Group 14 - Data Collection and Processing for Spotify Music Recommender Systems

Importing the required libraries for the data collection process.

```
import numpy as np
import pandas as pd
import time
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from numpy import percentile

import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
```

Setting our credentials to allow us to pull data from the API.

```
# To access authorised Spotify data you must create a developer account and obtain credentials.
# Please reach out to us if you need to use our credentials. We can send that to you privately, to run this.
client_id= "YOUR_CLIENT_ID"
client_secret= "YOUR_CLIENT_SECRET"
```

```
client_credentials_manager = SpotifyClientCredentials(client_id=client_id, client_secret=client_secret)

sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager) #spotify object to access API
```

Passing list of the required artists. This is going to be from the user's music history.

```
names = ["Drake","Future","Coldplay"] #artists name list

result_req = []

for i in range(0,len(names)):
    result = sp.search(names[i])
    result_req.append(result['tracks']['items'][0]['artists'])

result_req
```

```
    [[{'external_urls': {'spotify': 'https://open.spotify.com/artist/3TVXtAsR1Inumwj472S9r4'},
       'href': 'https://api.spotify.com/v1/artists/3TVXtAsR1Inumwj472S9r4',
       'id': '3TVXtAsR1Inumwj472S9r4',
       'name': 'Drake',
       'type': 'artist',
       'uri': 'spotify:artist:3TVXtAsR1Inumwj472S9r4'},
      {'external_urls': {'spotify': 'https://open.spotify.com/artist/1RyvyyTE3xzB2ZywiAwp0i'},
       'href': 'https://api.spotify.com/v1/artists/1RyvyyTE3xzB2ZywiAwp0i',
       'id': '1RyvyyTE3xzB2ZywiAwp0i',
       'name': 'Future',
       'type': 'artist',
       'uri': 'spotify:artist:1RyvyyTE3xzB2ZywiAwp0i'},
      {'external_urls': {'spotify': 'https://open.spotify.com/artist/50co4Is1HCEo8bhOyUWKpn'},
       'href': 'https://api.spotify.com/v1/artists/50co4Is1HCEo8bhOyUWKpn',
       'id': '50co4Is1HCEo8bhOyUWKpn',
       'name': 'Young Thug',
       'type': 'artist',
       'uri': 'spotify:artist:50co4Is1HCEo8bhOyUWKpn'}],
     [{'external_urls': {'spotify': 'https://open.spotify.com/artist/3TVXtAsR1Inumwj472S9r4'},
       'href': 'https://api.spotify.com/v1/artists/3TVXtAsR1Inumwj472S9r4',
       'id': '3TVXtAsR1Inumwj472S9r4',
       'name': 'Drake',
       'type': 'artist',
       'uri': 'spotify:artist:3TVXtAsR1Inumwj472S9r4'},
      {'external_urls': {'spotify': 'https://open.spotify.com/artist/1RyvyyTE3xzB2ZywiAwp0i'},
       'href': 'https://api.spotify.com/v1/artists/1RyvyyTE3xzB2ZywiAwp0i',
       'id': '1RyvyyTE3xzB2ZywiAwp0i',
       'name': 'Future',
       'type': 'artist',
       'uri': 'spotify:artist:1RyvyyTE3xzB2ZywiAwp0i'},
      {'external_urls': {'spotify': 'https://open.spotify.com/artist/50co4Is1HCEo8bhOyUWKpn'},
       'href': 'https://api.spotify.com/v1/artists/50co4Is1HCEo8bhOyUWKpn',
       'id': '50co4Is1HCEo8bhOyUWKpn',
       'name': 'Young Thug',
       'type': 'artist',
       'uri': 'spotify:artist:50co4Is1HCEo8bhOyUWKpn'}],
     [{'external_urls': {'spotify': 'https://open.spotify.com/artist/4gzpq5DPGxSnKTe4SA8HAU'},
       'href': 'https://api.spotify.com/v1/artists/4gzpq5DPGxSnKTe4SA8HAU',
       'id': '4gzpq5DPGxSnKTe4SA8HAU',
       'name': 'Coldplay',
       'type': 'artist',
       'uri': 'spotify:artist:4gzpq5DPGxSnKTe4SA8HAU'},
      {'external_urls': {'spotify': 'https://open.spotify.com/artist/3Nrfpe0tUJi4K4DXYWgMUX'},
       'href': 'https://api.spotify.com/v1/artists/3Nrfpe0tUJi4K4DXYWgMUX',
       'id': '3Nrfpe0tUJi4K4DXYWgMUX',
       'name': 'BTS',
       'type': 'artist',
       'uri': 'spotify:artist:3Nrfpe0tUJi4K4DXYWgMUX'}]]
```

