SI 206 Final Project Report

Keely Nykerk, Avery Schiff, Emily Williams

https://github.com/emilygrwilliams/Final_SI_206

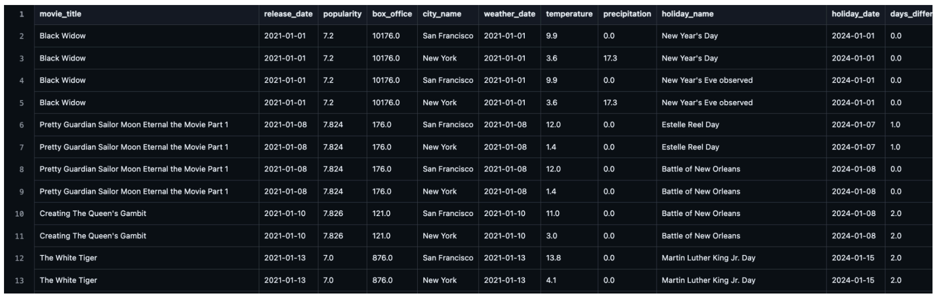
Goals

Our project aims to explore the relationship between movie popularity and external factors using data gathered from three APIs and one website, which we parsed using Beautiful Soup. First, we plan to parse the Marvel website to gather a list of Marvel character names. We will then use the TMDB API to extract all movies with Marvel character names. This API will also collect key movie details, including title, release date, and average vote. Additionally, we will utilize the Calendarific API and Mediostat (a Python library and API) to gather contextual information that may influence movie popularity, such as holidays near the release date and weather conditions like temperature and precipitation on the release day. By analyzing this data, we aim to understand how these external factors correlate with a movie's success.

Problems Faced

- Initially, we had a lot of trouble with APIs
- Figuring out how to create tables
- We did not know how we would do the queries for the movie API, but then we used Beautiful Soup and shifted our focus to Marvel characters as a query to refine the data in a way that was interesting to us
- Used tomorrow.io for weather API initially, but quickly ran out of free calls and needed to find a new one
- It was initially difficult to figure out how to only put 25 rows into the database at once

Calculation File



movie_title	release_date	popularity	box_office	city_name	weather_date	temperature	precipitation	holiday_name	holiday_date	days_differ
Black Widow	2021-01-01	7.2	10176.0	San Francisco	2021-01-01	9.9	0.0	New Year's Day	2024-01-01	0.0
Black Widow	2021-01-01	7.2	10176.0	New York	2021-01-01	3.6	17.3	New Year's Day	2024-01-01	0.0
Black Widow	2021-01-01	7.2	10176.0	San Francisco	2021-01-01	9.9	0.0	New Year's Eve observed	2024-01-01	0.0
Black Widow	2021-01-01	7.2	10176.0	New York	2021-01-01	3.6	17.3	New Year's Eve observed	2024-01-01	0.0
Pretty Guardian Sailor Moon Eternal the Movie Part 1	2021-01-08	7.824	176.0	San Francisco	2021-01-08	12.0	0.0	Estelle Reel Day	2024-01-07	1.0
Pretty Guardian Sailor Moon Eternal the Movie Part 1	2021-01-08	7.824	176.0	New York	2021-01-08	1.4	0.0	Estelle Reel Day	2024-01-07	1.0
Pretty Guardian Sailor Moon Eternal the Movie Part 1	2021-01-08	7.824	176.0	San Francisco	2021-01-08	12.0	0.0	Battle of New Orleans	2024-01-08	0.0
Pretty Guardian Sailor Moon Eternal the Movie Part 1	2021-01-08	7.824	176.0	New York	2021-01-08	1.4	0.0	Battle of New Orleans	2024-01-08	0.0
Creating The Queen's Gambit	2021-01-10	7.826	121.0	San Francisco	2021-01-10	11.0	0.0	Battle of New Orleans	2024-01-08	2.0
Creating The Queen's Gambit	2021-01-10	7.826	121.0	New York	2021-01-10	3.0	0.0	Battle of New Orleans	2024-01-08	2.0
The White Tiger	2021-01-13	7.0	876.0	San Francisco	2021-01-13	13.8	0.0	Martin Luther King Jr. Day	2024-01-15	2.0
The White Tiger	2021-01-13	7.0	876.0	New York	2021-01-13	4.1	0.0	Martin Luther King Jr. Day	2024-01-15	2.0

This is a snapshot of our CSV file, which finds which movies are near holidays and connects that to their rating and weather values. Some movies have no holidays listed as none are within 2 days of them.

