Konstantinos Hatzopoulos & Daniel Himmelfarb
Dr. Ericson
SI 206
30 April 2024

Perfect Programmers Final Project Report

GitHub Repo: https://github.com/dhimmel01/PerfectProgrammersFinalProj.git

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

We initially planned to use Spotify and Apple music APIs. We wanted to gather data on the top artists of the day, as well as their top songs, top albums, and the song durations. We were hoping to compare the data of the top charts between the two platforms.

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

We worked with the Spotify API and the Billboard API. For Spotify, we gathered data from 100 artists that were suggested to my personal Spotify account. It was mainly pop, rock, and rap artists. Then, we gathered data on the artist's title, the artist's albums, the song names from the albums, as well as release dates of the songs and albums. For Billboard, we used an API to gather data from the "Hot 100" list from May of 2019, which is our personal favorite era of music. With this data, we collected data on song titles, artists, rankings, peak positions, and amount of weeks on the "Hot 100" chart. This allowed us to analyze the popularity and longevity of these songs over time. All of this fulfilled our objective of creating a comprehensive dataset and visualizations for further analysis.

3. The problems that you faced (10 points)

We faced many problems throughout this project. First was finding a second API - Spotify was very easy to use, but it took us a long time to find the second API. Many that we found did not have relevant data or had a paywall. In terms of the coding portion, we struggled significantly with ensuring that only 25 pieces of data were added into the database at a time. It was easy to do it so that the data would all be added in batches of 25 in one run, but spreading it out across different runs was more difficult. Additionally, we encountered challenges in effectively and successfully capturing and storing the Billboard data. Despite these obstacles, we eventually overcame them and successfully completed the project!

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

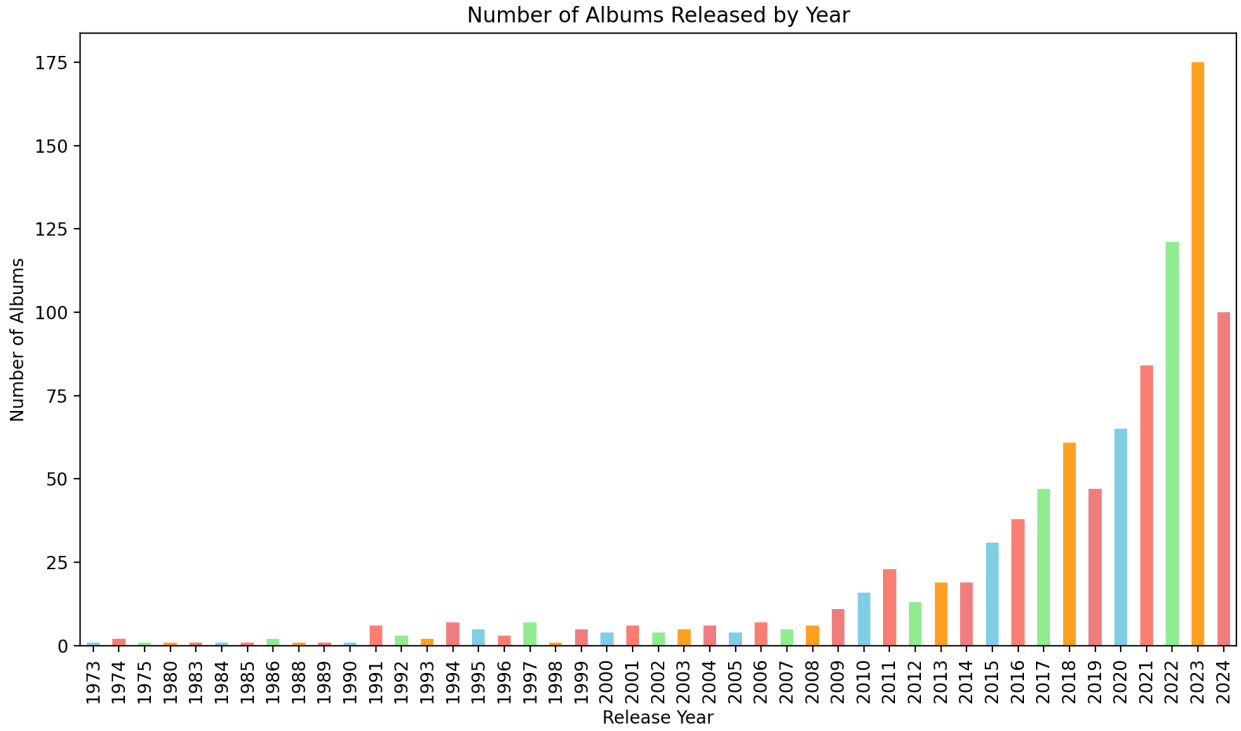
```
Average number of albums released per year: 21.533333333333335
```

