In [23]:
```
#Hlanhla Hlungwane
#Python Project 3
#Cohorts of songs

import numpy as np    #for numerical computing
import pandas as pd   #for analysis of dataframes
import seaborn as sns #for visualizations
```

In [24]:
```
#Import the data

df = pd.read_csv('rolling_stones_spotify.csv')     #pandas library to read f
data = pd.read_excel('1722506509_datadictionarycreatingcohortsofsongs.xlsx')
```

In [25]:
```
df.head()
```

Out[25]:

	Unnamed: 0	name	album	release_date	track_number	id	
0	0	Concert Intro Music - Live	Licked Live In NYC	2022-06-10	1	2IEkywLJ4ykbhi1yRQvmsT	s
1	1	Street Fighting Man - Live	Licked Live In NYC	2022-06-10	2	6GVgVJBKkGJoRfarYRvGTU	spo
2	2	Start Me Up - Live	Licked Live In NYC	2022-06-10	3	1Lu761pZ0dBTGpzxaQoZNW	spo
3	3	If You Can't Rock Me - Live	Licked Live In NYC	2022-06-10	4	1agTQzOTUnGNggyckEqiDH	spo
4	4	Don’t Stop - Live	Licked Live In NYC	2022-06-10	5	7piGJR8YndQBQWVXv6KtQw	spot

In [26]:
```
df.isnull().sum()
#There are no missing values
```

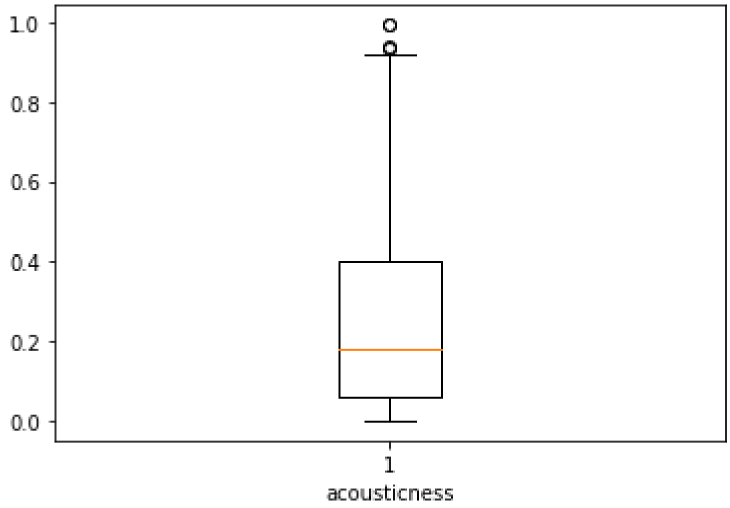
Out[26]:
```
Unnamed: 0          0
name                0
album               0
release_date        0
track_number        0
id                  0
uri                 0
acousticness        0
danceability        0
energy              0
instrumentalness    0
liveness            0
loudness            0
speechiness         0
tempo               0
valence             0
popularity          0
duration_ms         0
dtype: int64
```

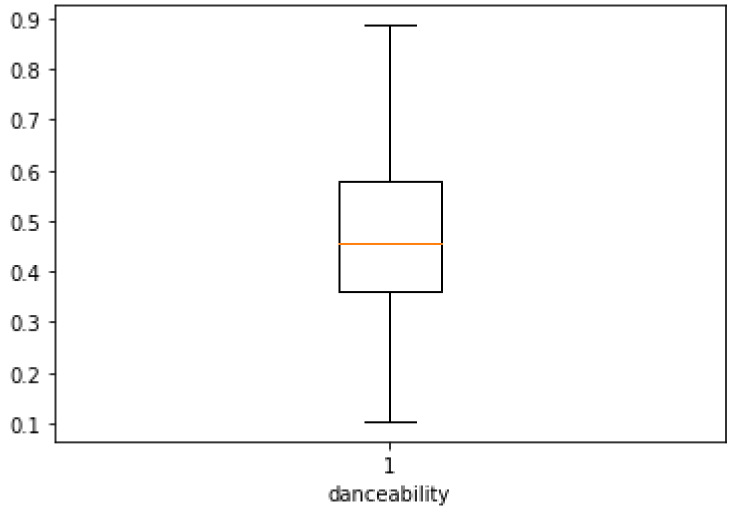
In [27]:
```
only_duplicates = df[df.duplicated()]
print(only_duplicates)
```

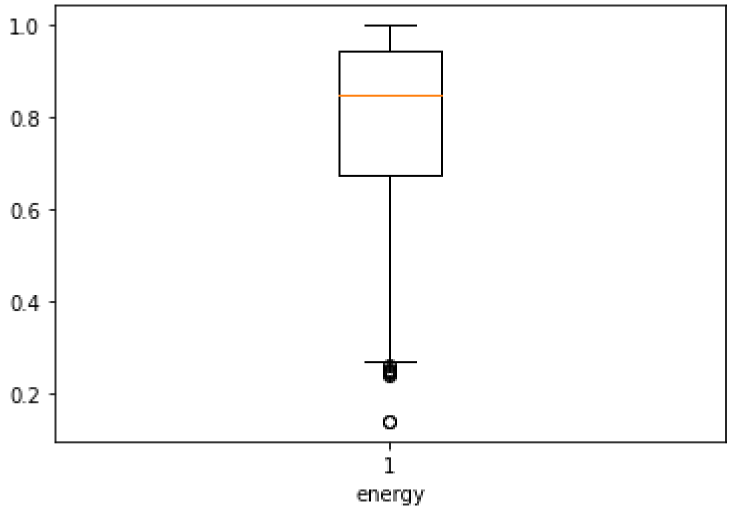
```
Empty DataFrame
Columns: [Unnamed: 0, name, album, release_date, track_number, id, uri, aco
usticness, danceability, energy, instrumentalness, liveness, loudness, spee
chiness, tempo, valence, popularity, duration_ms]
Index: []
```

