Isabelle Fisher, Phoebe Miller, Shreya Mani
SI 206 - Final Project

Link to Final Project Github Repo: https://github.com/shreyamani02/FinalProjectSI.git

1. The goals for your project (10 points)

Our main goal was to create a database of Taylor Swift's entire discography and analyze it based on the different albums. We wanted to use the Spotify API to create a master list of songs and albums, and the Youtube API to obtain which songs have music videos and data about those music videos. We also wanted to scrape a list of awards that Taylor Swift has been nominated for, and data about awards she won.

2. The goals that were achieved (10 points)

- We were able to use the Spotify API, Youtube API, and Wikipedia's list of awards to obtain the data we wanted and analyze it accordingly. We navigated through these APIs to analyze features including song name, album name, energy, danceability, etc. to comprehensively understand and analyze the features of Taylor Swift songs.

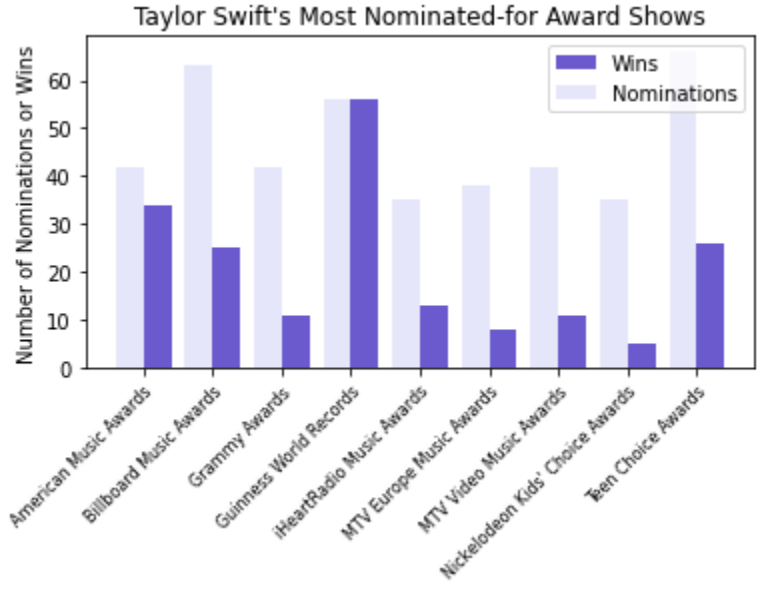
3. The problems that you faced (10 points)

- We unfortunately did not have authorization to access all the data that we wanted to from the Youtube API. The data we were able to obtain was not incredibly useful for analysis, as there were music videos for every song in our database, and the published dates referenced the last time the video was updated rather than when it was originally published.
- Additionally, we struggled with utilizing github to work on separate computers.

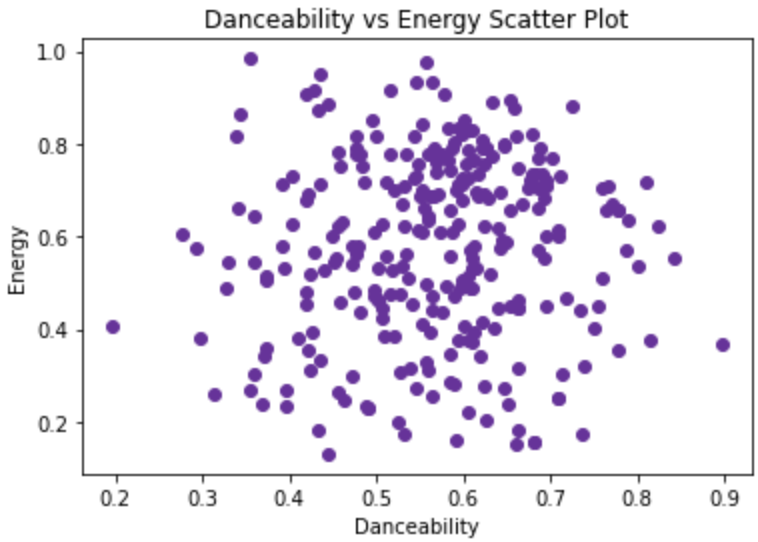
4. Your file that contains the calculations from the data in the database (10 points)

