

creating-cohorts-of-songs-1

November 18, 2024

1 PGC_DS_ML_Course_End-Project(Creating_coh9orts-of-songs-spottify)

2 Name- Sourajyoti Ghosh

3 Objective:

As a data scientist, you should perform exploratory data analysis and cluster analysis to create cohorts of songs. The goal is to beter understand the variou factors that create a cohort of songs.

```
[55]: # Lets import the required python libraries
```

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

```
[5]: df = pd.read_csv('rolling_stones_spotify.csv')
```

```
[6]: df
```

```
[6]:
```

	Unnamed: 0	name	album \
0	0	Concert Intro Music - Live	Licked Live In NYC
1	1	Street Fighting Man - Live	Licked Live In NYC
2	2	Start Me Up - Live	Licked Live In NYC
3	3	If You Can't Rock Me - Live	Licked Live In NYC
4	4	Don't Stop - Live	Licked Live In NYC
...
1605	1605	Carol	The Rolling Stones
1606	1606	Tell Me	The Rolling Stones
1607	1607	Can I Get A Witness	The Rolling Stones
1608	1608	You Can Make It If You Try	The Rolling Stones
1609	1609	Walking The Dog	The Rolling Stones

	release_date	track_number	id \
0	2022-06-10	1	2IEkywLJ4ykbhi1yRQvmsT

1	2022-06-10	2	6GVgVJBKkGJoRfarYRvGTU
2	2022-06-10	3	1Lu761pZ0dBTGpzxaQoZNW
3	2022-06-10	4	1agTQzOTUnGNnggyckEqiDH
4	2022-06-10	5	7piGJR8YndQBQWVXv6KtQw
...
1605	1964-04-16	8	08l7M5UpRnffG10FyuRiQZ
1606	1964-04-16	9	3JZ1lQBstM6WwoJdzFDLhx
1607	1964-04-16	10	0t2qvfsBQ3Y081zRRoVTdb
1608	1964-04-16	11	5ivIs5vwSjORChOIvly30n
1609	1964-04-16	12	43SkTJJ2xleDaeiE4TIM70

	uri	acousticness	danceability	\
0	spotify:track:2IEkywLJ4ykbhi1yRQvmsT	0.0824	0.463	
1	spotify:track:6GVgVJBKkGJoRfarYRvGTU	0.4370	0.326	
2	spotify:track:1Lu761pZ0dBTGpzxaQoZNW	0.4160	0.386	
3	spotify:track:1agTQzOTUnGNnggyckEqiDH	0.5670	0.369	
4	spotify:track:7piGJR8YndQBQWVXv6KtQw	0.4000	0.303	
...	
1605	spotify:track:08l7M5UpRnffG10FyuRiQZ	0.1570	0.466	
1606	spotify:track:3JZ1lQBstM6WwoJdzFDLhx	0.0576	0.509	
1607	spotify:track:0t2qvfsBQ3Y081zRRoVTdb	0.3710	0.790	
1608	spotify:track:5ivIs5vwSjORChOIvly30n	0.2170	0.700	
1609	spotify:track:43SkTJJ2xleDaeiE4TIM70	0.3830	0.727	

	energy	instrumentalness	liveness	loudness	speechiness	tempo	\
0	0.993	0.996000	0.9320	-12.913	0.1100	118.001	
1	0.965	0.233000	0.9610	-4.803	0.0759	131.455	
2	0.969	0.400000	0.9560	-4.936	0.1150	130.066	
3	0.985	0.000107	0.8950	-5.535	0.1930	132.994	
4	0.969	0.055900	0.9660	-5.098	0.0930	130.533	
...	
1605	0.932	0.006170	0.3240	-9.214	0.0429	177.340	
1606	0.706	0.000002	0.5160	-9.427	0.0843	122.015	
1607	0.774	0.000000	0.0669	-7.961	0.0720	97.035	
1608	0.546	0.000070	0.1660	-9.567	0.0622	102.634	
1609	0.934	0.068500	0.0965	-8.373	0.0359	125.275	

	valence	popularity	duration_ms
0	0.0302	33	48640
1	0.3180	34	253173
2	0.3130	34	263160
3	0.1470	32	305880
4	0.2060	32	305106
...
1605	0.9670	39	154080
1606	0.4460	36	245266
1607	0.8350	30	176080

```
1608    0.5320         27      121680
1609    0.9690         35      189186
```

```
[1610 rows x 18 columns]
```

```
[14]: df = df.drop(['Unnamed: 0'], axis=1)
```

```
[15]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1610 entries, 0 to 1609
Data columns (total 17 columns):
#   Column                Non-Null Count  Dtype
---  -
0   name                  1610 non-null  object
1   album                 1610 non-null  object
2   release_date          1610 non-null  object
3   track_number          1610 non-null  int64
4   id                    1610 non-null  object
5   uri                   1610 non-null  object
6   acousticness          1610 non-null  float64
7   danceability           1610 non-null  float64
8   energy                 1610 non-null  float64
9   instrumentalness       1610 non-null  float64
10  liveness               1610 non-null  float64
11  loudness               1610 non-null  float64
12  speechiness           1610 non-null  float64
13  tempo                  1610 non-null  float64
14  valence                1610 non-null  float64
15  popularity             1610 non-null  int64
16  duration_ms           1610 non-null  int64
dtypes: float64(9), int64(3), object(5)
memory usage: 214.0+ KB
```

```
[17]: df1=df
      df_copy=df
```

```
[19]: df1.describe()
```

```
[19]:
```

	track_number	acousticness	danceability	energy \
count	1610.000000	1610.000000	1610.000000	1610.000000
mean	8.613665	0.250475	0.468860	0.792352
std	6.560220	0.227397	0.141775	0.179886
min	1.000000	0.000009	0.104000	0.141000
25%	4.000000	0.058350	0.362250	0.674000
50%	7.000000	0.183000	0.458000	0.848500
75%	11.000000	0.403750	0.578000	0.945000

max	47.000000	0.994000	0.887000	0.999000	
-----	-----------	----------	----------	----------	--

	instrumentalness	liveness	loudness	speechiness	tempo \
count	1610.000000	1610.000000	1610.000000	1610.000000	1610.000000
mean	0.164170	0.49173	-6.971615	0.069512	126.082033
std	0.276249	0.34910	2.994003	0.051631	29.233483
min	0.000000	0.02190	-24.408000	0.023200	46.525000
25%	0.000219	0.15300	-8.982500	0.036500	107.390750
50%	0.013750	0.37950	-6.523000	0.051200	124.404500
75%	0.179000	0.89375	-4.608750	0.086600	142.355750
max	0.996000	0.99800	-1.014000	0.624000	216.304000

	valence	popularity	duration_ms
count	1610.000000	1610.000000	1610.000000
mean	0.582165	20.788199	257736.488199
std	0.231253	12.426859	108333.474920
min	0.000000	0.000000	21000.000000
25%	0.404250	13.000000	190613.000000
50%	0.583000	20.000000	243093.000000
75%	0.778000	27.000000	295319.750000
max	0.974000	80.000000	981866.000000

```
[20]: # checking total no of null values
df.isnull().sum()
```

```
[20]: name          0
      album        0
      release_date  0
      track_number  0
      id           0
      uri          0
      acousticness  0
      danceability  0
      energy        0
      instrumentalness  0
      liveness      0
      loudness      0
      speechiness   0
      tempo         0
      valence       0
      popularity    0
      duration_ms   0
      dtype: int64
```

```
[21]: df.dtypes
```

```
[21]: name          object
      album         object
      release_date  object
      track_number  int64
      id           object
      uri          object
      acousticness  float64
      danceability  float64
      energy        float64
      instrumentalness float64
      liveness      float64
      loudness      float64
      speechiness   float64
      tempo         float64
      valence       float64
      popularity    int64
      duration_ms   int64
      dtype: object
```

```
[24]: ## getting rid of all the strings for correlation calculation
      df2 = df.drop(df.select_dtypes(include='object'), axis=1)
```

```
[27]: df2.corr()
```

```
[27]:
```

	track_number	acousticness	danceability	energy	\
track_number	1.000000	-0.035675	-0.112004	0.096314	
acousticness	-0.035675	1.000000	0.070017	-0.363819	
danceability	-0.112004	0.070017	1.000000	-0.300536	
energy	0.096314	-0.363819	-0.300536	1.000000	
instrumentalness	-0.002772	0.061403	-0.031812	0.120261	
liveness	0.188351	-0.117739	-0.516387	0.511188	
loudness	0.100835	-0.237083	-0.249406	0.698039	
speechiness	0.040617	-0.021774	-0.322684	0.417214	
tempo	-0.023934	-0.171003	-0.324398	0.201885	
valence	-0.104567	-0.138803	0.546210	0.046217	
popularity	-0.145115	0.108046	0.141205	-0.057272	
duration_ms	0.156455	0.039128	-0.220045	0.148876	

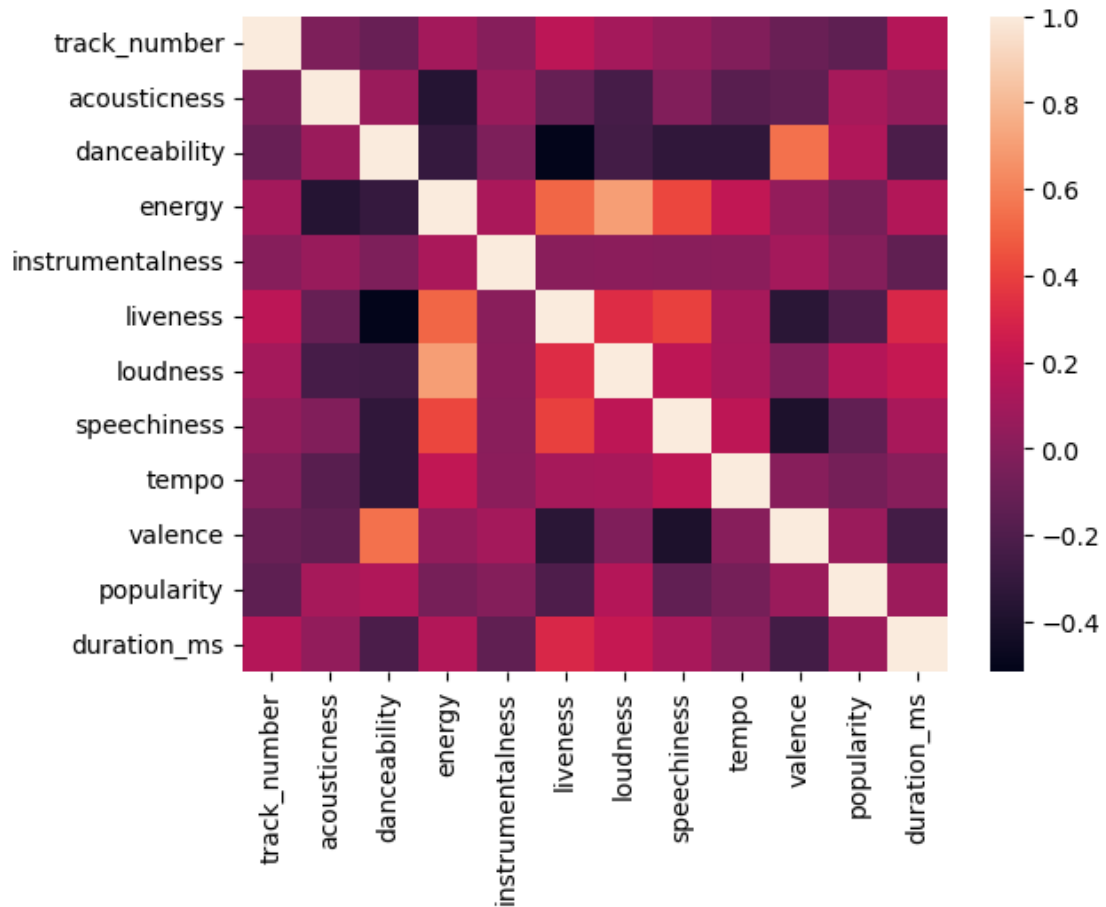
	instrumentalness	liveness	loudness	speechiness	tempo	\
track_number	-0.002772	0.188351	0.100835	0.040617	-0.023934	
acousticness	0.061403	-0.117739	-0.237083	-0.021774	-0.171003	
danceability	-0.031812	-0.516387	-0.249406	-0.322684	-0.324398	
energy	0.120261	0.511188	0.698039	0.417214	0.201885	
instrumentalness	1.000000	0.008873	0.012524	0.009586	0.010961	
liveness	0.008873	1.000000	0.327036	0.400018	0.108855	
loudness	0.012524	0.327036	1.000000	0.189904	0.112837	
speechiness	0.009586	0.400018	0.189904	1.000000	0.192687	

tempo	0.010961	0.108855	0.112837	0.192687	1.000000
valence	0.103480	-0.347451	-0.027571	-0.399751	0.000558
popularity	-0.010612	-0.205845	0.156323	-0.136745	-0.061061
duration_ms	-0.137599	0.304735	0.221558	0.114546	0.001465

