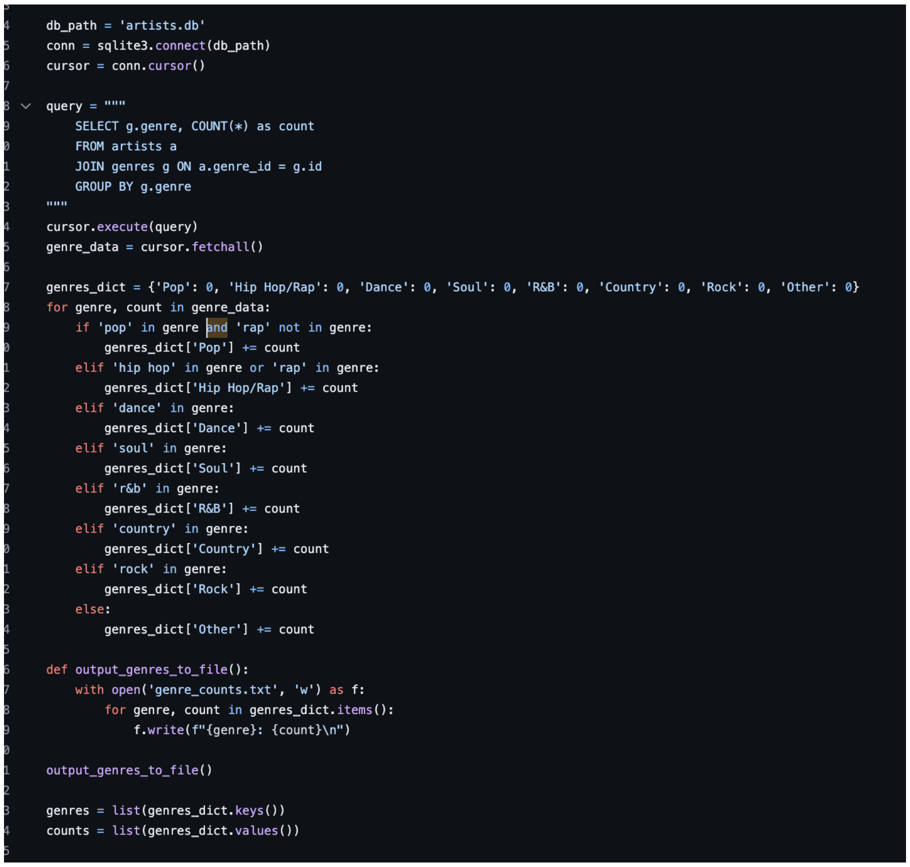
SI 206 Final Report
The Music Trio

1. In this project, our goal was to create a database in SQL with three different tables each containing data from an API and we were going to compare that data to each other. We originally planned to use the Spotify API to gather the top 100 artists on Spotify, putting the artist id, artist name and spotify id in one SQL table. We then, also using the Spotify API, gathered the top 5 tracks from each of the top 100 artists on Spotify and put that into a table called Tracks.Then, using the Shazam API we planned to gather the number of times each of those tracks was shazamed and insert the data into the tracks table to compare the songs to how many times they had been shazamed. Using the Shazam counts we planned to compare those top 500 songs to each other within an artist, compare artists to other artists within a country, and compare those countries to each other. Lastly we planned to make one more chart using a billboard website and use beautiful soup to get the top songs on the billboard chart and compare the spotify top songs to the billboard chart placements.

2. What we actually ended up doing was still using the Spotify API to gather the top 100 artists (we got the artists id, artist name and spotify id in one SQL table called artists) and each artist's top 2 songs each (we got the artist name, track id, and track name and put it into a SQL table called tracks). Where we had to pivot was we decided to use the Billboard API instead to find if any of the 200 tracks from each Spotify were currently on the Billboard hot 100 list and if so, the number of weeks they were charted for. We ensured that duplicate string data wasn't included

3. The biggest problem that we faced was that after writing all of the code to get the shazam count for each track in our SQL table using the Shazam API was that there was an issue with the API being able to access data from Shazam and it kept giving us the same number of Shazams for every song. We realized that all of the URL and API key provided by the Shazam API were invalid and thus could not actually access the data for each track. An issue we had running the billboard data was that it would not return a valid number for songs with features, so we had to implement a remove_features function that would get rid of the features when calling the billboard api. An example is "IDGAF (feat. Yeat)" by Drake. If the song was uploaded with the feature, then the number of weeks on the hot 100 would return zero. But, if we uploaded just "IDGAF" to the api, then we would collect the correct number of weeks (8).

