PART 5 – Report (100 points)

Github Repository: https://github.com/fateha-b/206_FinalProject.git

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

Our original goal was to investigate how weather patterns in college towns influence student behavior, particularly in terms of reading habits and online engagement. To do this, we planned to use three different APIs: the Open-Meteo API to collect daily weather data, the New York Times Books API to track popular book genres and titles, and the Reddit API to examine top posts and discussions within student-focused subreddits like r/uofm and r/college. The weather data we hoped to gather included temperature, precipitation, and overall conditions for specific dates and locations. From the NYT Books API, we intended to collect lists of best-selling books and their categories, while from Reddit, we aimed to gather post titles, scores, comment counts, and timestamps to measure engagement levels. All data would be stored in a database to help us analyze how shifts in weather may correlate with changes in reading interest or social media activity.

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

We successfully achieved our original goals by working with all three APIs as planned. From the Open-Meteo API, we collected daily weather data for Ann Arbor, including the date, minimum and maximum temperatures, total precipitation levels and humidity. This allowed us to identify weather patterns over several weeks. Using the NYT Books API, we gathered data on best-selling book genres each week, including the list name, number of books listed per genre, and average rankings. These were calculated using SQL SELECT statements from the stored data. This helped us analyze which genres gained popularity under different weather conditions. From the Reddit API, we pulled post data from college-related subreddits such as r/uofm and r/college, collecting information like post titles, scores, number of comments, and posting dates. All of this data was stored in a structured database, and we used it to explore correlations between weather conditions, book interest, and student online activity. By integrating all three APIs, we were able to generate meaningful insights and create visualizations that captured behavioral trends across platforms in relation to changing weather.

Findings:

- Hardcover Fiction and Young Adult genres were consistently popular, especially during weeks with light to moderate precipitation.
    - Supports the idea that students may turn to leisure reading (fiction/YA) during colder or rainier weeks.
- Series Books had a spike in listings during a week with clear weather, suggesting casual or ongoing reading may rise when students are outdoors or in a more relaxed mindset.
    - Suggests that weather influences the type of content students engage with.
- Reddit posts increased in volume and engagement during weeks with more rain (i.e., more posts and higher comment counts).

    - Supports the hypothesis that students spend more time online when the weather is worse.

- Weather patterns (especially rain) aligned with lower-ranking books gaining traction, which could suggest students explore beyond top-ranked titles when stuck indoors.
    - Reinforces our idea that weather doesn't just affect engagement quantity, but possibly variety too.
- Dry weeks showed more consistent genres (business, education), possibly linked to midterms or academic focus when students are out and about.
    - Indicates that study-related content may rise in drier, more active weeks.

Overall: The analysis confirmed that weather does appear to influence both the type and level of student engagement across platforms. Rainier days were linked with higher Reddit use and more engagement with fiction/YA genres, while clearer days favored more practical genres and steady online activity.

3. **The problems that you faced (10 points)**

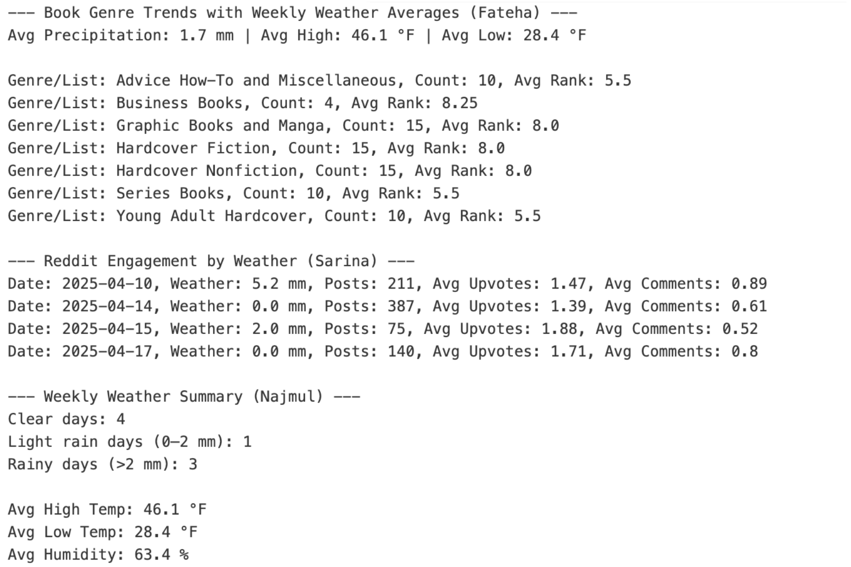
    a. One challenge we faced was switching from the Google Books API to the NYT Books API, since Google lacked consistent popularity data. While the NYT API solved that issue, its weekly format didn't initially align with the daily data from the other two APIs. However, this ended up working better — averaging weather and Reddit data weekly gave us more consistent and meaningful comparisons.

