The goals for your project including what APIs/websites you planned to work with and what data you planned to gather (5 points)

From Genius, we used it to get if a track exists, if it does it adds it to the database. When it is done running it will print out how many tracks are in the database.

From Lyrics You, This API provided the full lyrics text for each song. Because lyrics retrieval frequently fails for certain songs, we implemented: Title normalization, tries to get lyrics and A failure catch system,
These bits of code prevented repeated requests for songs that cannot return lyrics. We will extract the actual lyrics of each song, inputting the artist and track title name.

From Billboard Hot 100, we used web scraping with BeautifulSoup, we extracted the weekly top 100 songs from Billboard.com. We stored Chart position, Artist name, and Song title
Each run inserts up to 25 rows due to project constraints.

From TheAudioDB API we tried to get genre, BPM, mood, and album artwork. While many holiday tracks do not exist in TheAudioDB catalog we still tried to implement the code none the less

2. The goals that were achieved including what APIs/websites you actually worked with and what data you did gather (5 points)

We were able to achieve a complete working system that successfully integrates multiple data sources via API and scraping, cleans and organizes that data in a relational database that performs complex SQL analysis, and generates insightful visualizations. The APIs we worked with are Genius, Lyrics.ovh, Audio DB and Billboard Hot 100(Beautiful Soup). The data we gathered from Genius was our primary source for tracks and artists, Lyrics.ovh was our secondary source for lyrical data. Audio DB was our supplemental source for track metadata (Genre, Mood, BPM). Billboard Hot 100(Beautiful Soup) was our web scraping component for popularity data.

3. The problems that you faced (5 points)

One of our challenges was that KKbox (an api we chose to use) ended up not being available in the United States. we had to choose a new api to work with because of that so we chose the audio DB and BeautifulSoup to generate quantitative data from Billboard Top 100.

APIs continuously failed (timeouts, missing songs): we had to add retries, normalization, and failure caching.

String matching was difficult (different punctuation, featured artists, different spellings across sites): This was most problematic between the Genius APi and BeautifulSoup that used Billboard Top 100 data.