Extracting each artist's unique (uri).

```
artist_uri = []

for i in range(len(result_req)):
    for j in range(len(result_req[i])):
          artist_uri.append(result_req[i][j]['uri'])

# Using set() function on the obtained list to remove duplicates.

artist_uri = set(artist_uri)

artist_uri
```

```
{'spotify:artist:1RyvyyTE3xzB2ZywiAwp0i',
 'spotify:artist:3Nrfpe0tUJi4K4DXYWgMUX',
 'spotify:artist:3TVXtAsR1Inumwj472S9r4',
 'spotify:artist:4gzpq5DPGxSnKTe4SA8HAU',
 'spotify:artist:50co4Is1HCEo8bhOyUWKpn'}
```

Using the uri's. We then used spotipy's method sp.artist_albums() to get the info about all albums of the artist in Spotify's database.

```
# sp_albums = sp.albums

album_names = []
album_uris = []

#Pull data for all of the artist's albums
for artist in artist_uri:
    sp_albums = sp.artist_albums(artist, album_type='album')
    for i in range(len(sp_albums['items'])):
        album_names.append(sp_albums['items'][i]['name'])
        album_uris.append(sp_albums['items'][i]['uri'])

# Names and uris in same order to prevent duplicacy.

album_names
album_uris
```

```
      'spotify:album:38dLksLxrS6SBps345nbJI',
      'spotify:album:4dBp8rzdqH9unSndGk6g6o',
      'spotify:album:19CvkGjYpifkdwgVJSbog2',
      'spotify:album:6K1fjArmQ4SEiSDIG9fY01',
      'spotify:album:3cfAM8b8KqJRoIzt3zLKqw',
      'spotify:album:5IDGBfcVjwMoGPKOsfyXLN',
      'spotify:album:1hNS0RsxPTFjmKXCgmjSLS',
      'spotify:album:2G4AUqfwxcV1UdQjm2ouYr',
      'spotify:album:5KW3bBpEv3bvxAZ3MACEEz',
      'spotify:album:2R7iJz5uaHjLEVnMkloO18',
      'spotify:album:5GykKNn2KjofEoA8SpNnuw',
      'spotify:album:3pboBm7GTa6V5dFXXCt52b',
      'spotify:album:4XTT0NcNHyyl6h9JX2AfEj',
      'spotify:album:1xIT0NCNhyVI6h98XZhIEI',

      'spotify:album:2LIlrvVO0NP48jamVdlDo3',
      'spotify:album:6JlhIoegCcjtdbTQbypS8R',
      'spotify:album:71pRFAwHBLrjKYRG7V1Q2o',
      'spotify:album:1MnAljVs4hcGU6pTcK2jdT',
      'spotify:album:4Uo9tGSEkAUYHWfVGHhhZm',
      'spotify:album:3aITAVBURujVe8fhI2seeR',
      'spotify:album:48xpWR8K6CGpy3ETAym3pt',
      'spotify:album:27fzM2E0lgovCD7PCq6eh4',
      'spotify:album:6HcU64bPPXTHIbWmGblIkT',
      'spotify:album:4bNPOFOzxGhF5jhfIK6lit',
      'spotify:album:4V6ur2bnlbPUvpjkTHTnMW',
      'spotify:album:65hRa80KDworqyE8242rLN',
      'spotify:album:4SiRpStqaM5Xmd3nuGFM6R',
      'spotify:album:3LpIwZdzFwc10psLingT8x',
      'spotify:album:3ta7Nm07uIxMPVEiqpkLEN',
      'spotify:album:6P9PZjWXoCRF5b66BafPKY',
      'spotify:album:1uCLzanq1xy3eX2zyM4Sr0',
      'spotify:album:6Ew52HWkgfbth9ihRAq2Xd',
      'spotify:album:0KyO7XcPyKdqrbN08h8avh',
      'spotify:album:0C0Vs4XobImmqpr6kIasde',
      'spotify:album:0US0Bhn0oOLwVqXVnz0HDb',
      'spotify:album:187UNqZ7MX3neMYkkevmdm',
      'spotify:album:1P8NvRvykmDrKyfglMerMv',
      'spotify:album:4YtTX4GPvBvewbJvBfXCS2',
      'spotify:album:7K6OykPbezfgKgBufihn6X',
      'spotify:album:3SpBlxme9WbeQdI9kx7KAV',
      'spotify:album:6sp02aeyiwfX35xRqwNiPv',
      'spotify:album:6OQ9gBfg5EXeNAEwGSs6jK',
      'spotify:album:45c1tgTktunRMmfh3WVh8U',
      'spotify:album:7dqpveMVcWgbzqYrOdkFTD',
      'spotify:album:6CY70qRxPutN3VKfYhNREa',
      'spotify:album:2podUJIFG8hLfFz7Kqe8yJ',
      'spotify:album:1ATL5GLyefJaxhQzSPVrLX',
      'spotify:album:42wvKYHFezpmDuAP43558f',
      'spotify:album:1lXY618HWkwYKJWBRYR4MK',
      'spotify:album:4dvkEfxroInqojJWP06R2V',
      'spotify:album:40GMAhriYJRO1rsY4YdrZb',
      'spotify:album:2yIwhsIWGRQzGQdn1czSK0',
      'spotify:album:15QCBYjP6HwHvsff100UBx',
      'spotify:album:1ozpmkWcCHwsQ4QTnxOOdT',
      'spotify:album:0ptlfJfwGTy0Yvrk14JK1I',
      'spotify:album:5bqZfS9HUBTtxW0UiG05qC',
      'spotify:album:2ZUFSbIkmFkGag000RWQpA',
      'spotify:album:2Z6FSbIXmFXCdgJ00KWOpK',
      'spotify:album:5mz0mJxb80gqJIcRf9LGHJ',
      'spotify:album:766Pi8jEi9JZRvi4y9KRdP']
```

