

Kevin Chang (kvchang@umich.edu)
Yousol Bae (yousolbae@umich.edu)
Lydia Tan (lydiatan@umich.edu)

GitHub link: <https://github.com/Kevin4335/SI206-final>

REPORT:

I. Goals for the Project

Our initial plan was to utilize the Pokemon API and the age and gender API to guess the Pokemon's ages and genders according to their name. However, we pivoted because we realized that the age and gender APIs were struggling with some of the more unique Pokemon names.

- Pokemon API: To collect Pokemon information such as Pokemon name and Pokemon type (grass, fire, water, etc.)
- Ageify API: To collect the estimated age of the Pokemon given the Pokemon's name
- Genderize API: To collect the guessed gender of the Pokemon given the Pokemon's name

Instead of the initial plan, our final plan for the project was to create a comprehensive platform that provides users with information about events happening in various cities, along with the weather conditions for those cities that the events took place in. We planned to achieve this by utilizing three different APIs:

- Ticketmaster API: To collect event information such as event name, event genre, date of the event, time of the event, and locational information such as the city name and state the event takes place in.
- Weather API: To gather weather data given a certain date and time, including temperature and condition.
- Geonames API: To retrieve a list of US cities to populate a table with their geographical information such as their city name, the state, and the population of the city.

II. Goals Achieved

We did utilize the Ticketmaster, Weather, and Geonames API. There were some modifications to the data we planned to gather and the data we ended up gathering.

- Weather API: We found that the weather API we were using did not allow us to pass in a specific date without needing to pay for those queries. So instead, the free tier allowed us to get the weather for the current date/time that the query is made. We gathered the information accordingly.
- Ticketmaster API: Because we knew that the weather API could not get the predicted weather at a specific time and date in the future, we did not gather the time of the events.

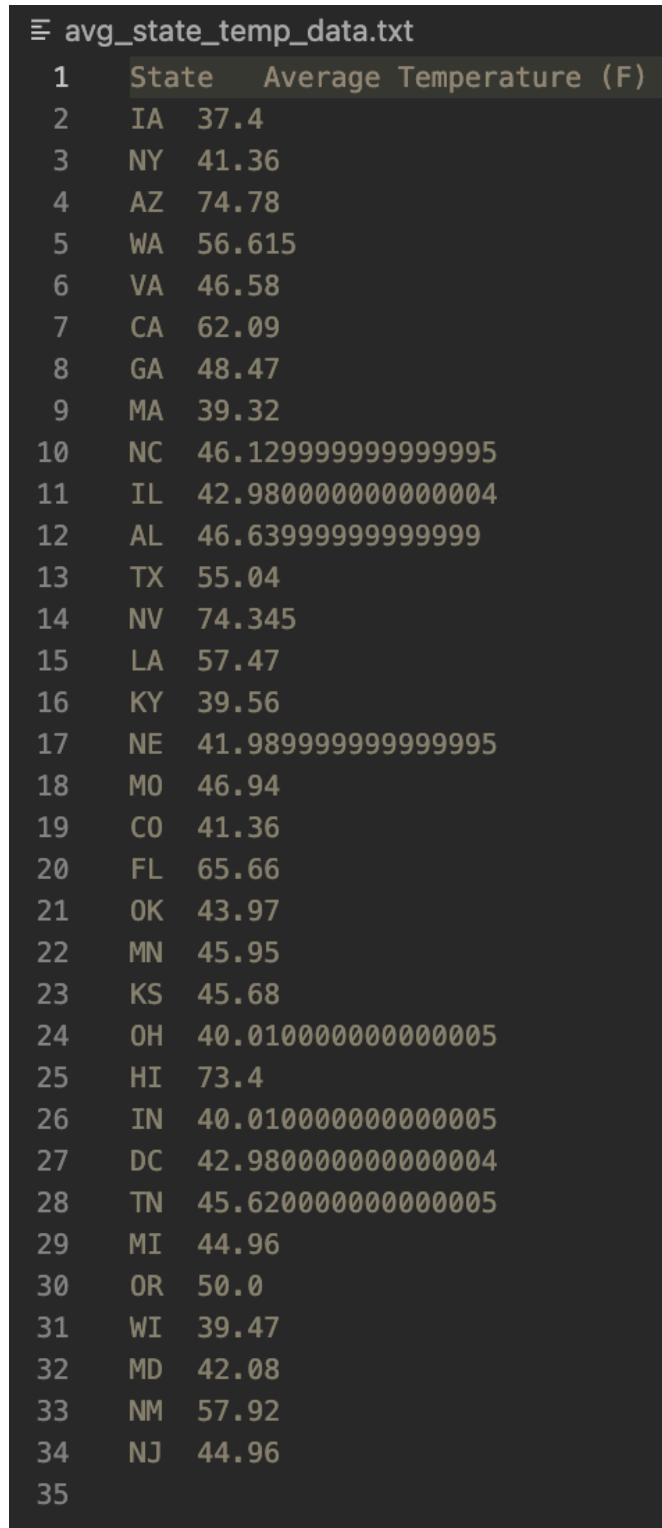
- Geonames API: We successfully gathered all of the information that we planned to get from this API.