Visualizations

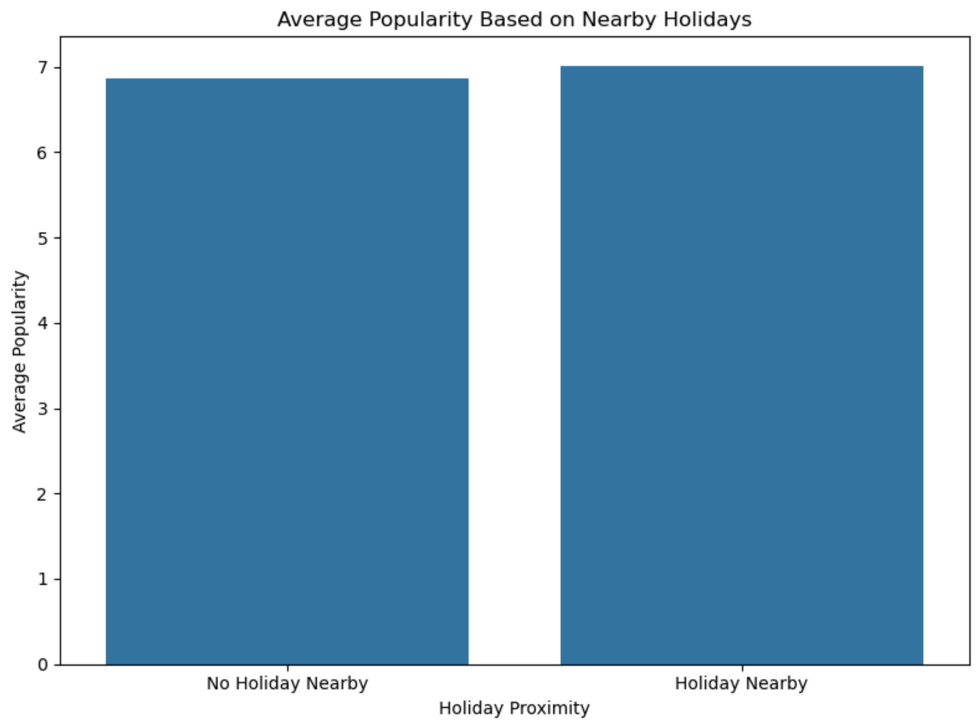


Figure 1: This bar graph compares the popularity (vote average out of 10) of the movies in the database to a boolean of whether or not the movie release date is near a national holiday.

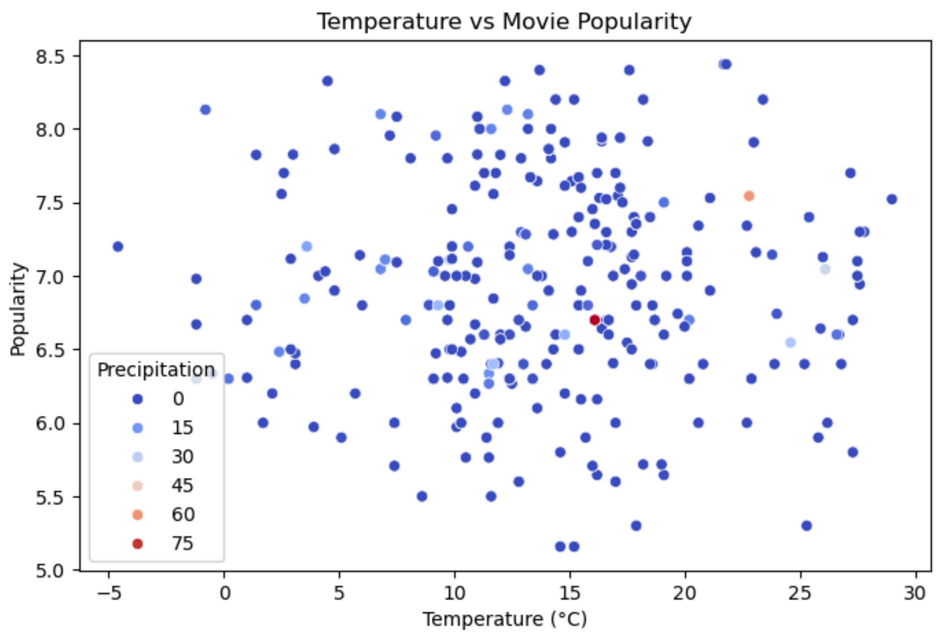


Figure 2: This scatter plot compares the movie's popularity (vote average out of 10) in the database to the temperature and precipitation intensity on the movie’s release date.

