Niurca Quirarte
Gloria Yu
Nadia Jahan
Apr 30, 2024
SI 206 Final Project Report

Exploring Music Genres: A Data Analysis of Top Artists and Songs

Goals for Project

Our project aimed to understand music better by analyzing different types of songs and artists. We planned to gather data from three main sources: Spotify, iTunes, and the Billboard Top 100 website. From Spotify, we wanted to learn about artists, their top songs, and how popular they are. Using the iTunes API, we aimed to collect similar information, focusing on artists' primary genres. Additionally, we intended to scrape the Billboard website to obtain a list of the top 100 artists, which would form the basis of our analysis. We planned to calculate the percentage of each genre within the top 100 artists and determine the average length of the top song for each artist. For visualization purposes, we decided to use Matplotlib.

Goals Achieved

Throughout the project, we were able to achieve the majority of our goals by working with the Spotify API, using the Spotipy library to gather data on artists and songs. First, we got a list of 100 artists from the Billboard website. This helped us start gathering data. We then used the Spotify API to carefully gather details about artists and their songs, like their Spotify IDs, genres, and how popular they are. We did the same with the iTunes API, getting information about artists' iTunes IDs, primary genres, and how long their songs are. Once we had all the data we needed, we filled up our database systematically. We used specific functions for Spotify and iTunes data, making sure we covered everything: artist details, song lengths, and more. By working with the Spotify API and using tools like Spotipy and the requests library for iTunes, we collected a lot of information, like artist names, IDs, genres, and song details. This gave us a big dataset to learn from, helping us understand music genres and how they're related to popular artists, just like we wanted.

Challenges

We encountered difficulties in finding suitable APIs that met our requirements. Additionally, we struggled with handling and processing the gathered data efficiently. Specifically, we faced issues with the Spotipy API, particularly in setting up client credentials, which caused some delays and setbacks. Similarly, navigating the iTunes API we ran into issues, such as syntax errors and difficulties in extracting the top songs of an artist. We needed to adjust our approach to ensure progress in the data collection. We also, at times, made too many excessive API calls that we were denied access to, in which we had to wait a while to start making API calls again.

Calculations

As part of our analysis, we conducted various calculations from the collected data. We calculated the average length of songs for each genre within the Spotify and iTunes platforms.

We did this by joining tables for songs and artists, joining the data by genre, and computing the average song length in minutes. Additionally, we calculated the percentage of artists in each genre for both Spotify and iTunes by querying the database tables and counting the occurrences of each genre. This helped to analyze and compare the genre distribution between the two platforms. We finally displayed the top 25 artists and their popularity on Spotify to compare how Billboard's top 25 artists were rated on Spotify.

itunes_data.txt

You, 12 hours ago | 1 author (You)

```
Genre,Avg Length of Songs(min)
Classics,0.91
Reggae,3.02
K-Pop,3.02
Hip-Hop/Rap,3.21
Country,3.52
Alternative,3.52
Pop,3.56
Música Mexicana,3.56
R&B/Soul,3.67
Urbano latino,4.18
Hard Rock,4.3
Metal,5.53
Rock,5.54
```

spotify_data.txt

You, 8 minutes ago | 1 author (You)

```
Genre,Avg Length of Songs(min)
southern hip hop,2.3
corrido,2.53
reggaeton,2.71
None,2.81
alt z,2.86
sad sierreno,2.88
neo soul,3.08
country,3.37
pop,3.45
r&b,3.56
rap,3.58
indie,3.58
k-pop,3.71
reggae,3.95
rock,4.2
trap queen,5.32
```

percentage_of_genres_itunes.txt

You, 13 hours ago | 1 author (You)

```
Genre,Percentage
Pop,22
Hip-Hop/Rap,22
Country,18
Rock,9
Alternative,9
R&B/Soul,6
K-Pop,4
Música Mexicana,3
Hard Rock,3
Urbano latino,1
Reggae,1
Metal,1
Classics,1
```

percentage_of_genres_spotify.txt

You, 13 hours ago | 1 author (You)