Defining function for adding album info like songs and track uri to a dictionary. This will be called recursively for each album.

```
def albumSongs(uri):
    album = uri #assign album uri to a_name
    spotify_albums[album] = {} #Creates dictionary for that specific album
    #Create keys-values of empty lists inside nested dictionary for album
    spotify_albums[album]['album'] = [] #create empty list
    spotify_albums[album]['track_number'] = []
    spotify_albums[album]['id'] = []
    spotify_albums[album]['name'] = []
    spotify_albums[album]['uri'] = []
    tracks = sp.album_tracks(album) #pull data on album tracks
    for n in range(len(tracks['items'])): #for each song track
            spotify_albums[album]['album'].append(album_names[album_count]) #append album name tracked via album_c
            spotify_albums[album]['track_number'].append(tracks['items'][n]['track_number'])
            spotify_albums[album]['id'].append(tracks['items'][n]['id'])
            spotify_albums[album]['name'].append(tracks['items'][n]['name'])
            spotify_albums[album]['uri'].append(tracks['items'][n]['uri'])
```

Calling the function to create the required dictionary. Spotify_albums { }.

```
spotify_albums = {} # initializing the dict so the function can add values to it.
album_count = 0
for i in album_uris: #each album being called using loop iterator i.
    albumSongs(i)
    print("Album " + str(album_names[album_count]) + " songs has been added to spotify_albums dictionary")
    album_count+=1 #Updates album count once all songs are added to move to the next album.
```

```
    Album Everyday Life songs has been added to spotify_albums dictionary
```

```
Album Everyday Life songs has been added to spotify_albums dictionary
Album Everyday Life songs has been added to spotify_albums dictionary
Album Live in Buenos Aires songs has been added to spotify_albums dictionary
Album Love in Tokyo songs has been added to spotify_albums dictionary
Album A Head Full of Dreams songs has been added to spotify_albums dictionary
Album A Head Full of Dreams Tour Edition songs has been added to spotify_albums dictionary
Album Ghost Stories Live 2014 songs has been added to spotify_albums dictionary
Album Ghost Stories songs has been added to spotify_albums dictionary
Album Mylo Xyloto songs has been added to spotify_albums dictionary
Album Mylo Xyloto songs has been added to spotify_albums dictionary
Album Mylo Xyloto songs has been added to spotify_albums dictionary
Album LeftRightLeftRightLeft (Live) songs has been added to spotify_albums dictionary
Album Viva La Vida (Prospekt's March Edition) songs has been added to spotify_albums dictionary
Album Viva La Vida (Prospekt's March Edition) songs has been added to spotify_albums dictionary
Album Viva La Vida (Prospekt's March Edition) songs has been added to spotify_albums dictionary
Album Viva La Vida (Prospekt's March Edition) songs has been added to spotify_albums dictionary
Album Viva La Vida (Prospekt's March Edition) songs has been added to spotify_albums dictionary
Album Viva La Vida or Death and All His Friends songs has been added to spotify_albums dictionary
Album Pluto x Baby Pluto (Deluxe) songs has been added to spotify_albums dictionary
Album Pluto x Baby Pluto songs has been added to spotify_albums dictionary
Album Pluto x Baby Pluto (Deluxe) songs has been added to spotify_albums dictionary
Album Pluto x Baby Pluto songs has been added to spotify_albums dictionary
Album High Off Life songs has been added to spotify_albums dictionary
Album High Off Life songs has been added to spotify_albums dictionary
Album SAVE ME songs has been added to spotify_albums dictionary
Album SAVE ME songs has been added to spotify_albums dictionary
Album Future Hndrxx Presents: The WIZRD songs has been added to spotify_albums dictionary
Album Future Hndrxx Presents: The WIZRD songs has been added to spotify_albums dictionary
Album Future & Juice WRLD Present... WRLD ON DRUGS songs has been added to spotify_albums dictionary
Album Future & Juice WRLD Present... WRLD ON DRUGS songs has been added to spotify_albums dictionary
Album BEASTMODE 2 songs has been added to spotify_albums dictionary