    b. Throughout the course of the project, we encountered several technical and coordination challenges that required collaborative problem-solving. Initially, we planned to use the OpenWeather API to collect daily weather data for Ann Arbor; however, we quickly realized that the free version of the API only provided current weather data and not historical information for specific dates. To address this limitation, we switched to the Open-Meteo API, which allowed us to access daily historical data for 2024 and 2025 without requiring an API key. Another issue we faced was that Open-Meteo's data was often a few days behind the current date, so we implemented logic in our script to stop data collection once it reached the most recent available data, avoiding errors from attempting to retrieve future values.

    We also encountered version control issues while using Git—particularly with git pull and git push—due to branch divergence and conflicts with binary files like our shared database. We resolved these problems by configuring pull settings correctly and learning how to handle merge conflicts. Lastly, one of our biggest challenges was integrating all of our data into a single database. At first, each team member was working in their own database file, which made collaboration difficult. We overcame this by consolidating all data into a shared final_project.db and standardizing our table structures. These challenges strengthened our teamwork and deepened our understanding of APIs, data visualization, and collaborative development practices.

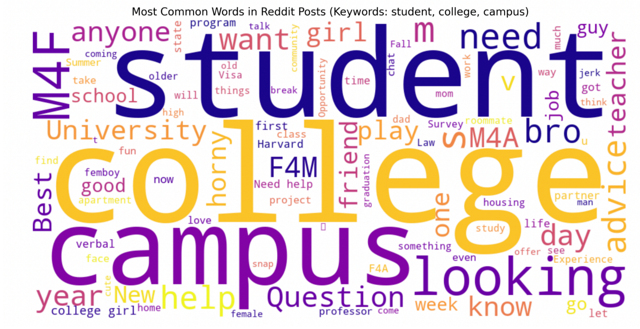
c. Another challenge we faced was the limited amount of data available from the Reddit API, as PRAW does not provide access to historical posts. This meant we only had access to Reddit data that we collected in real time during the specific dates we worked on the project. As a result, we couldn't retrieve older posts that matched our keywords ("college," "student," "campus") for past dates, so we aligned Reddit activity with the closest available days by running our script manually during that period. To work around this, we ran our Reddit data collection script on four different days between April 10-20 to ensure we were capturing posts that aligned with our weather and NYT data. Although this approach limited the size of our Reddit dataset, it still allowed us to make meaningful observations about engagement trends across different weather conditions.

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

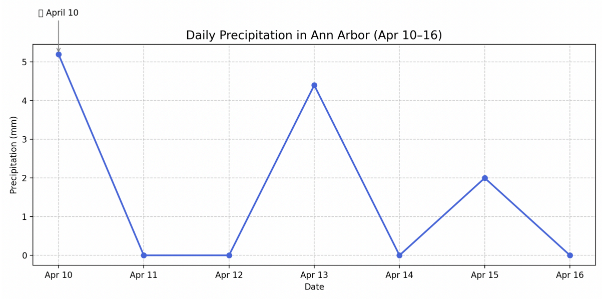
a. 