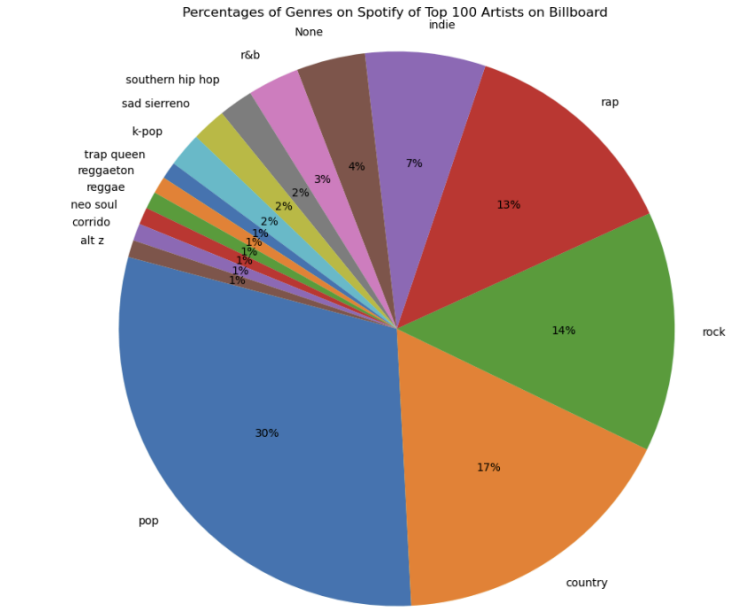
```
Genre,Percentage
pop,30
country,17
rock,14
rap,13
indie,7
None,4
r&b,3
southern hip hop,2
sad sierreno,2
k-pop,2
trap queen,1
reggaeton,1
reggae,1
neo soul,1
corrido,1
alt z,1
```

