SI 206 Final Project Report

Link to Github: https://github.com/skstupak/SI-206-Final-Project-.git

Original and Achieved Goals

The original idea was to get the number of plays on a specific track on the Spotify API and the SoundCloud API and compare it to where it is ranked on the Billboard Hot 100 chart. However, we were unable to get access to the SoundCloud API. Instead, we decided to find the correlation between the number of streams on a song, its genre, and where it is ranked on Billboard. To do this, we wanted to use at least three APIs/websites and make three visualizations from them.

We achieved every one of these goals, beginning with finding the relation of average popularity of songs and their genre to where they are ranked on Billboard. To achieve this, we had to create calculations using retrieved data from API requests and web scraping. These calculations allowed us to determine the average popularity of songs based on different parameters used on each platform, such as position, streams, and the number of weeks spent on the charts. Our three platforms we ended up using were Billboard, Spotify, and Deezer. From these sites and the calculations we created three visualizations: two bar charts and one pie chart.

Problems

The first major problem we faced was that we wanted to use the SoundCloud API. However, we were unable to get access to it. To work around this setback, we used the github link with all the public APIs listed. Going through the music section, we discovered the AudioDB API. We were able to get access to it, but it did not have enough data for the table. Going back to the public APIs list, we then decided to switch to Deezer in order to get more data.

Another problem we encountered happened when scraping Spotify's Top 200 chart. Spotify's titles tended to include [Feat.] or [Remix] in the song name, while Billboard tended to exclude these extraneous details. This meant when we were looping through Billboard's table to find which songs Spotify also had on its charts, we were missing a lot of important songs. During lecture on December 3, Ewelina talked about a similar issue she faced when doing her project about music platforms. This gave us the idea for our solution, which was to create if statements in the ScrapeSpotify function that would identify any songs with (Feat.), [Feat.], dashes, or parentheses in their titles and remove them so we could match with songs in Billboard more precisely later on.

Later in the project, we began receiving the error, "OperationalError: table billboard not found." After looking at discussion boards online and Piazza to see how other students handled this common issue, we discovered a solution. To solve it, we needed to make sure we were in the right directory when running our code because there were multiple Billboard.db in our computers. If we weren't referencing the right one, it would cause this error.

The last problem we faced occurred with the Deezer platform. We needed to get the Deezer specific track id in order to make an API request for specific track information. We also needed to get the genre of each track from the Deezer API but that information was only stored within an API request based on album and not track. To solve this issue, we used an advanced API search to find the track id for each track title and used that information to make a regular API request using each track title. We then recorded that information within the API request by track to get each corresponding album ID, and used that ID to make an API request based on album id to get the information we needed (genre).

Calculation File

This file contains the "average-popularity" and genre for each song on Spotify that is also found on Billboard. "Average-popularity" is the number of streams of a song divided by its position on the Spotify Top 200 list. It is sorted from the highest average popularity to the lowest, helping us to see which genres are extremely popular based on our calculations.


calculations.txt

```
The calculation shows the song title with the highest average popularity and it's genre
Body: 558295.0, Rap/Hip Hop
For The Night: 342714.0, Rap/Hip Hop
Positions: 248939.0, Pop
Stay: 193409.0, Rap/Hip Hop
WAP: 158494.66666666666, Rap/Hip Hop
Blinding Lights: 98458.66666666667, R&B
Before You Go: 87880.6, Alternative
Therefore I Am: 77608.27272727272, Alternative
Lemonade: 70992.08333333333, Rap/Hip Hop
Mood: 59825.5, Rap/Hip Hop,Pop
Blue & Grey: 50687.666666666664, Rock
Hawai: 43920.125, Rap/Hip Hop
Monster: 33773.57894736842, Pop
Holy: 31350.9, Pop
Be Like That: 28324.090909090908, Pop
Laugh Now Cry Later: 25406.916666666668, Rap/Hip Hop
Said Sum: 22190.46153846154, Rap/Hip Hop
34+35: 18642.066666666666, Pop
One Of Them Girls: 17959.09677419355, Country
Levitating: 13115.648648648648, Pop
The Christmas Song (Merry Christmas To You): 12317.28205128205, Pop
Forever After All: 11941.775, Pop
Kings & Queens: 11318.714285714286, Reggae
One Beer: 10938.279069767443, Country
Pretty Heart: 10318.355555555556, Country
Dynamite: 9779.826086956522, Asian Music
Go Crazy: 8714.68, Rap/Hip Hop,R&B
Love You Like I Used To: 8413.43137254902, Country
I Hope: 8243.673076923076, Country
ily: 7058.551724137931, Dance
Life Goes On: 6607.606557377049, Asian Music
Watermelon Sugar: 5523.173913043478, Pop
Still Goin Down: 5427.614285714286, Country
Somebody's Problem: 5344.154929577465, Country
Whats Poppin: 4817.184210526316, Rap/Hip Hop
Let It Snow, Let It Snow, Let It Snow: 3859.770114942529, Holiday
Last Christmas: 3732.258426966292, Holiday
It's The Most Wonderful Time Of The Year: 3214.530612244898, Holiday
What You Know Bout Love: 3074.6960784313724, Rap/Hip Hop
Better Together: 2999.5436893203882, Country
More Than My Hometown: 2930.269230769231, Country
Dakiti: 2848.103773584906, Latin Music
Big, Big Plans: 2721.862385321101, Country,Rock
Savage Love (Laxed - Siren Beat): 2367.8067226890757, Pop
Starting Over: 2025.6439393939395, Country
Jingle Bell Rock: 1753.8275862068965, Pop
All I Want For Christmas Is You: 1726.9795918367347, Holiday
Rockin' Around The Christmas Tree: 1714.4391891891892, Holiday
Feliz Navidad: 1700.275167785235, Holiday
Lonely: 1553.4683544303798, Pop
Bang!: 1451.8484848484848, Alternative
Happy Anywhere: 1377.9298245614036, Country
Rockstar: 1220.4324324324325, Rap/Hip Hop
```

