Team: Dennis Liu, Alexander Liu, Landon Pugh

Link to Github Repo:

- https://github.com/AlexL320/SI206-Final-Project

Original Goals:

- Have 3 weather APIs to collect data from and find the difference between them.

Achieve Goals:

- Have 2 APIs, one from the weather website we found and another from a NFL API to gather data about the attendance of every NFL game within a certain time frame. The third website was a Wikipedia article that has a list of cities in the United States.
- We used Matplotlib to make the graphs we used for our visualizations. For bar graphs, we used it to visualize the average percentile attendance of each NFL stadium where a game was played within our specified timeframe.
- For the work division. Dennis Liu would be tasked with finding the average attendance of each NFL game in our time frame, find the location, and calculate the average percentile attendance for each stadium. Landon Pugh found the weather conditions from every home team's city on the day of a game from 100 games in 2023.

Problems we faced:

- We had to change our goals after we found out that the project required us to find different websites and topics to link together. A few days were spent on finding websites that have data we could link together.
- Finding a third website to relate to football and weather took several days, as most of the websites on the github repository didn't have past data or were unique enough.
- Pulling data from the database and making the graph was a major challenge, as I had to use several for loops and if statements to get the data I wanted.

Calculations:

- Calculates the percentage of attendance to the capacity of the stadium.

```
import sqlite3
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from data_gather import *

def create_graph():
    # Lists for the x and y axis
```

```
    graph_x = []
    graph_y = []

    connection= sqlite3.connect('final_test.db')

    # Pulls data from the Location table in the database
    cursor_loct = connection.cursor()
    cursor_loct.execute("SELECT * FROM Location")
    rows_loct = cursor_loct.fetchall()

    # Pulls data from the Games table in the database
    cursor_cap = connection.cursor()
    cursor_cap.execute("SELECT * FROM Games")
    rows_cap = cursor_cap.fetchall()

    for loct in rows_loct:
        visited = []
        loct_id = loct[0]
        loct = loct[1]
        loct_lst = loct.split(',')
        if len(loct_lst) > 1:
            loct_city = loct_lst[0].strip('(').strip("'")
            loct_state = loct_lst[1].strip(')').strip("'")
            loct_state = loct_state.strip(" '")
            location = loct_city + ", " + loct_state
            percent_list = []
            # Loops through row_cap and creates a list of attendance
percentage for each location
            for cap in rows_cap:
                percent = 0.0
                cap_id = cap[3]
                if cap_id == loct_id:
                    attendance = cap[4]
                    capacity = cap[5]
                    percent = (attendance / capacity)
                    percent_list.append(percent)
            # Calculates the average attendance percentage for each
location
            average = sum(percent_list) / len(percent_list)
            average = int(average * 10000) / 100
```

```
            input_data = average
            temp_tup = (location, average)
            # Adds a unique temp_tup to the visited list
            if temp_tup not in visited:
                graph_x.append(average)
                graph_y.append(location)
                visited.append(temp_tup)

    # Plots the attendance as a bar graph
    plt.barh(graph_y, graph_x)
    # Adds the title and labels to the bar graph
    plt.title("Average attendance percentage for each NFL stadium")
    plt.xlabel("Percentage")
    plt.ylabel("Stadium City Location")

    # Adds the percentage to the end of each bar in the bar graph
    for i, (location, percent) in enumerate(zip(graph_y, graph_x)):
        # Add a little space to the right of the bar to position the label
        plt.text(percent + 1, i, f"{percent:.2f}%", va='center')
    plt.show()


def create_scatter_graph():
    #visualization
    graph_x = []
    graph_y = []
    labels = []

    connection= sqlite3.connect('final_test.db')
    cursor_coor = connection.cursor()
    cursor_coor.execute("SELECT * FROM Coordinates")
    rows_coor = cursor_coor.fetchall()

    cursor_loct = connection.cursor()
    cursor_loct.execute("SELECT * FROM Location")
    rows_loct = cursor_loct.fetchall()

    cursor_cap = connection.cursor()
    cursor_cap.execute("SELECT * FROM Games")
    rows_cap = cursor_cap.fetchall()
```

