WN 24 SI 206 Final Project Report
Aiden Westphal & Asa Garcia
Link to Repository

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

Originally, we planned on doing a project using an API of NFL Player Data and an API of NCAA Football player data. We decided to scrap this idea because there would not have been enough data point references due to there only being 32 NFL teams. We pivoted to using Spotify and Shazam API, but unfortunately Shazam was not accepting new applications. We then pivoted to using the Spotify API along with the NASA image of the day API.

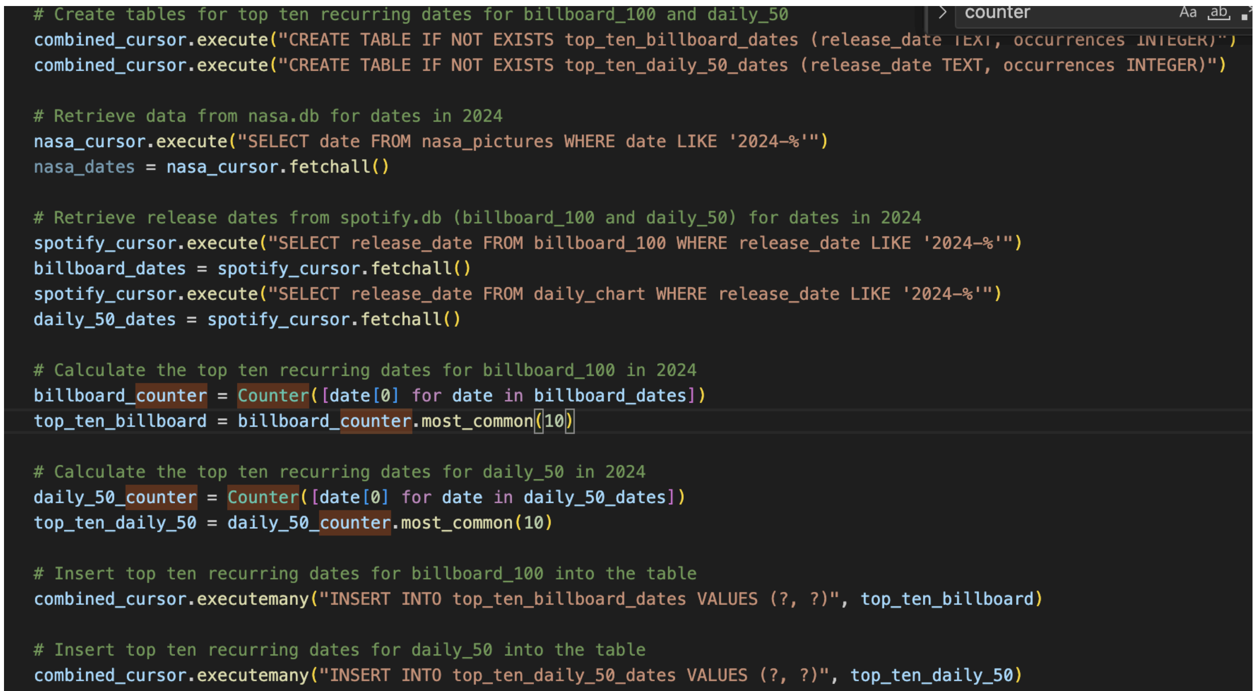
2. The goals that were achieved including what APIs/websites you actually worked with and what data you did gather (10 points)

With these API's we gathered the Billboard Top 100 and Daily Global Top 50 charts from Spotify, as well as every image of the day since 01/01/2024 from NASA. Our goal was to create a program that sorted through all of the songs on the charts and determine which Release Dates charted the most songs, and then display the image of the day from the days that charted the most songs.