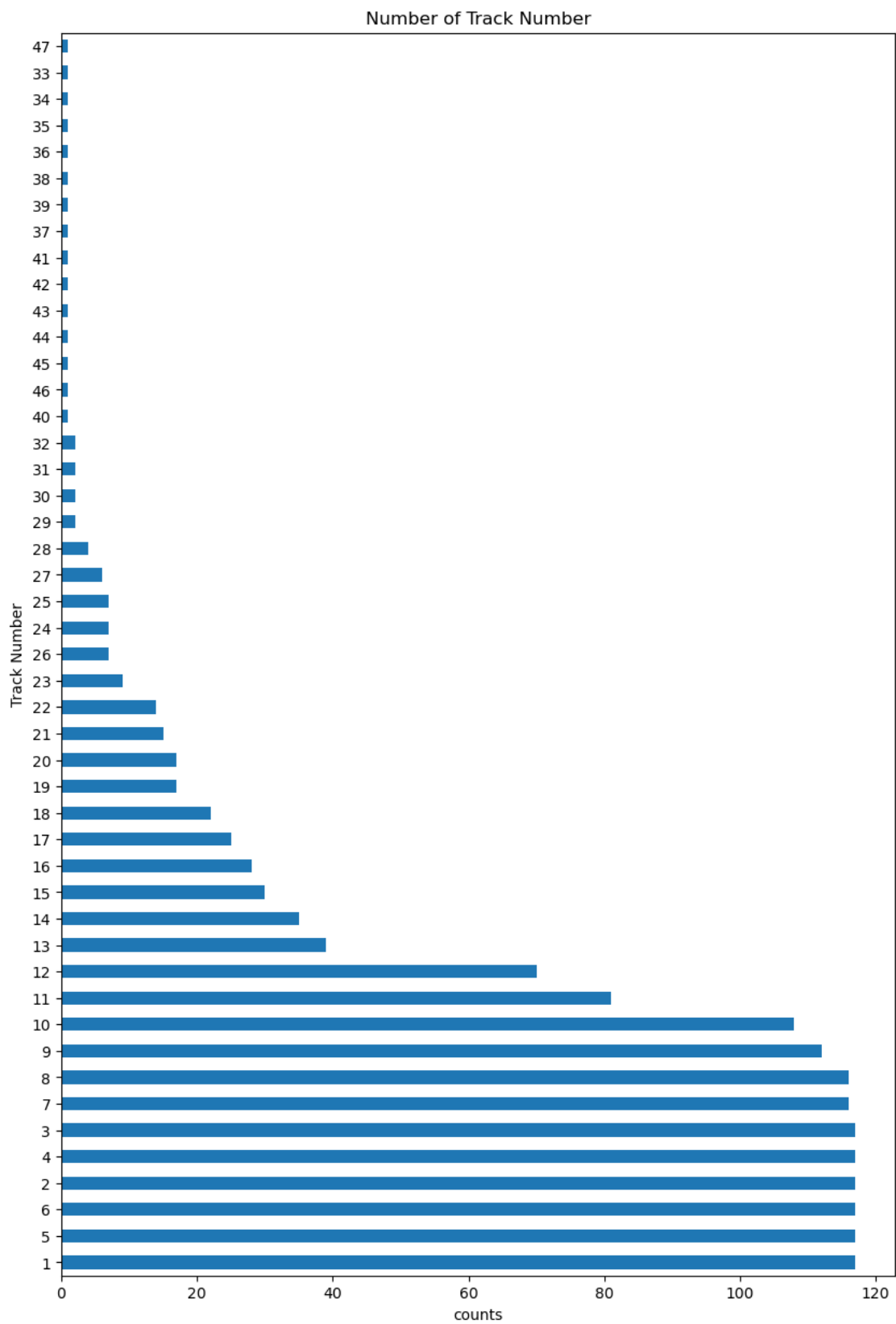
	valence	popularity	duration_ms
track_number	-0.104567	-0.145115	0.156455
acousticness	-0.138803	0.108046	0.039128
danceability	0.546210	0.141205	-0.220045
energy	0.046217	-0.057272	0.148876
instrumentalness	0.103480	-0.010612	-0.137599
liveness	-0.347451	-0.205845	0.304735
loudness	-0.027571	0.156323	0.221558
speechiness	-0.399751	-0.136745	0.114546
tempo	0.000558	-0.061061	0.001465
valence	1.000000	0.065333	-0.244833
popularity	0.065333	1.000000	0.074102
duration_ms	-0.244833	0.074102	1.000000

```
[28]: sns.heatmap(df2.corr())
```

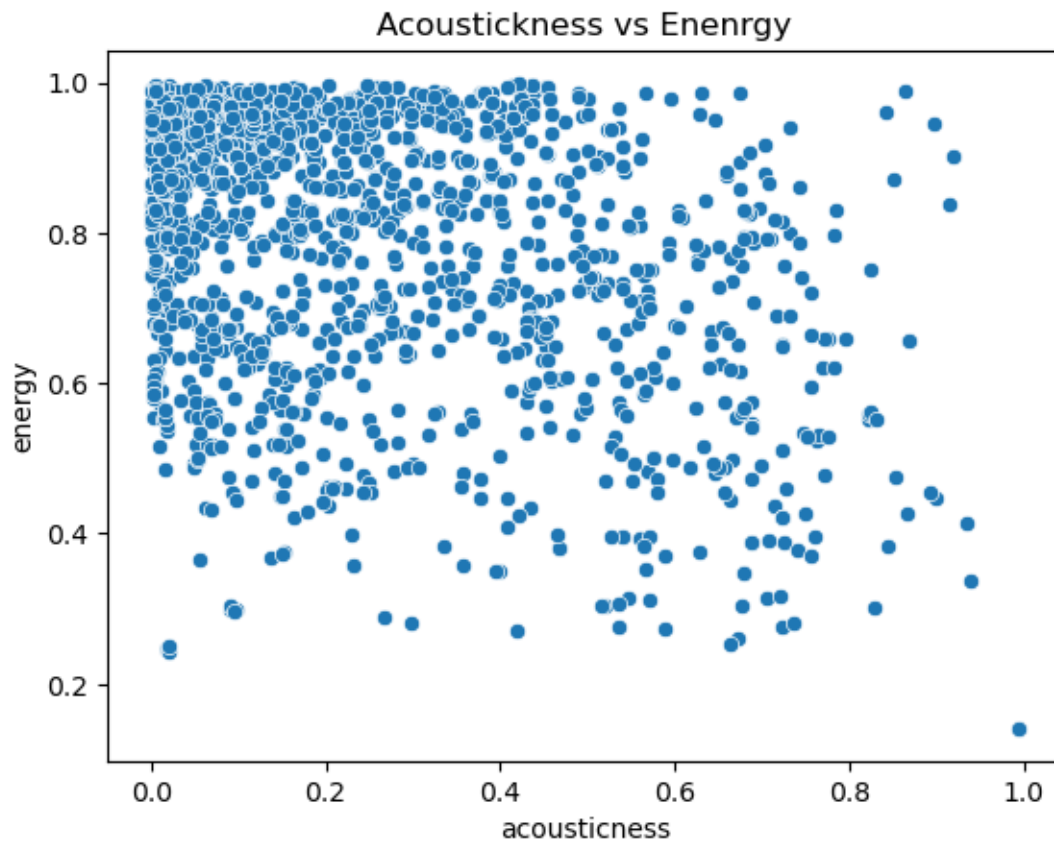
```
[28]: <Axes: >
```



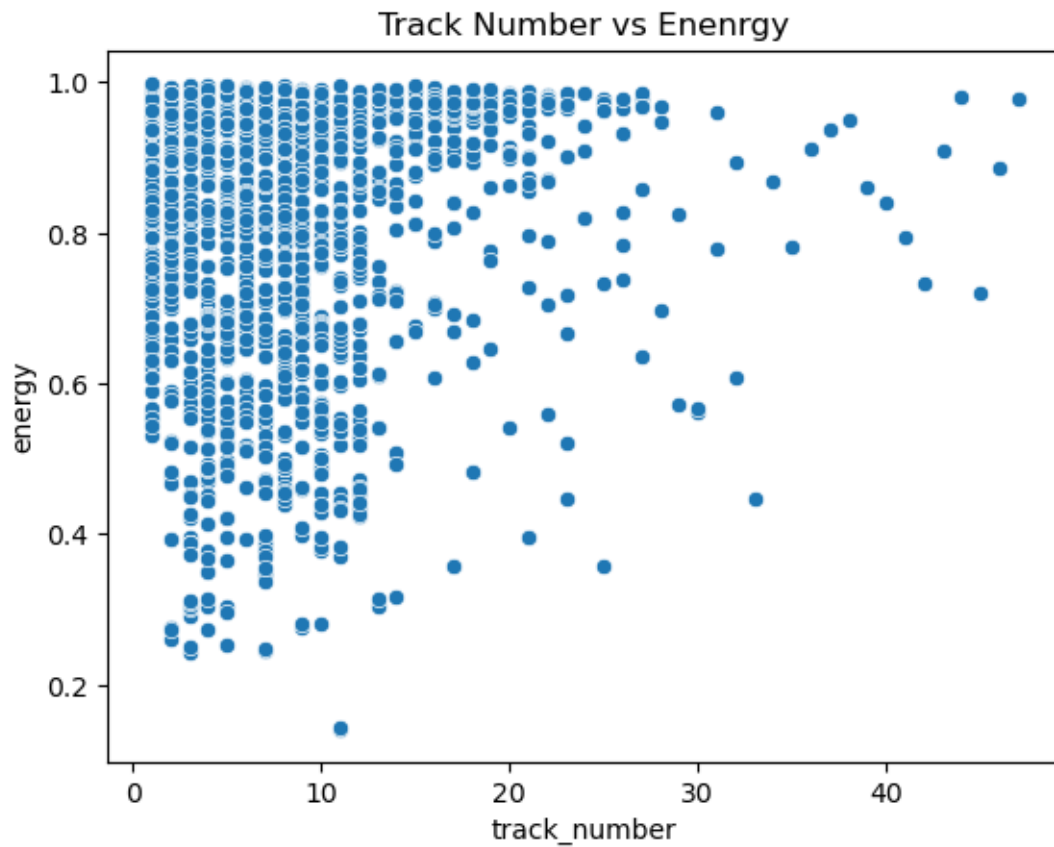
```
[29]: df1['track_number'].value_counts().plot(kind='barh', figsize=(10,15))
plt.xlabel('counts')
plt.ylabel('Track Number')
plt.title('Number of Track Number')
plt.show()
```



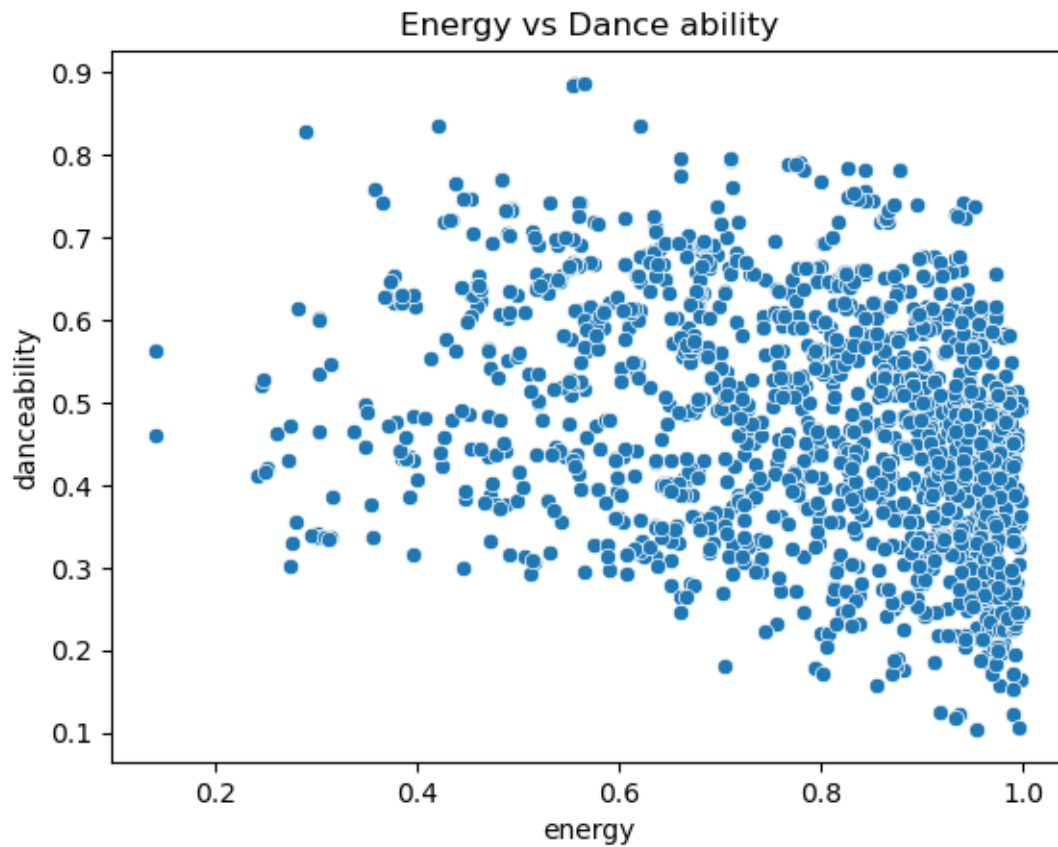

```
[30]: sns.scatterplot(x=df['acoustickness'],y=df['energy'])  
plt.title('Acoustickness vs Enerngy')  
plt.show()
```