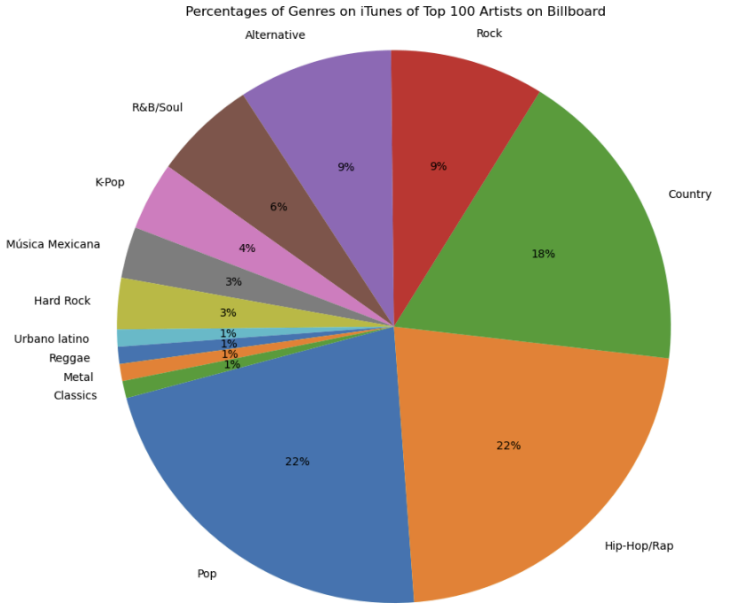
spotify_popularity.txt

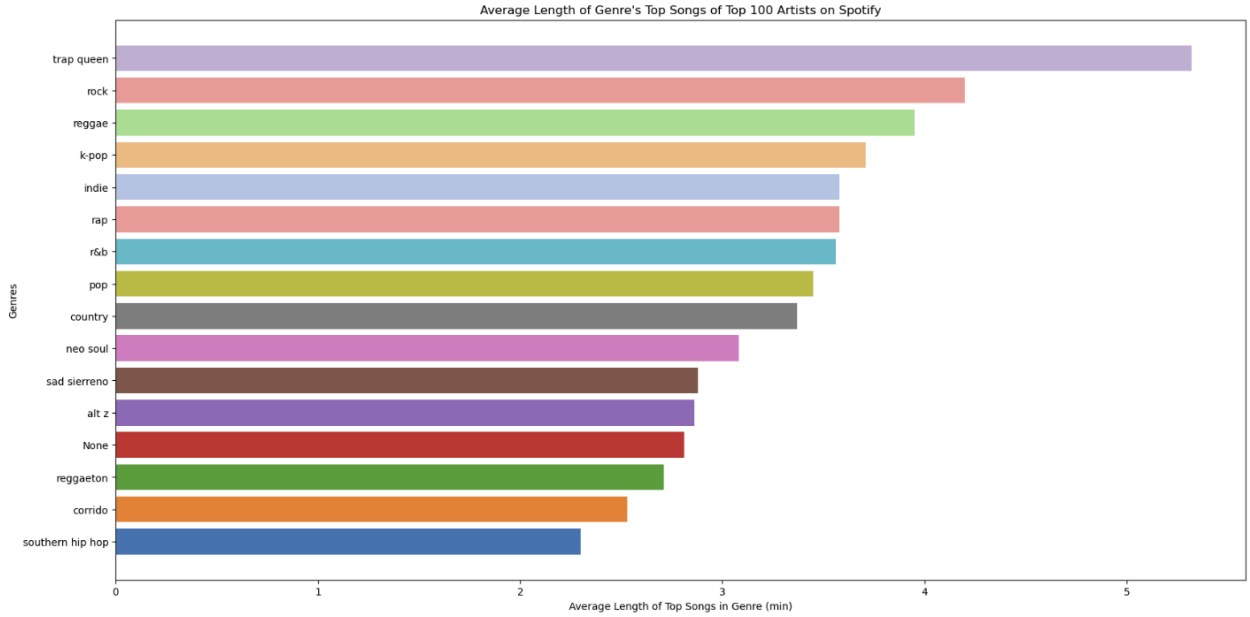
```
Artist,Popularity
Taylor Swift,100
Beyonce,88
Morgan Wallen,85
Future,91
Metro Boomin,90
Linkin Park,83
SZA,87
Luke Combs,81
Benson Boone,83
Zach Bryan,85
Noah Kahan,83
TOMORROW X TOGETHER,77
Ariana Grande,90
Jelly Roll,74
Hozier,84
Teddy Swims,79
Chris Stapleton,78
Drake,92
Dua Lipa,85
Olivia Rodrigo,86
Lana Del Rey,87
Sabrina Carpenter,81
J. Cole,84
Travis Scott,90
The Weeknd,91

