TriHarmony - SI 201 Final Project Report

Link to GitHub Repository: https://github.com/taylorcarm/si-201-final-project.git

1. The goals for your project including what APIs/websites you planned to work with and what data you planned to gather (5 points)

The overall goal for this project was to understand how different musical attributes, such as valence (happiness), explicit lyrical content, popularity rank, genre, release information, and audio features, are connected and influence one another. By integrating data from multiple music-related APIs, we aimed to explore relationships between a song's emotional characteristics, its explicitness, and its popularity across platforms. Additionally, the project sought to examine how contextual factors such as genre, country of release, and commercial pricing relate to listener preferences and market trends. Through these analyses, we aimed to gain a more holistic understanding of how musical content, audience reception, and commercial value interact within the modern music ecosystem.
The APIs our team used are listed below:

- *Deezer*
    - From Deezer, our team planned to gather track rank (popularity metrics), explicit lyrics, track titles, and artist names
    - This collection would allow us to analyze explicitness and popularity trends for various tracks
- *Last.fm*
    - From Last.fm, our team planned to gather track names, artist names, genres, and track durations
    - This API would act as our primary data source by providing a broad collection of top tracks across 12 musical genres
- *MusicBrainz*
    - From MusicBrainz, our team planned to gather release data and country (where the track was originally released)
    - This data would allow for temporal and geographic analysis of the music dataset
- *Itunes*
    - From the iTunes Search API, our team planned to gather track names, artist names, primary genres, and track pricing information. This API was used as an additional data source to analyze how music pricing varies across different genres. By collecting iTunes track prices, we were able to calculate the average price of songs within each genre and compare pricing trends across genres. The iTunes data also provided a useful contrast to streaming-focused APIs by offering insight into the commercial pricing side of music consumption.

2. The goals that were achieved including what APIs/websites you actually worked with and what data you did gather (5 points)

The goals that were achieved through this project was that our team gained a better understanding of how different music attributes, such as ranking or song duration, connect to each other and influence other variables. Through our work with four APIs, Deezer, Last.fm, MusicBrainz, and Itunes, our team was able to complete several calculations and create many visualizations that demonstrate the overall connection between music's many attributes and the correlation between them.

- *Deezer* gathered data on track popularity and content ratings.
    - Rank: Popularity score for each track, which indicates how popular the song is on Deezer's platform
    - Explicit Lyrics: This is a boolean indicator, stored as an integer 0 (not-explicit) or 1 (explicit), that shows whether or not a track contains explicit content
    - Track Title: This is a verification of track names from Deezer's database
    - Artist Name: Verification of artist names from Deezer's database
- *Last.fm* gathered data on top tracks across multiple music genres.
    - Track Name: Title of each song
    - Artist Name: Performing artist for each track
    - Genre: Musical genre classification (hip-hop, rock, rnb, pop, jazz, country, metal, electronic, indie, classical, house, and punk)
    - Duration: Length of each track in seconds
- *MusicBrainz* gathered data on track release information and geographic distribution
    - Release Data: Official release date of each recording (when track was first released)
    - Country: Country where track was originally released
- iTunes API
    - Used the iTunes Search API to collect 199 music tracks. The data was stored in a new SQLite table called itunes_tracks, which includes track name, artist, genre, and price. Our team then calculated the average track price by genre using SQL aggregation. The results show that most genres are priced around $1.29, while some genres such as Soundtrack and Christmas music have lower average prices. The calculated results were written to a text file (itunes_analysis.txt). This analysis was intended to identify pricing patterns across musical genres and compare how different genres are valued in the iTunes marketplace. There are no duplicate tracks in the iTunes dataset when considering the combination of track name and artist.

3. The problems that you faced (5 points)

Our team struggled with separating the databases into separate tables to remove duplicate string data due to the high amount of duplicate data there was and the rather time-consuming process of ensuring that all duplicates were put in their individual tables. We also had a hard time trying to get the Spotify API to work because the songs used a specific Spotify Track ID, and couldn't be found just based on the song title and artist. Moreover, for datasets such as the iTunes API, we defined uniqueness using a combination of track name and artist name, which allowed us to prevent duplicate tracks from being inserted into the database while still preserving valid cases where different artists may have songs with the same title. We also used SQL features such as UNIQUE constraints and INSERT OR IGNORE statements to automatically handle duplicates during data collection, reducing manual data cleaning and ensuring data integrity across runs. This approach allowed us to maintain a clean dataset while efficiently integrating data from multiple APIs.

4. The calculations from the data in the database (i.e. a screen shot) (5 points)

Spotify Stats Calculation

Text File Output:

spotify_stats.txt

```
=== avg_energy_by_genre ===
classical: 0.0297
country: 0.70825
electronic: 0.7473636363636363
hip-hop: 0.7966666666666667
house: 0.876
indie: 0.7093333333333334
jazz: 0.3488230769230769
metal: 0.9103333333333333
pop: 0.761
punk: 0.8844666666666666
rnb: 0.5412857142857143
rock: 0.8316111111111111

=== avg_danceability_by_genre ===
classical: 0.231
country: 0.6045
electronic: 0.648
hip-hop: 0.7003333333333334
house: 0.666
indie: 0.5397333333333333
jazz: 0.5489999999999999
metal: 0.45049999999999996
pop: 0.7183076923076922
punk: 0.4726
rnb: 0.6238571428571429
rock: 0.4925555555555555

=== avg_valence_by_genre ===
classical: 0.07
country: 0.6457499999999999
electronic: 0.5406363636363637
hip-hop: 0.6546666666666666
house: 0.552
indie: 0.39620000000000005
jazz: 0.5292307692307692
metal: 0.37128333333333335
pop: 0.5474615384615384
punk: 0.6348
rnb: 0.2951428571428571
rock: 0.4432055555555556
```