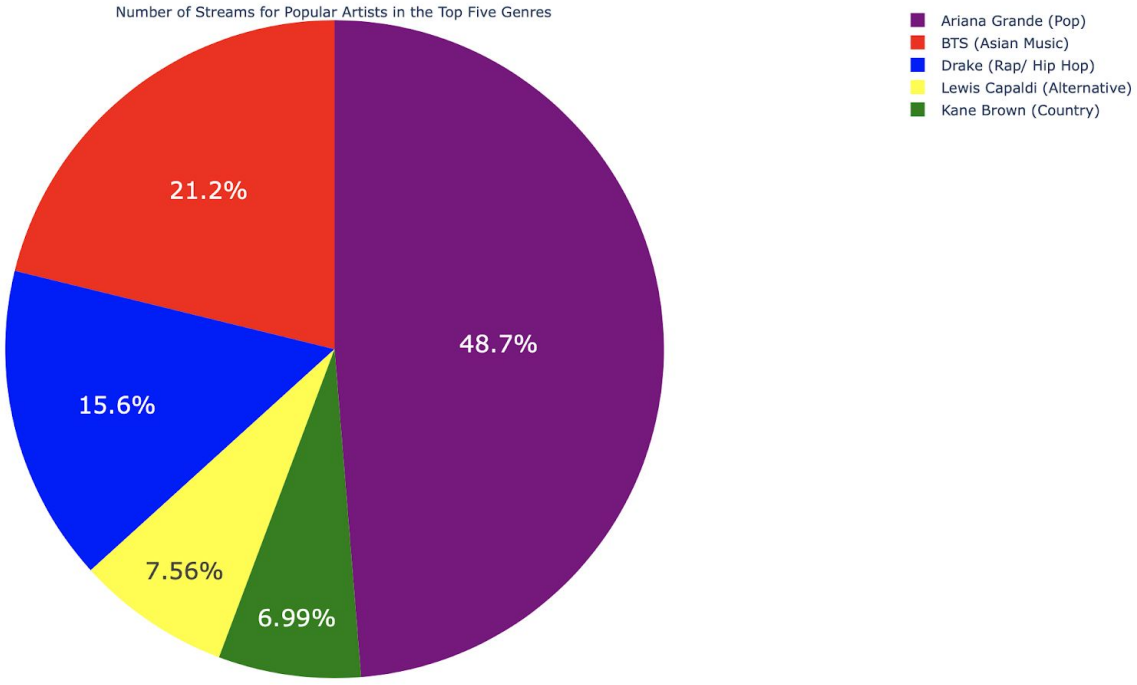
Visualizations

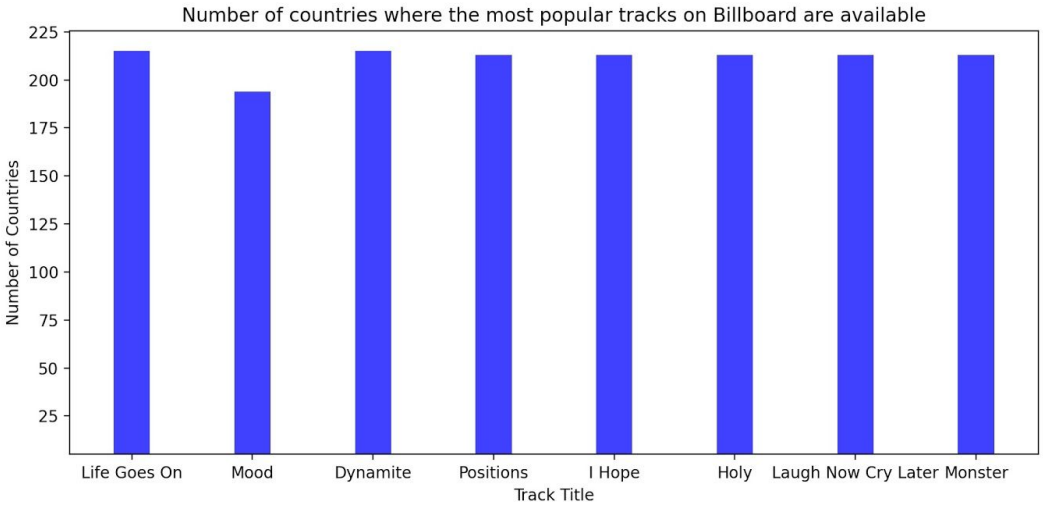
The first visualization is a pie chart that displays five genres of music (Pop, Asian Music, Rap/Hip-Hop, Alternative, and Country) from a popular artist in that genre. The