4. The calculations from the data in the database (i.e. a screen shot) (5 points)

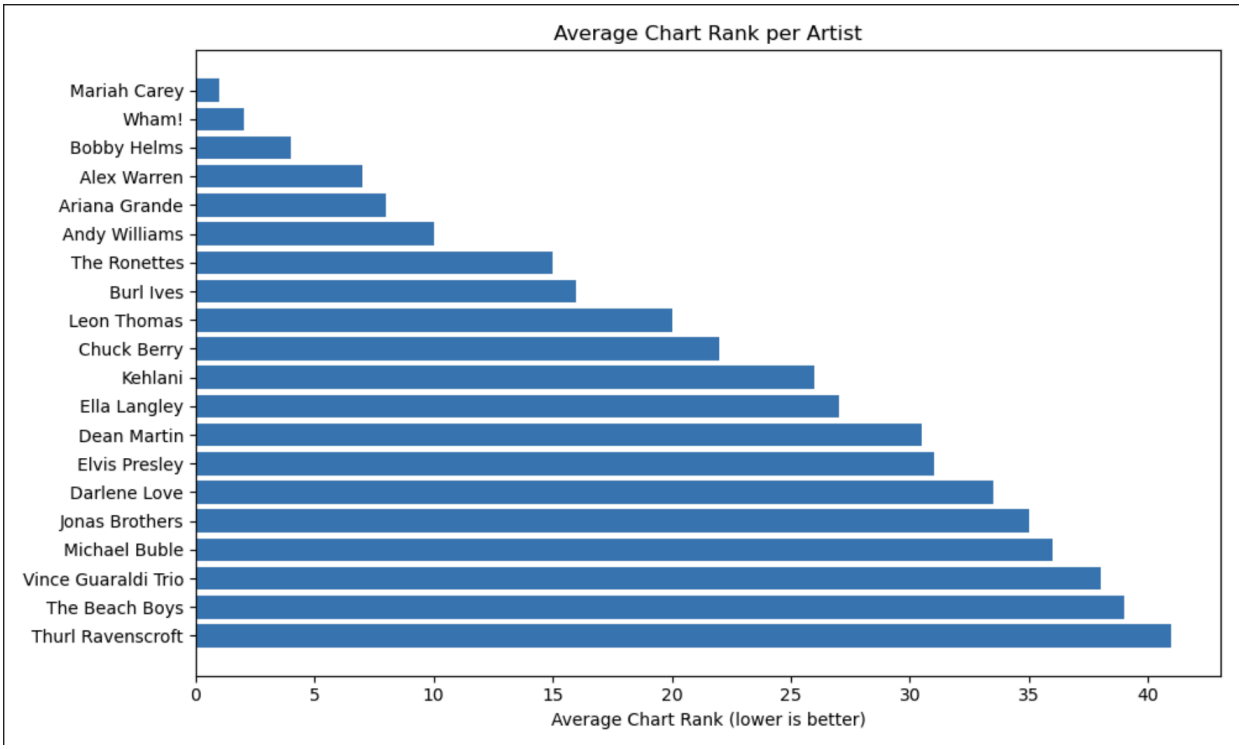
python analyze_visualize.py

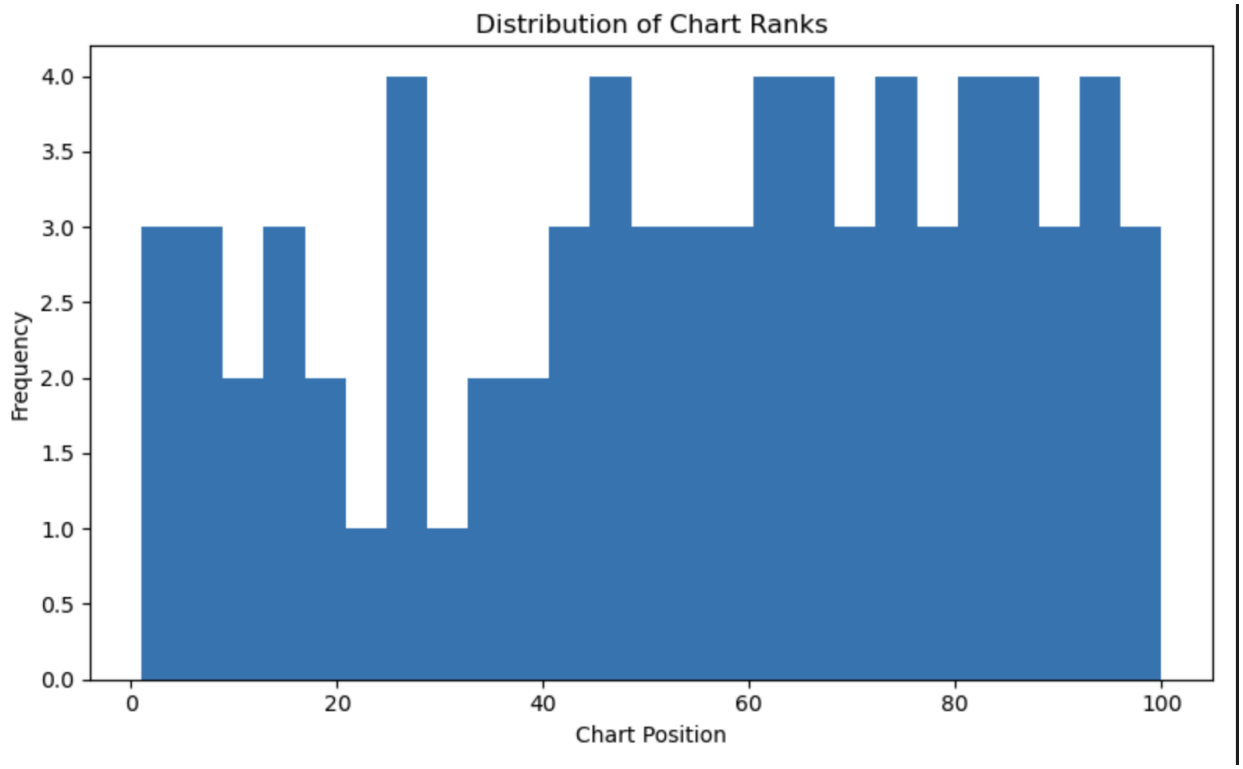
```
=== Average chart rank per artist (first 10) ===
Taylor Swift     avg rank:   12.5
Drake            avg rank:   34.0
Adele            avg rank:   56.0
...

=== Average lyrics length per artist (first 10) ===
Taylor Swift     avg words: 438.0 from 2 song(s)
Drake            avg words: 392.5
...

=== Chart position stats ===
Min position: 1
Max position: 25
Total entries: 200
```

Open DB Browser for SQLite, go to the Execute SQL tab.

```
SELECT artists.name,
       AVG(chart_popularity.chart_position) AS avg_rank
FROM chart_popularity
JOIN tracks ON tracks.id = chart_popularity.track_id
JOIN artists ON artists.id = tracks.artist_id
GROUP BY artists.name
ORDER BY avg_rank ASC;
```

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





6. Instructions for running your code (5 points)

Option 1: Run the Full Pipeline Automatically (Recommended)

1. Run the database setup script:

   python db_setup.py

2. Run the pipeline script:

   python pipeline.py

3. Run the calculation and visualization script:

   python analyze_visualize.py

.Option 2: Run Each Component Manually (Step-by-Step)

This option allows the user to run each API and processing step individually.