Average Danceability by Genre Calculation

Text File Output:

TEXTavg_danceability_by_genre.txt > data

```
genre   danceability
classical   0.231
country 0.6045
electronic  0.648
hip-hop 0.7003333333333334
house   0.666
indie   0.5397333333333333
jazz    0.5489999999999999
metal   0.45049999999999996
pop 0.7183076923076922
punk    0.4726
rnb 0.6238571428571429
rock    0.4925555555555555
```

Average Deezer Rank Calculation

Text File Output:

TEXTavg_deezer_rank.txt > data

```
genre   rank
classical   383638.0
country 561889.24
punk    608581.96
jazz    608693.4
house   682667.8333333334
hip-hop 741249.52
indie   749270.76
metal   765564.92
pop 800437.76
rnb 806507.0
electronic  819031.84
rock    856071.6
```

Average Energy by Genre Calculation

Text File Output:

TEXTavg_energy_by_genre.txt > data

```
genre   energy
classical   0.0297
country 0.70825
electronic  0.7473636363636363
hip-hop 0.7966666666666667
house   0.876
indie   0.7093333333333334
jazz    0.3488230769230769
metal   0.9103333333333333
pop 0.761
punk    0.8844666666666666
rnb 0.5412857142857143
rock    0.8316111111111111
```

Average Tempo by Genre Calculation

Text File Output:

TEXTavg_tempo_by_genre.txt > data

```
genre   tempo
indie   109.9602
jazz    113.71069230769231
metal   115.88199999999999
pop 119.3773076923077
rnb 123.06857142857143
country 123.92025
electronic  124.52645454545454
rock    128.92855555555556
house   129.884
classical   131.413
punk    134.8126
hip-hop 144.18866666666665
```

Average Valence by Genre Calculation

Text File Output:

TEXTavg_valence_by_genre.txt > data

```
genre   valence
classical   0.07
country 0.6457499999999999
electronic  0.5406363636363637
hip-hop 0.6546666666666666
house   0.552
indie   0.39620000000000005
jazz    0.5292307692307692
metal   0.37128333333333335
pop 0.5474615384615384
punk    0.6348
rnb 0.2951428571428571
rock    0.4432055555555556
```

Tracks Per Genre Per Country Calculation

Text File Output:

TEXTtracks_per_genre_per_country.txt

```
country hip-hop
AU  1
GB  2
US  3
XE  1
XW  2
```

Yearly Tempo Calculation

Text File Output:

TEXTyearly_tempo.txt > data

	year	tempo
1	1994.0	95.244
2	1999.0	174.322
3	2001.0	97.79466666666667
4	2002.0	144.07
5	2003.0	82.3005
6	2004.0	131.9375
7	2005.0	114.35875
8	2006.0	123.64066666666668
9	2007.0	139.195
10	2008.0	145.06
11	2009.0	109.73949999999999
12	2010.0	111.446
13	2011.0	114.2795
14	2012.0	129.0191111111111
15	2013.0	114.27133333333335
16	2014.0	111.277
17	2015.0	106.018
18	2016.0	169.977
19	2017.0	115.5665
20	2018.0	120.423
21	2019.0	109.7675
22	2021.0	135.0485
23	2022.0	106.966
24	2024.0	135.095
25	2025.0	138.559

Yearly Valence Calculation

Text File Output:

TEXTyearly_valence.txt > data

	year	valence
1	1994.0	0.829
2	1999.0	0.598
3	2001.0	0.54
4	2002.0	0.3568
5	2003.0	0.4985
6	2004.0	0.636
7	2005.0	0.68575
8	2006.0	0.4692333333333334
9	2007.0	0.7715
10	2008.0	0.31775
11	2009.0	0.4815
12	2010.0	0.4771
13	2011.0	0.31074999999999997
14	2012.0	0.44633333333333336
15	2013.0	0.35700000000000004
16	2014.0	0.503
17	2015.0	0.103
18	2016.0	0.648
19	2017.0	0.667
20	2018.0	0.578
21	2019.0	0.6134999999999999
22	2021.0	0.3025
23	2022.0	0.412
24	2024.0	0.594
25	2025.0	0.772

Average Valence by Explicit Calculation

Text File Output:

TEXTaverage_valence_by_explicit.txt

```
explicit_lyrics valence
0   0.4995905263157895
1   0.4635909090909091
```

