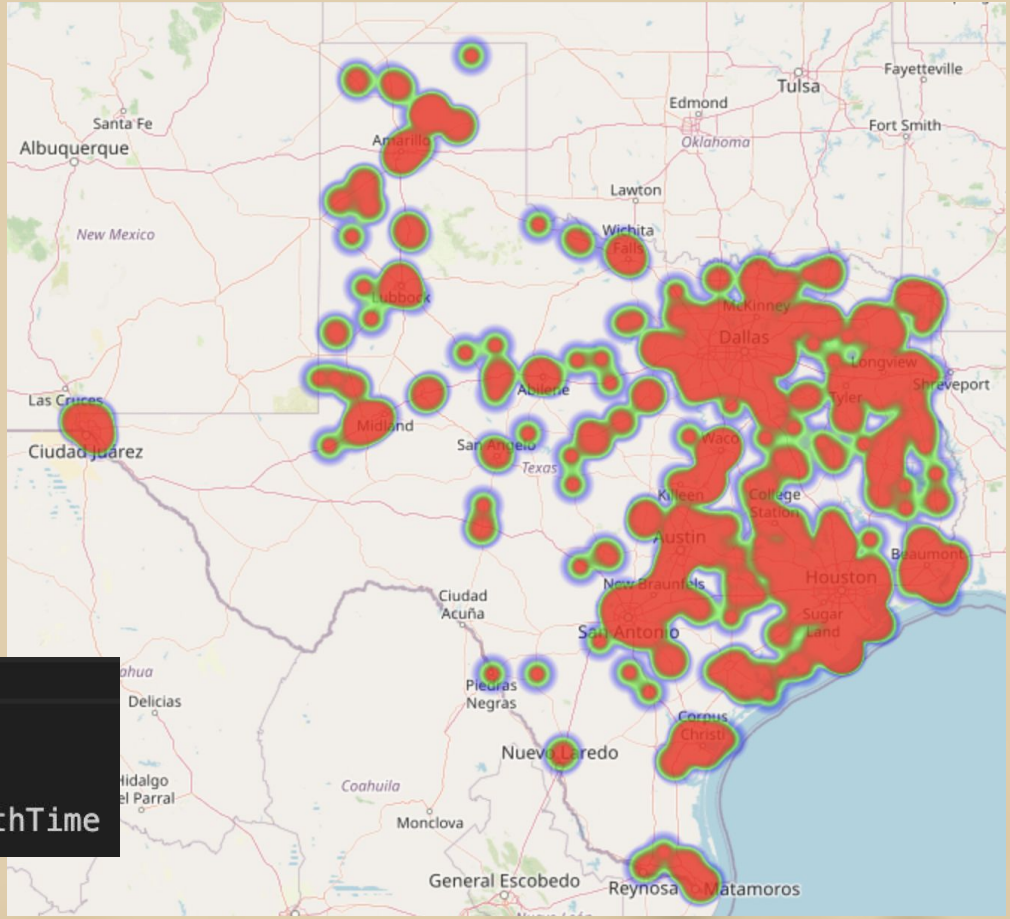
First we took the csv's for each year, then cleaned those up to focus only on Texas. We then put these into mongoDB, merged them, exported as jsons, then imported them into our codes. Which we then manipulated into the following maps.

```
merging_9_to_11 = ["dumping_list_9", "dumping_list_10", "dumping_list_11"]
merging_12_to_14 = ["dumping_list_12", "dumping_list_13", "dumping_list_14"]
merging_15_to_17 = [ "dumping_list_15", "dumping_list_16", "dumping_list_17"]
merging_18_to_20 = ["dumping_list_18", "dumping_list_19", "dumping_list_20"]
merging_21_to_23 = ["dumping_list_21", "dumping_list_22", "dumping_list_23"]

def merge_and_export(collection_names, output_filename):

    all_collections = db["all_data_combined"]
    all_collections.drop()


    for collection in collection_names:
        dump_collections = db[collection]
        docs = list(dump_collections.find())
        all_collections.insert_many(docs)
        print(f"Merged {dump_collections.count_documents({})} documents from {collection}")


    all_docs = list(all_collections.find())
    with open(output_filename, 'w') as json_file:
        json.dump(all_docs, json_file, default=str)

    print(f"Data has been successfully exported to '{output_filename}'.")


merge_and_export(merging_9_to_11, 'data_9_to_11.json')
merge_and_export(merging_12_to_14, 'data_12_to_14.json')
merge_and_export(merging_15_to_17, 'data_15_to_17.json')
merge_and_export(merging_18_to_20, 'data_18_to_20.json')
merge_and_export(merging_21_to_23, 'data_21_to_23.json')

print("All collections have been combined and exported into separate JSON files.")
```