```

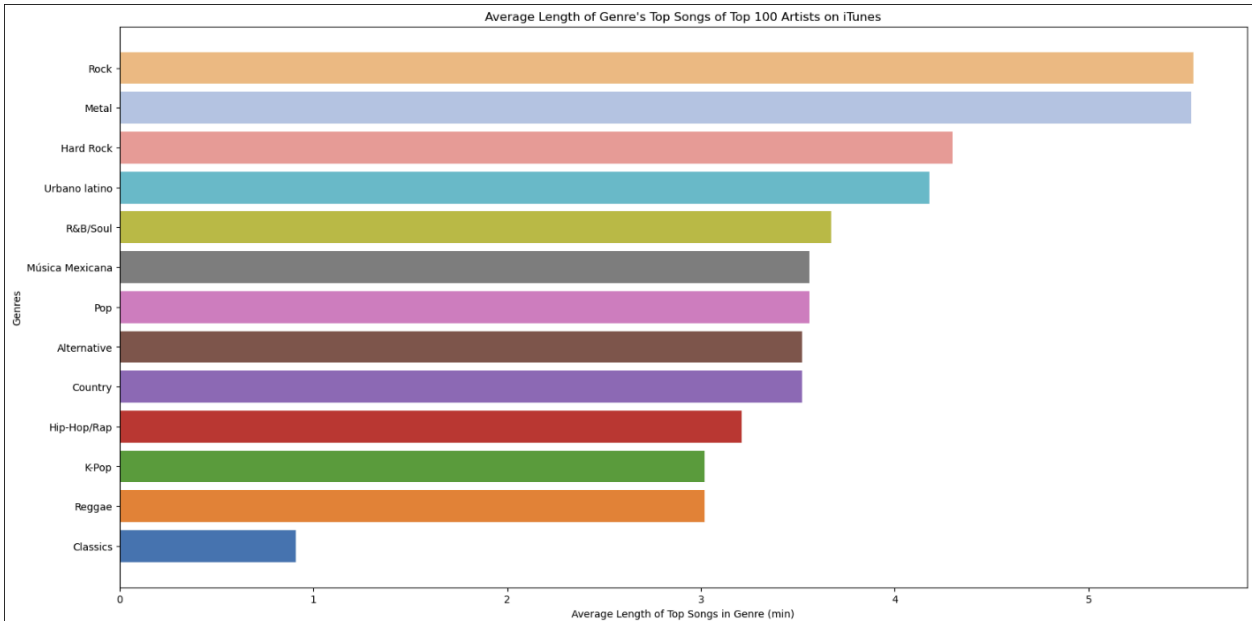
Visualizations

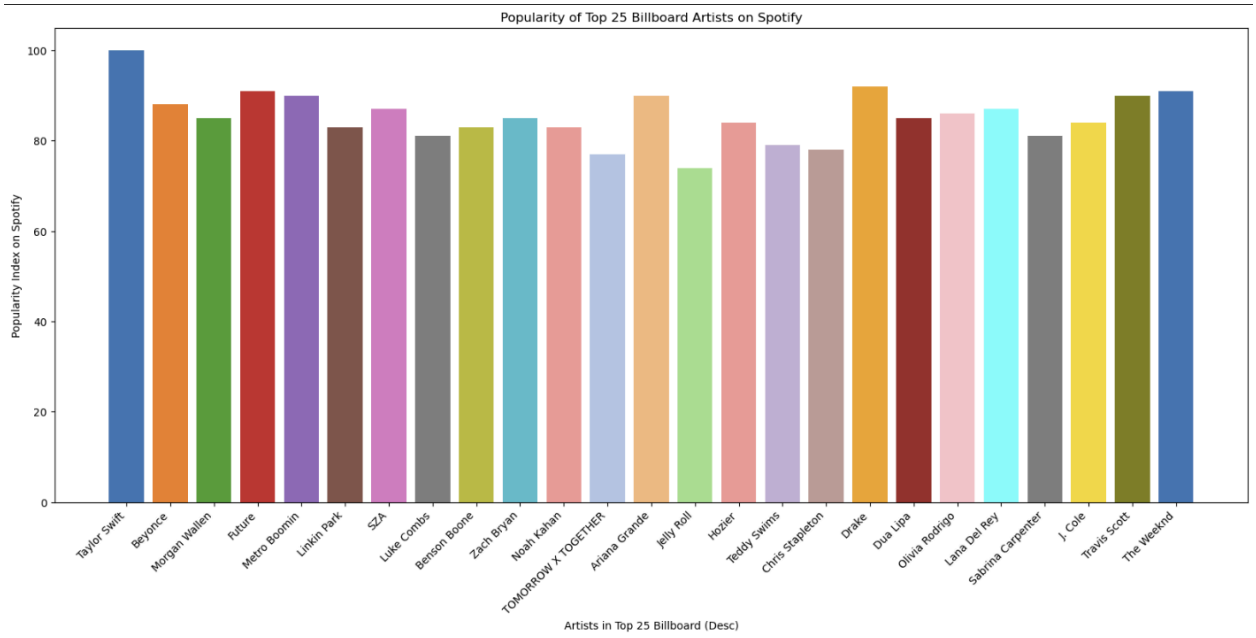
We created visualizations using Matplotlib to better understand and present the data. The first two visuals display the percentage of each genre in the top 100 artists across different music platforms. The following two visualizations provide the distribution of genres and song lengths across different music platforms. Finally, the last visual provides an idea of the differences and similarities between Spotify's rankings and Billboard's rankings for the top 25 artists on Billboard.


Percentages of Genres on Spotify of Top 100 Artists on Billboard
indie
rap
rock
country
pop
alt z
corrido
neo soul
reggae
reggaeton
trap queen
k-pop
sad sierreno
southern hip hop
r&b
None
7%
13%
14%
17%
30%
1%
1%
1%
1%
1%
2%
2%
2%
3%
4%


Percentages of Genres on iTunes of Top 100 Artists on Billboard
Rock
Country
Hip-Hop/Rap
Pop
Classics
Metal
Reggae
Urbano latino
Hard Rock
Música Mexicana
K-Pop
R&B/Soul
Alternative
9%
18%
22%
22%
1%
1%
1%
1%
3%
3%
4%
6%
9%


Average Length of Genre's Top Songs of Top 100 Artists on Spotify
trap queen
rock
reggae
k-pop
indie
rap
r&b
pop
country
neo soul
sad sierreno
alt z
None
reggaeton
corrido
southern hip hop
Genres
0
1
2
3
4
5
Average Length of Top Songs in Genre (min)


Average Length of Genre's Top Songs of Top 100 Artists on iTunes
Rock
Metal
Hard Rock
Urbano latino
R&B/Soul
Música Mexicana
Pop
Alternative
Country
Hip-Hop/Rap
K-Pop
Reggae
Classics
Genres
0
1
2
3
4
5
Average Length of Top Songs in Genre (min)



Instructions for running the code

Follow the instructions from Spotipy API, you should create an account and then an app. Install Spotipy in your terminal. Then export into the terminal your client id and client secret as variables, these are in your dashboard>settings when you create an app. Next run main_data.py four times and then run main_results.py once to complete the calculations and create the visuals.