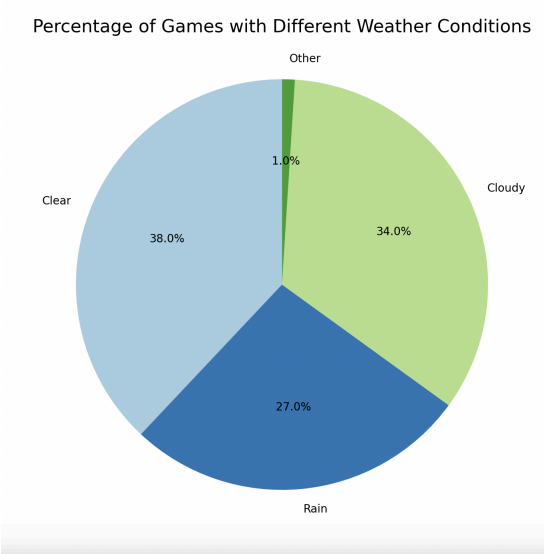
```
    cursor_guide = connection.cursor()
    cursor_guide.execute("SELECT * FROM Coord_Guide")
    rows_guide = cursor_guide.fetchall()

    location_lst = []
    for coor in rows_coor:
        state = None
        population = None
        state_num = coor[1]
        for guide in rows_guide:
            if guide[0] == state_num:
                state = guide[1]
                population = coor[4]
                city = coor[0]
        for cap in rows_cap:
            percentage = int((cap[4] / cap[5]) * 10000) / 100
            cap_num = cap[3]
            for loct in rows_loct:
                loct_num = loct[0]
                if cap_num == loct_num:
                    location_str = loct[1]
                    loct_lst = location_str.split(',')
                    loct_city = loct_lst[0].strip('(').strip("'")
                    loct_state = loct_lst[1].strip(')').strip("'")
                    loct_state = loct_state.strip(" '")
                    location = (loct_city, loct_state)
                    temp_tup = (city, state)
                    if temp_tup == location:
                        if location not in location_lst:
                            graph_x = [percentage] + graph_x
                            graph_y = [population] + graph_y
                            labels = [temp_tup] + labels
                            location_lst.append(location)
    for i, label in enumerate(labels):
        plt.annotate(label, (graph_x[i], graph_y[i]), textcoords="offset
points", xytext=(5,5), ha='center')
    plt.xlabel("percentage of stadium filled")
    plt.ylabel("population of city")
    plt.scatter(graph_x, graph_y)
```

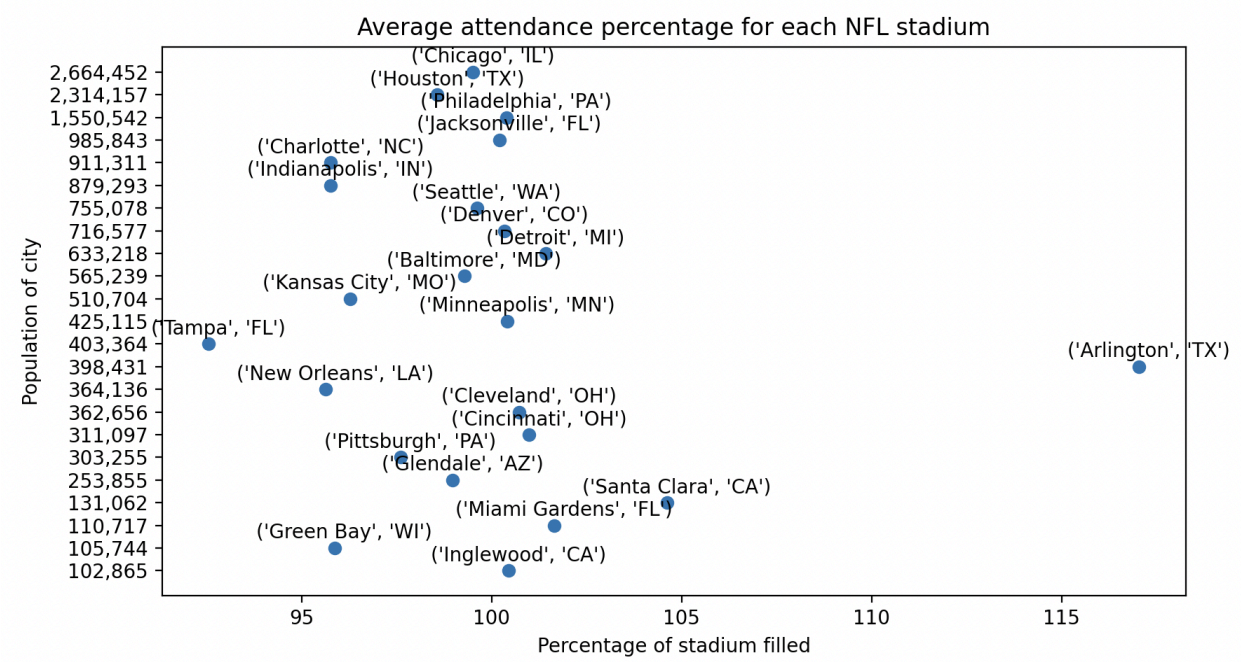
```
    plt.show()
    return (graph_x, graph_y)

# Function to create a pie chart for weather conditions
def make_pie_chart(conn):
    cur = conn.cursor()
    cur.execute("SELECT conditions FROM Weather")
    conditions_data = cur.fetchall()
    conditions_list = [condition[0] for condition in conditions_data]

    # Define categories for weather conditions
    def categorize_condition(condition):
        if "rain" in condition.lower():
            return "Rain"
        elif "clear" in condition.lower():
            return "Clear"
        elif "cloudy" in condition.lower():
            return "Cloudy"
        elif "snow" in condition.lower():
            return "Snow"
        else:
            return "Other"

    grouped_conditions = [categorize_condition(condition) for condition in
conditions_list]
    condition_counts = {}
    for condition in grouped_conditions:
        if condition in condition_counts:
            condition_counts[condition] += 1
        else:
            condition_counts[condition] = 1

    labels = list(condition_counts.keys())
    sizes = list(condition_counts.values())

    plt.figure(figsize=(8, 8))
    plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90,
colors=plt.cm.Paired.colors)
    plt.title('Percentage of Games with Different Weather Conditions',
fontsize=16)
```