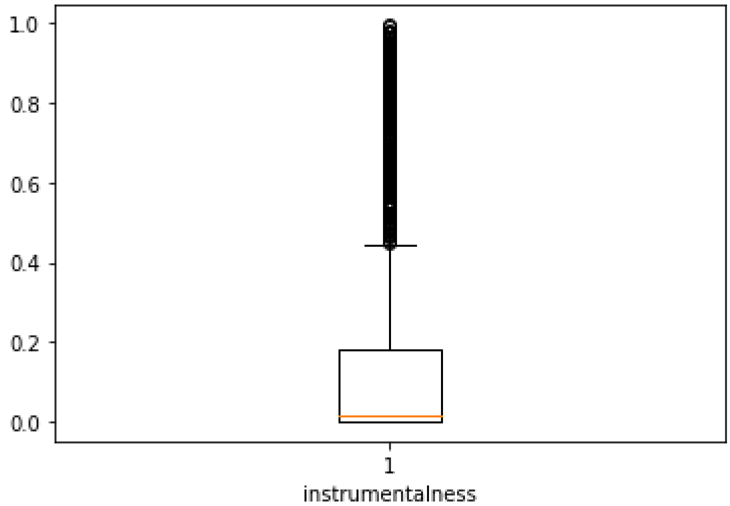
In [28]:
```
from sklearn.preprocessing import MinMaxScaler  #Normalizes numerical data t
import matplotlib.pyplot as plt       #Library for visualizations


Scaler = MinMaxScaler()    #craeting an instance
columns_to_normalize = df[["loudness","tempo","popularity","duration_ms"]]
normalized_df = Scaler.fit_transform(columns_to_normalize)     #scaling the
new_df = pd.DataFrame(normalized_df)
new_df.columns = ["norm_loudness","norm_tempo","norm_popularity","norm_durat
print(new_df)
merged_df = pd.concat([df, new_df], axis =1)  #combines the new and old data
merged_df
String_columns = df[['Unnamed: 0','name','album', 'release_date','track_numb
for i in merged_df:
    if i not in String_columns:
        plt.boxplot(merged_df[i])
        plt.xlabel(i)
        plt.show()
```

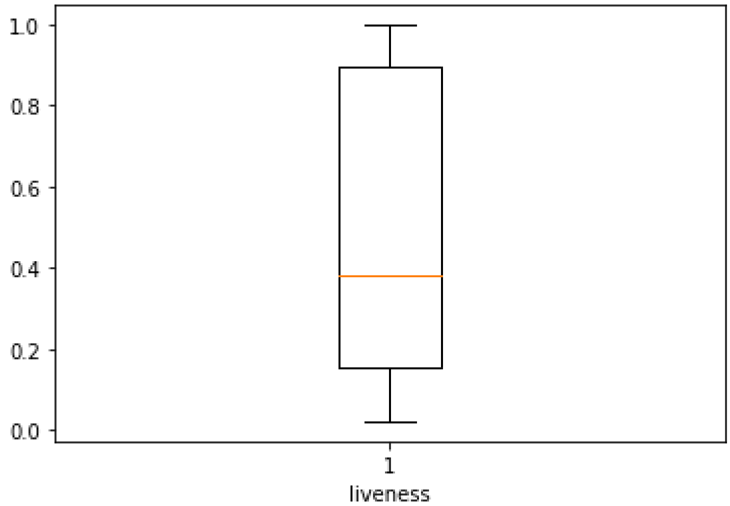
```
      norm_loudness  norm_tempo  norm_popularity  norm_duration_ms
0          0.491365    0.420994           0.4125          0.028766
1          0.838035    0.500239           0.4250          0.241629
2          0.832350    0.492057           0.4250          0.252023
3          0.806745    0.509303           0.4000          0.296483
4          0.825425    0.494808           0.4000          0.295677
...             ...         ...              ...               ...
1605       0.649483    0.770502           0.4875          0.138500
1606       0.640378    0.444637           0.4500          0.233400
1607       0.703044    0.297504           0.3750          0.161396
1608       0.634393    0.330483           0.3375          0.104780
1609       0.685432    0.463838           0.4375          0.175036

[1610 rows x 4 columns]
```