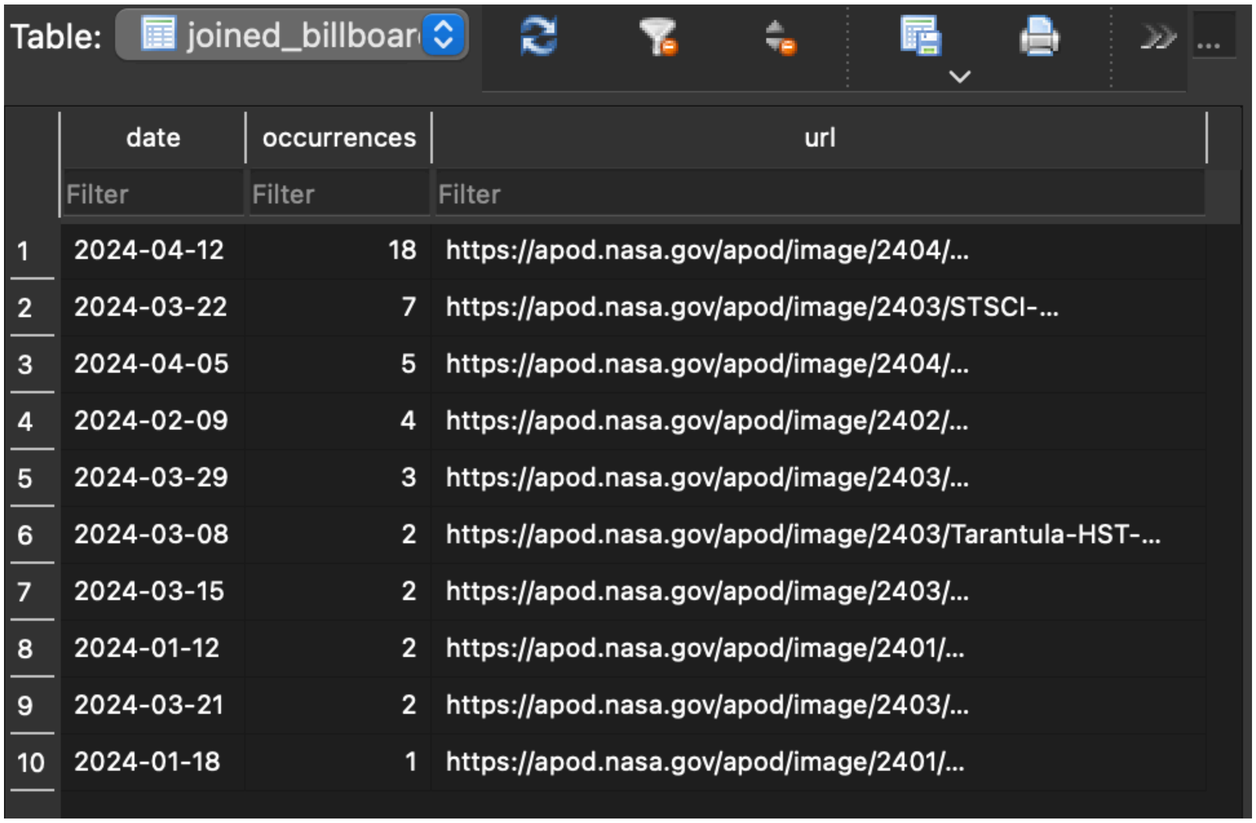
3. The problems that you faced (10 points)

A big problem that we faced was figuring out an algorithm that could add 25 items to the database at the time and joining the two databases by date and url. We ended up adding the tables to a combined database, so we were able to join them and output the data together in order to visualize something meaningful.

