Ethan Finestone & Andrew Bradley
Dr. Ericson
SI 206
12/11/23
Github Repository:
https://github.com/dbrad09/Finalproject206/tree/mainhttps://github.com/dbrad09/Finalproject206.git

Final Report

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

For our original proposal we had planned to work with the spotipy and Apple music API's to gather data on the top charts from a variety of different regions and compare which albums, songs, and artists have the top ratings and most plays.

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

In our project we used data from two API's, Spotipy and Billboard Top 100. From Spotify we gathered the top 100 songs and their artists from the charts. From the Billboard top 100 we gathered the top 100 artists and data related to their rankings in the charts. This included their peak position, position from the previous week, and number of weeks in the chart. After much work we were able to create two visualizations. The first chart shows the top 50 artists and the amount of time they have spent on the charts. This illustrates the time popular artists today have spent on the top charts. The second chart displays the artists with more than one appearance on the top 100 charts, illustrating a more current representation of popularity.

3. The problems that you faced

The main problems we first faced were actually gathering data from API's. Originally we had planned to use the Apple Music API but that first required a subscription plan and a long process to get a developer token. We also encountered problems with finding a chart where we could split the data into two tables without encountering duplicate strings. After switching over to a special chart in Billboard-API we were able to do so. We also had trouble when creating the databases without duplicate strings, as many of the artists often appeared multiple times in the top charts, so we ended up giving them ArtistID's. Another problem we encountered was compiling each of our tables to a single

Database. This problem took a while to figure out but after manipulating the code we were able to do so.

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

artist_info.txt

```
Artist Count:
A Mar dream: 1
Ado: 1
Andy Williams: 1
Ariana Grande: 1
Bad Bunny: 1
Bing Crosby: 1
Ken Darby Singers: 1
John Scott Trotter &
Bobby Helms: 1
Brenda Lee: 1
Burl Ives: 1
Calle 24: 1
Chino Pacas: 1
Fuerza Regida: 2
Chuck Berry: 1
Darlene Love: 1
Dean Martin: 1
Doja Cat: 1
Drake: 7
J. Cole: 2
SZA: 3
Sexyy Red: 1
Yeat: 1
Eagles: 1
Eartha Kitt: 1
Ed Sheeran: 1
Elton John: 2
Elvis Presley: 1
Frank Sinatra: 1
Marshmello: 1
Future: 4
Tems: 1
The Weeknd: 1
Gunna: 1
JID: 1
21 Savage: 1
Baby Tate: 1
Jack Harlow: 1
Jonas Brothers: 1
José Feliciano: 1
Jung Kook: 1
Latto: 1
Justin Bieber: 1
Kelly Clarkson: 1
King Gnu: 1
LE SSERAFIM: 1
Lil Baby: 1
Mariah Carey: 1
Michael Bublé: 1
Miley Cyrus: 1
Morgan Wallen: 1
Nat King Cole: 1
Nicki Minaj: 9
Lil Uzi Vert: 1
Lil Wayne: 1
Tate Kobang: 1
Lourdiz: 1
Skillibeng: 1
Skeng: 1
Tasha Cobbs Leonard:
Noah Kahan: 1
OFFICIAL HIGE DANDISM
Olivia Rodrigo: 1
Paul McCartney: 1
Peso Pluma: 1
Gabito Ballesteros: 1
Junior H: 1
Rod Wave: 1
Senzei: 1
Sia: 1
Tate McRae: 1
Taylor Swift: 1
The Beach Boys: 1
The Ronettes: 1
Travis Scott: 1
Tyla: 1
Wham!: 1
Zach Bryan: 1
Kacey Musgraves: 1
nezsa: 1

Top Performing Artist
Fuerza Regida: 2
Drake: 7
J. Cole: 2
SZA: 3
Elton John: 2
Future: 4
Nicki Minaj: 9