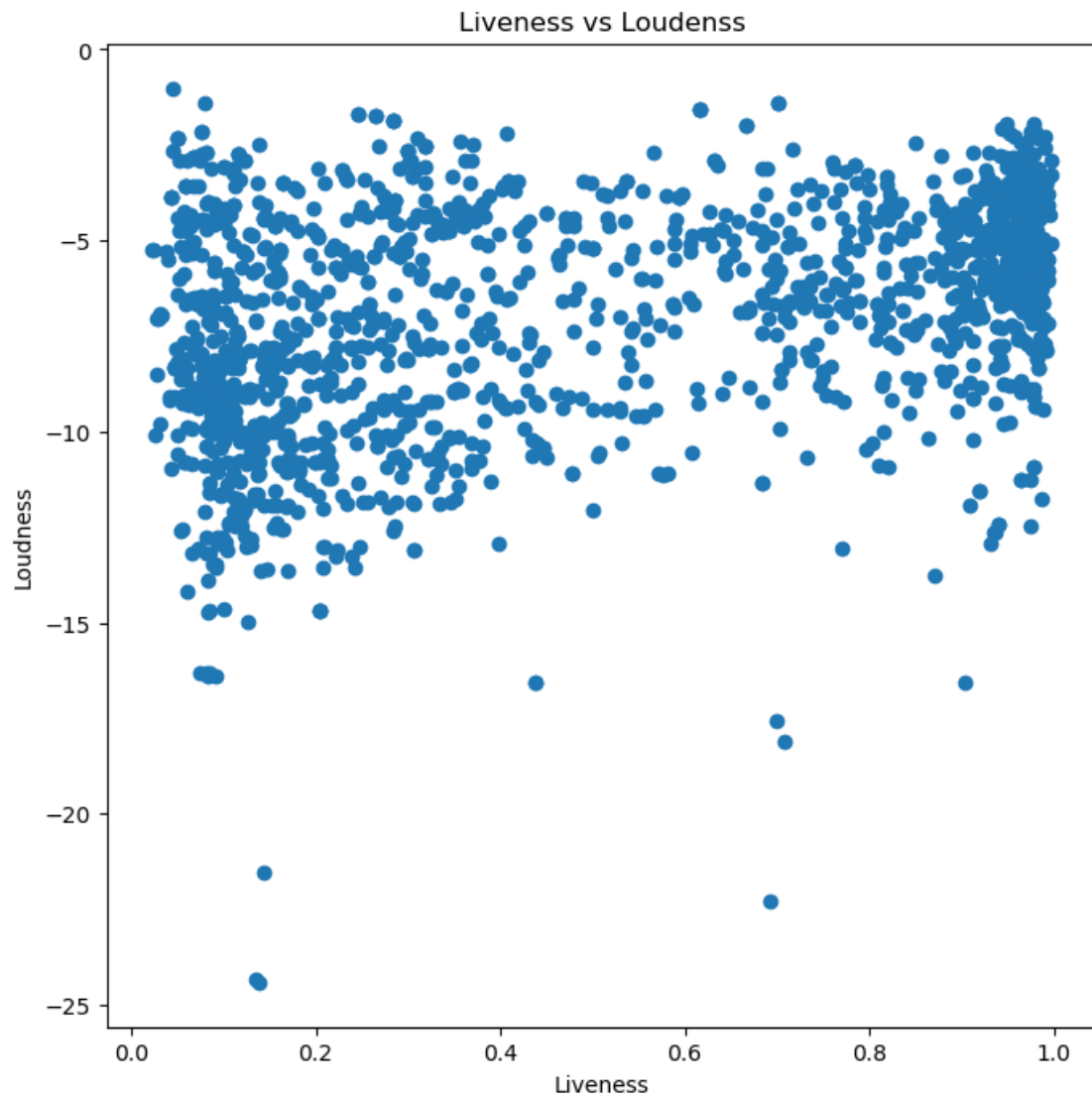
```
[31]: sns.scatterplot(x=df['track_number'],y=df['energy'])  
plt.title('Track Number vs Enerngy')  
plt.show()
```



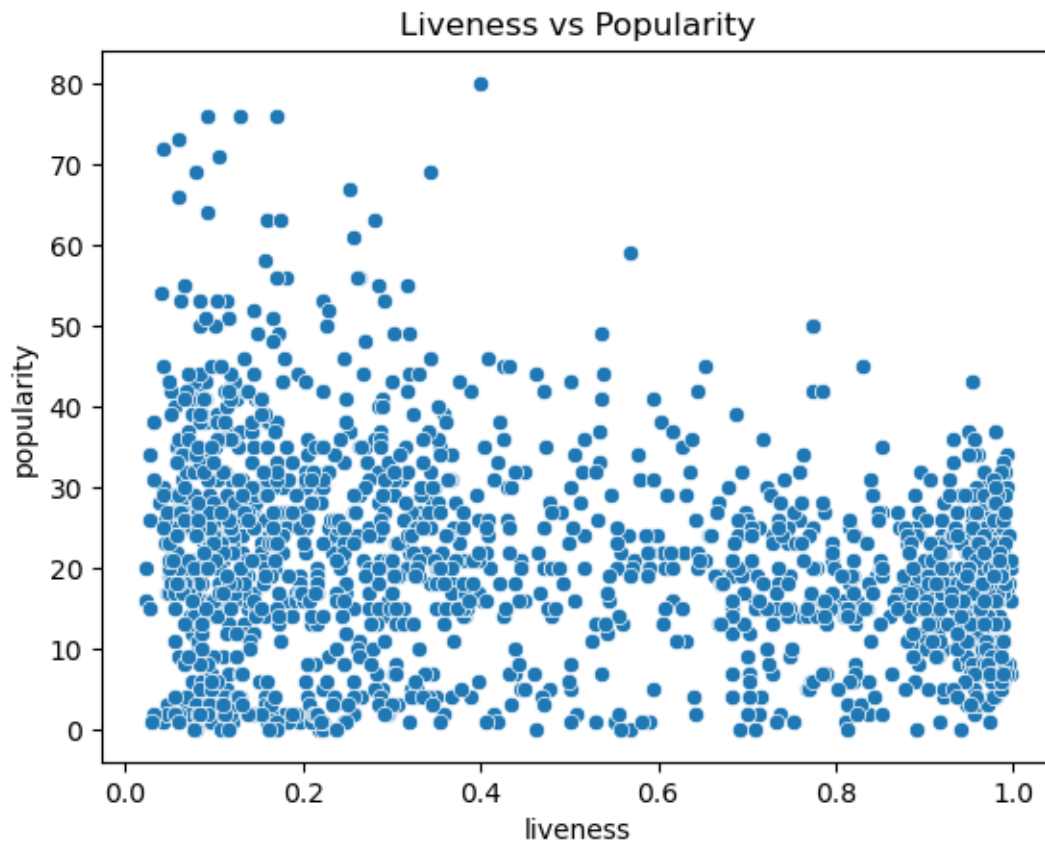
```
[32]: sns.scatterplot(x=df['energy'],y=df['danceability'])  
plt.title('Energy vs Dance ability')  
plt.show()
```



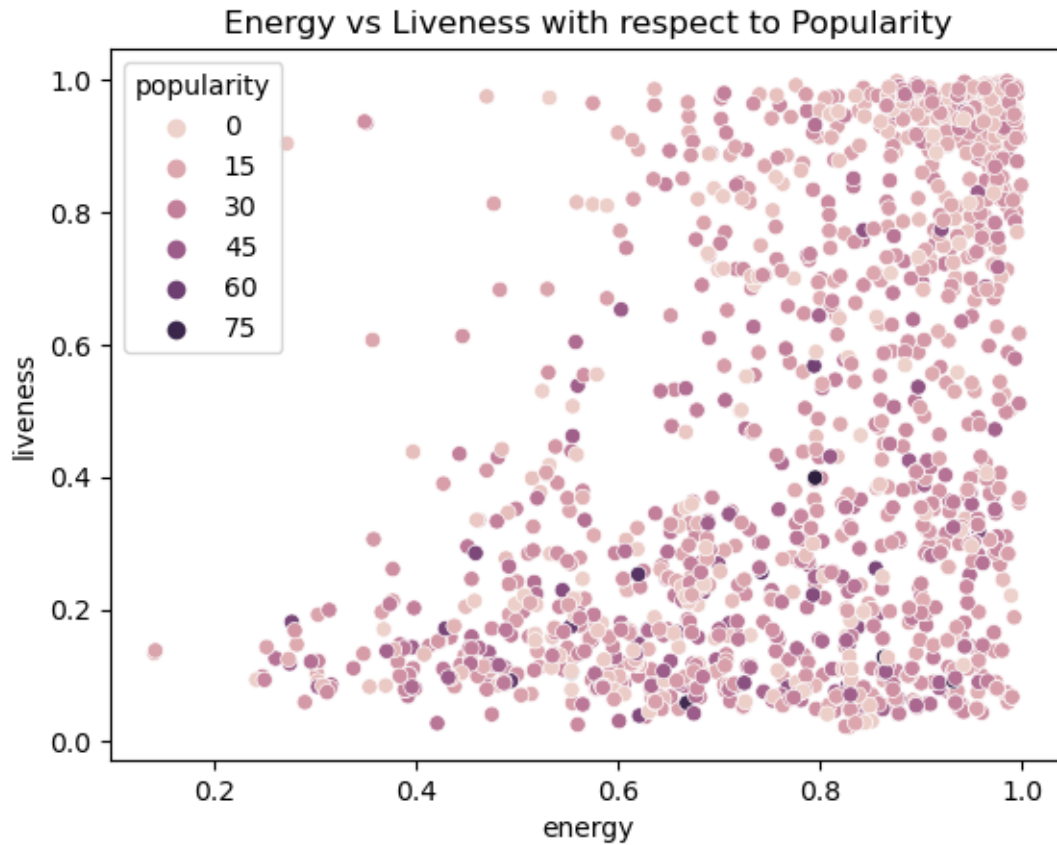
```
[33]: plt.figure(figsize=(8,8),dpi=100)
plt.scatter(x=df['liveness'],y=df['loudness'])
plt.xlabel('Liveness')
plt.ylabel('Loudness')
plt.title('Liveness vs Loudenss')
plt.show()
```



```
[34]: sns.scatterplot(x=df['liveness'],y=df['popularity'])  
plt.title('Liveness vs Popularity')  
plt.show()
```

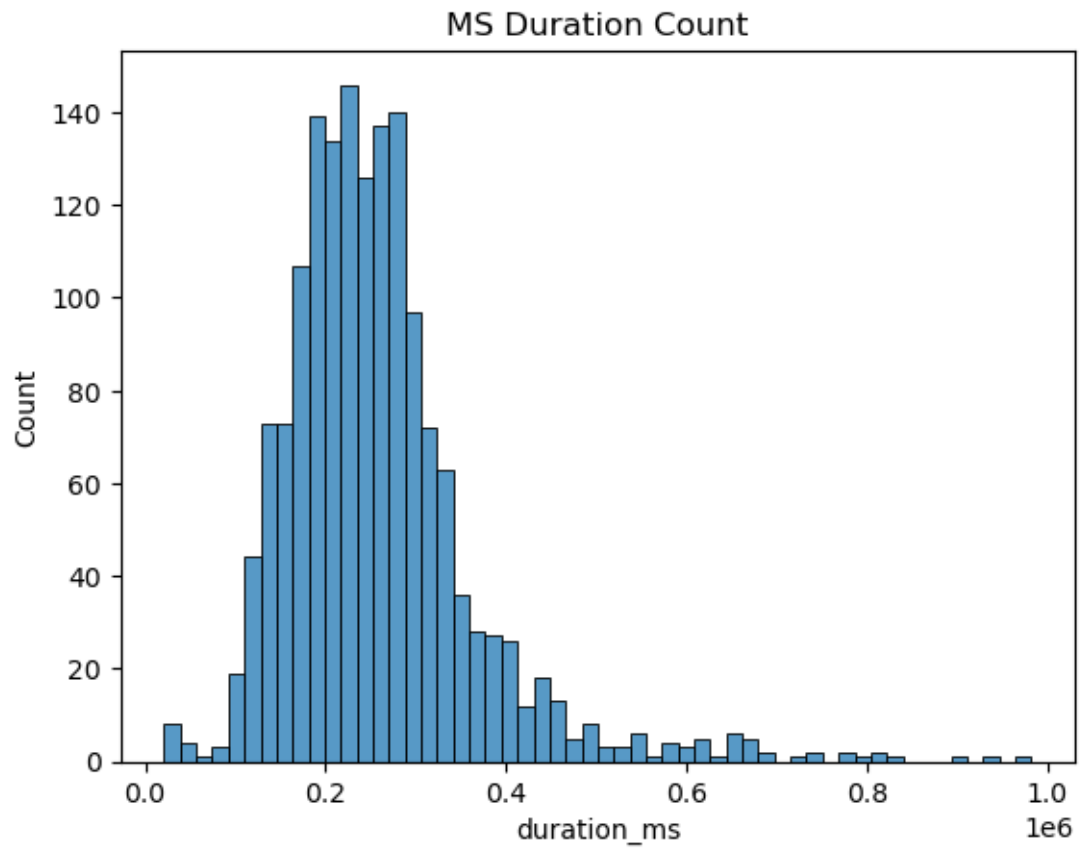


```
[35]: sns.scatterplot(x=df['energy'],y=df['liveness'],hue=df['popularity'])  
plt.title('Energy vs Liveness with respect to Popularity')  
plt.show()
```

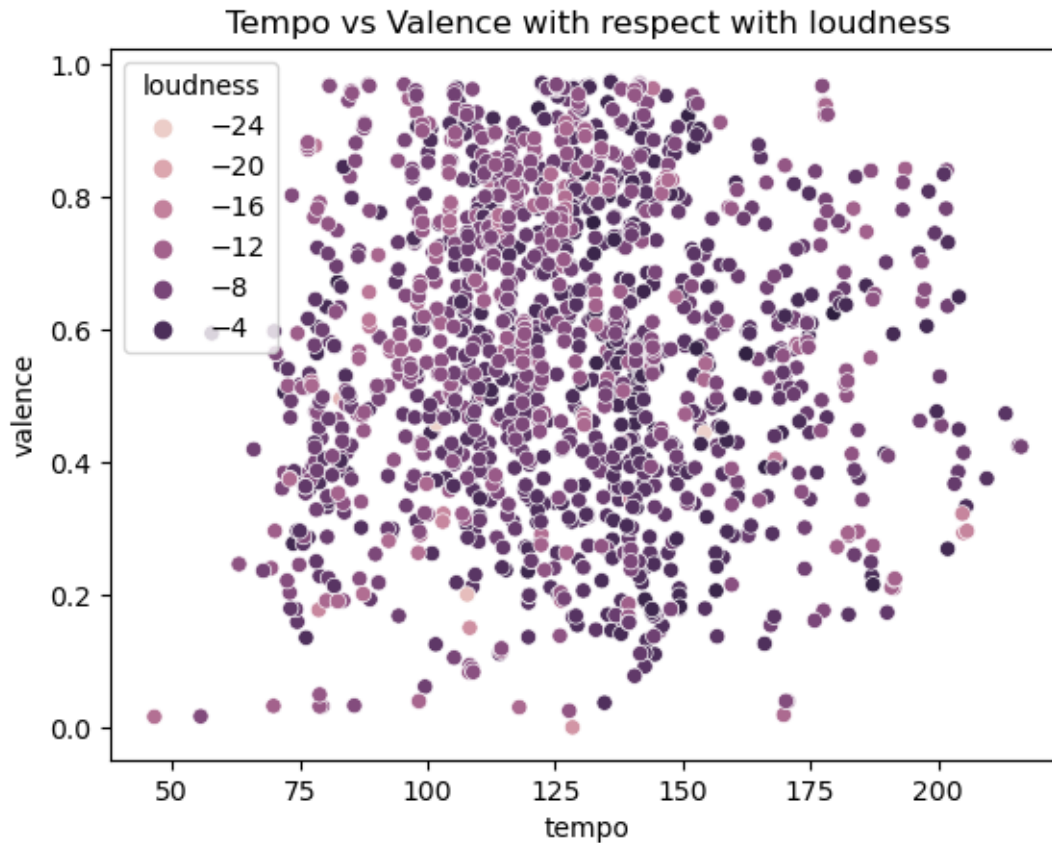


```
[37]: sns.histplot(df['duration_ms'])  
plt.title('MS Duration Count')  
plt.show()
```