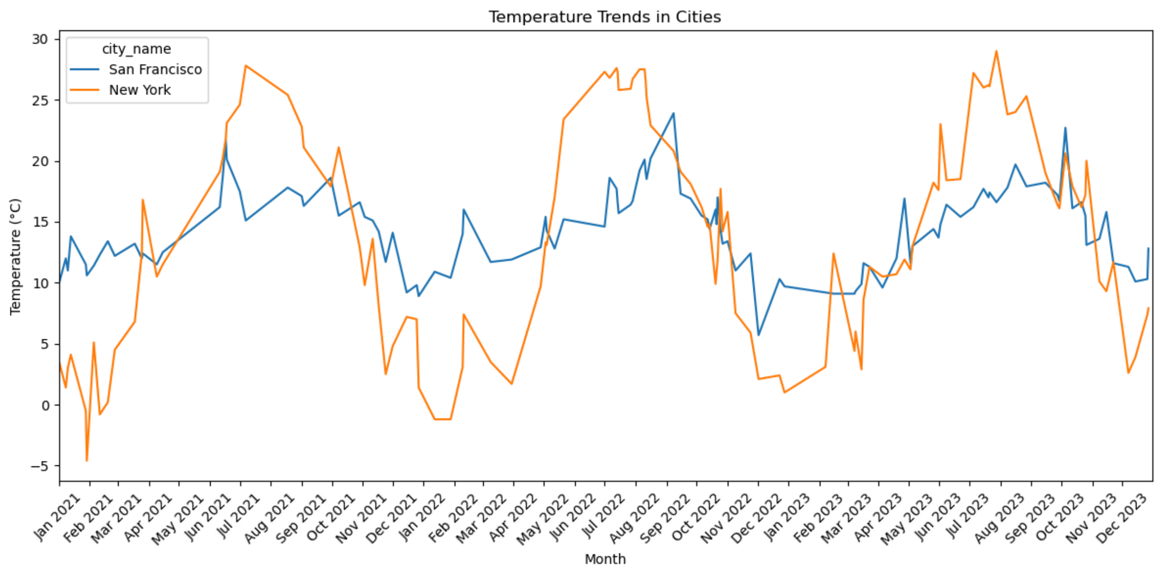


Figure 3: This bar graph compares the temperature throughout the year in both locations stored in the database. It gives insight into how temperate it is on average in both locations.

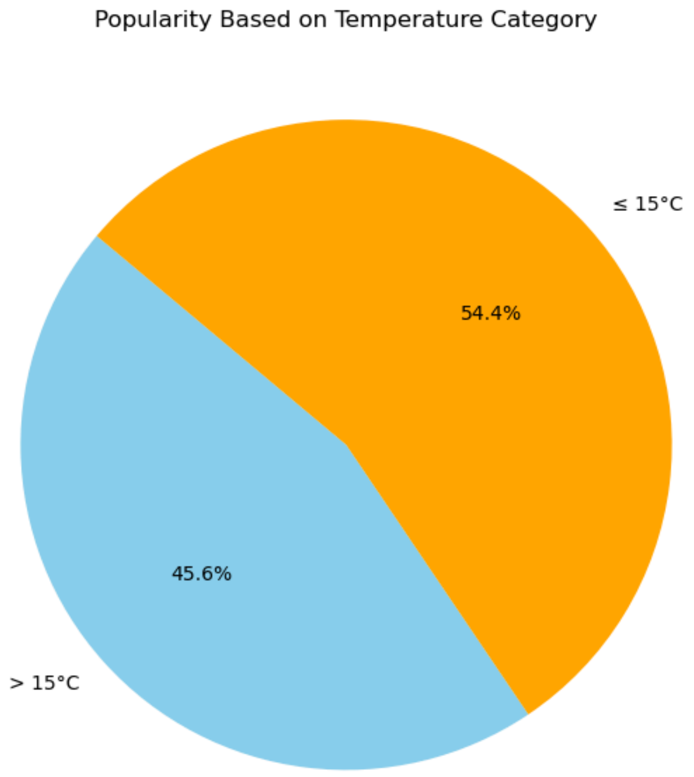


Figure 4: This pie chart compares the popularity of the movies to the temperature on their release date.

Instructions for running code:

To run the project, start by installing the required dependencies using the commands:

- pip install meteostat
- pip install requests
- pip install beautifulsoup4
- pip install pandas

Next, run data_collection.py **SEVEN** times.

- Each run of this file will put 25 rows into each table (except for the weather table)
- This script will collect data from the APIS, perform web scraping, and populate the SQLite database (movies.db).

Then, run data_analysis.py, which will query the database, perform calculations, and generate visualizations based on the collected data.