```

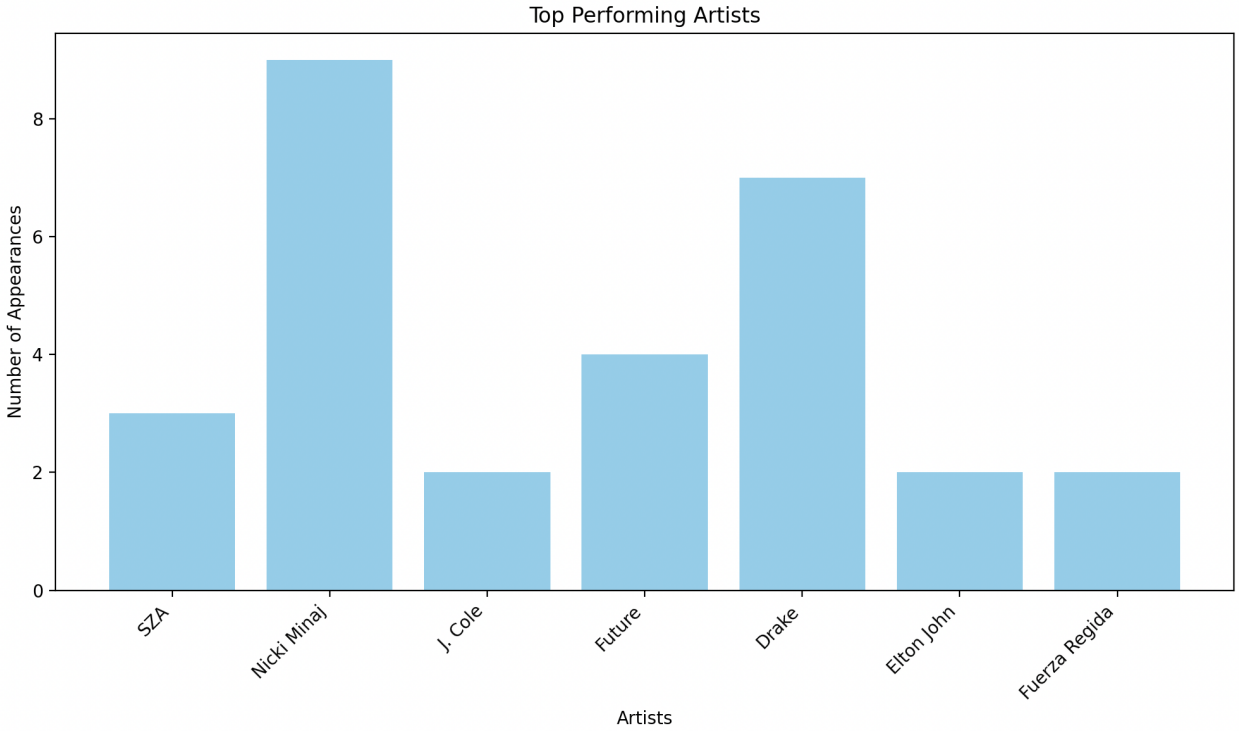
```
Top 50 Artists and Their Total Weeks on Chart:
Drake: 490 weeks
Taylor Swift: 486 weeks
Ed Sheeran: 483 weeks
Bruno Mars: 468 weeks
Justin Bieber: 456 weeks
Eminem: 454 weeks
Jason Aldean: 442 weeks
The Weeknd: 436 weeks
Chris Stapleton: 421 weeks
Metallica: 417 weeks
Ariana Grande: 417 weeks
Chris Brown: 414 weeks
Kendrick Lamar: 411 weeks
Michael Jackson: 406 weeks
Post Malone: 386 weeks
Travis Scott: 385 weeks
J. Cole: 360 weeks
Luke Combs: 350 weeks
Future: 313 weeks
The Beatles: 300 weeks
Bad Bunny: 294 weeks
Rihanna: 289 weeks
Dua Lipa: 289 weeks
Lil Baby: 288 weeks
Selena Gomez: 279 weeks
Billie Eilish: 265 weeks
Morgan Wallen: 245 weeks
Kanye West: 243 weeks
Fleetwood Mac: 241 weeks
Harry Styles: 229 weeks
SZA: 220 weeks
Doja Cat: 195 weeks
AC/DC: 190 weeks
Nirvana: 180 weeks
Kodak Black: 176 weeks
Lana Del Rey: 174 weeks
21 Savage: 168 weeks
Miley Cyrus: 160 weeks
YoungBoy Never Broke Again: 153 weeks
```