Maps

This map that you will see in the dashboard is a Texas time lapse heatmap years 2009 - 2023, that grows and declines with how much waste the county's produced between these years.

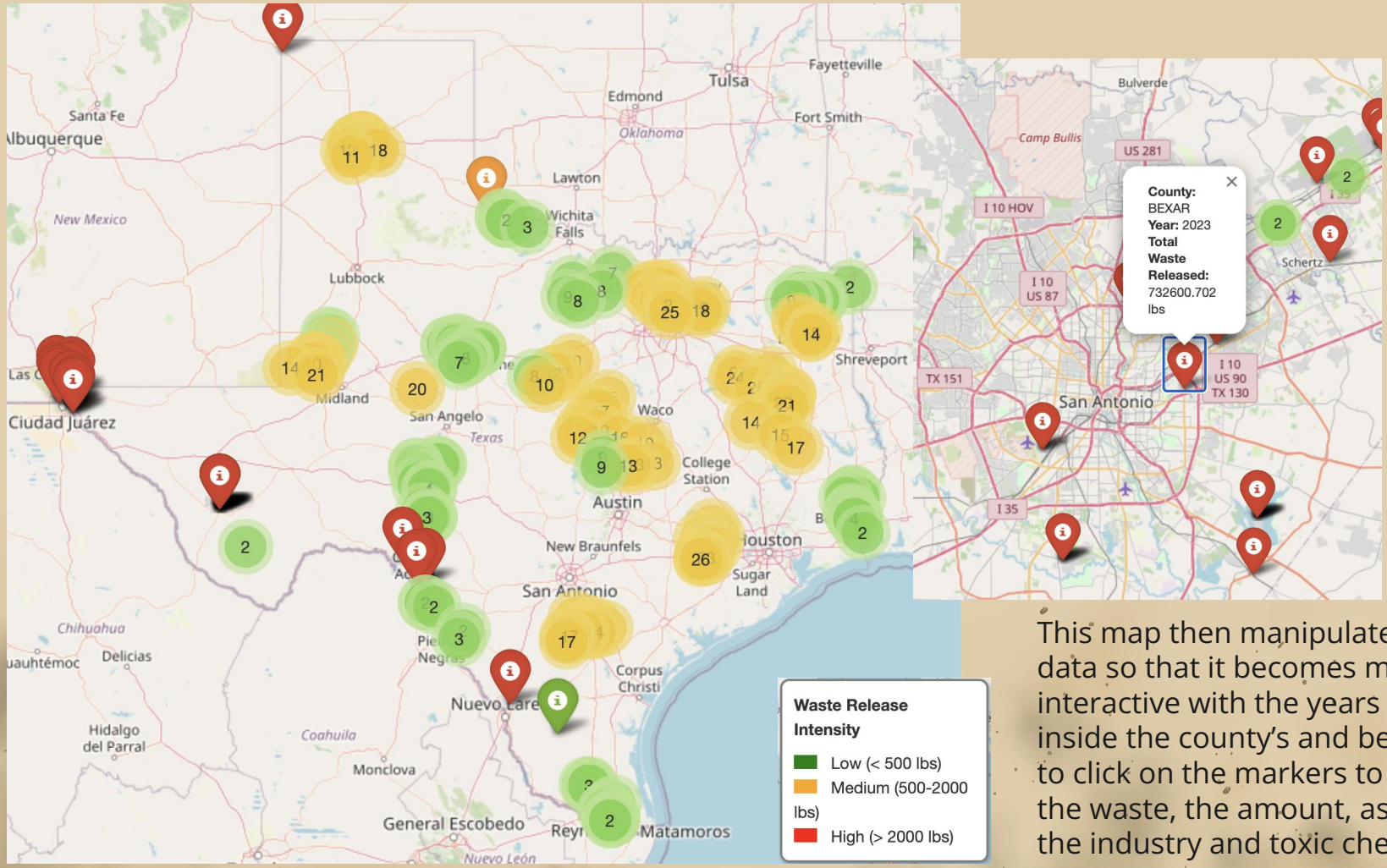
```
import json
import folium
import pandas as pd
from folium.plugins import HeatMapWithTime
```



```
# Load data from each JSON file and combine them
for file in files:
    with open(file, 'r') as f:
        data = json.load(f)
        df = pd.DataFrame(data)
        all_data = pd.concat([all_data, df], ignore_index=True)

# Ensure required columns
required_columns = {'FACILITY_NAME', 'YEAR', 'LATITUDE', 'LONGITUDE', 'TOTAL_RELEASES'}
if not required_columns.issubset(all_data.columns):
    raise ValueError("Data must contain the columns: 'FACILITY_NAME', 'YEAR', 'LATITUDE', 'LONGITUDE', and 'TOTAL_RELEASES'.")

# Prepare data for each year for the time-lapse
heat_data_time = []
years = sorted(all_data['YEAR'].unique())  # Sorted list of unique years

for year in years:
    # Filter data for the specific year and drop rows with NaN values in required columns
    year_data = all_data[(all_data['YEAR'] == year) & all_data[['LATITUDE', 'LONGITUDE', 'TOTAL_RELEASES']].notnull().all(axis=1)]

    # Create the data format for the heatmap with time: [[latitude, longitude, intensity], ...]
    heat_data_year = [[row['LATITUDE'], row['LONGITUDE'], row['TOTAL_RELEASES']] for index, row in year_data.iterrows()]

    # Add this year's data to the heat_data_time list
    heat_data_time.append(heat_data_year)

# Create a base map centered around Texas with a zoom level suitable for an overview
map_center = [31.0, -99.0]  # Texas approximate center
m = folium.Map(location=map_center, zoom_start=6)

# Add time-lapse heatmap to the map
HeatMapWithTime(
    heat_data_time,  # Data prepared for each year
    radius=15,       # Radius of each heat point
    max_opacity=0.8, # Max opacity of heat points
    gradient={0.4: 'blue', 0.65: 'lime', 1: 'red'}  # Gradient from low to high intensity
).add_to(m)

# Save the map to an HTML file and display it
m.save("time_lapse_heatmap.html")
print("Time-lapse map has been created and saved as 'time_lapse_heatmap.html'. Open this file to view the animated map.")
```