Code Documentation

Data Limitation Strategy

- Restricted database entries to 25 items per table per script run (except for weather due to the large number of rows that the finalized database has)
- Only saved movies with:
- Release year 2021 to 2023.
- Vote count greater than 100.
- Filtered weather and holiday data to relevant locations and dates (United States)
- i.e. Big cities and national/ local holidays

Data_collection.py

- **Function**: Get_db_connection
    - **Purpose**: Establishes and returns a connection to the SQLite database
    - **Input**: None
    - **Output**: SQLite database connection object.
- **Function**: marvel_list (url)
    - **Purpose**: Scrapes the marvel website to extract a list of character names.
    - **Input**: The URL of the Marvel characters webpage
    - **Output**: character_list, a sorted list of Marvel character names
- **Function**: fetch_movie_data (query)
    - **Purpose**: Fetches movie data from the TMDB API based on a given query.
    - **Input**: query (string, a keyword to search for movies)
    - **Output**: response, dictionary JSON object containing movie details
- **Function**: save_movie_data (movies)
    - **Purpose**: Saves movie data to the movies table in the database

    - **Input**: movies, list of dictionaries containing movie metadata
    - **Output**: (None) Writes to the database
- **Function**: fetch_holiday_data (country, year)
    - **Purpose**: Fetches holiday data form the Calendarific API for a specific country and year.
    - **Input**: string, the country code (US)
    - **Output**: JSON object containing holiday details
- **Function**: save_holiday_data (holidays)
    - **Purpose**: Saves holiday data to the holidays table in the database
    - **Input**: holidays, list of dictionaries containing the holiday data
    - **Output**:
- **Function**: save_city_data (city_name, latitude, longitude)
    - **Purpose**: Saves city data into the database if not already present
    - **Input**:
        - city_name (string) - Name of the city
        - latitude (float) - Latitude coordinate of city
        - longitude (float) - Longitude coordinate of the city
    - **Output**: None (data is saved to database)
- **Function**: fetch_weather_data (latitude, longitude, start_date, end_date)
    - **Purpose**: Fetches historical weather data for a specific location and data range using the Meteostat library.
    - **Input**:
        - latitude (float) - Latitude coordinate
        - longitude (float) - Longitude coordinate
        - start_date (datetime object) - Start date of the range
        - end_date (datetime object) - End date of the range
    - **Output**: Dataframe containing data
- **Function**: save_weather_data (weather_data, city_name)
    - **Purpose**: saves weather data into database for a specific city
    - **Input**:
        - weather_data (DataFrame) - including dates, average temperatures, and precipitation levels
        - City_name (string) - Name of the city
    - **Output**: None (data is saved to database)

Data_analysis.py

- **Function**: get_db_connection ()
    - **Purpose**: Connects to the SQLite database (movies.db)
    - **Input**: None
    - **Output**: Returns a connection object to the movies.db SQLite database.
- **Function**: analyze_data ()
    - **Purpose**: Fetches and prepares data for analysis by joining data from multiple tables. The query computes the absolute difference in days between movie

release dates and holiday dates and retrieves relevant weather data for the movie's release date.
        - **Input**:
            - No direct inputs, utilizes database, movies.db
        - **Output**: Pandas DataFrame
- **Function**: create_bar_plot (df)
    - **Purpose**: Creates a bar plot to compare the average popularity of movies released near a holiday versus those released with no holiday nearby.
    - **Input**: A Pandas DataFrame from analyze_data ()
    - **Output**: A bar plot of average popularity based on nearby holidays
- **Function**: create_scatter_plot (df)
    - **Purpose**: Generates a scatter plot to analyze the relationship between temperature and movie popularity, using precipitation levels as a hue.
    - **Input**: A Pandas DataFrame from analyze_data ()
    - **Output**: A scatter plot of temperature vs movie popularity
- **Function**: create_line_graph (df)
    - **Purpose**: Create a line graph showing the dataset's temperature trends for different cities over time.
    - **Input**: A Pandas DataFrame from analyze_data ()
    - **Output**: A line graph of temperature trends in cities
- **Function**: create_pie_chart (df)
    - **Purpose**: Produce a pie chart comparing box office revenue for movies released on days categorized by temperature.
    - **Input**: A Pandas DataFrame from analyze_data ()
    - **Output**: A pie chart of box office revenue based on temperature category

Resources Used

- Visual Studio Code: To code the project
- ChatGPT: Help with debugging and creating certain elements of code
- DB Browser for SQ Lite: Observing the database
- requests: Sends HTTP requests to APIs and websites.
- sqlite3: Manages SQLite database connections and queries.
- os: Interacts with the file system (e.g., checking file existence).
- datetime: Handles date and time operations.
- BeautifulSoup: Parses HTML for web scraping.
- re: Performs regular expression matching.
- meteostat: Fetches historical weather data.
- pandas: Manages and analyzes data in tabular format.
- matplotlib.pyplot: Creates plots and visualizations.
- seaborn: Generates advanced statistical plots.
- numpy: Performs numerical operations on arrays.
- matplotlib.dates: Formats dates on plots.

Key Takeaways

- Several confounding variables affect movie popularity including but certainly not limited to holidays and weather
- Based on our charts, there is a slight preference for movies released during mid-range temperatures (spring/fall)
- Since we analyzed averages, there could be a possible correlation between an increase in movie releases and a rise in the number of movie-goers
- Overall, there is no robust data to support a definitive conclusion