MUSIC LISTENERS BEHAVIOR

Pre COVID - COVID

RESEARCH QUESTIONS

In this data analysis project, my focus lies in understanding the pervasive influence of late-capitalism on the multifaceted aspects of contemporary society. Specifically, I aim to explore the 'aesthetic of convergence,' which centers around the idea of cultural homogenization. To this end, I have gathered a comprehensive dataset from Spotify, incorporating various features to facilitate a nuanced examination of the pertinent research questions:

"What will be the formula to create a hit song?"
"How long should an average track last according to today's standards?"
"Music listening behaviors pre-COVID vs after-COVID"

Through this exploration, I aim to contribute to a deeper comprehension of the interplay between artistic expression, commercial interests, and societal trends in the context of the modern music industry.

GUIDE TO READ THE DATASET

Instrumentalness: This value represents the amount of vocals in the song. The closer it is to 1.0, the more instrumental the song is.

Acousticness: This value describes how acoustic a song is. A score of 1.0 means the song is most likely to be an acoustic one.

Liveness: This value describes the probability that the song was recorded with a live audience. According to the official documentation “a value above 0.8 provides strong likelihood that the track is live”.

Speechiness: “Speechiness detects the presence of spoken words in a track”. If the speechiness of a song is above 0.66, it is probably made of spoken words, a score between 0.33 and 0.66 is a song that may contain both music and words, and a score below 0.33 means the song does not have any speech.

Energy: “(energy) represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy”.

Danceability: “Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable”.

Valence: “A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry)”.

Loudness: pretty self-explanatory.

Energy: Represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.

Tempo: The pulse (BPM) of a song.

DATA PREPARATION

Prepare Essential Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import spotipy
import requests
import time

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from spotipy.oauth2 import SpotifyClientCredentials
from requests.adapters import HTTPAdapter
from requests.packages.urllib3.util.retry import Retry
```

Prepare The Dataset

```
cid = "dac99349fb7a4fab87d1357e361e0012"
secret = "4a22925e0b2a494d9017d4bc9d98c7f7"
client_credentials_manager = SpotifyClientCredentials(client_id=cid, client_secret=secret)
sp = spotipy.Spotify(client_credentials_manager = client_credentials_manager)
```

```
pre_2020_music_url = "https://open.spotify.com/playlist/4WrehQQEAcGedInZybGhcL?si=59254a07e05c4d39"
after_2020_music_url= "https://open.spotify.com/playlist/2HakbK9b5uA1RlwozL3odE?si=de2431be54e44616"
```

```
df = pd.read_csv("dataset.csv")
```

LET’S HAVE A LOOK AT THE FIRST DATASET

General Insights About the Data

```
1 df.head(10)
```

	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	...	loudness
0	0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy	73	230666	False	0.676	0.4610	...	-6.746
1	1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55	149610	False	0.420	0.1660	...	-17.235
2	2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	57	210826	False	0.438	0.3590	...	-9.734
3	3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love	71	201933	False	0.266	0.0596	...	-18.515
4	4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On	82	198853	False	0.618	0.4430	...	-9.681
5	5	01MVOl9KtVTNfFiBU9I7dc	Tyrone Wells	Days I Will Remember	Days I Will Remember	58	214240	False	0.688	0.4810	...	-8.807
6	6	6Vc5wAMmXdKIAM7WUoEb7N	A Great Big World;Christina Aguilera	Is There Anybody Out There?	Say Something	74	229400	False	0.407	0.1470	...	-8.822
7	7	1EzrEOXmMH3G43AXT1y7pA	Jason Mraz	We Sing. We Dance. We Steal Things.	I'm Yours	80	242946	False	0.703	0.4440	...	-9.331
8	8	0lktbUcnAGrvD03AWnz3Q8	Jason Mraz;Colbie	We Sing. We Dance. We	Lucky	74	189613	False	0.625	0.4140	...	-8.700

SPLIT THE DATASETS INTO DIFFERENT SECTIONS

```python
# Setting up the functions for visualizations

class DataSplit():

    def __init__(self, df):
        self.df = df
    def numericalSplit(self):
        num = self.df.select_dtypes(include = ['int64'])
        countable = []
        uncountable = []
        for c in num.columns:
            if num[c].nunique() < 100:
                countable.append(c)
            else:
                uncountable.append(c)
        return countable, uncountable
    def continousSplit(self):
        flt = self.df.select_dtypes(include = ['float64'])
        return list(flt.columns)
    def categoricalSplit(self):
        category = self.df.select_dtypes(include = ['bool', 'object'])
        return list(category.columns)
```

```python
# Splitting the dataset into different types:
ds = DataSplit(df)
continous = ds.continousSplit()
category = ds.categoricalSplit()
countable, uncountable = ds.numericalSplit()
```

- Continuous variable: Continuous values
- Category variable: Categorical values
- Countable, Uncountable: Discrete Values

TRACK_GENRES ANALYSIS

```
1 df['track_genre'].value_counts().sort_values(ascending = False)
```

```
mandopop    1000
samba       1000
mpb          999
cantopop     999
tango        999
            ...
rock         717
country      698
dance        692
soul         675
jazz         538
Name: track_genre, Length: 114, dtype: int64
```

Too many track_genres. We need a proper way to see the trend here

```
#Too many genres so we need a cluster of genres to display it properly:

df_genres = df[['danceability', 'energy',
       'key', 'loudness', 'mode', 'speechiness', 'acousticness',
       'instrumentalness', 'liveness', 'valence', 'tempo']]

scaler = StandardScaler()
scaled_features  = scaler.fit_transform(df_genres)
wcss = []

for k in range(1, 9):
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(scaled_features)
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 9), wcss)
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('WCSS')
plt.show()