III. Problems Faced

As we worked on this project, we realized that Ticketmaster didn't have many events listed in smaller cities. Cities with a smaller population seemed to have almost no events. To fix this, when we were grabbing cities from the Geonames api, we worked to collect cities that had a larger population. This way, we could feed those cities to the Ticketmaster api with better odds of obtaining events from those places.

IV. Calculations from the Data in the Database (Screenshots)

Figure 1. Average State Temperatures (F)



	State	Average Temperature (F)
1		
2	IA	37.4
3	NY	41.36
4	AZ	74.78
5	WA	56.615
6	VA	46.58
7	CA	62.09
8	GA	48.47
9	MA	39.32
10	NC	46.129999999999995
11	IL	42.980000000000004
12	AL	46.639999999999999
13	TX	55.04
14	NV	74.345
15	LA	57.47
16	KY	39.56
17	NE	41.989999999999995
18	MO	46.94
19	CO	41.36
20	FL	65.66
21	OK	43.97
22	MN	45.95
23	KS	45.68
24	OH	40.010000000000005
25	HI	73.4
26	IN	40.010000000000005
27	DC	42.980000000000004
28	TN	45.620000000000005
29	MI	44.96
30	OR	50.0
31	WI	39.47
32	MD	42.08
33	NM	57.92
34	NJ	44.96
35		

Figure II. Population Ranges vs. the Average Number of Events

```
≡ baskets_averages.txt
```

1	Population Ranges	Avg # of Events (F)
2	200k-300k:	89.0909090909091
3	300k-400k:	212.9
4	400k-500k:	333.5
5	500k-600k:	176.2
6	600k-700k:	608.3
7	700k-800k:	1136.5
8	>800k:	839.6666666666666

Figure III. Event Genre Distribution

```
≡ event_genre_percentages.txt
```

1	Event Genre Percentage
2	Arts & Theatre 7.06%
3	Miscellaneous 2.35%
4	Music 58.82%
5	Sports 31.76%
6	

Figure IV. Temperatures (F) in Different Cities

```
≡ temperature_data.txt
1  City      Temperature (F)
2  Des Moines 37.4
3  Rochester 37.94
4  Maryvale  77.18
5  Tacoma    50.0
6  Arlington 42.980000000000004
7  Oxnard    57.92
8  Columbus  48.02
9  Worcester 35.96
10 Moreno Valley 60.440000000000005
11 Fayetteville 49.82
12 Huntington Beach 59.18
13 Yonkers 41.0
14 Glendale 60.08
15 Aurora 42.980000000000004
16 Montgomery 51.08
17 Scottsdale 80.06
18 Irving 55.04
19 Chesapeake 48.92
20 North Las Vegas 77.18
21 Fremont 62.6
22 Baton Rouge 53.96
23 Richmond 42.08
24 Lexington 35.06
25 Paradise 78.08
26 Jamaica 46.94
27 San Bernardino 57.02
28 Huntsville 42.8
29 Spokane 44.06
30 Fontana 60.440000000000005
31 Birmingham 46.04
32 Modesto 64.94
33 Omaha 42.08
34 Kansas City 46.94
35 Long Beach 59.18
36 Mesa 78.98
37 Staten Island 44.96
38 Atlanta 48.92
39 Colorado Springs 33.8
40 Virginia Beach 50.0
41 Raleigh 46.4
42 Miami 75.02
```