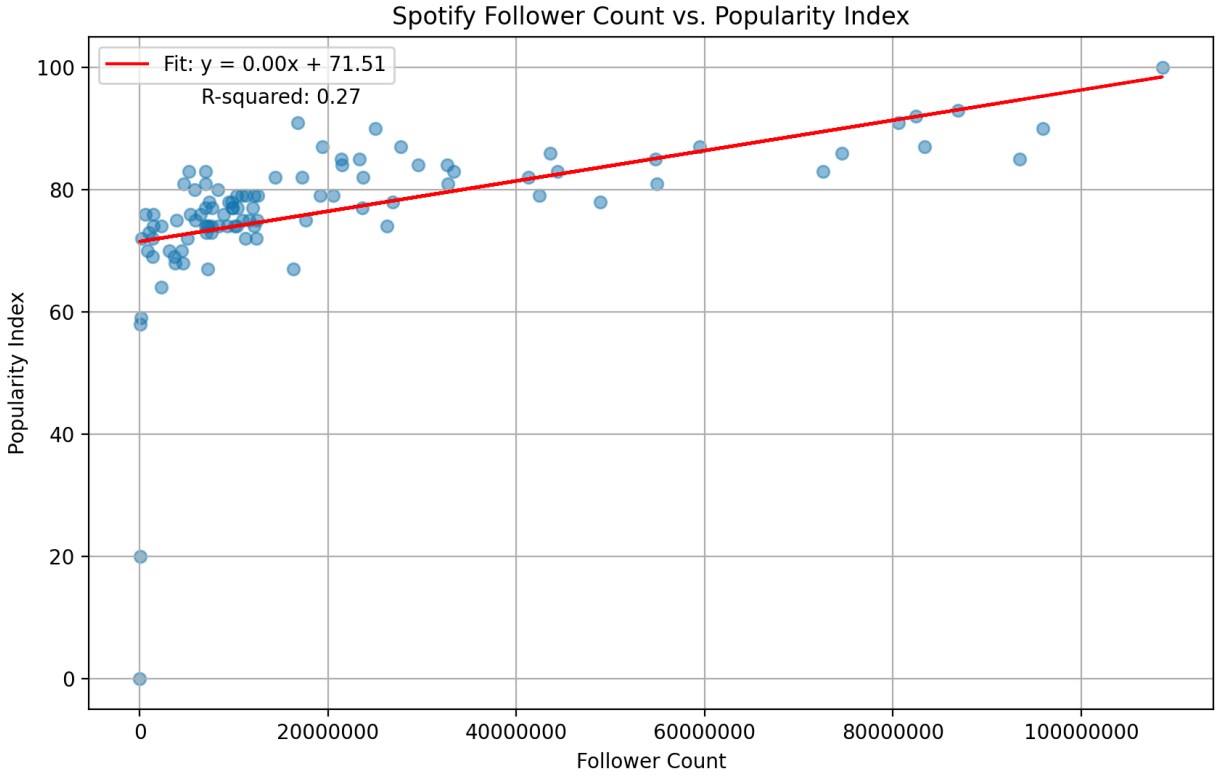
```
Artist: Billie Eilish, Songs on Chart: 6
Artist: Ariana Grande, Songs on Chart: 3
Artist: Panic! At The Disco, Songs on Chart: 2
Artist: Lil Uzi Vert, Songs on Chart: 2
Artist: Khalid, Songs on Chart: 2
Artist: Jonas Brothers, Songs on Chart: 2
Artist: benny blanco, Halsey & Khalid, Songs on Chart: 1
Artist: Yo Gotti Featuring Lil Baby, Songs on Chart: 1
Artist: YNW Melly Featuring Kanye West, Songs on Chart: 1
Artist: YNW Melly, Songs on Chart: 1
Artist: YK Osiris, Songs on Chart: 1
Artist: YG, Songs on Chart: 1
Artist: Travis Scott, Songs on Chart: 1
Artist: Thomas Rhett, Songs on Chart: 1
Artist: The Chainsmokers Featuring 5 Seconds Of Summer, Songs on Chart: 1
Artist: Taylor Swift Featuring Brendon Urie, Songs on Chart: 1
Artist: Summer Walker X Drake, Songs on Chart: 1
Artist: Ski Mask The Slump God, Songs on Chart: 1
Artist: ScHoolboy Q Featuring 21 Savage, Songs on Chart: 1
Artist: ScHoolboy Q + Travis Scott, Songs on Chart: 1
Artist: ScHoolboy Q, Songs on Chart: 1
Artist: Sam Smith & Normani, Songs on Chart: 1
Artist: SZA, The Weeknd & Travis Scott, Songs on Chart: 1
Artist: Post Malone & Swae Lee, Songs on Chart: 1
Artist: Post Malone, Songs on Chart: 1
Artist: Polo G Featuring Lil Tjay, Songs on Chart: 1
Artist: Pinkfong, Songs on Chart: 1
Artist: Pedro Capo X Farruko, Songs on Chart: 1
Artist: P!nk Featuring Chris Stapleton, Songs on Chart: 1
Artist: P!nk, Songs on Chart: 1
Artist: Ozuna x Daddy Yankee x J Balvin x Farruko x Anuel AA, Songs on Chart: 1
Artist: Old Dominion, Songs on Chart: 1
Artist: Offset Featuring Cardi B, Songs on Chart: 1
Artist: Nipsey Hussle Featuring Roddy Ricch & Hit-Boy, Songs on Chart: 1
Artist: NLE Choppa, Songs on Chart: 1