Album BEASTMODE 2 songs has been added to spotify_albums dictionary
Album SUPERFLY (Original Motion Picture Soundtrack) songs has been added to spotify_albums dictionary
Album SUPER SLIMEY songs has been added to spotify_albums dictionary
Album SUPER SLIMEY songs has been added to spotify_albums dictionary
Album HNDRXX songs has been added to spotify_albums dictionary
Album FUTURE songs has been added to spotify_albums dictionary
Album HNDRXX songs has been added to spotify_albums dictionary
Album Certified Lover Boy songs has been added to spotify_albums dictionary
Album Certified Lover Boy songs has been added to spotify_albums dictionary
Album Dark Lane Demo Tapes songs has been added to spotify_albums dictionary
Album Dark Lane Demo Tapes songs has been added to spotify_albums dictionary
Album Care Package songs has been added to spotify_albums dictionary
Album Care Package songs has been added to spotify_albums dictionary
Album Care Package songs has been added to spotify_albums dictionary
Album So Far Gone songs has been added to spotify_albums dictionary
Album Scorpion songs has been added to spotify_albums dictionary
Album Scorpion songs has been added to spotify_albums dictionary
Album More Life songs has been added to spotify_albums dictionary
Album More Life songs has been added to spotify_albums dictionary
Album Views songs has been added to spotify_albums dictionary
Album Views songs has been added to spotify_albums dictionary
Album What A Time To Be Alive songs has been added to spotify_albums dictionary
Album What A Time To Be Alive songs has been added to spotify_albums dictionary
Album If You're Reading This It's Too Late songs has been added to spotify_albums dictionary
Album If You're Reading This It's Too Late songs has been added to spotify_albums dictionary
Album Nothing Was The Same (Deluxe) songs has been added to spotify_albums dictionary
Album Nothing Was The Same (Deluxe) songs has been added to spotify_albums dictionary
Album Nothing Was The Same (Deluxe) songs has been added to spotify_albums dictionary
```

Defining function to extract audio features of all the songs from each album we need to train the model.

```
def audio_features(album):
    #Add new keys where we will add value for the corresponding audio feature.
    spotify_albums[album]['acousticness'] = []
    spotify_albums[album]['danceability'] = []
    spotify_albums[album]['energy'] = []
    spotify_albums[album]['instrumentalness'] = []
    spotify_albums[album]['liveness'] = []
    spotify_albums[album]['loudness'] = []
    spotify_albums[album]['speechiness'] = []
    spotify_albums[album]['tempo'] = []
    spotify_albums[album]['valence'] = []
    spotify_albums[album]['popularity'] = []
    spotify_albums[album]['duration_ms'] = []
    #create a track counter
    song_number = 0
    for song in spotify_albums[album]['uri']:
        #pull audio features per song
        features = sp.audio_features(song)
        #Adding corresponding values for each song into each audio feature key.
        spotify_albums[album]['acousticness'].append(features[0]['acousticness'])
        spotify_albums[album]['danceability'].append(features[0]['danceability'])
        spotify_albums[album]['energy'].append(features[0]['energy'])
        spotify_albums[album]['instrumentalness'].append(features[0]['instrumentalness'])
        spotify_albums[album]['liveness'].append(features[0]['liveness'])
        spotify_albums[album]['loudness'].append(features[0]['loudness'])
        spotify_albums[album]['speechiness'].append(features[0]['speechiness'])
        spotify_albums[album]['tempo'].append(features[0]['tempo'])
        spotify_albums[album]['valence'].append(features[0]['valence'])
        spotify_albums[album]['duration_ms'].append(features[0]['duration_ms'])
        #using function sp.track to get the song's popularity. Not available in the track info.
        pop_of_song = sp.track(song)
        spotify_albums[album]['popularity'].append(pop_of_song['popularity'])
        song_number+=1
```

Running the extract audiofeatures function on each album's every song and updating the dictionary accordingly.

