

206 Final Project Report

Remi Goldfarb, Caleigh Crossman and Ava Webster

REPO: <https://github.com/caleighc/Music-APIs>

1. The goals for your project (10 points)

The goal for our project was to grab data on the top 100 songs on Spotify, Apple Music, and Napster. We then wanted to use the data about the songs to make comparisons across the 3 platforms (Spotify, Apple Music, and Napster).

2. The goals that were achieved (10 points)

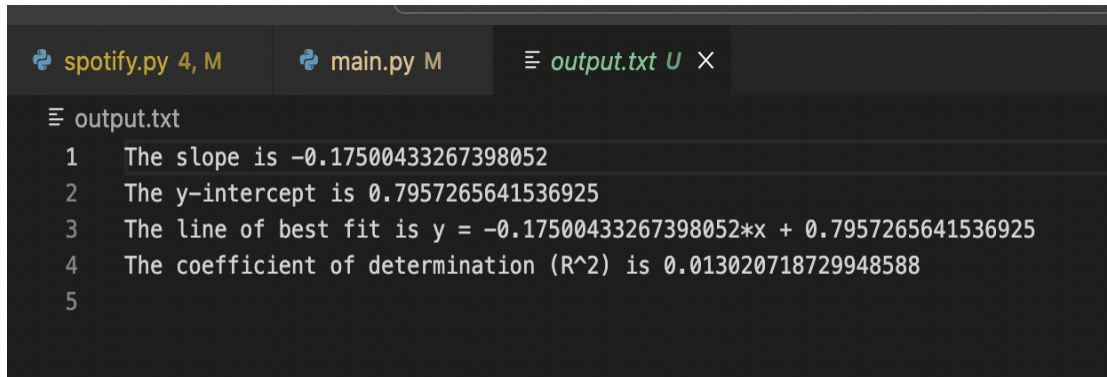
Initially, we were going to look at the top 100 songs on Spotify, Apple Music, and Napster, but we came across difficulties using the Apple Music and Napster APIs, so instead we used APIs from TheAudioDB and Genius, as well as Spotify, to look at top tracks from both artists and music charts. With TheAudioDB, I was able to look at the most popular songs for 10 different top artists. I was able to compare the popularity of artists' by calculating the average score that users give to their songs from their top 10 tracks. Additionally, I was able to see what genres were most popular and had the most songs in them. By analyzing/comparing the two charts, I was able to draw conclusions about what music types and artists are most popular. For example, I found that pop was the most popular genre but some artists whose music was in the pop genre had some of the lower averages. For the Genius API, I was able to find the average amount of words in the titles of the songs from four artists. I was then able to assign the average number of words in the artist's songs to the artist id number.

3. The problems that you faced (10 points)

A problem we faced was not being able to connect the data between our 3 APIs. Originally we had decided to use the Spotify API, Apple Music API, and Napster API. However, the Apple Music API cost \$99 to use and the Napster API didn't have the data we wanted. Therefore, we switched those APIs to the Genius API and TheAudioDB API. In the end, we were still not able to find exact data that was similar across the 3 APIs but we tried to keep our data cohesive by each looking at data on top tracks that were available on our APIs. Initially, I was going to look at the top 10 tracks by the top 10 artists (based on the Billboard Top Artists Chart), but TheAudioDB API did not have data on 10 songs for each of the top 10 artists, so I ended up looking at data for 10 artists within the top 100, but not exclusively the top 10 on the chart. Ultimately, I think this reflects better data because the Billboard charts are updated weekly, so I ended up using artists that are popular in general and not just popular for a short period of time. One problem I faced with the Genius API was that when I was trying to print my data I would constantly get a message in the console saying "request timeout=5" and my data would not all print. This was a huge issue because I needed all of my data to print in order for it to be able to populate in the table I made.

4. Your file that contains the calculations from the data in the database (10 points)

The file that contains the calculations from the data for Spotify is in the database is output.txt. The coefficient of determination is 0.003 which means that there is very little to no correction and the linear model does not fit the data. The slope of -0.079 indicates that there is a negative relationship between the danceability and energy scores of a song. Calculations will vary based on the number of songs in the database when the visualization function is run.