Artist: Mustard & Migos, Songs on Chart: 1
Artist: Morgan Wallen, Songs on Chart: 1
Artist: Megan Thee Stallion, Songs on Chart: 1
```

```
Artist: Lil Dicky, Songs on Chart: 1
Artist: Lil Baby & Gunna, Songs on Chart: 1
Artist: Lil Baby, Songs on Chart: 1
Artist: Lee Brice, Songs on Chart: 1
Artist: Lauv & Troye Sivan, Songs on Chart: 1
Artist: Lauren Daigle, Songs on Chart: 1
Artist: Lady Gaga & Bradley Cooper, Songs on Chart: 1
Artist: Kelsea Ballerini, Songs on Chart: 1
Artist: Kane Brown, Songs on Chart: 1
Artist: Juice WRLD, Songs on Chart: 1
Artist: Jon Pardi, Songs on Chart: 1
Artist: J. Cole, Songs on Chart: 1
Artist: Halsey, Songs on Chart: 1
Artist: Florida Georgia Line, Songs on Chart: 1
Artist: Ellie Goulding X Diplo Featuring Swae Lee, Songs on Chart: 1
Artist: Ella Mai, Songs on Chart: 1
Artist: Eli Young Band, Songs on Chart: 1
Artist: Dean Lewis, Songs on Chart: 1
Artist: Daddy Yankee & Katy Perry Featuring Snow, Songs on Chart: 1
Artist: DaBaby, Songs on Chart: 1
Artist: Cody Johnson, Songs on Chart: 1
Artist: City Girls, Songs on Chart: 1
Artist: Chase Rice, Songs on Chart: 1
Artist: Cardi B & Bruno Mars, Songs on Chart: 1
Artist: Calboy, Songs on Chart: 1
Artist: Brett Young, Songs on Chart: 1
Artist: Brett Eldredge, Songs on Chart: 1
Artist: Blueface, Songs on Chart: 1
Artist: Blake Shelton, Songs on Chart: 1
Artist: Beyonce, Songs on Chart: 1
Artist: BTS Featuring Halsey, Songs on Chart: 1
Artist: BLACKPINK, Songs on Chart: 1
Artist: Avicii Featuring Aloe Blacc, Songs on Chart: 1
Artist: Ava Max, Songs on Chart: 1
Artist: Anuel AA & Karol G, Songs on Chart: 1
Artist: A Boogie Wit da Hoodie Featuring 6ix9ine, Songs on Chart: 1
Artist: A Boogie Wit da Hoodie, Songs on Chart: 1
Artist: 5 Seconds Of Summer, Songs on Chart: 1
```