- Attached in github under "calculations.txt"

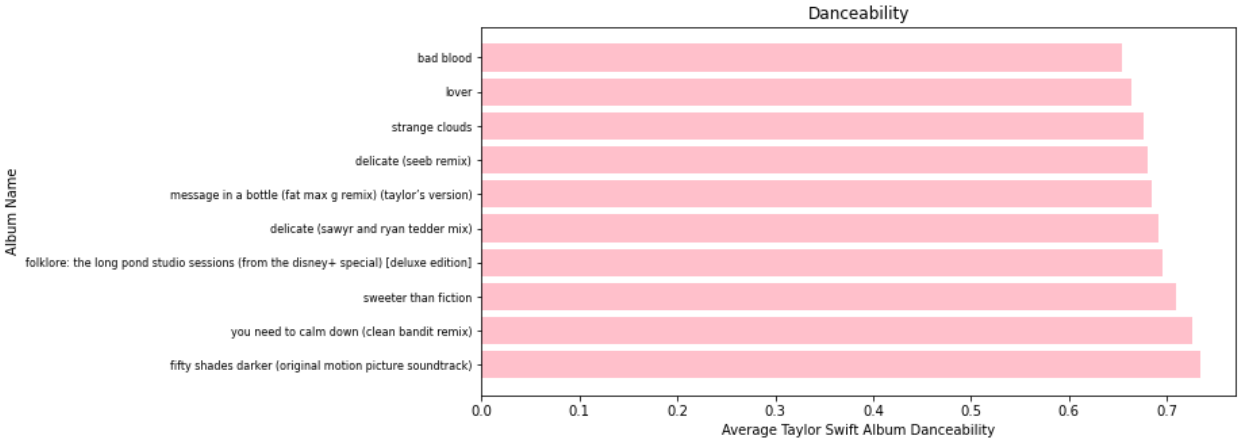
5. The visualization that you created (i.e. screenshot or image file) (10 points)



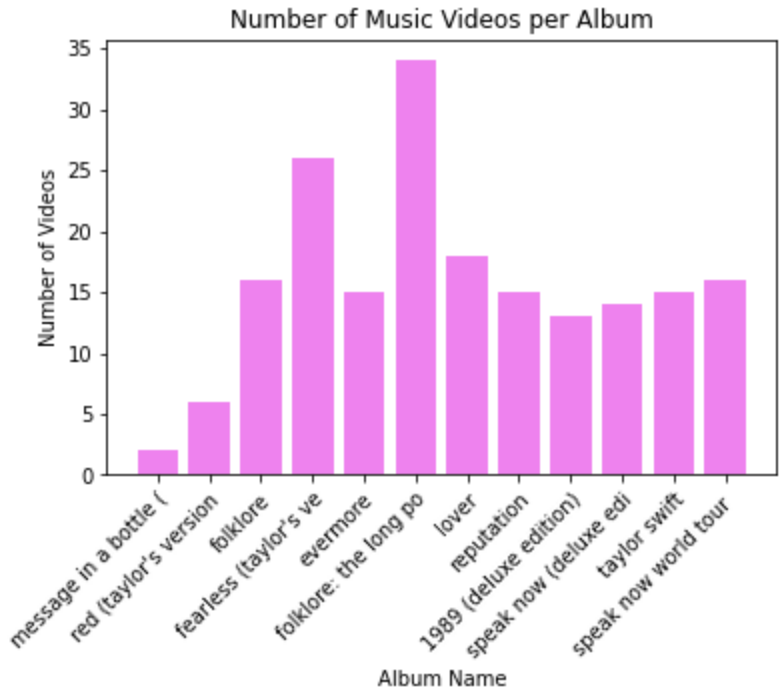
-


Danceability vs Energy Scatter Plot
Energy
Danceability
1.0
0.8
0.6
0.4
0.2
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