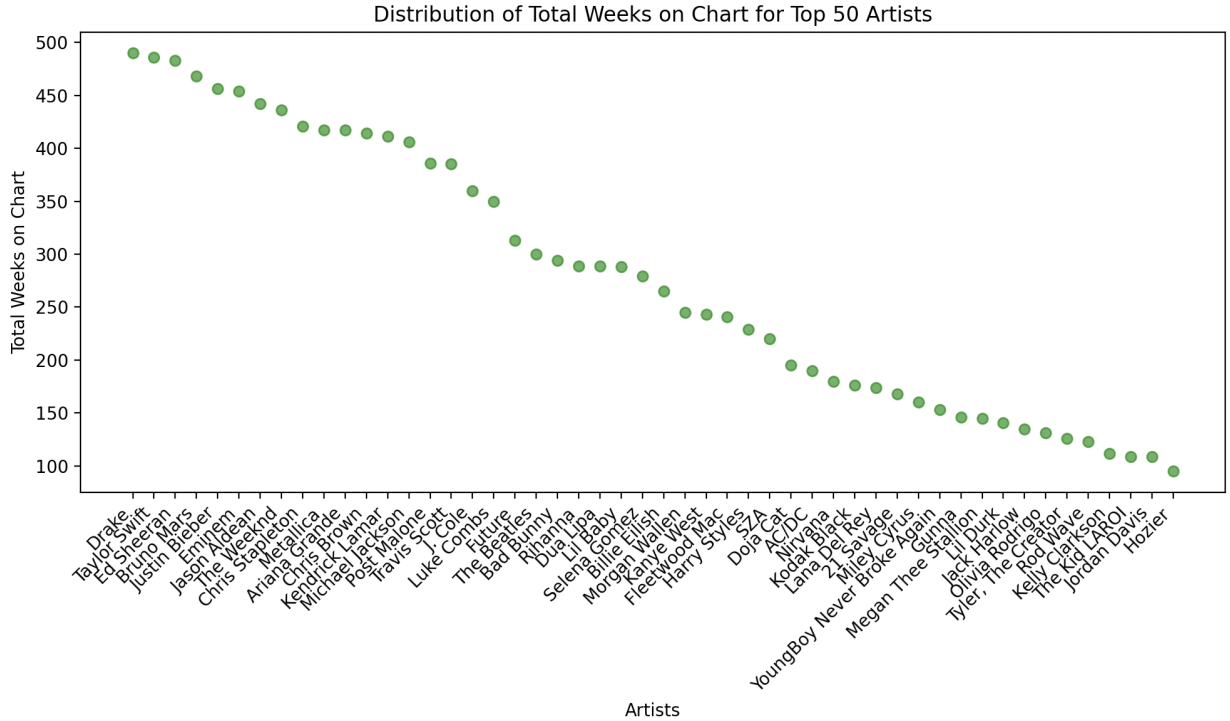
```
21 Savage: 168 weeks
Miley Cyrus: 160 weeks
YoungBoy Never Broke Again: 153 weeks
Gunna: 146 weeks
Megan Thee Stallion: 145 weeks
Lil Durk: 141 weeks
Jack Harlow: 135 weeks
Olivia Rodrigo: 131 weeks
Tyler, The Creator: 126 weeks
Rod Wave: 123 weeks
Kelly Clarkson: 112 weeks
The Kid LAROI: 109 weeks
Jordan Davis: 109 weeks
Hozier: 95 weeks

```

Note: The two pictures on the right are one text file in two screenshots.

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





6. Instructions for running your code

For Billboard.py:

Instructions are very straightforward and simple. Every time the play button is clicked, 25 items will be displayed in the chart_entries table and the second_table table. The visualization will also update and display each time the code is run. There is a limit on how many times you can get a response for the API at 40 times. If problems arise, please let me know so I can generate a new key.

For spot-api.py:

Similar to running billboard.py, you have to run spot-api.py four times in order to get the top 100 songs. Each time the code is run, it stores 25 songs in the database. To get the visualization representing the amount of time artists appear in the top 100 songs more than once, you need to get the file to store all 100 songs.