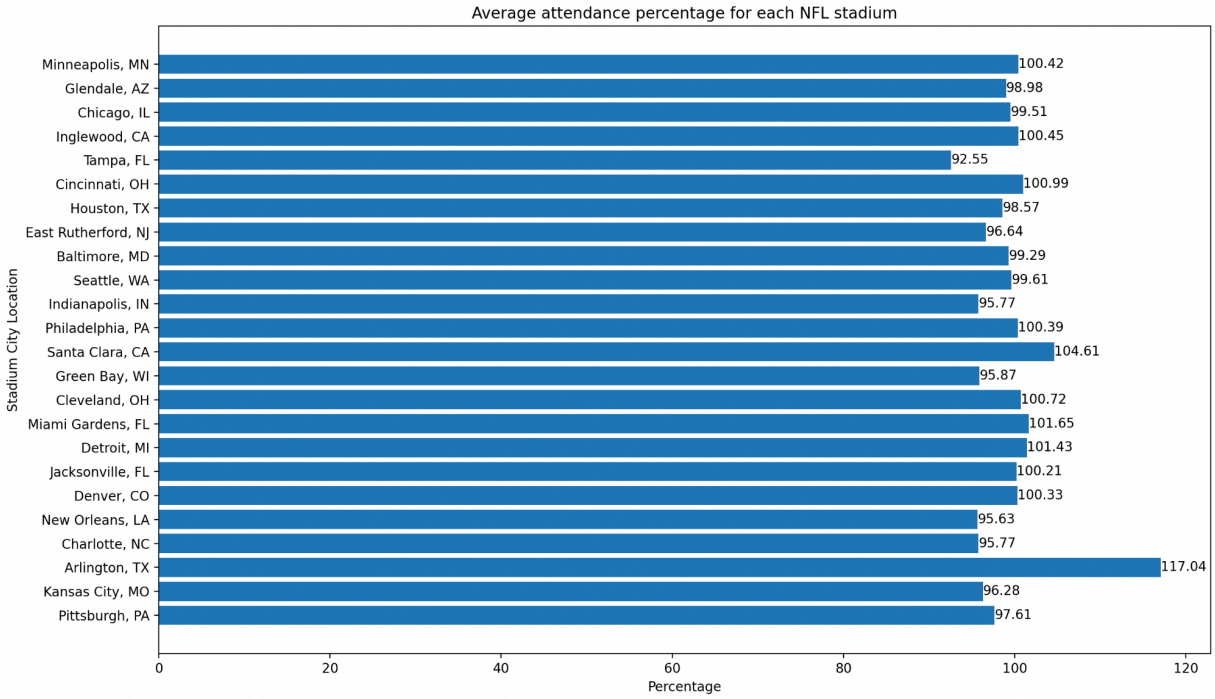
```
    plt.axis('equal')
    plt.show()

def create_weather_attendance_graph(conn):
    # Query the Weather and Games tables to get weather conditions and
attendance data
    cur = conn.cursor()
    cur.execute("""
    SELECT conditions, AVG(attendance)
    FROM Weather
    JOIN Games ON Weather.location_id = Games.location
    GROUP BY conditions
    """)
    weather_attendance_data = cur.fetchall()

    # Process the data into a dictionary: key = weather condition, value =
list of attendance
    weather_attendance = {}

    for condition, attendance in weather_attendance_data:
        # Categorize weather conditions into broader categories
        if "rain" in condition.lower():
            condition_category = "Rain"
        elif "clear" in condition.lower():
            condition_category = "Clear"
        elif "cloudy" in condition.lower():
            condition_category = "Cloudy"
        elif "snow" in condition.lower():
            condition_category = "Snow"
        else:
            condition_category = "Other"

        # Add the attendance to the list of the corresponding weather
condition category
        if condition_category not in weather_attendance:
            weather_attendance[condition_category] = []

        weather_attendance[condition_category].append(attendance)

    # Calculate the average attendance for each weather condition
```