Running this cell will take significant time!!!

The function is run recursively over all albums increasing the time.

```
for i in spotify_albums:
    audio_features(i)
%time
```

```
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 5.96 µs
```

Creating a dictionary to add all the information of the audio features and track info that we will convert to dataframe for extraction.

```
dic_df = {}
dic_df['album'] = []
dic_df['track_number'] = []
dic_df['id'] = []
dic_df['name'] = []
dic_df['uri'] = []
dic_df['acousticness'] = []
dic_df['danceability'] = []
dic_df['energy'] = []
dic_df['instrumentalness'] = []
dic_df['liveness'] = []
dic_df['loudness'] = []
dic_df['speechiness'] = []
dic_df['tempo'] = []
dic_df['valence'] = []
dic_df['popularity'] = []
dic_df['duration_ms'] = []
for album in spotify_albums:
    for feature in spotify_albums[album]:
        dic_df[feature].extend(spotify_albums[album][feature])

len(dic_df['album']) # checking total number of songs after the data collection process.
```

```
1560
```

Converting the dictionary into df using Pandas.

```
df = pd.DataFrame.from_dict(dic_df)
df.head() # Checking the head of the df to ensure everything was smoothly run.
```

	album	track_number	id	name	
0	BE	1	249gnXrbfmV8NG6jTEMSwD	Life Goes On	spotify:track:249gnXrb
1	BE	2	3QH8rQGNFX8VLbCgZ7uPTS	Fly To My Room	spotify:track:3QH8rQGN
2	BE	3	0n2moJpAEWHwaPYYjkzMDl	Blue & Grey	spotify:track:0n2moJpA
3	BE	4	4GVwjLRT7oSsKby7Vy8EHr	Skit	spotify:track:4GVwjLF
4	BE	5	2FVpOsjT1iquZ3SpCjZ9Ne	Telepathy	spotify:track:2FVpOs

This cell is run if you want to drop duplicates (IF ANY) based on any feature.

We used this to check if there were any duplicates based on id.

```
print(len(df))
final_df = df.sort_values('popularity', ascending=False).drop_duplicates('id').sort_index()
print(len(final_df))
```

```
1560
1560
```

Finally writing to csv to save the df which will be used for eda and visualization.

```
final_df.to_csv("Spotify_Audio_Features.csv")
```

In order to get a comprehensive dataset for our Recommender system, we would have to scale this for several more artists and playlists.

We do not have the computing resources as making Get() requests from the API for 3 artists took us over 10 minutes

Hence we found a dataset with 1.2 million rows for our recommendation system

Dataset link: https://drive.google.com/drive/folders/1nWFP1p93zER8FEg0E8dKGDZVJd-A4zt2?usp=sharing

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
tracks = pd.read_csv('/content/drive/MyDrive/Data Mining - Group 14 - Dataset/tracks_features.csv')
```

```
tracks.head()
```

	id	name	album	album_id	artists	
0	7lmeHLHBe4nmXzuXc0HDjk	Testify	The Battle Of Los	2eia0myWFgoHuttJytCxgX	['Rage Against The	[

```
tracks.isna().sum()
```

```
id                  0
name                0
album               0
album_id            0
artists             0
artist_ids          0
track_number        0
disc_number         0
explicit            0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
duration_ms         0
time_signature      0
year                0
release_date        0
dtype: int64
```

Here we can see that there are no null values. This is consistent with the data that we pulled using the API.

If additional data is required then we can fetch selective data using the API

```
tracks.describe()
```

	track_number	disc_number	danceability	energy	key	
count	1.204025e+06	1.204025e+06	1.204025e+06	1.204025e+06	1.204025e+06	1.2
mean	7.656352e+00	1.055906e+00	4.930565e-01	5.095363e-01	5.194151e+00	-1.1
std	5.994977e+00	2.953752e-01	1.896694e-01	2.946839e-01	3.536731e+00	6.9
min	1.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-6.0
25%	3.000000e+00	1.000000e+00	3.560000e-01	2.520000e-01	2.000000e+00	-1.5
50%	7.000000e+00	1.000000e+00	5.010000e-01	5.240000e-01	5.000000e+00	-9.7
75%	1.000000e+01	1.000000e+00	6.330000e-01	7.660000e-01	8.000000e+00	-6.7
max	5.000000e+01	1.300000e+01	1.000000e+00	1.000000e+00	1.100000e+01	7.2