C:\Users\soura\anaconda3\Lib\site-packages\seaborn_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
with pd.option_context('mode.use_inf_as_na', True):

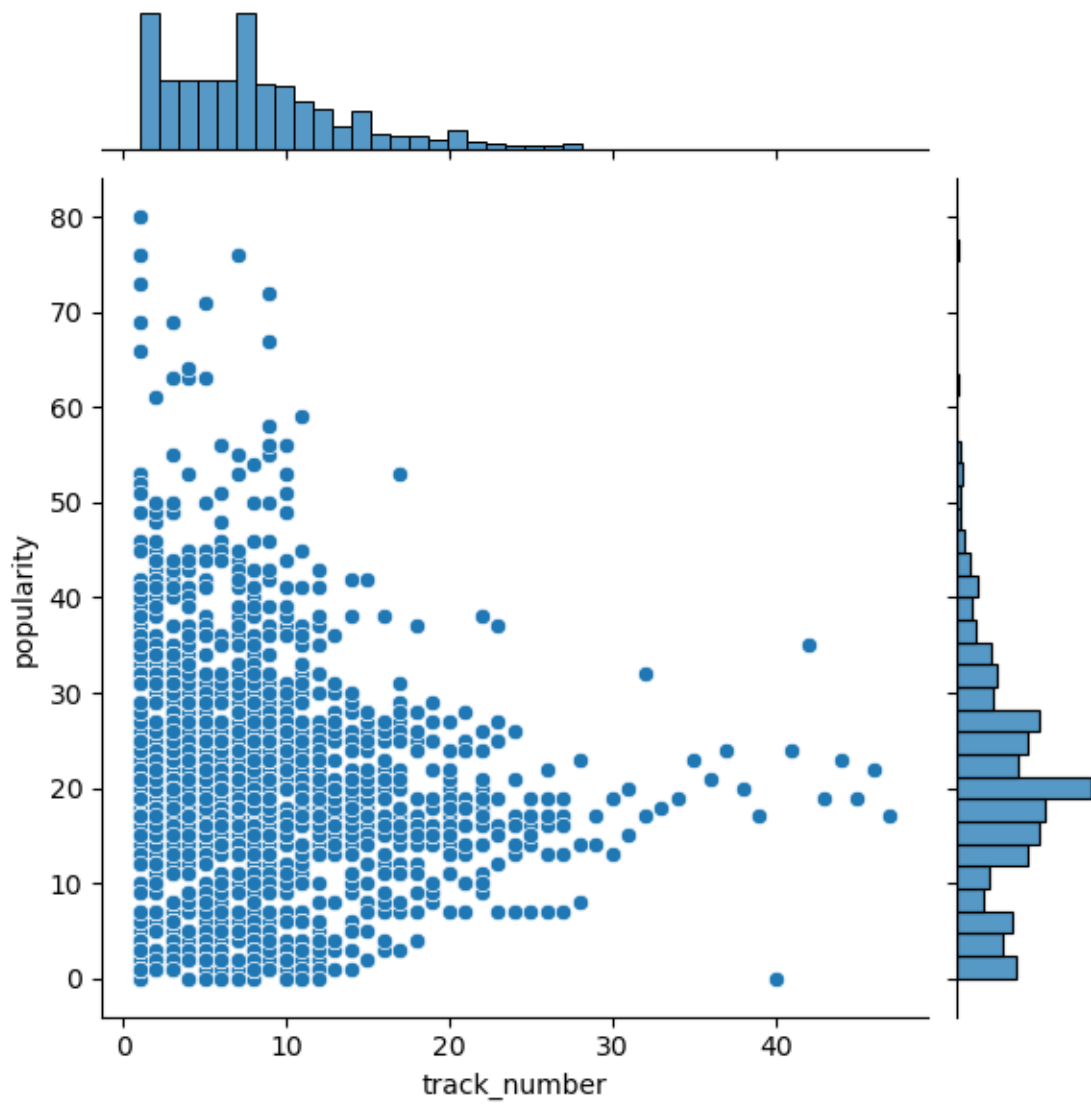


```
[38]: sns.scatterplot(x=df['tempo'],y=df['valence'],hue=df['loudness'])  
plt.title('Tempo vs Valence with respect with loudness')  
plt.show()
```

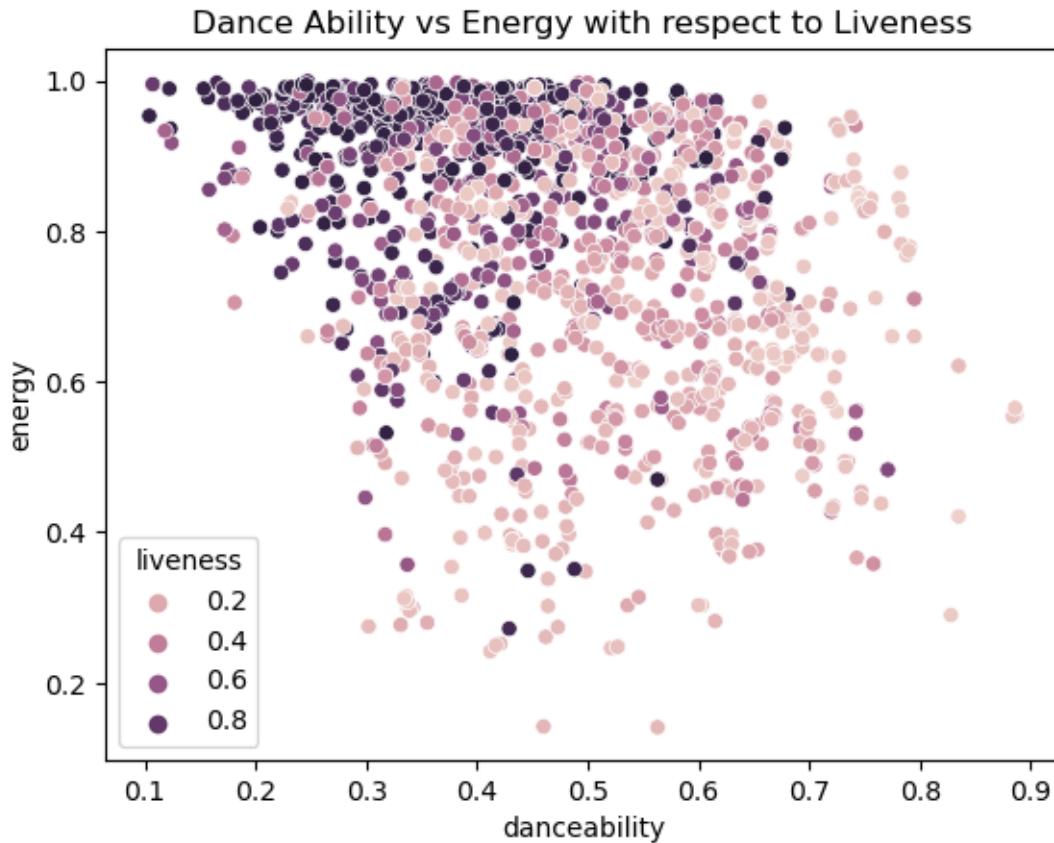


```
[39]: sns.jointplot(x=df['track_number'],y=df['popularity'])
plt.show()
```

```
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

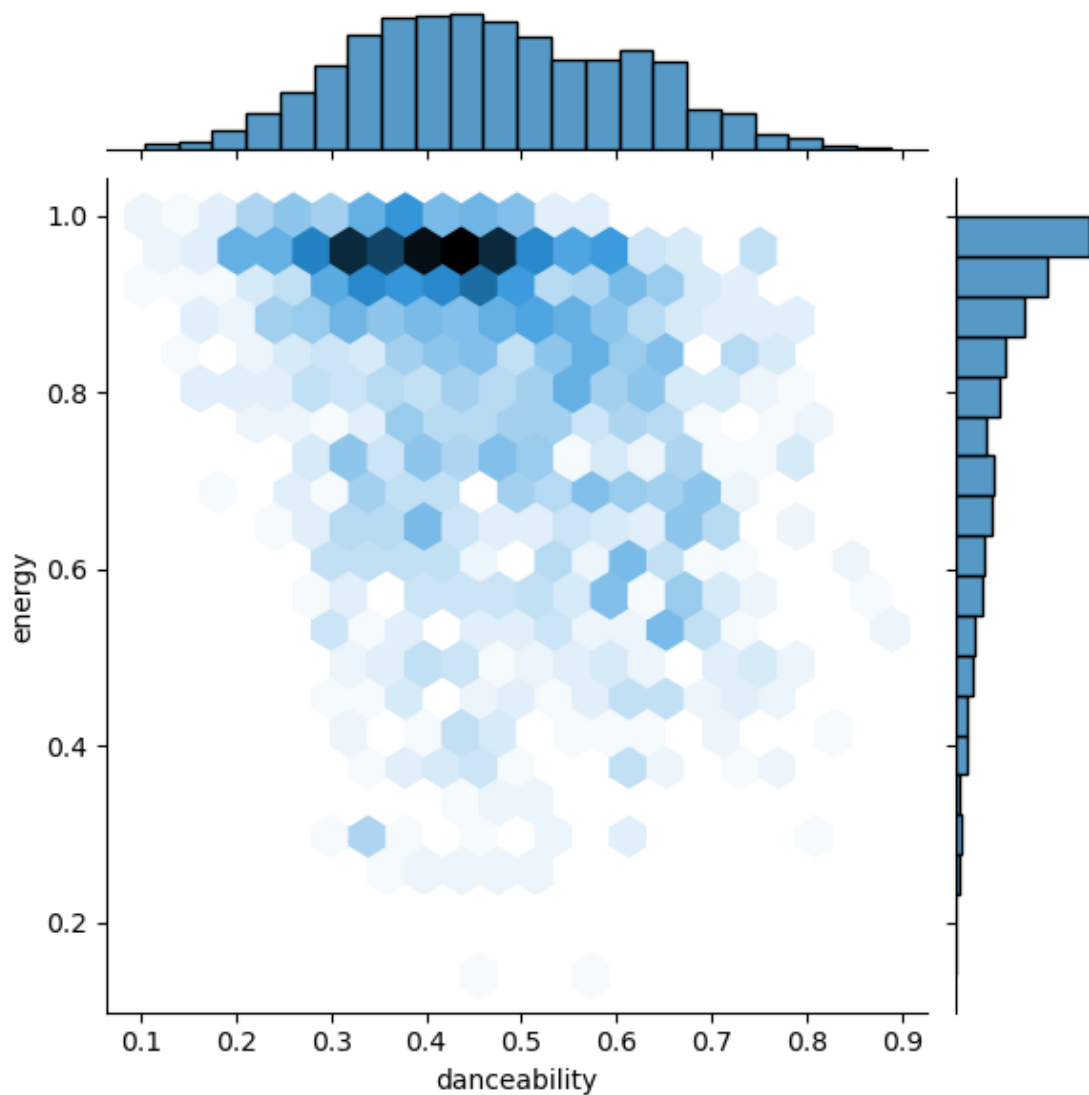