The screenshot shows a code editor with three tabs: 'spotify.py 4, M', 'main.py M', and 'output.txt U X'. The 'output.txt' tab is active, displaying the following text:

```
output.txt
1 The slope is -0.17500433267398052
2 The y-intercept is 0.7957265641536925
3 The line of best fit is y = -0.17500433267398052*x + 0.7957265641536925
4 The coefficient of determination (R^2) is 0.013020718729948588
5
```

The calculations from the data gathered from TheAudioDB are in the calculations1.csv and calculations2.csv files. Users are able to assign a score to a song, so the calculations1.csv file shows the average score of the top 10 songs from 10 top artists. Additionally, I did a second calculation for a second visualization for extra credit where I calculated the genres of the 100 songs from my data and got the count for how many songs were in each genre. This is shown in the calculations2.csv file.

```

calculations1.csv
1 Artist Name,Average Score
2 Ariana Grande,9.63
3 Drake,10.0
4 Ed Sheeran,9.72
5 Justin Bieber,10.0
6 Kendrick Lamar,10.0
7 Metallica,8.99
8 Michael Jackson,8.95
9 Rihanna,8.62
10 Taylor Swift,8.64
11 The Weeknd,9.96
12

```

```

calculations2.csv
1 Genre,Number of Songs in Genre
2 Country,8
3 Hip-Hop,20
4 Pop,52
5 R&B,10
6 Thrash Metal,10
7

```

```

new_file_1.csv
1 Artist ID,Average Amount of Words in Song Title
2 41579,3.32
3 15740,5.08
4 639521,6.96
5 1177,3.96
6

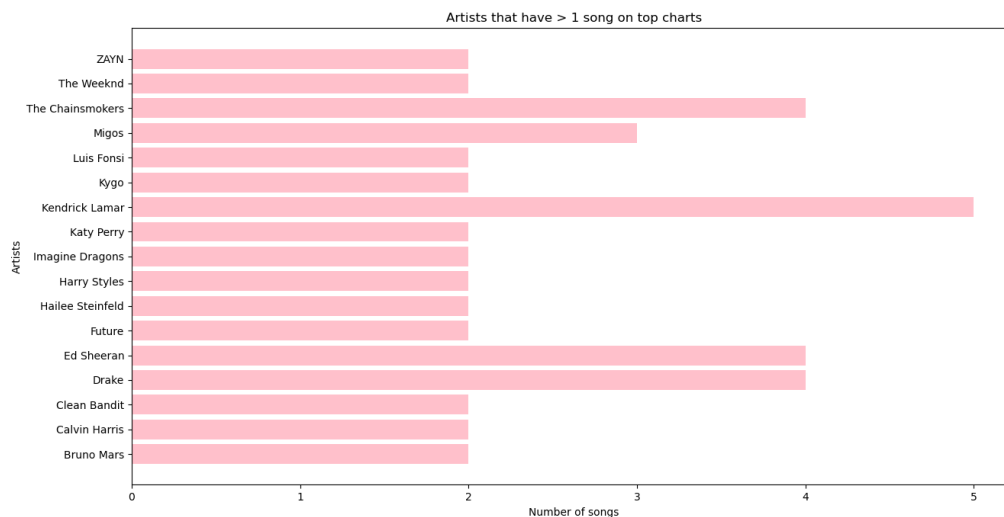
```

This file is for the Genius API. This file includes artist id's and the average amount of words in each of their song title's.