```
tracks.info()
```

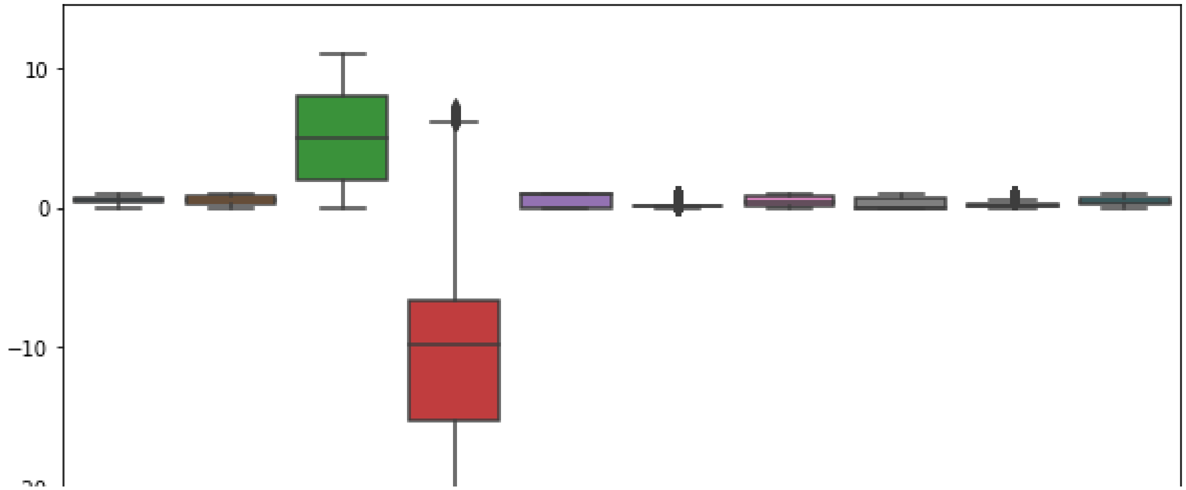
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1204025 entries, 0 to 1204024
Data columns (total 24 columns):
 #   Column            Non-Null Count    Dtype  
---  ------            --------------    -----  
 0   id                1204025 non-null  object 
 1   name              1204025 non-null  object 
 2   album             1204025 non-null  object 
 3   album_id          1204025 non-null  object 
 4   artists           1204025 non-null  object 
 5   artist_ids        1204025 non-null  object 
 6   track_number      1204025 non-null  int64  
 7   disc_number       1204025 non-null  int64  
 8   explicit          1204025 non-null  bool   
 9   danceability      1204025 non-null  float64
 10  energy            1204025 non-null  float64
 11  key               1204025 non-null  int64  
 12  loudness          1204025 non-null  float64
 13  mode              1204025 non-null  int64  
 14  speechiness       1204025 non-null  float64
 15  acousticness      1204025 non-null  float64
 16  instrumentalness  1204025 non-null  float64
```

```
  17  liveness          1204025 non-null  float64
  18  valence           1204025 non-null  float64
  19  tempo             1204025 non-null  float64
  20  duration_ms       1204025 non-null  int64
  21  time_signature    1204025 non-null  float64
  22  year              1204025 non-null  int64
  23  release_date      1204025 non-null  object
 dtypes: bool(1), float64(10), int64(6), object(7)
 memory usage: 212.4+ MB
```

Plotting the audio features as boxplot to view outliers

```
plt.figure(figsize = (10,10))
sns.boxplot(data=tracks.iloc[:,9:19], )
plt.xticks(rotation=90)
```

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 <a list of 10 Text major ticklabel objects>)
```