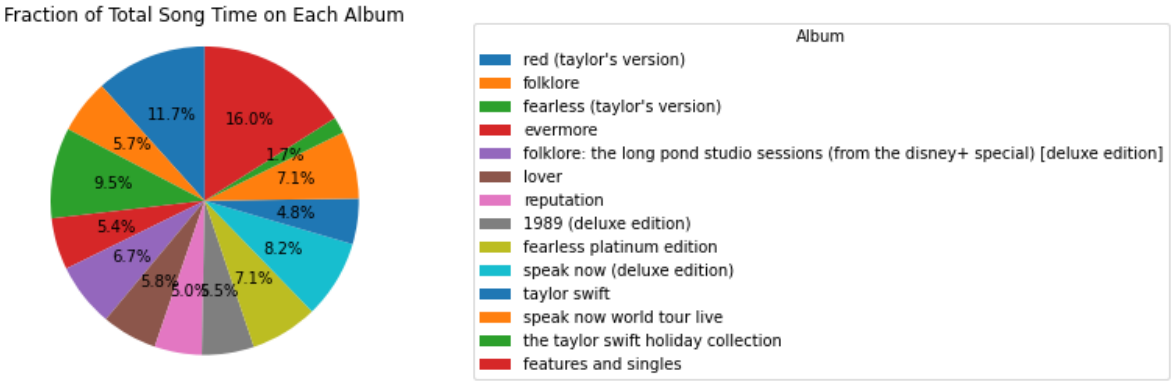
-


Danceability
Album Name
bad blood
lover
strange clouds
delicate (seeb remix)
message in a bottle (fat max g remix) (taylor's version)
delicate (sawyr and ryan tedder mix)
folklore: the long pond studio sessions (from the disney+ special) [deluxe edition]
sweeter than fiction
you need to calm down (clean bandit remix)
fifty shades darker (original motion picture soundtrack)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Average Taylor Swift Album Danceability

-



-



-

6. Instructions for running your code (10 points)

- The database is named in main. To test our code by creating a new database, you can change the name of the .db file. The code must be restarted 6 times to fully fill our limited database and complete the other calculations and create the charts. There is no other user input necessary for the code to run.

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

- setUpDatabase(db_name)
    - This function creates the database specified by the input name, and returns the cursor and connection to edit the database
- createTables(cur, conn)

    - This function creates the Songs, Albums, Music_Videos, and Awards tables that will be filled in later. It inputs cur and conn, and it outputs nothing.

- scrapeWiki(soup, cur, conn)
    - This function inputs a beautiful soup object and database cursor and connector. It scrapes the table of awards on Taylor Swift's Wikipedia page, and adds the information it gleans less than 25 items at a time, exiting the program each time. There are 126 awards, so to completely add all data to the database the program needs to be run 5 times. It returns nothing.
- spotifyAPI()
    - This function does not input anything. It uses spotipy to scrape a spotify playlist of all Taylor Swift songs. The Spotify API limits requests to 100 songs, so to get a complete list of data an offset method was used. The dictionary that is returned from the API is parsed to create a list of song ids (used by spotify to identify the songs), and a list of each unique album in the playlist. It returns the list of song ids, and the list of albums.
- updateSpotifyData(cur, conn, ids, album_list)
    - This function inputs the database cursor and connection, and the lists of song ids and albums. It then uses the album list to fill in the album table on the database. The second part of the code uses the list of song ids to fetch the names, length, popularity, danceability, and energy of the song, as well as the album id from the album table. It returns nothing.
- youtubeAPI(cur, conn)
    - This function inputs the database cursor and connection, and uses a dictionary of Taylor Swift Music Video Playlists and their ids to identify each music video's name, what album it belongs to, and what date it was last edited, and add this information to the Music Video table.
- avg_winsnoms_ratio(cur, conn)
    - This function inputs in the database cursor and connection, and accesses the database to compute and return the average ratio of wins and nominations for every award that she's been nominated for.
- avg_length_album(album, cur, conn)

    - This function inputs an album and database information. It accesses the database, collects the album id and all the songs for that id, and finds the average length of a song on that album. It returns this average length.