V. Visualizations

Figure I. Average State Temperatures (F)

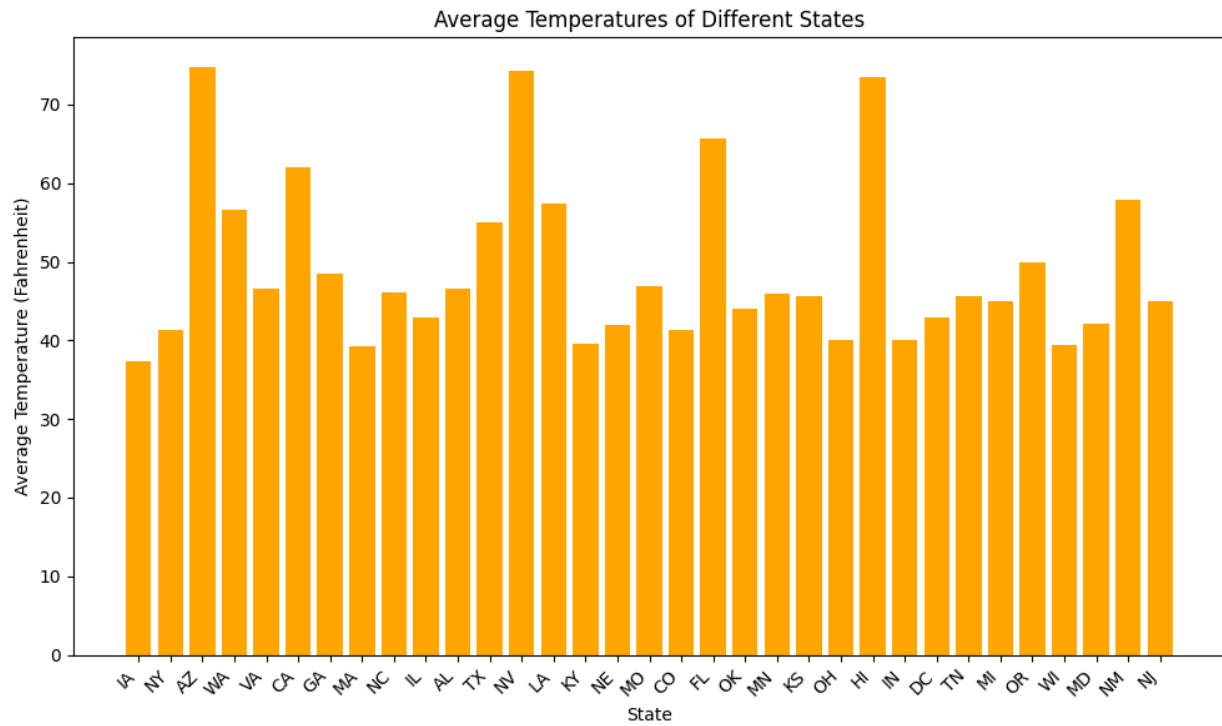


Figure II. Population Ranges vs. the Average Number of Events