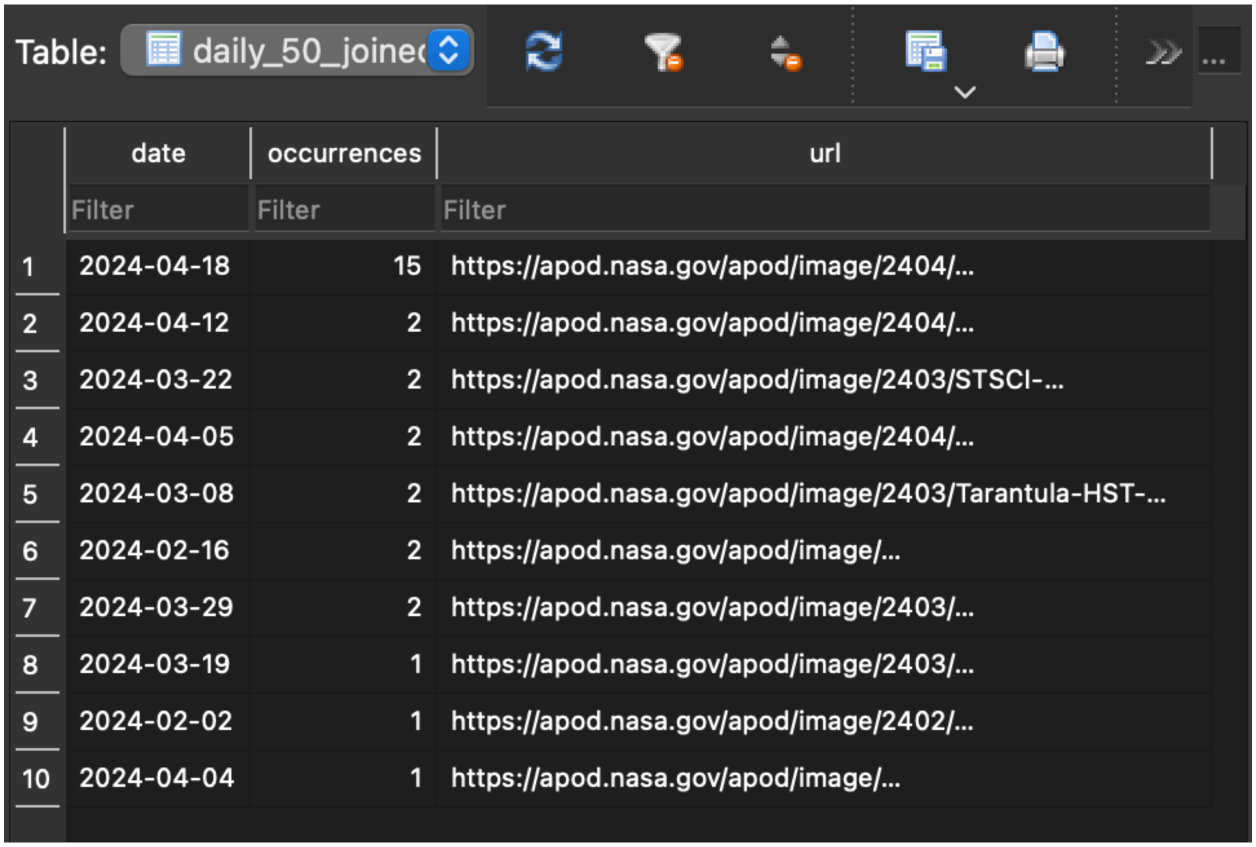
4. The calculations from the data in the database (i.e. a screen shot) (10 points)



```
# Create tables for top ten recurring dates for billboard_100 and daily_50
combined_cursor.execute("CREATE TABLE IF NOT EXISTS top_ten_billboard_dates (release_date TEXT, occurrences INTEGER)")
combined_cursor.execute("CREATE TABLE IF NOT EXISTS top_ten_daily_50_dates (release_date TEXT, occurrences INTEGER)")

# Retrieve data from nasa.db for dates in 2024
nasa_cursor.execute("SELECT date FROM nasa_pictures WHERE date LIKE '2024-%'")
nasa_dates = nasa_cursor.fetchall()

# Retrieve release dates from spotify.db (billboard_100 and daily_50) for dates in 2024
spotify_cursor.execute("SELECT release_date FROM billboard_100 WHERE release_date LIKE '2024-%'")
billboard_dates = spotify_cursor.fetchall()
spotify_cursor.execute("SELECT release_date FROM daily_chart WHERE release_date LIKE '2024-%'")
daily_50_dates = spotify_cursor.fetchall()

# Calculate the top ten recurring dates for billboard_100 in 2024
billboard_counter = Counter([date[0] for date in billboard_dates])
top_ten_billboard = billboard_counter.most_common(10)

# Calculate the top ten recurring dates for daily_50 in 2024
daily_50_counter = Counter([date[0] for date in daily_50_dates])
top_ten_daily_50 = daily_50_counter.most_common(10)

# Insert top ten recurring dates for billboard_100 into the table
combined_cursor.executemany("INSERT INTO top_ten_billboard_dates VALUES (?, ?)", top_ten_billboard)

# Insert top ten recurring dates for daily_50 into the table
combined_cursor.executemany("INSERT INTO top_ten_daily_50_dates VALUES (?, ?)", top_ten_daily_50)
```