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

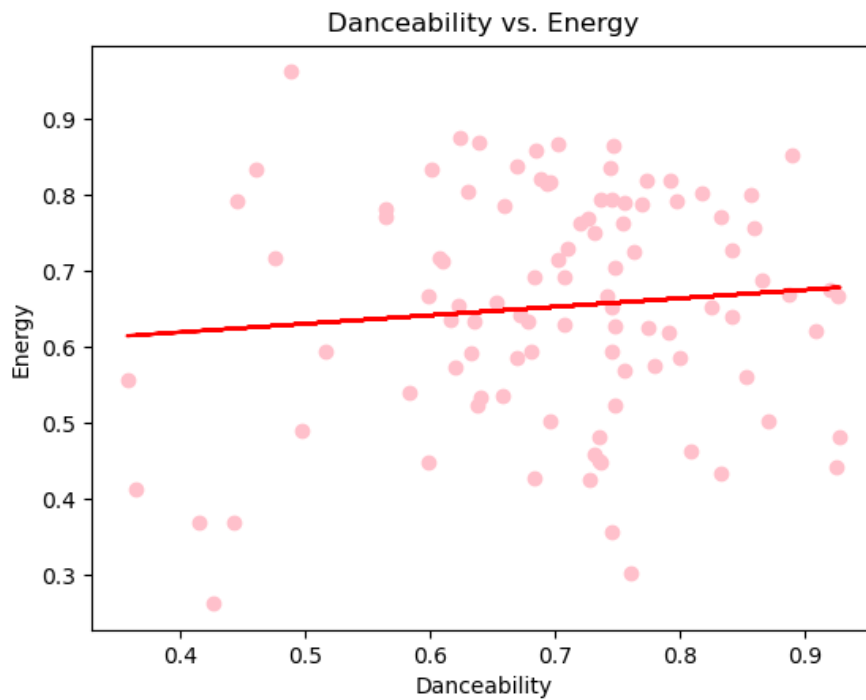
This bar chart shows the artists who have more than one song on the top charts on the y-axis and the number of songs they have on the top charts on the x-axis. I made this visualization in order to see if artists had multiple songs on the top charts and how many songs they had. I was curious to see if popular artists (artists on the top charts) typically had one top hit or multiple popular songs. I was able to get the correct data from my tables using the query

```
SELECT artists.artist, COUNT(*)  
FROM songs  
JOIN artists ON artists.artist_id = songs.artist_id  
GROUP BY artist  
HAVING COUNT(*) > 1
```

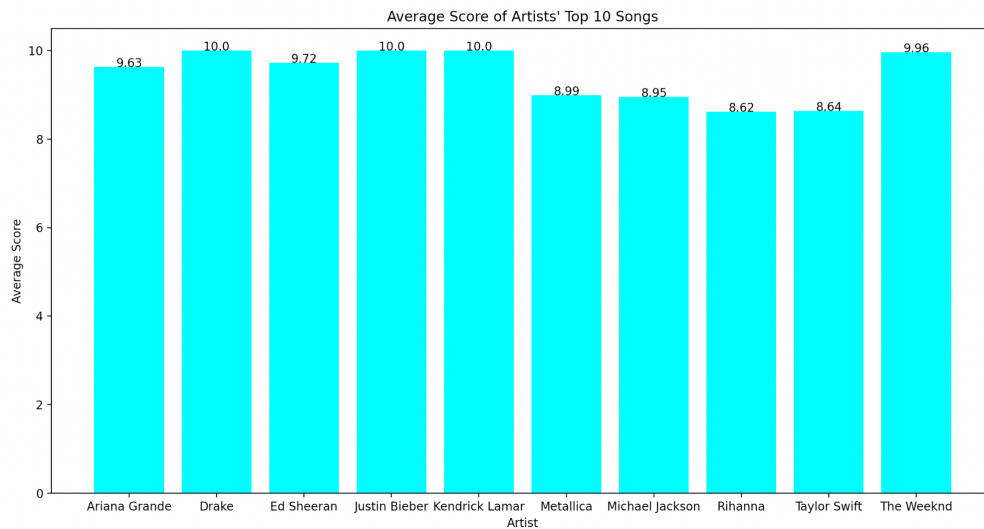


This scatter plot shows the energy score on the y-axis and the danceability score on the x-axis for each of the songs in my table. I made this scatter plot to analyze the relationship between danceability and energy for a song. In addition, I calculated and graphed the line of best fit for this relationship. As seen, there is not a linear relationship between danceability and energy. I was able to get the data from my table to create the visualization below using the query