```
[40]: sns.scatterplot(x=df['danceability'],y=df['energy'],hue=df['liveness'])
plt.title('Dance Ability vs Energy with respect to Liveness')
plt.show()
```



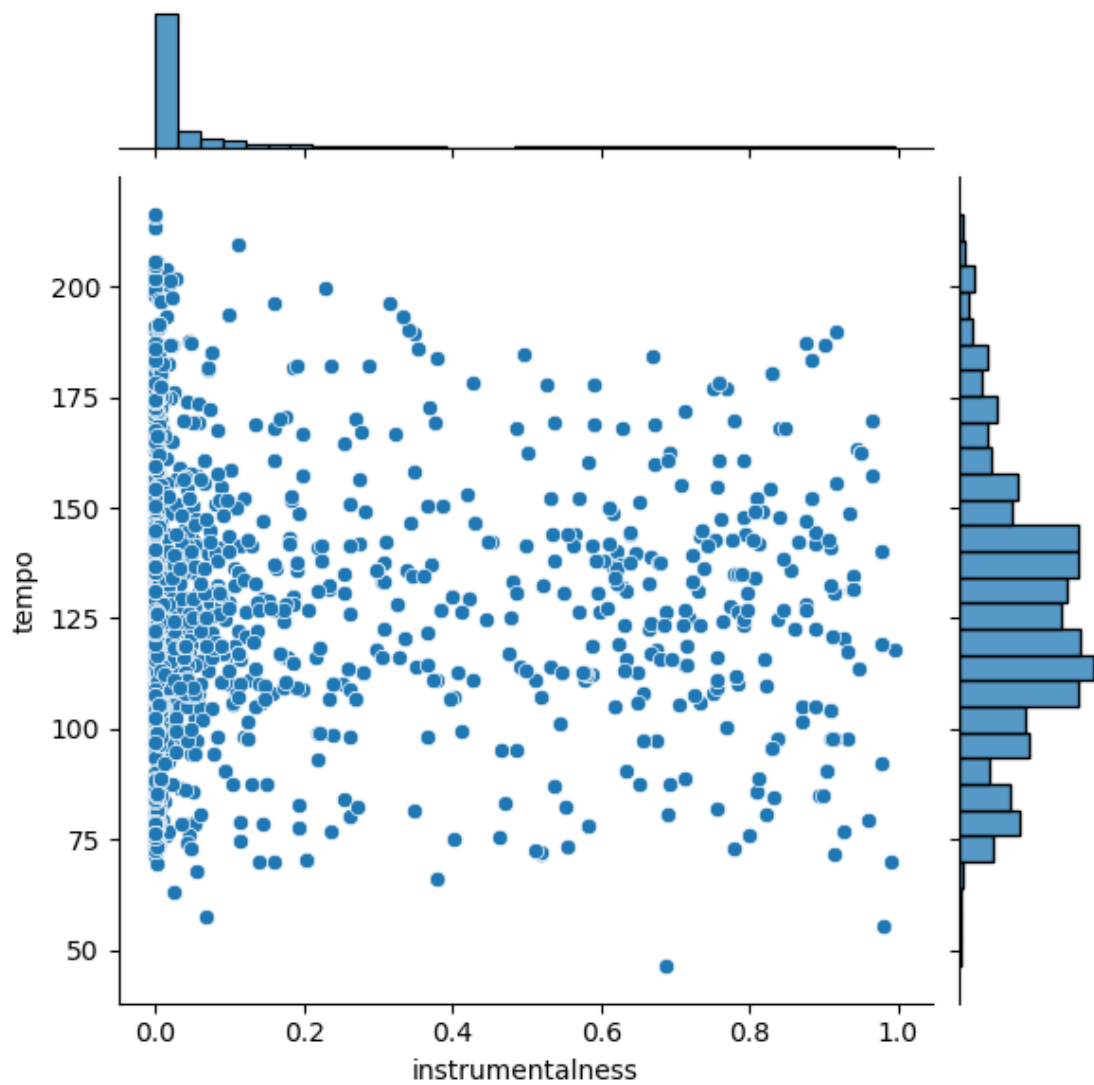
```
[41]: sns.jointplot(x=df['danceability'],y=df['energy'],kind='hex')  
plt.show()
```

```
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:  
FutureWarning: use_inf_as_na option is deprecated and will be removed in a  
future version. Convert inf values to NaN before operating instead.  
    with pd.option_context('mode.use_inf_as_na', True):  
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:  
FutureWarning: use_inf_as_na option is deprecated and will be removed in a  
future version. Convert inf values to NaN before operating instead.  
    with pd.option_context('mode.use_inf_as_na', True):
```

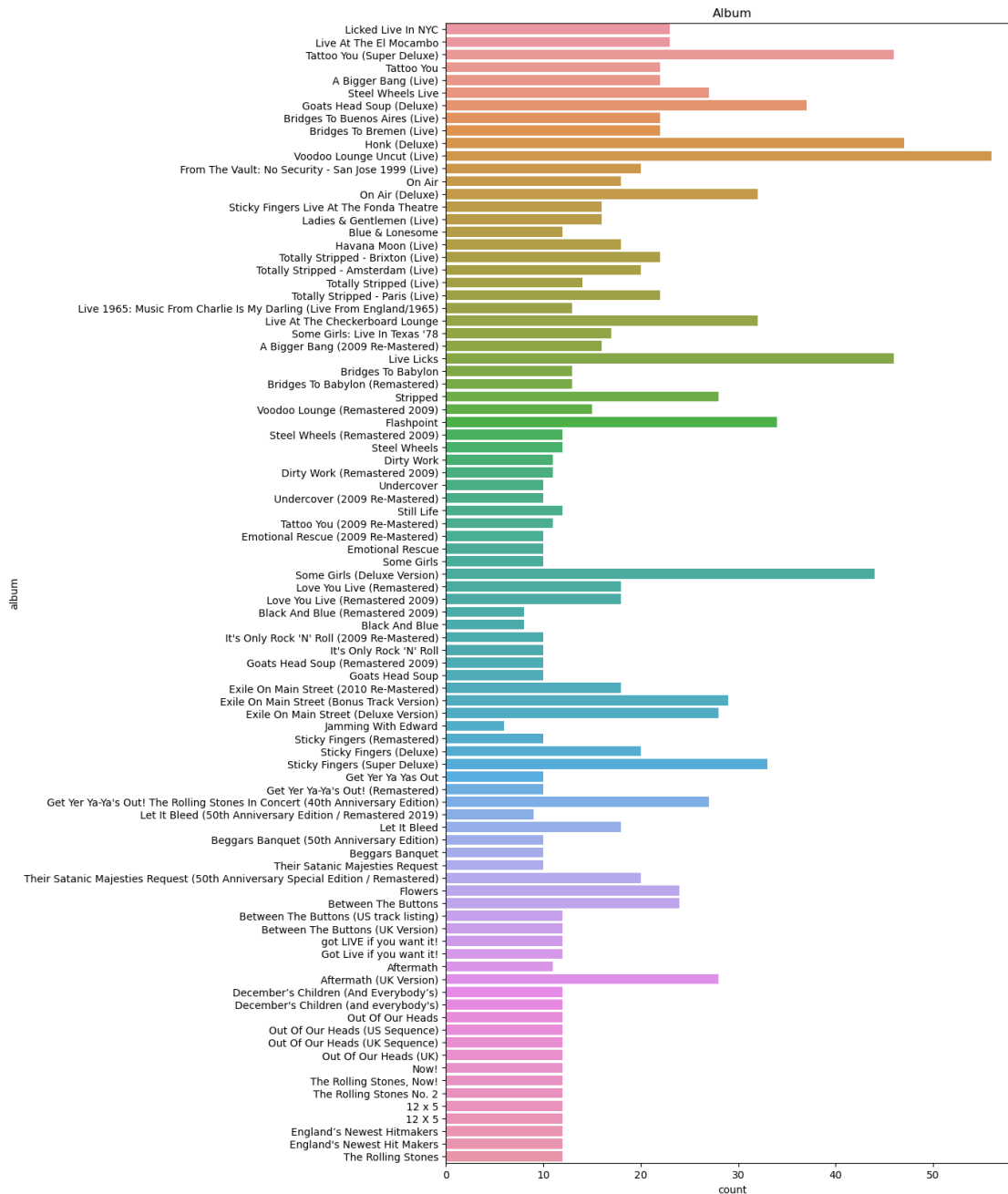


```
[42]: sns.jointplot(x=df['instrumentalness'],y=df['tempo'])
plt.show()
```

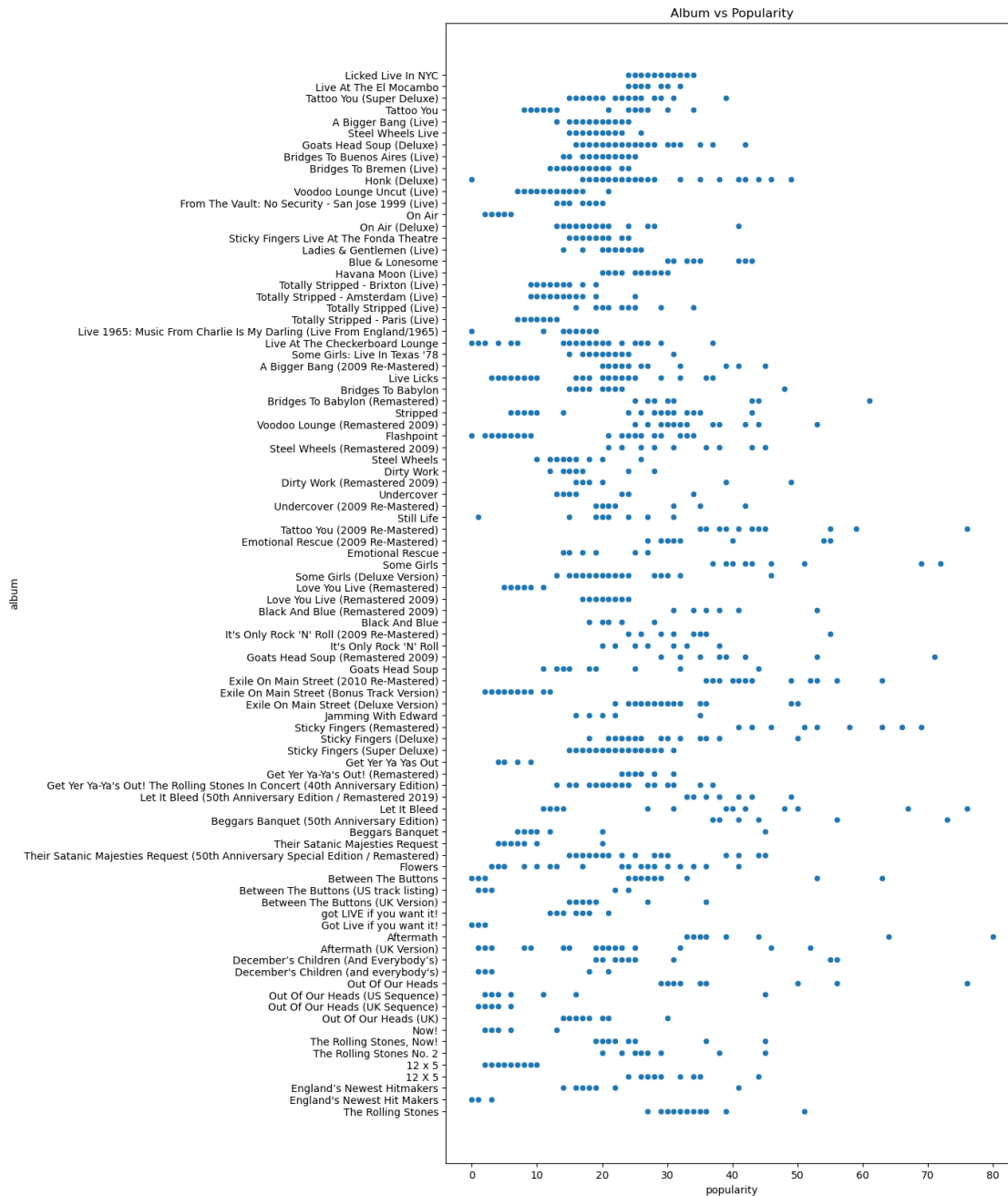
```
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



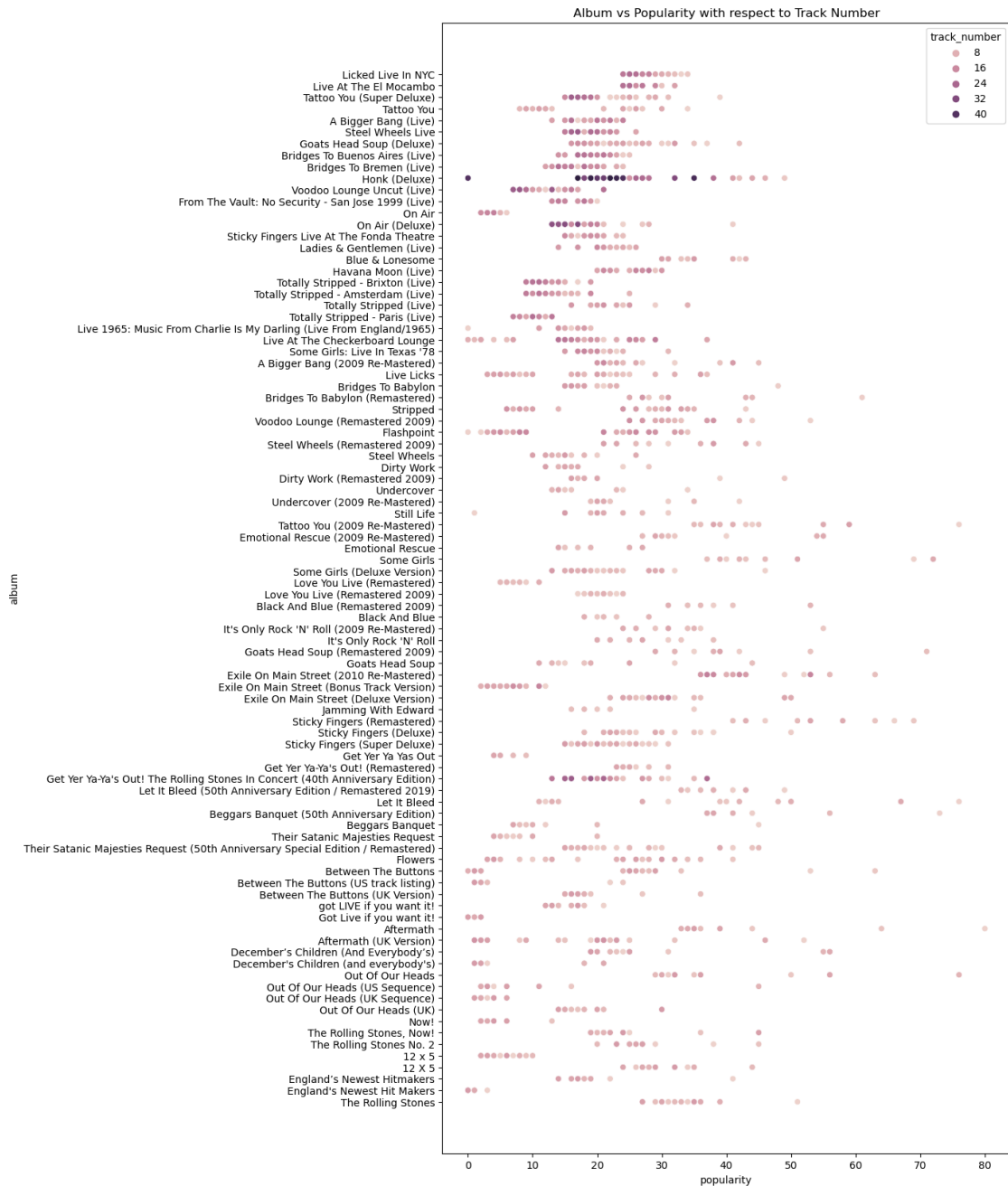
```
[43]: plt.figure(figsize=(10,20))
sns.countplot(y=df['album'])
plt.title('Album')
plt.show()
```