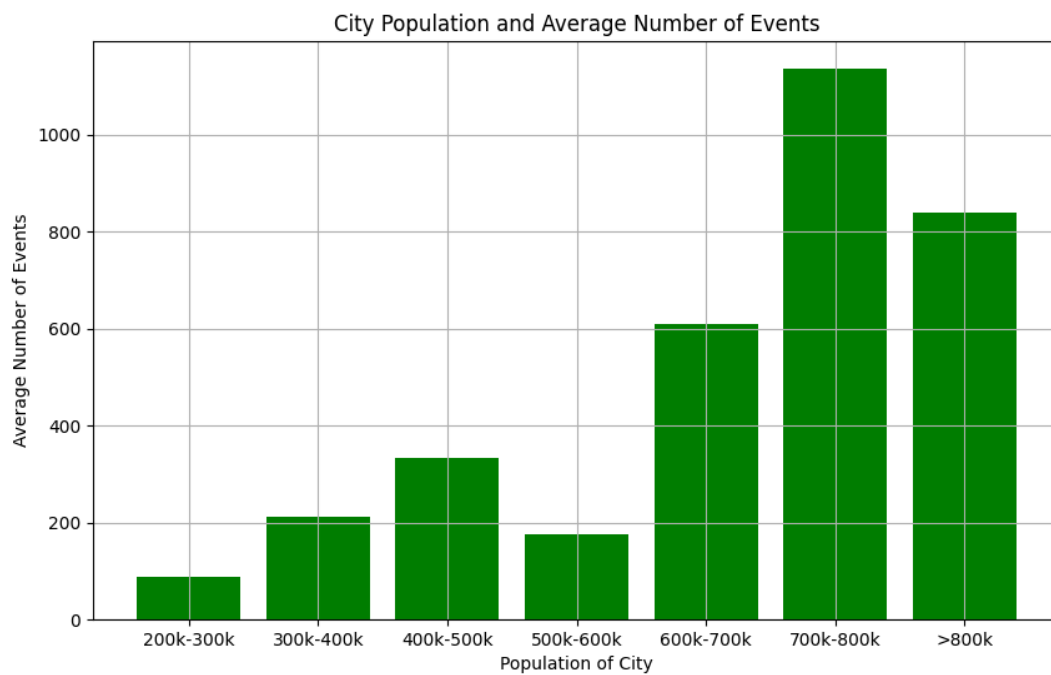


Figure III. Event Genre Distribution

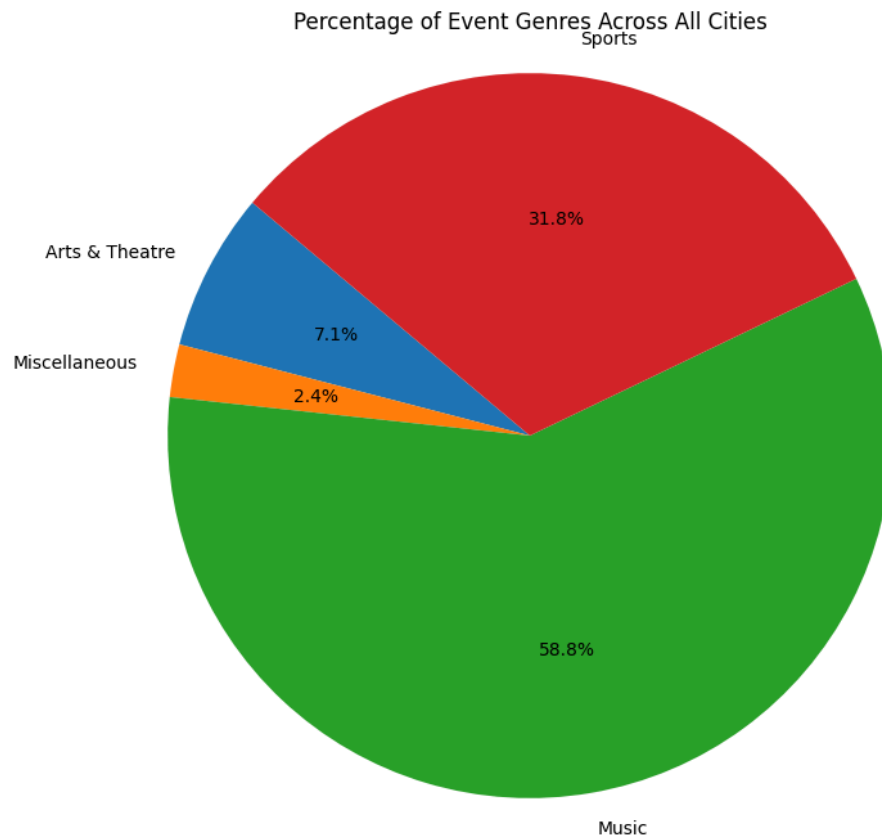
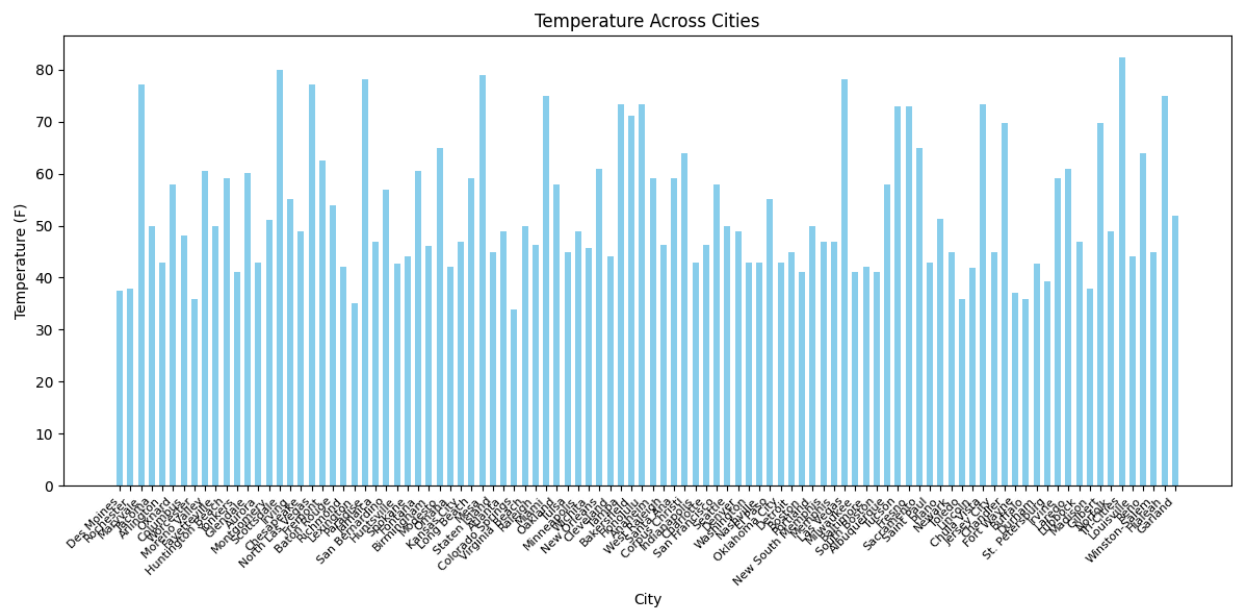