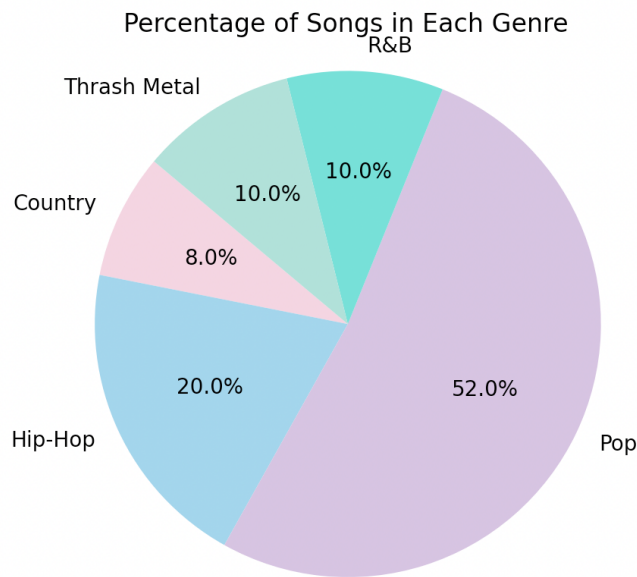
```
SELECT danceability,energy FROM songs
```



The first visualization I created was a bar graph that displays the average score for the top 10 tracks for each of the 10 artists, which represents not only an artists' average rating, but also shows how their scores compare to each other.



The second visualization I created is a pie chart that represents the percentage of the 100 songs that belong in each genre.



6. Instructions for running your code (10 points)

- Run main.py 4 times and then 2 spotify visualizations will show up behind one another
- Then in main.py in main() function comment out run_spotify(cur,conn), uncomment run_audio(cur,conn), and run main.py 4 times to get all 100 rows of data in (25 rows inserted each time code runs). By the end of the 4 times, both completed visualizations should pop up

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

Name	Function	Description	Input	Output
Caleigh	set_up_db(db_name)	This function in main takes the name of the database and creates it	Name of the database (music)	returns cur, conn
Caleigh	run_spotify(cur,conn)	This function runs the functions in spotify.py	cur,conn	None
Caleigh	main()	Main function that calls set_up_db, run_spotify, run_audio, and run_genius	None	None
Caleigh	make_requests(playlist_id)	Takes in a spotify playlist ID and makes a request to	Playlist_id (the list of playlist ids are in	playlist which is a dictionary

		the API to return the playlist JSON data	run_spotify)	that contains the data about the tracks
Caleigh	create_artists_table(cur,conn)	Creates the Artists table in the music database	cur,conn	None
Caleigh	create_songs_table(cur,conn)	Creates the Songs table in the music database	cur,conn	None
Caleigh	add_artists_id(data, cur,conn)	Adds the artist names and artist_ids to the Artists table in the database	data,cur,conn (data is the dictionary that is returned from make_requests)	None
Caleigh	add_songs(data,cur, conn)	Adds each songs and song data to the Songs table in the database	data,cur,conn (data is the dictionary of song data that is returned from make_requests)	None
Caleigh	danceability_vs_energy_plot(cur,conn)	Uses a query to the database to create a scatter plot of the danceability vs energy scores of each song in the Songs table	cur,conn	Visualization
Caleigh	artists_visualization(cur,conn)	Uses a query to the database to create a horizontal bar plot of the number of songs each artists has on the top charts	cur,conn	Visualization
Ava	request_data(artist_ids)	Requests data from API and parses through JSON object using regex to get the desired data	artist_ids - list of artist ids to use in url	None
Ava	create_table(cur, conn)	Creates the SongsAudiodb and ArtistsAudiodb tables	cur, conn	None
Ava	insert_data(cur, conn)	Inserts the data into the SongsAudiodb table (25 rows each time program runs) and into the ArtistsAudiodb table (only has 10 items)	cur, conn	None