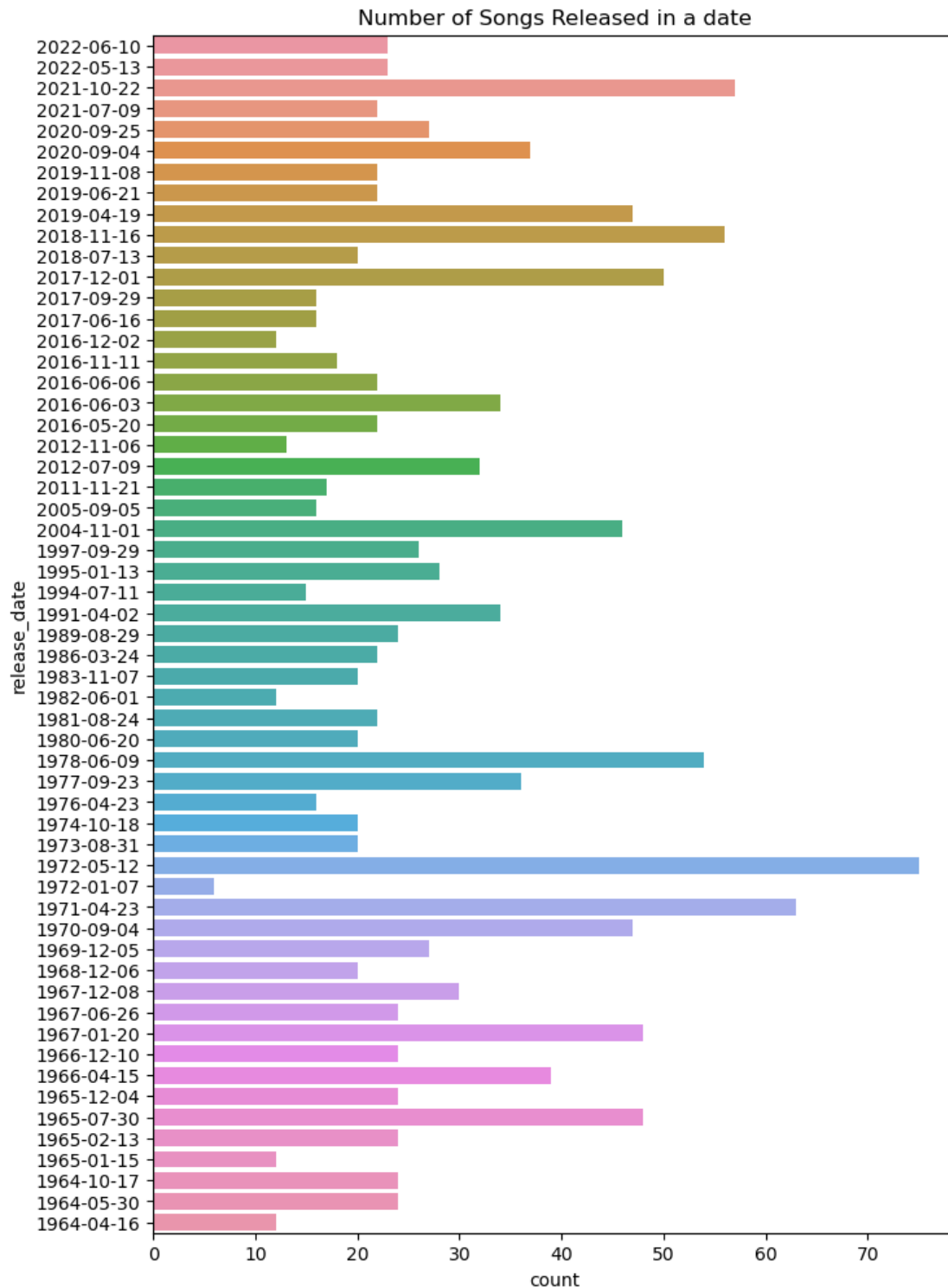
```
[44]: plt.figure(figsize=(10,20))
sns.scatterplot(y=df['album'],x=df['popularity'])
plt.title('Album vs Popularity')
plt.show()
```



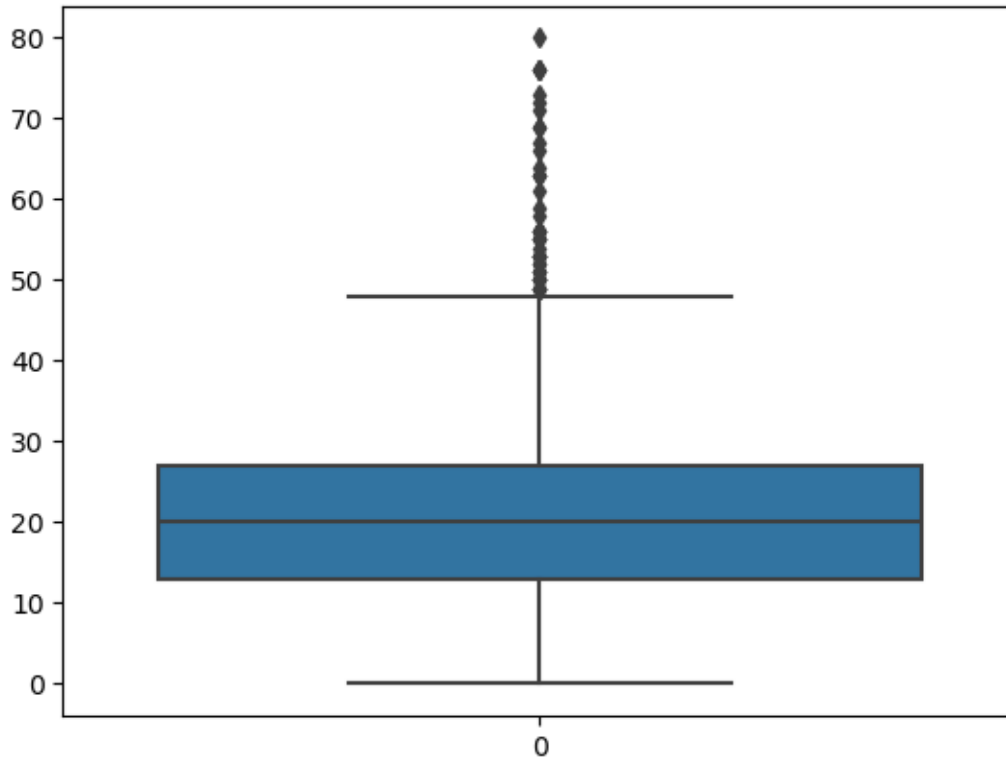
```
[45]: plt.figure(figsize=(10,20))
sns.scatterplot(y=df['album'],x=df['popularity'],hue=df['track_number'])
plt.title('Album vs Popularity with respect to Track Number')
plt.show()
```



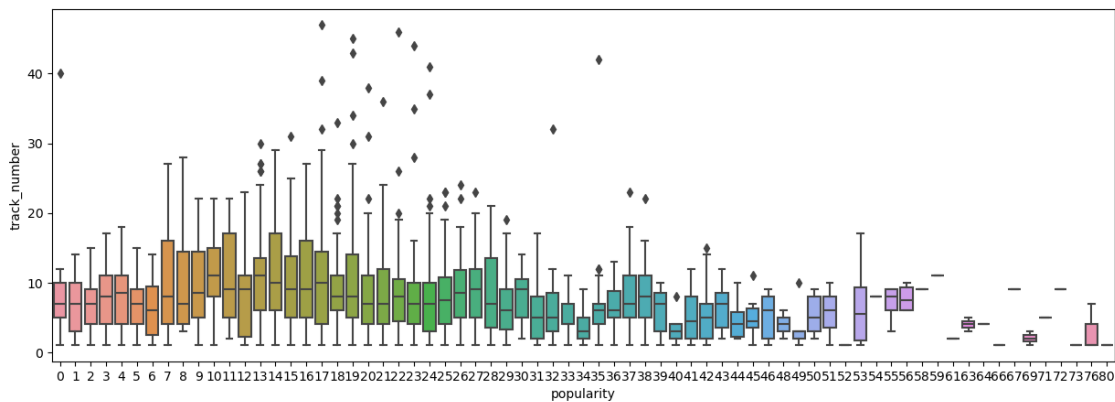
```
[46]: plt.figure(figsize=(8,12))
sns.countplot(y=df['release_date'])
plt.title('Number of Songs Released in a date')
plt.show()
```



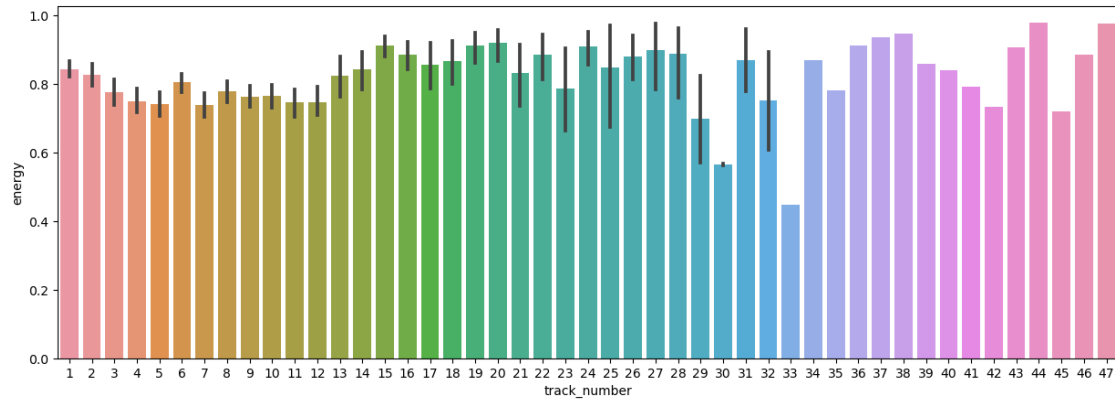
```
[47]: sns.boxplot(df['popularity'])
plt.show()
```

```
[48]: plt.figure(figsize=(15,5))
sns.boxplot(x=df['popularity'],y=df['track_number'])
plt.show()
```

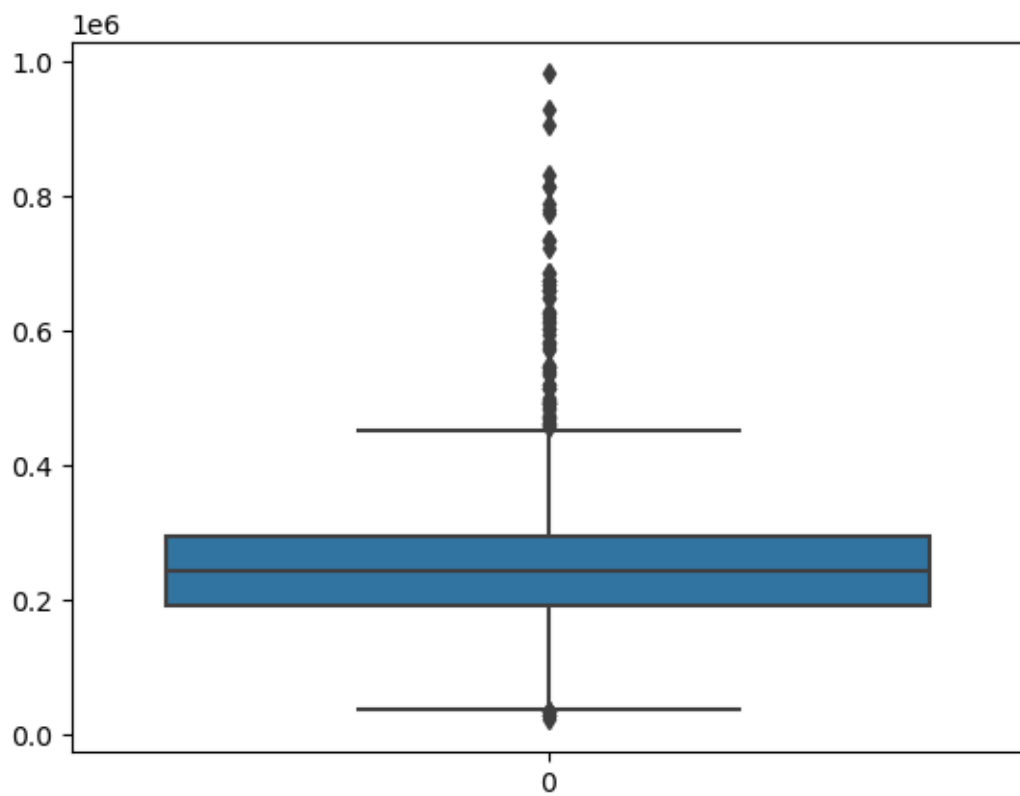


```
[49]: plt.figure(figsize=(15,5))
sns.barplot(x=df['track_number'],y=df['energy'])
plt.show()
```



```
[50]: sns.boxplot(df['duration_ms'])
```

```
[50]: <Axes: >
```