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

Billboard.py file:

1. First lines of code import packages for requests, analysis, SQL, and set variables for API call.
2. create_table(cursor):
    a. This function first creates the two tables chart_entries and second_Table and the correct column names.
    b. It takes in the cursor object from SQL.
3. create_databse():
    a. This function does not take any parameters but simply just checks if the databases were created and if not creates them.
4. insert_entry_into_db(entry, cursor):
    a. This function takes information from the entry variable which is information from a dictionary and correctly places it into one of the two tables. It takes in the parameters entry(information from a dict) and the cursor object.
5. get_current_entry_count(cursor,table_name):
    a. This function has a very simple task to find the number of entries currently in the table. This helps when we are putting entries in 25 at a time.
6. process_entries(entries,cursor,table_name,start_index, batch_size=25):
    a. This function works at a little bit higher of a level than insert entries but uses the previous function to find a start index and then calls insert_entry_into_db for each entry only processing 25 entries at a time.
7. delete_database():

    a. This was a function I had to insert just in case I needed to delete the database and is not currently being called in the main function.
8. calculate_top_avg_weeks()
    a. This function first joins two tables together on the rank INTEGER key and then finds the top 50 artists with the most weeks on the charts. This function returns a tuple(Artist, Numweeks on charts)
9. visualize():
    a. Uses a bar chart to visualize the top 5- artists by number of weeks on the charts.
10. write_to_file():
    a. Writes the top 50 artists and their number of weeks on the top charts to a file called top_artists_weeks.txt

Spot-api.py file:

1. top_songs
    a. Retrieves Top 100 Songs from a Spotify Playlist
    b. Creates or Checks for a database table
        i. Connects to SQLite database named 'chart_entries' and creates a table named TopSongs if it doesn't already exist and another called Artists
        ii. table has columns SongName and Artists
    c. Checks Database Entries and Adds New Entries:
        i. processes batches of 25 songs at a time, only adding songs that dont already exist
        ii. For each song, every artist on the track is given a unique ID stored in Artists database
2. Artist_appearances
    a. Connects to a SQLite Database chart_entries.db
    b. Executes an SQL Query:
        i. retrieves information from the TopSongs table
        ii. uses GROUP BY ArtistIDs to group the songs by their ArtistIDs and COUNT(*)
    c. Identifies Top Performing Artists
        i. After counting the appearances of each artist, it filters out the artists who have appeared more than once and stores this information in the top_artists dictionary
    d. Prints out the amount of times ArtistIDs appear in top 100 songs and the ArtistIDs of top performing artists
    e. Returns Data artist_count and top_artists

3. top_artist_vis(top_artists)
    a. Connects to chart_entries.db
    b. References ArtistName for each ArtistID in top_artists
    c. Visualizes the data in top_artists in bar chart with artist names and the amount of time they appear in top 100
4. text_file(artist_count, top_artists, file_name)
    a. takes in two dictionaries artist_count and top_artists along with a file_name string
    b. For each artist in artist_count, and top_artists it identifies ArtistName for each by referencing ArtistID in Artists db
    c. Writes out the artist and their count of appearances in the format Artist: Count to the file_name provided
5. main()
    a. serves as the main entry point of the program
    b. calls the top_songs() function
    c. Then, it calls the artist_appearances()
        i. captures the returned artist_count and top_artists
    d. text_file() with artist_count, top_artists, and file name as arguments to write this artist information into a text file named artist_counts.txt

8. You must also clearly document all resources you used.

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
12-5-23	Getting started with and working with Spotipy API	https://spotipy.readthedocs.io/en/2.22.1/#examples	Yes, however, I had trouble with this considering my file name was originally Spotipy.
12-6-23	Accessing credentials for Spotify account	https://www.youtube.com/watch?v=3RGm4jALukM	Yes, I was able to get my credentials for my account.
12-7-23	Accessing top 100 playlist on Spotify	https://github.com/spotipy-dev/spotipy/tree/master/examples	Yes, I was successfully able to access the top 100 playlist by following the example code.
12-7-23	Accessing Rapid API	https://rapidapi.com/DevoCat/api/billboard-api5/	Yes, I was able to use this to get starter code and developer token for API request.
12-8-23	Loading data 25 rows at a time	Chat GPT	Yes, I was able to create an additional function with some help from chat gpt that helped me load in the data only 25 rows at a time to SQL
12-9-23	Fetching Data from SQL	https://www.dataquest.io/blog/sql-commands/	Was not necessarily a problem but this website helped with SQL commands.
12-11-23	Writing code to visualize calculations	https://python-charts.com/	Yes