percentage is based on the number of streams from songs of that artist. In our first visualization it can be seen

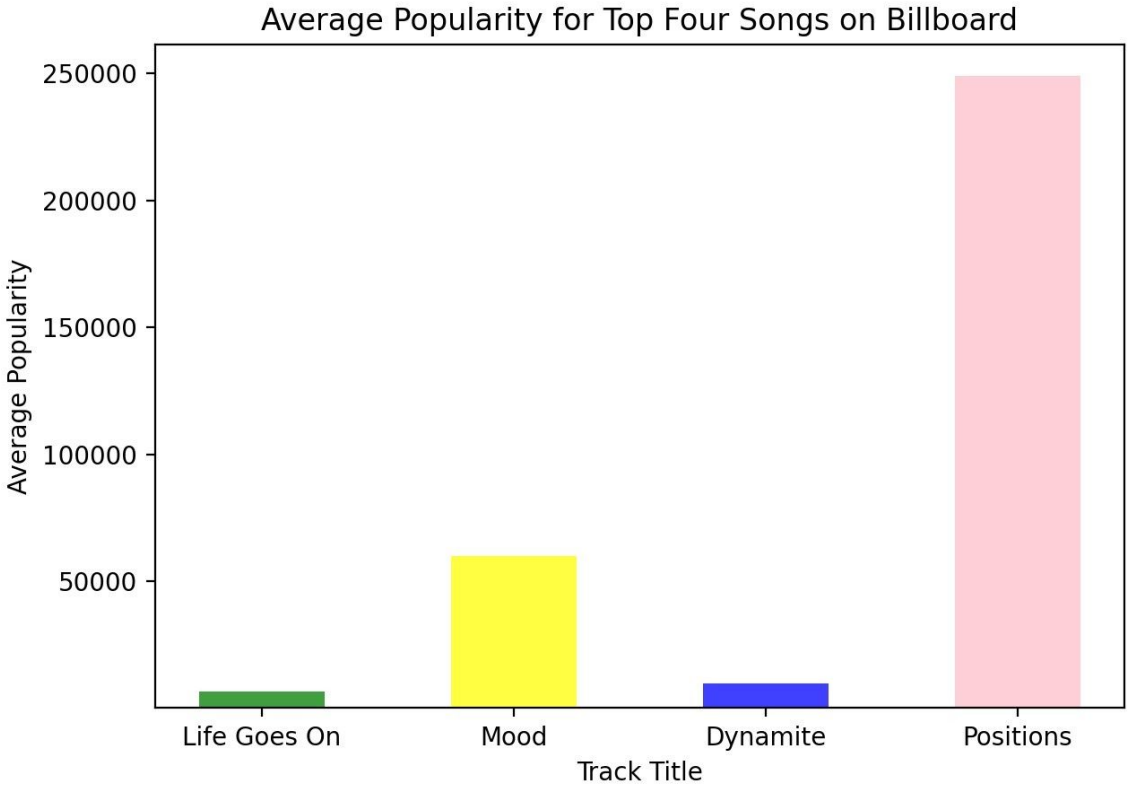
that Ariana Grande and the Pop genre make up a large percentage of streams compared to the other top genres.