#We can see that k=3 shows the optimal result
```

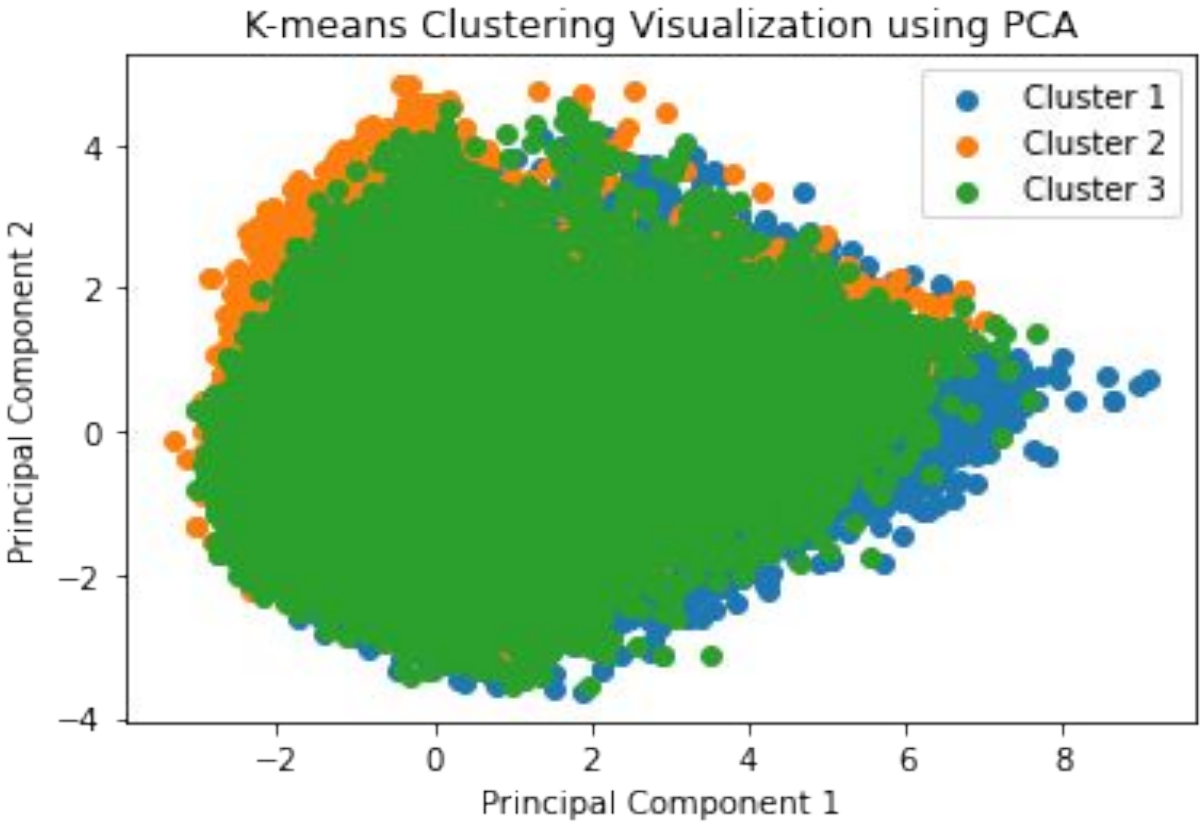
As not much insights extracted from track_genre feature, we will first use several features of a song to cluster it into different types of songs, to have a much more insightful look at the trend here!

```
kmeans = KMeans(n_clusters = 3, random_state = 0)
kmeans.fit(df_genres)
df['Clusters'] = kmeans.labels_
```

```
pca = PCA(n_components=2)
pca_features = pca.fit_transform(scaled_features)
```

```
for i in range(3):
    plt.scatter(pca_features[kmeans.labels_ == i, 0], pca_features[kmeans.labels_ == i, 1], label=f'Cluster {i+1}')