- most_popular_album(cur, conn)
    - The function inputs in the database cursor and connection and joins both the songs and albums table to calculate which album is the most popular. The function takes the sum of popularity for every song off the album and computes the average popularity based on the number of songs off the album.

- album_time(album, cur, conn)
    - This function takes in the database cursor, connection, and the name of a specific album. Based on the album name, the function reads the length of every song on the album and computes the sum of all the lengths. It returns the total length of the album. This function was used to make a pie chart.

- danceable_album(cur, conn)
    - This function takes in the database information, and accesses the database to determine the average danceability score of each album. It then graphs this data, and returns the average danceability of all albums.

- videos_per_album(album, cur, conn)
    - This function inputs the database information and an album name. The album name is used to access the album id and that is used to determine how many music videos correspond to that album. It returns an integer of music videos for that album.

- how_hard_was_taylors_yt_team_working_that_day(cur, conn)
    - This function inputs the database information and creates a dictionary of all days that the Taylor Swift Youtube Channel published a music video and the number of music videos published that day.

- awards_chart(cur, conn)
    - This function prints out a bar graph that displays (top 10) the number of wins and nominations that Taylor has received at various award shows. The bars for nominations and wins are displayed in different colors.

- pie_chart_album_lengths(cur, conn)

    - This function creates a pie chart that displays the total length of each album as a fraction of the total time of Taylor's discography. It inputs cur and conn, and returns nothing.

- energyvsdanceability_plot(cur, conn)
    - This function takes in a database cursor and connection and reads in the energy and danceability for every song in the database. It then plots the data on a scatterplot.
- write_calculations(filename, cur, conn)
    - This function takes in database information and the name of .txt file. It opens this file and runs the avg_length_album, most_pop_album, avg_winsnoms_ratio, and videos_per_album function. It writes the results of these functions to the .txt file and closes the file. It returns nothing.
- video_bar_graphs(cur, conn)
    - This function inputs the database information and creates a matplotlib bar graph of the number of music videos for each album. It returns nothing.

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
4/21	We weren't able to find the exact information for the number of streams or how streams change over time.	https://medium.com/@boplantinga/what-do-spotifys-audio-features-tell-us-about-this-year-s-eurovision-song-contest-66ad188e112a#:~:text=Spotify%20Audio%20Analysis&text=Danceability%3A%20Danceability%20describes%20how%20suitable,and%201.0%20is%20most%20danceable	We changed our project to focus on the audio features of each song, which we could pull from spotify.
4/23	Accessing Youtube API data without proper authorization for the Taylor Swift channel	https://developers.google.com/youtube/v3	We ended up pulling the name of the music video and date posted and doing our calculations on that.

4/23	We had an excessive amount of merge errors when trying to collaborate on github	https://docs.github.com/en/desktop/contributing-and-collaborating-using-github-desktop/making-changes-in-a-branch/managing-branches	We made branches on github in which to save our commits without creating extreme merge errors
4/25	Scatterplot	https://matplotlib.org/stable/gallery/shapes_and_collections/scatter.html	Used to make energy vs danceability scatter plot
4/26	Dividing a list of tuples by second element	https://stackoverflow.com/questions/8092877/split-a-list-of-tuples-into-sub-lists-of-the-same-tuple-field	Used to split data read for danceability graph by album.
4/23	Data from spotify api using artist search was incomplete	Spotify api documentation - https://betterprogramming.pub/how-to-extract-any-artists-data-using-spotify-s-api-python-and-spotipy-4c079401bc37	Changed method- used playlist search instead
4/25	Data from spotify limited to 100 tracks, needed to write loop and offset	Spotify api documentation - https://towardsdatascience.com/extracting-song-data-from-the-spotify-api-using-python-b1e79388d50	Offset works! Data set is complete
4/24	Pie Chart	https://matplotlib.org/3.5.0/api/_as_gen/matplotlib.pyplot.pie.html?highlight=pie#matplotlib.pyplot.pie	I made a pie chart!