The second visualization is a bar graph that displays the most popular tracks on Billboard and the number of countries that they are available in.

The third visualization is a bar graph that displays the average popularity for the top four ranked songs (Life Goes On, Mood, Dynamite, and Positions) on Billboard. The average popularity was calculated using both the number of streams and the chart position from Spotify. We found this calculation interesting because some songs have a higher position on the Spotify chart, yet have a low number of streams, so this allowed us to actually determine the average popularity based on the two parameters. In the chart, it can be seen that Positions by Ariana Grande has a much higher average popularity than the other top songs when taking into consideration both her number of streams and the song's position on the chart.







Instructions for Running the Code

- Run Billboard.py first and do it four times as it populates the Billboard table 25 songs at time, reaching 100 songs after the fourth run.
- Run DeezerAPI_info#1.py second
- Run DeezerAPI_info#2.py third

- Run SpotifyWebScrape.py fourth
- Run SpotifyWebScrape_#2.py fifth
- Run calculations.py sixth
- Run visualizations.py last

Instructions to run BillboardAPI_info.py

- Git clone on your computer https://github.com/guoguo12/billboard-charts.git
- After it is cloned, run **python setup.py install** on your terminal
- This will enable you to run **billboard.py,** which will retrieve information from the API and put it into a database called Billboard.db. This will create a table with the song titles on the Hot100 chart, the song's artist, the songs ranking on the Hot100 chart, and the number of weeks the song has been on the Hot100 chart.
- Run Billboard.py four times to insert 100 songs in the table Billboard. When running it, make sure that you are in the right directory to avoid getting an operational error saying that the Billboard table does not exist.

Instructions to run DeezerAPI_info#1.py

- Run this file after running BillboardAPI_info.py four times
- This file will loop through the Billboard table and retrieve the genre for each song on the Billboard table.
- It will create a new table called Deezer where the song title and genre can be found.

Instructions to run DeezerAPI_info#2.py

- Run this file after running the DeezerAPI_info#1.py
- This file will loop through the Billboard table and retrieve the song's rank on Deezer, the number of countries that the song is available in, and when it was released.

Instructions to run SpotifyWebScrape.py

- Run BillboardAPI_info.py, DeezerAPI_info#1.py, and DeezerAPI_info#2.py first before running this file
- Running this file will create one table: Spotify.The Spotify table is created by running through all of the songs in the Billboard table and adding only the songs that match.

Instructions to run SpotifyWebScrape_#2.py

- Run this file after SpotifyWebScrape.py to get the Top 200 songs. You will need to run it at least four times because each run will only insert 25 songs into the SpotifyTop200 table. You can run it up to eight times, though, to get the full list of songs.
- The SpotifyTop200 table is created after the ScrapeSpotify function is enacted which makes a dictionary of every single Top 200 Spotify song with the corresponding artist, position, and number of streams as the key's values.

Instructions to run calculations.py

- Run BillboardAPI_info.py, DeezerAPI_info#1.py, DeezerAPI_info#2.py, SpotifyWebScrape.py, and SpotifyWebScrape_#2.py
- If you want to see the full calculation, you need to run Billboard.py to its full extent (which means executing the function 4 times) .

Instructions to run visualizations.py

- Before running this, in your terminal do the following:
    - pip install chart_studio
    - pip install plotly
- Run this file after running every other file
- The first visualization will be a bar graph, when you are done looking at it, click the exit button and the next bar graph will be displayed.
- After you look at the second bar graph, exit it and then the pie chart will be displayed.

Code Documentation

a. Functions in BillboardAPI_info.py

- def db_setup:
    - Input: database name
    - Output: creates a database called Billboard.db
- def get_data:
    - Input: None
    - Output: Returns the Hot100 chart with the title of the songs and the artist in order of the top 100 ranked songs from 11/21/2020.
- def create_database:
    - Input: None
    - Output: Creates Billboard table in the database
- def main:
    - The main function executes the create_database function

b. Functions in SpotifyWebScrape.py

- def GetSoupObject:
    - Input: url
    - Output: Creates and returns a soup object from the URL
- def ScrapeSpotify:
    - Input: a soup object
    - Output: Scrapes the website to create a dictionary of song titles as keys and the corresponding artist, position on chart, and number of streams as the values
- def join_tables:
    - Input: cur, conn