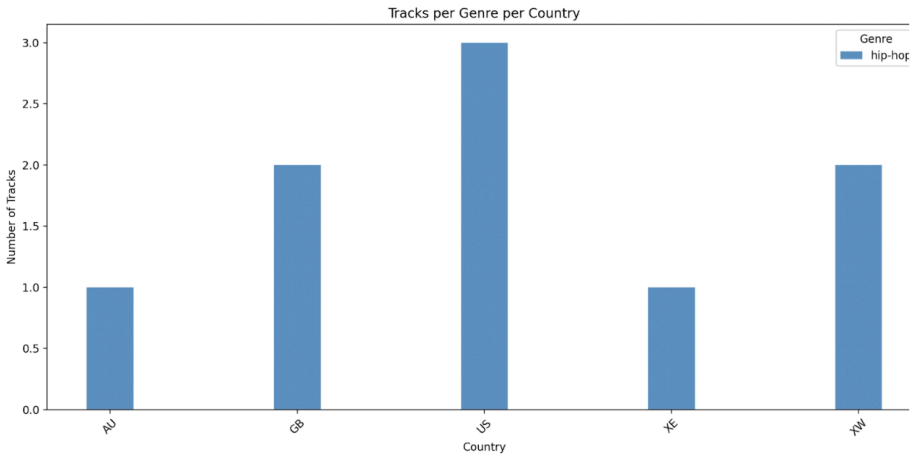
Itunes API Calculation

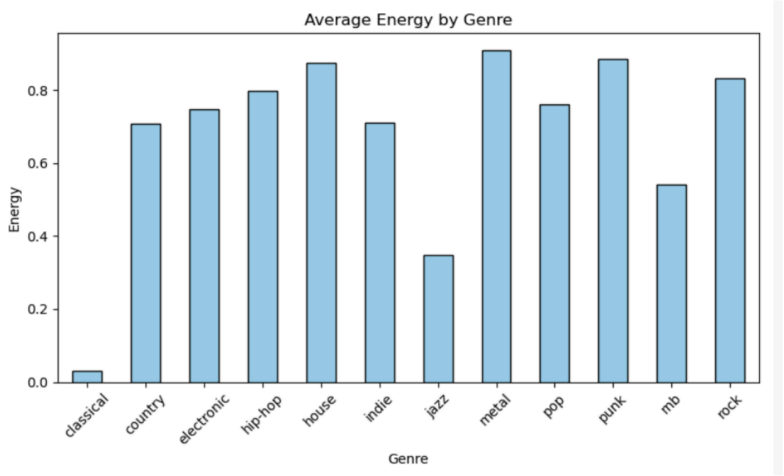
Text File Output:

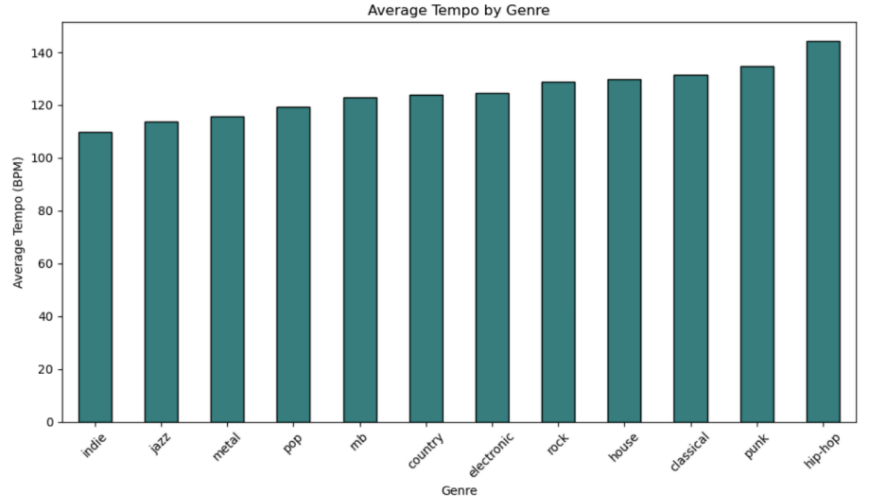
itunes_analysis.txt

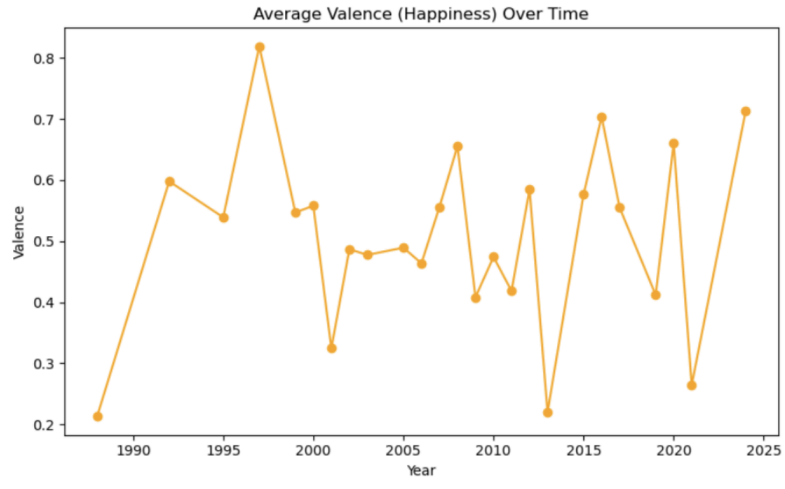
```
Average iTunes Track Price by Genre
==================================

Alternative: $1.29
Urbano latino: $1.29
Rock: $1.29
Rap: $1.29
R&B/Soul: $1.29
Pop: $1.29
New Age: $1.29
Musicals: $1.29
Metal: $1.29
K-Pop: $1.29
Indie Rock: $1.29
Holiday: $1.29
Hip-Hop/Rap: $1.29
Hard Rock: $1.29
Christian: $1.29
Country: $1.26
Dance: $1.19
Soundtrack: $1.14
Alte: $0.99
Christmas: Pop: $0.69
```