1.0
0.8
0.6
0.4
0.2
0.0
1
acousticness

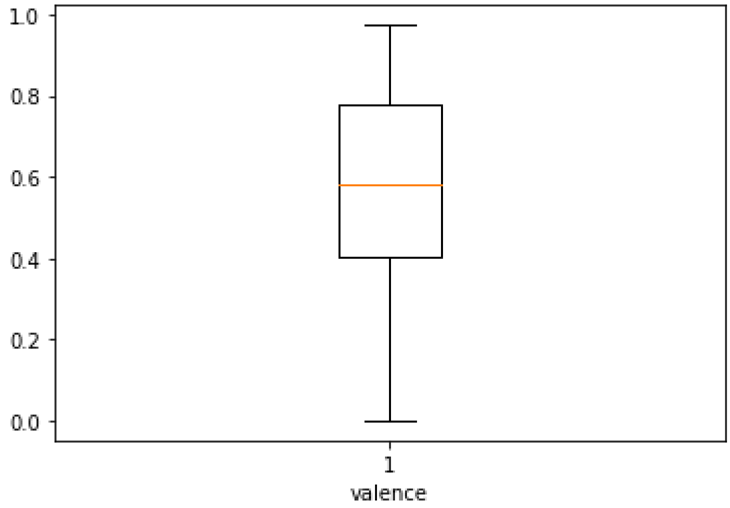

0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1
danceability

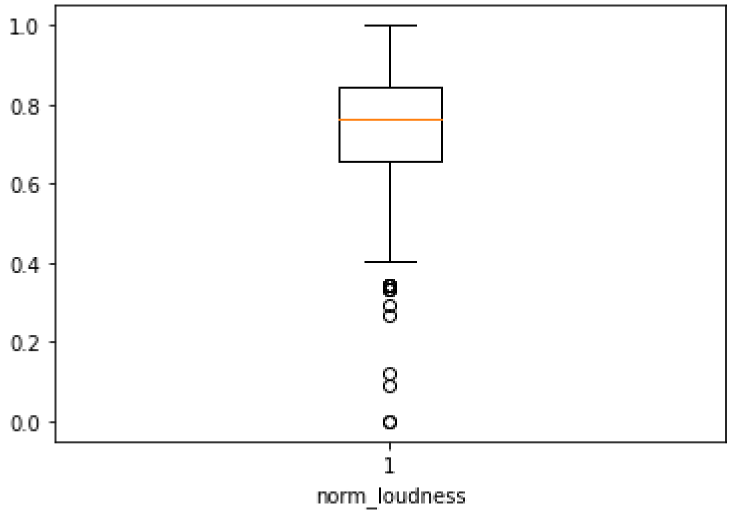

1.0
0.8
0.6
0.4
0.2
1
energy

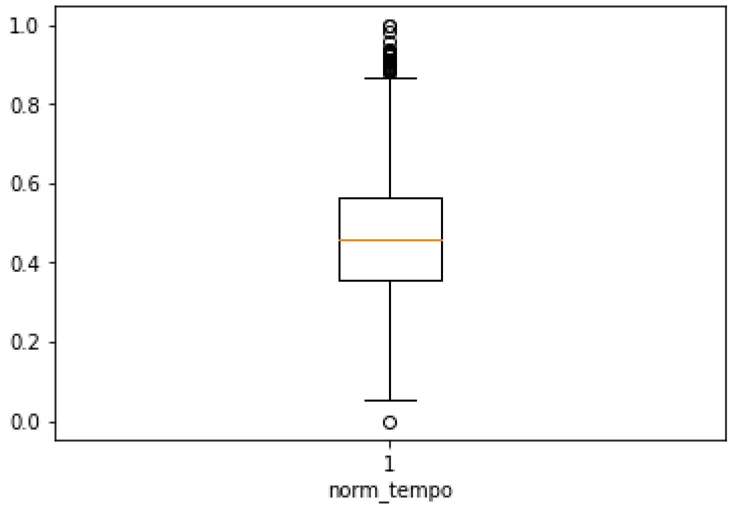

1.0
0.8
0.6
0.4
0.2
0.0
1
instrumentalness

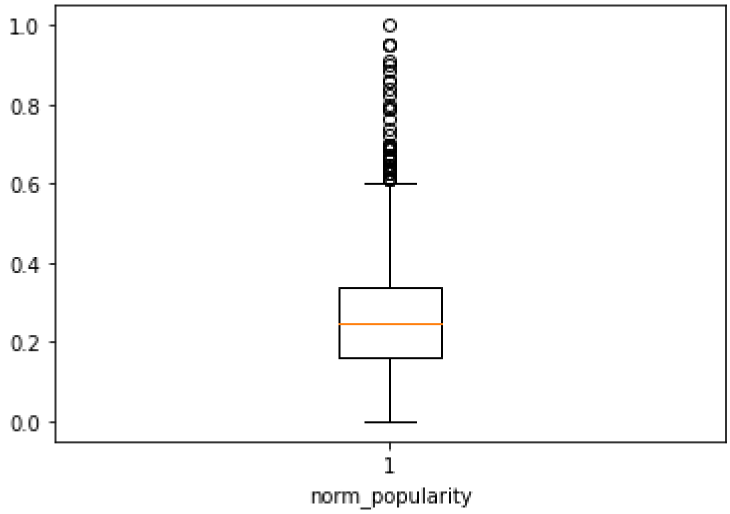

1.0
0.8
0.6
0.4
0.2
0.0
1
liveness

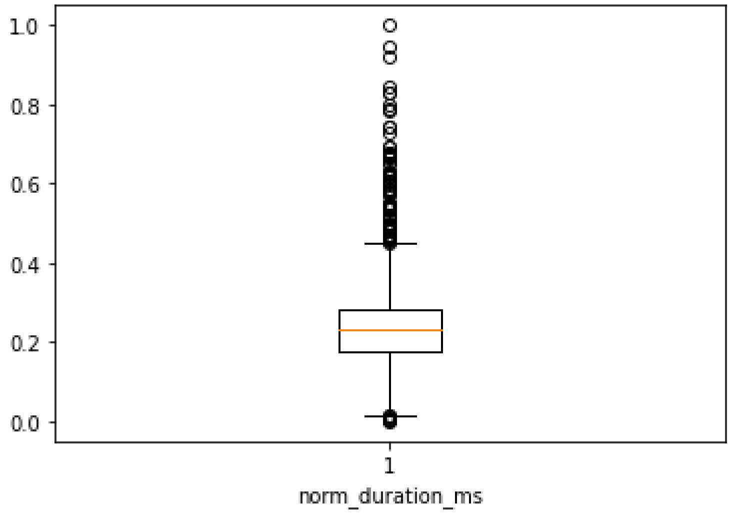

0.6
0.5
0.4
0.3
0.2
0.1
0.0
1
speechiness


1.0
0.8
0.6
0.4
0.2
0.0
1
valence

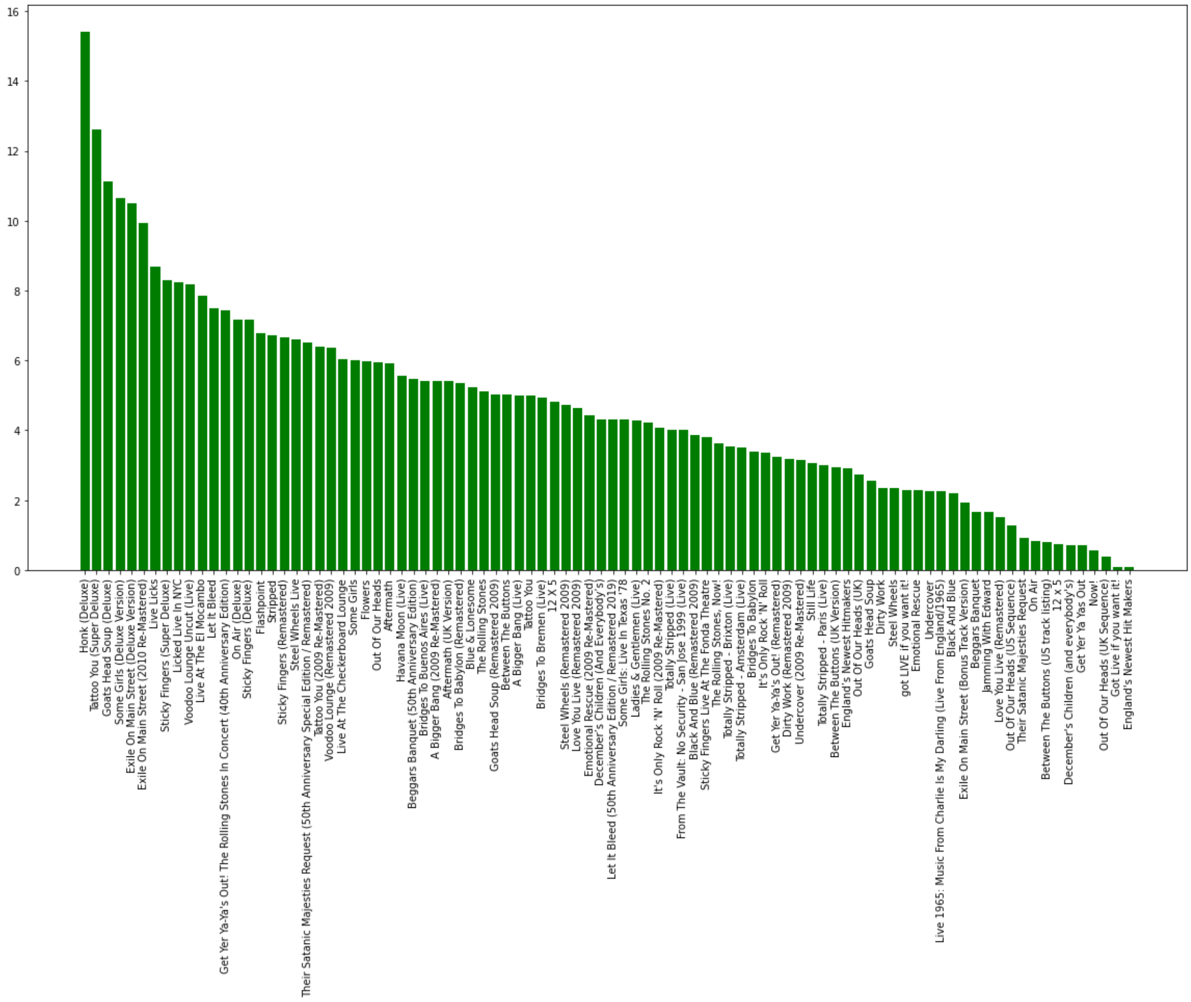

1.0
0.8
0.6
0.4
0.2
0.0
1
norm_loudness


1.0
0.8
0.6
0.4
0.2
0.0
1
norm_tempo





In [29]:
```
#Exploratory data analysis
#Group data by album and aggregate popularity
album_data = merged_df.groupby('album')['norm_popularity'].agg(total_popular
sorted_plot = album_data.sort_values(by='total_popularity', ascending  =Fals
plt.figure(figsize=(20,10))
plt.bar(x=sorted_plot['album'],height=sorted_plot['total_popularity'], color
plt.xticks(rotation = 90)
plt.show()
```

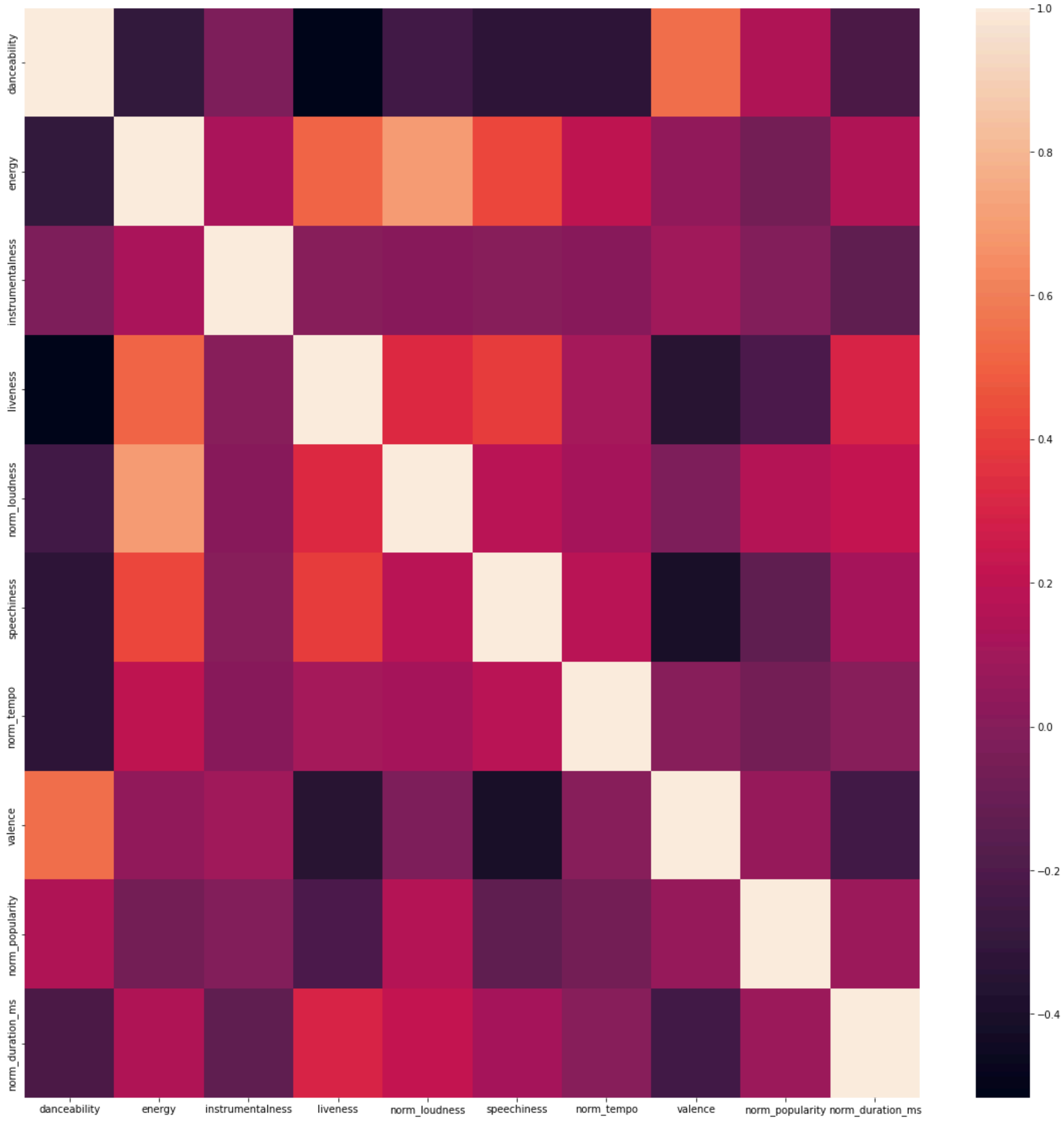


```
In [30]: corr_matrix_df = merged_df[["danceability", "energy", "instrumentalness", "l
         corr_matrix = corr_matrix_df.corr()
         print(corr_matrix)
```

```
                  danceability    energy  instrumentalness  liveness  \
danceability          1.000000 -0.300536         -0.031812 -0.516387   
energy               -0.300536  1.000000          0.120261  0.511188   
instrumentalness     -0.031812  0.120261          1.000000  0.008873   
liveness             -0.516387  0.511188          0.008873  1.000000   
norm_loudness        -0.249406  0.698039          0.012524  0.327036   