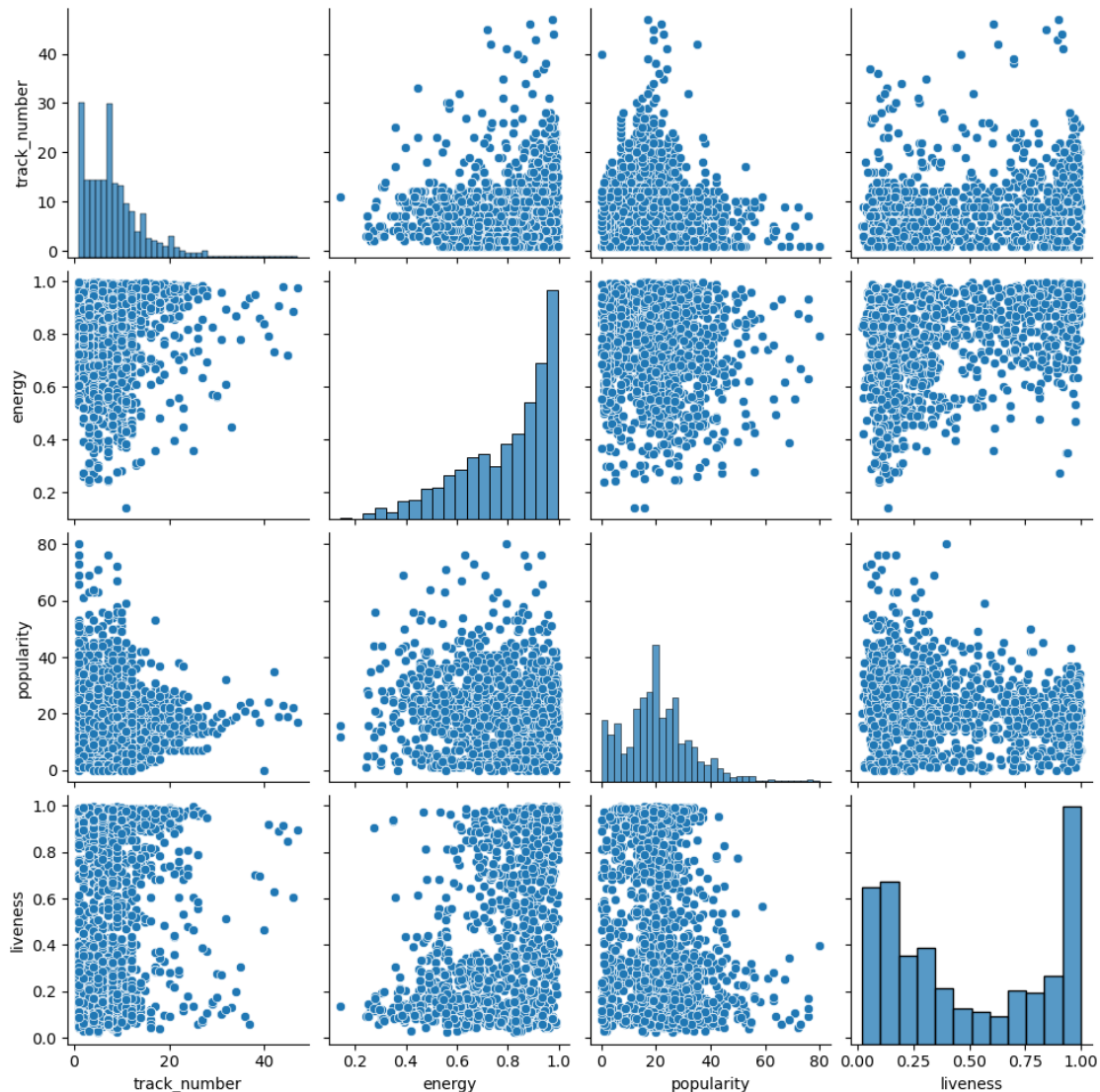


```
[51]: cols = ['track_number', 'energy', 'popularity', 'liveness']
sns.pairplot(df, vars=cols)
plt.show()
```

```

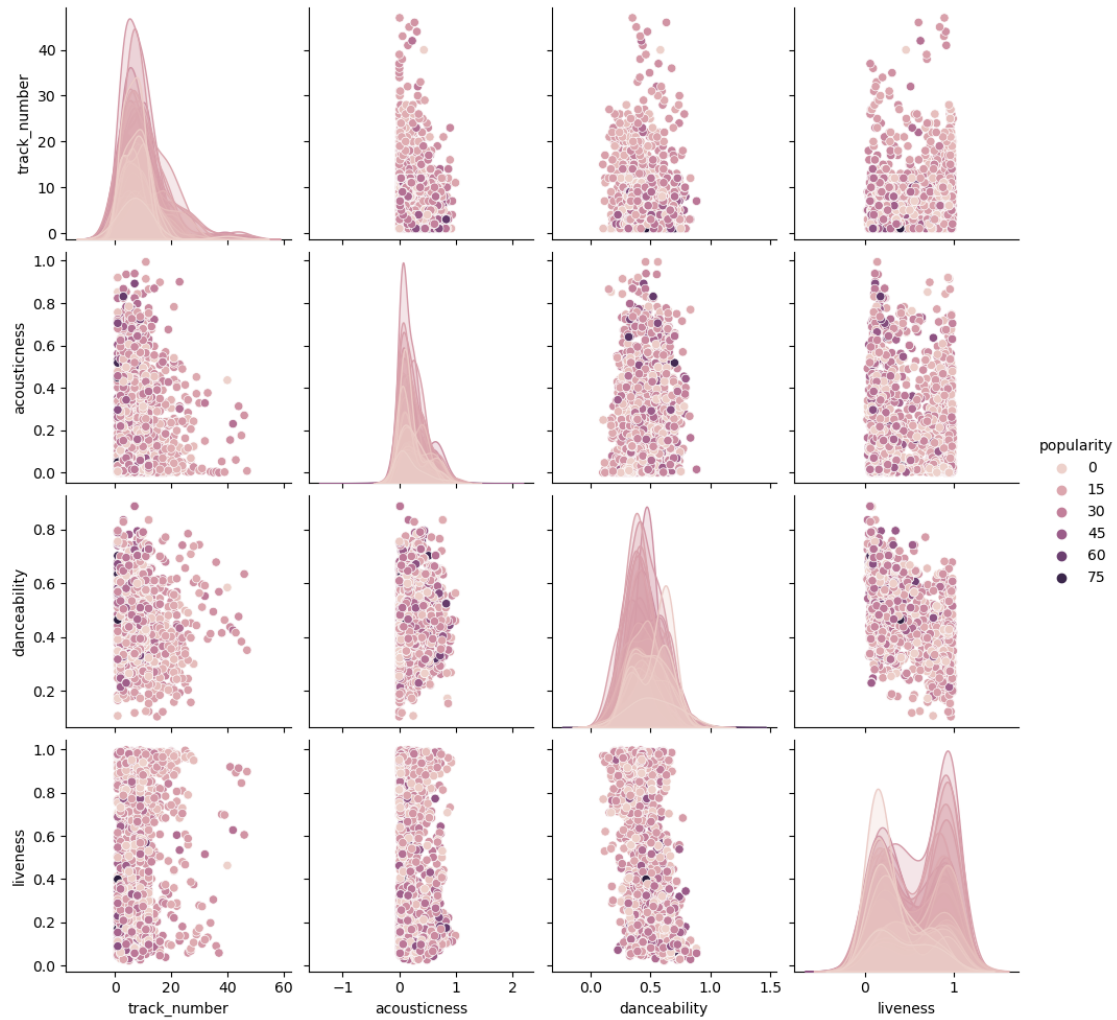
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):

```



```
[52]: cols = ['track_number', 'acousticness', 'danceability', 'liveness']
sns.pairplot(df, vars=cols, hue='popularity')
plt.show()
```

```
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
C:\Users\soura\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
```



```
[54]: plt.figure(figsize=(8,15))
plt.subplots_adjust(hspace=0.5,wspace=0.5)

plt.subplot(5,2,1)
plt.hist(df['energy'])
plt.title('Energy')

plt.subplot(5,2,2)
plt.hist(df['track_number'])
plt.title('Track Number')

plt.subplot(5,2,4)
plt.hist(df['popularity'])
plt.title('Popularity')
```

```
plt.subplot(5,2,4)
plt.hist(df['liveness'])
plt.title('Liveness')

plt.subplot(5,2,5)
plt.hist(df['tempo'])
plt.title('Tempo')

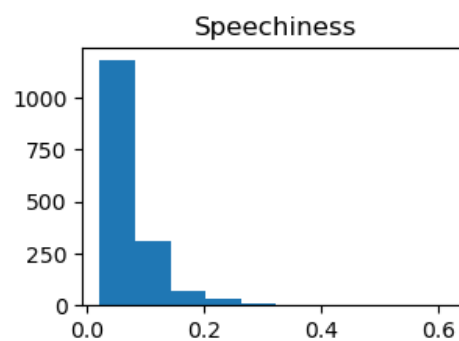
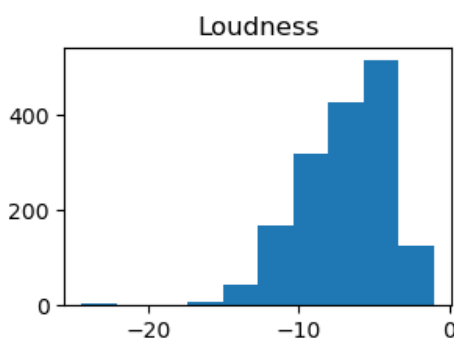
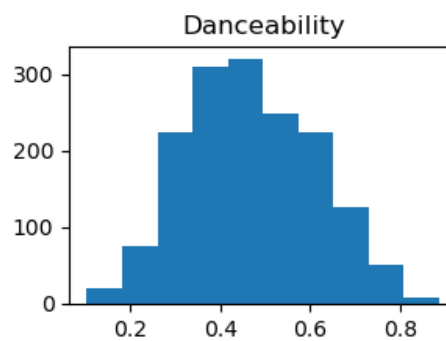
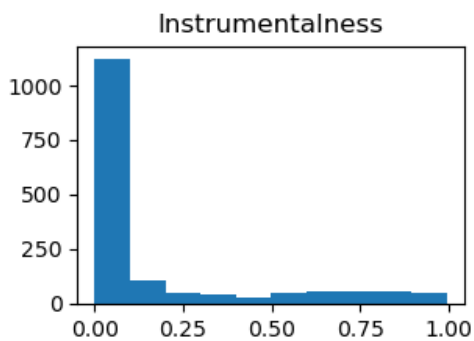
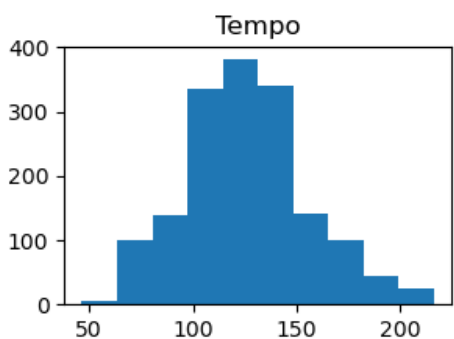
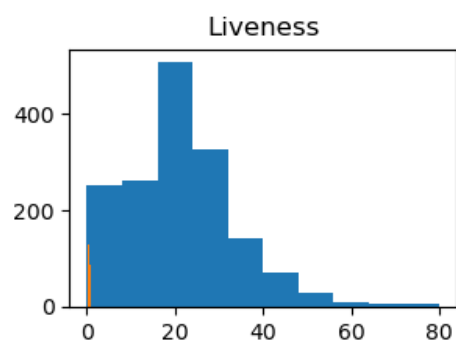
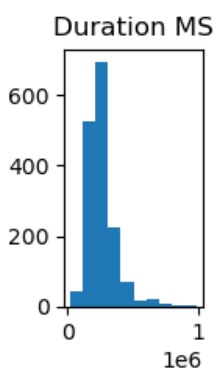
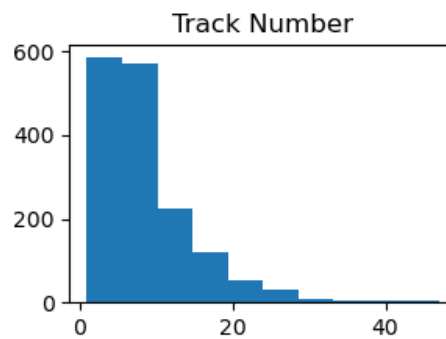
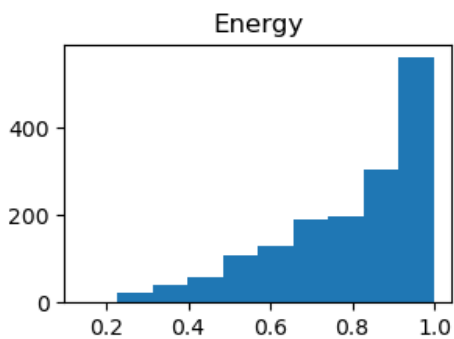
plt.subplot(5,5,6)
plt.hist(df['duration_ms'])
plt.title('Duration MS')

plt.subplot(5,2,7)
plt.hist(df['instrumentalness'])
plt.title('Instrumentalness')

plt.subplot(5,2,8)
plt.hist(df['danceability'])
plt.title('Danceability')

plt.subplot(5,2,9)
plt.hist(df['loudness'])
plt.title('Loudness')

plt.subplot(5,2,10)
plt.hist(df['speechiness'])
plt.title('Speechiness')
plt.show()
```