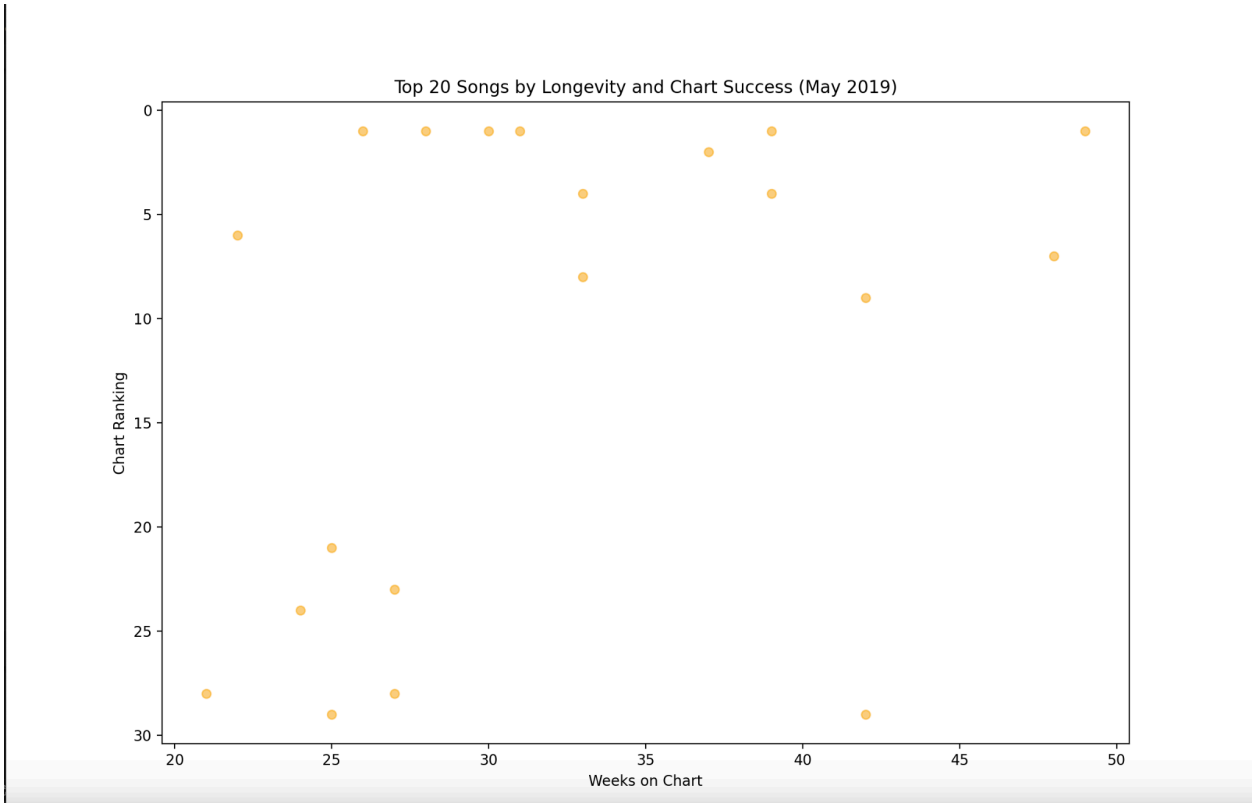
5. The visualization that you created (i.e. screen shot or image file) (10 points)

