Daniel Kates
Aaron Benyamini
SI 206 Final Project
GITLINK:

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

The original goal of this project was to collect the top 100 songs from 2022 through Spotify and Apple Music APIs and store the title, artist, genre and total number of listeners. We wanted to create a table for each platform, giving the breakdown of total number of listeners per genre, as well as number of songs from each genre in the top 100 by percentage for each category. The goal of this project was to see what are the similarities and differences between different music platforms. We wanted to gather the top 100 songs from two different platforms and compare the lists with different factors like genre.

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

The goal we achieved was very similar to our original goal except we had to shift platforms and change the visualizations. Apple Music API did not work for us and tried to charge us, so we switched to Genius API. Also, we realized that we could not access genres via the top 100 songs so we changed our visualizations to repeated songs in both top 100's and the most listened to artists across both lists. We ended up being able to use Spotify API through "spotipy" which was a great success. We were able to pull the artist, title, and the Spotify popularity of that song which comes from a number of different data points. For Genius, we were able to get the top 100 songs as well including artist and title, and from that list create a second table that used a foreign key to show the top 10 artists, as in the artists with the most songs from the top 100 list.

3. The problems that you faced (10 points)

The biggest problem we faced was finding a second music platform that had a free API. We built out the code from Apple Music and then realized we were about to be charged $100. Then, we switched to soundcloud in which our terminal was having issues downloading the soundcloud extension. We tried many different approaches but it never worked out. We then looked into Amazon Music, Pandora, and other music platforms ultimately landing on Genius. Genius had a top 100 list and an API so we were able to execute what we wanted.

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

common_song_count.txt

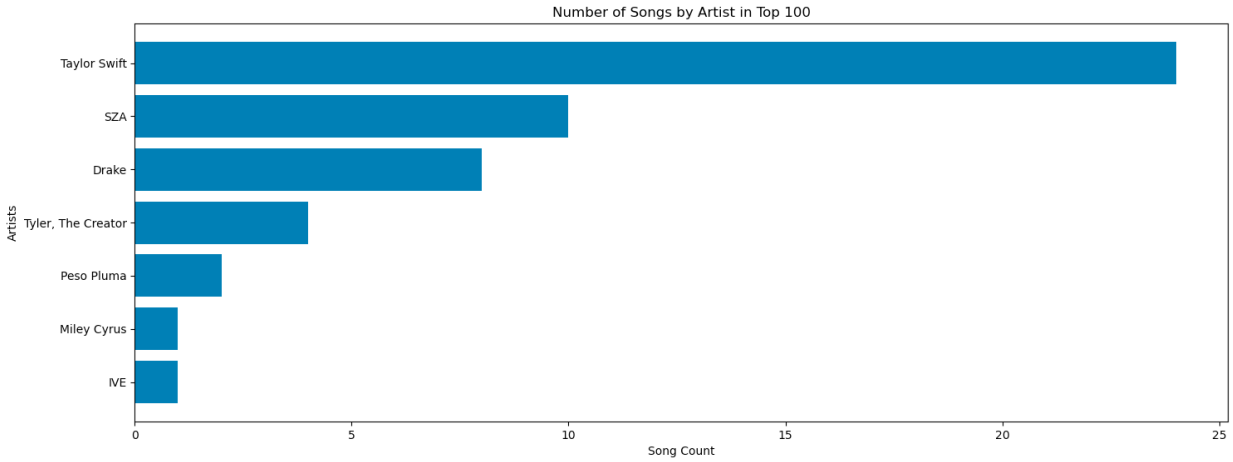
```
5 songs are in both top 100 tables.
```

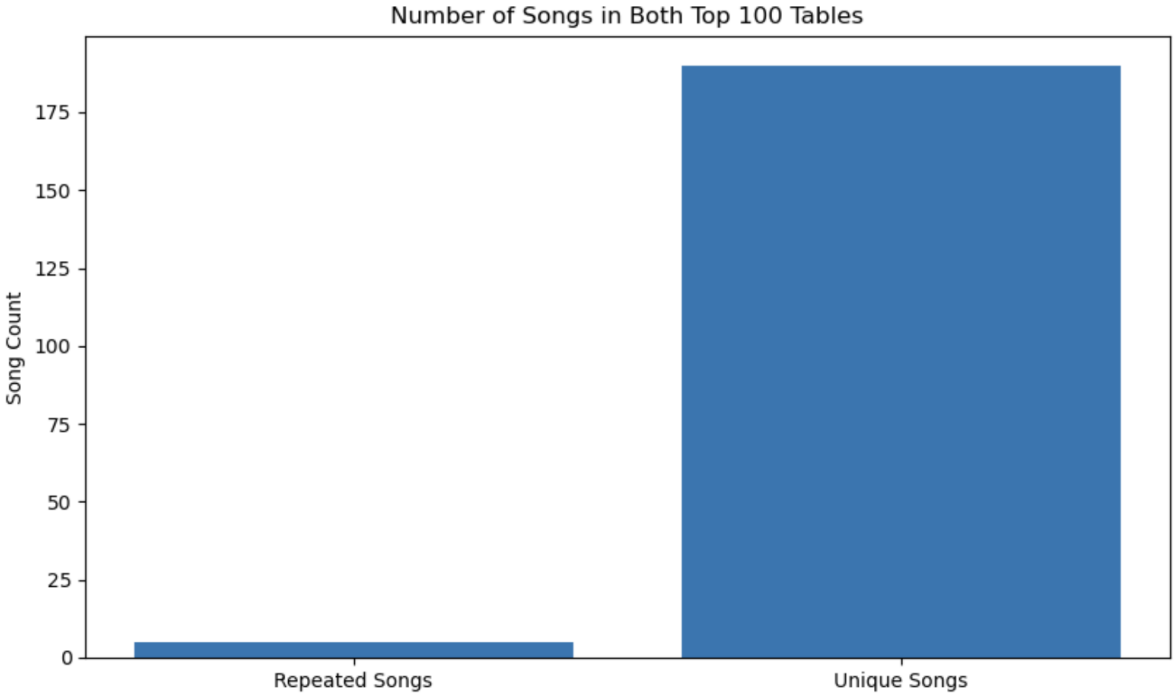
artist_song_count.txt

```
Taylor Swift: 32
SZA: 10
Drake: 8
The Weeknd: 6
Tyler, The Creator: 2
Peso Pluma: 2
Miley Cyrus: 1
IVE: 1