speechiness          -0.322684  0.417214          0.009586  0.400018   
norm_tempo           -0.324398  0.201885          0.010961  0.108855   
valence               0.546210  0.046217          0.103480 -0.347451   
norm_popularity       0.141205 -0.057272         -0.010612 -0.205845   
norm_duration_ms     -0.220045  0.148876         -0.137599  0.304735   

                  norm_loudness  speechiness  norm_tempo   valence  \
danceability          -0.249406    -0.322684   -0.324398  0.546210   
energy                 0.698039     0.417214    0.201885  0.046217   
instrumentalness       0.012524     0.009586    0.010961  0.103480   
liveness               0.327036     0.400018    0.108855 -0.347451   
norm_loudness          1.000000     0.189904    0.112837 -0.027571   
speechiness            0.189904     1.000000    0.192687 -0.399751   
norm_tempo             0.112837     0.192687    1.000000  0.000558   
valence               -0.027571    -0.399751    0.000558  1.000000   
norm_popularity        0.156323    -0.136745   -0.061061  0.065333   
norm_duration_ms       0.221558     0.114546    0.001465 -0.244833   

                  norm_popularity  norm_duration_ms  
danceability             0.141205         -0.220045  
energy                  -0.057272          0.148876  
instrumentalness        -0.010612         -0.137599  
liveness                -0.205845          0.304735  
norm_loudness            0.156323          0.221558  
speechiness             -0.136745          0.114546  
norm_tempo              -0.061061          0.001465  
valence                  0.065333         -0.244833  
norm_popularity          1.000000          0.074102  
norm_duration_ms         0.074102          1.000000  
```