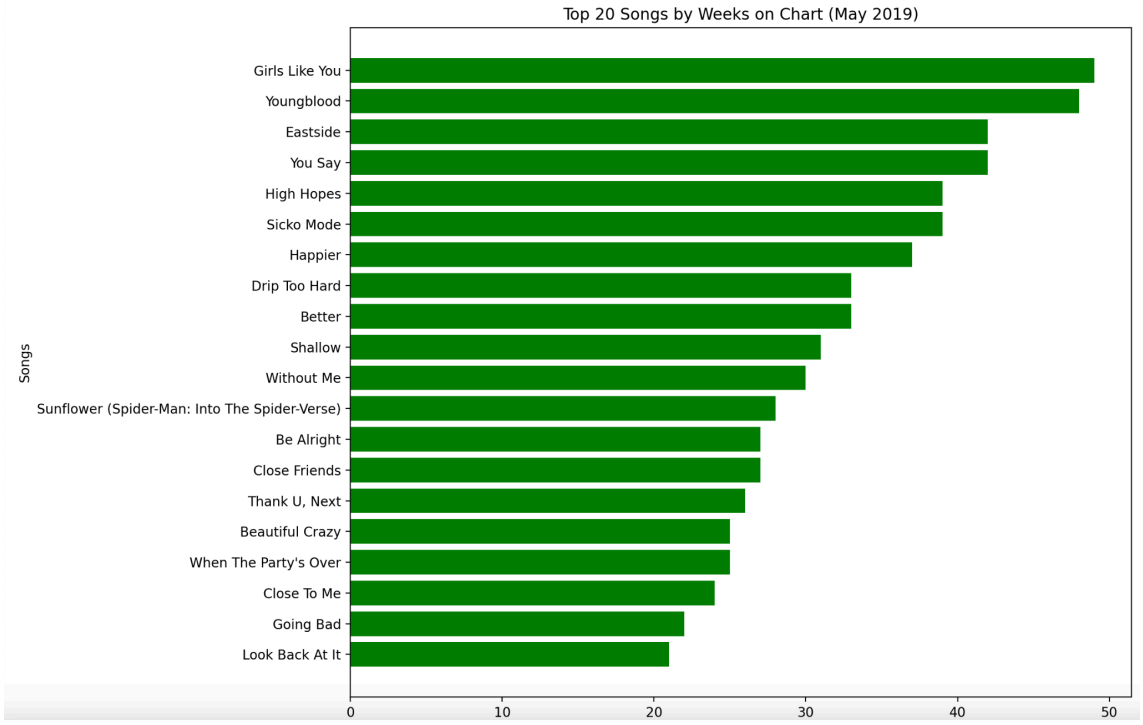
Spotify Visualizations:





Billboard Visualizations:





6. Instructions for running your code (10 points):

Step 1: run billboard.py four times to add all 100 items to the spotify.db database, view the visualizations, and to create the "database_data.txt" file
Step 2: run spotify.py four times to add all 100 items to the spotify.db database
Step 3: run plotting.py for linear regression graph
Step 4: run plottingpt2.py for albums released per year graph
Step 5: run analyze_spotify_data.py for calculation txt file to be created
Step 6: run the databasewriter.py file to write all the data and tables to a txt file

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

spotify.py:

1) get_access_token(client_id, client_secret):

Inputs: client_id (string) - The client ID for accessing the Spotify API (pulled from spotify web developer website)
client_secret (string) - The client secret for accessing the Spotify API ((pulled from spotify web developer website)
Output: access_token (string or None) - The access token obtained from the Spotify API. Returns None if an error occurs during the token retrieval process.

2) create_database(db_file):

Input: db_file (string) - The file path name as a string to the SQLite database file.
Output: None
Functionality: Creates two tables in the SQLite database (artists and albums). The artists table contains columns for artist ID, name, followers, and popularity. The albums table contains columns for album ID, artist ID, name, and release date.

3) get_artist_data(artist_ids, access_token):

Inputs:
artist_ids (list of strings - txt file) - List of artist IDs for which data is from a txt file
access_token (string) - Access token obtained from the Spotify API.
Output: artist_data (list of dictionaries) - List of dictionaries containing artist data (ID, name, followers, and popularity).

Functionality: Uses spotify API and list of artist IDs to return a list of dictionaries containing the fetched artist data.

4) get_artist_albums(artist_id, access_token):

Inputs:
artist_id (string) - ID of the artist for which albums are to be fetched, pulled from the txt file
access_token (string) - Access token obtained from the Spotify API.
Output: albums_data (list of dictionaries) - List of dictionaries containing album data (ID, name, and release date) for specified artist.
Functionality: Fetches album data for the specified artist ID from the Spotify API using a request. Returns a list of dictionaries containing the fetched album data.

5) save_data_to_database(conn, artist_data, albums_data):

Inputs:
conn (SQLite connection object) - connection object to the SQLite database
artist_data (list of dictionaries) - List of dictionaries containing artist data (ID, name, followers, and popularity)
albums_data (list of dictionaries) - List of dictionaries containing album data (artist ID, name, and release date).
Output: None
Functionality: Saves the fetched artist and album data to the SQLite database (spotify.db). Inserts the data into the artists and albums tables.

6) Main():

Input: None
Output: None
Functionality: Utilizes all previous functions. Pulls API access token, collects and adds artist and album data from Spotify API in batches of 25 until the end of the artist id list is reached. The subsequent time the code is ran, the database is reset. It also writes the number of items in the database to a txt file.

plotting.py:

1) fetch_data_from_database(conn):

Input: conn (SQLite connection object) - Connection object to the spotify.db database
Output: artists_data (Pandas DataFrame) - pandas DataFrame containing data fetched from the artists table in the database.
Functionality: Fetches data from the artists table in the SQLite database using the provided connection object. Returns the fetched data as a Pandas DataFrame to be used for graphing.

2) plot_followers_vs_popularity(artists_data):

Input: artists_data (Pandas DataFrame) - DataFrame containing artist data including followers and popularity.
Output: None
Functionality: Plots a scatter plot of follower count vs. popularity index using the provided Spotify artist data. Performs linear regression on the data to find the line of best fit and plots it on the graph. Displays the R-squared value of the regression model on the plot.

3) main():

Input: None
Output: None
Functionality: Runs the previously defined functions. Connects to the SQLite database, fetches artist data, plots follower count vs. popularity index, and closes the database connection. Launches the graph in Matplotlib.

plottingpt2.py:

1) visualize_album_data():

Input: db_file (str): The path to the SQLite database file - the spotify.db file that has album data
Output: None
Functionality: Reads spotify.db data, counts the number of albums from the database and groups by year. Displays a bar chart showing frequency distribution of albums based on years. Displays graph in Matplotlib

analyze_spotify_data.py

1) visualize_album_data(db_file):

Inputs:
db_file (string) - Path to the SQLite database file containing album information (spotify.db).
Output: None
Functionality: Reads album data from the specified SQLite database file. Calculates the average number of albums released per year and writes it as a float. Writes the average number to a separate text file named “average_albums_per_year.txt”. Output is shown in number 4.

billboard.py:

1) create_or_connect_database(db_file):

   Inputs:
   db_file (string) - The file path name as a string to the SQLite database file.
   Output: None

Functionality: Connects to the specified SQLite database file. If it does not exist, it creates the database. It also ensures that the ‘songs’ and ‘artists’ tables are created within the database if they do not already exist.