```

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





6. Instructions for running your code (10 points)
    a. Click on the *genius_api.py* file. Run it exactly **four** times. This will generate the two tables adding no more than 25 rows at a time.
    b. Next, click on the *spotify_api.py* file. Run it exactly **four** times. This will generate the two tables adding no more than 25 rows at a time.

c. Lastly, run the *data_analysis.py* file **once**. This will use the JOIN to perform calculations and create visualizations as well as write the calculations to two new TXT files.

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

spotify_api.py

function	description	input	output
add_next_25_songs()	This function uses a spotify api through *spotipy* to grab the top 100 songs from the global top 100 playlist. It creates a table called spotify within the all_tables database. Each time you run the file it runs this function once adding 25 values each time to the table.	Nothing	Adds 25 rows to the spotify table
main()	Runs the add_next_25_songs() function	Nothing	Runs the above function

Genius_api.py

function	description	input	output
get_top_100_songs()	This function uses a genius api to grab the top 100 songs from the top songs chart. It creates a table called genius and adds 25 rows each time to the table. It also creates a second table linked to the first that gets the 10 artists with the most songs in the current table. It spits out a new top 10 each time you run the file.	Nothing	Adds 25 rows to genius table and 10 rows to genius_artist table.
main()	Runs the get_top_100_songs()	nothing	Runs the above function

	function		

Data_analysis.py

Function	description	input	output
get_artist_song_count()	Joins the tables and creates a text that shows the count of artists that have songs in both charts and then the count of how many total songs in the charts.	None	Writes to a txt file the calculations
create_artist_count_plot()	Makes the calculations above into a bar plot of artists and count	None	Bar plot
get_common_song_count()	Gets the count of how many songs are within both charts. If the same song is in both charts it is added to the count. It performs the calculations and writes it to a file	None	Writes to a txt file the calculations
create_common_song_count_plot()	Makes the calculations above into a bar plot of songs repeated and unique songs	None	Bar plot
main()	Runs all 4 functions above	None	Runs and executes all calculations, and visualizations.

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?
04/01/23	Public API resource to browse different topics and brainstorm ideas. We looked into soundcloud and spotify to begin.	https://github.com/public-apis/public-apis#music	We were able to dive deeper into both API's. We were unable to download the soundcloud extensions but were successful with spotipy.
04/03/23 + 04/07/23 *used throughout	Started to read about how spotipy API worked and what I can pull using this API.	https://spotipy.readthedocs.io/en/2.22.1/	We found the playlist_tracks and realized that is my best bet to pull the information we wanted.
04/07/23 *used throughout	Began to read about soundcloud API. Tried to use the steps to set it up but failed multiple times.	https://developers.soundcloud.com/docs/api/guide	Could not get soundcloud API to work and realized we needed to do more research regarding what an API is and how it can vary per website/platform.
04/12/23	Wanted to learn more about API's to see if we can figure out the soundcloud issue.	https://blog.hubspot.com/website/api-documentation	Realized it was not a me issue it was just the soundcloud API issue. Showed me that we needed to change my approach.
04/12/23	Checked if Apple Music could work again.	https://developer.apple.com/documentation/applemusicapi/	Found out it was $100 for a year of an API.
04/13/23	Checked Amazon Music for a free API	https://developer.amazon.com/docs/music/API_browse_overview.html	We would not be able to pull the information we wanted.

04/14/23	Went back to see what other music platforms we can use for free API	https://github.com/public-apis/public-apis#music	Found Genius and looked at the website to see what is on Genius.
04/14/23 *used throughout	Read about what we can pull from Genius API	https://docs.genius.com/	Realized we can pull the chart to get the top 100 songs
04/14/23 + 04/16/23 *used throughout	Wanted to read more about what python extensions can help me.	https://lyricsgenius.readthedocs.io/en/master/reference/genius.html	Found lyricsgenius extension which showed me many different paths.
04/14/23	Created my API credentials	https://genius.com/api-clients	Had my client Id and secret key to allow my code to work
04/16/23 + 04/17/23 *used throughout	When trying to get my tables to link, execute JOIN or SELECT correctly.	https://www.w3schools.com/mysql/mysql_foreignkey.asp	We were able to master my tables and calculations