Documentation for each function

Top_artists_web.py

1. Function: artist_retrieval
   Input : None
   Output: Returns a list of artist names scraped from the Billboard Top 100 Artist chart - 'artist_list' (list of strings)
   The function accesses the Billboard Top 100 Artists chart webpage and uses requests to get the HTML content of the page and BeautifulSoup to parse it. It extracts information based on the HTML element (h3 tags with the class "a-no-truncate") and strips any white spaces.

2. Function: artist_table
   Input : cur, conn, artists (list of strings) - A list of artist names to be inserted into the database
   Output: None, performs database operations
   First, the function checks the current count of the artist entries in database, then it inserts a new batch of artist names into the database until it reaches a maximum of 100. The function eliminates duplicates and increases efficiency.

Spotify_api.py

1. Function: shortest_genre
   Input: genres (list of strings) - A list containing genre names associated with an artist.
   Output: minG (string or None) - Returns the shortest genre name that matches a predefined list
   This function determines the shortest genre name from a list of genre strings matched from a predefined list of genre type keywords. The functions got the shortest genre by iterating through the genre list and updating the shortest genre based on string length.

2. Function: check_database_entries
   Input: cur, (string) table - the name of the database table to check for entries
   Output: A tuple for the range of entries to be processed (0-25, 25-50, etc.), or 'None' if the table already contains 100 entries.
   The functions checks the database table to determine the current count and returns a range or 'None'.

3. Function: cur, gather_spotify_artist_info
   Input: artists (list of strings) - A list containing artist names.
   Output: l (list of tuples) - Returns a list of tuples, each containing the Spotify ID, genre, and popularity of an artist. If no data is found, the tuple will contain None values for each element.
   This function calls the Spotipy API to gather data (ID, genre, popularity).

4. Function: gather_spotify_songs
   Input: artists_info (list of tuples) - A list of tuples where each tuple contains information about an artist including their Spotify ID.
   Output: l (list of tuples) - Returns a list of tuples, each containing the name and duration (in milliseconds) of the top track for each artist specified in the input list.
   This function gets the duration of the top songs of each artist in the list and gathers the duration.

5. Function: spotify_tables
   Input: cur, conn, artists_info (list of tuples) - Contains artist or song information to be inserted into the database, art_table (boolean) - Flag to determine whether to insert into the artists table (True) or songs table (False).
   Output: No output, performs database operations.
   This function inserts the data into the database, given the provided artist information and changes are committed to the database.

Itunes_api.py

1. Function: check_database_entries
   Input: cur, (string) table of name of the database table to check for entries
   Output: a tuple representing the range of new entries or 'None'.

This function checks how many entries are in the database table and determines the following batch to be processed into the table.

2. Function: gather_itunes_artist_info
   Input: A list containing names of artists - ‘artists’ (list of strings)
   Output: A list of tuples, each containing the iTunes ID and primary genre of an artist.
   This function retrieves artist information from iTunes API bases on the ‘artists’ list and collects their unique iTunes ID and genre.

3. Function: gather_itunes_songs
   Input: A list of tuples (‘artist_info’),
   Output: A list of tuples, each containing the name and duration (in milliseconds) of the first song found for each artist, ‘l’.
   This function finds the duration of the top song (first song) from iTunes for each artist.

4. Function: itunes_tables
   Input: cur, conn, a list of tuples containing artist or song information (‘artists_info’), ‘table’ which specifies if the data should be inserted into the itunes_artists or itunes_songs table.
   Output: None, performs database operations
   This function inserts data about an artist or song into an iTunes database table and commits these changes to the database.