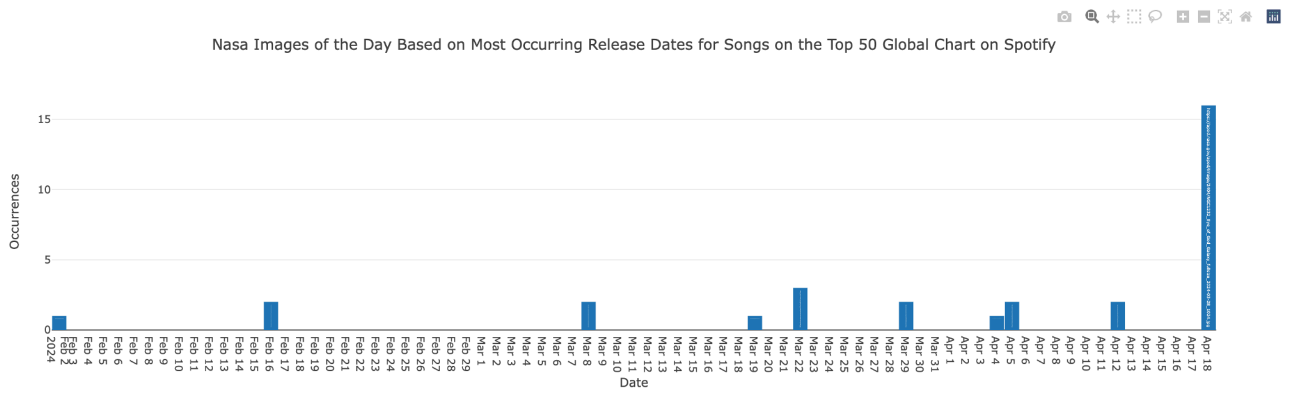


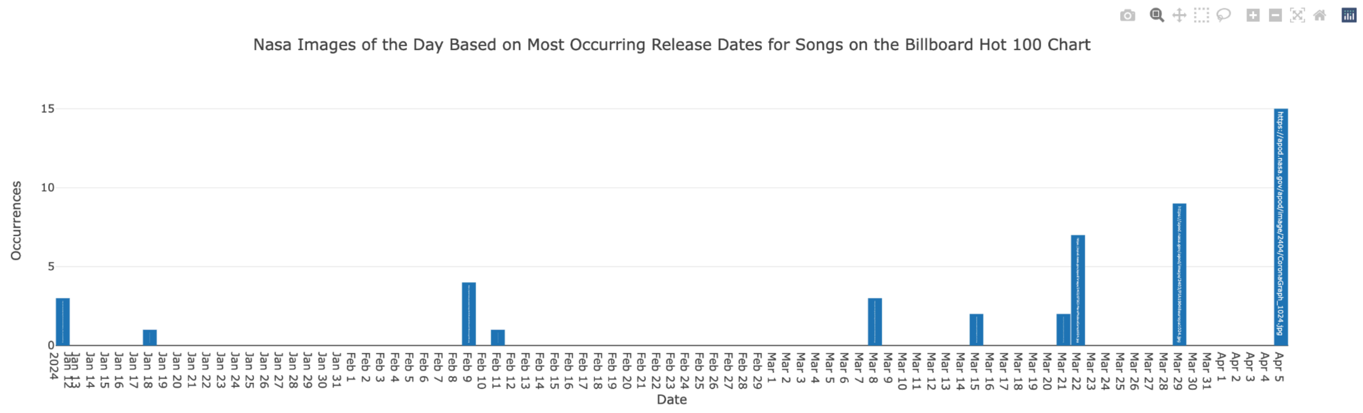
	date	occurrences	url
1	2024-04-12	18	https://apod.nasa.gov/apod/image/2404/...
2	2024-03-22	7	https://apod.nasa.gov/apod/image/2403/STSCI-...
3	2024-04-05	5	https://apod.nasa.gov/apod/image/2404/...
4	2024-02-09	4	https://apod.nasa.gov/apod/image/2402/...
5	2024-03-29	3	https://apod.nasa.gov/apod/image/2403/...
6	2024-03-08	2	https://apod.nasa.gov/apod/image/2403/Tarantula-HST-...
7	2024-03-15	2	https://apod.nasa.gov/apod/image/2403/...
8	2024-01-12	2	https://apod.nasa.gov/apod/image/2401/...
9	2024-03-21	2	https://apod.nasa.gov/apod/image/2403/...
10	2024-01-18	1	https://apod.nasa.gov/apod/image/2401/...