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('K-means Clustering Visualization using PCA')
plt.legend()
plt.show()
```

To make it more visually appealing, we will use Kmeans and PCA to reduce the dimensionality of the data

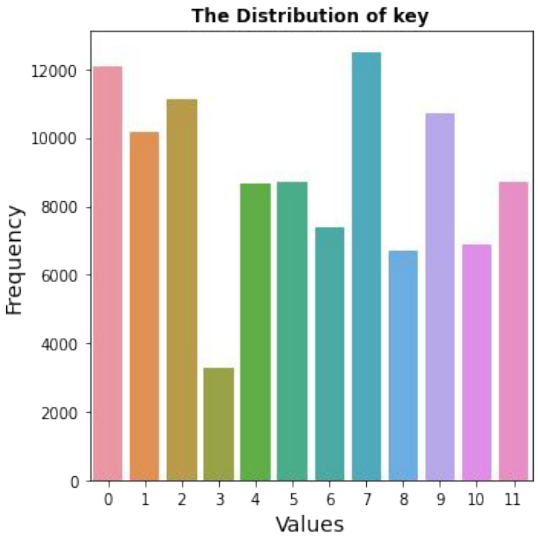

K-means Clustering Visualization using PCA
Cluster 1
Cluster 2
Cluster 3
Principal Component 2
Principal Component 1
4
2
0
−2
−4
−2
0
2
4
6
8

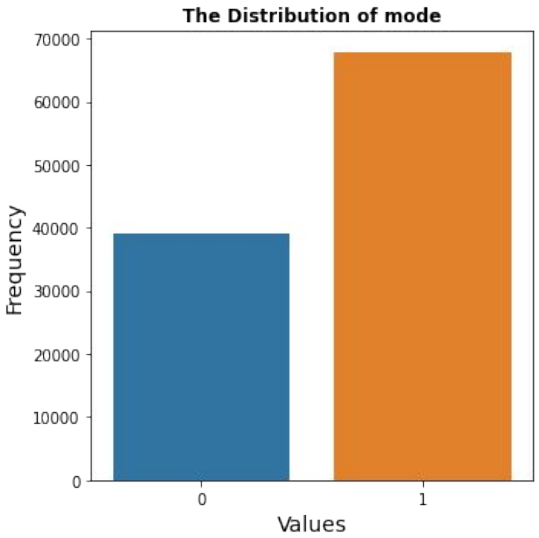
Clusters	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness
0	35.341866	221714.964770	0.097507	0.552126	0.579940	5.281682	-9.264354	0.640693	0.092690	0.396431	0.157579	0.219314
1	35.306507	224544.488143	0.091821	0.497613	0.723653	5.302119	-6.881931	0.640230	0.100437	0.239791	0.128546	0.222180
2	35.394359	238817.023416	0.072384	0.605030	0.655632	5.321041	-8.189822	0.626299	0.073254	0.282701	0.183142	0.209685

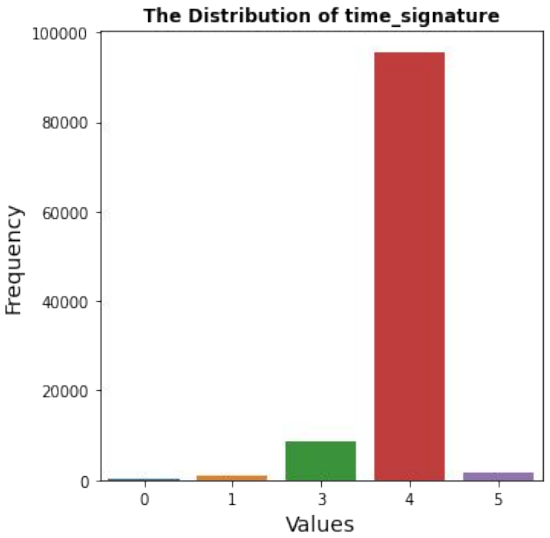
We can distinguish these track genres into three distinct clusters based on their discernible features:

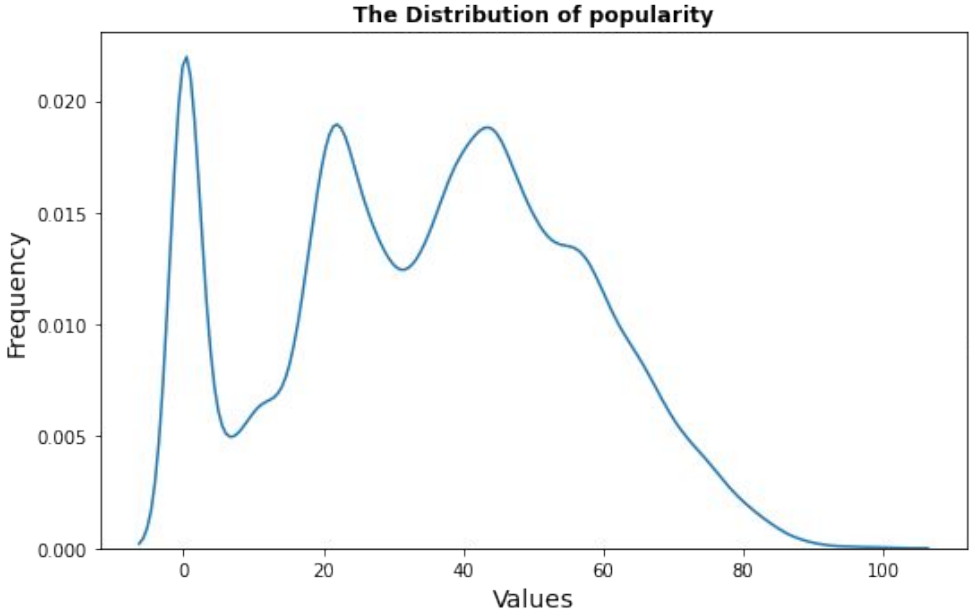
- The first cluster (Cluster 0) comprises music characterized by relatively lower levels of energy, loudness, and tempo.
- In contrast, the second cluster (Cluster 1) is constituted by genres demonstrating high tempo, liveness, energy, and loudness.