1. Run the database setup script:

   python db_setup.py

2. Gather track and artist data from the Genius API:

   python gather_genius.py

3. Gather lyrics data from the Lyrics.ovh API:

   python gather_lyrics.py

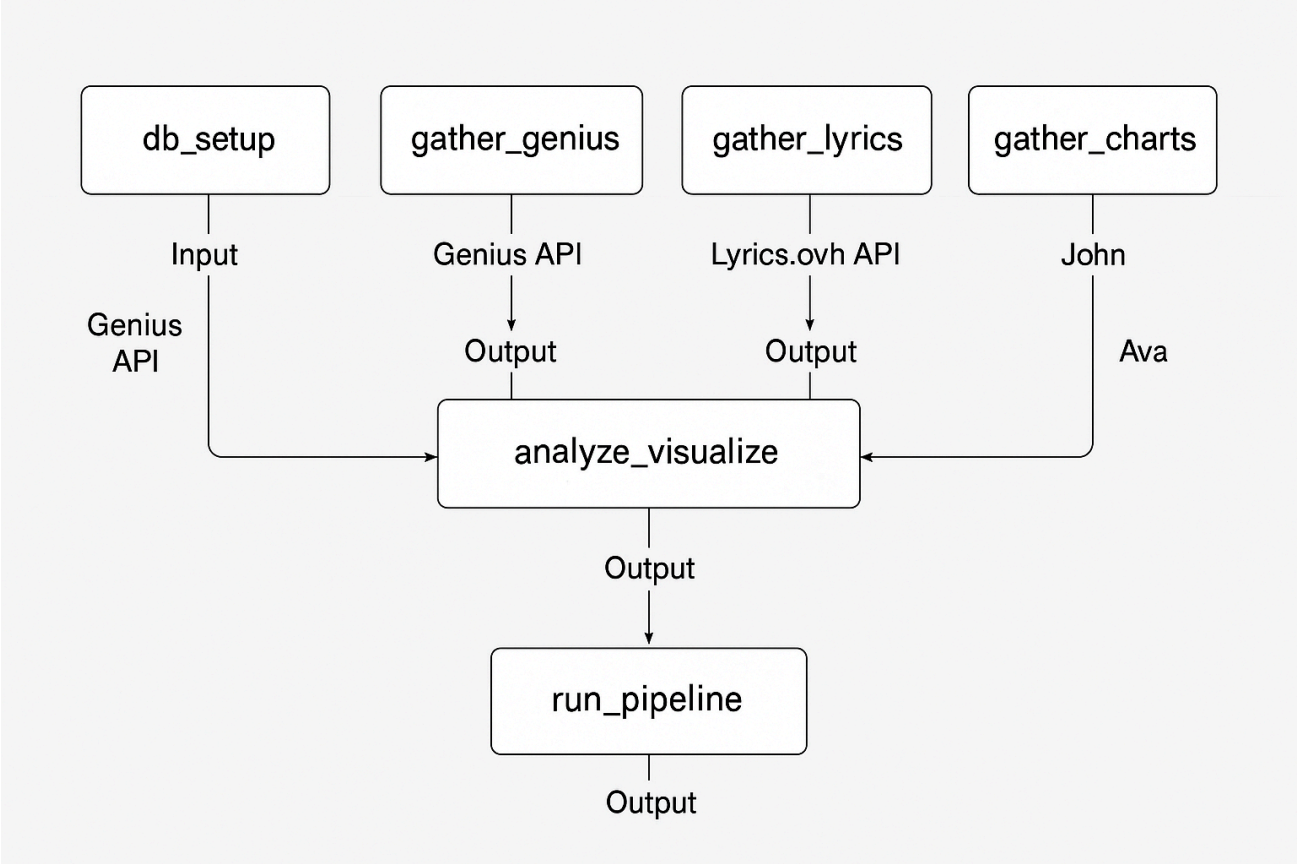
4. Gather chart popularity data from the Billboard Hot 100 website (BeautifulSoup):

   python gather_charts.py

5. Run calculations and generate visualizations:

   python analyze_visualize.py

7. An updated function diagram with the names of each function, the input, and output and who was responsible for that function (10 points)



8. You must also clearly document all resources you used. The documentation should be of the following form (10 points)

Date	Issue Description	Location of Resource	Result
12/9/25	Needed to use an API to gather artist names and song titles to populate the database with music metadata	Genius API Documentation (https://docs.genius.com/)	Yes
12/9/25	Needed to retrieve full song lyrics in order to calculate lyrical length and perform text-based analysis	Lyrics.ovh API Documentation (https://lyricsovh.docs.apiary.io/)	Yes

12/10/25	Needed to gather chart ranking data for songs and match them to existing tracks in the database	Billboard Hot 100 website scraped using BeautifulSoup (https://www.billboard.com/charts/hot-100)	Yes
12/10/25	Needed guidance on SQLite database normalization to avoid duplicate string data and design proper table relationships	ChatGPT (OpenAI)	Yes
12/10/25	Needed help debugging Python errors related to SQL joins, UNIQUE constraints, and batching data inserts	ChatGPT (OpenAI)	Yes
12/11/25	Needed reference examples for Matplotlib visualizations and customization of chart colors	Matplotlib Documentation (https://matplotlib.org/stable/contents.html)	Yes
12/11/25	Needed clarification on Python error messages and standard debugging patterns	Stack Overflow (https://stackoverflow.com/)	