4 cluster analysis

[56]: df1

```
[56]:
```

		name	album	release_date	\
0		Concert Intro Music - Live	Licked Live In NYC	2022-06-10	
1		Street Fighting Man - Live	Licked Live In NYC	2022-06-10	
2		Start Me Up - Live	Licked Live In NYC	2022-06-10	
3		If You Can't Rock Me - Live	Licked Live In NYC	2022-06-10	
4		Don't Stop - Live	Licked Live In NYC	2022-06-10	
...		
1605		Carol	The Rolling Stones	1964-04-16	
1606		Tell Me	The Rolling Stones	1964-04-16	
1607		Can I Get A Witness	The Rolling Stones	1964-04-16	
1608		You Can Make It If You Try	The Rolling Stones	1964-04-16	
1609		Walking The Dog	The Rolling Stones	1964-04-16	

	track_number	id	\
0	1	2IEkywLJ4ykbhi1yRQvmsT	
1	2	6GVgVJBKkGJoRfarYRvGTU	
2	3	1Lu761pZ0dBTGpzxaQoZNW	
3	4	1agTQzOTUnGNggycEqiDH	
4	5	7piGJR8YndQBQWVXv6KtQw	
...	
1605	8	08l7M5UpRnffG10FyuRiQZ	
1606	9	3JZ1lQBstM6WwoJdzFDLhx	
1607	10	0t2qvfsBQ3Y081zRRoVTdb	
1608	11	5ivIs5vwSjORCh0Iv1Y30n	
1609	12	43SkTJJ2xleDaeiE4TIM70	

	uri	acousticness	danceability	\
0	spotify:track:2IEkywLJ4ykbhi1yRQvmsT	0.0824	0.463	
1	spotify:track:6GVgVJBKkGJoRfarYRvGTU	0.4370	0.326	
2	spotify:track:1Lu761pZ0dBTGpzxaQoZNW	0.4160	0.386	
3	spotify:track:1agTQzOTUnGNggycEqiDH	0.5670	0.369	
4	spotify:track:7piGJR8YndQBQWVXv6KtQw	0.4000	0.303	
...	
1605	spotify:track:08l7M5UpRnffG10FyuRiQZ	0.1570	0.466	
1606	spotify:track:3JZ1lQBstM6WwoJdzFDLhx	0.0576	0.509	
1607	spotify:track:0t2qvfsBQ3Y081zRRoVTdb	0.3710	0.790	
1608	spotify:track:5ivIs5vwSjORCh0Iv1Y30n	0.2170	0.700	
1609	spotify:track:43SkTJJ2xleDaeiE4TIM70	0.3830	0.727	

	energy	instrumentalness	liveness	loudness	speechiness	tempo	\
--	--------	------------------	----------	----------	-------------	-------	---

0	0.993	0.996000	0.9320	-12.913	0.1100	118.001
1	0.965	0.233000	0.9610	-4.803	0.0759	131.455
2	0.969	0.400000	0.9560	-4.936	0.1150	130.066
3	0.985	0.000107	0.8950	-5.535	0.1930	132.994
4	0.969	0.055900	0.9660	-5.098	0.0930	130.533
...
1605	0.932	0.006170	0.3240	-9.214	0.0429	177.340
1606	0.706	0.000002	0.5160	-9.427	0.0843	122.015
1607	0.774	0.000000	0.0669	-7.961	0.0720	97.035
1608	0.546	0.000070	0.1660	-9.567	0.0622	102.634
1609	0.934	0.068500	0.0965	-8.373	0.0359	125.275

	valence	popularity	duration_ms
0	0.0302	33	48640
1	0.3180	34	253173
2	0.3130	34	263160
3	0.1470	32	305880
4	0.2060	32	305106
...
1605	0.9670	39	154080
1606	0.4460	36	245266
1607	0.8350	30	176080
1608	0.5320	27	121680
1609	0.9690	35	189186

[1610 rows x 17 columns]

```
[57]: df1.dtypes
```

```
[57]: name          object
      album        object
      release_date object
      track_number  int64
      id           object
      uri          object
      acousticness  float64
      danceability  float64
      energy        float64
      instrumentalness float64
      liveness      float64
      loudness      float64
      speechiness   float64
      tempo         float64
      valence       float64
      popularity    int64
      duration_ms   int64
      dtype: object
```

```
[58]: X = df1.drop(['name', 'release_date', 'id', 'uri'], axis=1)
```

```
[59]: X
```

```
[59]:
```

	album	track_number	acousticness	danceability	energy	\
0	Licked Live In NYC	1	0.0824	0.463	0.993	
1	Licked Live In NYC	2	0.4370	0.326	0.965	
2	Licked Live In NYC	3	0.4160	0.386	0.969	
3	Licked Live In NYC	4	0.5670	0.369	0.985	
4	Licked Live In NYC	5	0.4000	0.303	0.969	
...	
1605	The Rolling Stones	8	0.1570	0.466	0.932	
1606	The Rolling Stones	9	0.0576	0.509	0.706	
1607	The Rolling Stones	10	0.3710	0.790	0.774	
1608	The Rolling Stones	11	0.2170	0.700	0.546	
1609	The Rolling Stones	12	0.3830	0.727	0.934	

	instrumentalness	liveness	loudness	speechiness	tempo	valence	\
0	0.996000	0.9320	-12.913	0.1100	118.001	0.0302	
1	0.233000	0.9610	-4.803	0.0759	131.455	0.3180	
2	0.400000	0.9560	-4.936	0.1150	130.066	0.3130	
3	0.000107	0.8950	-5.535	0.1930	132.994	0.1470	
4	0.055900	0.9660	-5.098	0.0930	130.533	0.2060	
...	
1605	0.006170	0.3240	-9.214	0.0429	177.340	0.9670	
1606	0.000002	0.5160	-9.427	0.0843	122.015	0.4460	
1607	0.000000	0.0669	-7.961	0.0720	97.035	0.8350	
1608	0.000070	0.1660	-9.567	0.0622	102.634	0.5320	
1609	0.068500	0.0965	-8.373	0.0359	125.275	0.9690	

	popularity	duration_ms
0	33	48640
1	34	253173
2	34	263160
3	32	305880
4	32	305106
...
1605	39	154080
1606	36	245266
1607	30	176080
1608	27	121680
1609	35	189186

[1610 rows x 13 columns]

```
[60]: y = df1['popularity']
```

```
[62]: y
```

```
[62]: 0      33
      1      34
      2      34
      3      32
      4      32
      ..
     1605    39
     1606    36
     1607    30
     1608    27
     1609    35
      Name: popularity, Length: 1610, dtype: int64
```

```
[63]: from sklearn.preprocessing import LabelEncoder
      le = LabelEncoder()
```

```
[65]: X['album'] = le.fit_transform(X['album'])
```

```
[69]: X.head()
```

```
[69]:   album  track_number  acousticness  danceability  energy  instrumentalness  \
0     47             1         0.0824         0.463    0.993           0.996000
1     47             2         0.4370         0.326    0.965           0.233000
2     47             3         0.4160         0.386    0.969           0.400000
3     47             4         0.5670         0.369    0.985           0.000107
4     47             5         0.4000         0.303    0.969           0.055900

      liveness  loudness  speechiness   tempo  valence  popularity  duration_ms
0     0.932   -12.913         0.1100  118.001   0.0302           33         48640
1     0.961    -4.803         0.0759  131.455   0.3180           34        253173
2     0.956    -4.936         0.1150  130.066   0.3130           34        263160
3     0.895    -5.535         0.1930  132.994   0.1470           32        305880
4     0.966    -5.098         0.0930  130.533   0.2060           32        305106
```

```
[70]: from sklearn.preprocessing import MinMaxScaler
```

```
[71]: ms = MinMaxScaler()
```

```
[72]: cols = X.columns
```

```
[73]: X = ms.fit_transform(X)
```

```
[77]: X
```

```
[77]: array([[0.52808989, 0.02876572],
            [0.52808989, 0.02173913, 0.43963279, ..., 0.32648871, 0.425,
            0.24162891],
            [0.52808989, 0.04347826, 0.41850584, ..., 0.32135524, 0.425,
            0.25202265],
            ...,
            [0.85393258, 0.19565217, 0.3732338, ..., 0.85728953, 0.375,
            0.16139607],
            [0.85393258, 0.2173913, 0.21830283, ..., 0.54620123, 0.3375,
            0.10478048],
            [0.85393258, 0.23913043, 0.38530634, ..., 0.99486653, 0.4375,
            0.17503585]])
```

```
[78]: X = pd.DataFrame(X,columns=cols)
```

```
[81]: X
```

```
[81]:
```

	album	track_number	acousticness	danceability	energy	\
0	0.528090	0.000000	0.082889	0.458493	0.993007	
1	0.528090	0.021739	0.439633	0.283525	0.960373	
2	0.528090	0.043478	0.418506	0.360153	0.965035	
3	0.528090	0.065217	0.570419	0.338442	0.983683	
4	0.528090	0.086957	0.402409	0.254151	0.965035	
...	
1605	0.853933	0.152174	0.157940	0.462324	0.921911	
1606	0.853933	0.173913	0.057939	0.517241	0.658508	
1607	0.853933	0.195652	0.373234	0.876117	0.737762	
1608	0.853933	0.217391	0.218303	0.761175	0.472028	
1609	0.853933	0.239130	0.385306	0.795658	0.924242	

	instrumentalness	liveness	loudness	speechiness	tempo	valence	\
0	1.000000	0.932384	0.491365	0.144474	0.420994	0.031006	
1	0.233936	0.962094	0.838035	0.087716	0.500239	0.326489	
2	0.401606	0.956972	0.832350	0.152796	0.492057	0.321355	
3	0.000107	0.894478	0.806745	0.282623	0.509303	0.150924	
4	0.056124	0.967216	0.825425	0.116178	0.494808	0.211499	
...	
1605	0.006195	0.309497	0.649483	0.032790	0.770502	0.992813	
1606	0.000002	0.506198	0.640378	0.101698	0.444637	0.457906	
1607	0.000000	0.046102	0.703044	0.081225	0.297504	0.857290	
1608	0.000070	0.147628	0.634393	0.064913	0.330483	0.546201	
1609	0.068775	0.076427	0.685432	0.021138	0.463838	0.994867	

	popularity	duration_ms
0	0.4125	0.028766
1	0.4250	0.241629

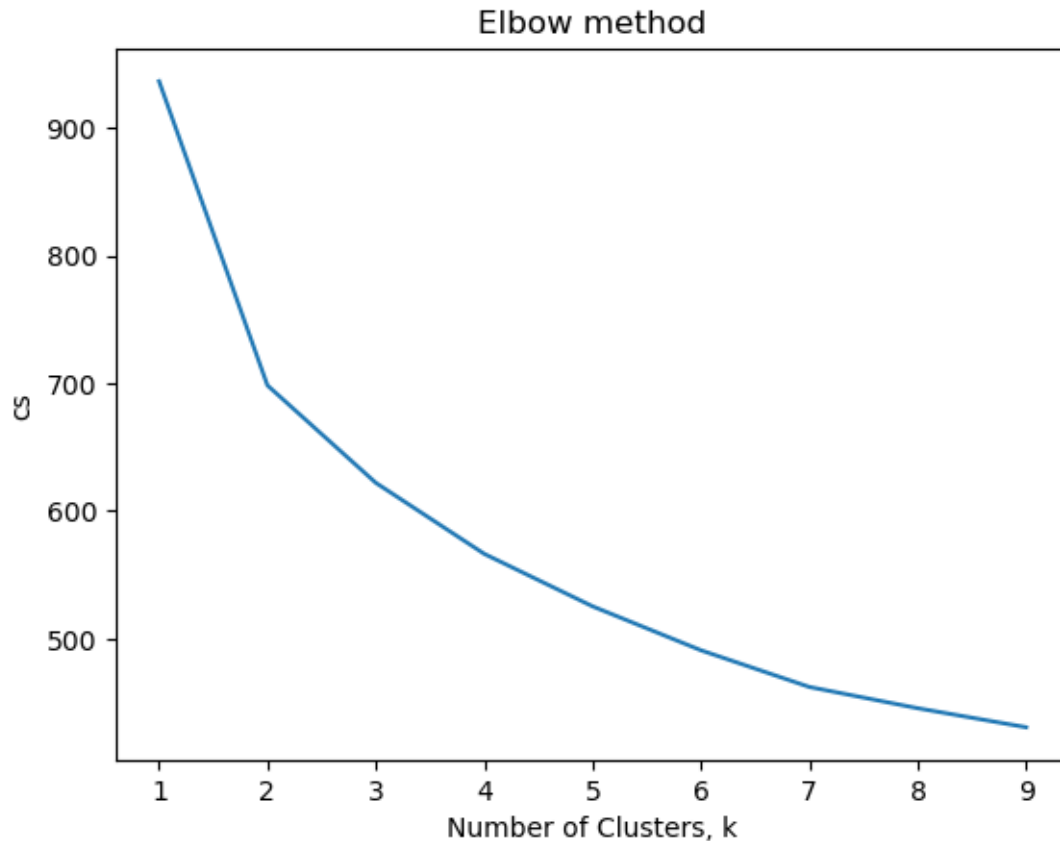
2	0.4250	0.252023
3	0.4000	0.296483
4	0.4000	0.295677
...
1605	0.4875	0.138500
1606	0.4500	0.233400
1607	0.3750	0.161396
1608	0.3375	0.104780
1609	0.4375	0.175036

[1610 rows x 13 columns]

```
[82]: from sklearn.cluster import KMeans
```

```
[83]: cs = []
      for i in range(1,10):
          kmeans = KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
          kmeans.fit(X)
          cs.append(kmeans.inertia_)
```

```
[84]: plt.plot(range(1,10),cs)
      plt.title('Elbow method')
      plt.xlabel('Number of Clusters, k')
      plt.ylabel('cs')
      plt.show()
```



```
[85]: kmeans = KMeans(n_clusters=2,random_state=0)
```

```
[88]: kmeans.fit(X)
```

C:\Users\soura\anaconda3\Lib\site-packages\sklearn\cluster_kmeans.py:870:
 FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
 1.4. Set the value of `n_init` explicitly to suppress the warning
 warnings.warn(

```
[88]: KMeans(n_clusters=2, random_state=0)
```

```
[89]: labels = kmeans.labels_
```

```
[90]: correct_labels = sum(y==labels)
```

```
[91]: print('Results {} out of {} samples were correctly labels'.  

      ↪format(correct_labels,y.size))
```

Results 33 out of 1610 samples were correctly labels

```
[92]: print('Accuracy Score :{0:0.2f}'.format(correct_labels/float(y.size)))
```

Accuracy Score :0.02

```
[ ]:
```