The Interactive Heat Map was created by passing in the clean json files then using the python library folium to manipulate the data and produce this interactive timelapse map for the years 2009 - 2023.

```
import json
import folium
import pandas as pd
from folium.plugins import HeatMapWithTime
```

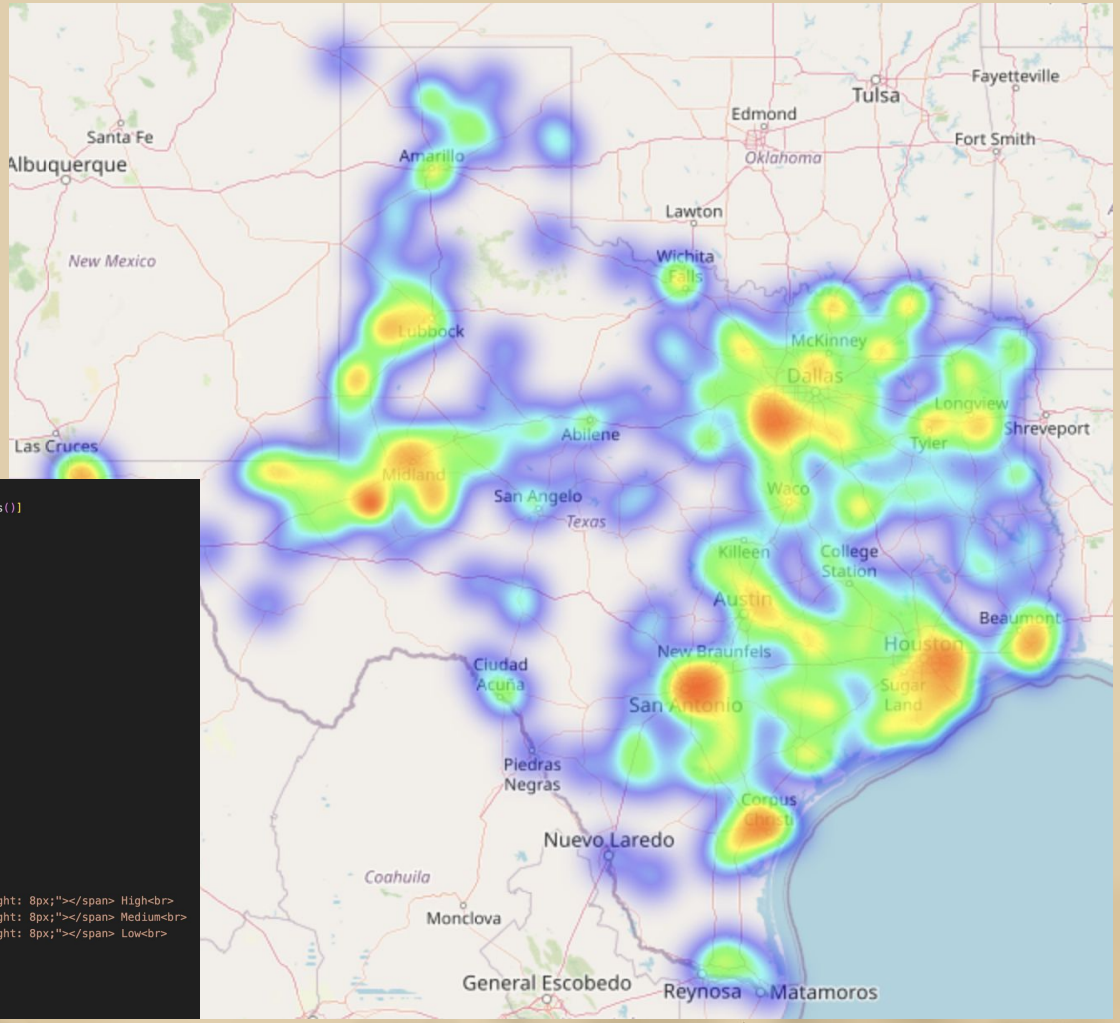


This map then manipulates the data so that it becomes more interactive with the years being inside the county's and being able to click on the markers to explore the waste, the amount, as well as the industry and toxic chemicals.