Ava	<code>calculate1(cur, conn)</code>	Does the first calculation which calculates the average score of each artist based on the scores for each of their songs	<code>cur, conn</code>	Returns result, which is a list of tuples with the artist and their average score
Ava	<code>calculate2(cur, conn)</code>	Does the second calculation which calculates the number of songs in each genre	<code>cur, conn</code>	Returns result, which is a list of tuples with the genre and number of songs in that genre
Ava	<code>write_to_csv1(result)</code>	Writes the results of the first calculation to a CSV file	result (from <code>calculate1</code>)	None
Ava	<code>write_to_csv2(result)</code>	Writes the results of the second calculation to a CSV file	result (from <code>calculate2</code>)	None
Ava	<code>make_chart1(result)</code>	Makes first visualization (bar graph) of first calculation	Result (from <code>calculate1</code>)	visualization
Ava	<code>addLabels(x, y)</code>	Adds labels to bar graph to be used to show average score in first visualization	<code>x, y</code> - <code>x</code> and <code>y</code> axis used in graph	None
Ava	<code>make_chart2()</code>	Makes second visualization (pie chart) of second calculation	None	visualization
Ava	<code>run_audio</code>	In <code>main.py</code> , this function runs all of the above functions from <code>my audio.py</code> file	<code>cur.conn</code>	None (visualizations/r results from functions within this function are output)
Remi	<code>run_genius</code>	In <code>main.py</code> , this function runs all of the above functions from <code>my genius.py</code> file	<code>cur, conn</code>	None (visualizations/r results from functions within this function are output)
Remi	<code>genius.new_tables(cur, conn)</code>	This function creates my two tables.	<code>Cur, conn</code>	none

Remi	<pre>input_data(cur, conn)</pre>	Adds data to my tables in the database.	Cur, conn	none
remi	<pre>add_data_6(cur, conn) add_data_7 add_data_8</pre>	Adds data to GeniusArtist table.	Cur, conn	none
remi	<pre>calculation</pre>	Calculates average amount of words in each artists' songs.	Cur, conn	Returns result, which is a dictionary of with the artist id and average number of words in each artists songs.

8. You must also clearly document all resources you used. The documentation should be of the following form (20 points)

Name	Date	Issue Description	Location of Resource	Result (did it solve the issue?)
Caleigh	11/29/2022	Having trouble calling functions in main to create table in database	Homework 6 & 7	Yes, I was able to figure out how to do it properly
Caleigh	12/4/2022	Was having trouble using matplotlib	Discussion 12	Yes, I was able to see how to properly use matplotlib to create visualizations
Caleigh	12/4/2022	Client credentials flow	https://github.com/spotify/web-api-auth-examples/blob/master/client_credentials/app.js	Yes, I was able to figure out the flow for the client credentials
Caleigh	12/9/2022	Calculations	https://numpy.org/	Yes, I was able to import numpy and use it to help me make calculations

Caleigh	12/3/2022	Horizontal bar plot and scale on x and y axis	https://matplotlib.org/stable/api/pyplot_summary.html	Yes, I was able to figure out how to make the plot and change the scale
Ava	12/08/2022	Accessing data in the json object was not easy to do via indexing; found best solution was regex	Regex summary/glossary on Runestone	Yes, I was able to use regex expressions to extract the data i wanted from the json object
Ava	12/08/2022	Formatting the API request	https://www.theaudio db.com/api_guide.php	Yes, this webpage for my API provided documentation get requests and how to request specific data
Ava	12/08/2022	Couldn't find artist ID to use in url request	https://musicbrainz.org/doc/MusicBrainz_API	Yes, i was able to search this website to get the artist id that would work in the url request
Ava	12/11/2022	Writing calculations to csv file	https://tutorial.eyehunts.com/python/python-file-modes-open-write-append-r-r-w-w-x-etc/ https://www.pythontutorial.net/python-basics/python-write-csv-file/	Yes, using these resources I was able to write my calculations to a CSV file
Ava	12/10/2022	Adding labels to bar graph	https://www.geeksforgeeks.org/adding-value-labels-on-a-matplotlib-bar-chart/	Yes, I was able to create an addlabels function to add labels to my bar graph
Remi	12/10/2022	I could not figure out how to print information about an artists' songs.	https://lyricsgenius.readthedocs.io/en/master/	Yes, I was able to learn that I could use <code>genius.search_artist</code> and <code>artist.save_lyrics()</code> in order to save information about the artists and songs I was finding so I could reference them later.
Remi	12/11/2022	Figuring out how to use my access client	https://rapidapi.com/Clavier/api/genius-s	Yes, this resource showed me how to use

		token and API key.	ong-lyrics1/	my access token and API key correctly.
Remi	12/15/2022	Limiting how many songs I would access per artist.	https://medium.com/geekculture/easily-obtain-song-lyrical-information-in-python-5b2c85d4f589	Yes, this resource provided me with this layout <code>artist = genius.search_artist('Par amore', max_songs=5, sort='title')</code> <code>print(artist.songs)</code>