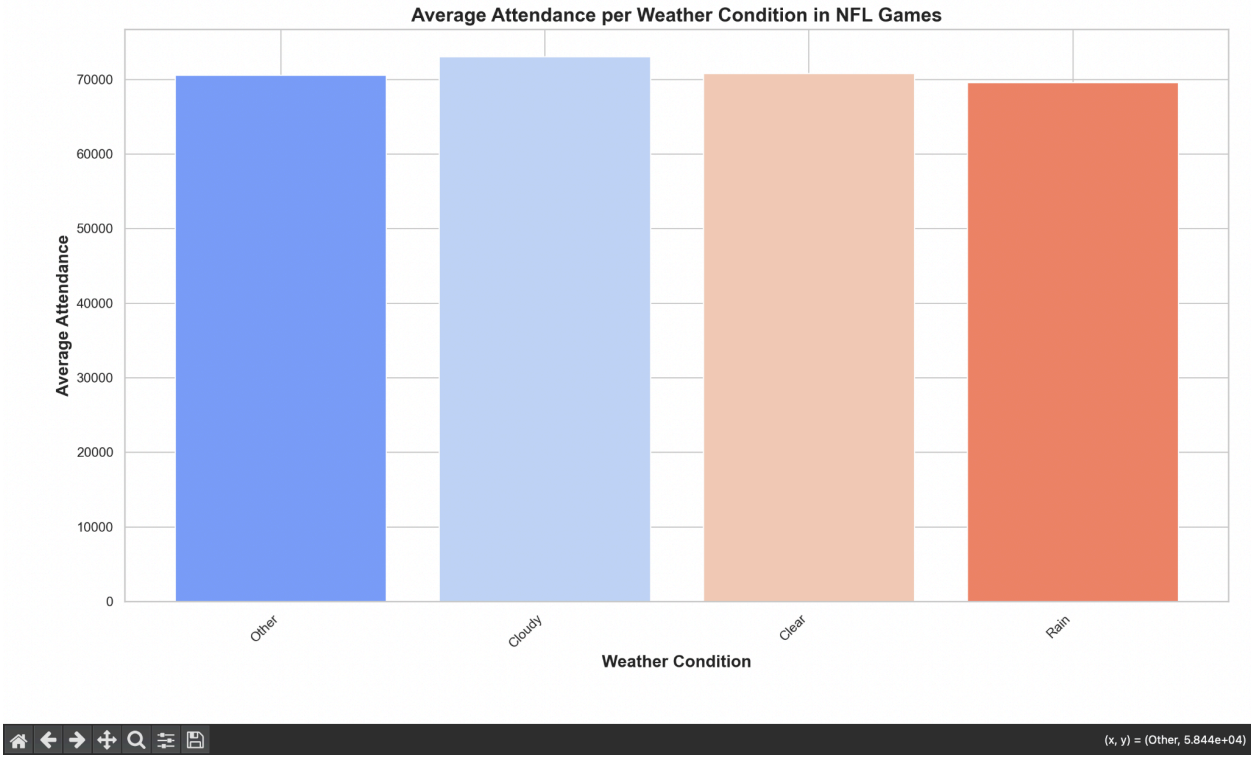
```
    average_attendance = {condition: np.mean(attendances) for condition,
attendances in weather_attendance.items()}

    # Prepare data for the graph
    conditions = list(average_attendance.keys())
    avg_attendance = list(average_attendance.values())

    sns.set(style="whitegrid")  # Set a seaborn style
    plt.figure(figsize=(10,6))  # Increase figure size for better
visibility
    plt.bar(conditions, avg_attendance,
color=sns.color_palette("coolwarm", len(conditions)))  # Use coolwarm
color palette

    # Add labels and title with customized fonts
    plt.xlabel('Weather Condition', fontsize=14, fontweight='bold')
    plt.ylabel('Average Attendance', fontsize=14, fontweight='bold')
    plt.title('Average Attendance per Weather Condition in NFL Games',
fontsize=16, fontweight='bold')

    # Display the plot with rotated labels and tight layout
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()

# Define API key and weather elements to fetch
api_key = 'N9DKDVJTSMT2WMRKEJBM7ZQ83'
weather_elements =
"datetime,tempmax,tempmin,humidity,precip,preciptype,windspeedmax,windspee
dmin,uvindex,description"
max_entries = 25

input_data1 = create_scatter_graph()[0]
input_data2 = create_scatter_graph()[1]
f = open("input.txt", 'a')
for i in range(len(input_data1)):
    f.write(f"The calculation for the average attendance for each game:
average: {input_data1[i]}, population of city: {input_data2[i]}\n")
```