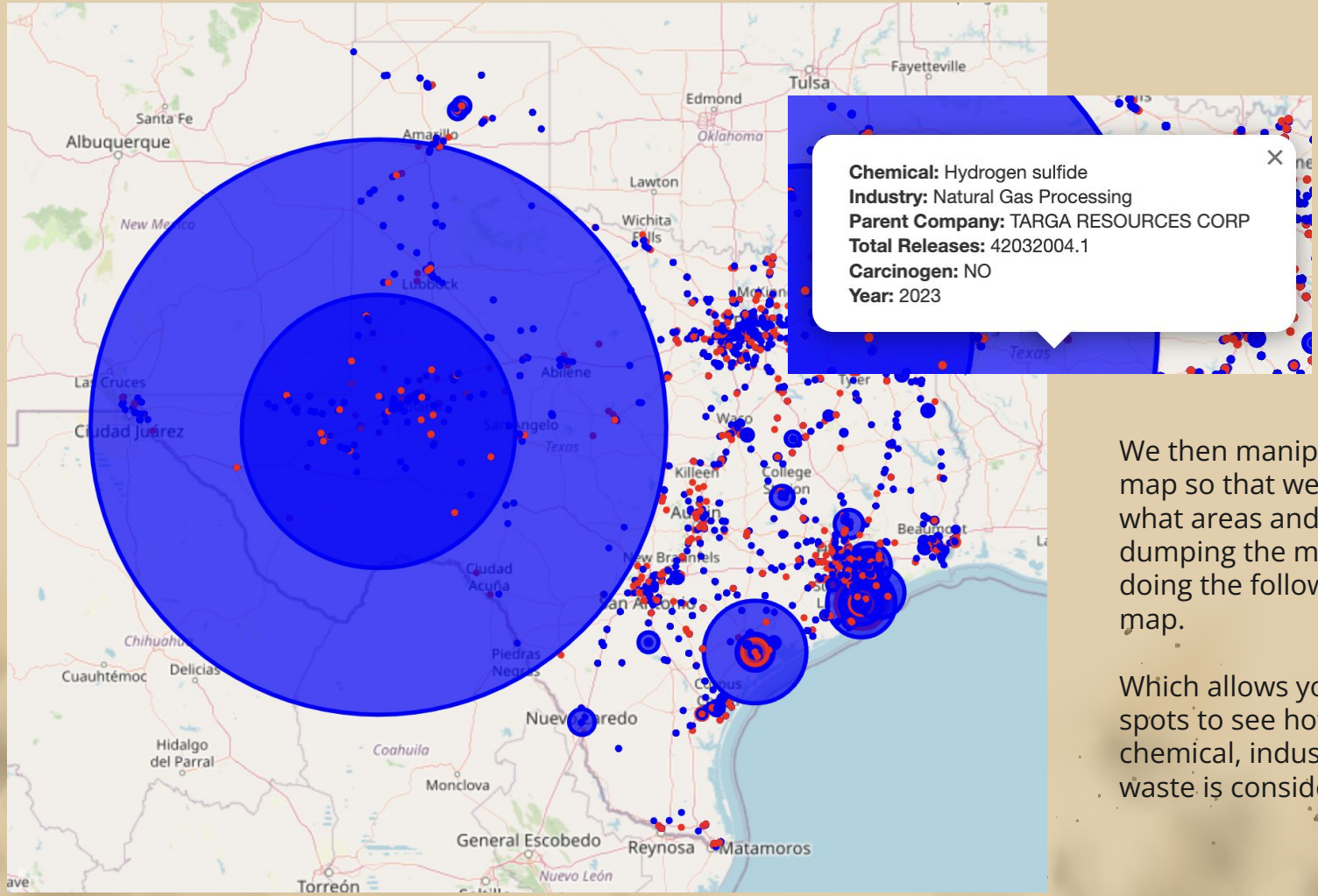
The json files were loaded in similarly with a different map layer and features to bindpop each of the county's stats with the year.

```
# Create a folium map centered around Texas
map_center = [31.0, -99.0]  # Texas approximate center
m = folium.Map(location=map_center, zoom_start=6)

# Define color scale for different ranges of waste intensity
def get_marker_color(total_waste):
    if total_waste < 500:
        return 'green'
    elif total_waste < 2000:
        return 'orange'
    else:
        return 'red'

# Create a feature group for each year and add markers
years = county_year_data['YEAR'].unique()
for year in years:
    # Create a feature group for the specific year
    year_group = FeatureGroup(name=f"{year} Waste Releases")
    # Filter data for the specific year
    year_data = county_year_data[county_year_data['YEAR'] == year]

    # Use MarkerCluster to manage overlapping markers
    marker_cluster = MarkerCluster().add_to(year_group)

    for _, row in year_data.iterrows():
        # Determine color based on total waste
        color = get_marker_color(row['total_waste'])

        # Popup content with county, year, and waste details
        popup_text = f"<strong>County:</strong> {row['COUNTY']}<br>" \
                     f"<strong>Year:</strong> {int(row['YEAR'])}<br>" \
                     f"<strong>Total Waste Released:</strong> {row['total_waste']} lbs"

        # Add marker to the cluster with color and popup info
        folium.Marker(
            location=[row['latitude'], row['longitude']],
            popup=popup_text,
            icon=folium.Icon(color=color, icon="info-sign")
        ).add_to(marker_cluster)

    # Add year group to map
    year_group.add_to(m)

# Add a layer control panel to toggle year layers on/off
LayerControl().add_to(m)
```