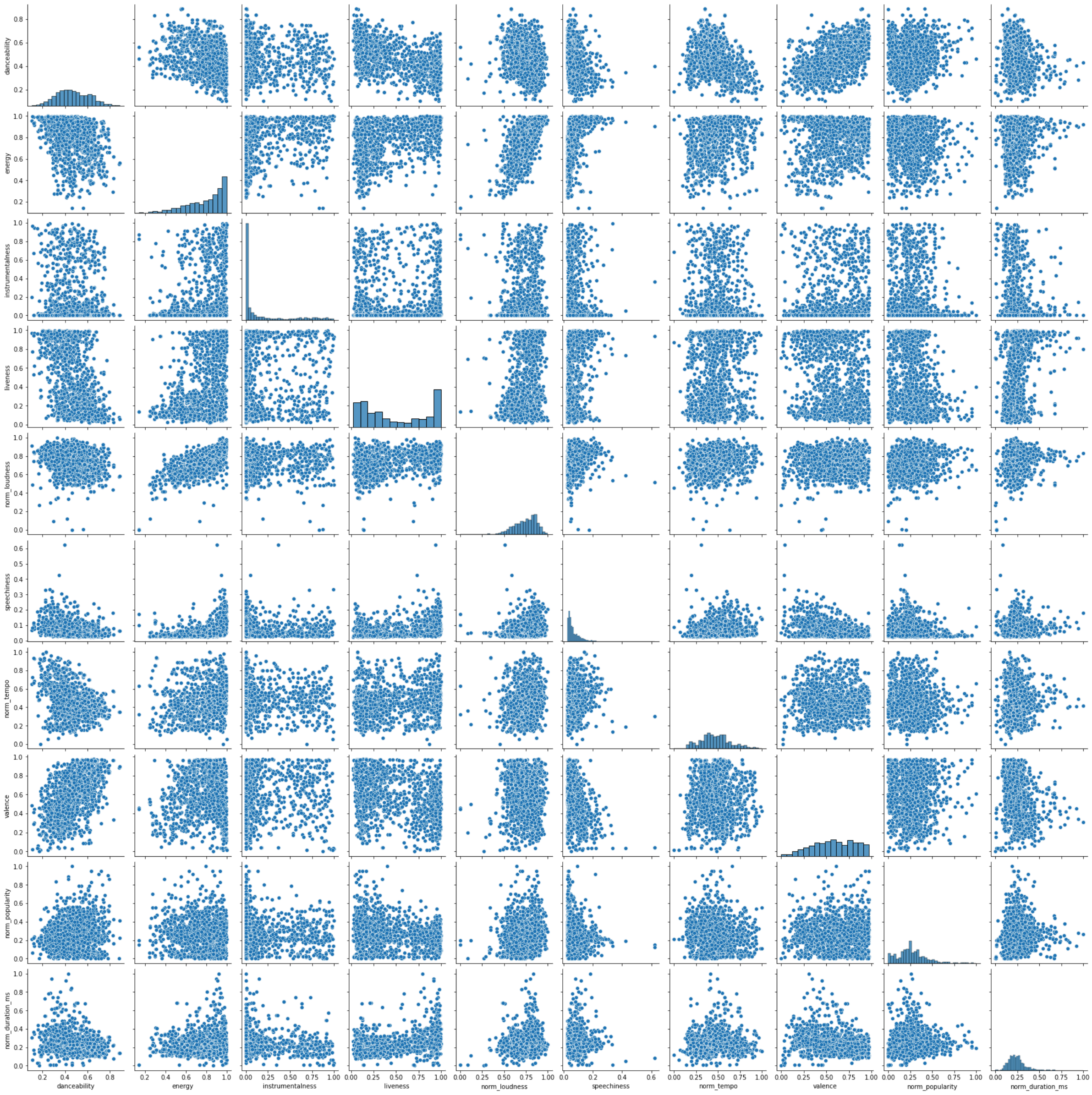
In [31]:

```
import seaborn as sns
plt.figure(figsize=(20,20))
sns.heatmap(corr_matrix)
plt.show()
```



In [32]:
```
plt.figure(figsize=(20,20))
sns.pairplot(corr_matrix_df)
plt.show()
```