Visualizations:


Percentage of Games with Different Weather Conditions
Other
1.0%
Cloudy
34.0%
Clear
38.0%
27.0%
Rain


Average attendance percentage for each NFL stadium
Population of city
2,664,452
2,314,157
1,550,542
985,843
911,311
879,293
755,078
716,577
633,218
565,239
510,704
425,115
403,364
398,431
364,136
362,656
311,097
303,255
253,855
131,062
110,717
105,744
102,865
('Chicago', 'IL')
('Houston', 'TX')
('Philadelphia', 'PA')
('Jacksonville', 'FL')
('Charlotte', 'NC')
('Indianapolis', 'IN')
('Seattle', 'WA')
('Denver', 'CO')
('Detroit', 'MI')
('Baltimore', 'MD')
('Kansas City', 'MO')
('Minneapolis', 'MN')
('Tampa', 'FL')
('Arlington', 'TX')
('New Orleans', 'LA')
('Cleveland', 'OH')
('Cincinnati', 'OH')
('Pittsburgh', 'PA')
('Glendale', 'AZ')
('Santa Clara', 'CA')
('Miami Gardens', 'FL')
('Green Bay', 'WI')
('Inglewood', 'CA')
95
100
105
110
115
Percentage of stadium filled


Average attendance percentage for each NFL stadium
Stadium City Location
Minneapolis, MN 100.42
Glendale, AZ 98.98
Chicago, IL 99.51
Inglewood, CA 100.45
Tampa, FL 92.55
Cincinnati, OH 100.99
Houston, TX 98.57
East Rutherford, NJ 96.64
Baltimore, MD 99.29
Seattle, WA 99.61
Indianapolis, IN 95.77
Philadelphia, PA 100.39
Santa Clara, CA 104.61
Green Bay, WI 95.87
Cleveland, OH 100.72
Miami Gardens, FL 101.65
Detroit, MI 101.43
Jacksonville, FL 100.21
Denver, CO 100.33
New Orleans, LA 95.63
Charlotte, NC 95.77
Arlington, TX 117.04
Kansas City, MO 96.28
Pittsburgh, PA 97.61
0
20
40
60
80
100
120
Percentage


Average Attendance per Weather Condition in NFL Games
Average Attendance
70000
60000
50000
40000
30000
20000
10000
0
Other
Cloudy
Clear
Rain
Weather Condition
(x, y) = (Other, 5.844e+04)

Instructions:
Open final_file.py and run it to create the database, the visualizations, and calculations, but wait for all three to appear before running the calculation file again.