```
loudness  = tracks['loudness']
loudness
```

```
0         -5.399
1         -5.764
2         -5.424
3         -5.830
4         -6.729
           ...
1204020   -6.970
1204021   -6.602
1204022   -5.960
1204023   -6.788
1204024   -9.279
Name: loudness, Length: 1204025, dtype: float64
```

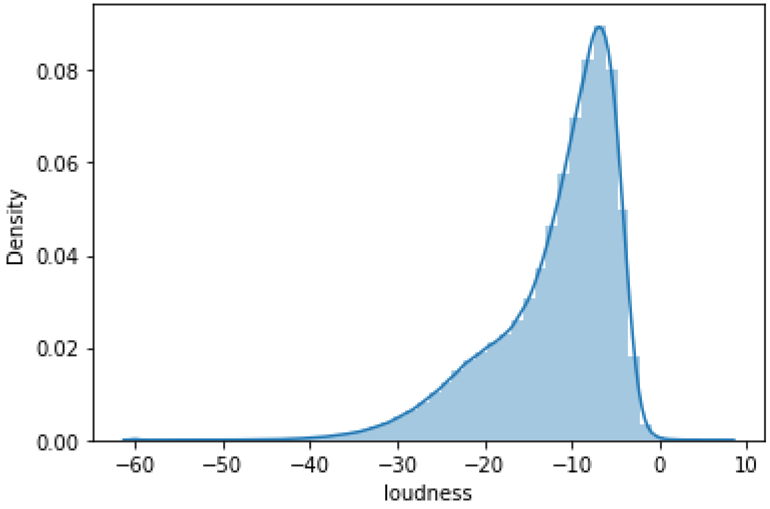
Loudness has a wide range of outliers

Using IQR to clean the outliers

```
q25, q75 = percentile(loudness, 25), percentile(loudness, 75)
iqr = q75 - q25
iqr = q75 - q25
print('Percentiles: 25th=%.3f, 75th=%.3f, IQR=%.3f' % (q25, q75, iqr) )
cut_off = iqr * 1.5
lower, upper = q25 - cut_off, q75 + cut_off

    Percentiles: 25th=-15.254, 75th=-6.717, IQR=8.537
```

Plotting the loudness values to get an idea of how the data is distributed

```
sns.distplot(tracks['loudness'])

    /usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Futur
      warnings.warn(msg, FutureWarning)
    <matplotlib.axes._subplots.AxesSubplot at 0x7fcdaf04a790>
```



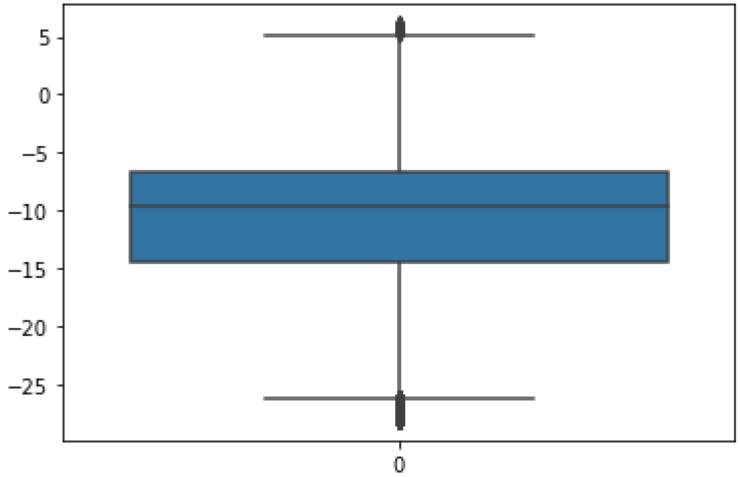
```
loudness.describe()

    count    1.204025e+06
    mean    -1.180870e+01
    std      6.982132e+00
    min     -6.000000e+01
    25%     -1.525400e+01
    50%     -9.791000e+00
```

```
75%     -6.717000e+00
max      7.234000e+00
Name: loudness, dtype: float64
```

```
clean_tracks = tracks[(tracks['loudness'] > lower) &( tracks['loudness'] < upper)]
```