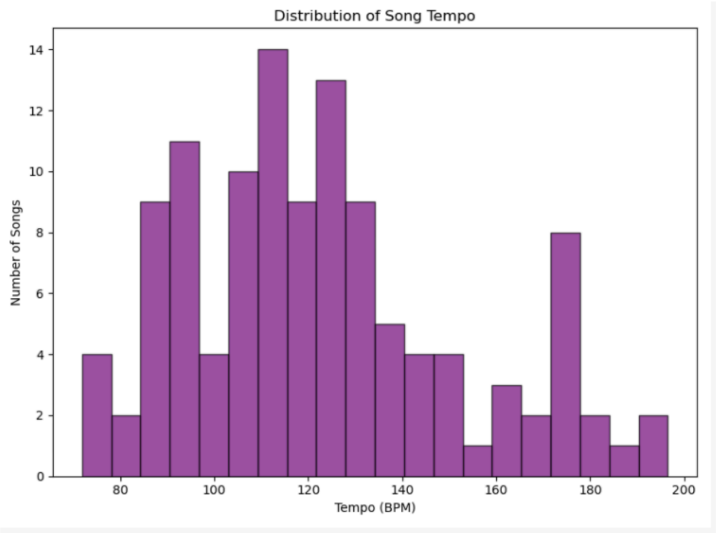
5. The visualization that you created (i.e. screen shot or image file) (5 points)

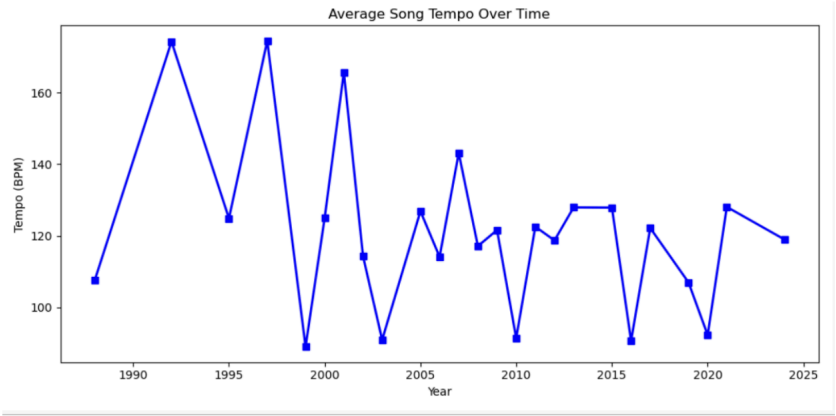


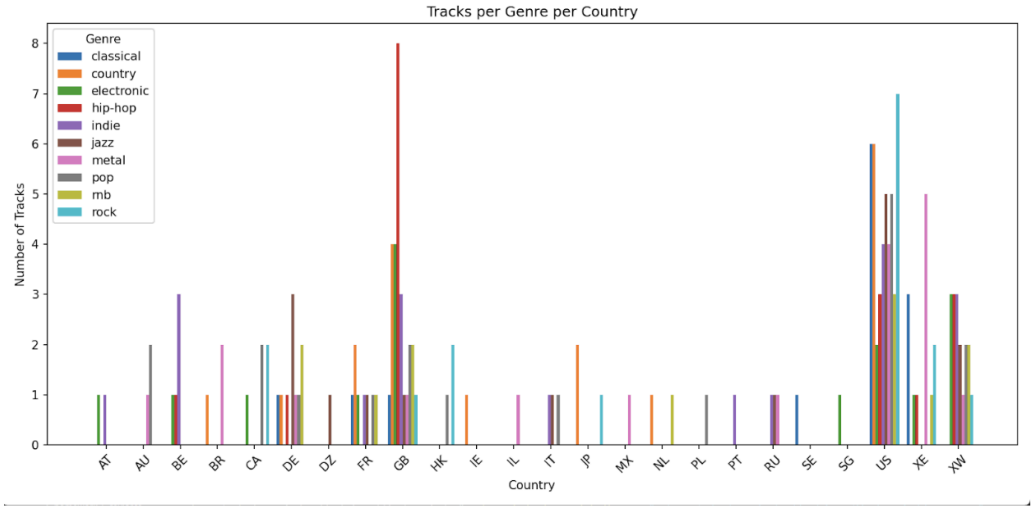

Average Energy by Genre
Energy
0.0
0.2
0.4
0.6
0.8
classical
country
electronic
hip-hop
house
indie
jazz
metal
pop
punk
rnb
rock
Genre

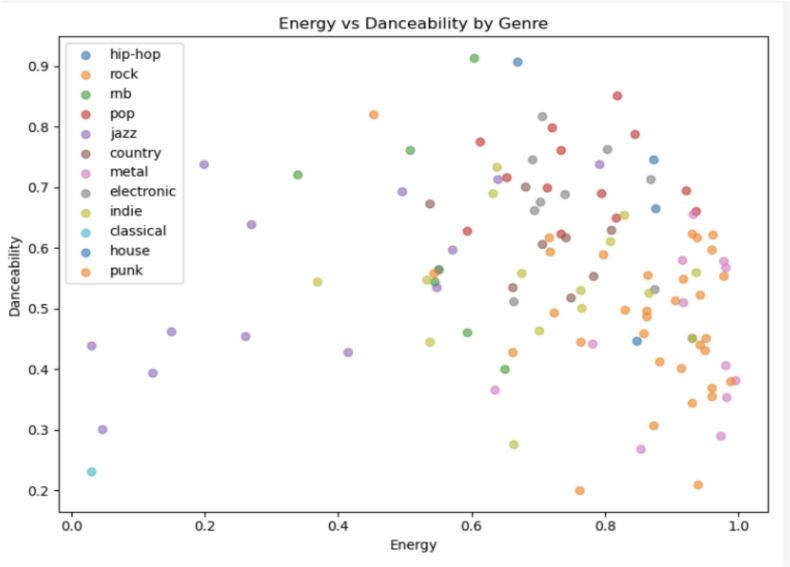

Average Tempo by Genre
Average Tempo (BPM)
0
20
40
60
80
100
120
140
indie
jazz
metal
pop
rnb
country
electronic
rock
house
classical
punk
hip-hop
Genre