Documentation for each function:
Alexander Liu:

- create_scatter_graph()
    - Input: None
    - Output: Creates a scatter plot with the population of the cities on the y-axis and the average percentage of how full the stadium was in each city on the x-axis. Each plot is labelled with the name of the corresponding city and the state the city is in.
- create_database()
    - Input: data_dict, city_dict, stadium_dict, tuple_lst, state_lst
        - Data_dict: Contains all the city, state, date, and attendance of each game; and the max capacity of the stadium it was played in.
        - City_dict: Contains the location id and the corresponding city name
        - Stadium_dict: A dictionary that has the names of all the cities that hosted an NFL game as keys and the max capacity of the NFL stadium in each city as the value.
        - Tuple_lst: Contains the city name, a state id, the latitude, the longitude, and the population of the city
        - State_lst: Contains the state id and the corresponding state name.
    - Output: Creates the database and tables to store the data of the variables.
- get_wiki_data()
    - Input: None
    - Output: A list with tuple_lst, state_lst, and cord_dict
        - Tuple_lst: Contains the city name, a state id, the latitude, the longitude, and the population of the city
        - State_lst: Contains the state id and the corresponding state name.
        - Cord_dict: Contains the location id and the corresponding city name. (The same as City_dict.)

Dennis Liu

- get_game_data()
    - Input: None
    - Output: A table that includes a tuple containing the city, state, date, and attendance of each game; and the max capacity of the stadium it was played in.
- get_max_capacity()
    - Input: city_dict

        - A dictionary that contains the names of all the cities that hosted an NFL game.
    - Output: A dictionary that has the names of all the cities that hosted an NFL game as keys and the max capacity of the NFL stadium in each city as the value.
- create_database()
    - Input: data_dict, city_dict, stadium_dict, tuple_lst, state_lst
        - Data_dict: Contains all the city, state, date, and attendance of each game; and the max capacity of the stadium it was played in.
        - City_dict: Contains the location id and the corresponding city name
        - Stadium_dict: A dictionary that has the names of all the cities that hosted an NFL game as keys and the max capacity of the NFL stadium in each city as the value.
        - Tuple_lst: Contains the city name, a state id, the latitude, the longitude, and the population of the city
        - State_lst: Contains the state id and the corresponding state name.
    - Output: Creates the database and tables to store the data of the variables.
- create_graph()
    - Input: None
    - Output: Creates a horizontal bar chart of each city and the average percentile of how full their stadium for all NFL games hosted in them within the timeframe of the database.

Landon Pugh

- fetch_weather_data()
    - Input: Games, api_key, weather_elements, max_entries
        - Games is a list of dates and locations of each NFL game
        - Api_key is my key for the weather API
        - Weather_elements is the column headers for the data I collected
        - Max_entries is set at 25 so only 25 values are inserted each run
    - Output:
        - This function inserts the weather data into the Weather table and writes it into a CSV file
- make_pie_chart()
    - Input: Conn
        - This is a variable assigned to the weather data retrieved from the fetch_weather_data function
    - Output:
        - This function makes a pie chart showing the percentage of games taking place in each type of weather (clear, cloudy, rainy)
- create_weather_attendance_graph()

    - Input: Conn
        - The weather data conditions and stadium attendance
    - Output:
        - This function makes a bar graph showing the average stadium attendance for each type of weather (clear, cloudy, rainy)

Changes Since Demo:

- We had 3 separate databases so we had to combine them into 1 singular one
- Fixed the formatting of some of the tables
- The weather table pulled from a hardcoded list instead of from another table in the database so we switched that
- Ensured that all tables only added 25 rows per file run

Resources:

Date	Issue Description	Location of Resource	Result
4-1-2025	Need weather data	https://www.visualcrossing.com/weather-query-builder/	Data for the weather of the days of NFL games acquired.
4-8-2025	Need data of NFL Games	https://gist.github.com/nntrn/ee26cb2a0716de0947a0a4e9a157bc1c#games	Data for NFL games acquired
4-10-2025	Need to find the location of cities where NFL games where played	https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population	Data for location of cities where NFL games are played acquired

- Weather API: https://www.visualcrossing.com/weather-query-builder/
- Wikipedia page: https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population
- NFL Games: https://gist.github.com/nntrn/ee26cb2a0716de0947a0a4e9a157bc1c#games