<Figure size 1440x1440 with 0 Axes>



In [33]:
```
# only danceability, energy, duration, tempo, loudness following normal dist

#1. danceability vs valence

import scipy.stats as stats
Correlation_coff, p_value = stats.spearmanr(merged_df["danceability"], merge
print(f"danceability vs valence,{Correlation_coff},{p_value}")
if 0 <= Correlation_coff <= 0.3 and p_value > 0.05:
    print("week correlation")
elif 0.3 < Correlation_coff < 0.5 and p_value < 0.05:
    print("moderate correlation")
else:
    print("Strong correlation")

#2. Energy Vs liveness
Correlation_coff, p_value = stats.spearmanr(merged_df["energy"], merged_df['
print(f"Energy Vs liveness,{Correlation_coff},{p_value}")
if 0 <= Correlation_coff <= 0.3 and p_value > 0.05:
    print("week correlation")
elif 0.3 < Correlation_coff < 0.5 and p_value < 0.05:
    print("moderate correlation")
else:
    print("Strong correlation")

#3.speechness vs enerygy
```

```
Correlation_coff, p_value = stats.spearmanr(merged_df["speechiness"], mergec
print(f"speechness vs energy,{Correlation_coff},{p_value}")
if 0 <= Correlation_coff <= 0.3 and p_value > 0.05:
    print("week correlation")
elif 0.3 < Correlation_coff < 0.5 and p_value < 0.05:
    print("moderate correlation")
else:
    print("Strong correlation")

#4. speechness vs liveness
Correlation_coff, p_value = stats.spearmanr(merged_df["speechiness"], mergec
print(f"speechness vs liveness,{Correlation_coff},{p_value}")
if 0 <= Correlation_coff <= 0.3 and p_value > 0.05:
    print("week correlation")
elif 0.3 < Correlation_coff < 0.5 and p_value < 0.05:
    print("moderate correlation")
else:
    print("Strong correlation")

#5. valence vs danceability
Correlation_coff, p_value = stats.spearmanr(merged_df["valence"], merged_df[
print(f"valence vs danceability,{Correlation_coff},{p_value}")
if 0 <= Correlation_coff <= 0.3 and p_value > 0.05:
    print("week correlation")
elif 0.3 < Correlation_coff < 0.5 and p_value < 0.05:
    print("moderate correlation")
else:
    print("Strong correlation")

#6. loudness vs energy
#both are showing normal distribution, hence using pearsonr
Correlation_coff, p_value = stats.pearsonr(merged_df["loudness"], merged_df[
print(f"loudness vs energy,{Correlation_coff},{p_value}")
if 0 <= Correlation_coff <= 0.3 and p_value > 0.05:
    print("week correlation")
elif 0.3 < Correlation_coff < 0.5 and p_value < 0.05:
    print("moderate correlation")
else:
    print("Strong correlation")
```

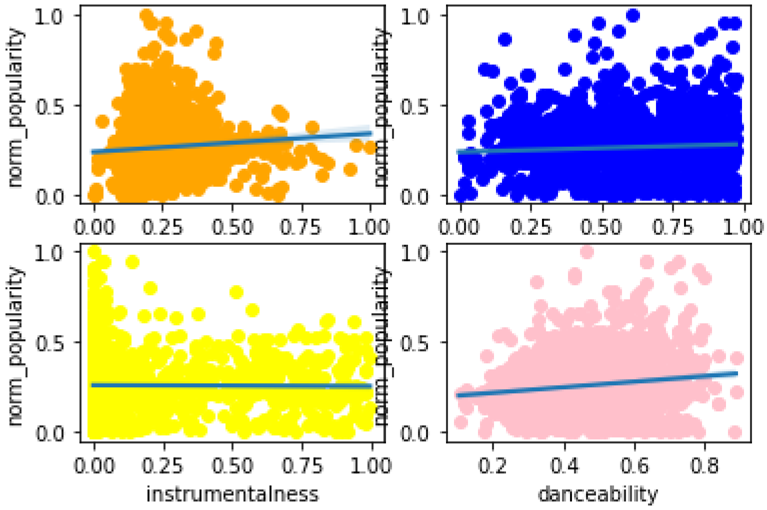
```
danceability vs valence,0.556965608318474,7.591672888090832e-132
Strong correlation
Energy Vs liveness,0.5300624101268263,2.512910353358011e-117
Strong correlation
speechness vs energy,0.6291263687108982,3.705847143101835e-178
Strong correlation
speechness vs liveness,0.4654680415791756,2.321090415649216e-87
moderate correlation
valence vs danceability,0.556965608318474,7.591672888090832e-132
Strong correlation
loudness vs energy,0.6980390870508658,1.6326393047455832e-235
Strong correlation
```

In [34]:

```
plt.subplot(2,2,1)
plt.scatter(merged_df["norm_duration_ms"], merged_df["norm_popularity"],colc
sns.regplot(x = merged_df["norm_duration_ms"], y = merged_df["norm_popularit
plt.show

plt.subplot(2,2,2)
plt.scatter(merged_df["valence"], merged_df["norm_popularity"],color='Blue')
sns.regplot(x = merged_df["valence"], y = merged_df["norm_popularity"], scat
plt.show
```

```
plt.subplot(2,2,3)
plt.scatter(merged_df["instrumentalness"], merged_df["norm_popularity"], col
sns.regplot(x = merged_df["instrumentalness"], y = merged_df["norm_popularit
plt.show

plt.subplot(2,2,4)
plt.scatter(merged_df["danceability"], merged_df["norm_popularity"], color =
sns.regplot(x = merged_df["danceability"], y = merged_df["norm_popularity"],
plt.show
```

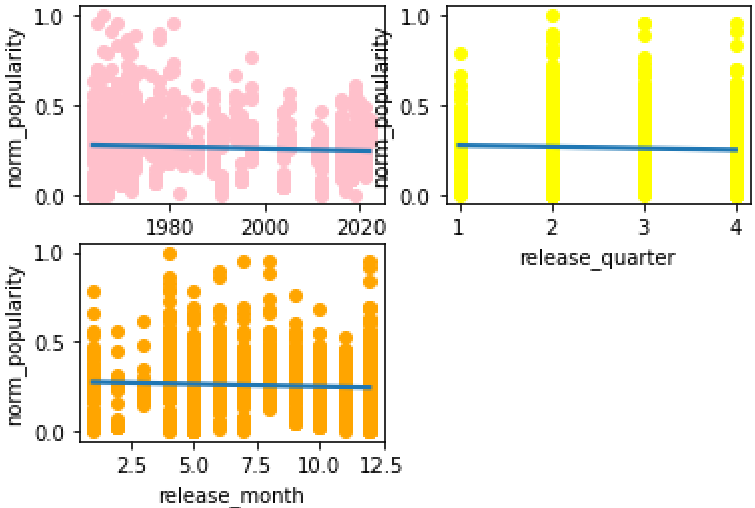
Out[34]: <function matplotlib.pyplot.show(close=None, block=None)>