2) fetch_billboard_data(limit, offset):
   Inputs:
   limit (integer) - The number of records to fetch from the Billboard Hot 100 API.
   offset (integer) - The starting position in the dataset from which to begin data retrieval.
   Output: data (list of tuples) - List of tuples containing data for each song fetched from the API. Returns an empty list if an error occurs during data fetching.
   Functionality: Fetches song data from the Billboard Hot 100 API based on the specified limit and offset. Each tuple in the returned list contains the rank, title, artist, last week's rank, peak position, and weeks on the chart for each song.

3) save_data_to_database(conn, data):
   Inputs:
   conn (SQLite connection object) - Connection object to the SQLite database.
   data (list of tuples) - List of tuples containing the song data to be saved.
   Output: None
   Functionality: Saves the fetched song data into the ‘songs’ and ‘artists’ tables in the SQLite database. If an artist or song already exists, it prevents duplicate entries by ignoring the insertion.

4) query_data_and_write_to_file()
   Inputs: None
   Output: None
   Functionality: This function queries the SQLite database to fetch and count the total number of songs for each artist in the database. It establishes a connection to the SQLite database file named ‘spotify.db’. A SQL query is executed to aggregate the song counts for each artist and the results are ordered by the count of songs in descending order. The function writes these results to a text file called ‘artist_song_counts.txt’. Each line in this file lists an artist's name followed by the number of songs they have on the chart, providing a straightforward report on artist popularity based on the number of charted songs. Finally, the function ensures that the database connection is properly closed after completing the operation.

5) visualize_data(db_file):
   Inputs:
   db_file (string) - The file path name as a string to the SQLite database file.
   Output: None
   Functionality: Generates visualizations for the songs stored in the database. It produces a scatter plot showing the relationship between weeks on chart and peak position and a horizontal bar chart showing the top 20 songs by weeks on chart.

6) reset_state():
   Inputs: None
   Output: None
   Functionality: Deletes the ‘state.json’ file and the SQLite database file if they exist. This function is used to reset the application state to start the data fetching and processing from scratch.

7) load_state():
   Inputs: None
   Output:
   state (dict) - A dictionary containing keys ‘total_records’ and ‘offset’, which track how many records have been fetched and the current offset for fetching, respectively.
   Functionality: Loads the current state of the data fetching process from the ‘state.json’ file. Returns a dictionary with default values if the file does not exist.

8) save_state(state):
   Inputs:
   state (dict) - A dictionary containing the current state of the fetching process, specifically ‘total_records’ and ‘offset’.
   Output: None
   Functionality: Saves the current state of the data fetching process to the ‘state.json’ file.

9) main():
   Inputs: None
   Output: None
   Functionality: Orchestrates the complete process of fetching, saving, and visualizing Billboard chart data. Manages application state, handles database connections, initiates data fetching, saving, visualization, and state management routines.

Database_writer.py

1) export_database_data
   Inputs: db_file - path to sql database file
   Output_file: the txt file that the database data and tables will be written to
   Functionality: The function connects to a SQLite database (spotify.db) and retrieves data from all the tables. It then writes it to a text file with the table names as headers.

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	Did it solve the issue?
4/13	Did not know how to find client id and client secret.	Spotify for Web Developers desktop	Yes
4/22	Did not understand how to use spotify API and what its capabilities were.	Spotify Web API Reference Website	Yes
4/22	Needed to find another API that was free and provided relevant data.	Rapidapi.com - searched in the Music APIs tab	Yes
4/23	Needed to figure out how to add only a specific number of items to the database after each run.	Stack Overflow & ChatGPT	Yes
4/23	Needed to figure out	Stack Overflow	Yes

	how to create tables with a shared integer key.		
4/25	Needed to figure out how to perform the calculations	ChatGPT	Yes
4/26	Realized that all tables should be in one database	Stack Overflow, Reddit, ChatGPT	Yes

Thank you to Dr. Ericson and the whole instructional team for an amazing semester!