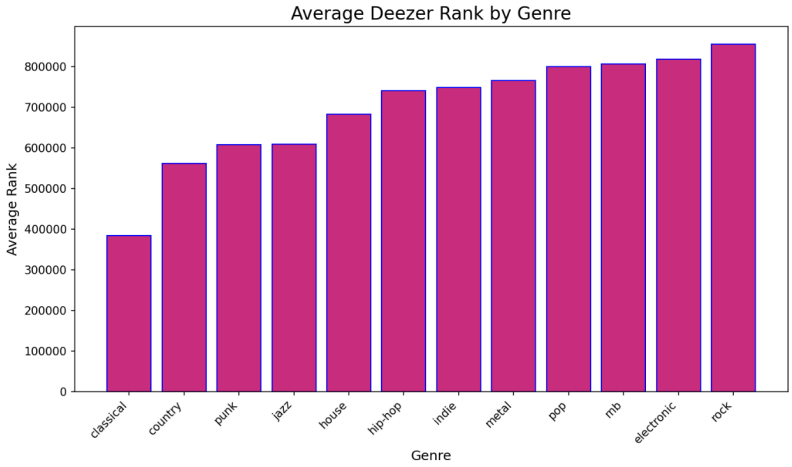

Average Valence (Happiness) Over Time
Valence
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1990
1995
2000
2005
2010
2015
2020
2025
Year

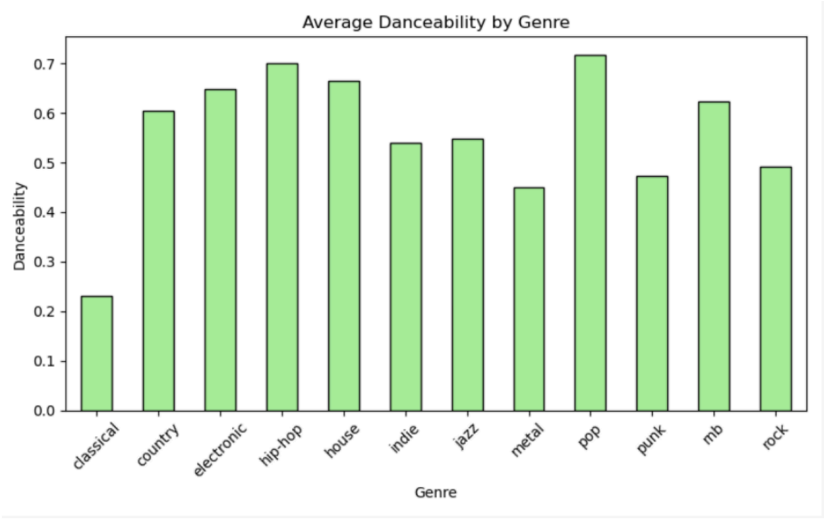

Distribution of Song Tempo
Number of Songs
Tempo (BPM)
0
2
4
6
8
10
12
14
80
100
120
140
160
180
200


Average Song Tempo Over Time
Tempo (BPM)
Year
100
120
140
160
1990
1995
2000
2005
2010
2015
2020
2025


Tracks per Genre per Country
Genre
classical
country
electronic
hip-hop
indie
jazz
metal
pop
rnb
rock
Number of Tracks
Country
0
1
2
3
4
5
6
7
8
AT
AU
BE
BR
CA
DE
DZ
FR
GB
HK
IE
IL
IT
JP
MX
NL
PL
PT
RU
SE
SG
US
XE
XW


Energy vs Danceability by Genre
hip-hop
rock
rnb
pop
jazz
country
metal
electronic
indie
classical
house
punk
Danceability
Energy
0.0
0.2
0.4
0.6
0.8
1.0
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


Average Deezer Rank by Genre
Average Rank
Genre
0
100000
200000
300000
400000
500000
600000
700000
800000
classical
country
punk
jazz
house
hip-hop
indie
metal
pop
rnb
electronic
rock


Average Danceability by Genre
Danceability
Genre
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
classical
country
electronic
hip-hop
house
indie
jazz
metal
pop
punk
rnb
rock