In [35]:
```
merged_df['release_date'] = pd.to_datetime(merged_df['release_date'])
merged_df['release_year'] = merged_df['release_date'].dt.year
date_df = merged_df.groupby('release_year')
print(date_df.head(5))
plt.subplot(2,2,1)
plt.scatter(merged_df["release_year"], merged_df["norm_popularity"], color =
sns.regplot(x = merged_df["release_year"], y = merged_df["norm_popularity"],
plt.show
coefficent, pvalue = stats.spearmanr(merged_df["release_year"], merged_df["r
print(f'release_year vs norm_popularity,{coefficent},{pvalue}')

merged_df['release_quarter'] = merged_df['release_date'].dt.quarter
date_df = merged_df.groupby('release_quarter')
plt.subplot(2,2,2)
plt.scatter(merged_df["release_quarter"], merged_df["norm_popularity"], colo
sns.regplot(x = merged_df["release_quarter"], y = merged_df["norm_popularity
plt.show
coefficent, pvalue = stats.spearmanr(merged_df["release_quarter"], merged_df
print(f'release_quarter vs norm_popularity,{coefficent},{pvalue}')

merged_df['release_month'] = merged_df['release_date'].dt.month
date_df = merged_df.groupby('release_month')
plt.subplot(2,2,3)
plt.scatter(merged_df["release_month"], merged_df["norm_popularity"], color
sns.regplot(x = merged_df["release_month"], y = merged_df["norm_popularity"]
plt.show
coefficent, pvalue = stats.spearmanr(merged_df["release_month"], merged_df['
print(f'release_month vs norm_popularity,{coefficent},{pvalue}')
```

```
      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   
...          ...                                   ...                 ...   
1550        1550                     Around And Around              12 x 5   
1551        1551                  Confessin' The Blues              12 x 5   
1552        1552                           Empty Heart              12 x 5   
1553        1553                     Time Is On My Side             12 x 5   
1554        1554  Good Times, Bad Times - Mono Version              12 x 5   

     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  1agTQzOTUnGNggyckEqiDH   
4      2022-06-10             5  7piGJR8YndQBQWVXv6KtQw   
...           ...           ...                     ...   
1550   1964-10-17             1  7DyLeriNYHyE57tV161YVx   
1551   1964-10-17             2  1YVgDulqwWJO8YhOQ0PRE8   
1552   1964-10-17             3  0XWEpqks9LFOQr96Nd9Ojl   
1553   1964-10-17             4  6aiALZZDFibaxE067I9Jf8   
1554   1964-10-17             5  2F3iN8io7kkrrIMXW7nhvd   

                                       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:1agTQzOTUnGNggyckEqiDH        0.5670         0.369   
4     spotify:track:7piGJR8YndQBQWVXv6KtQw        0.4000         0.303   
...                                    ...           ...           ...   
1550  spotify:track:7DyLeriNYHyE57tV161YVx        0.4770         0.317   
1551  spotify:track:1YVgDulqwWJO8YhOQ0PRE8        0.1670         0.479   
1552  spotify:track:0XWEpqks9LFOQr96Nd9Ojl        0.0069         0.498   
1553  spotify:track:6aiALZZDFibaxE067I9Jf8        0.2880         0.232   
1554  spotify:track:2F3iN8io7kkrrIMXW7nhvd        0.0546         0.438   

      energy  ...  speechiness    tempo  valence  popularity  duration_ms  
\
0      0.993  ...       0.1100  118.001   0.0302          33        48640   
1      0.965  ...       0.0759  131.455   0.3180          34       253173   
2      0.969  ...       0.1150  130.066   0.3130          34       263160   
3      0.985  ...       0.1930  132.994   0.1470          32       305880   
4      0.969  ...       0.0930  130.533   0.2060          32       305106   
...      ...  ...          ...      ...      ...         ...          ...   
1550   0.608  ...       0.0584  183.461   0.7820           7       183200   
1551   0.523  ...       0.0416  122.547   0.7900           5       167266   
1552   0.678  ...       0.0386  140.585   0.9610           2       156973   
1553   0.838  ...       0.1440  215.810   0.4240           9       172733   
1554   0.534  ...       0.0462  196.971   0.6430           2       150200   

      norm_loudness  norm_tempo  norm_popularity  norm_duration_ms  \
0          0.491365    0.420994           0.4125          0.028766   
1          0.838035    0.500239           0.4250          0.241629   
2          0.832350    0.492057           0.4250          0.252023   
3          0.806745    0.509303           0.4000          0.296483   
4          0.825425    0.494808           0.4000          0.295677   
...             ...         ...              ...               ...   
```

```
1550       0.536933    0.806554          0.0875           0.168806
1551       0.583355    0.447770          0.0625           0.152223
1552       0.599855    0.554014          0.0250           0.141511
1553       0.720398    0.997090          0.1125           0.157913
1554       0.628110    0.886128          0.0250           0.134462

      release_year
0             2022
1             2022
2             2022
3             2022
4             2022
...            ...
1550          1964
1551          1964
1552          1964
1553          1964
1554          1964

[175 rows x 23 columns]
release_year vs norm_popularity,0.01580625970558492,0.5262308704512082
release_quarter vs norm_popularity,-0.07166919185222259,0.00401247590091248
7
release_month vs norm_popularity,-0.08536566576799774,0.0006060755141697807
```



In [36]:
```
final_df = merged_df.drop(['id','uri','Unnamed: 0','name','loudness','tempo'
final_df.head(2)
```

Out[36]:

	album	track_number	acousticness	danceability	energy	instrumentalness	liveness	sp
0	Licked Live In NYC	1	0.0824	0.463	0.993	0.996	0.932	
1	Licked Live In NYC	2	0.4370	0.326	0.965	0.233	0.961	