- Finally, the third cluster (Cluster 2) encompasses music genres exhibiting an intermediate tempo level but notable danceability.

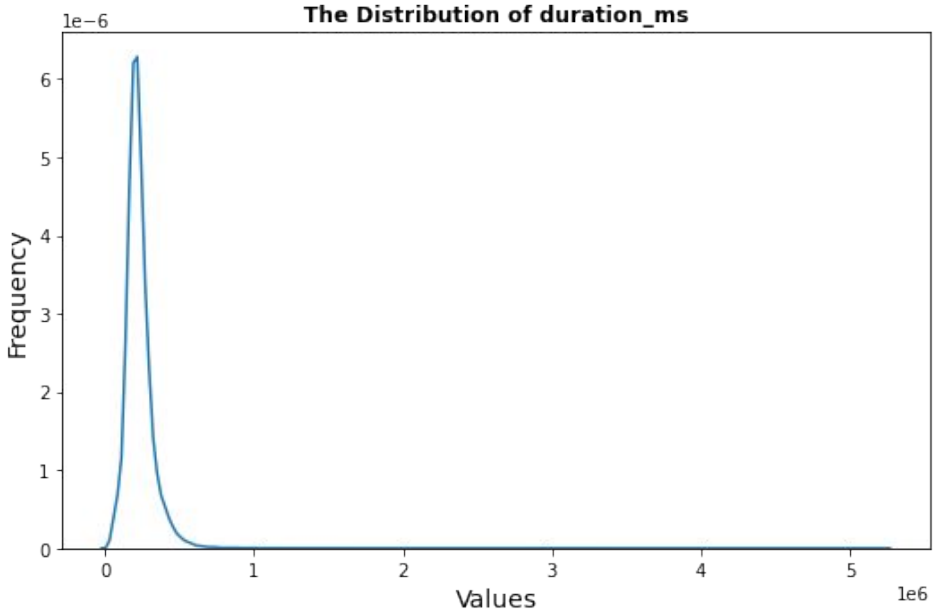
UNIVARIATE ANALYSIS

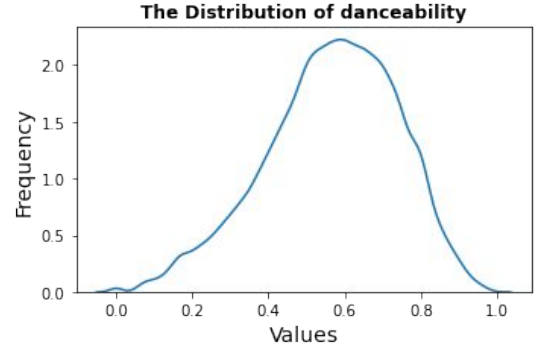


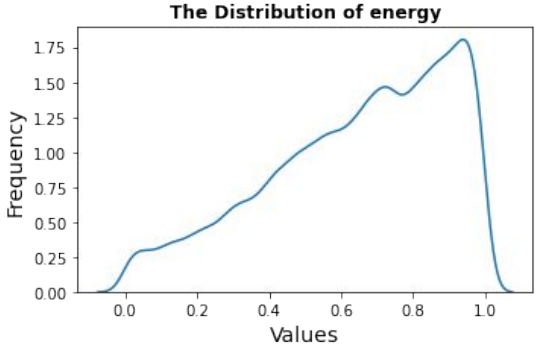


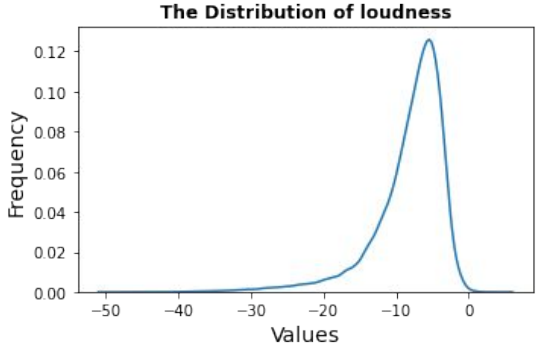


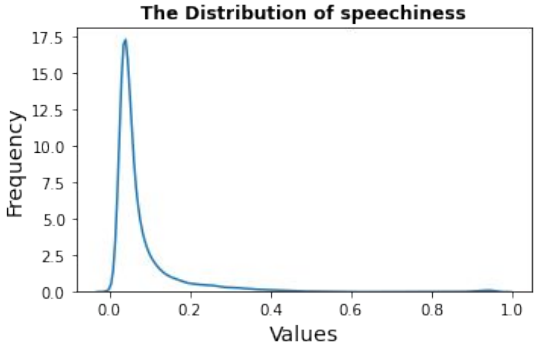

The Distribution of popularity
Frequency
Values
0.000
0.005
0.010
0.015
0.020
0
20
40
60
80
100

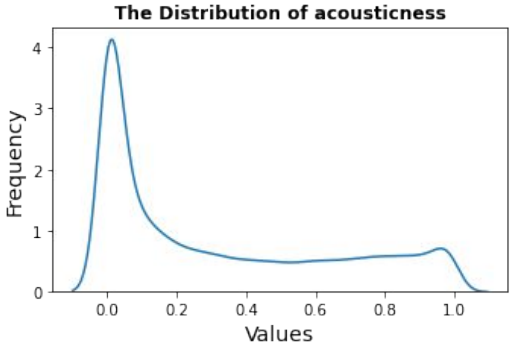

The Distribution of duration_ms
1e−6
Frequency
Values
1e6
0
1
2
3
4
5
6

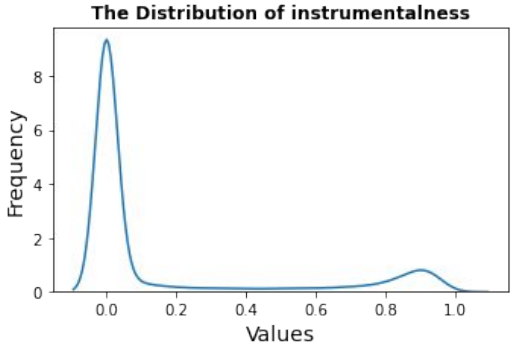

The Distribution of danceability
Frequency
Values

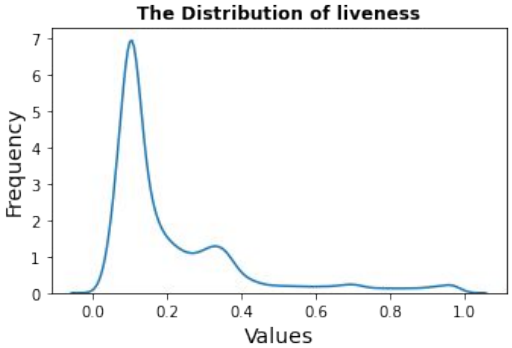

The Distribution of energy
Frequency
Values

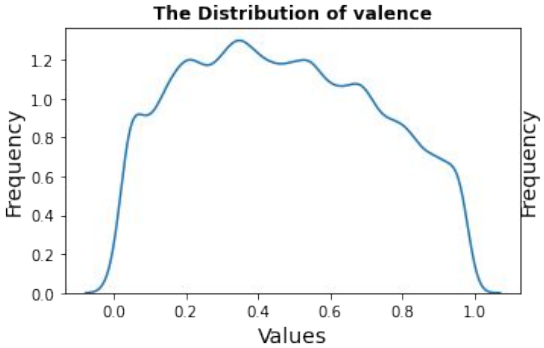

The Distribution of loudness
Frequency
Values

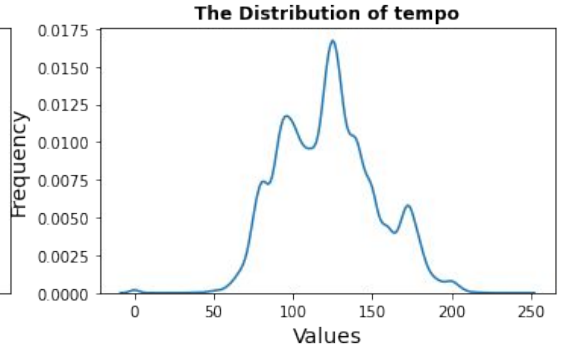

The Distribution of speechiness
Frequency
Values


The Distribution of acousticness
Frequency
Values


The Distribution of instrumentalness
Frequency
Values


The Distribution of liveness
Frequency
Values


The Distribution of valence
Frequency
Values


The Distribution of tempo
Frequency
Values

RESEARCH QUESTION 1: WHAT COULD POSSIBLY BE THE FORMULA OF A HIT SONG?

What will be the formula to create a hit song?