Calculations.py

1. Function: avg_time_per_genre_spotify
   Input: cur
   Output: None
   The function calculates the average song length per genre from Spotify data and joins song and artist tables, then it writes the average song length in minutes into the ‘spotify_data.txt’

2. Function: avg_time_per_genre_itunes
   Input: curr
   Output: None
   The function calculates the average song length per genre from iTunes data and joins song and artist tables, then it writes the average song length in minutes into the ‘itunes_data.txt’

3. Function: genre_percentages_spotify
   Input: cur
   Output: None
   This function computes the percentage of Spotify artists by genre and uses a query to count the number of artists per genre. It then writes the counts into ‘percentage_of_genres_spotify.txt’.

4. Function: genre_percentages_itunes
   Input: cur
   Output: None
   This function computes the percentage of iTunes artists by genre and uses a query to count the number of artists per genre. It then writes the counts into 'percentage_of_genres_itunes.txt'.

5. Function: spotify_popularity
   Input: cur
   Output: None
   This function uses a query to search for the popularity of Spotify artists and joins an artists table with a Spotify one. It then adds the popularity of the top 25 artists by artist id to 'spotify_popularity.txt.'.

Visuals.py

1. Function:read_spotify_percentages
   Input: None
   Output: Pie chart
   This function reads the genre percentages from the text file and visualizes the data into a pie chart, showing the distribution of genre among the selected artists.

2. Function: read_itunes_percentages
   Input: None
   Output: Pie graph
   Like the function above, this function gets the data from the text file and visualizes the genre percentages in a pie chart.

3. Function: read_spotify_data
   Input: None
   Output: Horizontal bar graph
   This function gets the average song length for each genre from Spotify and visualizes them into a horizontal bar chart.

4. Function: read_itunes_data
   Input: None
   Output: Horizontal bar graph
   Like the one above, this function gets the average song length for each genre from iTunes and visualizes them into a horizontal bar chart.

5. Function: read_spotify_pop
   Input: None
   Output: Bar chart

This function takes the popularity data from ‘spotify_popularity.txt’ and visualizes the data into a bar chart, showing the popularity of the selected artists.

Resources

Date	Issue Description	Location of Resource	Result(did it solve the issue?)
4/27-4/30	Issue revolved around correct syntax for api calls and what data type and data api calls returned for itunes.	https://performance-partners.apple.com/search-api This resource was used in the file itunes_api.py.	Yes this resolved it by giving a rough outline of how to make api calls with specific data to be returned.
4/18 - 4/30	Issue revolved around what an api call for an artist is going to return on the itunes api.	https://developer.spotify.com/documentation/web-api/reference/get-an-artist This resource was used in the file itunes_api.py.	Yes this resolved it by giving an example of what data/data types will be returned when making an api call for an artist.
4/18 - 4/30	Issue revolved around getting an error message that my token expired when trying to make api calls for spotify.	https://community.spotify.com/t5/Other-Podcasts-Partners-etc/API-Access-token-expired/td-p/4695256 This resource was used in the file spotify_api.py.	Yes this resolved it by leading me to the documentation for refreshing my token.
4/18 - 4/30	Issue revolved around refreshing my token and ensuring credentials were verified when making api calls to spotify.	https://spotipy.readthedocs.io/en/2.22.1/?highlight=access%20token#authorization-code-flow This resource was used in the file spotify_api.py.	Yes this resolved the issue by allowing me to refresh my access token and also make sure I was using the correct credentials to make api calls.

4/27 - 4/30	Issue was what data api returned and how to make api calls for itunes.	https://ruan.dev/blog/2018/05/08/use-python-requests-to-interact-with-the-itunes-api-to-search-for-music-info This resource was used in the file itunes_api.py.	Yes, this resolved the issue by giving an example of how to make api calls and how to load the data to be able to use it.
4/18 + 4/27	Issue revolved around how to connect to the database, create tables, insert data, & select data.	HW 7 This resource was used in the file spotify_api.py, itunes_api.py, main_data.py, top_artists_web.py, & calculations.py.	Yes this helped as it reminded us how to work with sqlite3 as a module so we can create the tables and insert & select data.
4/18	Issue revolved around using requests and Beautiful Soup objects to web scrape.	HW 6 This resource was used in top_artists_web.py & itunes_api.py.	Yes this helped us gather the data by web scraping our website with the top 100 artists on Billboard.
4/29 - 4/30	Issue revolved around how to make visuals and ensure that data is plotted accurately.	HW 8 This resource was used in visuals.py.	Yes this helped us by reminding us how to plot and add titles/labels to the diagrams.

GitHub Link
https://github.com/niurcaq/final_project206