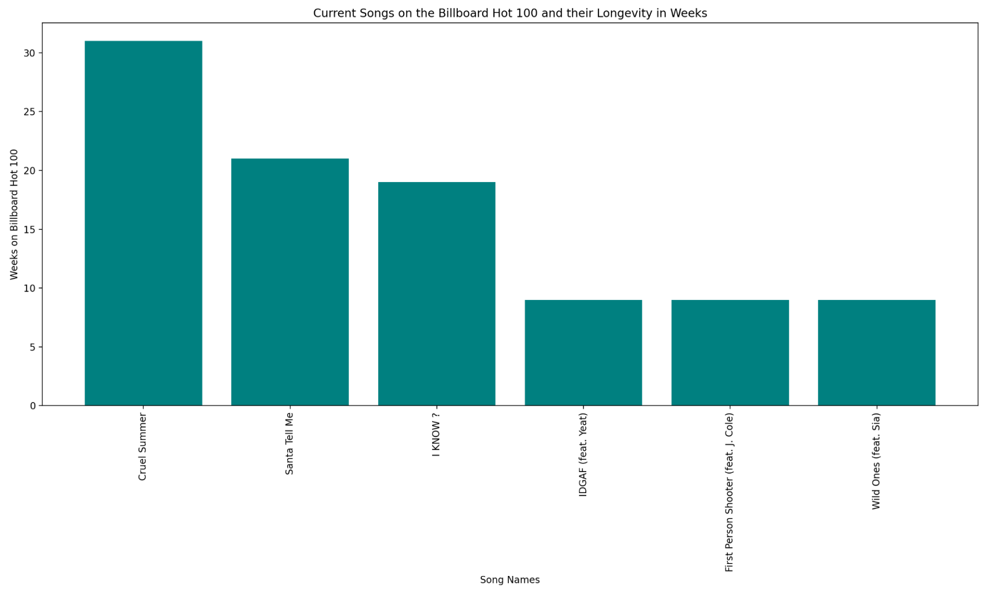
4. Calculations

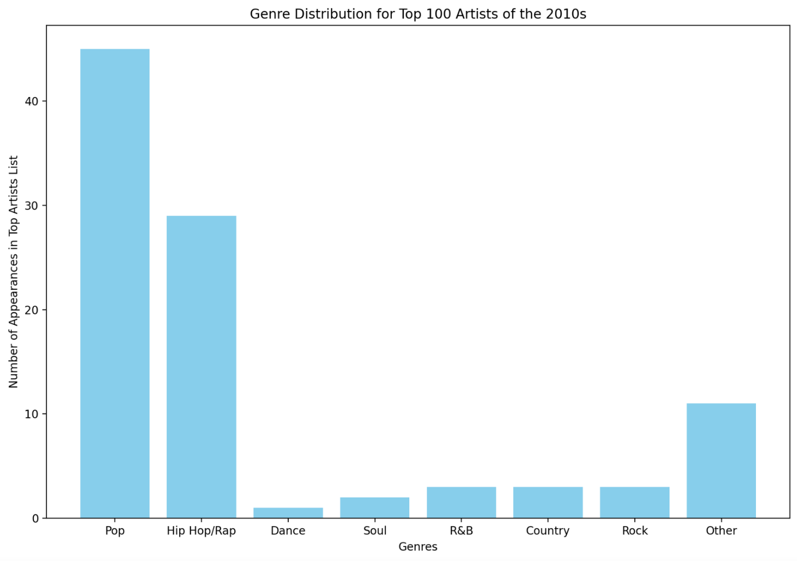


```python
db_path = 'artists.db'
conn = sqlite3.connect(db_path)
cursor = conn.cursor()

query = """
    SELECT g.genre, COUNT(*) as count
    FROM artists a
    JOIN genres g ON a.genre_id = g.id
    GROUP BY g.genre
"""
cursor.execute(query)
genre_data = cursor.fetchall()

genres_dict = {'Pop': 0, 'Hip Hop/Rap': 0, 'Dance': 0, 'Soul': 0, 'R&B': 0, 'Country': 0, 'Rock': 0, 'Other': 0}
for genre, count in genre_data:
    if 'pop' in genre and 'rap' not in genre:
        genres_dict['Pop'] += count
    elif 'hip hop' in genre or 'rap' in genre:
        genres_dict['Hip Hop/Rap'] += count
    elif 'dance' in genre:
        genres_dict['Dance'] += count
    elif 'soul' in genre:
        genres_dict['Soul'] += count
    elif 'r&b' in genre:
        genres_dict['R&B'] += count
    elif 'country' in genre:
        genres_dict['Country'] += count
    elif 'rock' in genre:
        genres_dict['Rock'] += count
    else:
        genres_dict['Other'] += count

def output_genres_to_file():
    with open('genre_counts.txt', 'w') as f:
        for genre, count in genres_dict.items():
            f.write(f"{genre}: {count}\n")

output_genres_to_file()

genres = list(genres_dict.keys())
counts = list(genres_dict.values())
```