```
# We will first filter out all the songs with high popularity score
df_pop = df[(df['popularity'] > 80)]
```

```
#Check to see what type of cluster has the most popular songs
df_pop['Clusters'].value_counts()
```

```
2    423
0    388
1    141
Name: Clusters, dtype: int64
```

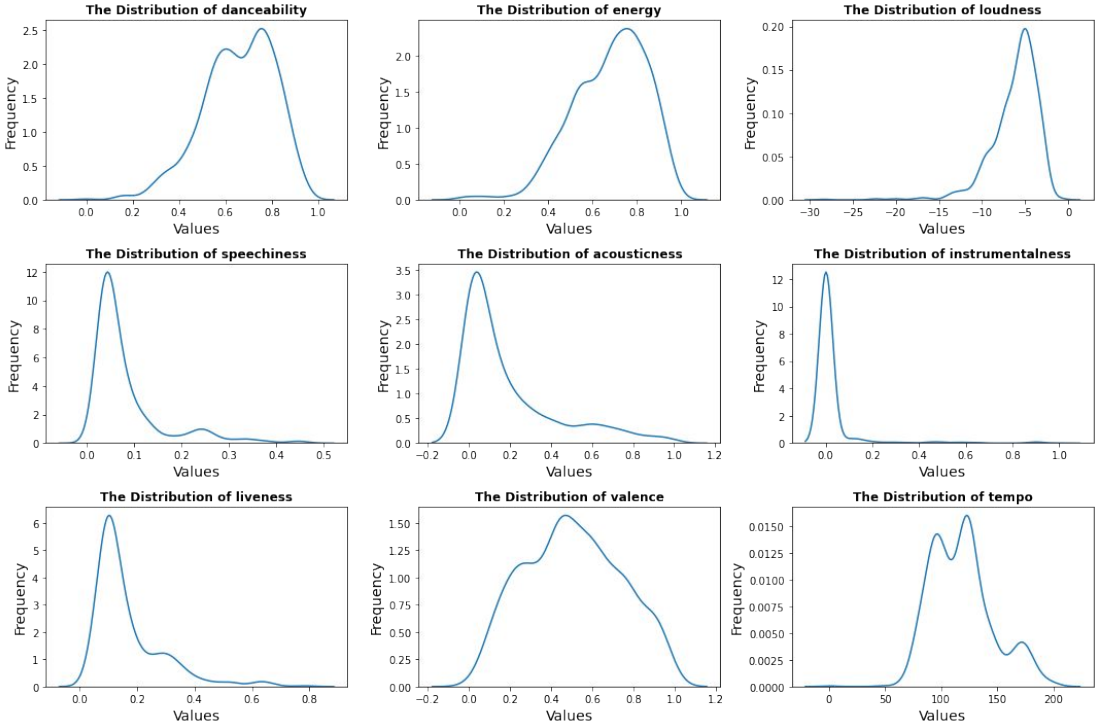
```
#Check to see what genre has the most popular songs
df_pop['track_genre'].value_counts()
```

```
pop           114
dance          88
rock           80
latino         70
hip-hop        58
              ...
pagode          1
metalcore       1
groove          1
indian          1
acoustic        1
Name: track_genre, Length: 61, dtype: int64
```

Firstly, I shrink the data down to only 1/10 of the original dataset, contains the most popular songs, which popularity score > 80.

We can see that most of the track_genre here are pretty prominent in the music industry: pop, dance, rock, latino, hip-hop

RESEARCH QUESTION 1: WHAT COULD POSSIBLY BE THE FORMULA OF A HIT SONG?



Except for valence, it seems that there are not much diversity in the distribution of other features.

We could see that danceability, energy, loudness, instrumentalness, acousticness, speechiness, liveness, tempo are skewed to one side of the graph.

A hit song may be a song with high daceability, energy, loudness, mid-tempo, low liveness, acousticnes, speechiness and instrumentalness.

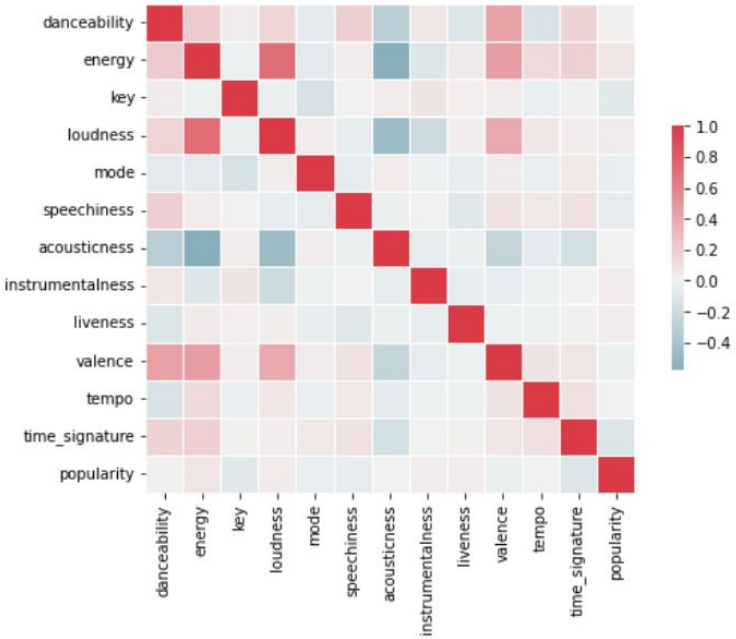
RESEARCH QUESTION 2: MUSIC LISTENING BEHAVIOURS PRE-COVID vs COVID