Due to the data we pull from the first map we then focused on 2023 with the next two maps, this allowed us to isolate some data about Texas.




```
# Prepare data for the heatmap, weighted by `TOTAL_RELEASES`
heat_data = [[row['LATITUDE'], row['LONGITUDE'], row['TOTAL_RELEASES']] for index, row in heat_data.iterrows()]

# Add the heatmap layer
HeatMap(heat_data, radius=15, max_opacity=0.7).add_to(m)

# Add a customized legend to the map
legend_html = '''
  <div style="
  position: fixed;
  bottom: 50px;
  left: 50px;
  width: 220px;
  height: 120px;
  background-color: white;
  border: 2px solid grey;
  z-index: 9999;
  font-size: 14px;
  padding: 12px;
  border-radius: 8px;
  box-shadow: 2px 2px 5px rgba(0,0,0,0.3);
  line-height: 1.8;
  ">
    <strong>Waste Release Intensity</strong><br>
    <div style="margin-top: 10px;">
        <span style="display: inline-block; width: 20px; height: 12px; background-color: #ff0000; margin-right: 8px;"></span> High<br>
        <span style="display: inline-block; width: 20px; height: 12px; background-color: #ffa500; margin-right: 8px;"></span> Medium<br>
        <span style="display: inline-block; width: 20px; height: 12px; background-color: #ffff00; margin-right: 8px;"></span> Low<br>
    </div>
  </div>
'''
m.get_root().html.add_child(folium.Element(legend_html))
```



We then manipulated the type of map so that we could make it clear what areas and companies were dumping the most. We did this by doing the following bubble style map.

Which allows you to hover over the spots to see how much waste, chemical, industry and even if the waste is considered a carcinogen.

```python
# Create a map centered on Texas
m = folium.Map(location=[31.0, -99.0], zoom_start=7)

# Function to determine bubble color
def get_color(carcinogen):
    return "red" if carcinogen == "YES" else "blue"

# Very small scaling factor for TOTAL_RELEASES
SCALING_FACTOR = 0.01  # Try a smaller factor to prevent oversized circles

# Add data points as bubbles
for _, row in df_filtered_year.iterrows():
    # Prevent extremely small or large circles with logarithmic scaling
    radius = max(5, row["TOTAL_RELEASES"] * SCALING_FACTOR)  # Minimum radius of 5 meters for visibility

    folium.Circle(
        location=[row["LATITUDE"], row["LONGITUDE"]],
        radius=radius,  # Adjusted and scaled radius
        color=get_color(row["CARCINOGEN"]),
        fill=True,
        fill_color=get_color(row["CARCINOGEN"]),
        fill_opacity=0.7,
        popup=folium.Popup(
            f"<b>Chemical:</b> {row['CHEMICAL']}<br>"
            f"<b>Industry:</b> {row['INDUSTRY_SECTOR']}<br>"
            f"<b>Parent Company:</b> {row['PARENT_CO_NAME']}<br>"
            f"<b>Total Releases:</b> {row['TOTAL_RELEASES']}<br>"
            f"<b>Carcinogen:</b> {row['CARCINOGEN']}<br>"
            f"<b>Year:</b> {row['YEAR']}",
            max_width=300
        )
    ).add_to(m)

# Save the map as an HTML file
m.save("texas_bubble_map.html")
```

Live Demo

We used a flask server to make our dashboard sharable and interactive.

Conclusion

With the story we are telling we were able to first look at Texas and visibly see what year has the most waste, as well as take a deep dive into what exactly was being dumped.

The last two maps allowed us to visibly see what area of Texas has the most amount of dumping as well as take a deeper dive into those specific companies while also putting a visualization of just how much more that specific company was dumping.

We discovered the company that dumps the most amount of waste at 42,032,004 lbs of waste in the year of 2023 is a company in Midland with its parent company being Targa Resources Group. However that's a lot, at least we can rest at night knowing that they aren't carcinogenic chemicals.

Targa Resources Corp. is a Fortune 500 company based in Houston, Texas. Targa, **a midstream energy infrastructure corporation**, is one of the largest infrastructure companies delivering natural gas and natural gas liquids in the United States.

Thanks!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**