	date	occurrences	url
1	2024-04-18	15	https://apod.nasa.gov/apod/image/2404/...
2	2024-04-12	2	https://apod.nasa.gov/apod/image/2404/...
3	2024-03-22	2	https://apod.nasa.gov/apod/image/2403/STSCI-...
4	2024-04-05	2	https://apod.nasa.gov/apod/image/2404/...
5	2024-03-08	2	https://apod.nasa.gov/apod/image/2403/Tarantula-HST-...
6	2024-02-16	2	https://apod.nasa.gov/apod/image/...
7	2024-03-29	2	https://apod.nasa.gov/apod/image/2403/...
8	2024-03-19	1	https://apod.nasa.gov/apod/image/2403/...
9	2024-02-02	1	https://apod.nasa.gov/apod/image/2402/...
10	2024-04-04	1	https://apod.nasa.gov/apod/image/...

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





6. Instructions for running your code (10 points)

1. Run spotify_data.py 6 times
2. Run nasa_data.py 6 times (this one takes a little longer because we had to add a time delay on the API due to errors)
3. Run data_gathering.py
4. Run data_analysis_billboard.py and then copy url and paste it in browser
5. Close or reset VSCode
6. Run data_analysis_daily50.py and then copy url and paste it in browser

7. Documentation for each function that you wrote. This includes describing the input and output for each function (20 points)

1. **Function Name**: `get_token`
   - **Description**: Obtains an authentication token from Spotify's API using client credentials.
   - **Parameters**: None

  - **Returns**: (str) - A string containing the access token needed to authenticate API requests.

2. **Function Name**: `get_auth_header`
  - **Description**: Constructs an authorization header using a provided token.
  - **Parameters**:
    - `token` (str) - The access token used for authenticating API requests.
  - **Returns**: (dict) - A dictionary representing the authorization header suitable for HTTP requests.

3. **Function Name**: `get_billboard_100`
  - **Description**: Retrieves data from the Billboard 100 playlist on Spotify using an authenticated API call.
  - **Parameters**:
    - `token` (str) - The access token used for authenticating the API request.
  - **Returns**: (dict) - A dictionary containing the JSON response from the Spotify API with details about the Billboard 100 playlist.

4. **Function Name**: `get_daily_chart`
  - **Description**: Retrieves data from Spotify's Global Top 50 daily chart playlist using an authenticated API call.
  - **Parameters**:
    - `token` (str) - The access token used for authenticating the API request.
  - **Returns**: (dict) - A dictionary containing the JSON response from the Spotify API with details about the Global Top 50 daily chart playlist.

5. **Function Name**: `add_position_to_json`
  - **Description**: Enhances a given JSON object (representing a Spotify playlist) by adding a position number to each track.
  - **Parameters**:
    - `json` (dict) - A JSON object representing a playlist, where each track is under a subdictionary keyed by `"tracks"` and `"items"`.
  - **Returns**: (dict) - The modified JSON object with each track now including its position in the playlist.