```
pre_2020 = []

#Initialize pagination variables
offset = 0
limit = 50

session = requests.Session()
retries = Retry(total=5, backoff_factor=1, status_forcelist=[429, 500, 502, 503, 504])
session.mount('https://', HTTPAdapter(max_retries=retries))

while True:
    try:
        results = sp.playlist_tracks(pre_2020_music_url, offset=offset, limit=limit)
        tracks = results['items']
        if len(tracks) == 0:
            break
        for track in tracks:
            # URI
            track_uri = track["track"]["uri"]
            audio_feats = sp.audio_features(track_uri)[0]
            pre_2020.append(audio_feats)
        offset += limit
        time.sleep(0.5)  # Adding a small delay to avoid overwhelming the API
    except Exception as e:
        print(f"Error occurred: {e}")
        time.sleep(5)  # Wait for a while before retrying
```

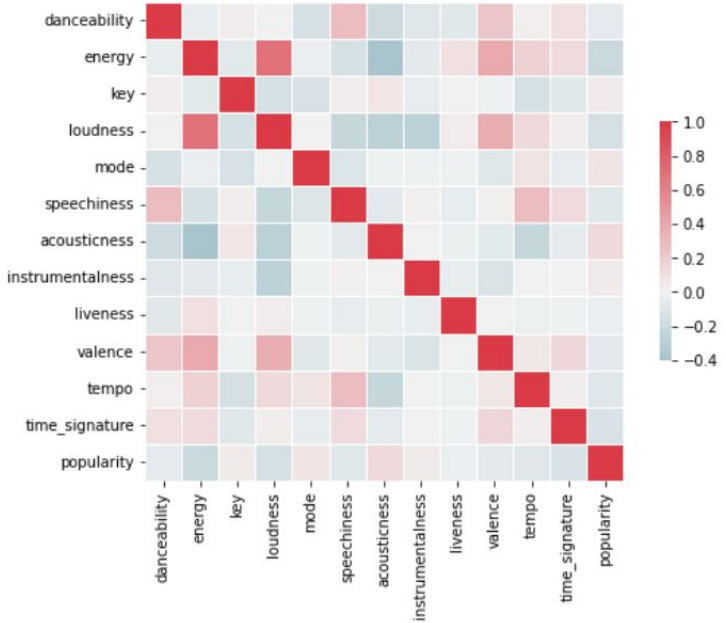