- Output: Returns a joined table where Spotify, Deezer, and Billboard share the same title for songs

c. Functions in DeezerAPI_info#1.py

- def getConnection:
  - Input: database
  - Output: This function returns a connection to the database which is defined in the input.
- def getSongInfo:
  - Input: cur
  - Output: This function returns a list of tuples containing the song information (track title, artist name) that was gathered from the Billboard API.
- def makeTable:
  - Input: cur
  - Output: This function uses the cursor to create a table within the database to hold the information gathered from the Deezer API.
- def makeRequest:
  - Input: url
  - Output: This function returns the loaded information from an API request from a URL that is inputted in the function.
- def getRequest:
  - Input: cur
  - Output: This function returns the information we requested from the Deezer API which includes a track title and its corresponding song genre.
- def main:
  - We use the Main function to execute the defined functions above. The main function also utilizes conn.commit() in order to populate the created table within the database with the information gathered from getRequest() which includes the song title and corresponding genre.

d. Functions in DeezerAPI_info#2.py

- def getConnection:
  - Input: database
  - Output:This function returns a connection to the database which is defined in the input.
- def getSongInfo:
  - Input: cur (cursor)
  - Output:This function returns a list of tuples containing the song information that was gathered from the Billboard API.
- def makeTable:
  - Input: cur (cursor)

    - Output: This function uses the cursor to create a table within the database to hold the information gathered from the Deezer API.
- def join_tables:
    - Input: cur, conn
    - Output: This function returns two tables that share a key for the Deezer API. The API requires different parameters and URLs, so that is why they are able to be joined with a primary key(the track title).
- def getRequest:
    - Input: cur (cursor)
    - Output: This function returns the loaded information from an API request from a URL that is inputted in the function.
- def calculating_popularity:
    - Input: cur
    - Output: This function returns the calculated popularity of the top songs on Billboard by fetching its Deezer ranking, the amount of weeks that track has been on Billboard's top 100, and the track's title.
- def main:
    - Input: None
    - Output: We use the Main function to execute the defined functions above. The main function also utilizes conn.commit() in order to populate the created table within the database with the information gathered from getRequest() which includes the song title, it's corresponding Deezer ranking, the amount of countries that song is available in, and that track's release date.

e. Functions in visualizations.py

- def createPieChart1:
    - Input: cur
    - Output: Creates pie chart of the number of streams of an artist in the top five genres
- def createBarChart:
    - Input: None
    - Output: This Bar chart displays the amount of countries each of the top 8 songs from Billboard are available in.
- def makeBarGraph2:
    - Input: None
    - Output: Creates a bar graph of the average popularity of the top four songs on Billboard
- def main:
    - The main function executes the functions above.

f. Functions in calculations.py

- lines 6 - 30

- Calculates the “average-popularity” by dividing the number of streams and the song’s position
- The song’s title is the key in a sorted dictionary where the values are a tuples of its average popularity and genre
- This is then written to a text file

g. Functions in SpotifyWebScrape_#2.py

- def GetSoupObject:
  - Input: url
  - Output: Creates and returns a soup object from the URL
- def ScrapeSpotify:
  - Input: a soup object
  - Output: Scrapes the website to create a dictionary of song titles as keys and the corresponding artist, position on chart, and number of streams as the values
- Lines 69 - 89
  - Creates table of top 200 songs

Documentation and Resources Used

Date	Issue Description	Location of Resource	Result (Did it solve the issue?)
11/21/2020	**Began the project and wanted to get data from Spotify, Billboard and Soundcloud.**	**We used the public APIs link to discover which platforms we should use.** **https://github.com/public-apis/public-apis**	**This helped us figure out platforms we would be able to use.**
11/22/2020	**Couldn’t extract data from SoundCloud API**	**https://github.com/public-apis/public-apis**	**We went back to the public APIs link and found the music section. This helped us discover other platforms we could use - AudioDB and Deezer.**
12/3/2020	**Needed help getting Pie Chart to display**	**Lecture in Week 13 (Plotly-v4)**	**Yes, it helped us figure out how to display the Pie Chart. We created a separate file to**

			house all of our visualizations which then allowed them all to display.
12/4/2020	**Wanted help creating a bar chart using matplot.lib**	**Matplotlib Bar chart - Python Tutorial (pythonspot.com)**	**It helped us start to create a bar chart and then we styled it to uniquely display our data effectively.**