6. **Function Name**: `setup_billboard_table`
  - **Description**: Establishes a database connection, creates a new table (if it doesn't already exist) specifically for storing Billboard 100 data, and then closes the connection.
  - **Parameters**: None
  - **Returns**: None

7. **Function Name**: `setup_daily_chart_table`

- **Description**: Sets up a database connection, creates a new table (if it doesn't already exist) for storing data related to Spotify's Global Top 50 daily chart, and then closes the connection.
- **Parameters**: None
- **Returns**: None

8. **Function Name**: `insert_25_tracks`
   - **Description**: Inserts up to 25 tracks into the `billboard_100` and `daily_chart` tables in the SQLite database, ensuring no duplicate entries based on track positions. The function prioritizes filling the `billboard_100` table to its capacity of 100 tracks before adding to the `daily_chart`.
   - **Parameters**:
     - `conn` (`sqlite3.Connection`) - The connection object to the SQLite database.
     - `cursor` (`sqlite3.Cursor`) - The cursor object used for executing SQL commands.
   - **Returns**: None

9. **Function Name**: `get_pictures`
   - **Description**: Fetches daily image URLs from a given API for each day starting from January 1, 2024, up to the current date. It logs both successful fetches and errors encountered.
   - **Parameters**: None
   - **Returns**: (dict) - A dictionary where keys are date strings (`'YYYY-MM-DD'`) and values are URLs of images fetched.

10. **Function Name**: `add_position_to_json`
    - **Description**: Enhances a dictionary containing image URLs by adding a position index and reformatting each entry.
    - **Parameters**:
      - `pictures` (dict) - A dictionary where keys are date strings and values are URLs.
    - **Returns**: (dict) - The modified dictionary with each value being another dictionary containing the date, URL, and position of the image.

11. **Function Name**: `set_up_date_table`
    - **Description**: Creates a SQLite database table named `nasa_pictures` if it does not already exist, designed to store NASA picture data including position, date, and URL.
    - **Parameters**: None
    - **Returns**: None

12.**Function Name**: `insert_pictures_25`
    - **Description**: Inserts up to 25 new picture entries into the `nasa_pictures` database table from fetched NASA picture data, ensuring no duplicate entries based on position.
    - **Parameters**:
      - `conn` (`sqlite3.Connection`) - The connection object to the SQLite database.
      - `cursor` (`sqlite3.Cursor`) - The cursor object used for executing SQL commands.

- **Returns**: None

13.**Function Name**: `calculate_top_ten_recurring_dates`
- **Description**: Computes the top ten recurring release dates from two tables in the Spotify database for the year 2024, comparing these with NASA picture dates, and inserts the results into two new tables in a combined database.
- **Parameters**: None
- **Returns**: None

14.**Function Name**: `add_nasa_pictures_to_combined`
- **Description**: Transfers all records from the `nasa_pictures` table in the NASA database to a corresponding table in the combined database.
- **Parameters**: None
- **Returns**: None

15.**Function Name**: `join_tables_BB`
- **Description**: Joins the `top_ten_billboard_dates` and `nasa_pictures` tables in the combined database, creating a new table called `joined_billboard` that includes the release date, occurrences, and the corresponding NASA picture URL.
- **Parameters**: None
- **Returns**: None

16.**Function Name**: `join_tables_DC`
- **Description**: Joins the `top_ten_daily_50_dates` and `nasa_pictures` tables in the combined database, creating a new table called `daily_50_joined` that includes the release date, occurrences, and the corresponding NASA picture URL.
- **Parameters**: None
- **Returns**: None

17. **Function Name**: `bar_graph_daily_50`
- **Description**: Visualizes the occurrences of song releases corresponding to specific dates from the `daily_50_joined` dataset using an interactive bar graph. The graph includes clickable bars that link to NASA images associated with each date.
- **Parameters**: None
- **Returns**: None

18.**Function Name**: `bar_graph_billboard`
- **Description**: Creates an interactive bar graph to display the occurrences of Billboard Hot 100 song releases by date, linking each bar to the corresponding NASA image of the day.
- **Parameters**: None
- **Returns**: None

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
April 26th 2024	Incrementing Databases by 25 items	Stack Overflow/ Chat GPT	Developed an algorithm to increment them
April 29th 2024	Joining two tables in a database	Stack Overflow/ Chat GPT	Made a combined database to join the data
April 29th 2024	Converting database to JSON	Stack Overflow	Exported SQLite to JSON