Figure IV. Temperatures (F) in Different Cities



VI. Instructions for Running the Code

First, we'll create the database:

1. Open the terminal and run:

```
python3 cities.py
```

2. Then run:

```
python3 weather.py
```

3. Run:

```
python3 events.py
```

Now, we'll create the graphs and the text file calculations:

1. Run:

```
python3 graph.py
```

VII. Documentation for Functions

cities.py:

```
def fetch_us_cities(num_cities):
```

Fetches US cities data from the GeoNames API.

Args:

- num_cities (int): The number of cities to fetch.

Returns:

- list: A list of dictionaries containing city data (name, state, population).

events.py:

```
def fetch_event(city_name, state_id):
```

Fetches event data from the Ticketmaster API for a given city and state.

Args:

- city_name (str): The name of the city.
- state_id (str): The state code (e.g., "NY" for New York).

Returns:

- dict: A dictionary containing event data obtained from the API response.

weather.py:

```
def fetch_weather(city_name, state_code):
```

Fetches current weather data for a given city and state from the WeatherAPI.

Args:

- city_name (str): The name of the city.
- state_code (str): The state code of the city.

Returns:

- dict: A dictionary containing weather information (temperature, condition, wind speed).

`def fetch_astro(city_name, state_code):`

Fetches astronomical data (sunrise and sunset times) for a given city and state from the WeatherAPI.

Args:

- city_name (str): The name of the city.
- state_code (str): The state code of the city.

Returns:

- dict: A dictionary containing astronomical information (sunrise, sunset).

`def get_weather_data(cities_data):`

Fetches weather data for multiple cities and inserts it into the weather table in the database.

Args:

- cities_data (list): A list of tuples containing city data (city_id, city_name, state, population).

Returns: nothing

`def get_times_data(cities_data):`

Fetches astronomical data for multiple cities and inserts it into the astro table in the database.

Args:

- cities_data (list): A list of tuples containing city data (city_id, city_name, state, population).

Returns: nothing

VIII. Resources Used

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
4/25/24	Help with understanding different types of graphs possible with matplotlib	https://www.geeksforgeeks.org/matplotlib-tutorial/	Yes, this site gave valuable knowledge on how to use matplotlib, as well as multiple types of graphs