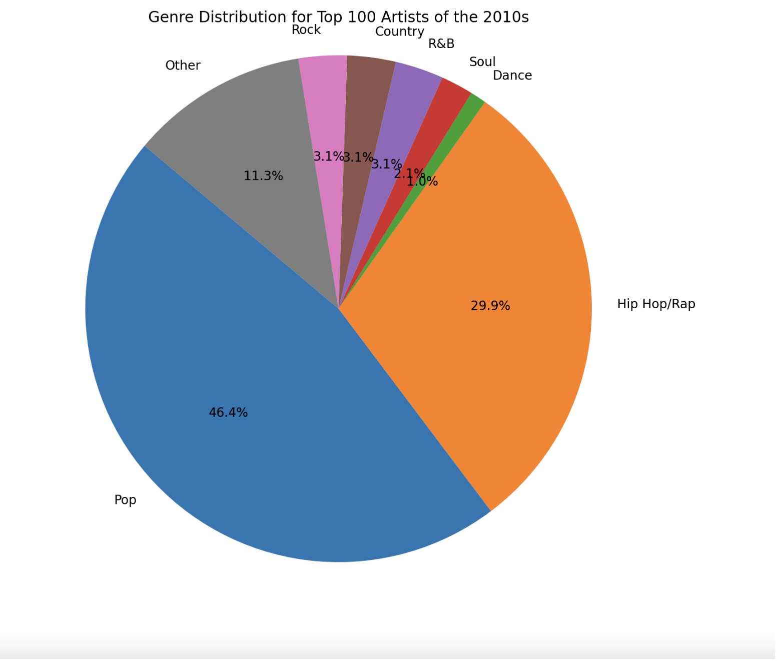
5. Visualizations

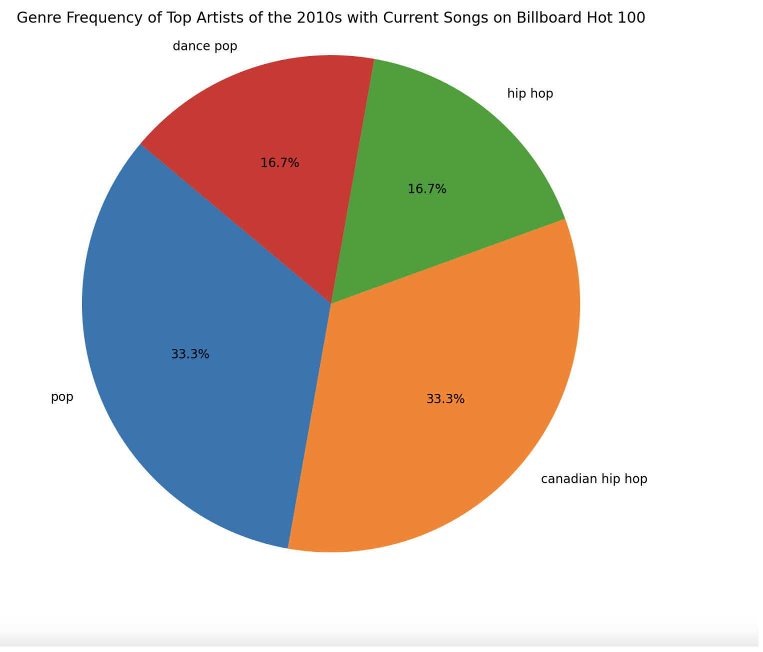
SI 206 Final Report
The Music Trio





SI 206 Final Report
The Music Trio





6. **Instructions for running code:** The way that you are supposed to run our code is first going into the load_data file and running it, then open the database artists.db that it generates in SQL so you can see the data filling into the table. Run that code a total of 4 times. Then, go into the top songs file and run that code 9 times. In SQL if you toggle to the tracks table that it generates after the first time running, you will see all the data for the tracks. Then, go into the billboard file and run that function 4 times to make sure that all of the tracks go into the billboard SQL table which you can also toggle within the artist database.

7. Documentation:

Load Data File:

Load_data function:

This fetches data from the URL 'https://top40weekly.com/top-100-artists-of-the-10s/' and extracts the HTML content. Using BeautifulSoup, it parses the HTML to extract artist names from a table on the webpage. It establishes a connection to the SQL database named 'artists.db' and creates a table named 'artists' if it doesn't exist, with columns for artist ID, name, Spotify ID, and genre. It queries the number of artists already in the database and retrieves the next batch of 25 artist names. For each artist in the batch, it queries the Spotify API to get information such as Spotify ID and genre.It inserts the artist information into the 'artists' table, considering uniqueness constraints. Finally, it prints the contents of the 'artists' table in the database.

Top Songs File:

Get_artists_without_tracks function: This takes a SQL database connection (conn) as input. It queries the 'artists' table in the database to retrieve the ID, name, and Spotify ID of artists who do not have corresponding entries in the 'tracks' table. Specifically, it looks for artists whose IDs are not present in the 'tracks' table, ensuring that only artists without associated tracks are included. The function then returns the result, which is a list of tuples containing the relevant artist information for up to 12 artists.

Add_top_tracks_to_db:

This function takes in the list of artists that is in the SQL database and gets the top two songs from each artist. It iterates through creating unique track IDs based on the artist and track index, and inserts the relevant information (artist ID, track ID, and track name) into the 'tracks' table of the database

Main function:

The `main` function is responsible for initializing the 'tracks' table in the database if it doesn't exist. It sets up the schema with columns for artist ID, track ID, and track name. Afterward, it establishes a connection to the 'artists.db' database, creates a cursor to execute SQL commands, and commits the changes. Then, it initializes the Spotify API using the provided client credentials, and calls the `add_top_tracks_to_db` function to populate the 'tracks' table with top tracks information retrieved from the Spotify API.

Billboard file:

Remove _features function:
This takes the `track_names` in the SQL database as input. The function checks if the track name contains specific patterns like ' (feat.', ' (with', or ' [with', which usually indicate featured artists. If any of these patterns are found, the function removes the portion of the track name starting from the identified pattern and returns the cleaned track name. If none of these patterns are present, the original track name is returned unchanged.

Get_weeks_on_chart function:
It takes a list of `song_titles` as input. The function uses the Billboard API to get the Billboard Hot 100 chart data for the current week and retrieves the number of weeks each track has spent on the chart. It uses the `remove_features` function to clean the song titles before comparing them with the provided list. The function returns a dictionary `song_weeks` mapping original track titles to the number of weeks they have spent on the Hot 100 chart. It creates a table in SQL to store an offset value, retrieves the current offset, queries a batch of tracks from the database, extracts track names, and uses the `get_weeks_on_chart` function to get the weeks on the Hot 100 chart for each track. The script then updates the database by adding a new column (`weeks_on_hot_100`) to the `tracks` table, and updates the weeks on the chart for each corresponding track. Finally, the offset is updated in the `update_offset` table, and the script prints the new offset.

8.

Date	Issue Description	Location of Resource	Result
December 8th-10th	Our Shazam Api was not allowing us to get the shazam count for each song. Initially we were getting null	Chat Gpt	The resource was not able to help and suggested that the api was likely too young, so we moved onto the

	values, which was really far from the desired result. Then, we got the count for one the first song, but we were getting the same count for every song in our database. We tried to figure this out ourselves and we couldn’t. We were concerned with the api’s capabilities. So, we consulted chat GPT to help identify the issues we were facing.		billboard api.