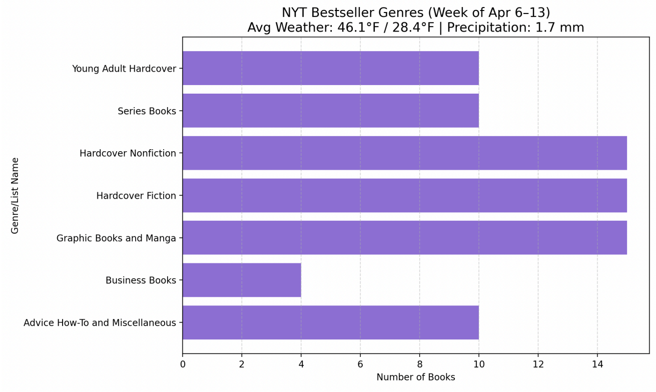
```
--- Book Genre Trends with Weekly Weather Averages (Fateha) ---
Avg Precipitation: 1.7 mm | Avg High: 46.1 °F | Avg Low: 28.4 °F

Genre/List: Advice How-To and Miscellaneous, Count: 10, Avg Rank: 5.5
Genre/List: Business Books, Count: 4, Avg Rank: 8.25
Genre/List: Graphic Books and Manga, Count: 15, Avg Rank: 8.0
Genre/List: Hardcover Fiction, Count: 15, Avg Rank: 8.0
Genre/List: Hardcover Nonfiction, Count: 15, Avg Rank: 8.0
Genre/List: Series Books, Count: 10, Avg Rank: 5.5
Genre/List: Young Adult Hardcover, Count: 10, Avg Rank: 5.5

--- Reddit Engagement by Weather (Sarina) ---
Date: 2025-04-10, Weather: 5.2 mm, Posts: 211, Avg Upvotes: 1.47, Avg Comments: 0.89
Date: 2025-04-14, Weather: 0.0 mm, Posts: 387, Avg Upvotes: 1.39, Avg Comments: 0.61
Date: 2025-04-15, Weather: 2.0 mm, Posts: 75, Avg Upvotes: 1.88, Avg Comments: 0.52
Date: 2025-04-17, Weather: 0.0 mm, Posts: 140, Avg Upvotes: 1.71, Avg Comments: 0.8

--- Weekly Weather Summary (Najmul) ---
Clear days: 4
Light rain days (0–2 mm): 1
Rainy days (>2 mm): 3

Avg High Temp: 46.1 °F
Avg Low Temp: 28.4 °F
Avg Humidity: 63.4 %
```

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

a. 

Most Common Words in Reddit Posts (Keywords: student, college, campus)

b. 

c. 

6. **Instructions for running your code (10 points)**
   a. All scripts should be run from the same directory. First, make sure Python is installed and download the required libraries using pip install wordcloud seaborn requests. Then run the following scripts in order:
      i. **nyt_books.py**
      ii. **reddit_api.py (does not store historical data)**
      iii. **weather_2025_data.py**
   b. These scripts will collect data from the NYT Books, Reddit, and Open-Meteo APIs and store everything in the final_project.db database.
   c. Next, run **part3.py** to analyze the data. This will generate the output file **weekly_analysis.txt**, which contains summary insights.
   d. Finally, run **visualizations.py** to create charts and word clouds. Double check that **word cloud, matplot,** and **seaborn** are installed, if not already, before running this last step.
7. **Documentation for each function that you wrote. This includes describing the input and output for each function (20 points)**

Function	**Input & Output**
(nyt_books.py) drop_books_table()	Deletes the Books table from the SQLite database if it exists. This is useful when resetting the table structure before inserting new data. **Inputs**: None **Outputs**: None (prints confirmation to console)
(nyt_books.py) create_tables()	Creates two tables in the SQLite database: Authors

	and Books. The Authors table stores unique author names with auto-incrementing IDs, while the Books table stores information about each book, including its title, rank, date, and associated author ID. **Inputs**: None **Outputs**: None (creates tables in the database)
(nyt_books.py) get_or_create_author_id(conn, cur, author_name)	Checks if an author already exists in the Authors table. If found, returns their existing ID. If not, inserts the new author and returns the newly created ID. **Inputs:** - conn: The active SQLite database connection - cur: The SQLite cursor object - author_name: A string with the author's full name **Outputs:** Returns the integer author_id for the given author
(nyt_books.py) insert_books()	Fetches data from the NYT Books API for a specific genre and date, then inserts unique books into the Books table. Also links books to the corresponding author in the Authors table. **Inputs**: None (uses predefined API_KEY, LIST_NAME, and DATE from global variables) **Outputs**: None (inserts data into the database and prints how many books were added)
(nyt_books.py) main()	Main driver function for the script. It sets up the database structure and inserts the latest book data from the API. It optionally allows for dropping the Books table if the line is uncommented. **Inputs**: None **Outputs**: None (calls setup and insert functions)