```
sns.boxplot(data = clean_tracks['loudness'])
```

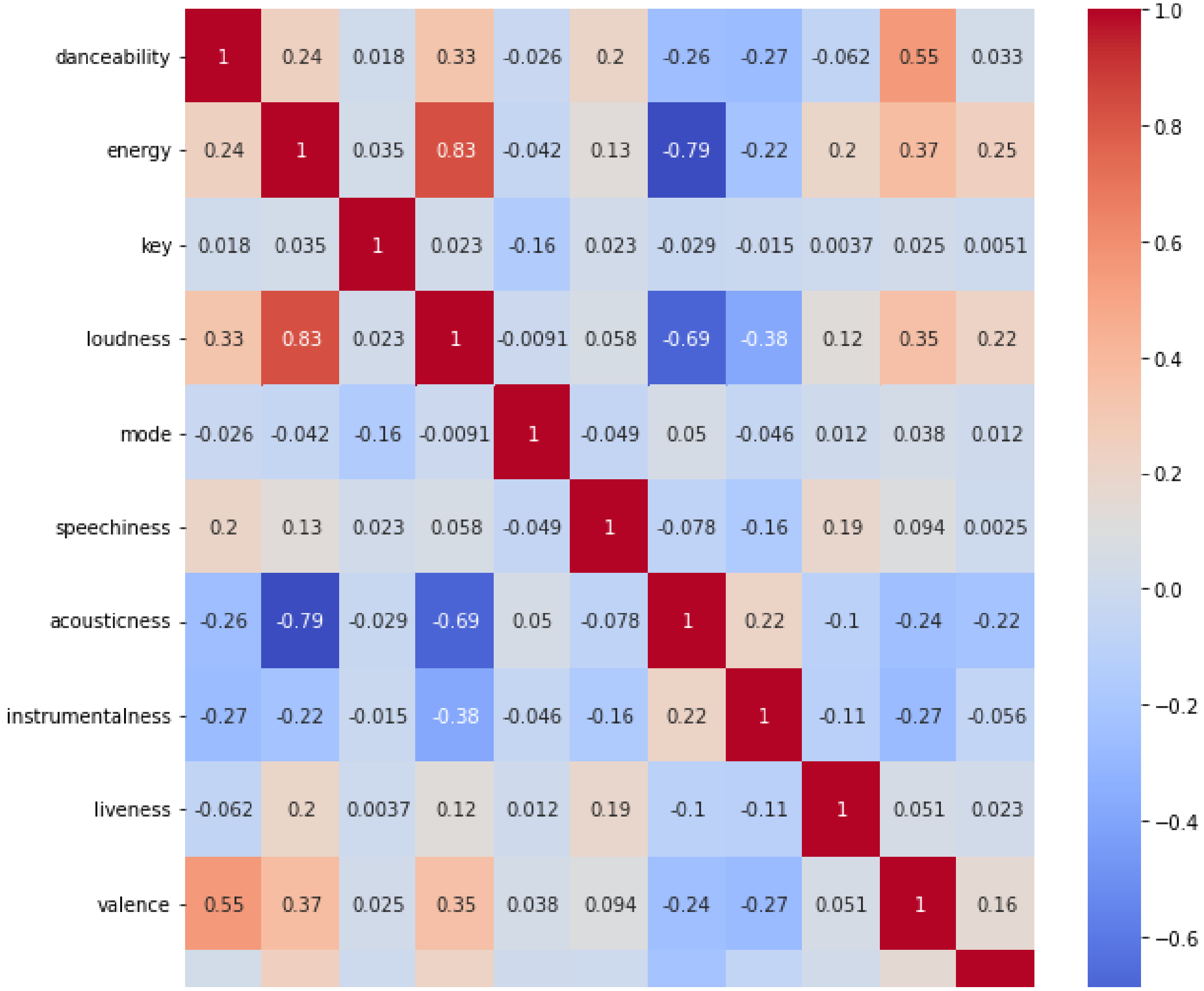
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcdaeb9c750>
```



Plotting heatmap to get the correlation between variables

```
plt.figure(figsize=(10,10))
sns.heatmap( clean_tracks.iloc[:,9:20].corr(),annot=True,cmap="coolwarm")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcda5772dd0>
```



danceability	1	0.24	0.018	0.33	-0.026	0.2	-0.26	-0.27	-0.062	0.55	0.033
energy	0.24	1	0.035	0.83	-0.042	0.13	-0.79	-0.22	0.2	0.37	0.25
key	0.018	0.035	1	0.023	-0.16	0.023	-0.029	-0.015	0.0037	0.025	0.0051
loudness	0.33	0.83	0.023	1	-0.0091	0.058	-0.69	-0.38	0.12	0.35	0.22
mode	-0.026	-0.042	-0.16	-0.0091	1	-0.049	0.05	-0.046	0.012	0.038	0.012
speechiness	0.2	0.13	0.023	0.058	-0.049	1	-0.078	-0.16	0.19	0.094	0.0025
acousticness	-0.26	-0.79	-0.029	-0.69	0.05	-0.078	1	0.22	-0.1	-0.24	-0.22
instrumentalness	-0.27	-0.22	-0.015	-0.38	-0.046	-0.16	0.22	1	-0.11	-0.27	-0.056
liveness	-0.062	0.2	0.0037	0.12	0.012	0.19	-0.1	-0.11	1	0.051	0.023
valence	0.55	0.37	0.025	0.35	0.038	0.094	-0.24	-0.27	0.051	1	0.16

Here we can see that there is high positive correlation between:

- Loudness and Energy
- Valence & Danceability

We can observe high negative correlation between:

- Acousticness & Energy
- Acousticness & Loudness

✓ 1s completed at 10:39 PM ● ✕