I created two playlists in my Spotify, both contains 290 most popular songs in a timeframe of 3 years (2017, 2018, 2019) vs (2020, 2021, 2022) to make this comparison.

After that, I pulled the data and converted it into two dataframes, which I named pre_2020_df and after_2020_df. I also pulled the popularity score for each song which I already added to the data_table in other code lines.

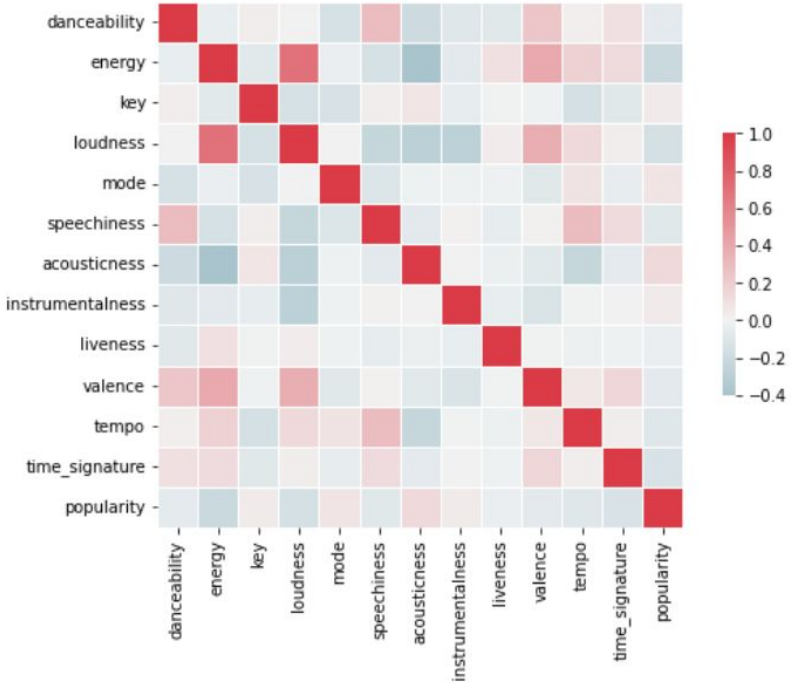
RESEARCH QUESTION 2: MUSIC LISTENING BEHAVIOURS PRE-COVID vs COVID



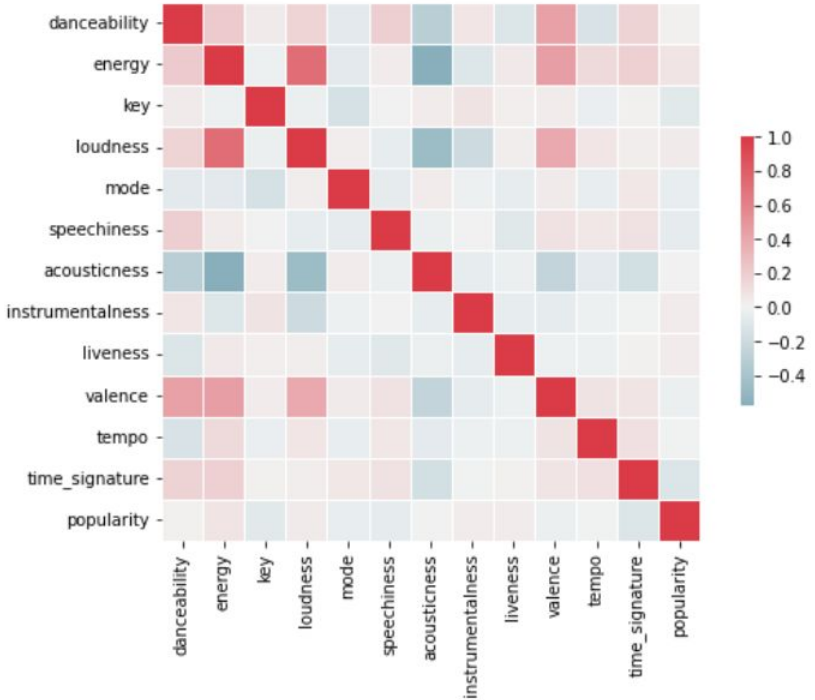
Pre 2020



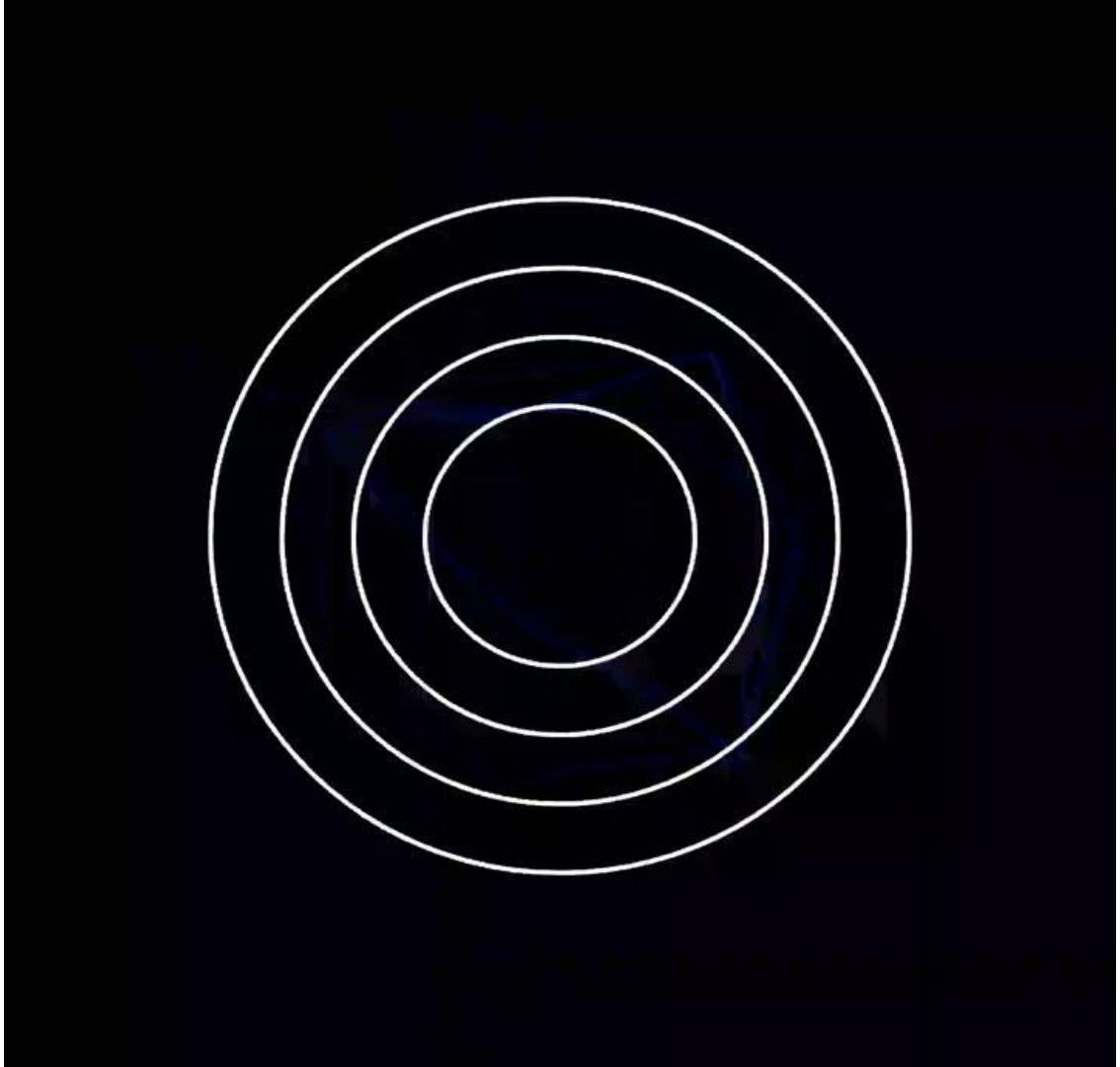
After 2020

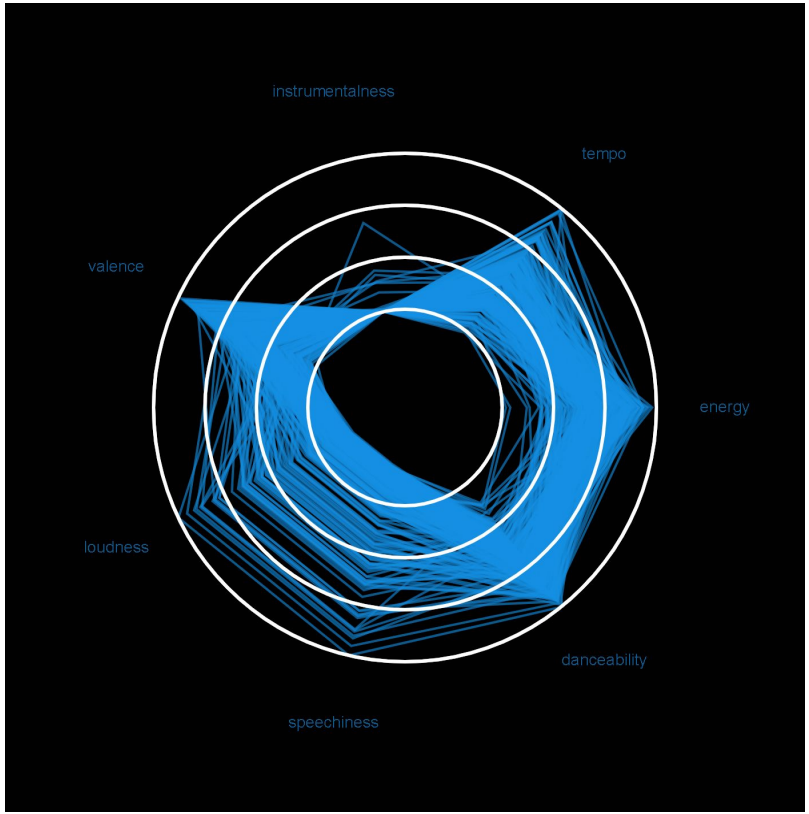


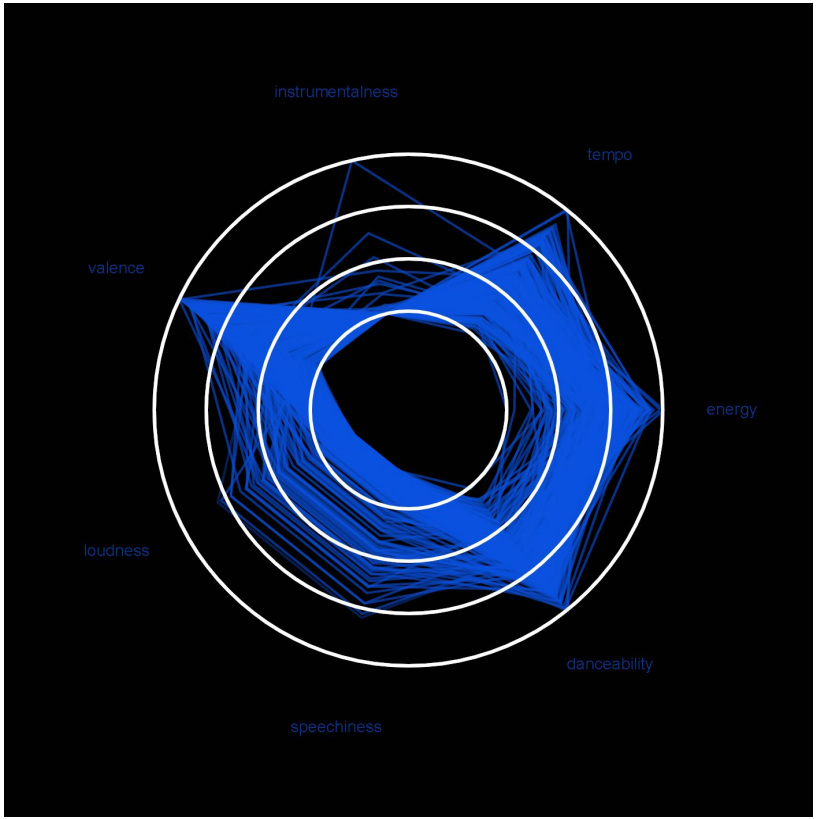
The danceability has a negative correlation with energy and loudness Tempo and Speechiness are highly correlated Valence and speechiness heavily affects the popularity of a song



The danceability has a positive correlation with valence, loudness and energy. The correlation between tempo and speechiness was not as strong as before anymore. Except for valence and speechiness, loudness and energy are the other two emerging features that affect popularity of a song the most







- The distribution of speechiness of after_2020 is not as dense as pre_2020 => Audience cares more about the vibe than the lyrics.
- Instrumentalness is more dense => Artists are going more experimental with their albums.

RESEARCH QUESTION 3: MUSIC LISTENING BEHAVIOURS PRE-COVID vs COVID

Songs are shrinking in terms of time

How long should an average track last according to today's standards?

```
after_2020_ms = round(after_2020_df['duration_ms'].mean(), 2)
pre_2020_ms = round(pre_2020_df['duration_ms'].mean(), 2)

print(f'The duration of average track pre-COVID: {after_2020_ms} The duration of average track during and after COVID {pre_2020_ms}'
```

```
The duration of average track pre-COVID: 196725.5 The duration of average track during and after COVID 209528.48
```