Weather API: create_db()	**Creates** the SQLite table Weather2025 if it doesn't already exist. **Inputs**: None **Outputs**: None (executes SQL command to create table)
get_next_start_date()	**Fetches** the latest date from the Weather2025 table and returns the next day's date to continue data collection. **Inputs**: None **Outputs**: datetime object

store_data(data)	**Processes** and stores weather data into the SQLite table. Skips incomplete or duplicate data. **Inputs**: JSON data **Outputs**: None (writes to DB, prints log)
main()	Controls the full workflow: creates the DB, figures out the next date, fetches data, and stores it. **Inputs**: None **Outputs**: None (prints updates)
c_to_f(c)	Converts Celsius to Fahrenheit. **Inputs**: temperature in Celsius c **Outputs**: temperature in Fahrenheit

Reddit API: fetch_and_store()	Fetches recent Reddit posts containing at least one specified keyword and stores them into a new SQLite table. Migrates old data only if posts mention a keyword. Prevents duplicates by using unique post IDs. **Inputs**: list of keyword strings (e.g., [“college”, “student”, “campus”]) **Outputs**: None (executes SQL commands to create, rename, and insert into table)

8. **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?)
April 10, 2025	Needed help choosing a more structured alternative to Google Books for consistent data tied to dates and popularity.	ChatGPT	Yes, ChatGPT helped identify the NYT Books API as a better source for weekly genre-based book rankings, which solved the inconsistency issue with Google Books.
April 11, 2025	Unsure whether the weekly granularity of the NYT Books API would mismatch with the daily data collected from Reddit and Open-Meteo.	ChatGPT	Yes, ChatGPT helped reframe the problem and suggested averaging the daily data by week instead. This made comparisons more consistent across datasets.
April 13,	Encountered a sqlite3.OperationalError when	SQLite - unique	Yes, the thread helped confirm the issue was caused by trying to

2025	inserting duplicate authors into the Authors table.	constraint failed	insert an existing author name without checking first. I added the get_or_create_author_id() function to prevent duplicates.
April 14, 2025	How to fetch temperature, precipitation, and humidity by date	Open-Meteo API Docs	Confirmed API parameters and timezone configuration.
April 14th, 2025	How to add data from select dates from Reddit API	PRAW description	Yes, explained RedditAPI does not save historical data, only current date, therefore we need to manually run for the dates we worked on the project to have as much versatility in data as we could control
April 15, 2025	How to convert Celsius to Fahrenheit	ChatGPT	Learned the correct formula: (C × 9/5) + 32.
April 15, 2025	Figuring out how to format the NYT Books API request URL using a specific date and list name.	get /lists/{date}/{list}.json	Yes, the documentation provided the correct structure for forming API requests using {date} and {list} parameters.
April 16, 2025	Understanding the structure of the response JSON and locating where the book list and metadata (e.g., title, author, rank) were stored.	Books API | Dev Portal	Yes, it helped identify the correct path (results.books) and fields for parsing the data we needed to insert into our database.

8. Changes after grading session

1. Reddit API file was altered to limit the amount of items per run to 25. It was previously 25 items per keyword and as we're searching for 3 keywords, the total became 75 items per run. It has not been fixed to have a counter across all searches, and once it hits 25 it breaks.
2. The Reddit table in final_project.db was edited to split up keywords into their own columns and split the date column into 3 separate columns. It now has a separate column for day, month, and year, with integers as the data. There is an old Reddit table as the code migrates the data to the new one instead of deleting it as we need to preserve our old data.
3. After the grading session, the NYT Books section was updated to reflect the normalized database structure. The list_name column in the Books table was removed and replaced

with a foreign key genre_id referencing the Genres table. The published_date was also split into separate year, month, and day columns to support more precise filtering and analysis. Associated queries and visualizations were updated to use JOIN statements and the new date fields.