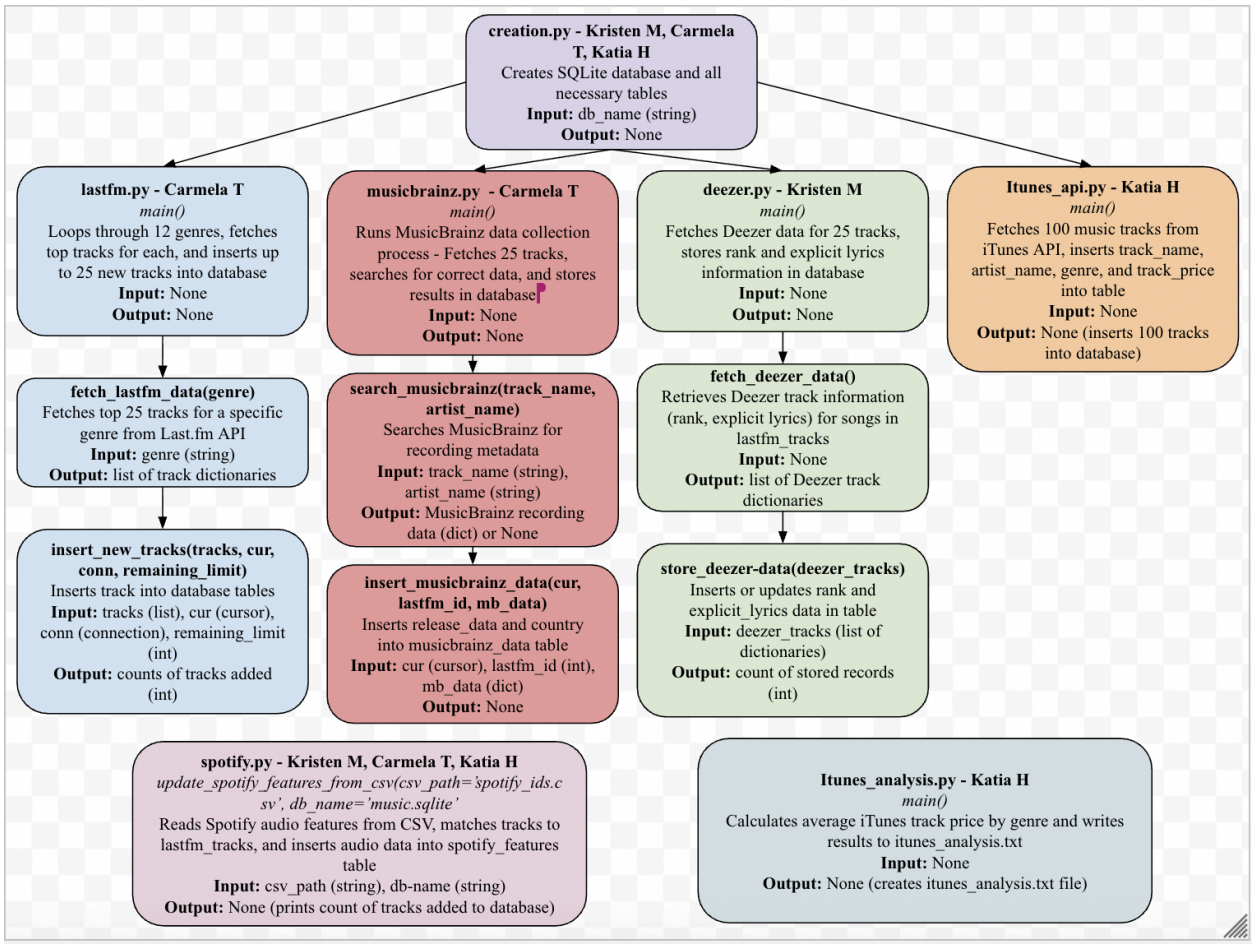
6. Instructions for running your code (5 points)

a. **Step 1:** create the database. Run creation.py once first- This creates the music.sqlite database file and sets up all necessary tables (tracks, artists, genres, lastfm_tracks, deezer_data, spotify_features, musicbrainz_data, lyrics_data, itunes_tracks)
b. **Step 2:** Collect Last.fm data. Run lastfm.py 12 times - Each run adds up to 25 new tracks to the database. Running it 12 times adds approximately 300 tracks to the database. Code cycles through 12 genres (hip-hop, rock, rnb, pop, jazz, country, metal, electronic, indie, classical, house, punk) and collects top tracks for each.
c. **Step 3:** Run deezer.py 12 times - Each run adds Deezer popularity and explicit lyrics data to 25 Last.fm tracks. Need 12 runs to all the Deezer data, around 300 tracks.
d. **Step 4:** Run musicbrainz.py 12 times - Adds release dates and country information to the 300 Last.fm tracks (25 tracks per run)
e. **Step 5:** Download the spotify csv from: https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset and make sure it is named 'spotify_ids' and in the same directory as the spotify.py script
f. **Step 6:** Run spotify.py once - Imports Spotify audio features from CSV file and matches them to existing lastfm_tracks
g. **Step 7:** Run itunes_api.py once to get 199 tracks - Collects tracks from iTunes API with pricing information
    i. (This is our extra credit API. The iTunes API script collects 199 tracks in a single run, rather than limiting to 25 tracks per execution. The project instructions specify the 25-item limit for the required APIs to ensure multiple runs and database inserts, but this limit applies to the three core APIs. Since the iTunes API was added as an extra credit source, collecting all tracks at once does not conflict with the instructions, and all tracks are uniquely stored in the database)
    ii. Step 7: Run itunes_analysis.py once - Calculates and saves average iTunes track prices by genre to a text file (EXTRA CREDIT)
h. **Step 8:** You can now view the complete database. After all data collection scripts have been run, execute the following calculation and analysis scripts to generate the joins, calculations, and visualizations presented in this report:
    - person.a.py
    - analysis.py
    - valence_by_explicit.py
    - musicbrainzcalcs.py

    These are some of the calculations mentioned above:
    i. **Average Danceability by Genre Calculation**
    ii. **Average Energy by Genre Calculation**
    iii. **Average Tempo by Genre Calculation**
    iv. **Average Valence by Genre Calculation**

v. **Average Deezer Rank per Genre Calculation**
vi. **Tracks Per Genre Per Country Calculation**
vii. **Yearly Tempo Calculation**
viii. **Yearly Valence Calculation**
ix. **Spotify Stats Calculation**

7. An updated function diagram with the names of each function, the input, and output and who was responsible for that function (10 points)



8. You must also clearly document all resources you used. The documentation should be of the following form (10 points)

Date	Issue Description	Location of Resource	Result (did it solve the issue?

12/7/25	Struggled to make one database with several tables rather than multiple databases	https://stackoverflow.com/questions/63972638/how-to-create-multiple-tables-in-sql	Yes
12/7/25	Trouble accessing spotify API	https://developer.spotify.com/documentation/web-api/concepts/spotify-uris-ids	No
12/7/25	Problems with inserting Last.fm tracks; first it wouldn't do 25 at a time and then it would only repeat one genre	ChatGPT	Yes - received suggestions for using insert or ignore and ways to cycle through the genres without manually changing the genre each run
12/9/25	The Deezer table wasn't matching the songs from the last fm table correctly, and we think we were getting bad requests	ChatGPT	Yes - debugged the way the Deezer code was getting data for specific songs and added breaks for when we exceed 25 songs or the request is bad
12/9/25	We could not get the spotify API to work	https://www.reddit.com/r/learnprogramming/comments/v890f0/how_to_work_with_spotify_api/	No
12/9/25	Since the spotify API didn't work, we found a kaggle dataset with similar enough data (though it didn't cover all the songs)	https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset	Yes
12/9/25	Needed help grabbing the spotify data for only existing song records and removing duplicates since the	ChatGPT	AI guidance helped explain how to query the database for track IDs, filter the values, and insert it and

	same song was listed multiple times		provided examples
12/10/25	Help navigating the MusicBrainz API to search for and insert the data; specifically help formatting the query with headers and the user agent that MusicBrainz required	whatismybrowser.com/detect/what-is-my-user-agent/faq/what-is-a-user-agent and chatgpt	Yes - we looked up info on a user agent and got suggestions/debugging help for our code
12/10/25	Understanding group by for our calculations	https://www.geeksforgeeks.org/pandas/python-pandas-dataframe-groupby/	Yes
12/10/25	Writing our calculation results to a text file easily using the to_csv function	https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_csv.html	Yes
12/10/25	Reshaping and reorganizing our dataset to support multi-dimensional calculations and visualizations, including creating pivot tables for stacked bar charts and grouped summaries.	https://www.geeksforgeeks.org/python/python-pandas-pivot_table/ & chatgpt	Yes - helped reshape the data for a calculation and for a stacked bar chart
12/10/25	Make plot look better	https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.tight_layout.html	Yes
12/15/25	Extra credit API and removing duplicate string data in it	https://hightouch.com/sql-dictionary/sql-foreign-key & chatgpt	Yes - was able to make it so the genre and artist used a numeric key instead of strings

12/7/25 - 12/15/25	General debugging and advice when stuck	chatgpt	Sometimes; was very helpful in interpreting/diagnosing errors/problems

Updates After Grading Session:

We received feedback to fix the duplicate string data in our tables. We originally had one table per API, so there were several instances of duplicate string data. We fixed this issue and this is what our tables look like now:

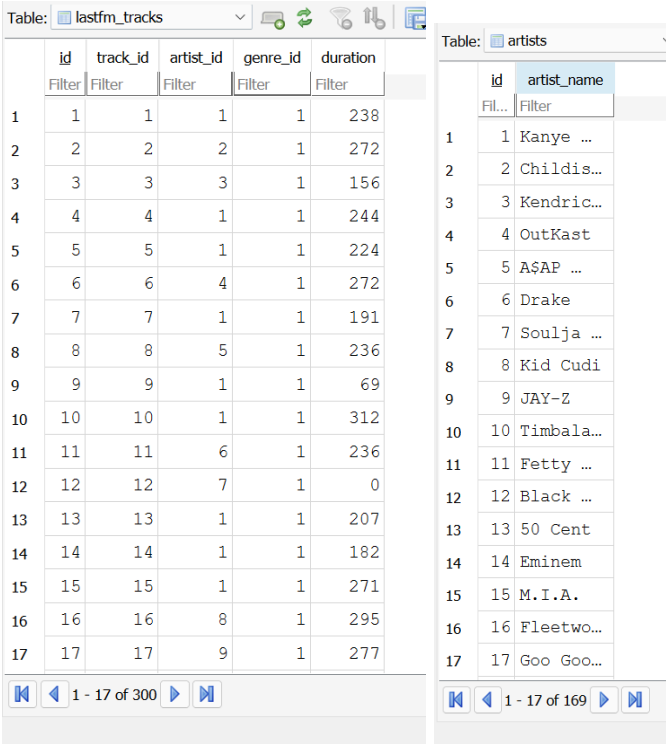


Table: lastfm_tracks

	id	track_id	artist_id	genre_id	duration
	Filter	Filter	Filter	Filter	Filter
1	1	1	1	1	238
2	2	2	2	1	272
3	3	3	3	1	156
4	4	4	1	1	244
5	5	5	1	1	224
6	6	6	4	1	272
7	7	7	1	1	191
8	8	8	5	1	236
9	9	9	1	1	69
10	10	10	1	1	312
11	11	11	6	1	236
12	12	12	7	1	0
13	13	13	1	1	207
14	14	14	1	1	182
15	15	15	1	1	271
16	16	16	8	1	295
17	17	17	9	1	277

1 - 17 of 300

Table: artists

	id	artist_name
	Fil...	Filter
1	1	Kanye …
2	2	Childis…
3	3	Kendric…
4	4	OutKast
5	5	A$AP …
6	6	Drake
7	7	Soulja …
8	8	Kid Cudi
9	9	JAY-Z
10	10	Timbala…
11	11	Fetty …
12	12	Black …
13	13	50 Cent
14	14	Eminem
15	15	M.I.A.
16	16	Fleetwo…
17	17	Goo Goo…

1 - 17 of 169

Table: tracks

	id	track_name
	Filter	Filter
1	1	Flashing Lights
2	2	Heartbeat
3	3	Swimming Pools (Drank)
4	4	I Wonder
5	5	All Falls Down
6	6	Ms. Jackson
7	7	Touch the Sky
8	8	Fashion Killa
9	9	Homecoming
10	10	Stronger
11	11	Headlines
12	12	Kiss Me Thru The Phone
13	13	Gold Digger
14	14	Can't Tell Me Nothing
15	15	Runaway
16	16	Pursuit of Happiness (Nightmare)
17	17	Empire State of Mind

1 - 17 of 292 Go

Table: genres

	id	genre_name
	Fil...	Filter
1	1	hip-hop
2	2	rock
3	3	rnb
4	4	pop
5	5	jazz
6	6	country
7	7	metal
8	8	electron...
9	9	indie
10	10	classical
11	11	house
12	12	punk

1 - 12 of 12

Table: deezer_data

	id	lastfm_id	rank	explicit_lyrics
	Filter	Filter	Filter	Filter
1	1	1	871367	1
2	2	2	782927	0
3	3	3	826262	1
4	4	4	726617	1
5	5	5	708788	1
6	6	6	867799	1
7	7	7	756635	1
8	8	8	681929	1
9	9	9	731834	1
10	10	10	615697	1
11	11	11	800639	1
12	12	12	742776	0
13	13	13	775001	1
14	14	14	774129	1
15	15	15	842009	1
16	16	16	417109	1
17	17	17	946707	1

1 - 17 of 299

Table: musicbrainz_data

	id	*lastfm_id*	release_date	country
	Fil...	Filter	Filter	Filter
1	1	1	2011-04-17	XW
2	2	2	*NULL*	*NULL*
3	3	3	*NULL*	*NULL*
4	4	4	*NULL*	*NULL*
5	5	5	*NULL*	*NULL*
6	6	6	2008	BR
7	7	7	2009-10-20	US
8	8	8	2013	DE
9	9	9	2011-04-17	XW
10	10	10	*NULL*	*NULL*
11	11	11	2011-11-14	GB
12	12	12	2009-12-13	AU
13	13	13	2005-08-29	GB
14	14	14	2007-11-27	CA
15	15	15	2010-11-01	GB
16	16	23	2006	US
17	17	24	2010-11-22	XW

1 - 17 of 162

Table: spotify_features

	lastfm_track_id	spotify_track_id	danceability	energy	valence	tempo
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	7Lh28afkQh...	0.533	0.808	0.274	140...
2	2	68Q6c22JNy...	0.811	0.647	0.708	120...
3	4	7bXWSg5gd0...	0.765	0.522	0.391	100...
4	5	1hnSI4JaTQ...	0.699	0.189	0.328	129...
5	9	2tH5sjtfoA...	0.302	0.382	0....	123...
6	10	3CrsuTJomW...	0.563	0.678	0.134	127...
7	11	1iXuVEywIU...	0.706	0.765	0.27	87....
8	15	6sFjQ6Gb5J...	0.81	0.924	0.872	118...
9	17	7pYVIpuFrH...	0.508	0.883	0.717	173...
10	19	3oiWRTSsBf...	0.462	0.659	0.358	186...
11	20	2d8JP84HNL...	0.746	0.873	0.817	148...
12	23	3HOcPWc2LU...	0.664	0.301	0.268	115...
13	25	1ixbwbeBi5...	0.447	0.848	0.485	172...
14	27	5oV8bZaShr...	0.497	0.13	0.179	130...
15	28	3ObS6mDRBs...	0.271	0....	0....	73....
16	30	5UWwZ5lm5P...	0.413	0.881	0.364	158...
17	32	6geCWidV8I...	0.494	0.723	0.152	144...

1 - 17 of 183 Go to: 1

*We also added a 4th API for extra credit. It gave us 199 rows and this is the calculation from that API:

```
Average iTunes Track Price by Genre
===================================

Urbano latino: $1.29
Rock: $1.29
Rap: $1.29
R&B/Soul: $1.29
Pop: $1.29
New Age: $1.29
M♦sica Mexicana: $1.29
Musicals: $1.29
Metal: $1.29
Latin: $1.29
K-Pop: $1.29
Indie Rock: $1.29
Hip-Hop/Rap: $1.29
Hard Rock: $1.29
Christian: $1.29
Alternative: $1.29
Country: $1.27
Dance: $1.23
Soundtrack: $1.14
Holiday: $0.99
Alte: $0.99
Christmas: Pop: $0.69
Children's Music: $0.69
```

Table: itunes_tracks

	id	track_name	artist_id	genre_id	track_price
	Filter	Filter	Filter	Filter	Filter
1	1	Happier	16	2	0.99
2	2	How It's Done	17	3	1.29
3	3	Jingle Bell Rock	18	4	0.69
4	4	Location	19	5	1.29
5	5	Clean White Noise - Loopable with no…	20	6	1.29
6	6	Young Dumb & Broke	19	5	1.29
7	7	Music	21	7	1.29
8	8	Fast Car	22	8	1.29
9	9	Your Idol	23	3	1.29
10	10	Surface Pressure	24	9	1.29
11	11	Sunflower (Spider-Man: Into the …	25	10	1.29
12	12	Oklahoma Smokeshow	26	8	1.29
13	13	Ain't No Love In Oklahoma	22	8	1.29
14	14	All I Want for Christmas Is You	27	11	0.69
15	15	She Got the Best of Me	22	8	1.29
16	16	Forever After All	22	8	1.29
17	17	You Make It Easy	28	8	1.29

1 - 17 of 199 Go to: 1