In [37]:
```
#Perform cluster analysis

#converting album to numeric values using label_encoder
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
final_df['album'] = label.fit_transform(final_df['album'])
print(final_df)
```

```
      album  track_number  acousticness  danceability  energy  \
0        47             1        0.0824         0.463   0.993
1        47             2        0.4370         0.326   0.965
2        47             3        0.4160         0.386   0.969
3        47             4        0.5670         0.369   0.985
4        47             5        0.4000         0.303   0.969
...     ...           ...           ...           ...     ...
1605     76             8        0.1570         0.466   0.932
1606     76             9        0.0576         0.509   0.706
1607     76            10        0.3710         0.790   0.774
1608     76            11        0.2170         0.700   0.546
1609     76            12        0.3830         0.727   0.934

      instrumentalness  liveness  speechiness  valence  norm_loudness  \
0             0.996000    0.9320       0.1100   0.0302       0.491365
1             0.233000    0.9610       0.0759   0.3180       0.838035
2             0.400000    0.9560       0.1150   0.3130       0.832350
3             0.000107    0.8950       0.1930   0.1470       0.806745
4             0.055900    0.9660       0.0930   0.2060       0.825425
...                ...       ...          ...      ...            ...
1605          0.006170    0.3240       0.0429   0.9670       0.649483
1606          0.000002    0.5160       0.0843   0.4460       0.640378
1607          0.000000    0.0669       0.0720   0.8350       0.703044
1608          0.000070    0.1660       0.0622   0.5320       0.634393
1609          0.068500    0.0965       0.0359   0.9690       0.685432

      norm_tempo  norm_popularity  norm_duration_ms
0       0.420994           0.4125          0.028766
1       0.500239           0.4250          0.241629
2       0.492057           0.4250          0.252023
3       0.509303           0.4000          0.296483
4       0.494808           0.4000          0.295677
...          ...              ...               ...
1605    0.770502           0.4875          0.138500
1606    0.444637           0.4500          0.233400
1607    0.297504           0.3750          0.161396
1608    0.330483           0.3375          0.104780
1609    0.463838           0.4375          0.175036

[1610 rows x 13 columns]
```

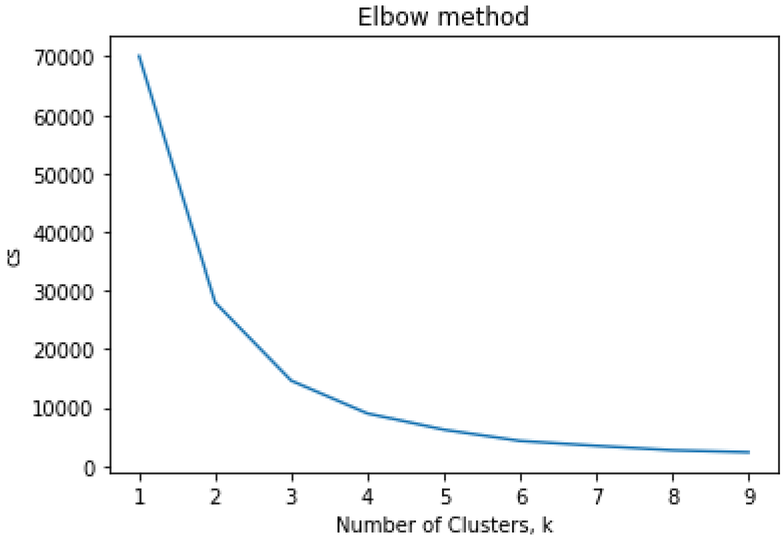
In [38]:

```
#normalizing the value of album since it is not in the range of 0 to 1
Scaler = MinMaxScaler()
normalize_column = final_df[['album']]
album_value = Scaler.fit_transform(normalize_column)
new_album_df= pd.DataFrame(album_value,columns=['album_normalized'])
final_df = final_df.drop(columns=['album'])
new_final_df=pd.concat([final_df,new_album_df],axis=1)
print(new_final_df)
```

```
      track_number  acousticness  danceability  energy  instrumentalness  \
0                1        0.0824         0.463   0.993          0.996000   
1                2        0.4370         0.326   0.965          0.233000   
2                3        0.4160         0.386   0.969          0.400000   
3                4        0.5670         0.369   0.985          0.000107   
4                5        0.4000         0.303   0.969          0.055900   
...            ...           ...           ...     ...               ...   
1605             8        0.1570         0.466   0.932          0.006170   
1606             9        0.0576         0.509   0.706          0.000002   
1607            10        0.3710         0.790   0.774          0.000000   
1608            11        0.2170         0.700   0.546          0.000070   
1609            12        0.3830         0.727   0.934          0.068500   

      liveness  speechiness  valence  norm_loudness  norm_tempo  \
0       0.9320       0.1100   0.0302       0.491365    0.420994   
1       0.9610       0.0759   0.3180       0.838035    0.500239   
2       0.9560       0.1150   0.3130       0.832350    0.492057   
3       0.8950       0.1930   0.1470       0.806745    0.509303   
4       0.9660       0.0930   0.2060       0.825425    0.494808   
...        ...          ...      ...            ...         ...   
1605    0.3240       0.0429   0.9670       0.649483    0.770502   
1606    0.5160       0.0843   0.4460       0.640378    0.444637   
1607    0.0669       0.0720   0.8350       0.703044    0.297504   
1608    0.1660       0.0622   0.5320       0.634393    0.330483   
1609    0.0965       0.0359   0.9690       0.685432    0.463838   

      norm_popularity  norm_duration_ms  album_normalized  
0              0.4125          0.028766          0.528090  
1              0.4250          0.241629          0.528090  
2              0.4250          0.252023          0.528090  
3              0.4000          0.296483          0.528090  
4              0.4000          0.295677          0.528090  
...               ...               ...               ...  
1605           0.4875          0.138500          0.853933  
1606           0.4500          0.233400          0.853933  
1607           0.3750          0.161396          0.853933  
1608           0.3375          0.104780          0.853933  
1609           0.4375          0.175036          0.853933  

[1610 rows x 13 columns]
```

In [39]:

```python
# #Identify the right number of clusters
# Elbow Method to identify the right number of clusters
from sklearn.cluster import KMeans
cs = []
for i in range(1,10):
    kmeans = KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,rar
    kmeans.fit(new_final_df)
    cs.append(kmeans.inertia_)

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



In [40]:
```
# #Use appropriate clustering algorithms
# Kmeans

kmeans = KMeans(n_clusters=2,random_state=0)
kmeans.fit(new_final_df)

#Quality Check by silhoutte score
```

```
/usr/local/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:1416: 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
  super()._check_params_vs_input(X, default_n_init=10)
```

Out[40]:
```
▾ KMeans
KMeans(n_clusters=2, random_state=0)
```

In [41]:
```
# #Define each cluster based on the features
clusters = kmeans.labels_
print(clusters)
```

```
[0 0 0 ... 0 0 0]
```

In [42]:
```
from sklearn.metrics import silhouette_score
silhouette_score(new_final_df, clusters)
```

Out[42]: 0.6224857443059663

In [ ]: