MACHINE LEARNING PROJECT-3

Creating Cohorts of Song.

Submitted by:
Raja Lairenmayum

Problem Scenario

The customer always looks forward to specialized treatment, whether shopping over an e-commerce website or watching Netflix. They want what they might like to see. To keep the customers engaged, it is also crucial for companies to always present the most relevant information. Spotify is a Swedish audio streaming and media service provider. The company has over 456 million active monthly users, including over 195 million paying subscribers, as of September 2022. The company intends to create cohorts of different songs that will aid in the recommendation of songs to users based on various relevant features. Each cohort would contain similar types of songs.

Problem Objective

As a data scientist, you should perform exploratory data analysis and perform cluster analysis to create cohorts of songs. The goal is to gain a better understanding of the various factors that contribute to creating a cohort of songs.

In [161…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import cm
import seaborn as sns
import plotly.express as px
import pickle as pkl
from scipy.special import boxcox
from sklearn.preprocessing import MinMaxScaler,StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from scipy.stats import skew
from sklearn.metrics import silhouette_score
from scipy.stats.mstats import winsorize
```

In [162…

```
# Configs
DATA_PATH='./dataset/'
IMG_PATH='./plots/'
PKL_PATH='./pkl/'
CSV_PATH='./csvs/'
RESOLUTION=200

sns.set()

def save_plots(fig_name):
 plt.savefig(IMG_PATH+fig_name,dpi=RESOLUTION)
```

Load the Dataset

In [163…
```
# Read the dataset and identify the shape and datatypes of the data
song_dat=pd.read_excel(DATA_PATH+'1673873388_rolling_stones_spotify.xlsx',index_col
print(f'Shape: {song_dat.shape}\n------------------')
print(f'Data Types:\n-------------------\n {song_dat.dtypes}')
song_dat.head()
```

```
Shape: (1610, 17)
------------------
Data Types:
-------------------
 name object
album object
release_date datetime64[ns]
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
```

Out[163]:

	name	album	release_date	track_number	id	uri
0	Concert Intro Music - Live	Licked Live In NYC	2022-06-10	1	2IEkywL J4ykbhi1yRQvmsT	spotify:track:2IEkywL J
1	Street Fighting Man - Live	Licked Live In NYC	2022-06-10	2	6GVgVJBKkGJoRfarYRvGTU	spotify:track:6GVgVJBK
2	Start Me Up - Live	Licked Live In NYC	2022-06-10	3	1Lu761pZ0dBTGpzxaQoZNW	spotify:track:1Lu761pZ0d
3	If You Can't Rock Me - Live	Licked Live In NYC	2022-06-10	4	1agTQzOTUnGNggyckEqiDH	spotify:track:1agTQzOTU
4	Donâ€™t Stop - Live	Licked Live In NYC	2022-06-10	5	7piGJR8YndQBQWVXv6KtQw	spotify:track:7piGJR8Ynd

Initial data inspection and data cleaning:

Check whether the data has duplicates, missing values, irrelevant (erroneous entries) values, or outliers.

In [164...
```
song_dat['release_date']=pd.to_datetime(song_dat['release_date'])
```

In [165...
```
def checking_data():
 null_cols=[]
 n_duplicates=0
 numerical_columns=[]
 non_numerical_columns=[]

#Checking null value columns
 cols1=[x for x in song_dat if song_dat[x].isna().sum()!=0]

#Checking duplicate value
n_duplicates=song_dat.duplicated().sum()

#Checking numerical columns
 numerical_columns=song_dat.select_dtypes('number').columns

#Checking non numerical columns
 non_numerical_columns=song_dat.select_dtypes('object').columns

if(len(cols1)==0):
 print('There was not any missing value')
 else:
 print(f'The columns with having null value are {cols1}')

print(f'Duplicates: {n_duplicates}')
 print(f'Numerical columns: {numerical_columns.values.tolist()}')
 print(f'Non Numerical columns: {non_numerical_columns.tolist()}')

return numerical_columns,non_numerical_columns



#Checking the dataset
numerical_columns,non_numerical_columns=checking_data()
```

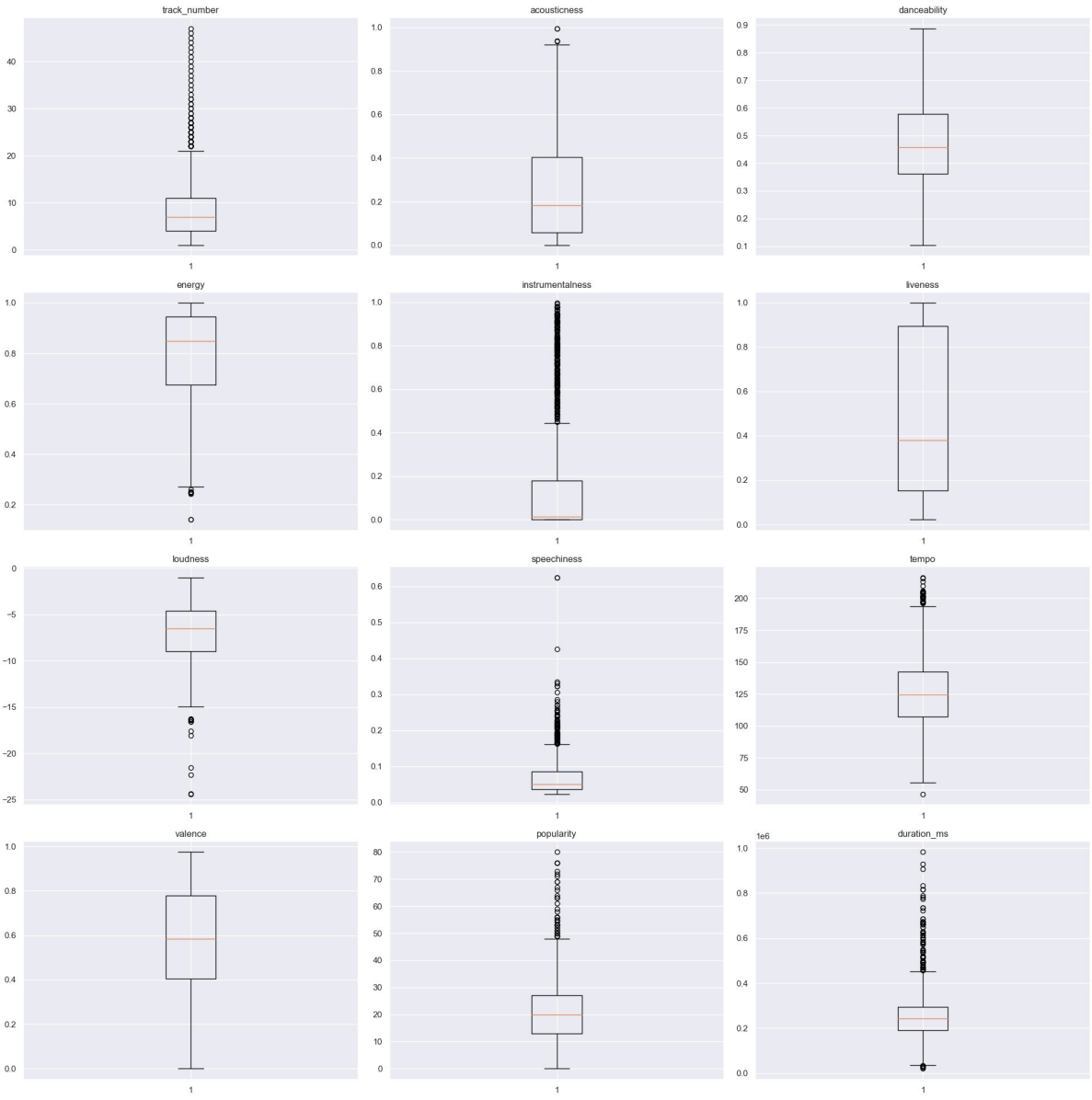
```
There was not any missing value
Duplicates: 0
Numerical columns: ['track_number', 'acousticness', 'danceability', 'energy', 'ins
trumentalness', 'liveness', 'loudness', 'speechiness', 'tempo', 'valence', 'popula
rity', 'duration_ms']
Non Numerical columns: ['name', 'album', 'id', 'uri']
```

In [166...
```
# Creating a datafrom for numerical_values
continuous_df=song_dat[numerical_columns]
```

In [167...
```
# Checking Outliers
def outlier_checker(coutinuous_df,fig_name):
 '''
 parameter type:positional and the given dataframe must contain numeric data
 This function gives the skewness of the given column and plot the box plot
'''
 print(f'Snewness Scores:\n {continuous_df.skew(axis=0)}')
 # Ploting box plot
 plt.figure(figsize=(20,20))
 for indx,col in enumerate(continuous_df):
 plt.subplot(4,3,indx+1)
 plt.title(col)
 plt.boxplot(continuous_df[col])
 plt.savefig('Skewness')
 plt.tight_layout()
 save_plots(fig_name)
```

```
outlier_checker(continuous_df,'box plot1')
```

```
Snewness Scores:
 track_number 1.777241
acousticness 0.868785
danceability 0.162052
energy -0.941379
instrumentalness 1.667856
liveness 0.220033
loudness -0.888034
speechiness 3.230335
tempo 0.361867
valence -0.195988
popularity 0.883930
duration_ms 1.924778
dtype: float64
```



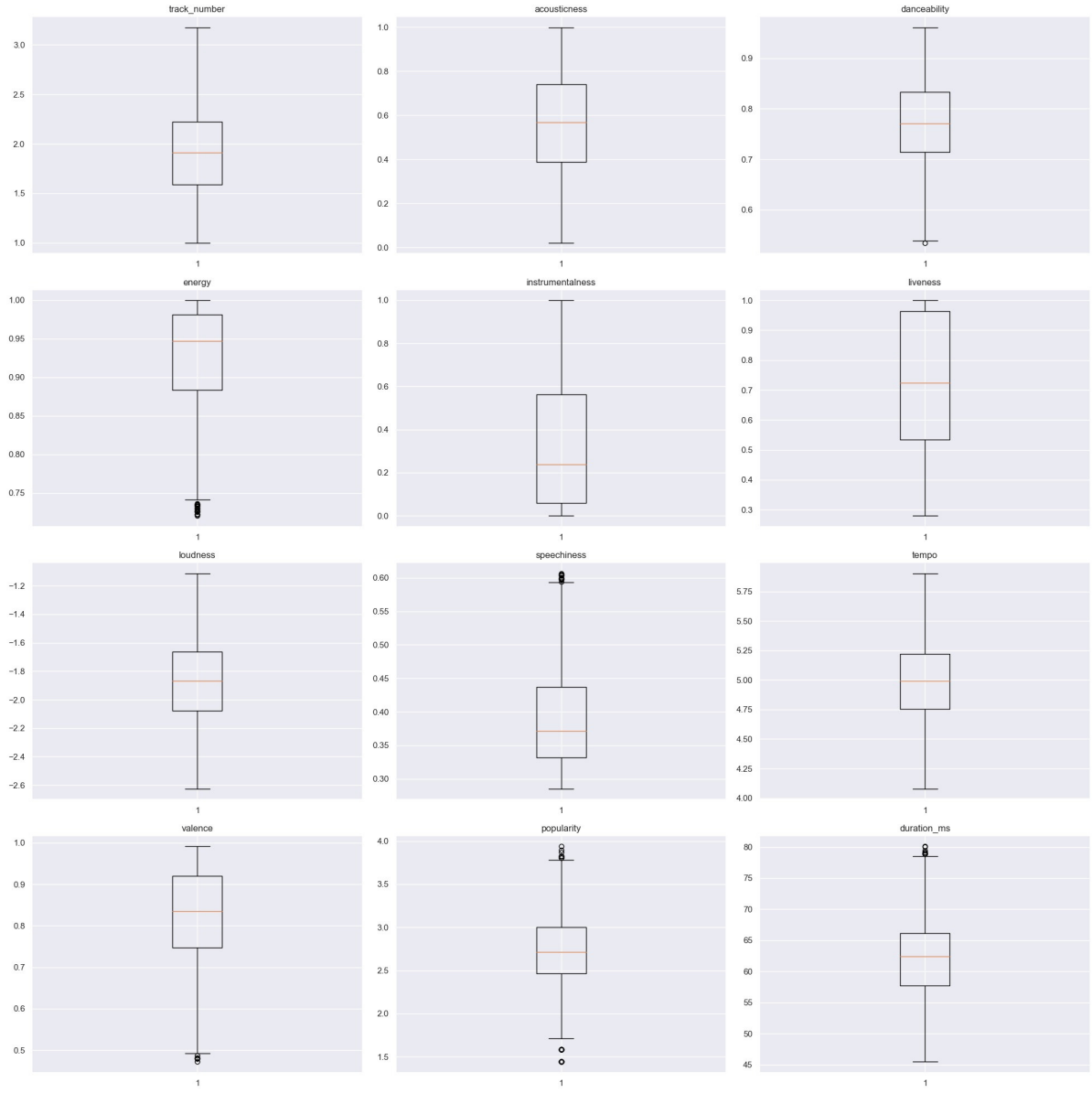
Depending on your finding, clean the data for further processing.

In [168…

```
# Treating outliers
# Taking cube root
continuous_df=np.cbrt(continuous_df)
```

```
# I use the inter quantile range to remove the outliers
for col in continuous_df.columns:
 q1,q3=np.percentile(continuous_df[col],[25,75])
 iqr=q3-q1
 # finding the upper bound and lower bound
 upper_bound=q3+1.5*iqr
 lower_bound=q1-1.5*iqr

 # getting the outlier data points
 outliers=continuous_df[col][(continuous_df[col]>upper_bound)| (continuous_df[co
 continuous_df.loc[outliers.index,col]=np.median(continuous_df[col])

# calling the outlier_checker function to check the data distribution again after t
# outlier datapoints with the median
outlier_checker(continuous_df,'boxplot2')
```

```
Snewness Scores:
 track_number -0.017543
acousticness -0.247978
danceability -0.253348
energy -1.058083
instrumentalness 0.731103
liveness -0.171067
loudness 0.028898
speechiness 0.895138
tempo -0.036787
valence -0.652202
popularity -0.462835
duration_ms 0.057372
dtype: float64
```

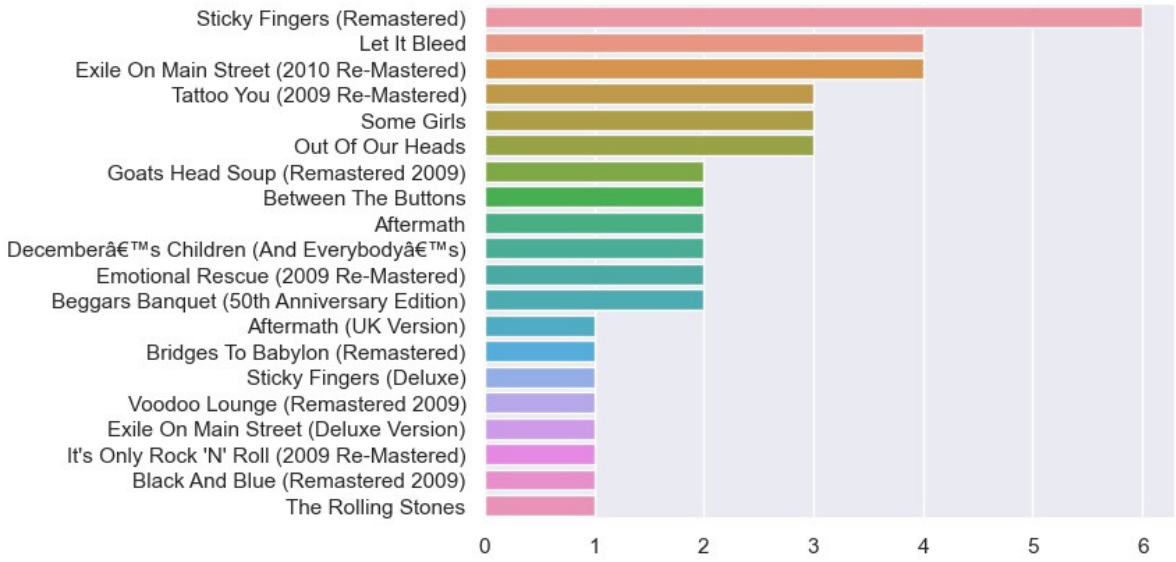


Perform Exploratory Data Analysis and Feature Engineering:

Use appropriate visualizations to find out which two albums should be recommended to anyone based on the number of popular song in an album.

In [169…

```
popularity_threshold=50
# Since the popularity rating maximum was 80 So lets take all the song that's popul
popular_songs=song_dat.loc[song_dat['popularity']>=popularity_threshold]

top_songs=popular_songs['album'].value_counts().sort_values(ascending=False)

sns.barplot(x=top_songs.values,y=top_songs.index)

save_plots('Popular albums')
plt.show()
```



Findings

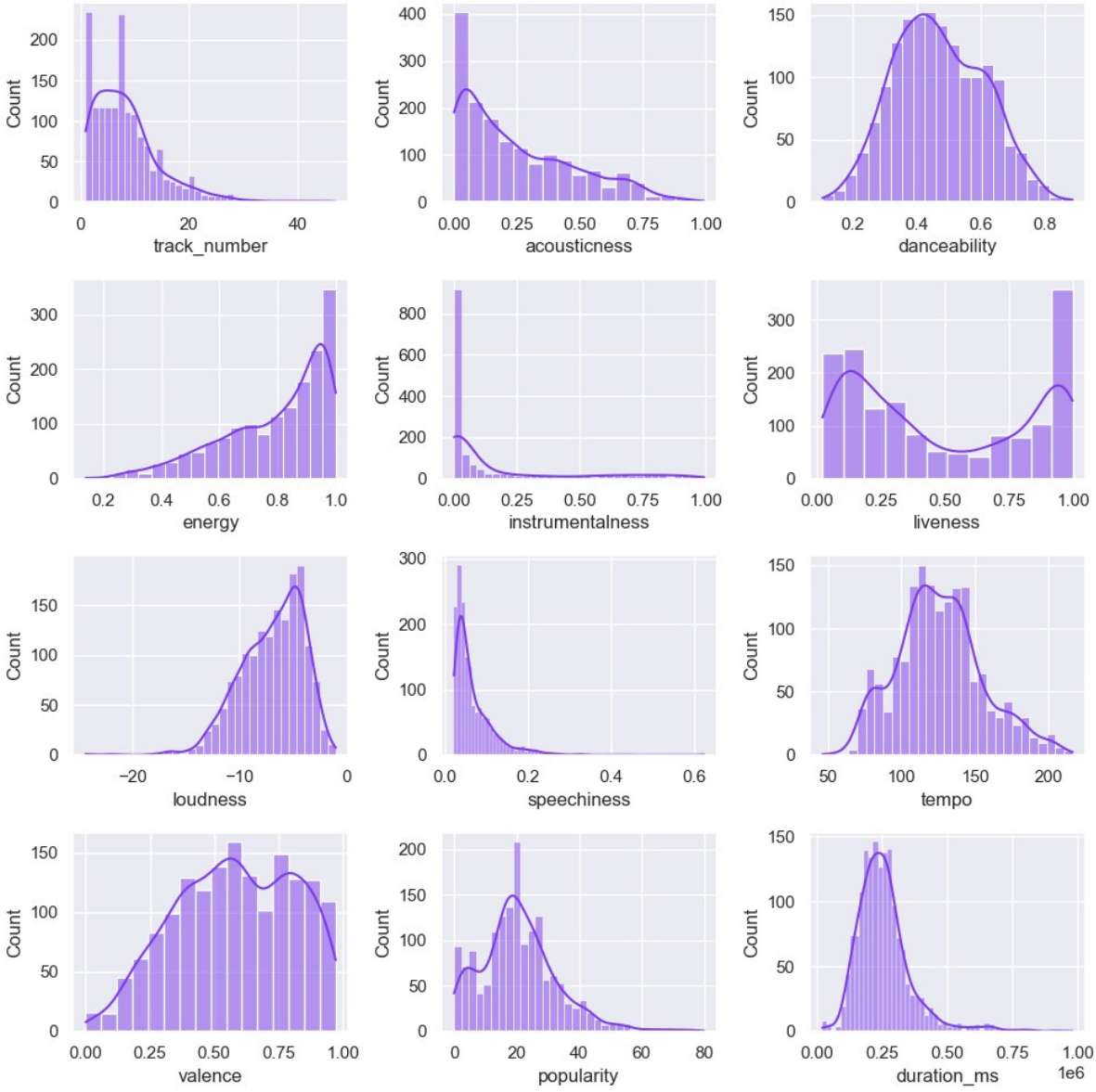
- The best two albums recommend based on the number of popular songs in the album are
- 1 Sticky Fingers (Remastered)
- 2 Let It Bleed/Exile On Main Street Re-Mastered

Perform exploratory data analysis to dive deeper into different features of songs and identify the pattern.

Check the data distribution for numerical features.

In [170…

```
fig,ax=plt.subplots(4,3,figsize=(10,10))
for col,ax in zip(song_dat.select_dtypes('number').columns,ax.flatten()):
sns.histplot(song_dat[col],kde=True,color='#7c3aed',ax=ax)
# sns.histplot(song_dat[col],kde=True,color='#16a34a',ax=ax)
plt.tight_layout()
plt.show()
```



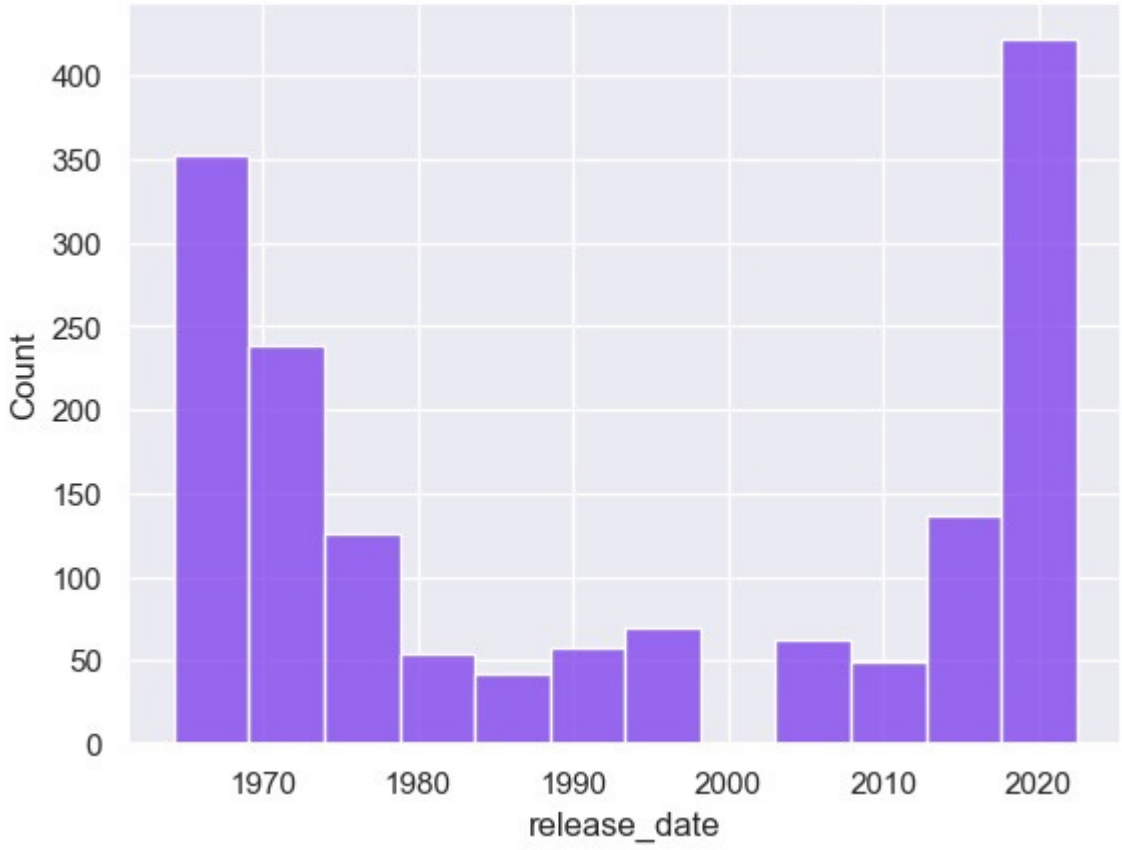
Finding:

- There was some feature who have skewed data distribution

Check the data distribution for release date.

In [171…

```
sns.histplot(song_dat['release_date'],color='#7c3aed')
plt.show()
```



Finding:

- There are number of song release in the year 1970 and 2020
- Years 2000 have lease number of release song. We can dig deeper to check why there is less release in song in the year 2000 what happen to this year.

Checking the numbers of unique value for the columns.

In [172…
```
print(f'Unique values: \n{song_dat.nunique().sort_values()}')
```

```
Unique values:
track_number 47
release_date 57
popularity 69
album 90
energy 511
danceability 518
speechiness 655
valence 701
liveness 757
acousticness 933
name 954
instrumentalness 1134
duration_ms 1320
loudness 1329
tempo 1424
uri 1610
id 1610
dtype: int64
```

Finding:

- We notice that some columns have less number or unique value and some are high. Less number of unique values indicates the data is categorical in nature.
- Even there are song that the same name, it may be because a song can have different version and one can cover the song

Describing the dataset

In [173…
```
song_dat.describe()
```

Out[173]:

	track_number	acousticness	danceability	energy	instrumentalness	liveness	lou
count	1610.000000	1610.000000	1610.000000	1610.000000	1610.000000	1610.00000	1610.0
mean	8.613665	0.250475	0.468860	0.792352	0.164170	0.49173	-6.9
std	6.560220	0.227397	0.141775	0.179886	0.276249	0.34910	2.9
min	1.000000	0.000009	0.104000	0.141000	0.000000	0.02190	-24.4
25%	4.000000	0.058350	0.362250	0.674000	0.000219	0.15300	-8.9
50%	7.000000	0.183000	0.458000	0.848500	0.013750	0.37950	-6.5
75%	11.000000	0.403750	0.578000	0.945000	0.179000	0.89375	-4.6
max	47.000000	0.994000	0.887000	0.999000	0.996000	0.99800	-1.0

In [174…
```
song_dat.describe(include='O')
```

Out[174]:

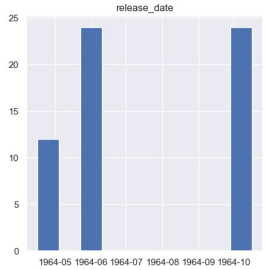
	name	album id	uri
count	1610	1610 1610	161
unique	954	90 1610	0
top	Brown Sugar - Live	Voodoo Lounge Uncut2IEkywLJ4ykbhi1yRQvmsT (Live)	spotify:track:2IEkywL J4ykbhi1yRQvmsT 0
freq	16	561	1

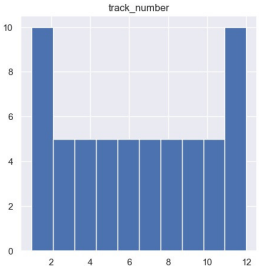
Show the data distribution for all the columns for the each years.

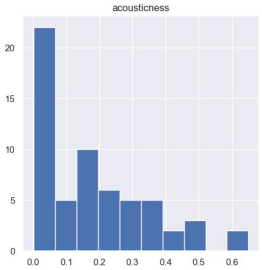
In [175…
```
# Find the years
years=song_dat['release_date'].dt.year.unique()

# Flip to arrange in increasing order
years=np.flip(years)
# Creating an array of dataframe based on the year as all the value that have the s
years_df=[]
for year in years:
 years_df.append(song_dat[song_dat['release_date'].dt.year==year])
```

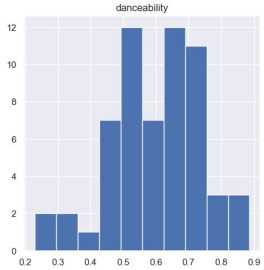
In [176…
```
for indx,df_ in enumerate(years):
 print(years[indx],"Data")
    years_df[indx].hist(figsize=(25,25))
save_plots(str(years[indx]))
 plt.show()
```

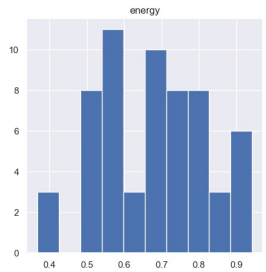
```
1964 Data
```

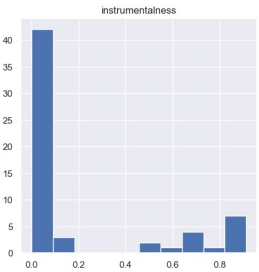

release_date
1964-05 1964-06 1964-07 1964-08 1964-09 1964-10

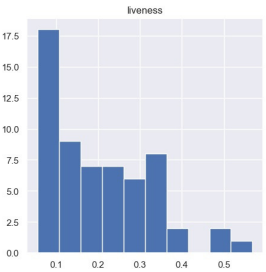

track_number

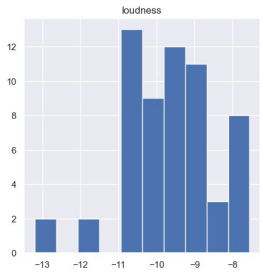

acousticness

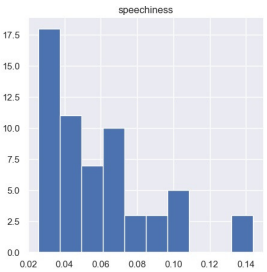

danceability

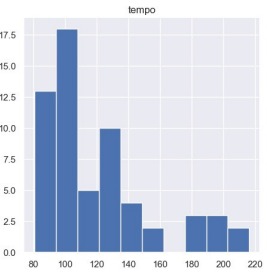

energy

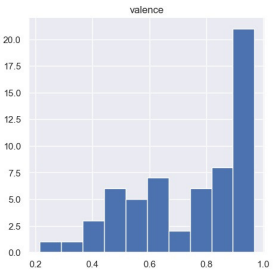

instrumentalness

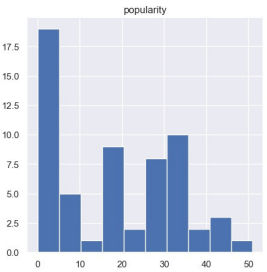

liveness

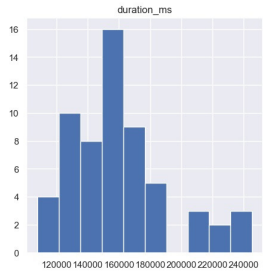

loudness


speechiness

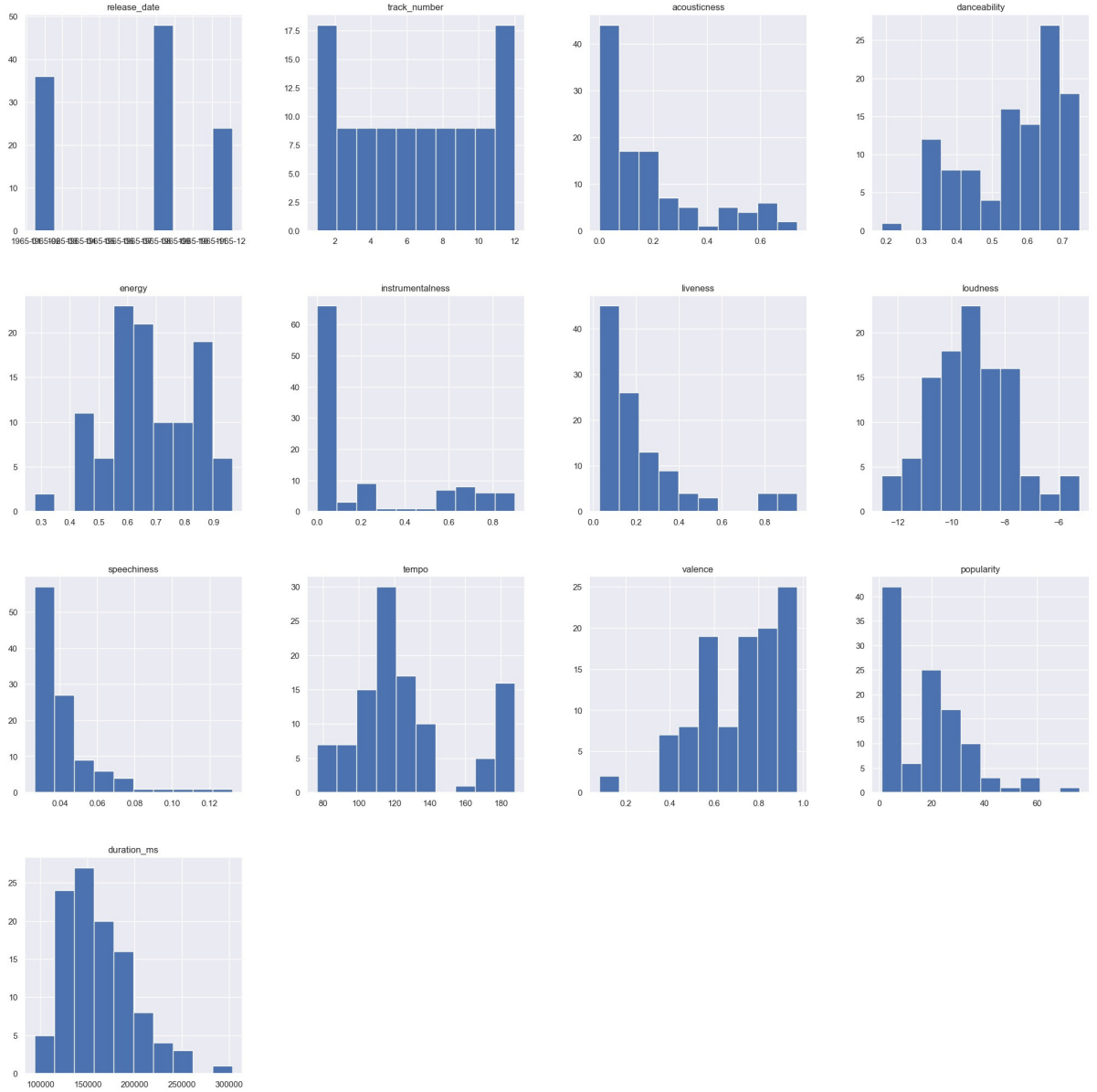

tempo


valence

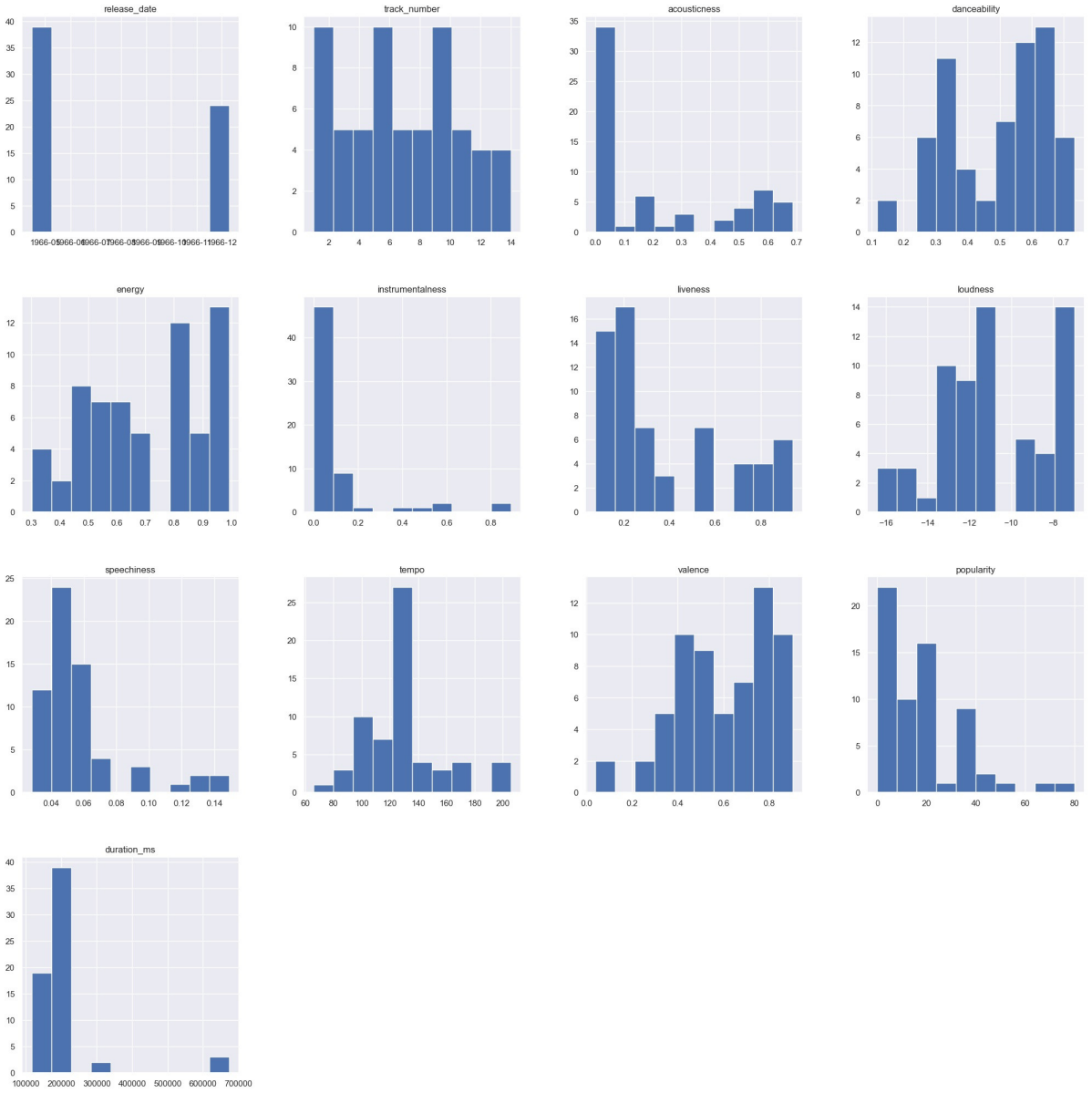

popularity


duration_ms

1965 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

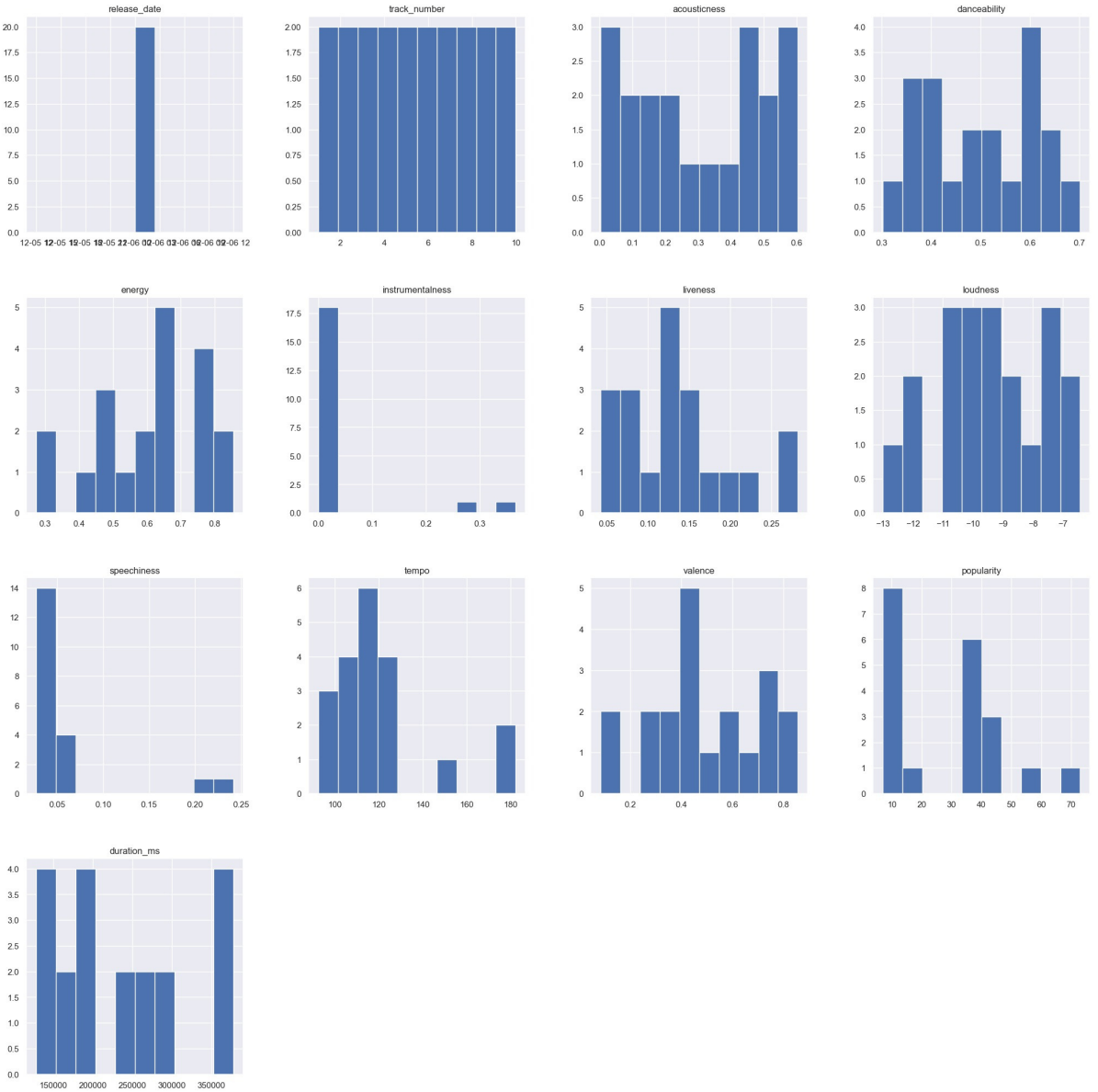
1966 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

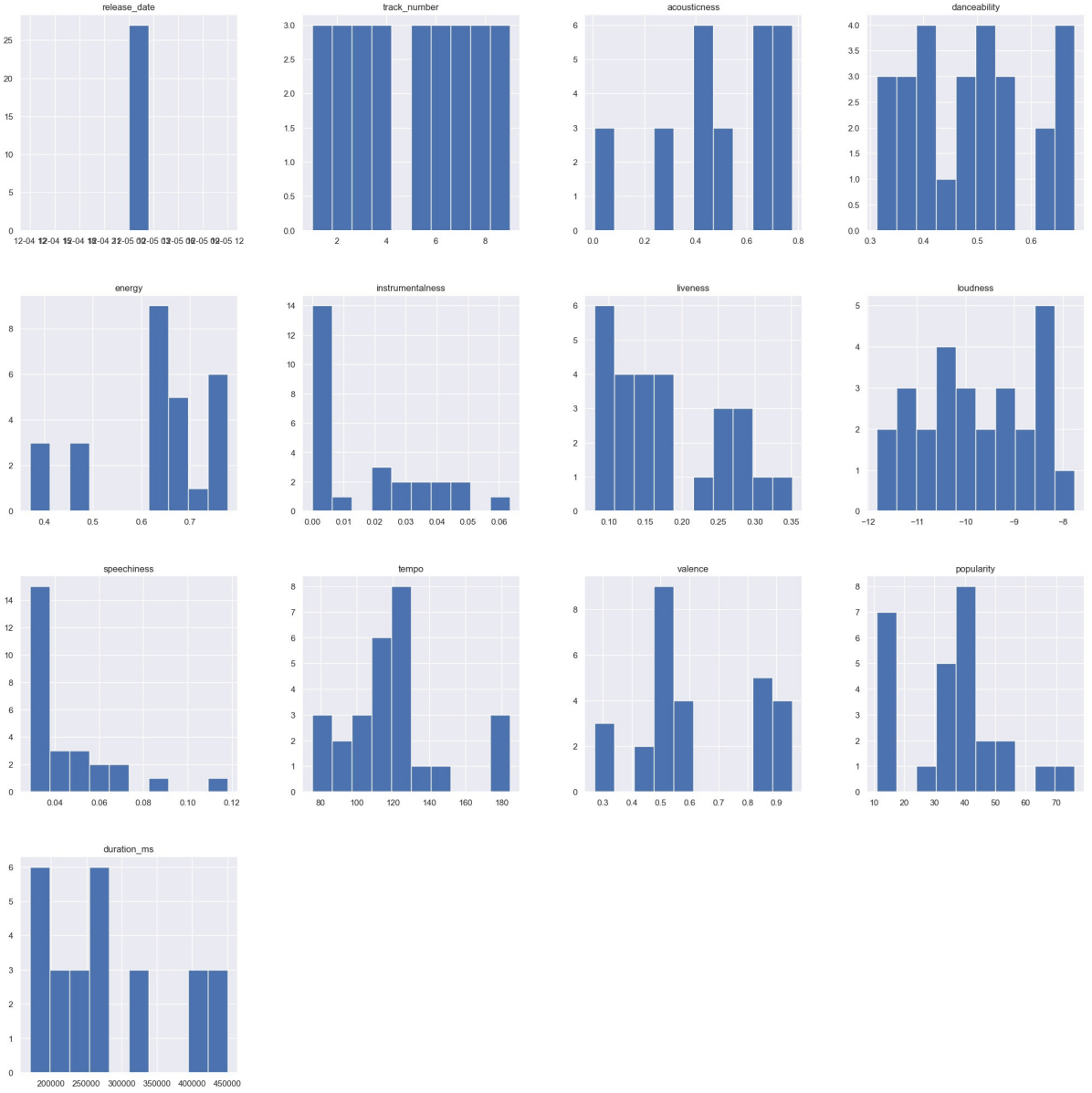
1967 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

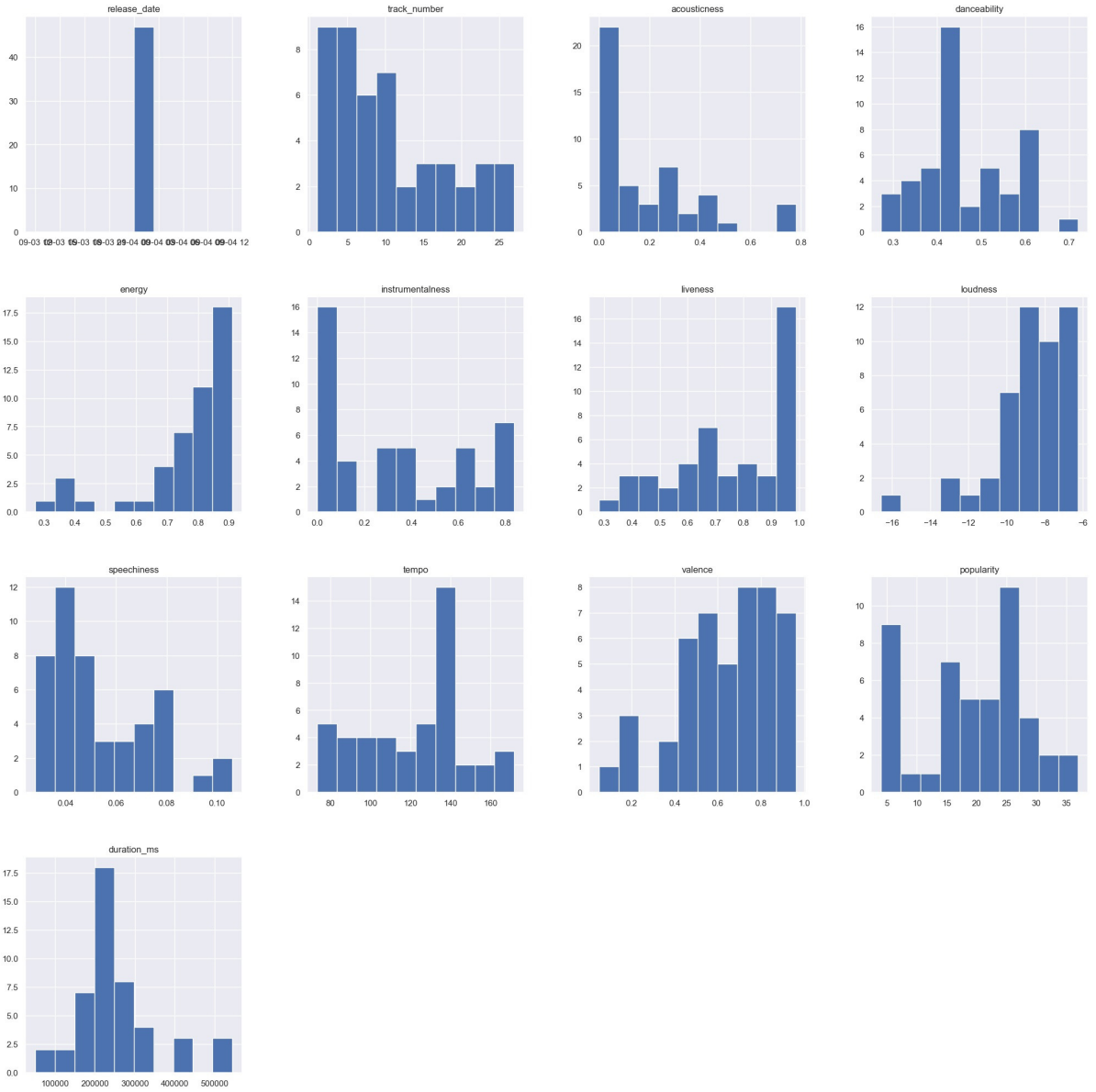
1968 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

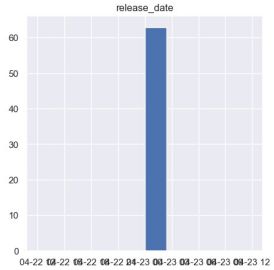
1969 Data

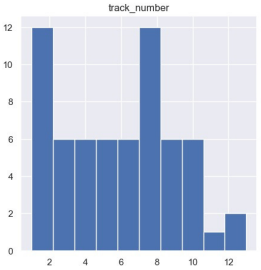

release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

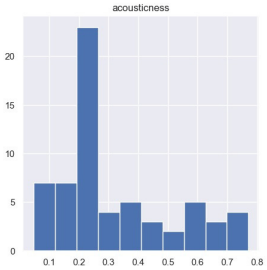
1970 Data

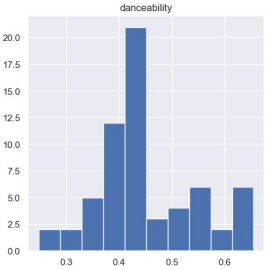

release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

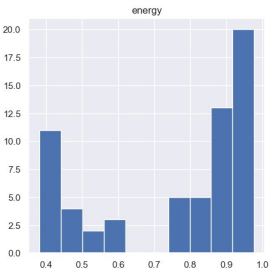
1971 Data

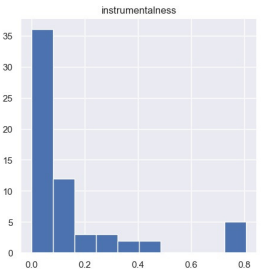

release_date

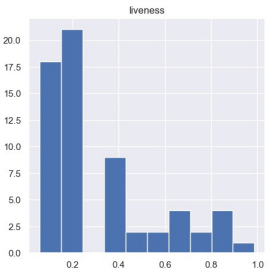

track_number

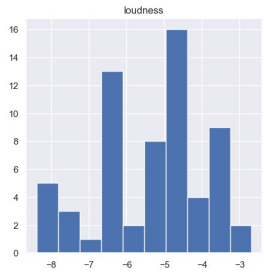

acousticness

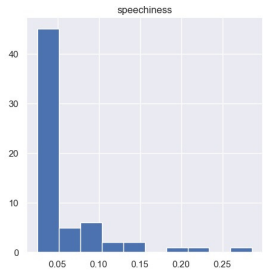

danceability

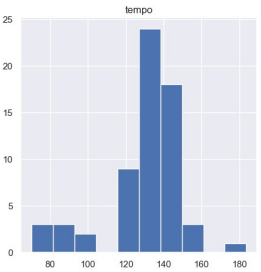

energy

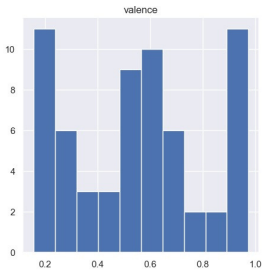

instrumentalness

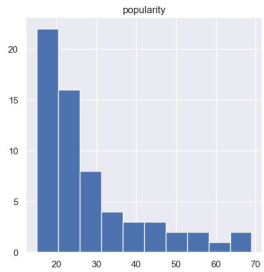

liveness

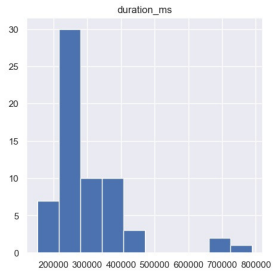

loudness


speechiness

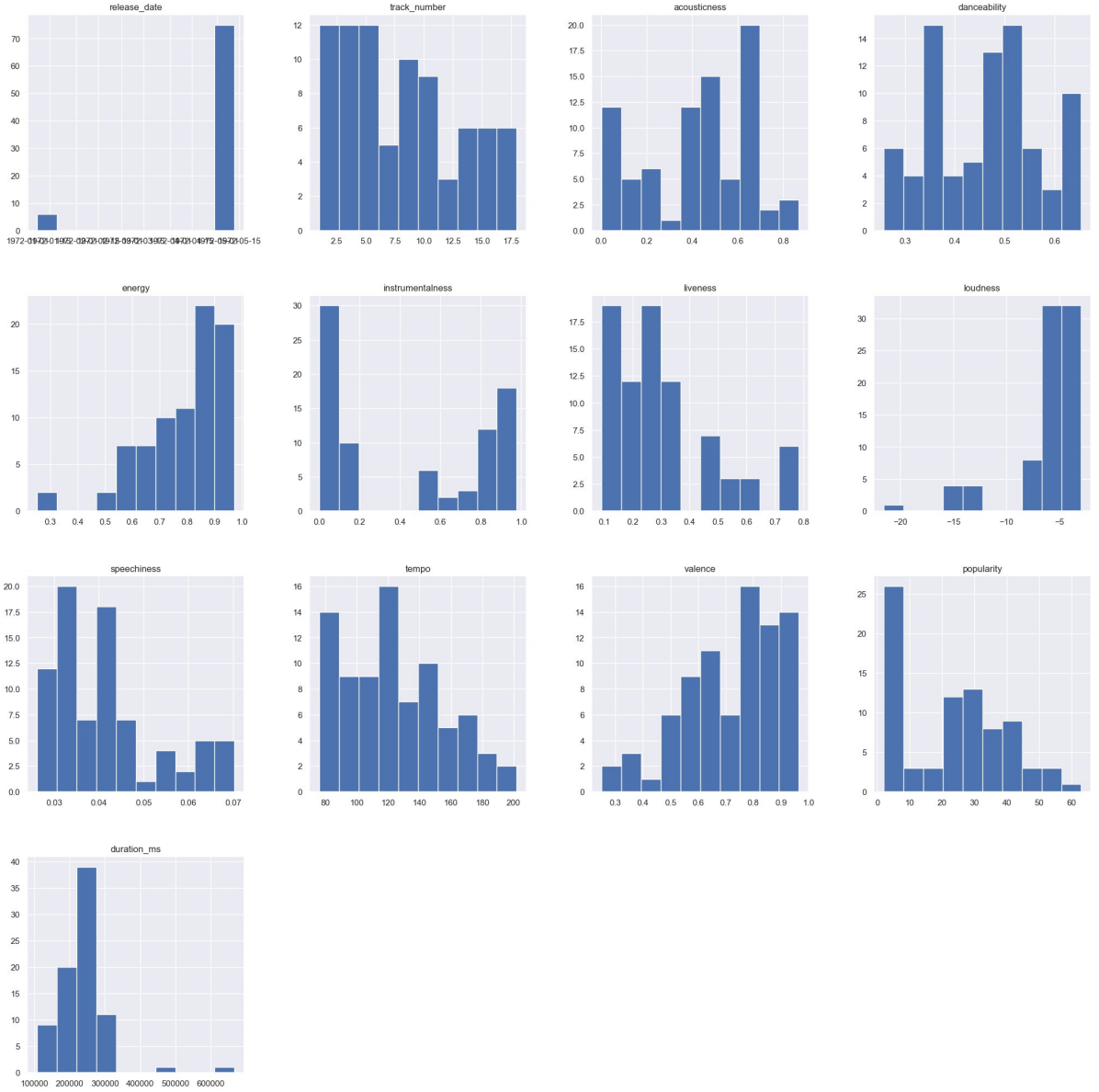

tempo


valence

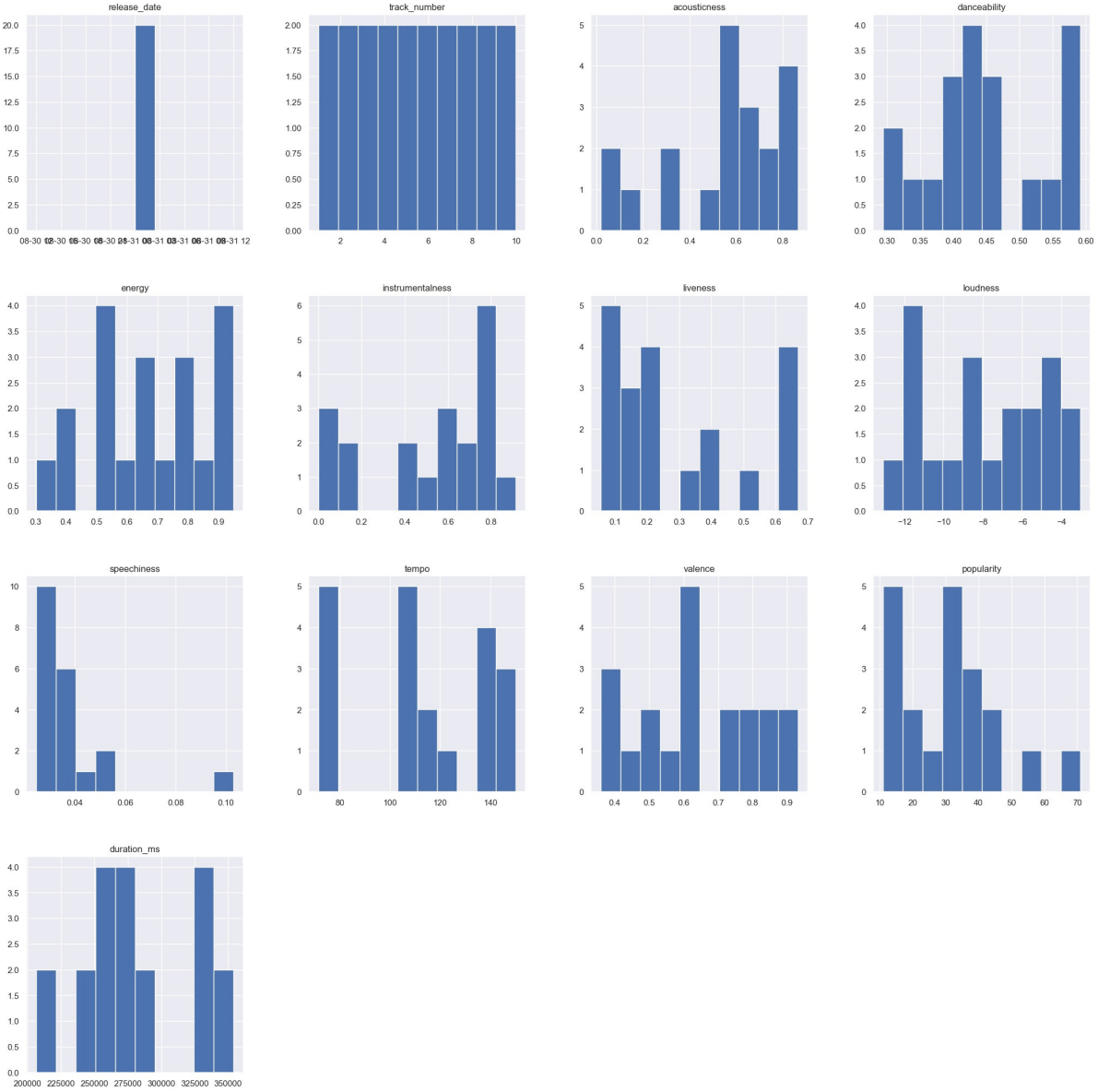

popularity


duration_ms

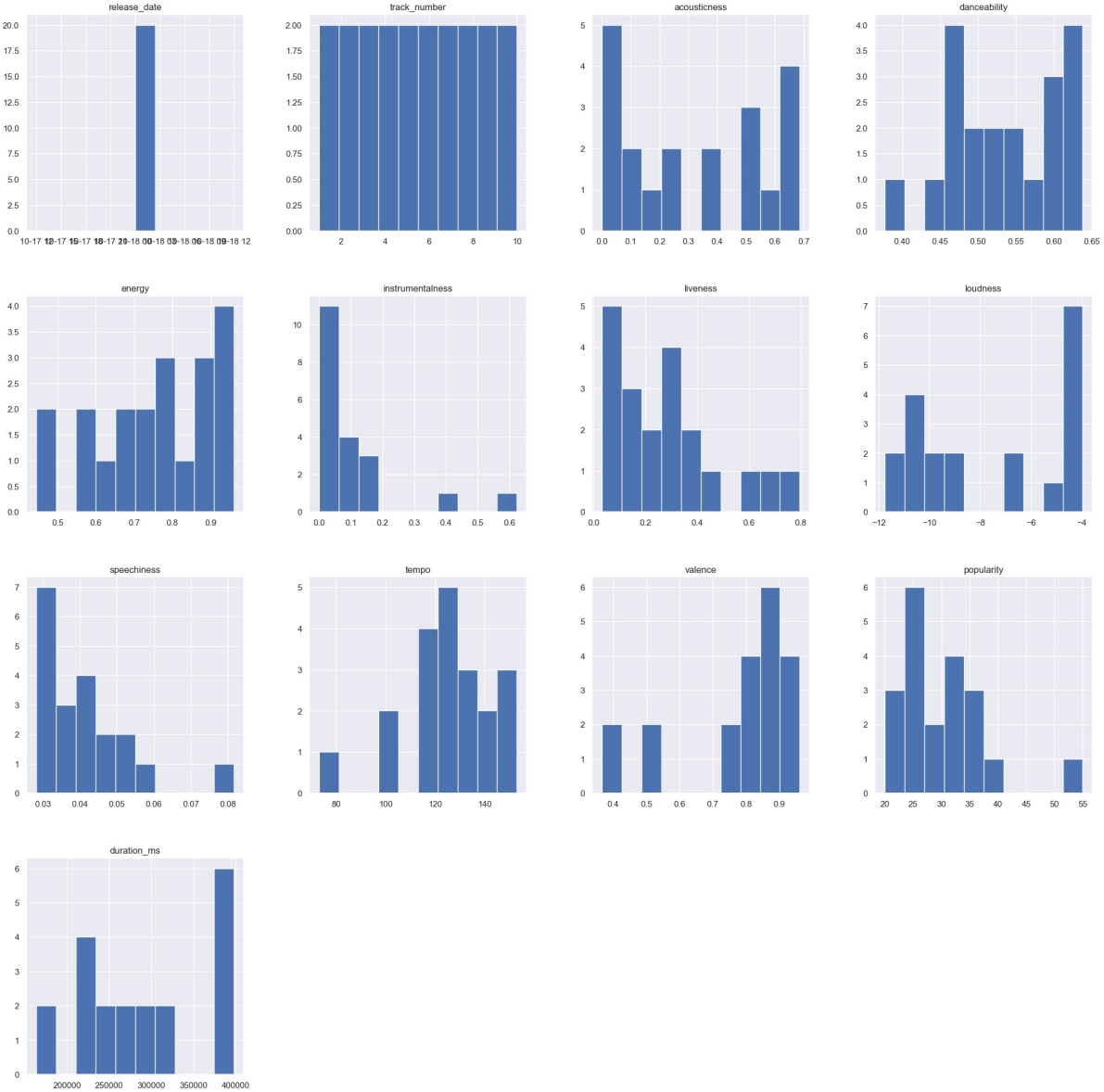
1972 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

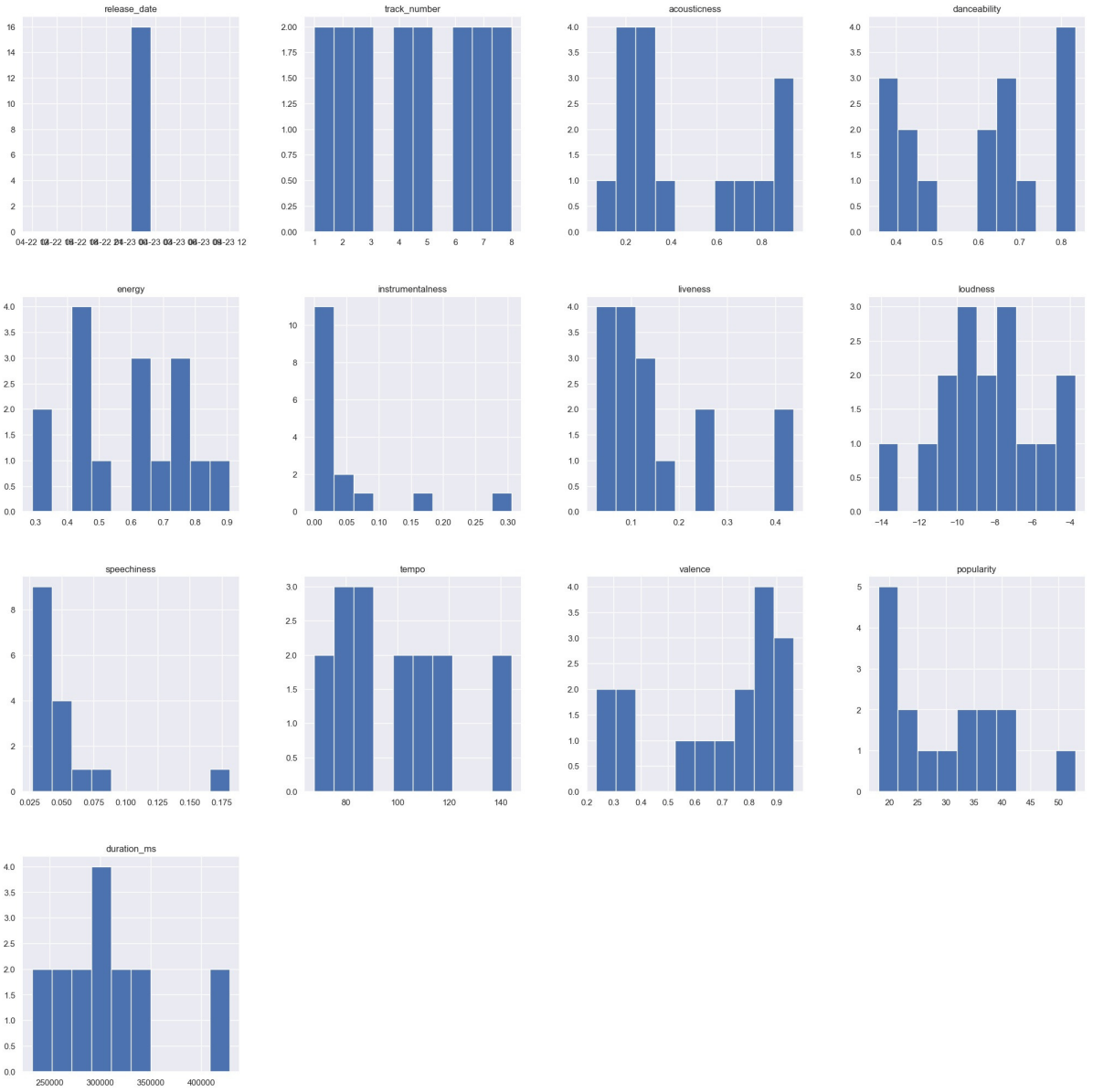
1973 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

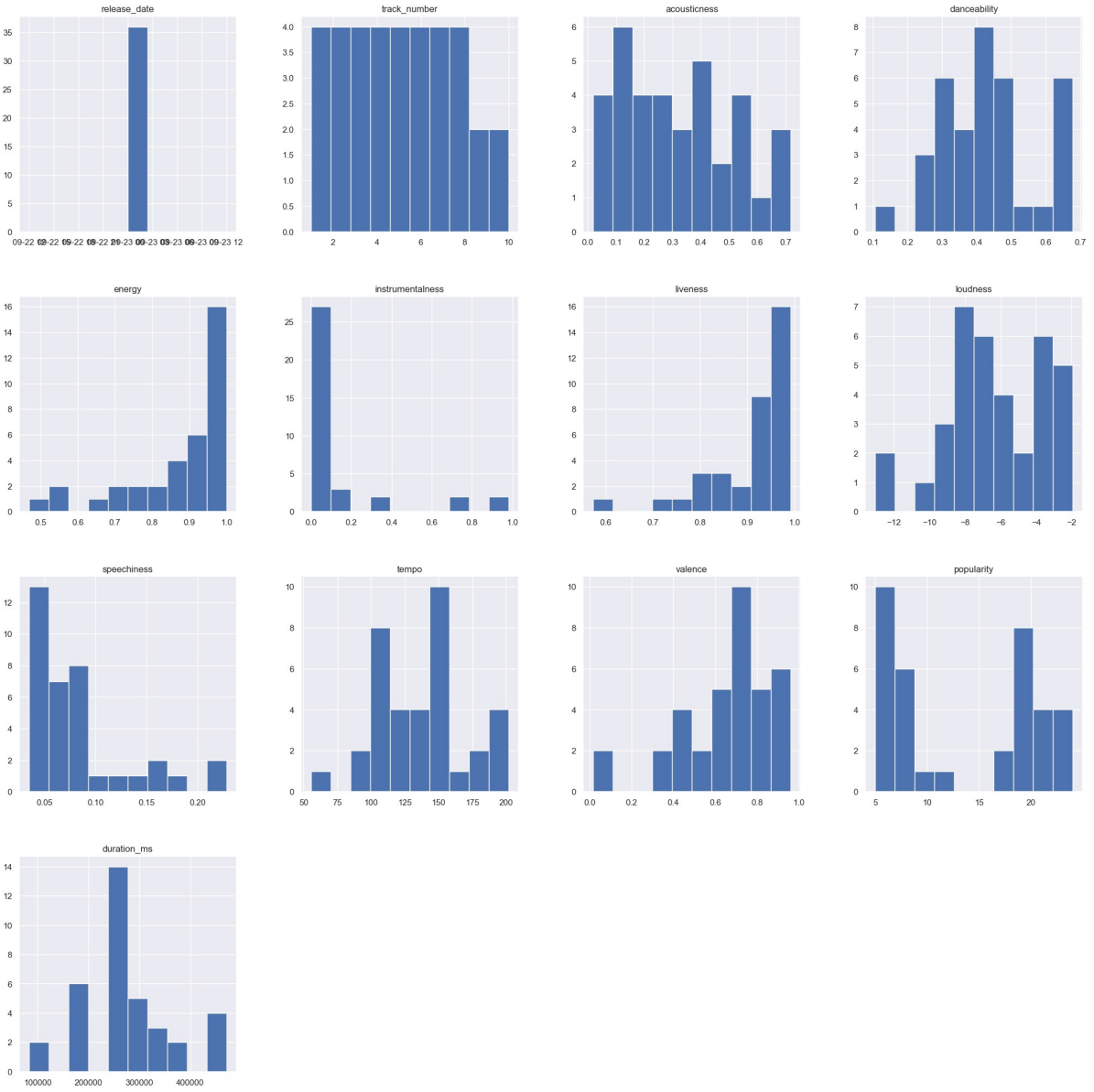
1974 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

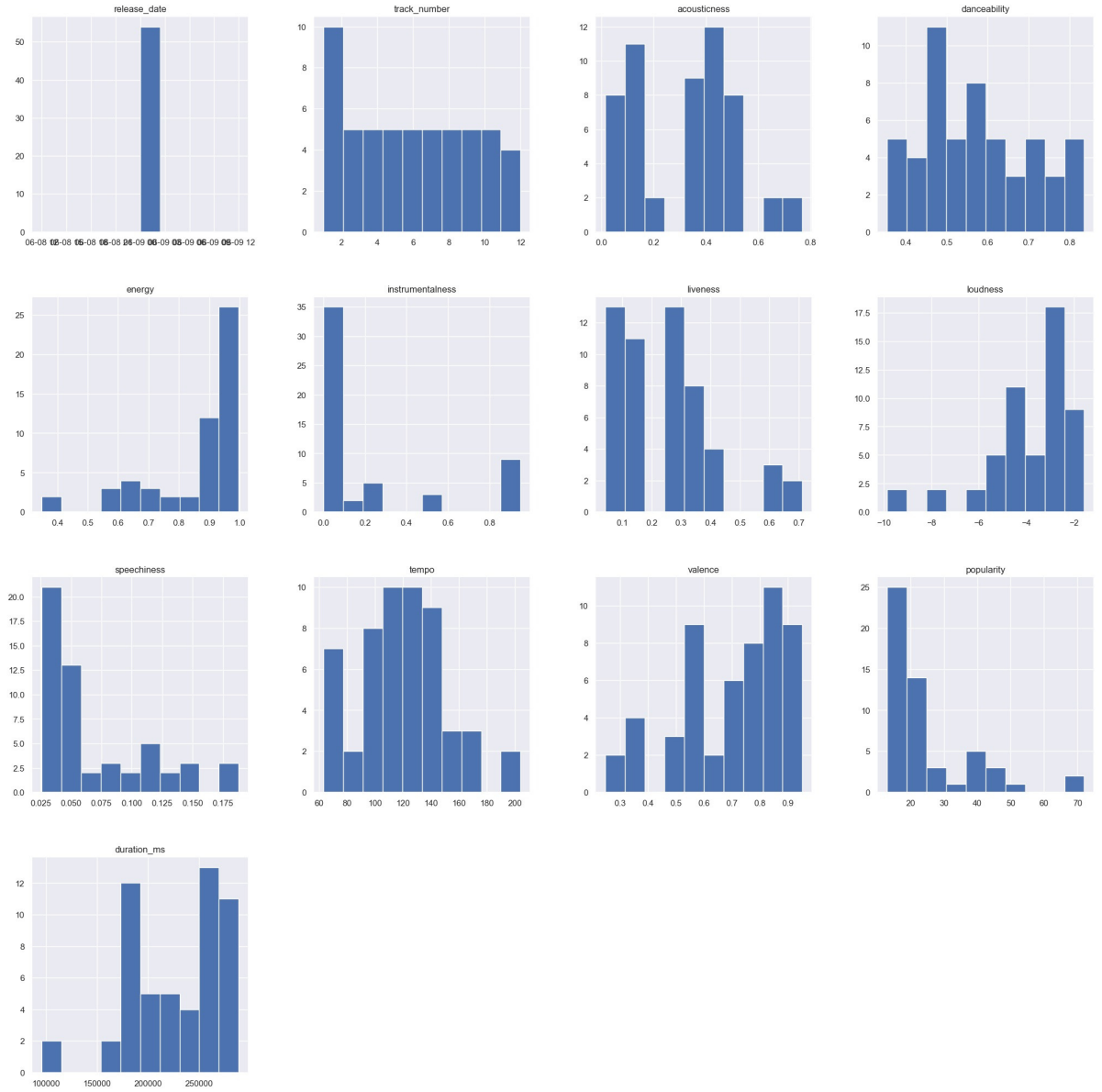
1976 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

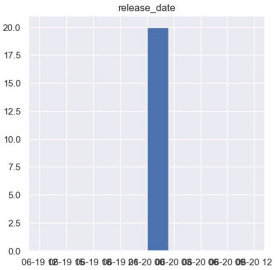
1977 Data

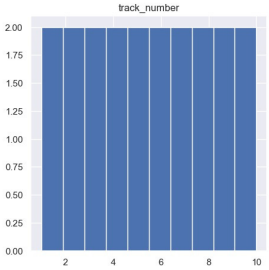

release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

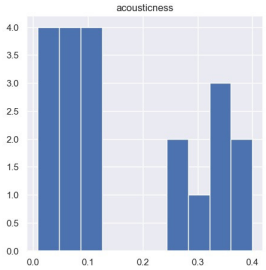
1978 Data

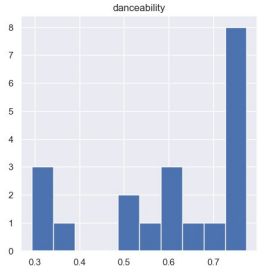

release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

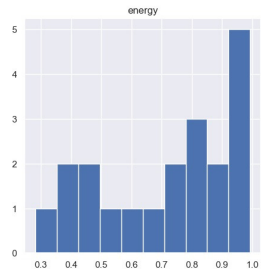
1980 Data

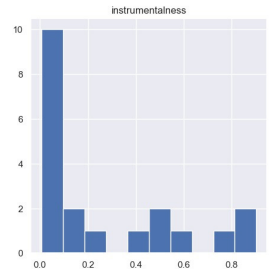

release_date

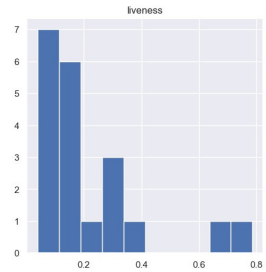

track_number

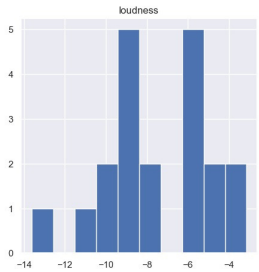

acousticness

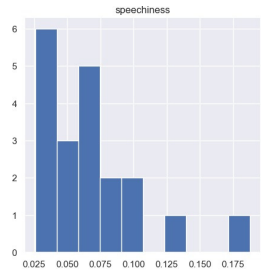

danceability

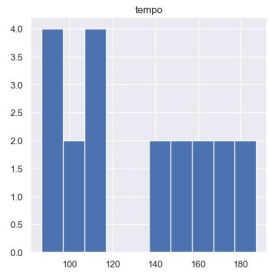

energy

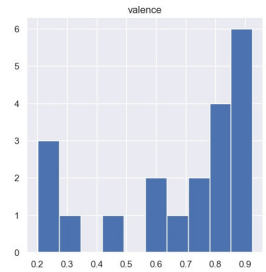

instrumentalness

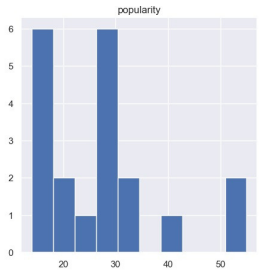

liveness

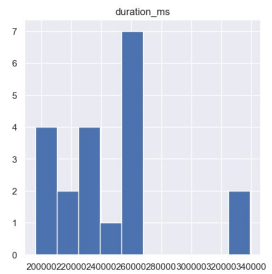

loudness


speechiness

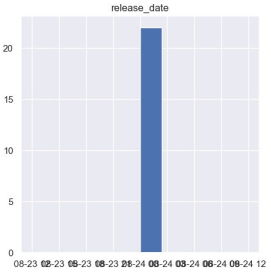

tempo

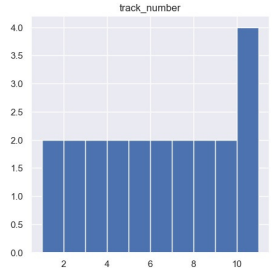

valence

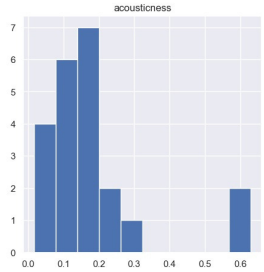

popularity

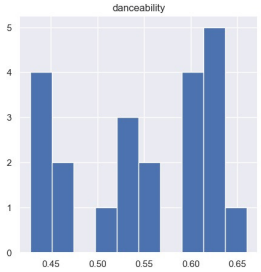

duration_ms

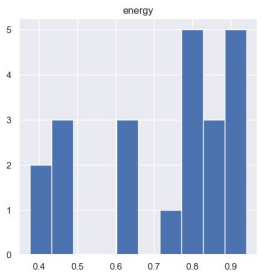
1981 Data

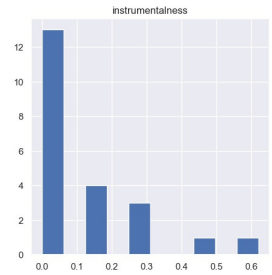

release_date

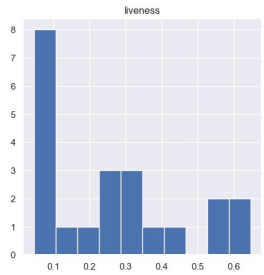

track_number

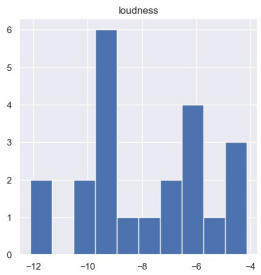

acousticness

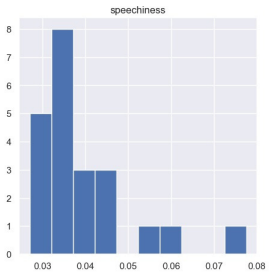

danceability

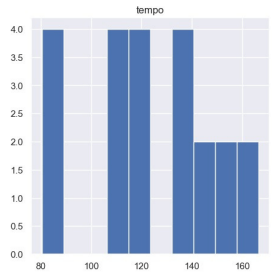

energy

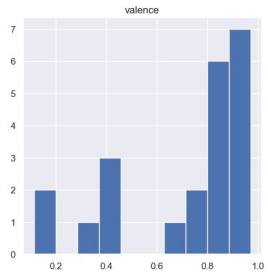

instrumentalness

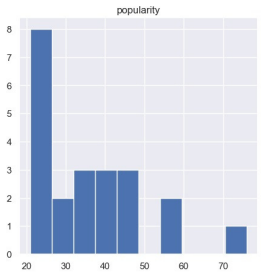

liveness

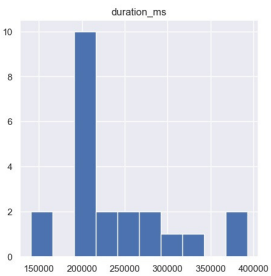

loudness


speechiness

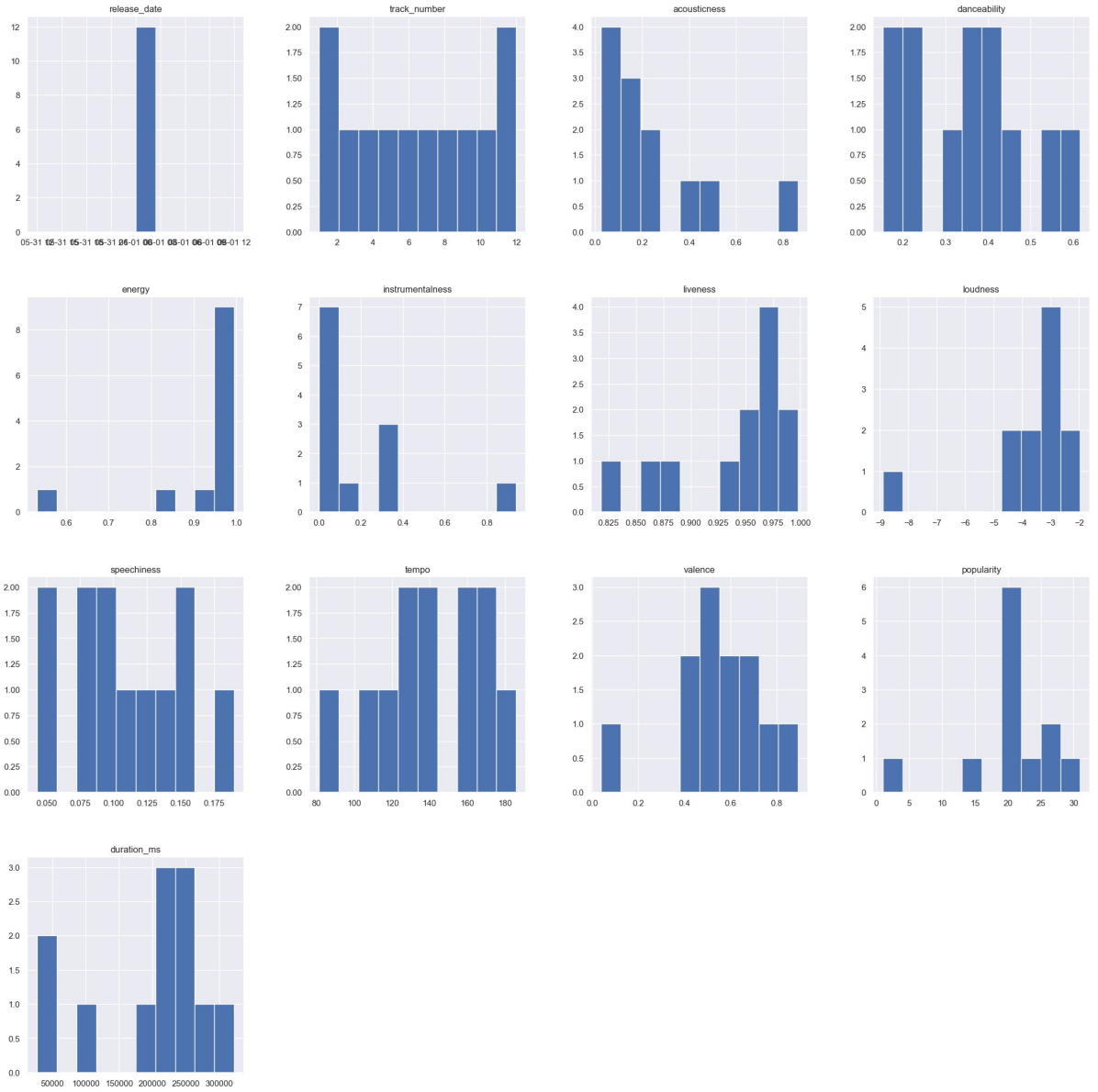

tempo


valence

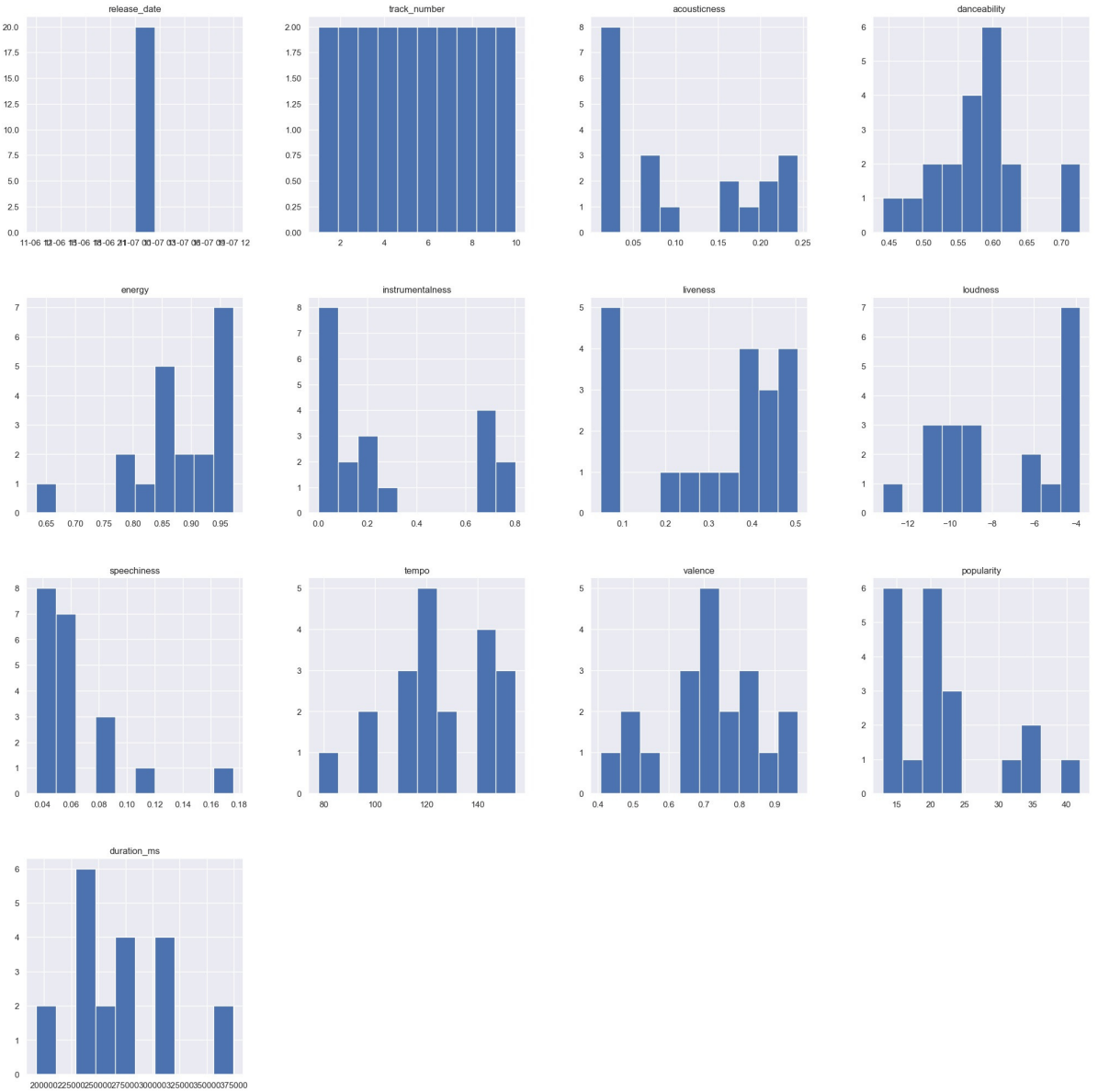

popularity


duration_ms

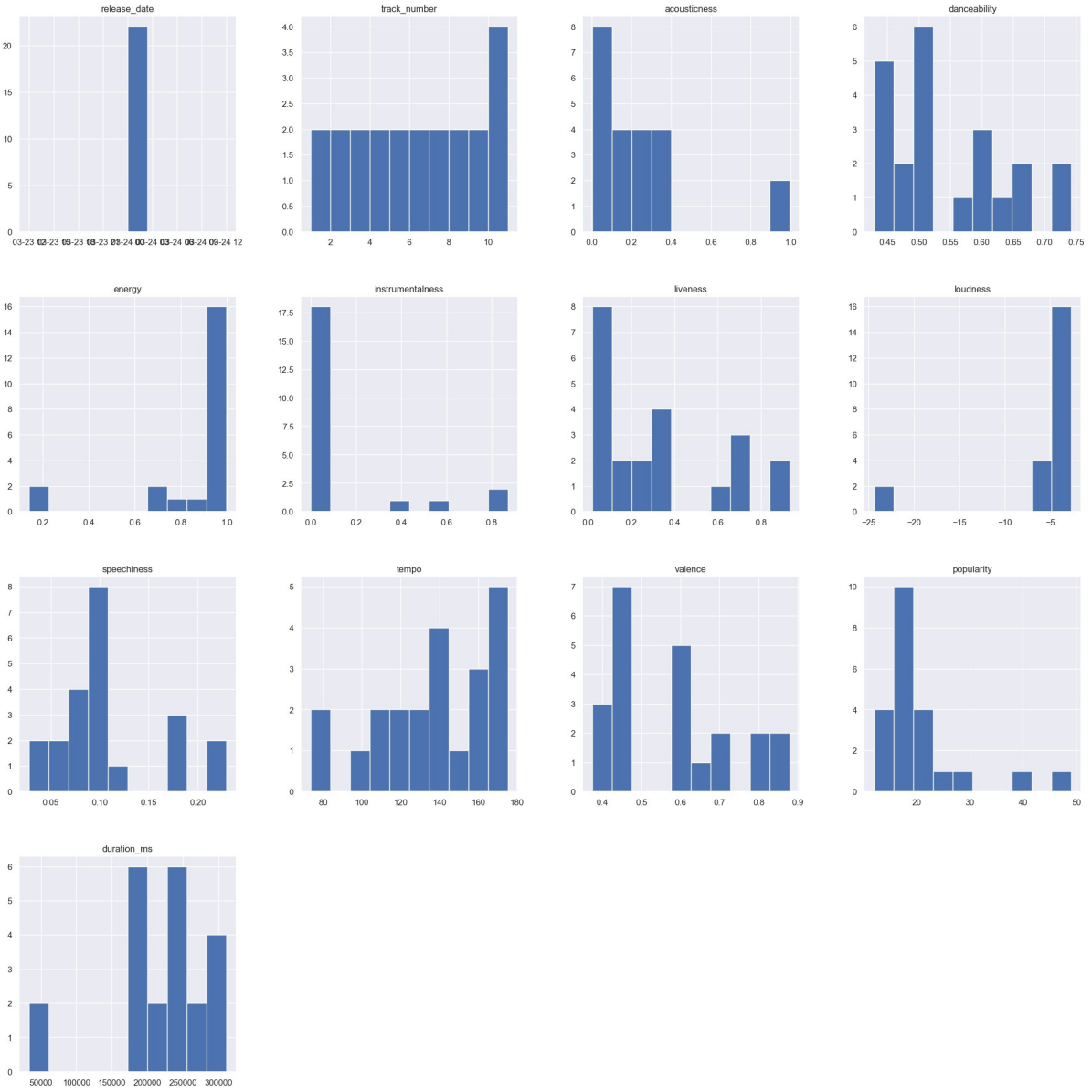
1982 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

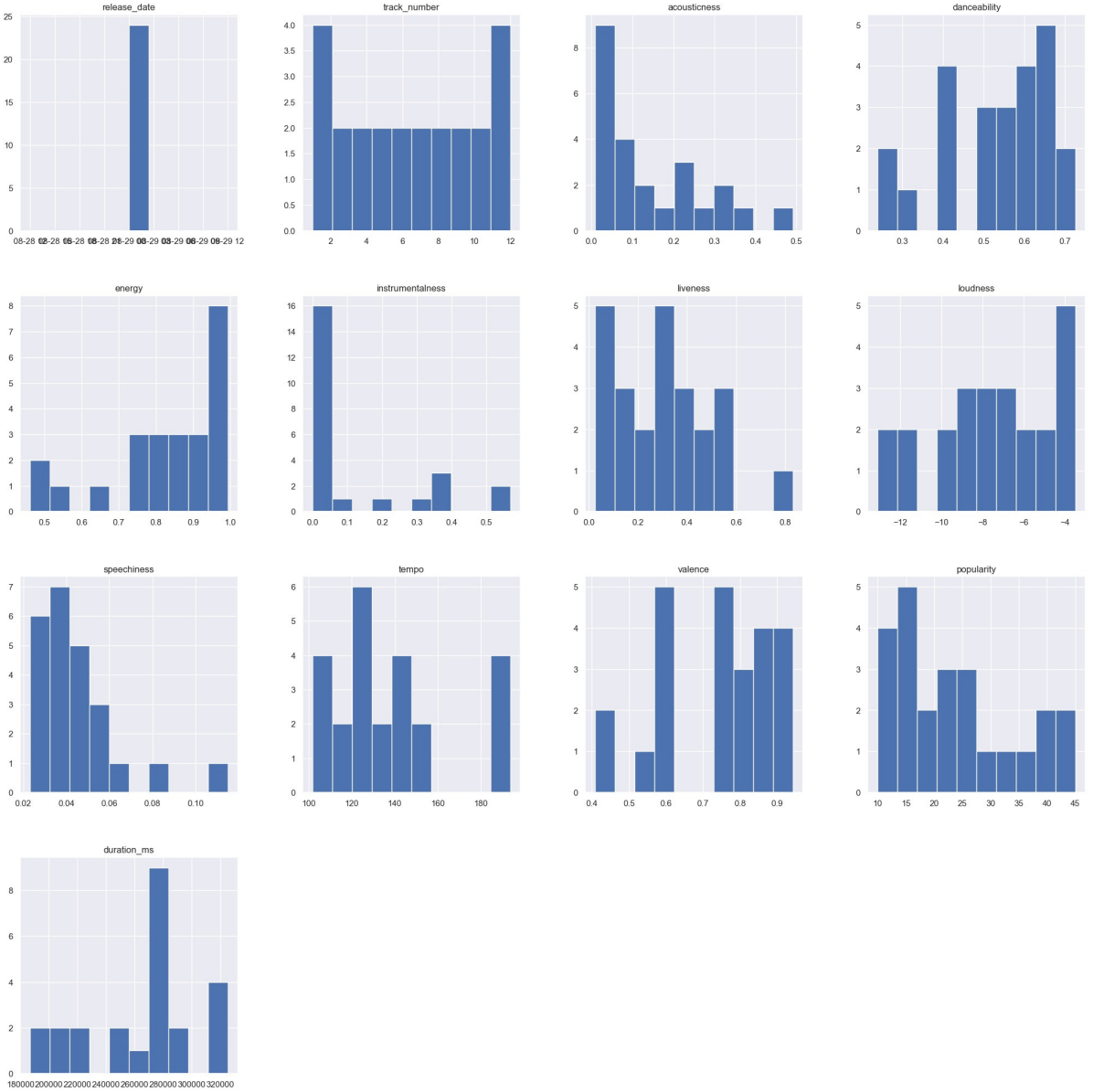
1983 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

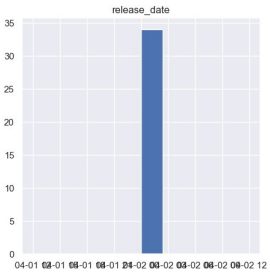
1986 Data

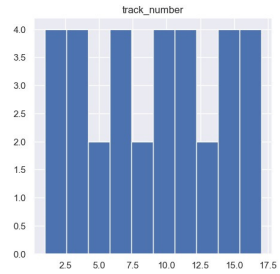

release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

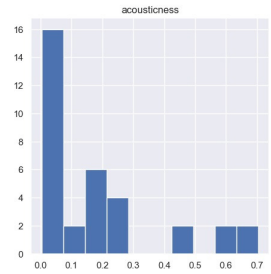
1989 Data

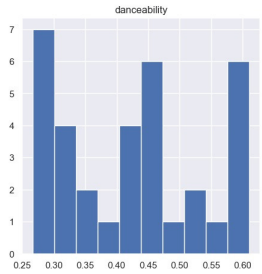

release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

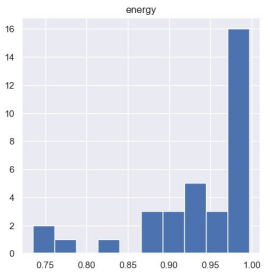
1991 Data

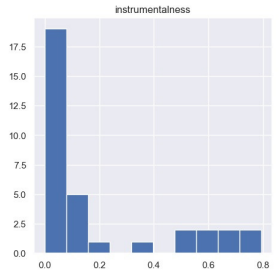

release_date

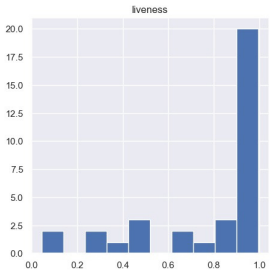

track_number

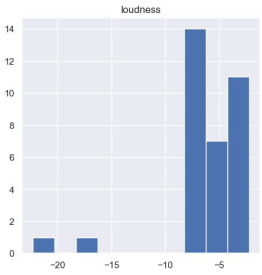

acousticness

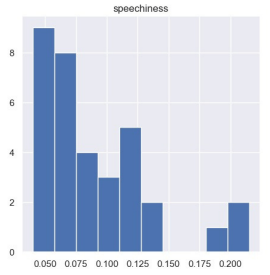

danceability

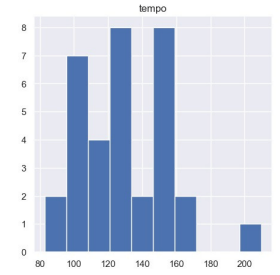

energy

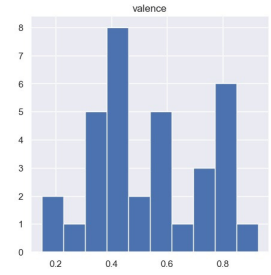

instrumentalness

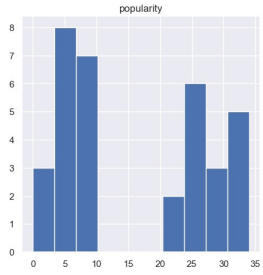

liveness

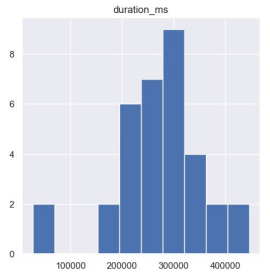

loudness


speechiness

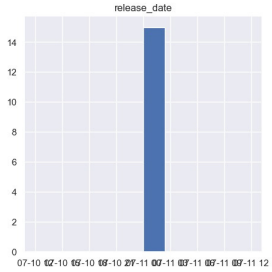

tempo

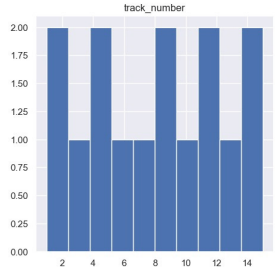

valence

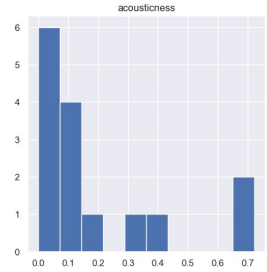

popularity

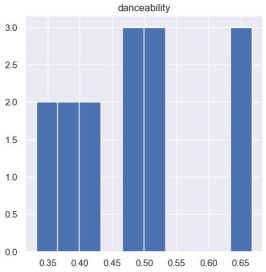

duration_ms

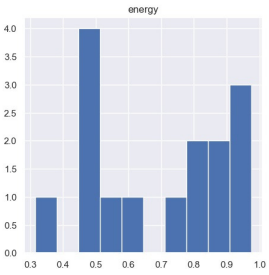
1994 Data

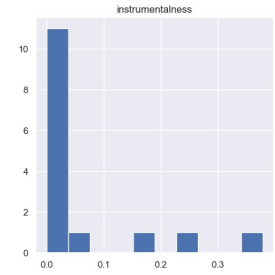

release_date

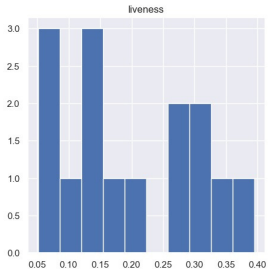

track_number

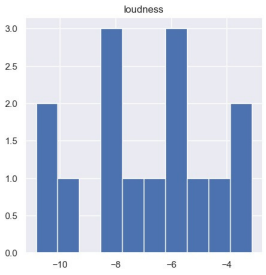

acousticness

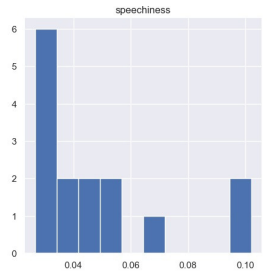

danceability

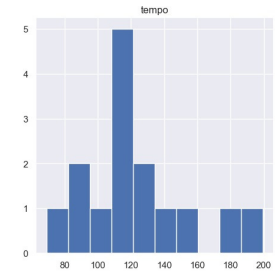

energy

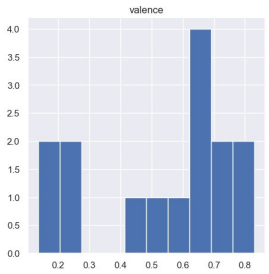

instrumentalness

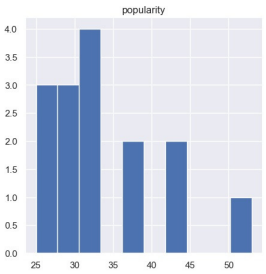

liveness

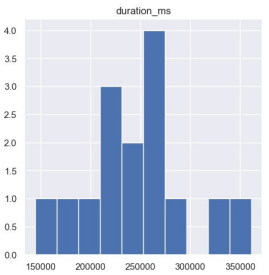

loudness


speechiness

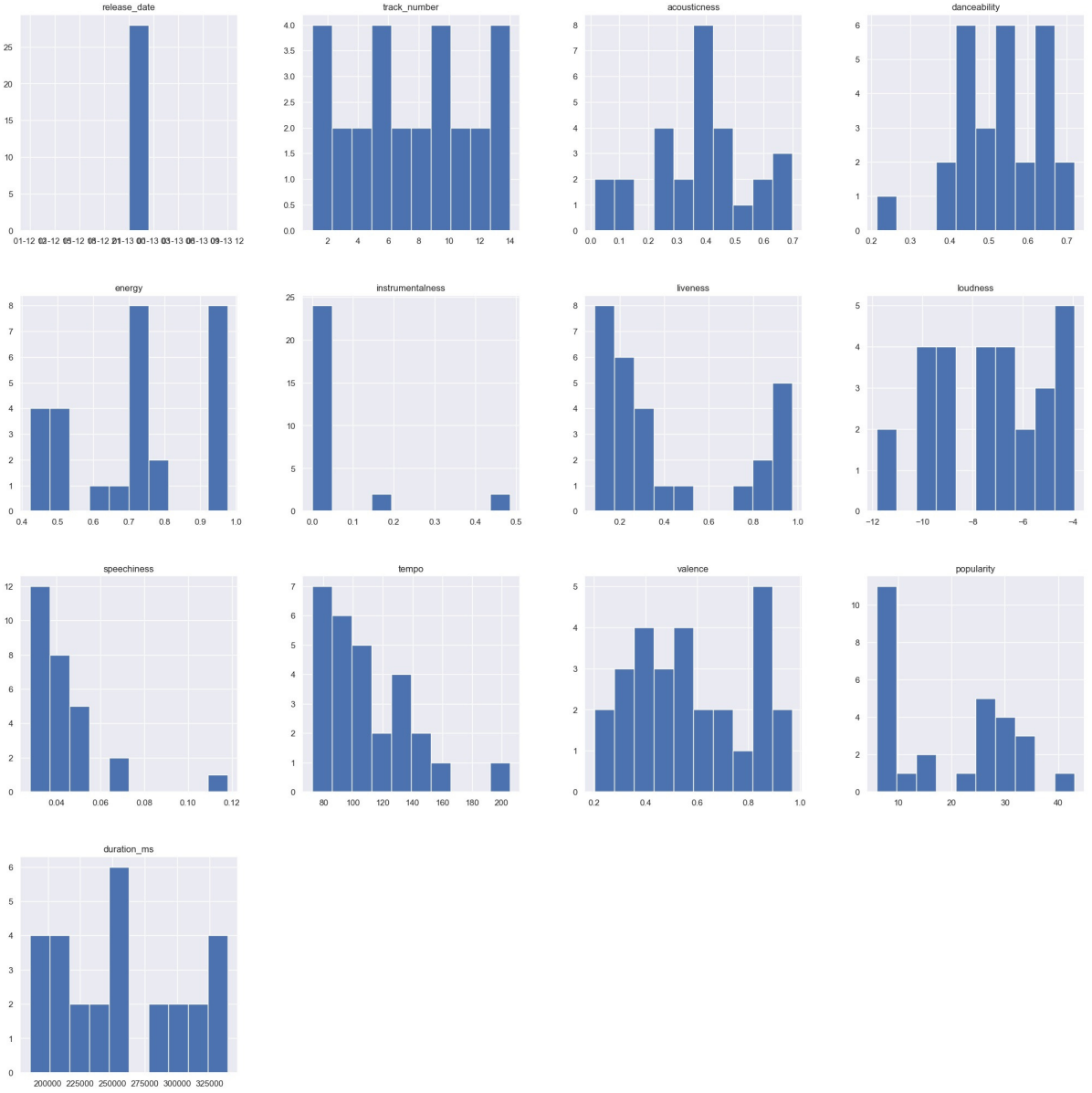

tempo


valence

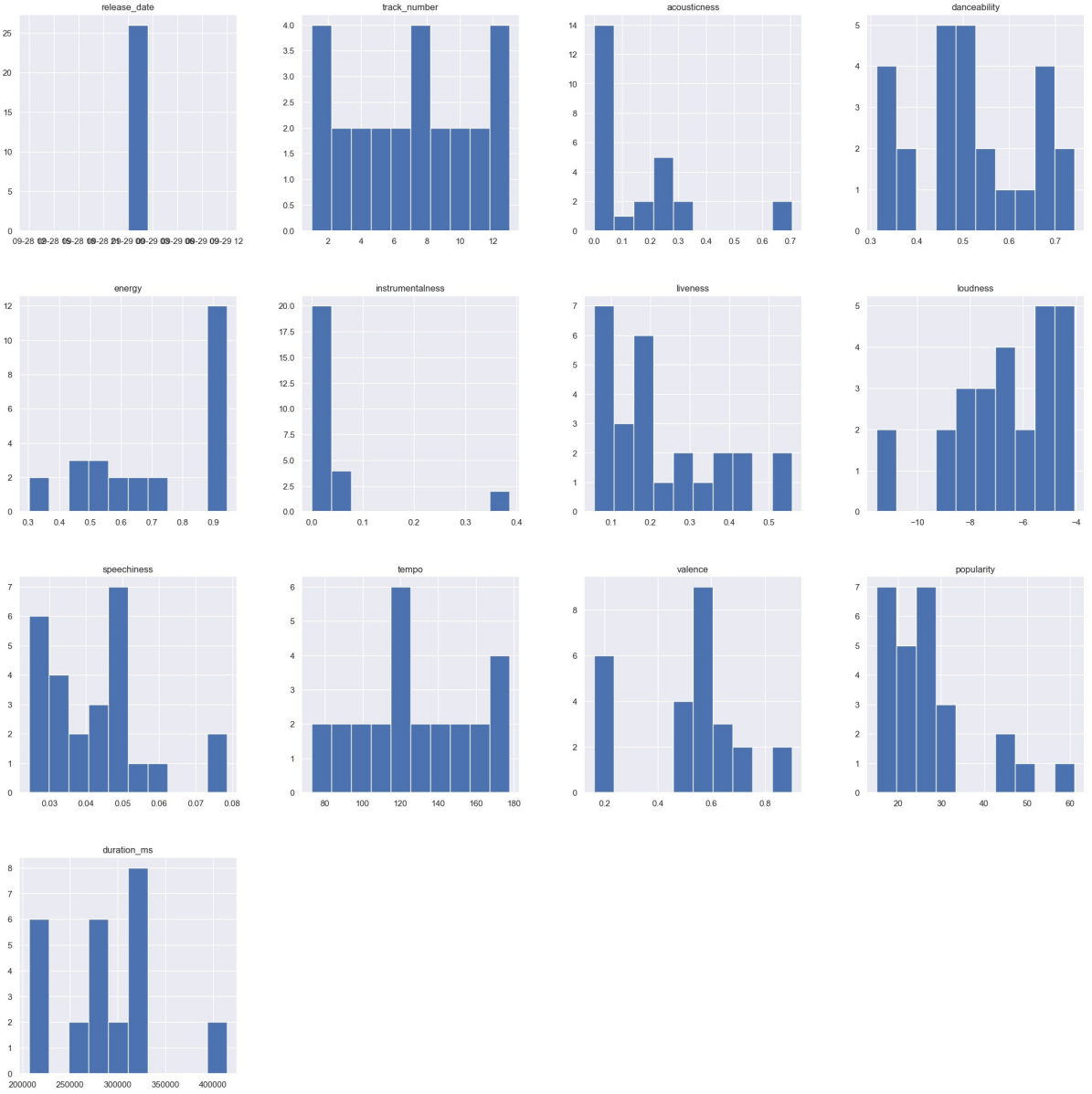

popularity


duration_ms

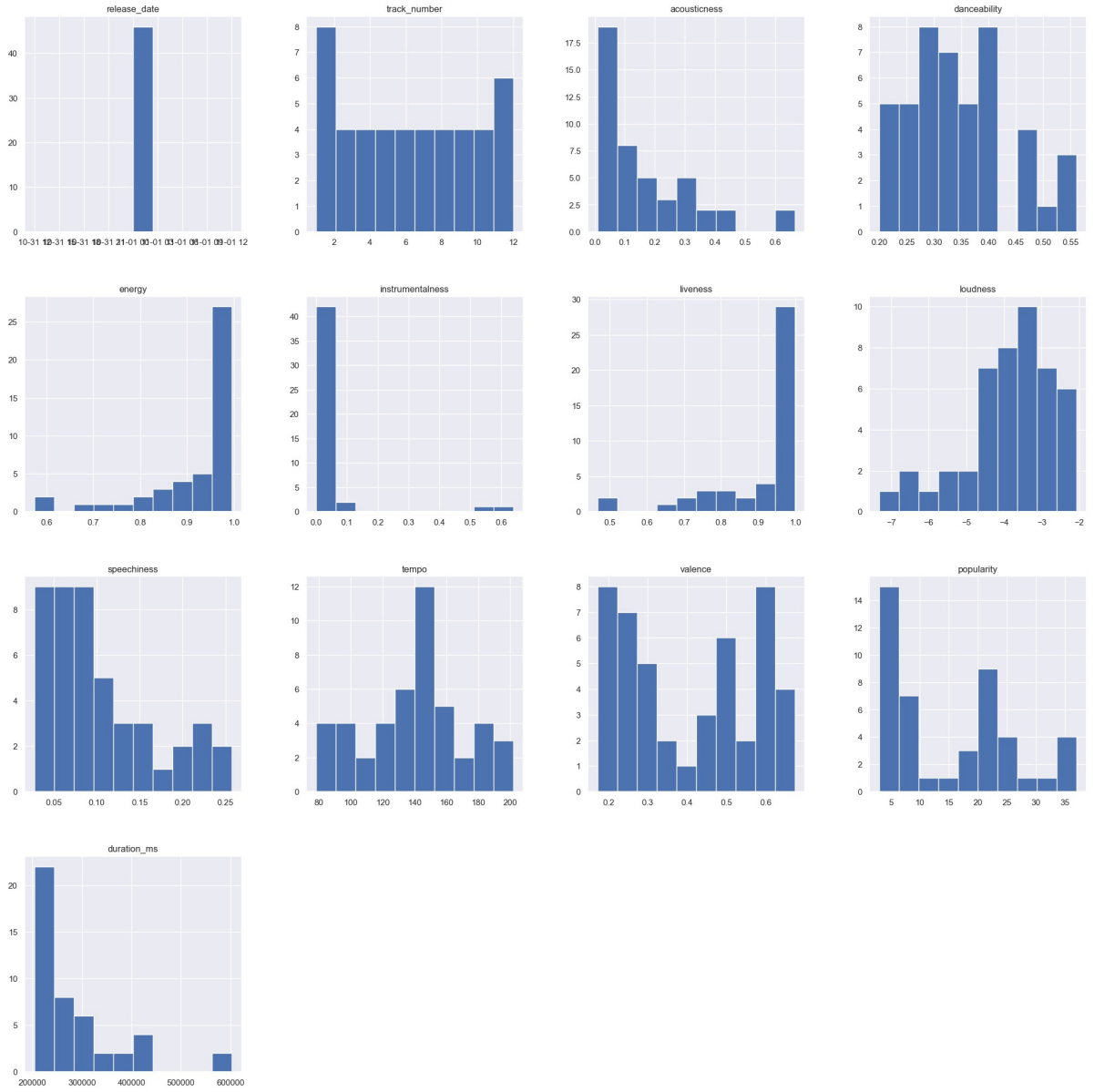
1995 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

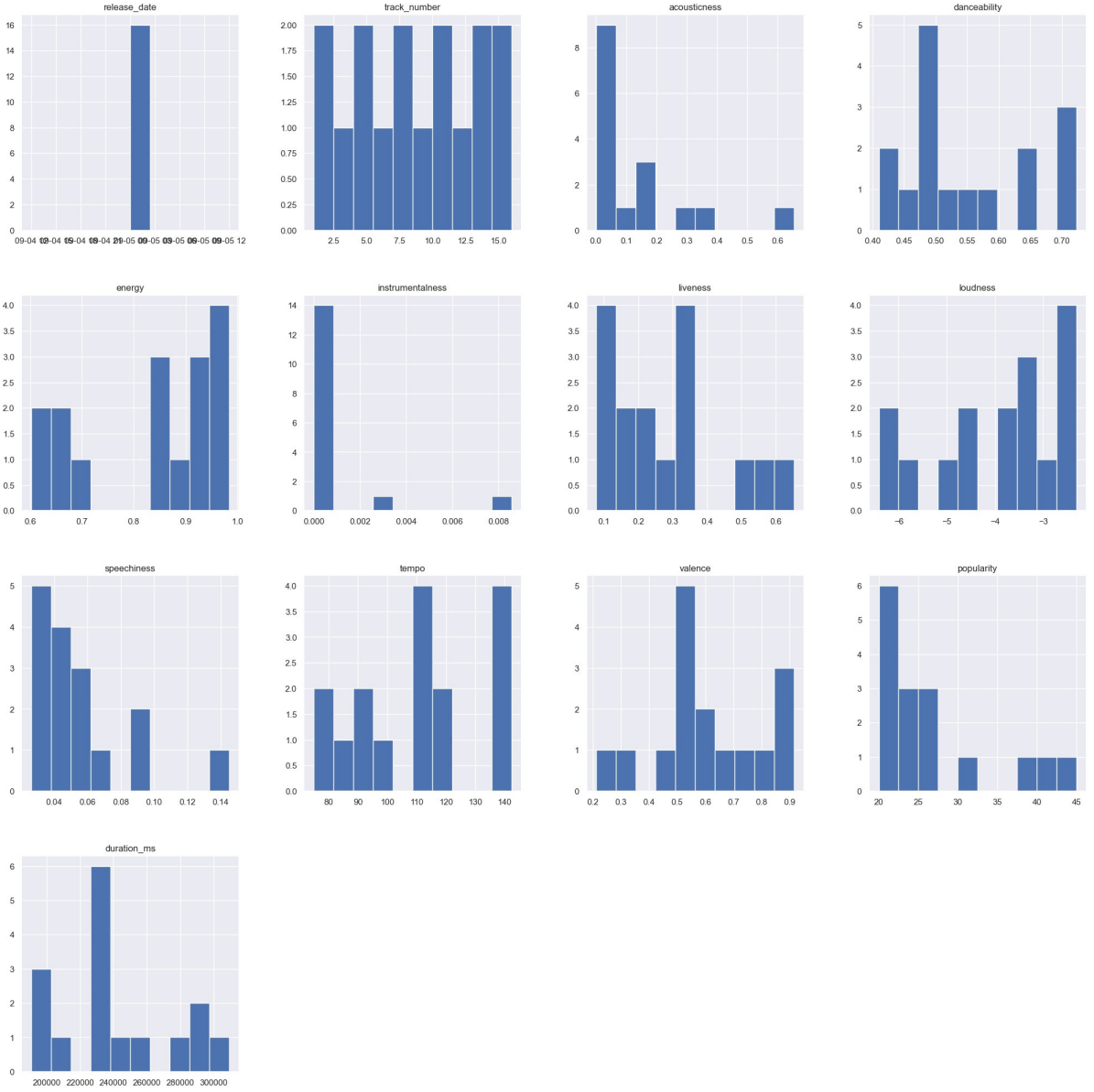
1997 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

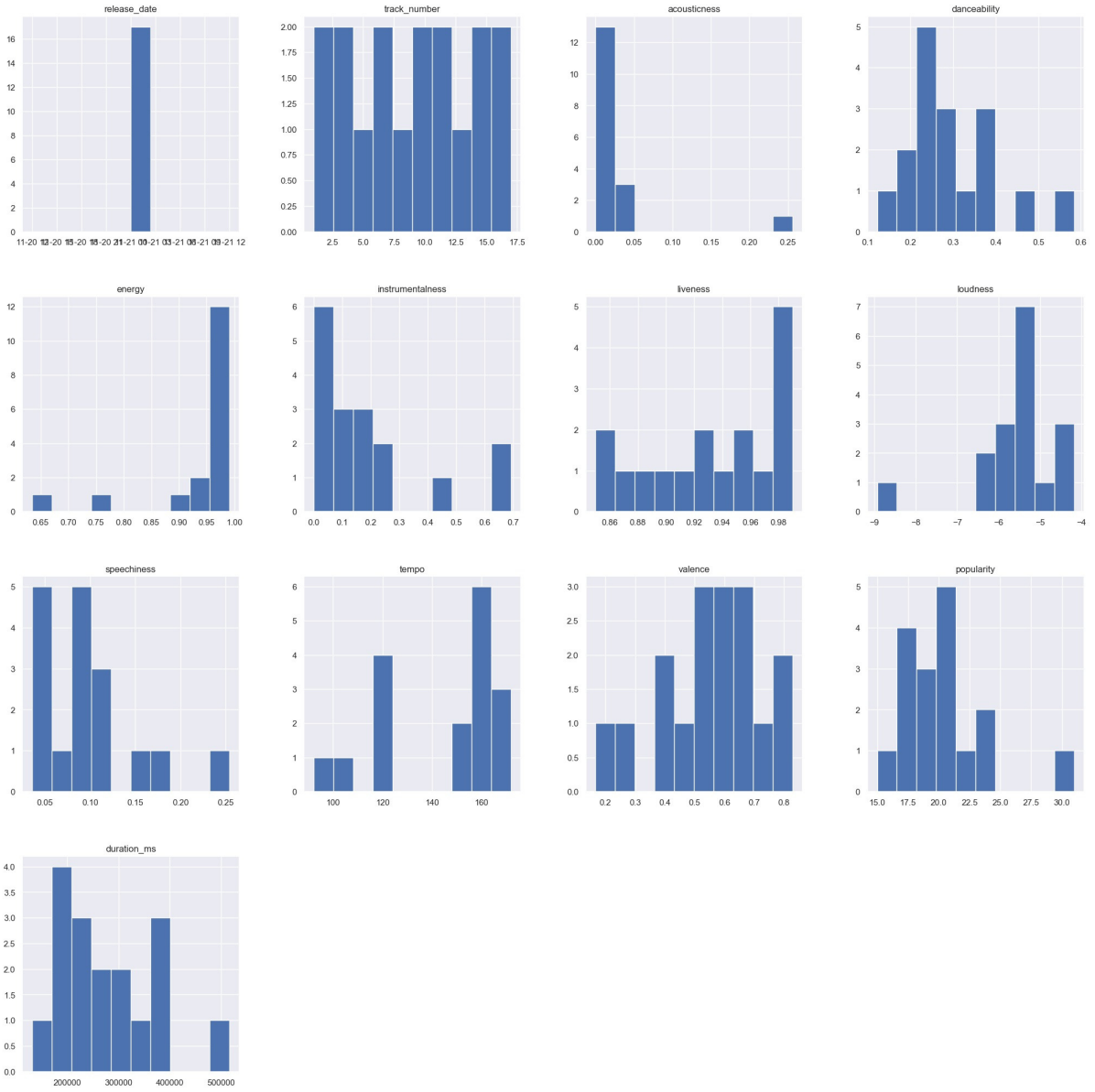
2004 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

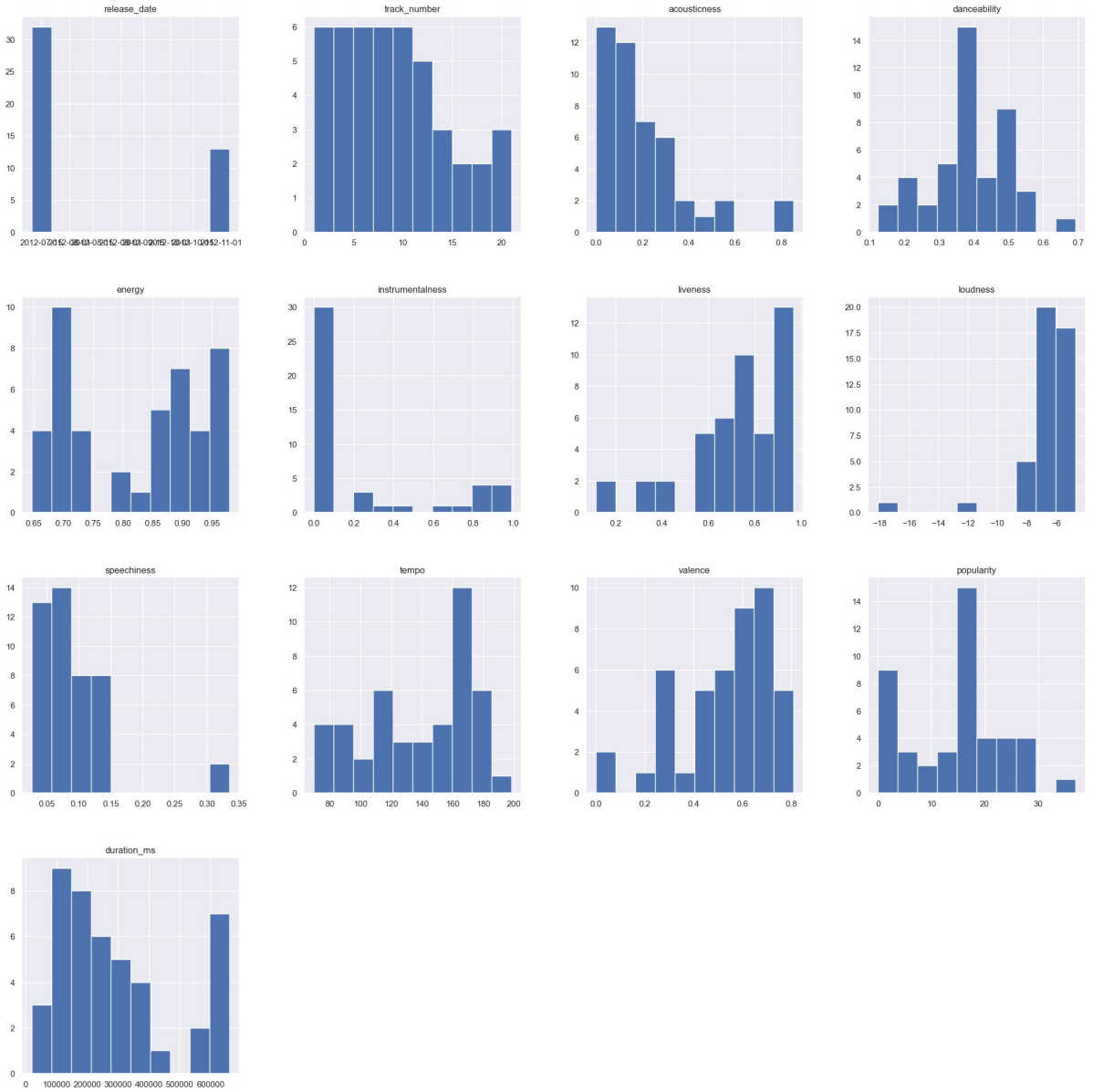
2005 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

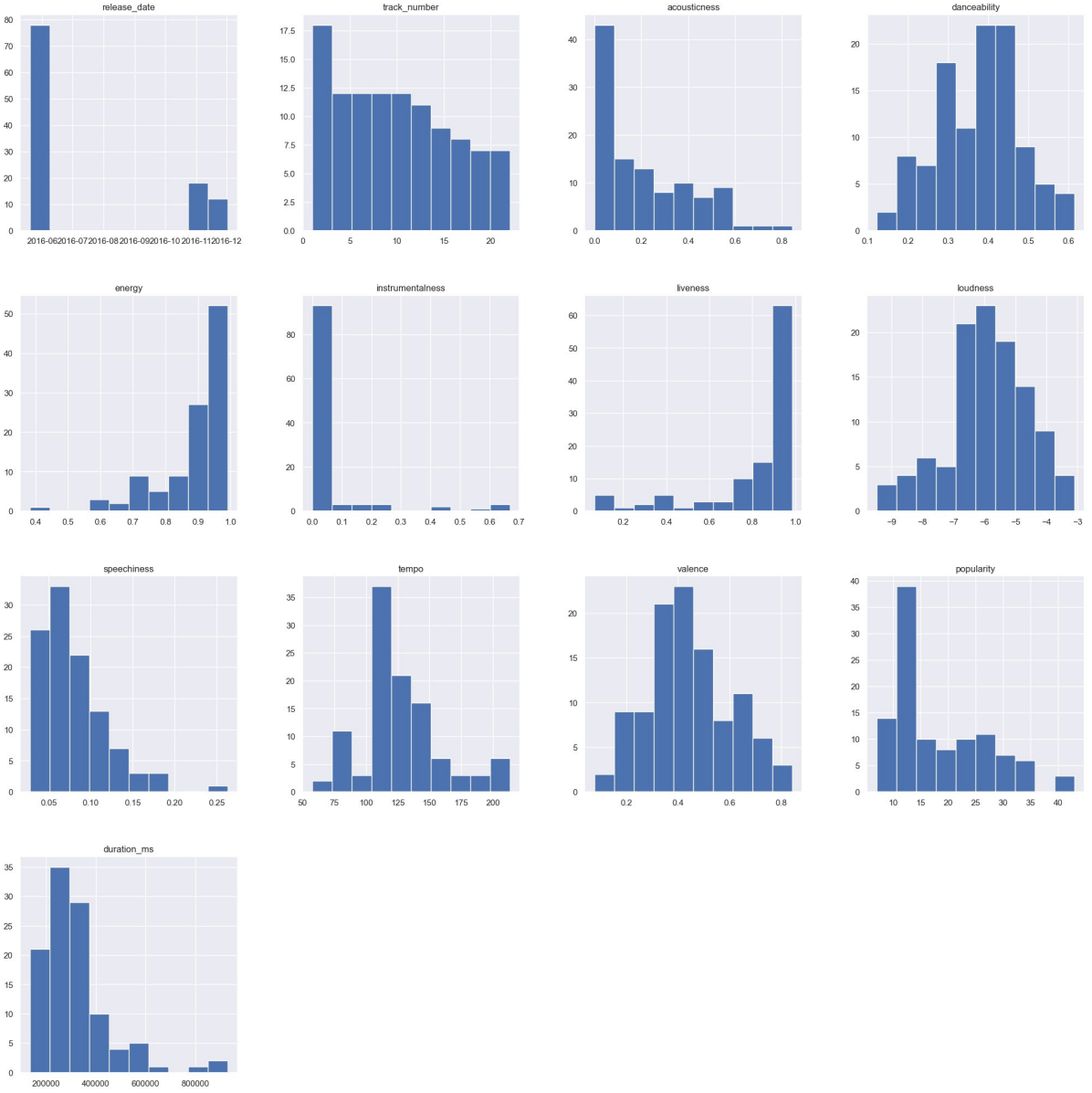
2011 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

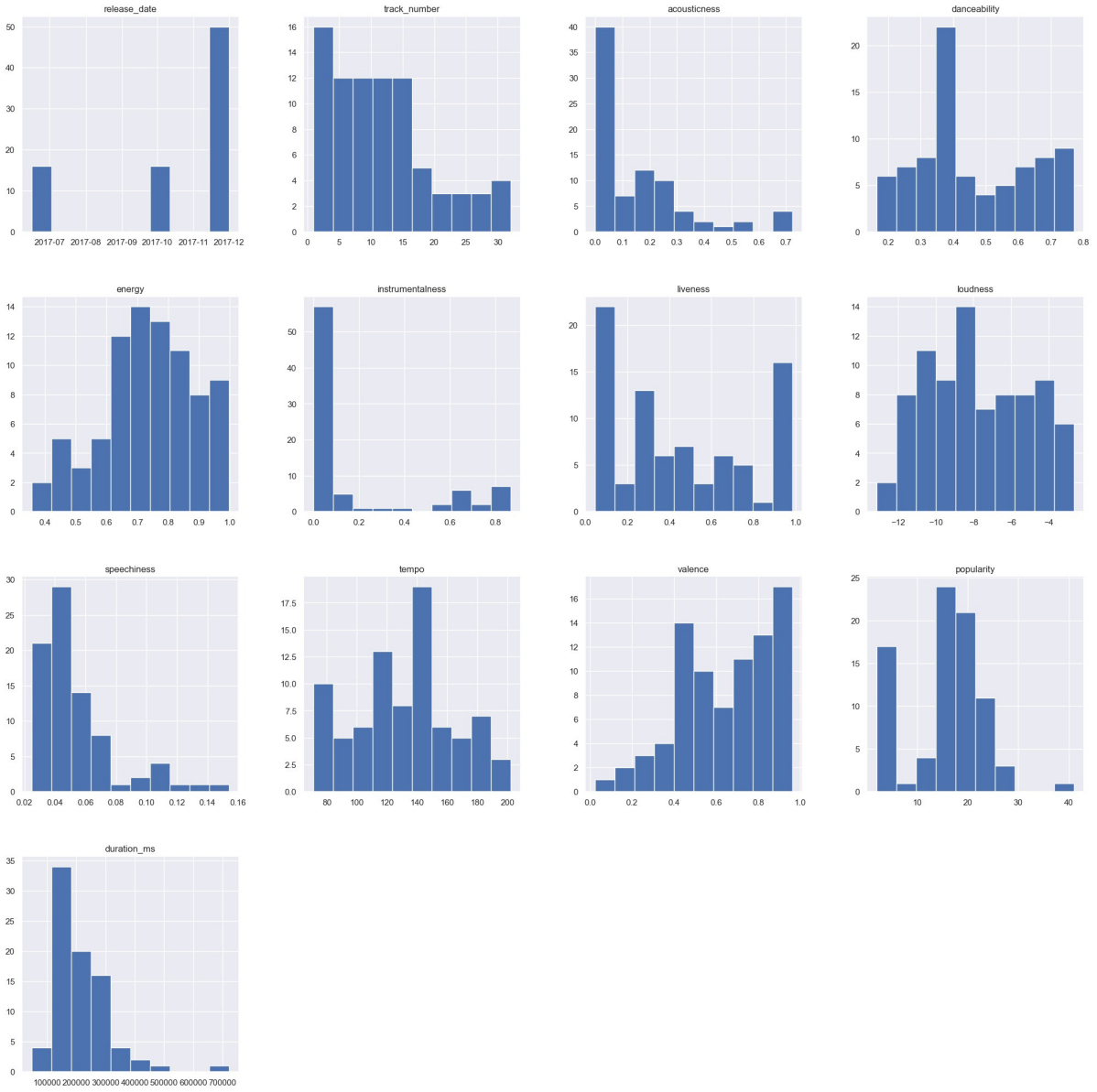
2012 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

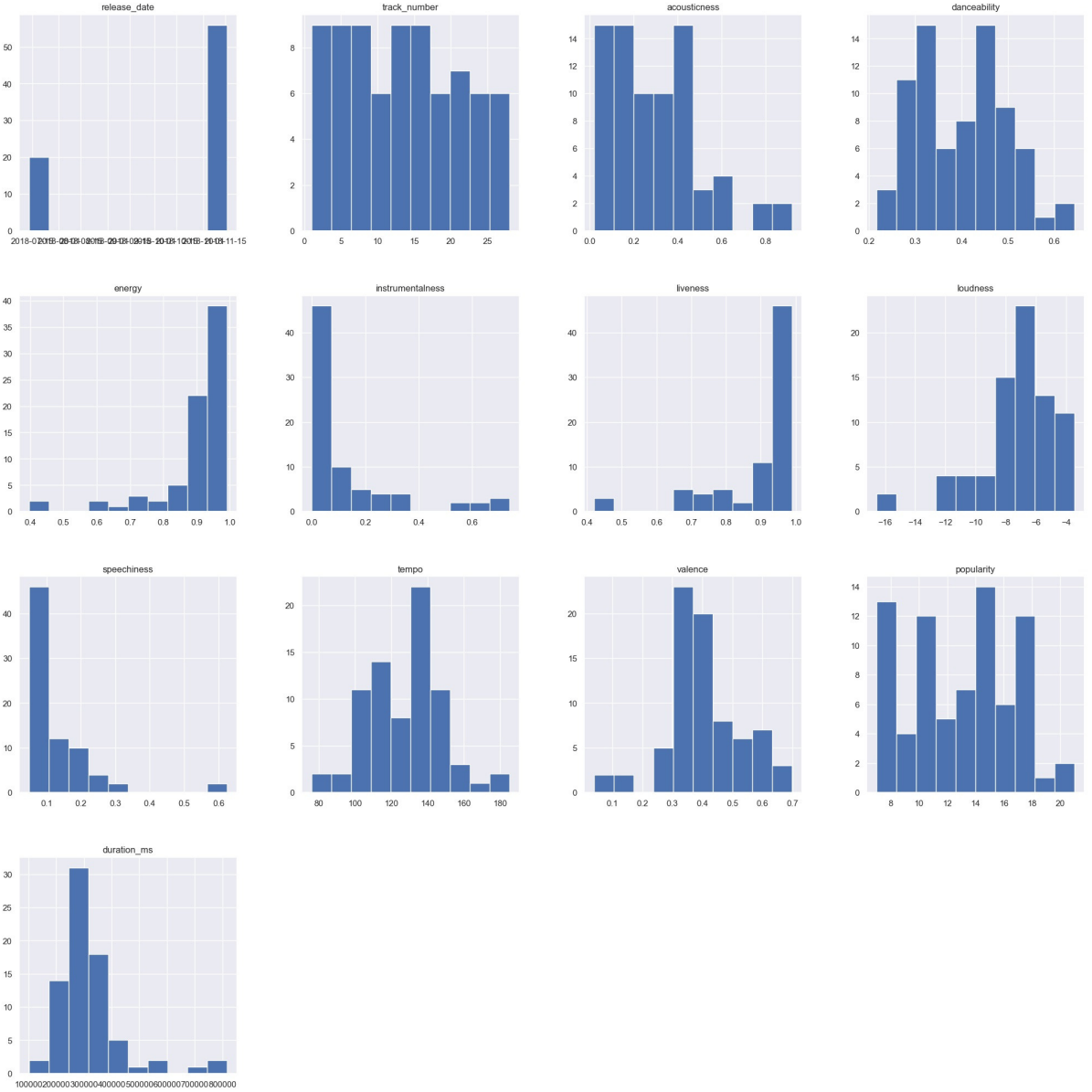
2016 Data


release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

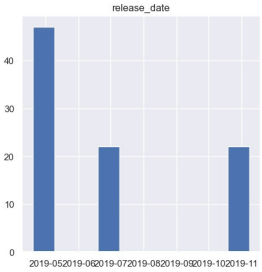
2017 Data

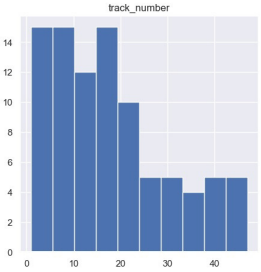

release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

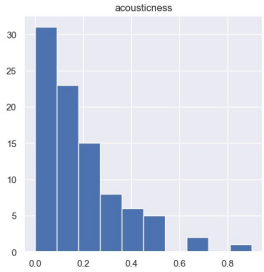
2018 Data

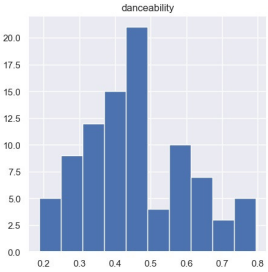

release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

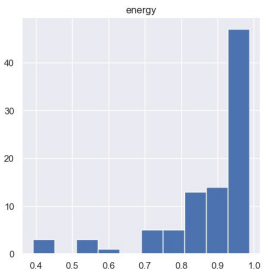
2019 Data

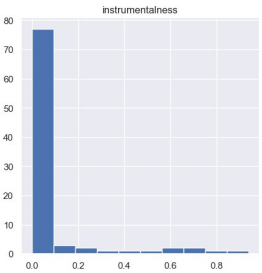

release_date
2019-05
2019-06
2019-07
2019-08
2019-09
2019-10
2019-11

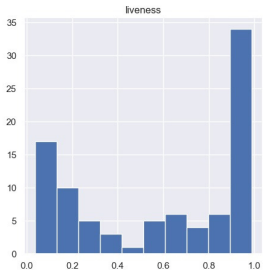

track_number

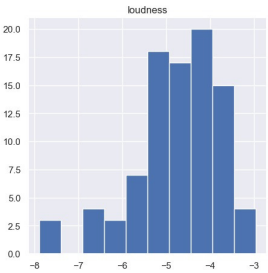

acousticness

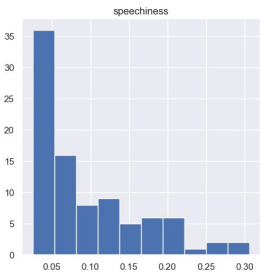

danceability

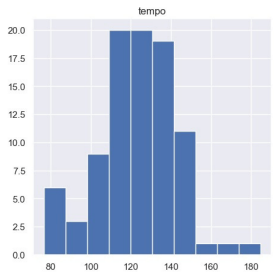

energy

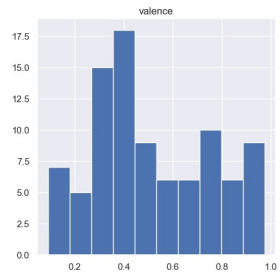

instrumentalness

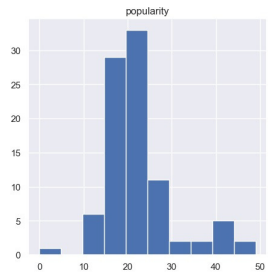

liveness

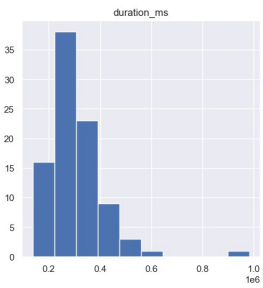

loudness


speechiness

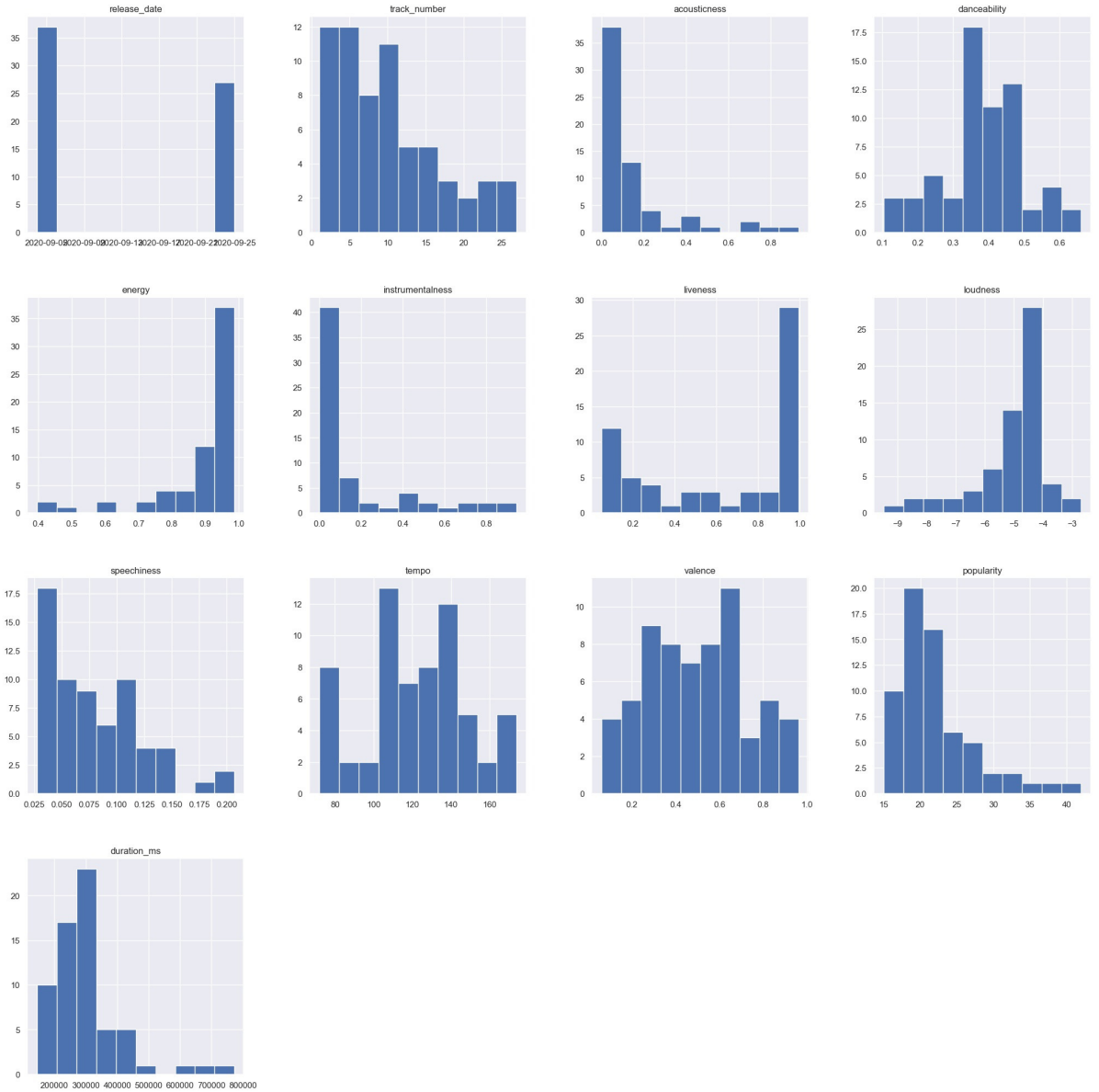

tempo


valence

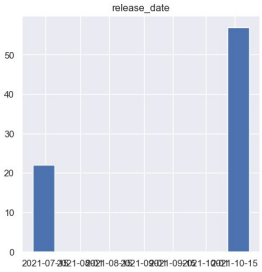

popularity

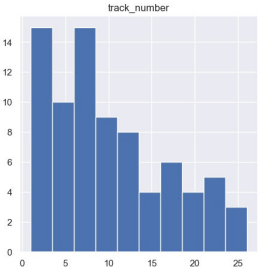

duration_ms
1e6

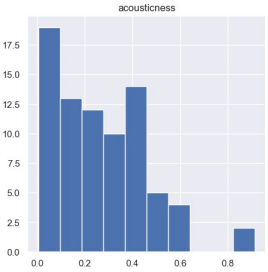
2020 Data

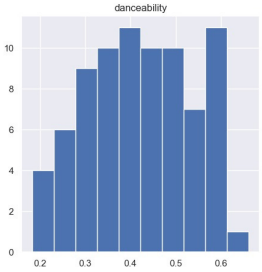

release_date
track_number
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
popularity
duration_ms

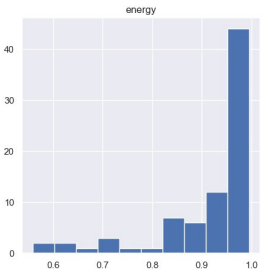
2021 Data

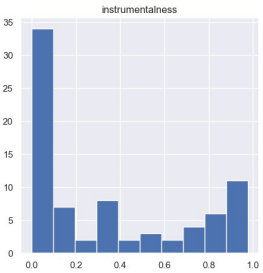

release_date

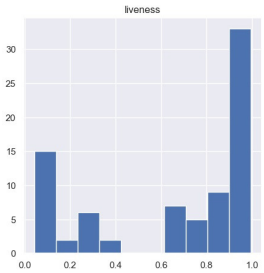

track_number

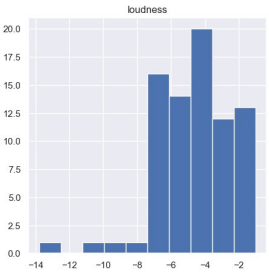

acousticness

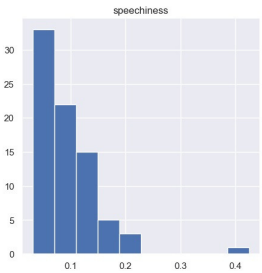

danceability

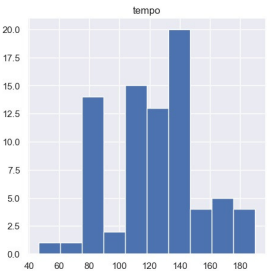

energy

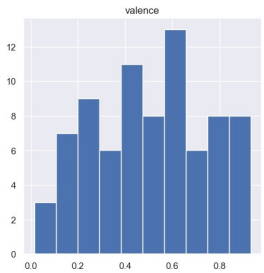

instrumentalness

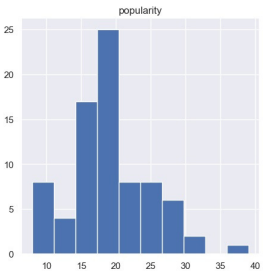

liveness

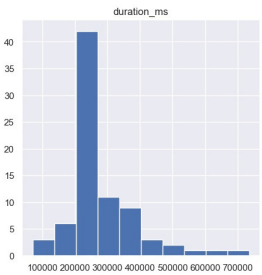

loudness


speechiness

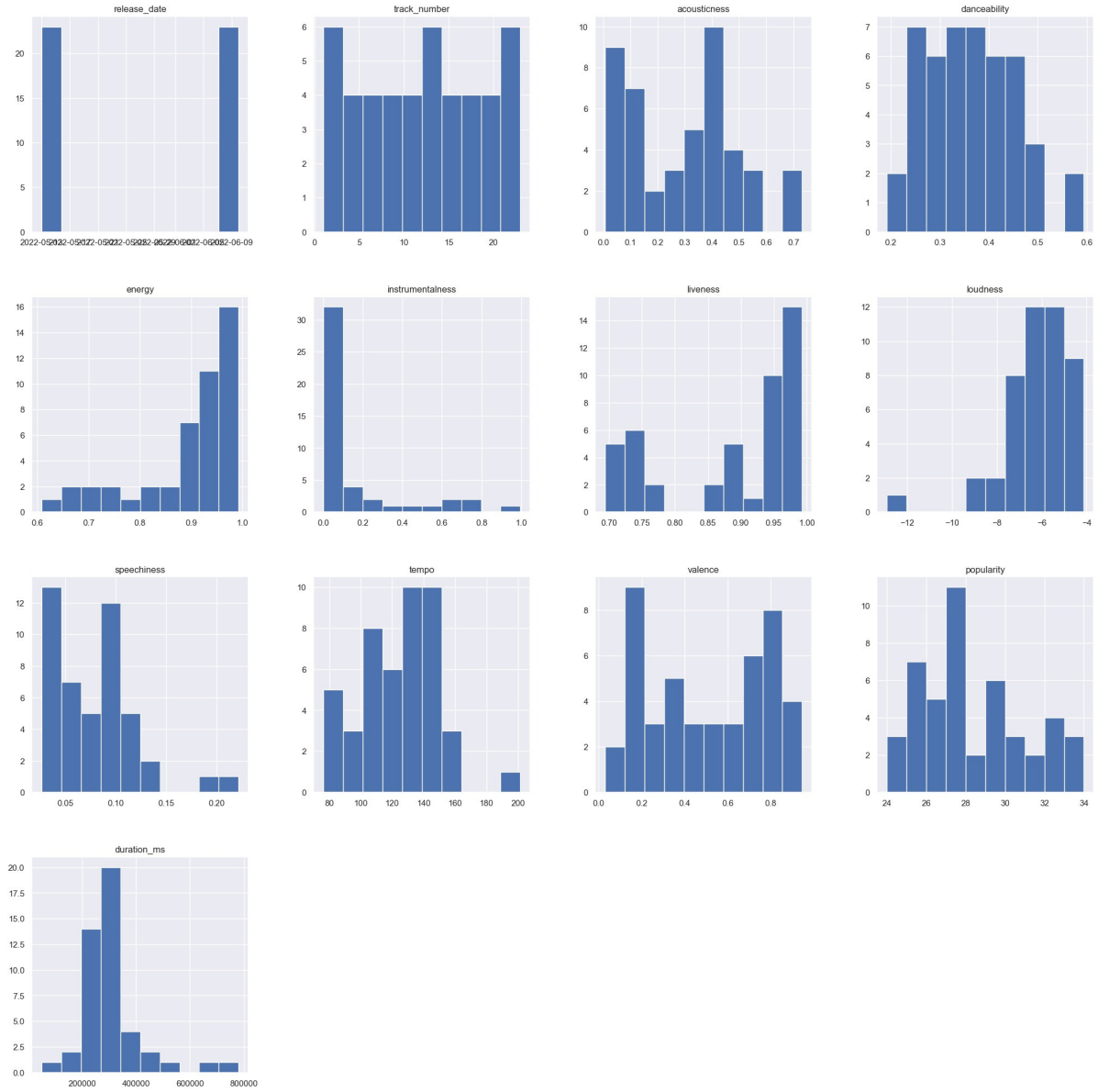

tempo


valence


popularity


duration_ms

2022 Data

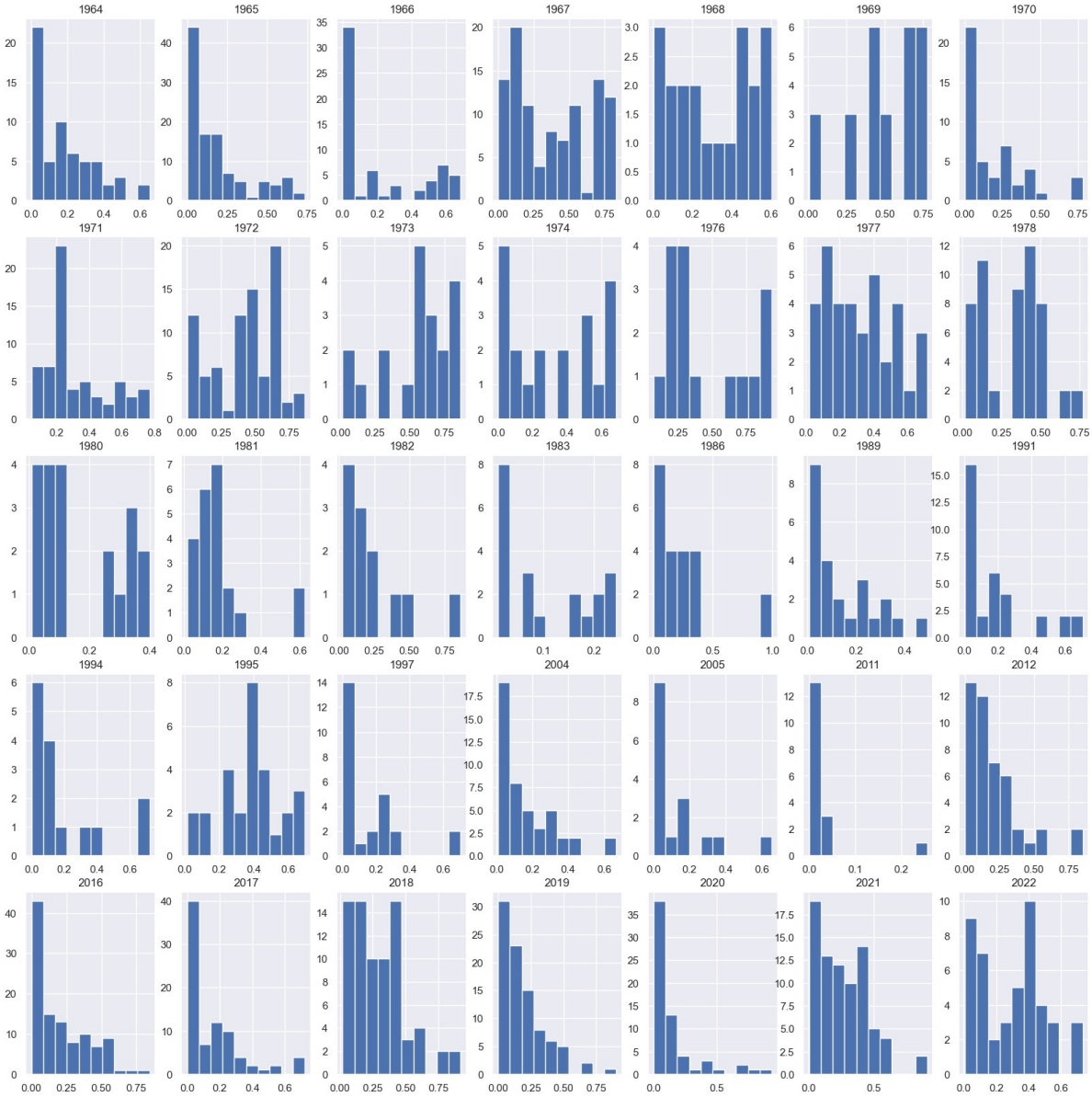


In the below I plot the numerical features for each year. From this we can gain insight how this change over time

Acousticness Data for all the year.

In [177…
```
df=song_dat.copy()
```

In [178…
```
plt.figure(figsize=(20,20))
for i in enumerate(years):
 plt.subplot(5,7,i[0]+1)
 plt.title(i[1])
 df_=df.loc[df['release_date'].dt.year==i[1]]['acousticness']
 df_.hist()
save_plots('Acousticness all years')
```



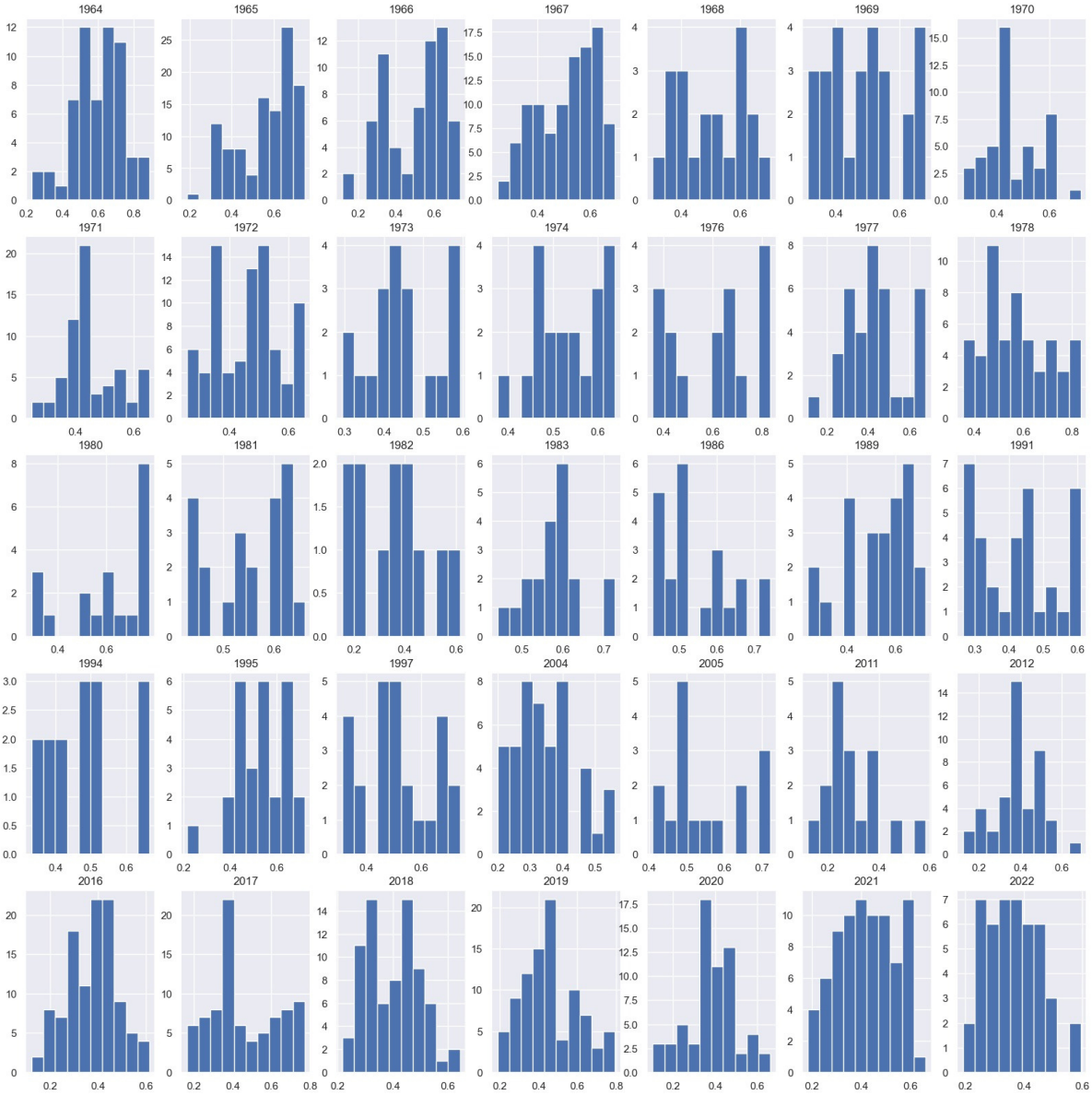
Finding:

- We see that acousticness is change over the year it become higher years after years

Danceability Data for all the year

In [179…

```
plt.figure(figsize=(20,20))
for i in enumerate(years):
 plt.subplot(5,7,i[0]+1)
 plt.title(i[1])
 df_=df.loc[df['release_date'].dt.year==i[1]]['danceability']
 df_.hist()
save_plots('Danceability for all the year')
plt.show()
```



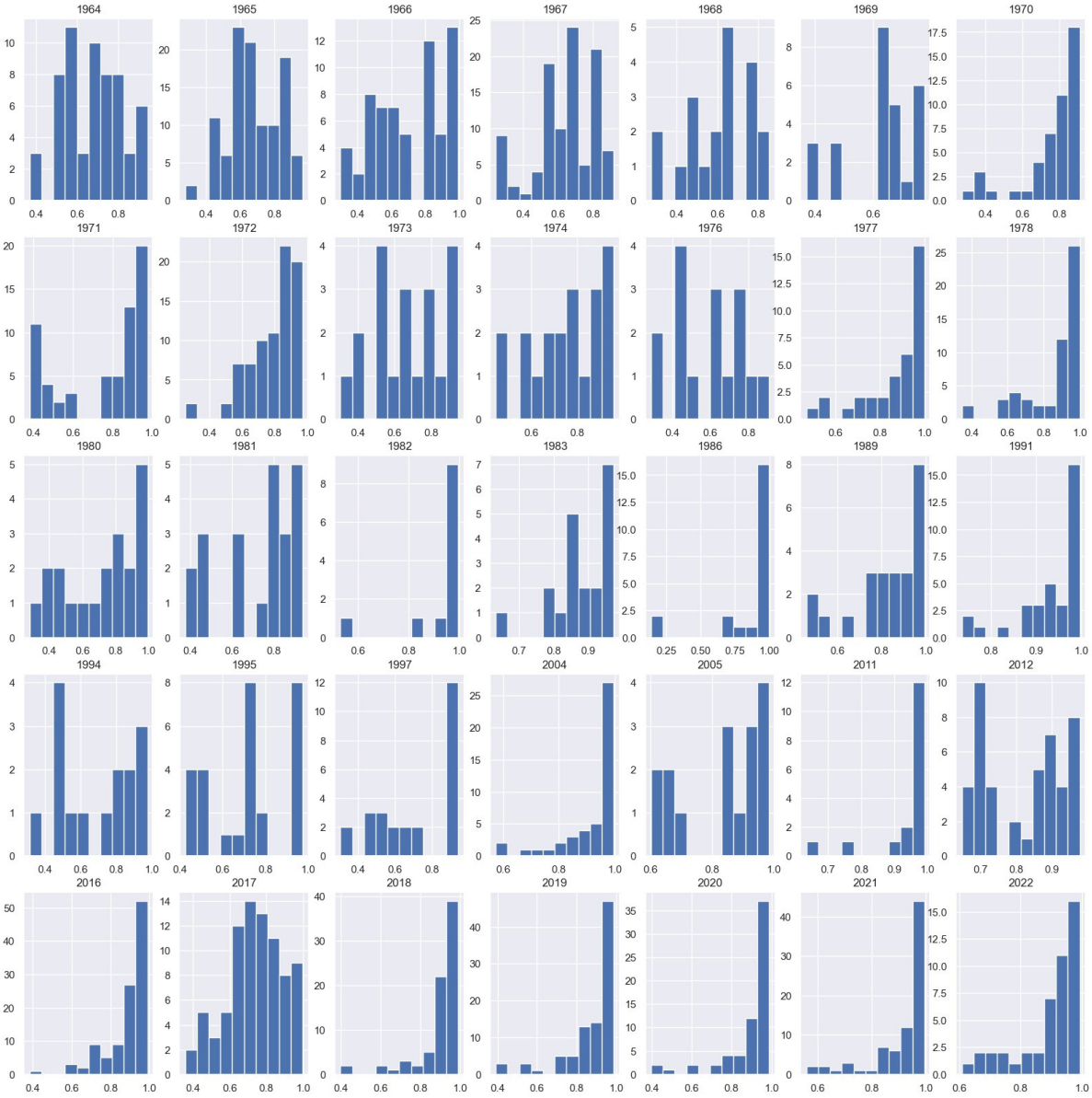
Finding:

- We see that danceability does not much change over time, But some years have the danceability lesser and some are higher like 2022 have most of the song are danceable and years like 1971 have less danceable.

Energy Data for all the year

In [180…

```
plt.figure(figsize=(20,20))
for i in enumerate(years):
 plt.subplot(5,7,i[0]+1)
 plt.title(i[1])
 df_=df.loc[df['release_date'].dt.year==i[1]]['energy']
 plt.savefig('Energy')
 df_.hist()
save_plots('Enerty data for all the years')
plt.show()
```



Finding:

- In some years enery less very less and some years have high energy.

Instrumentalness Data for all the yea..

In [181…

```
plt.figure(figsize=(20,20))
for i in enumerate(years):
 plt.subplot(5,7,i[0]+1)
 plt.title(i[1])
 df_=df.loc[df['release_date'].dt.year==i[1]]['instrumentalness']
 df_.hist()
save_plots('Instrumentalness data for all the years')
plt.show()
```

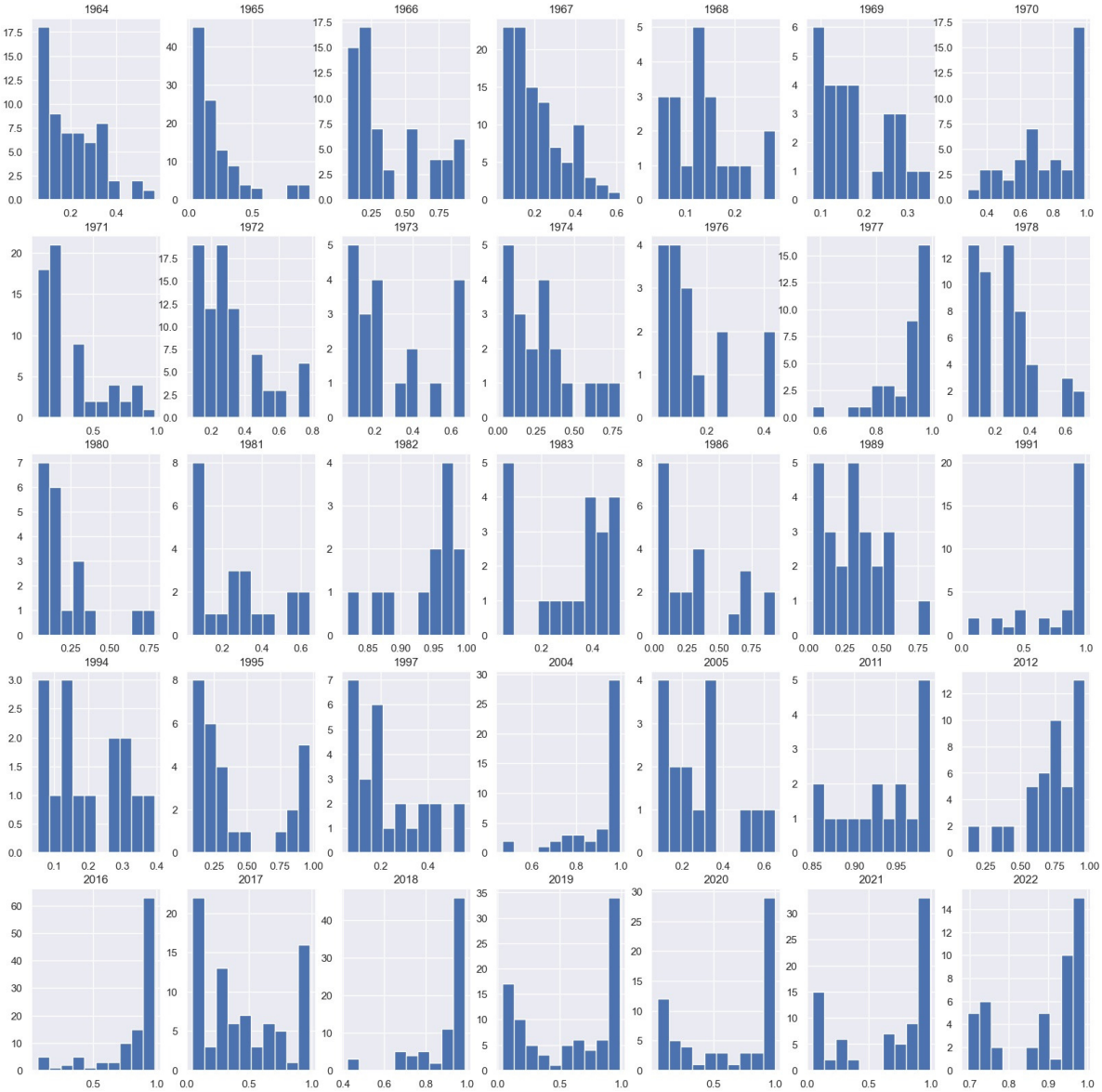
Finding:

- There are some years which instrumentalness are very less

Liveness Data for all the year

In [182…

```
plt.figure(figsize=(20,20))
for i in enumerate(years):
 plt.subplot(5,7,i[0]+1)
 plt.title(i[1])
 df_=df.loc[df['release_date'].dt.year==i[1]]['liveness']
 df_.hist()
save_plots('Liveness data for all the year')
plt.show()
```



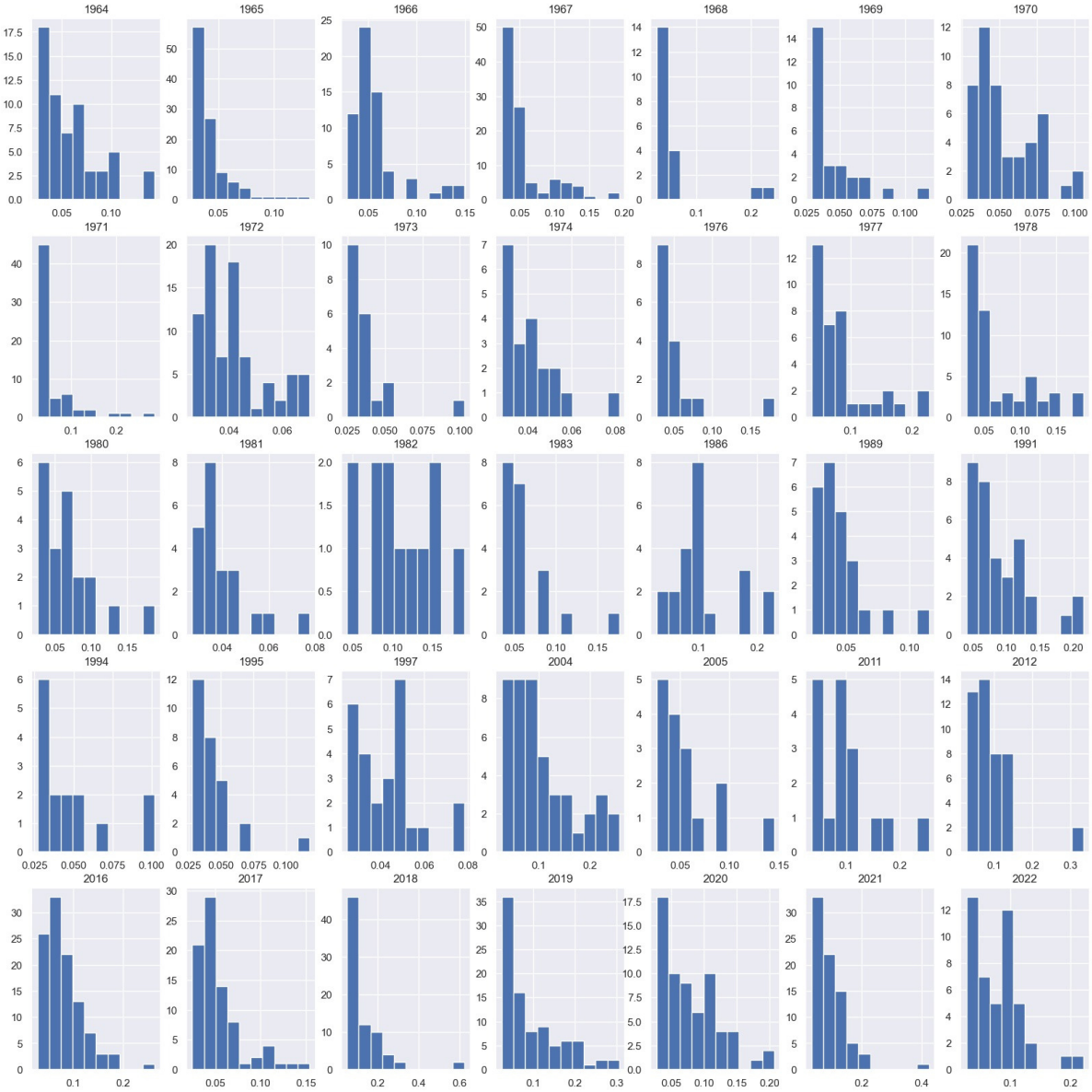
Finding:

- We can see some pattern like liveness in increase over years

Speechiness Data for all the year.

In [183…

```
plt.figure(figsize=(20,20))
for i in enumerate(years):
 plt.subplot(5,7,i[0]+1)
 plt.title(i[1])
 df_=df.loc[df['release_date'].dt.year==i[1]]['speechiness']
 df_.hist()
save_plots('Speechiness data for all the year')
plt.show()
```



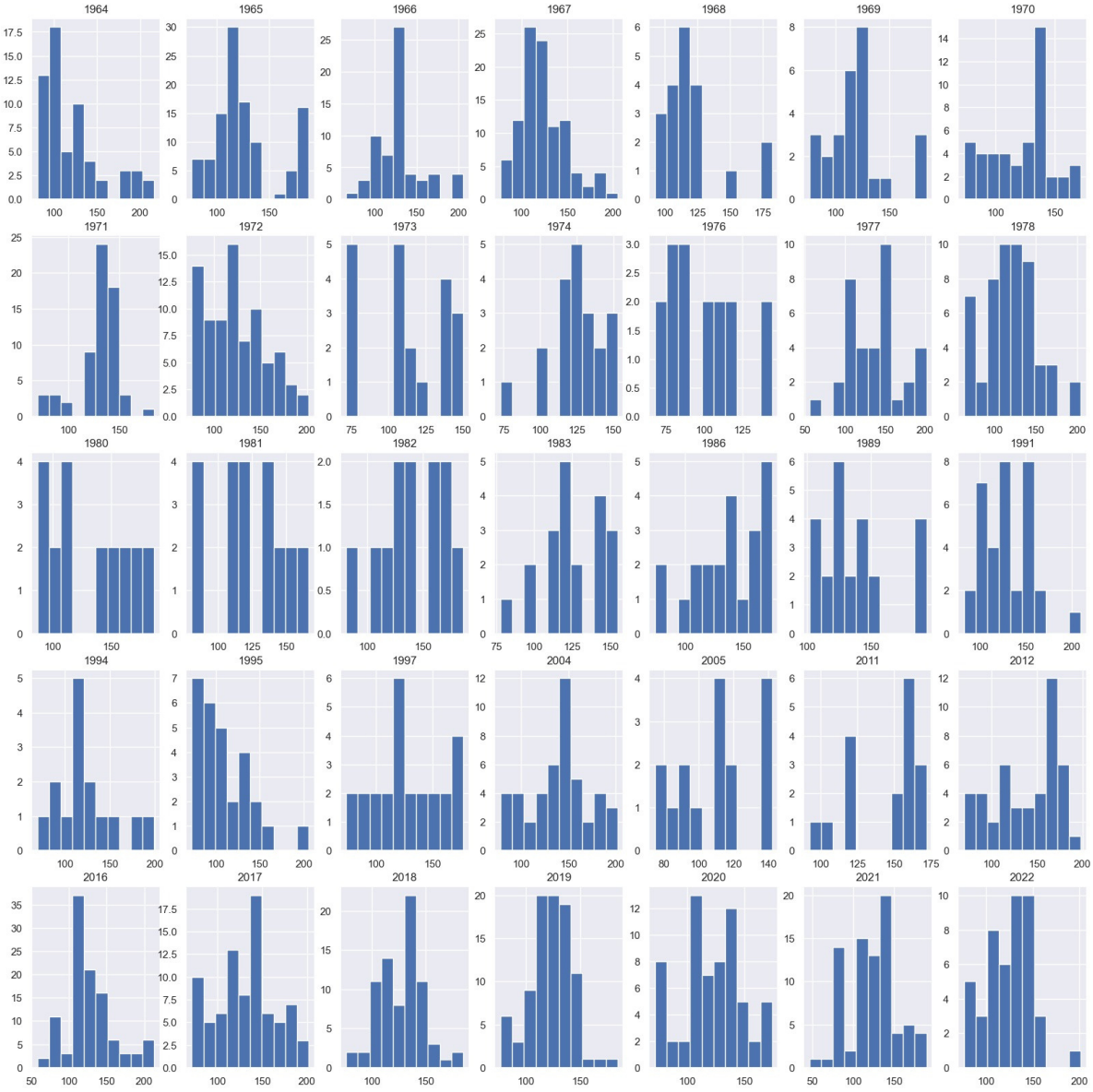
Finding:

- Speechness not much change over the time it is like same for most the years

Tempo Data for all the year

In [184…

```
plt.figure(figsize=(20,20))
for i in enumerate(years):
 plt.subplot(5,7,i[0]+1)
 plt.title(i[1])
 df_=df.loc[df['release_date'].dt.year==i[1]]['tempo']
 df_.hist()
save_plots('Tempo data for all the years')
plt.show()
```



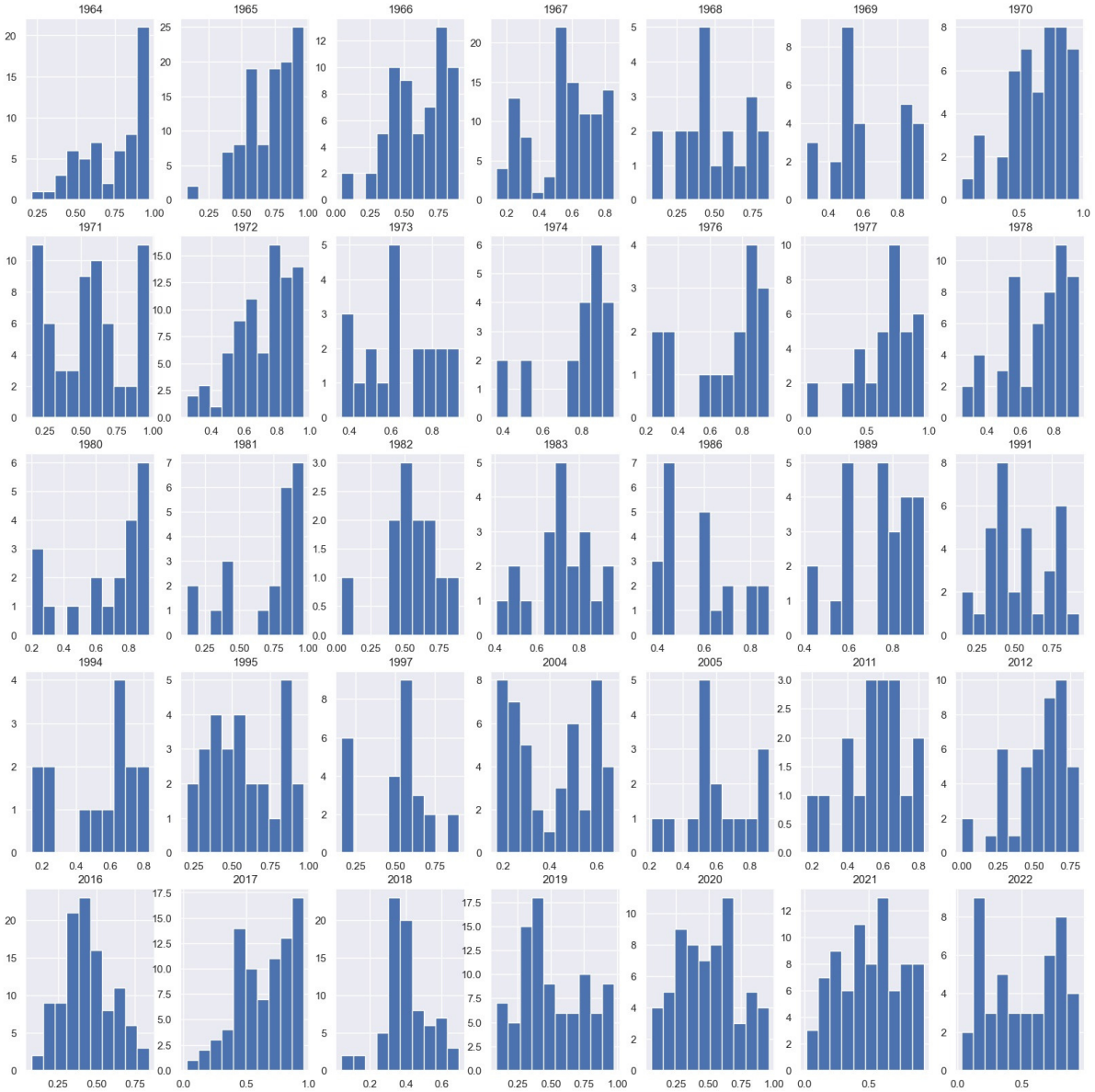
Finding:

- Not much change in tempo for the years, but some years have less tempo and some have more and balence

Valence Data for all the year

In [185…

```
plt.figure(figsize=(20,20))
for i in enumerate(years):
 plt.subplot(5,7,i[0]+1)
 plt.title(i[1])
 df_=df.loc[df['release_date'].dt.year==i[1]]['valence']
 df_.hist()
save_plots('Valence data for all the years')
plt.show()
```



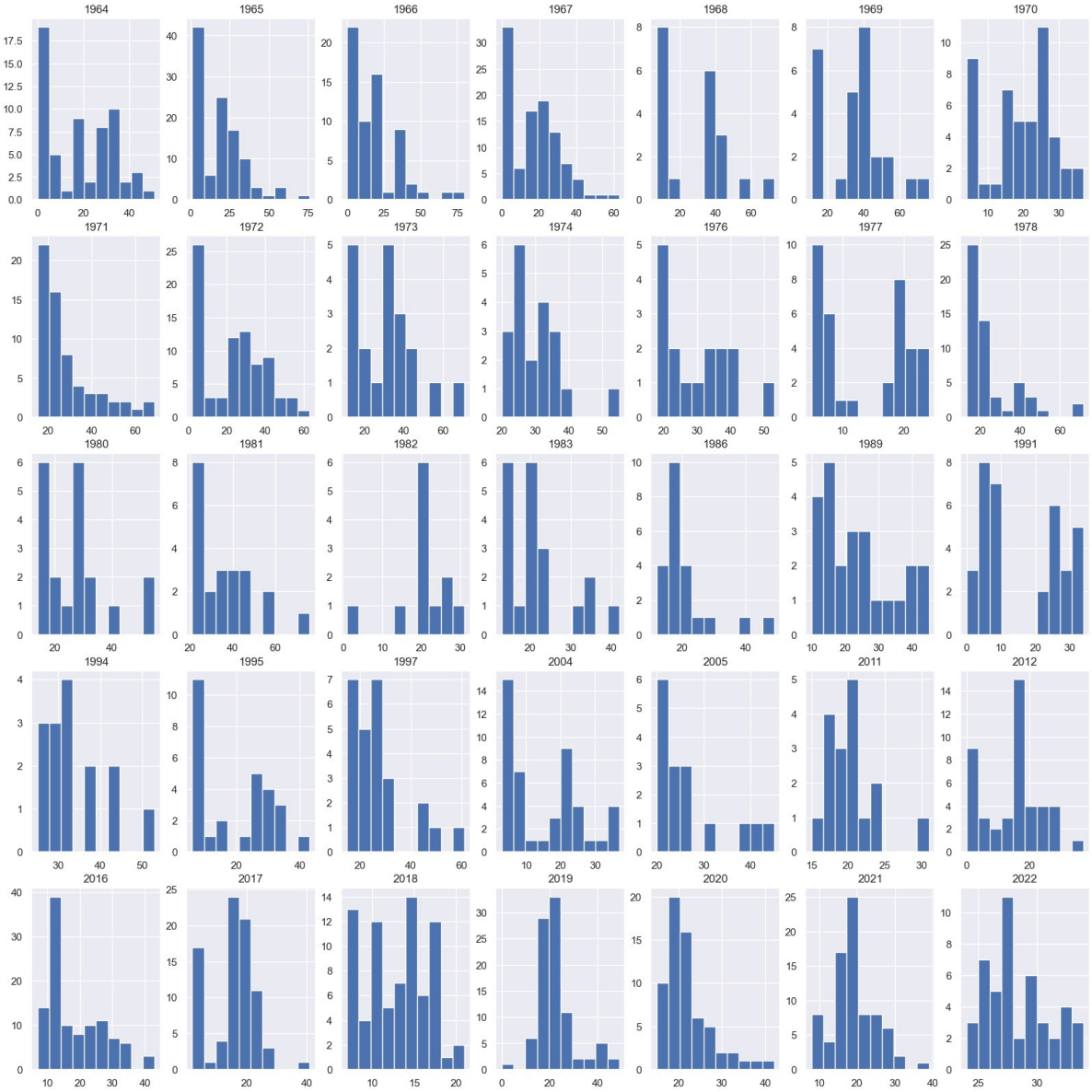
Finding:

- Valence is somewhat like increasing over time

Popularity Data for all the year

In [186…

```
plt.figure(figsize=(20,20))
for i in enumerate(years):
 plt.subplot(5,7,i[0]+1)
 plt.title(i[1])
 df_=df.loc[df['release_date'].dt.year==i[1]]['popularity']
 df_.hist()
save_plots('Popularity Data for all the years')
plt.show()
```



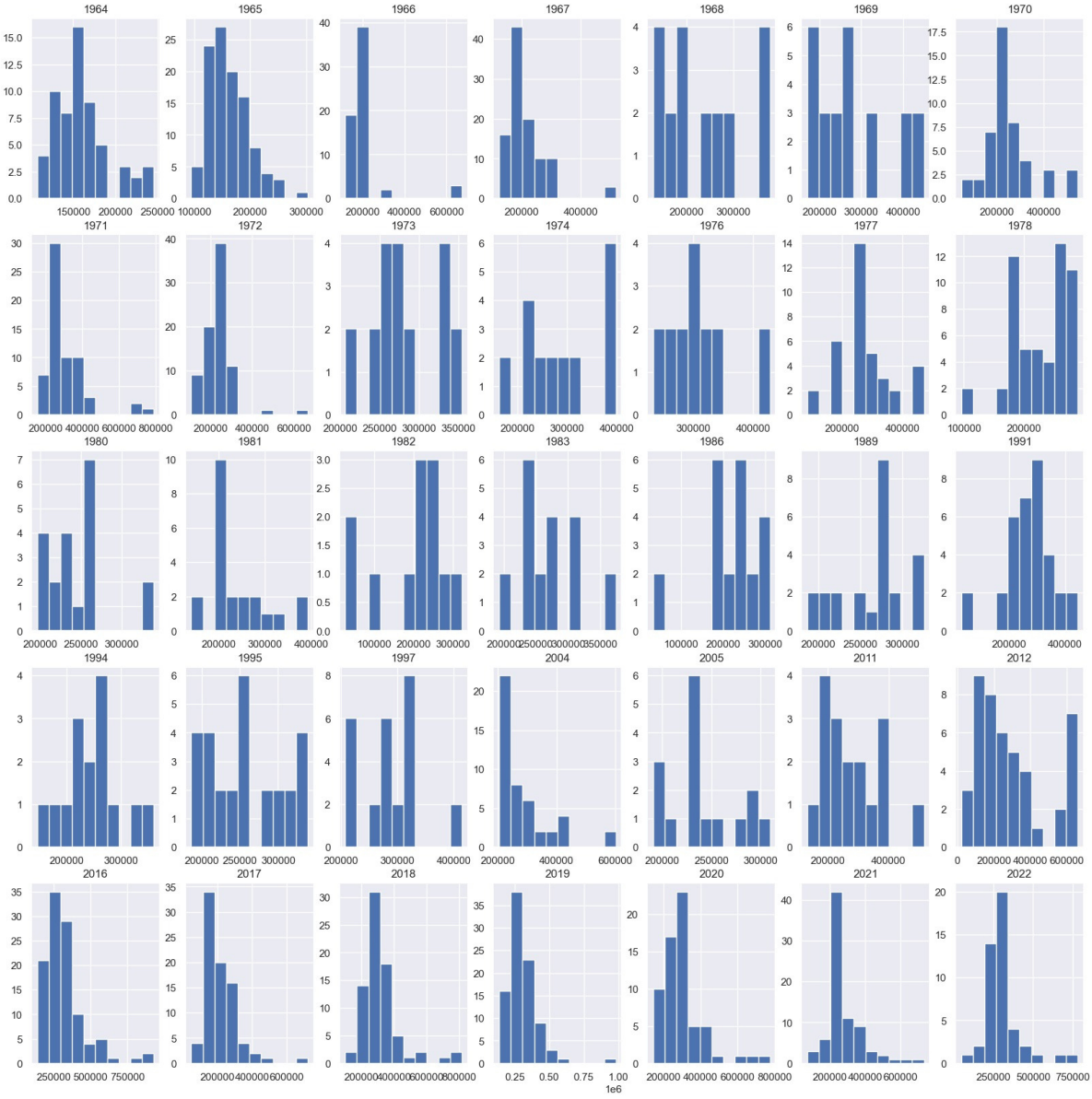
Finding:

- Some years have less number of popular song and some have high number of popular song.

Duration Data for all the year.

In [187…

```
plt.figure(figsize=(20,20))
for i in enumerate(years):
 plt.subplot(5,7,i[0]+1)
 plt.title(i[1])
 df_=df.loc[df['release_date'].dt.year==i[1]]['duration_ms']
 df_.hist()
save_plots('Duration data for all the years')
plt.show()
```



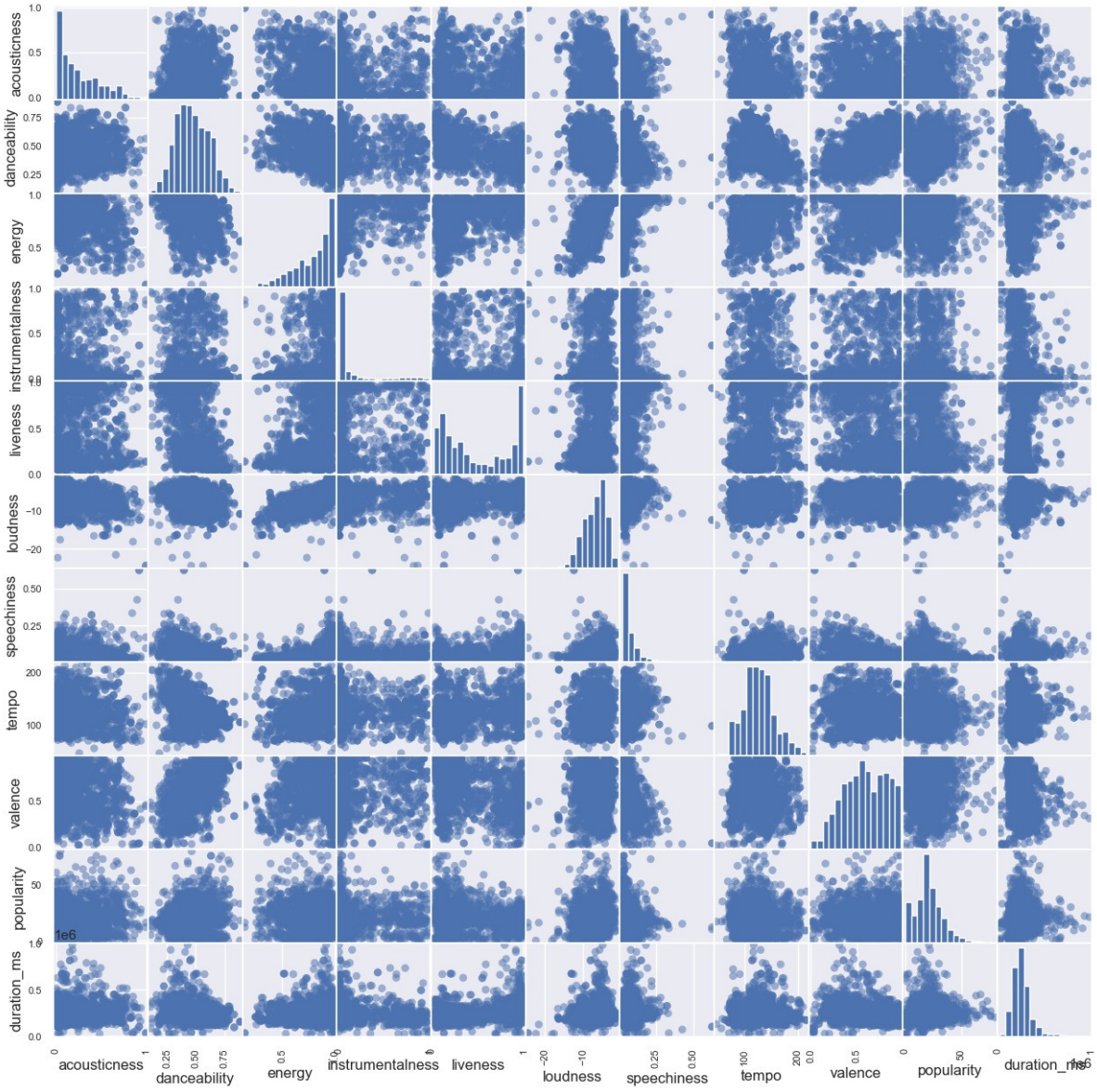
Finding:

- Most of the duration of the songs over the years are same, some years have more numbre of durable song and some have less

Perform exploratory data analysis to dive deeper into different features of songs and identify the pattern.

In [188…

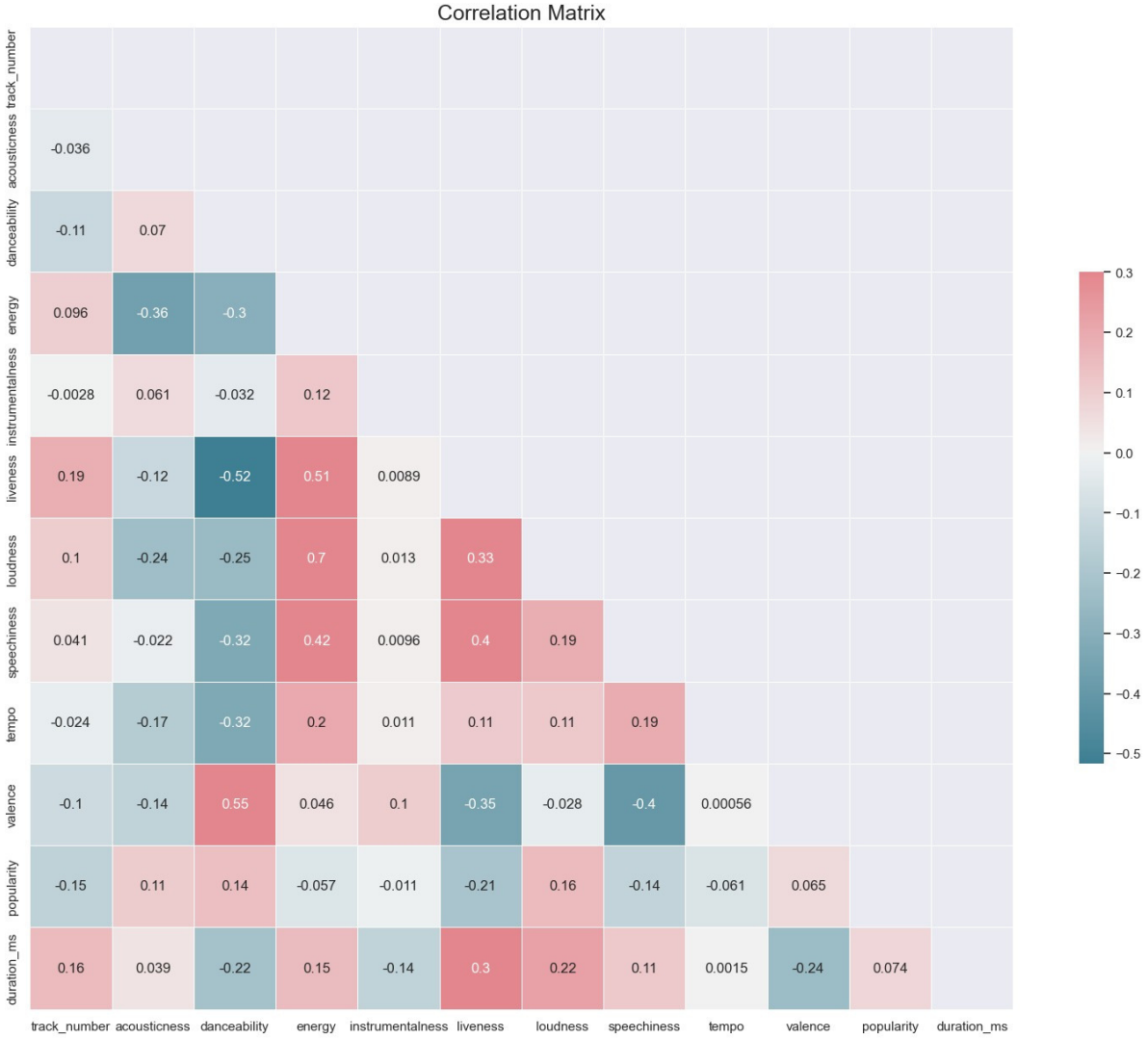
```
# audio features
# aud = ['valence', 'energy', 'speechiness','tempo']
X_aud = song_dat.iloc[:,6:]
cmap = cm.get_cmap('gnuplot')
scatter = pd.plotting.scatter_matrix(X_aud, marker = 'o', s=40, hist_kwds={'bins':1
save_plots('Features relationship')
```



Ploting heatmap to check how the other features are correlated with the popularity.

In [189…

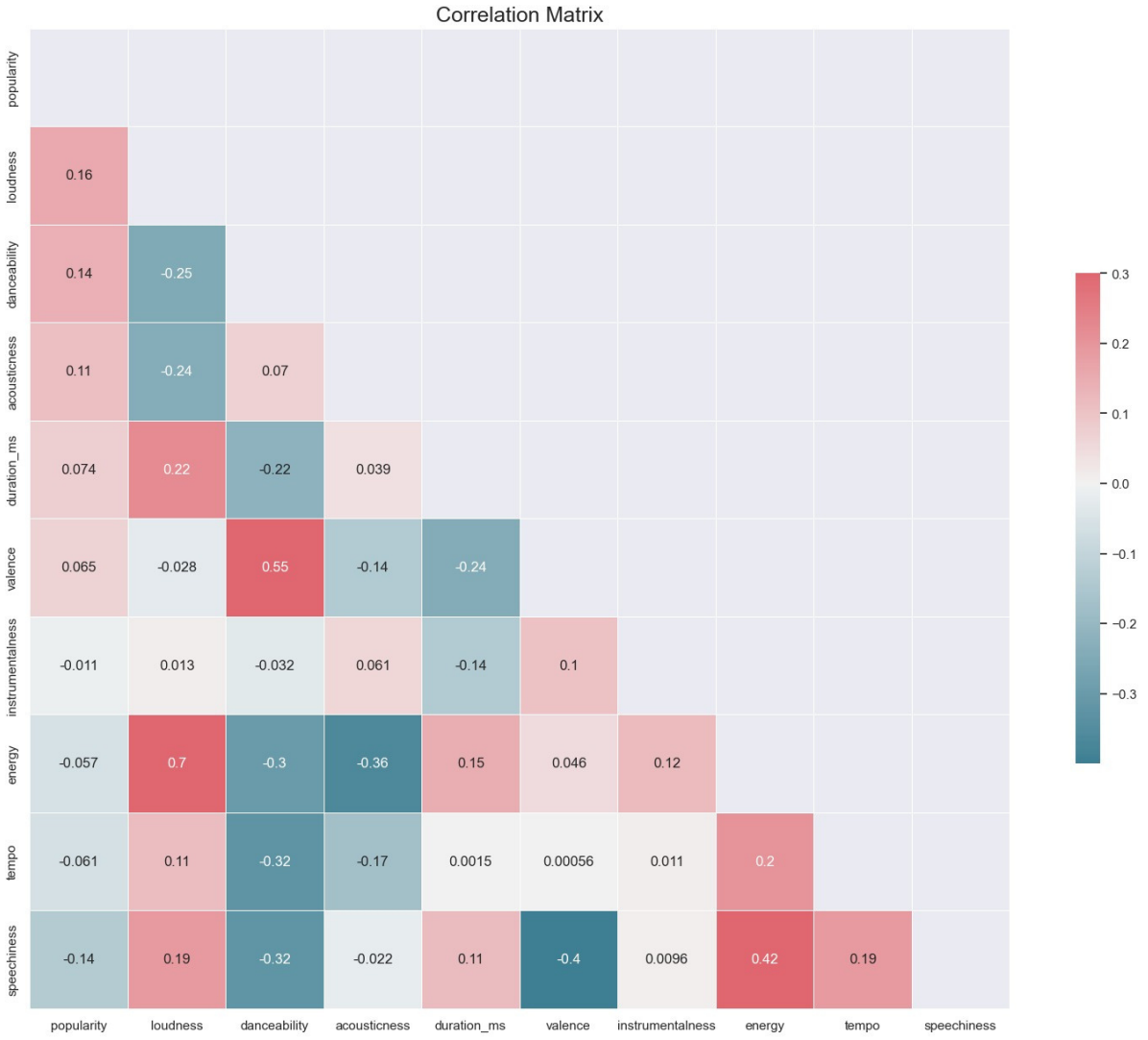
```
corr = song_dat.corr()

mask = np.triu(np.ones_like(corr, dtype=bool))

f, ax = plt.subplots(figsize=(20, 15))

cmap = sns.diverging_palette(220, 10, as_cmap=True)

plt.title('Correlation Matrix', fontsize=18)

sns.heatmap(corr,
 mask=mask,
 cmap=cmap,
 vmax=.3,
 center=0,
 square=True,
 linewidths=.5,
 cbar_kws={"shrink": .5},
 annot=True)
save_plots('Heatmap 1')
plt.show()
```



Reduced the features to select the most 10 coorelated features with the popularity.

In [190…

```
# number of variables to be selected
k = 10

# finding the most correlated variables
cols = song_dat.corr().nlargest(k, 'popularity')['popularity'].index
corr=song_dat[cols].corr()
mask = np.triu(np.ones_like(corr, dtype=bool))

f, ax = plt.subplots(figsize=(20, 15))

cmap = sns.diverging_palette(220, 10, as_cmap=True)

plt.title('Correlation Matrix', fontsize=18)

sns.heatmap(corr,
 mask=mask,
 cmap=cmap,
 vmax=.3,
 center=0,
 square=True,
 linewidths=.5,
 cbar_kws={"shrink": .5},
 annot=True)
save_plots('Heatmap 2')
plt.show()
```



Finding:

- Negative coorelation between popularity and speechiness
- Negative coorelation between tempo and popularity
- Negative coorelation between energy and popularity
- Positive coorelatoin between danceability and popularity
- Positive coorelaton between loudness and popularity
- Positive coorelaton between acousticness and popularity

Discover how a song's popularity relates to various factors and how this has changed over time.

Yearly data to check what features in which are more coolreated to the popularity. And how things change over time.

Relation between acousticness and popularity over time.

In [191…

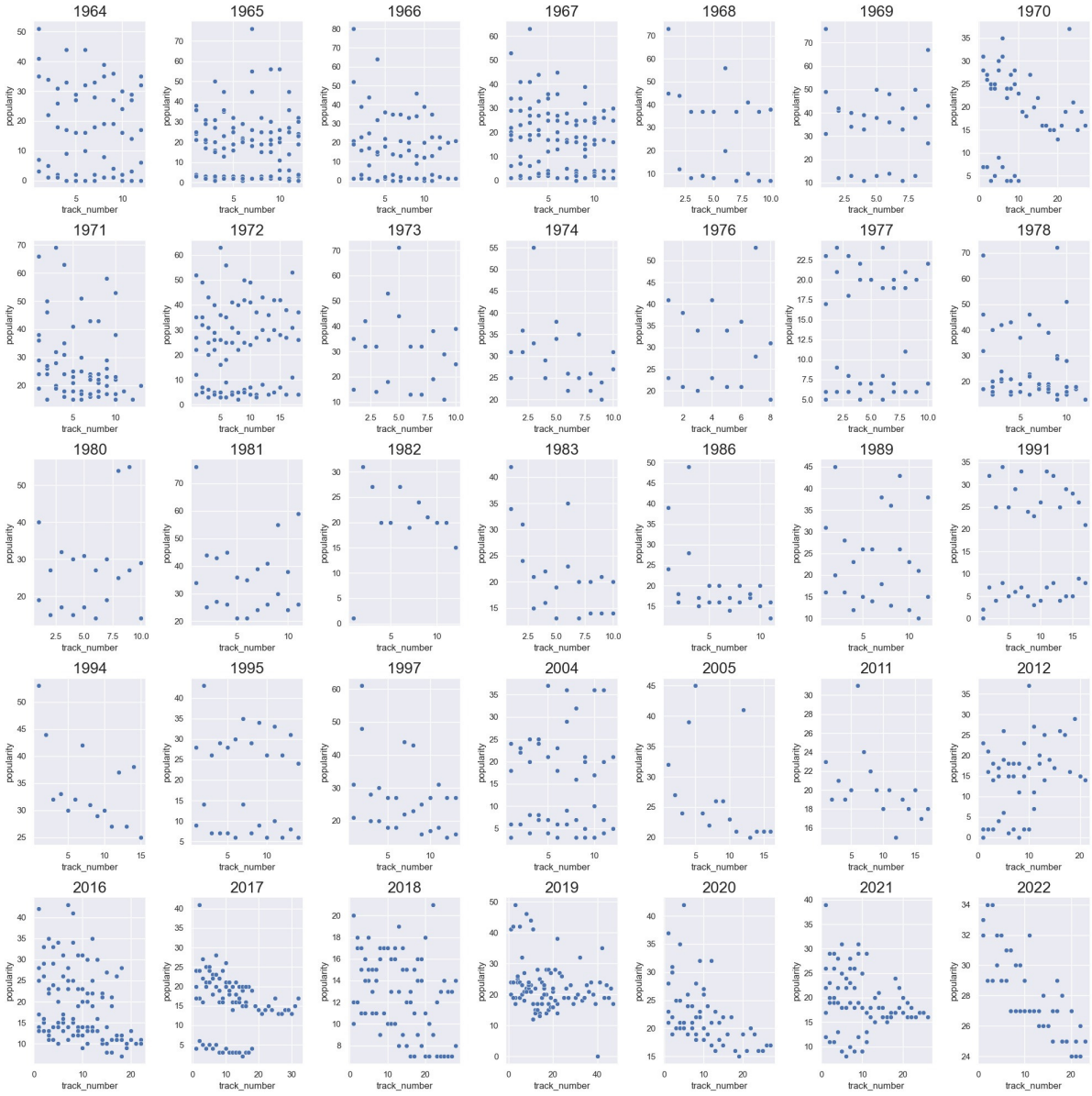
```
fig,ax=plt.subplots(5,7,figsize=(20,20))
for year,ax in zip(years,ax.flatten()):
 tmp_df=song_dat[song_dat['release_date'].dt.year==year]
    sns.scatterplot(data=tmp_df,x='acousticness',y='popularity',ax=ax)
ax.set_title(year,fontsize=20)
save_plots('acousticness vs popularity')
```

```
plt.tight_layout()
plt.show()
```

Relation between track_number and popularity over time.

In [192…

```
fig,ax=plt.subplots(5,7,figsize=(20,20))
for year,ax in zip(years,ax.flatten()):
 tmp_df=song_dat[song_dat['release_date'].dt.year==year]
    sns.scatterplot(data=tmp_df,x='track_number',y='popularity',ax=ax)
ax.set_title(year,fontsize=20)
save_plots('track_number vs popularity')
plt.tight_layout()
plt.show()
```



Relation between danceability and popularity over time.

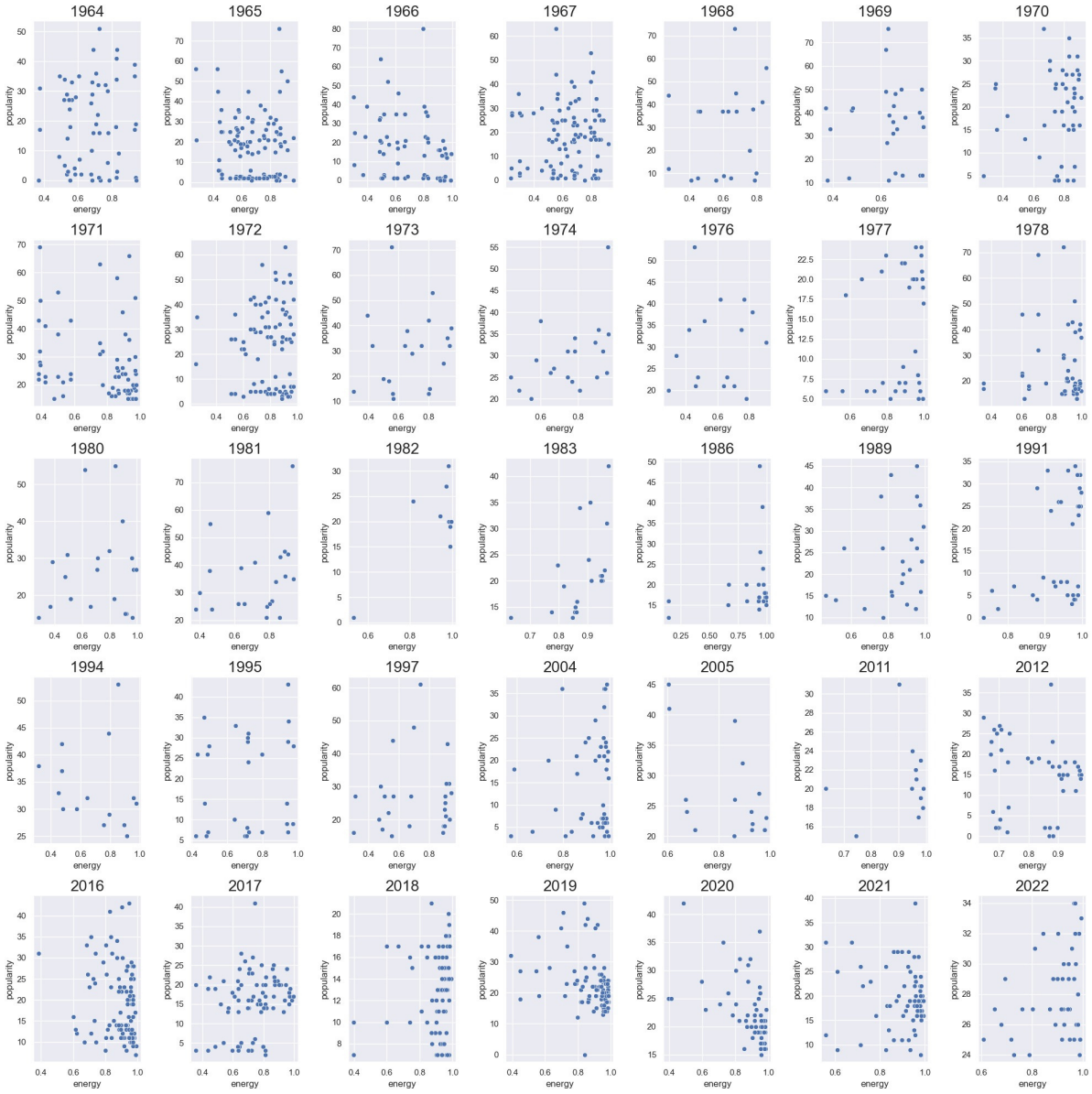
In [193…

```
fig,ax=plt.subplots(5,7,figsize=(20,20))
for year,ax in zip(years,ax.flatten()):
 tmp_df=song_dat[song_dat['release_date'].dt.year==year]
    sns.scatterplot(data=tmp_df,x='danceability',y='popularity',ax=ax)
ax.set_title(year,fontsize=20)
save_plots('daceability vs popularity')
plt.tight_layout()
plt.show()
```

Relation between energy and popularity over time.

In [194…

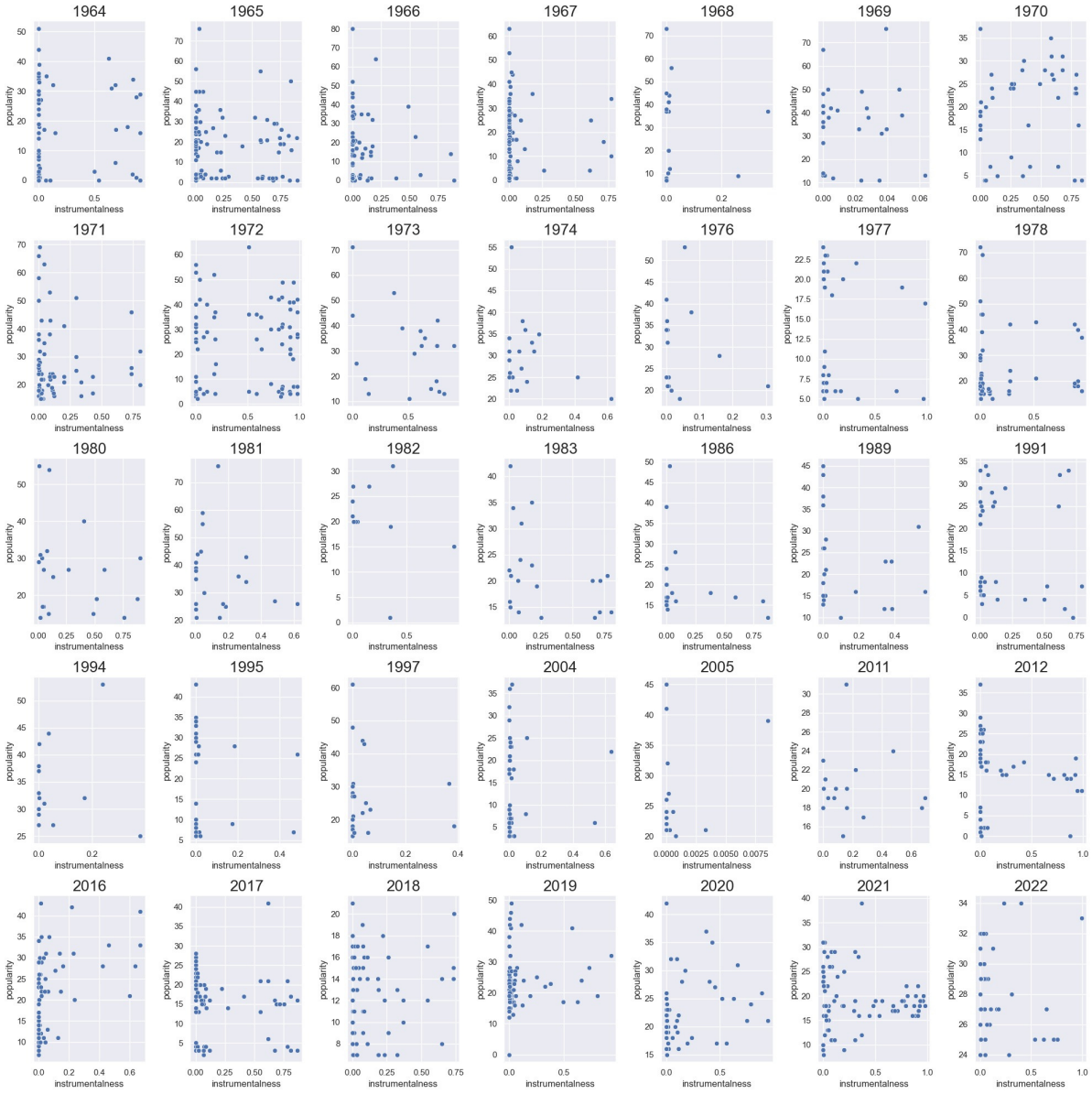
```
fig,ax=plt.subplots(5,7,figsize=(20,20))
for year,ax in zip(years,ax.flatten()):
 tmp_df=song_dat[song_dat['release_date'].dt.year==year]
 sns.scatterplot(data=tmp_df,x='energy',y='popularity',ax=ax)
 ax.set_title(year,fontsize=20)
save_plots('energy vs popularity')
plt.tight_layout()
plt.show()
```



Relation between instrumentalness and popularity over time.

In [195…

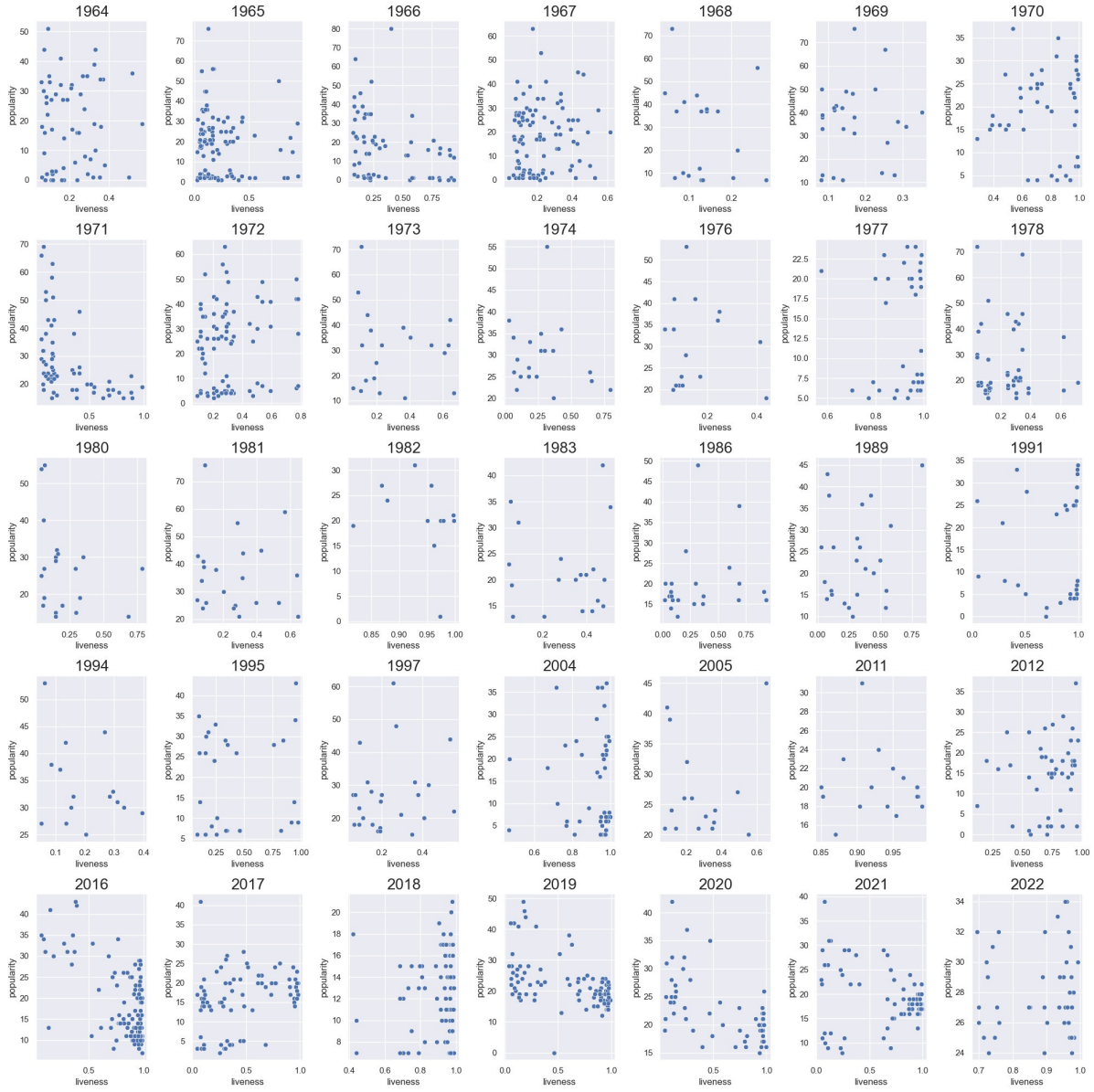
```
fig,ax=plt.subplots(5,7,figsize=(20,20))
for year,ax in zip(years,ax.flatten()):
 tmp_df=song_dat[song_dat['release_date'].dt.year==year]
 sns.scatterplot(data=tmp_df,x='instrumentalness',y='popularity',ax=ax)
 ax.set_title(year,fontsize=20)
save_plots('instrumetalness vs popularity')
plt.tight_layout()
plt.show()
```



Relation between liveness and popularity over time.

In [196…

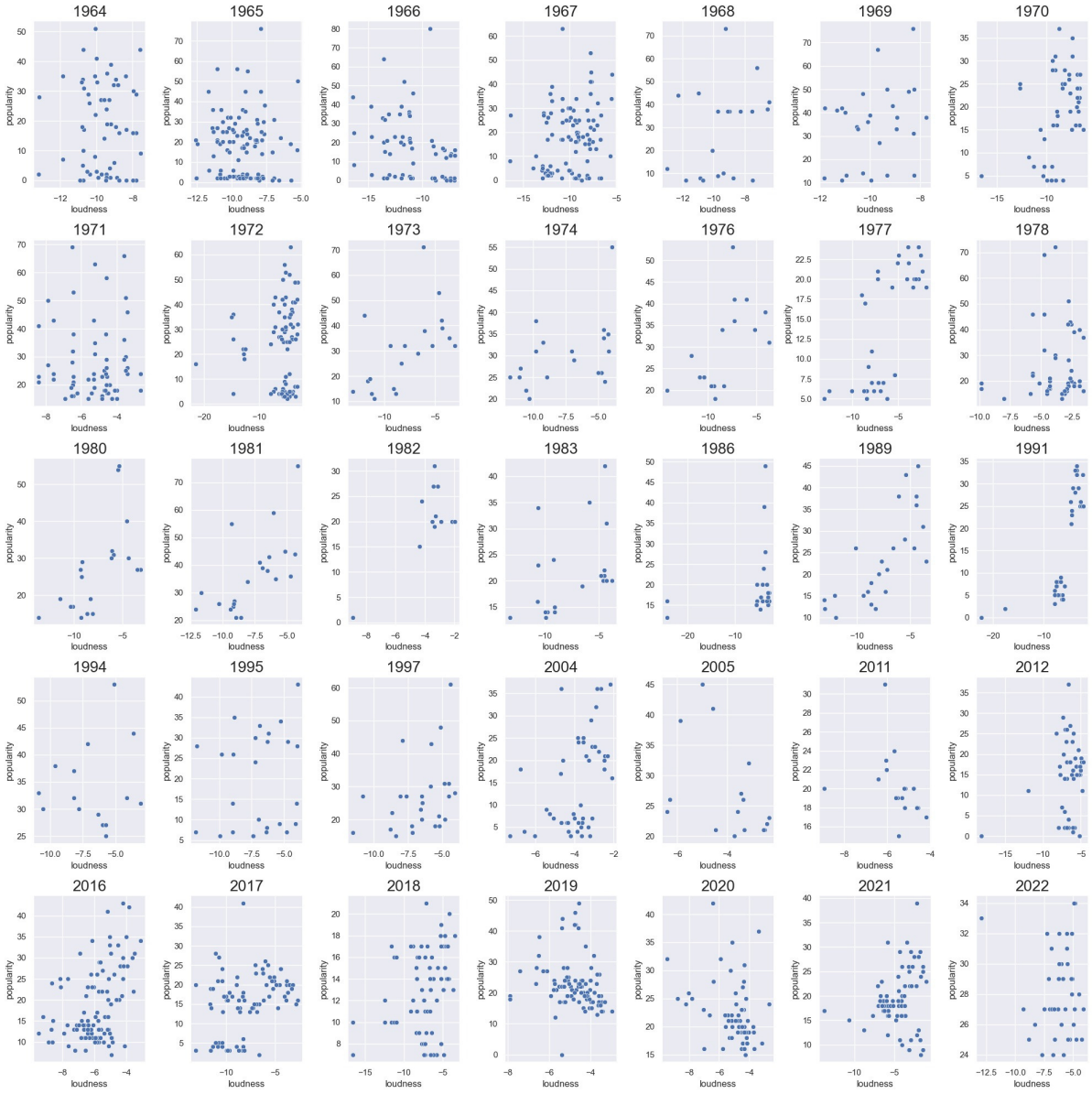
```
fig,ax=plt.subplots(5,7,figsize=(20,20))
for year,ax in zip(years,ax.flatten()):
 tmp_df=song_dat[song_dat['release_date'].dt.year==year]
 sns.scatterplot(data=tmp_df,x='liveness',y='popularity',ax=ax)
 ax.set_title(year,fontsize=20)
save_plots('liveness vs popularity')
plt.tight_layout()
plt.show()
```



Relation between loudness and popularity over time.

In [197…

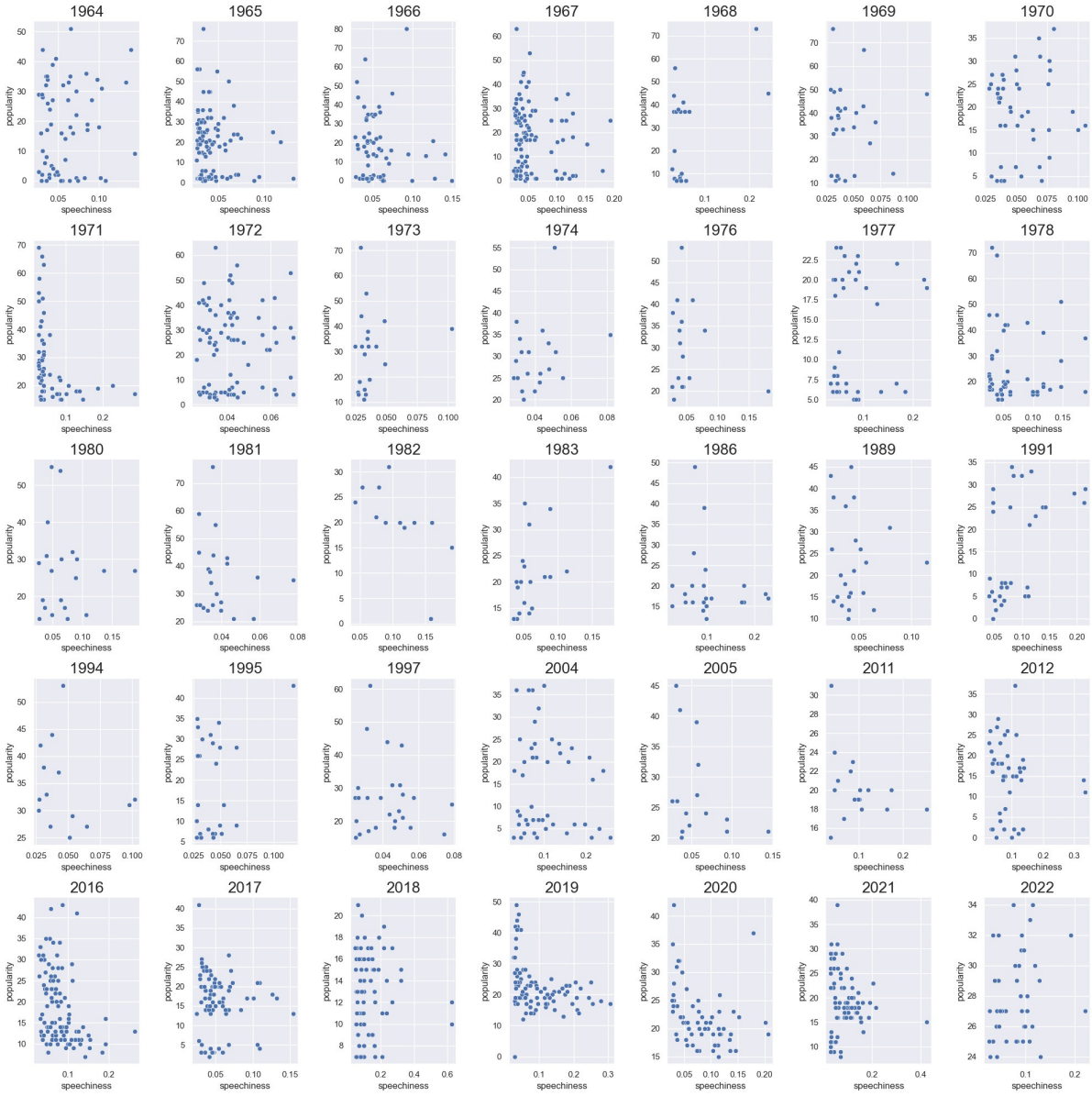
```
fig,ax=plt.subplots(5,7,figsize=(20,20))
for year,ax in zip(years,ax.flatten()):
 tmp_df=song_dat[song_dat['release_date'].dt.year==year]
 sns.scatterplot(data=tmp_df,x='loudness',y='popularity',ax=ax)
 ax.set_title(year,fontsize=20)
save_plots('loudness vs popularity')
plt.tight_layout()
plt.show()
```



Relation between speechiness and popularity over time.

In [198…

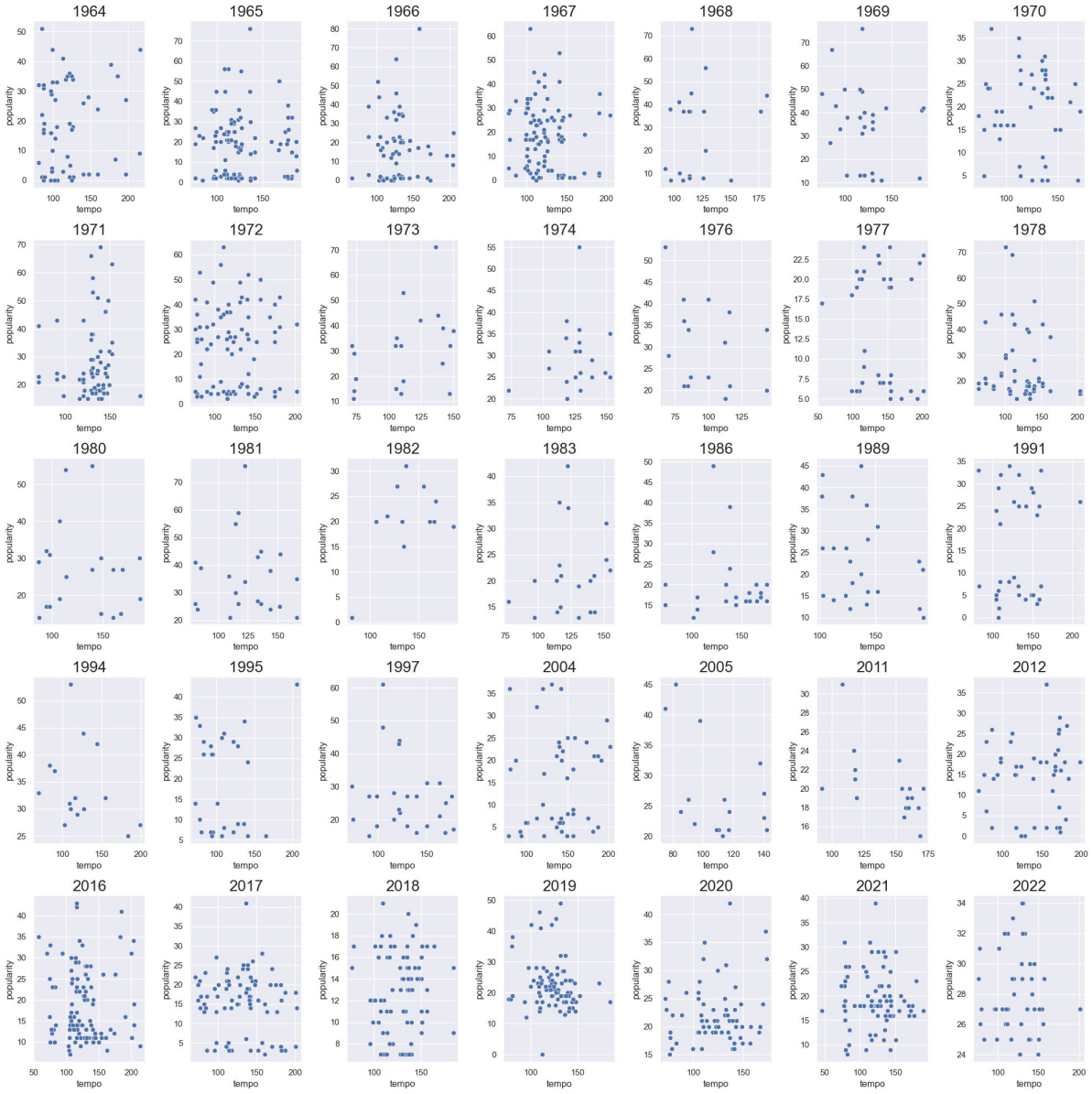
```
fig,ax=plt.subplots(5,7,figsize=(20,20))
for year,ax in zip(years,ax.flatten()):
 tmp_df=song_dat[song_dat['release_date'].dt.year==year]
    sns.scatterplot(data=tmp_df,x='speechiness',y='popularity',ax=ax)
ax.set_title(year,fontsize=20)
save_plots('speechiness vs popularity')
plt.tight_layout()
plt.show()
```



Relation between tempo and popularity over time.

In [199…

```
fig,ax=plt.subplots(5,7,figsize=(20,20))
for year,ax in zip(years,ax.flatten()):
 tmp_df=song_dat[song_dat['release_date'].dt.year==year]
    sns.scatterplot(data=tmp_df,x='tempo',y='popularity',ax=ax)
ax.set_title(year,fontsize=20)
save_plots('tempo vs popularity')
plt.tight_layout()
plt.show()
```



Relation between valence and popularity over time.

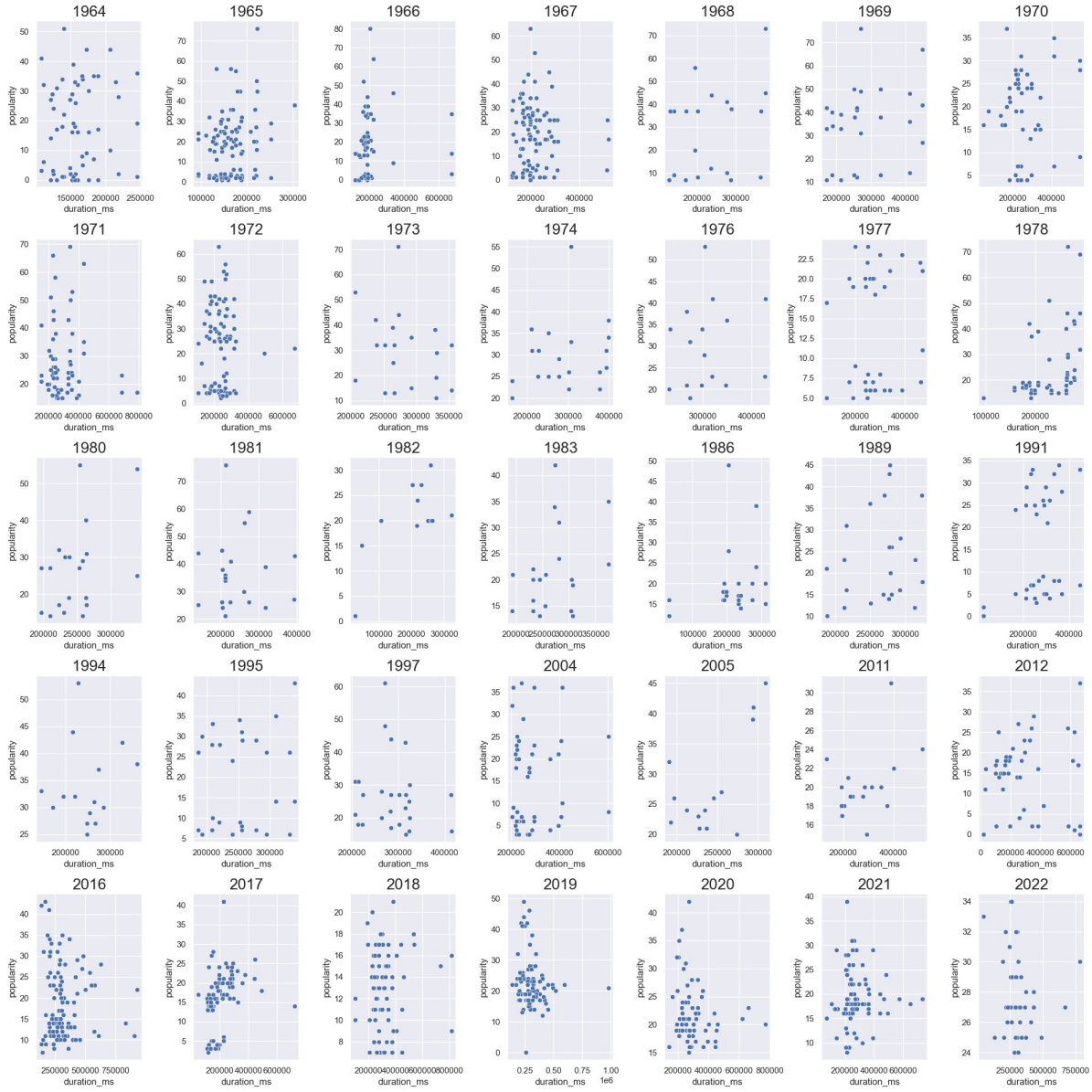
In [200…

```
fig,ax=plt.subplots(5,7,figsize=(20,20))
for year,ax in zip(years,ax.flatten()):
 tmp_df=song_dat[song_dat['release_date'].dt.year==year]
 sns.scatterplot(data=tmp_df,x='valence',y='popularity',ax=ax)
 ax.set_title(year,fontsize=20)
save_plots('valence vs popularity')
plt.tight_layout()
plt.show()
```

Relation duration_ms and popularity over time.

In [201…

```
fig,ax=plt.subplots(5,7,figsize=(20,20))
for year,ax in zip(years,ax.flatten()):
 tmp_df=song_dat[song_dat['release_date'].dt.year==year]
 sns.scatterplot(data=tmp_df,x='duration_ms',y='popularity',ax=ax)
 ax.set_title(year,fontsize=20)
save_plots('duration_ms vs popularity')
plt.tight_layout()
plt.show()
```

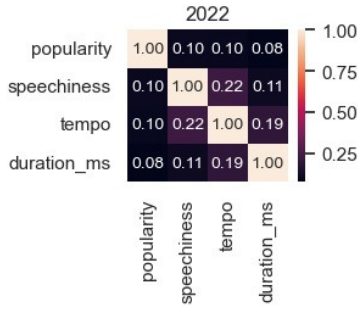


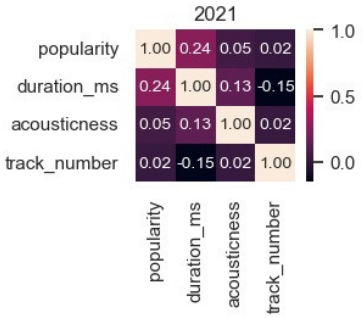
Finding:

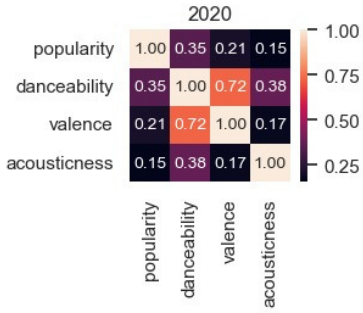
- We can see that there is a change in relationship between popularity and different feataures over time.

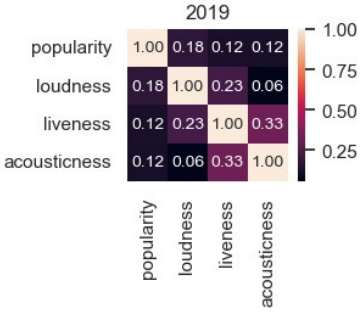
In [202…

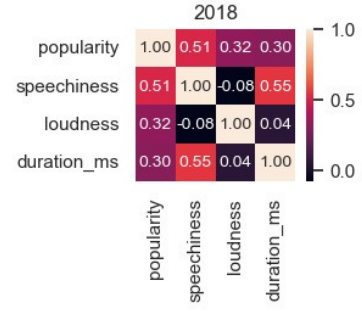
```
fig,ax=plt.subplots(12,3,figsize=(10,35))
for indx,year_ax in enumerate(zip(np.flip(years),ax.flatten())):
 cols = years_df[indx].corr().nlargest(4, 'popularity')['popularity'].index
 cm = np.corrcoef(years_df[indx][cols].values.T)
# plotting the heatmap
 year_ax[1].set_title(year_ax[0])
 sns.heatmap(cm, cbar=True, annot=True, fmt='.2f', annot_kws={'size': 10}, ytick
plt.tight_layout()
plt.show()
save_plots('Heatmap for the all the years')
```

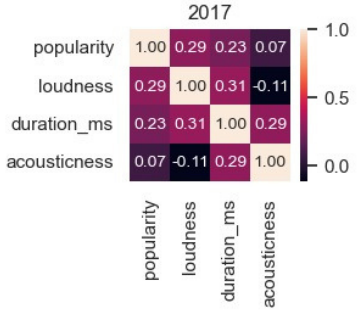

2022
popularity 1.00 0.10 0.10 0.08
speechiness 0.10 1.00 0.22 0.11
tempo 0.10 0.22 1.00 0.19
duration_ms 0.08 0.11 0.19 1.00
popularity speechiness tempo duration_ms
1.00 0.75 0.50 0.25

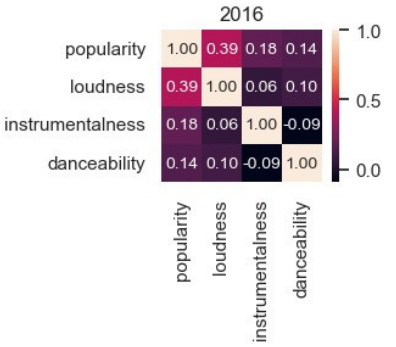

2021
popularity 1.00 0.24 0.05 0.02
duration_ms 0.24 1.00 0.13 -0.15
acousticness 0.05 0.13 1.00 0.02
track_number 0.02 -0.15 0.02 1.00
popularity duration_ms acousticness track_number
1.0 0.5 0.0

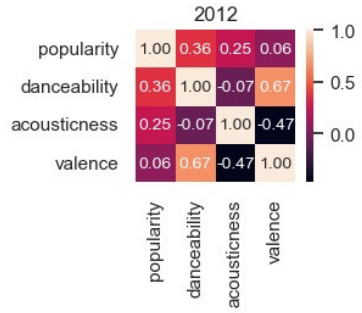

2020
popularity 1.00 0.35 0.21 0.15
danceability 0.35 1.00 0.72 0.38
valence 0.21 0.72 1.00 0.17
acousticness 0.15 0.38 0.17 1.00
popularity danceability valence acousticness
1.00 0.75 0.50 0.25

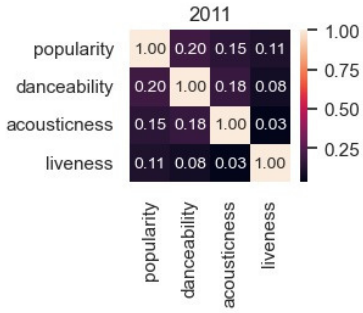

2019
popularity 1.00 0.18 0.12 0.12
loudness 0.18 1.00 0.23 0.06
liveness 0.12 0.23 1.00 0.33
acousticness 0.12 0.06 0.33 1.00
popularity loudness liveness acousticness
1.00 0.75 0.50 0.25

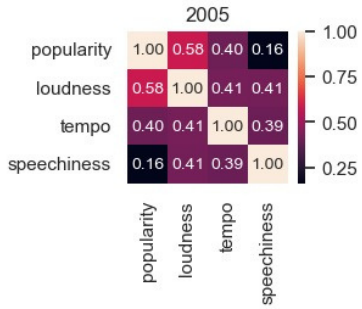

2018
popularity 1.00 0.51 0.32 0.30
speechiness 0.51 1.00 -0.08 0.55
loudness 0.32 -0.08 1.00 0.04
duration_ms 0.30 0.55 0.04 1.00
popularity speechiness loudness duration_ms
1.0 0.5 0.0

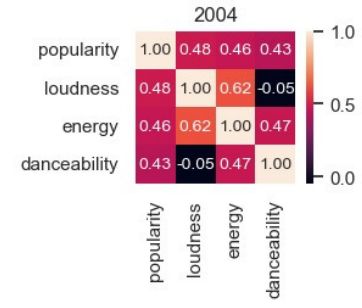

2017
popularity 1.00 0.29 0.23 0.07
loudness 0.29 1.00 0.31 -0.11
duration_ms 0.23 0.31 1.00 0.29
acousticness 0.07 -0.11 0.29 1.00
popularity loudness duration_ms acousticness
1.0 0.5 0.0

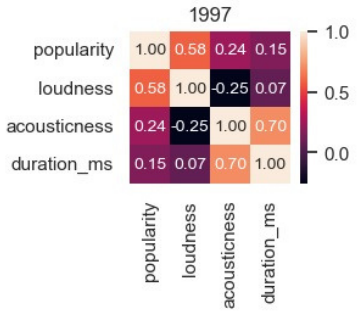

2016
popularity 1.00 0.39 0.18 0.14
loudness 0.39 1.00 0.06 0.10
instrumentalness 0.18 0.06 1.00 -0.09
danceability 0.14 0.10 -0.09 1.00
popularity loudness instrumentalness danceability
1.0 0.5 0.0

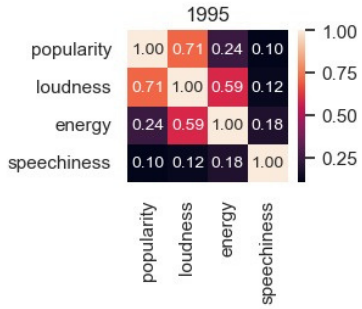

2012
popularity 1.00 0.36 0.25 0.06
danceability 0.36 1.00 -0.07 0.67
acousticness 0.25 -0.07 1.00 -0.47
valence 0.06 0.67 -0.47 1.00
popularity danceability acousticness valence
1.0 0.5 0.0

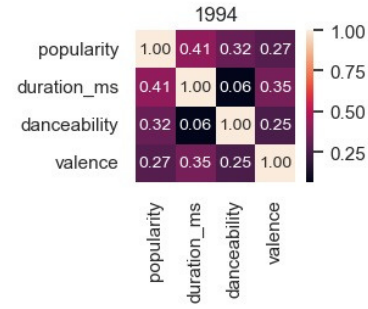

2011
popularity 1.00 0.20 0.15 0.11
danceability 0.20 1.00 0.18 0.08
acousticness 0.15 0.18 1.00 0.03
liveness 0.11 0.08 0.03 1.00
popularity danceability acousticness liveness
1.00 0.75 0.50 0.25

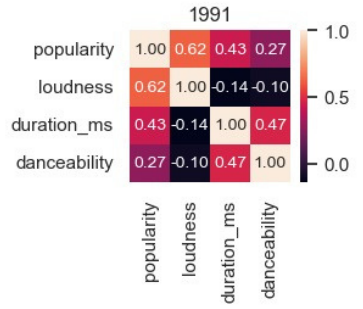

2005
popularity 1.00 0.58 0.40 0.16
loudness 0.58 1.00 0.41 0.41
tempo 0.40 0.41 1.00 0.39
speechiness 0.16 0.41 0.39 1.00
popularity loudness tempo speechiness
1.00 0.75 0.50 0.25

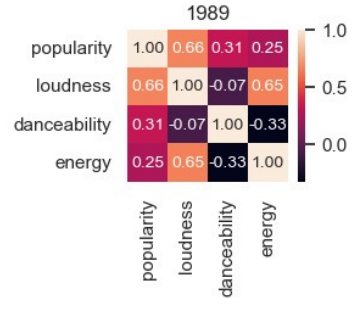

2004
popularity 1.00 0.48 0.46 0.43
loudness 0.48 1.00 0.62 -0.05
energy 0.46 0.62 1.00 0.47
danceability 0.43 -0.05 0.47 1.00
popularity loudness energy danceability
1.0 0.5 0.0

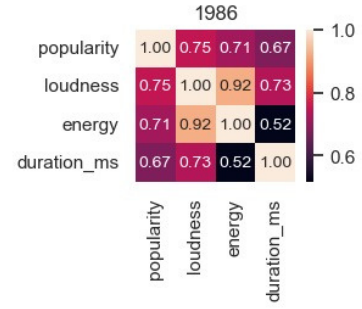

1997
popularity 1.00 0.58 0.24 0.15
loudness 0.58 1.00 -0.25 0.07
acousticness 0.24 -0.25 1.00 0.70
duration_ms 0.15 0.07 0.70 1.00
popularity loudness acousticness duration_ms
1.0 0.5 0.0

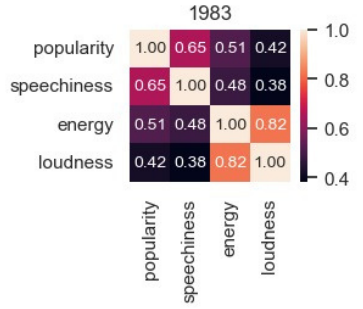

1995
popularity 1.00 0.71 0.24 0.10
loudness 0.71 1.00 0.59 0.12
energy 0.24 0.59 1.00 0.18
speechiness 0.10 0.12 0.18 1.00
popularity loudness energy speechiness
1.00 0.75 0.50 0.25


1994
popularity 1.00 0.41 0.32 0.27
duration_ms 0.41 1.00 0.06 0.35
danceability 0.32 0.06 1.00 0.25
valence 0.27 0.35 0.25 1.00
popularity duration_ms danceability valence
1.00 0.75 0.50 0.25


1991
popularity 1.00 0.62 0.43 0.27
loudness 0.62 1.00 -0.14 -0.10
duration_ms 0.43 -0.14 1.00 0.47
danceability 0.27 -0.10 0.47 1.00
popularity loudness duration_ms danceability
1.0 0.5 0.0


1989
popularity 1.00 0.66 0.31 0.25
loudness 0.66 1.00 -0.07 0.65
danceability 0.31 -0.07 1.00 -0.33
energy 0.25 0.65 -0.33 1.00
popularity loudness danceability energy
1.0 0.5 0.0


1986
popularity 1.00 0.75 0.71 0.67
loudness 0.75 1.00 0.92 0.73
energy 0.71 0.92 1.00 0.52
duration_ms 0.67 0.73 0.52 1.00
popularity loudness energy duration_ms
1.0 0.8 0.6


1983
popularity 1.00 0.65 0.51 0.42
speechiness 0.65 1.00 0.48 0.38
energy 0.51 0.48 1.00 0.82
loudness 0.42 0.38 0.82 1.00
popularity speechiness energy loudness
1.0 0.8 0.6 0.4


1982
popularity 1.00 0.52 0.38 0.25
danceability 0.52 1.00 0.70 0.10
valence 0.38 0.70 1.00 0.25
energy 0.25 0.10 0.25 1.00
popularity danceability valence energy
1981
popularity 1.00 0.75 0.31 0.22
loudness 0.75 1.00 0.78 -0.09
energy 0.31 0.78 1.00 -0.24
duration_ms 0.22 -0.09 -0.24 1.00
popularity loudness energy duration_ms
1980
popularity 1.00 0.68 0.51 0.43
loudness 0.68 1.00 0.38 0.73
speechiness 0.51 0.38 1.00 0.44
energy 0.43 0.73 0.44 1.00
popularity loudness speechiness energy
1978
popularity 1.00 0.15 0.12 0.12
acousticness 0.15 1.00 -0.59 0.49
loudness 0.12 -0.59 1.00 0.07
duration_ms 0.12 0.49 0.07 1.00
popularity acousticness loudness duration_ms
1977
popularity 1.00 0.30 0.16 0.13
speechiness 0.30 1.00 0.52 0.19
loudness 0.16 0.52 1.00 0.07
duration_ms 0.13 0.19 0.07 1.00
popularity speechiness loudness duration_ms
1976
popularity 1.00 0.36 0.33 0.17
loudness 0.36 1.00 -0.17 -0.24
danceability 0.33 -0.17 1.00 0.17
liveness 0.17 -0.24 0.17 1.00
popularity loudness danceability liveness
1974
popularity 1.00 0.46 0.14 0.08
loudness 0.46 1.00 0.85 0.47
energy 0.14 0.85 1.00 0.46
tempo 0.08 0.47 0.46 1.00
popularity loudness energy tempo
1973
popularity 1.00 0.64 0.28 0.06
duration_ms 0.64 1.00 0.37 0.24
instrumentalness 0.28 0.37 1.00 -0.29
liveness 0.06 0.24 -0.29 1.00
popularity duration_ms instrumentalness liveness
1972
popularity 1.00 0.87 0.46 0.41
danceability 0.87 1.00 0.55 0.39
valence 0.46 0.55 1.00 0.12
duration_ms 0.41 0.39 0.12 1.00
popularity danceability valence duration_ms
1971
popularity 1.00 0.35 0.22 0.20
track_number 0.35 1.00 0.11 0.22
loudness 0.22 0.11 1.00 -0.02
liveness 0.20 0.22 -0.02 1.00
popularity track_number loudness liveness
1970
popularity 1.00 0.46 0.42 0.27
instrumentalness 0.46 1.00 0.37 -0.00
loudness 0.42 0.37 1.00 -0.44
acousticness 0.27 -0.00 -0.44 1.00
popularity instrumentalness loudness acousticness
1969
popularity 1.00 0.35 0.33 0.32
loudness 0.35 1.00 0.68 0.55
liveness 0.33 0.68 1.00 0.55
duration_ms 0.32 0.55 0.55 1.00
popularity loudness liveness duration_ms
1968
popularity 1.00 0.27 0.18 0.13
loudness 0.27 1.00 0.26 0.79
liveness 0.18 0.26 1.00 0.34
energy 0.13 0.79 0.34 1.00
popularity loudness liveness energy
1967
popularity 1.00 0.48 0.34 0.12
danceability 0.48 1.00 0.71 0.14
valence 0.34 0.71 1.00 0.35
instrumentalness 0.12 0.14 0.35 1.00
popularity danceability valence instrumentalness
1966
popularity 1.00 0.47 0.26 0.25
danceability 0.47 1.00 0.25 0.58
instrumentalness 0.26 0.25 1.00 0.29
valence 0.25 0.58 0.29 1.00
popularity danceability instrumentalness valence
1965
popularity 1.00 0.25 0.22 0.19
danceability 0.25 1.00 0.44 0.47
loudness 0.22 0.44 1.00 0.45
valence 0.19 0.47 0.45 1.00
1964
popularity 1.00 0.28 0.19 0.02
energy 0.28 1.00 0.57 -0.03
speechiness 0.19 0.57 1.00 0.37
acousticness 0.02 -0.03 0.37 1.00

<Figure size 640x480 with 0 Axes>

In [203…
```
print("Features That are coorelated with the popularity for different years")
print("------------------------------------------------------------------------")
corelation_arr=[]
for indx,year in enumerate(np.flip(years)):
 cols = years_df[indx].corr().nlargest(4, 'popularity')['popularity'].index
 cm = np.corrcoef(years_df[indx][cols].values.T)
 for col in cols[1:]:
 corelation_arr.append(col)
 print(year,cols.values[1:])
```

```
Features That are coorelated with the popularity for different years
------------------------------------------------------------------------
2022 ['speechiness' 'tempo' 'duration_ms']
2021 ['duration_ms' 'acousticness' 'track_number']
2020 ['danceability' 'valence' 'acousticness']
2019 ['loudness' 'liveness' 'acousticness']
2018 ['speechiness' 'loudness' 'duration_ms']
2017 ['loudness' 'duration_ms' 'acousticness']
2016 ['loudness' 'instrumentalness' 'danceability']
2012 ['danceability' 'acousticness' 'valence']
2011 ['danceability' 'acousticness' 'liveness']
2005 ['loudness' 'tempo' 'speechiness']
2004 ['loudness' 'energy' 'danceability']
1997 ['loudness' 'acousticness' 'duration_ms']
1995 ['loudness' 'energy' 'speechiness']
1994 ['duration_ms' 'danceability' 'valence']
1991 ['loudness' 'duration_ms' 'danceability']
1989 ['loudness' 'danceability' 'energy']
1986 ['loudness' 'energy' 'duration_ms']
1983 ['speechiness' 'energy' 'loudness']
1982 ['danceability' 'valence' 'energy']
1981 ['loudness' 'energy' 'duration_ms']
1980 ['loudness' 'speechiness' 'energy']
1978 ['acousticness' 'loudness' 'duration_ms']
1977 ['speechiness' 'loudness' 'duration_ms']
1976 ['loudness' 'danceability' 'liveness']
1974 ['loudness' 'energy' 'tempo']
1973 ['duration_ms' 'instrumentalness' 'liveness']
1972 ['danceability' 'valence' 'duration_ms']
1971 ['track_number' 'loudness' 'liveness']
1970 ['instrumentalness' 'loudness' 'acousticness']
1969 ['loudness' 'liveness' 'duration_ms']
1968 ['loudness' 'liveness' 'energy']
1967 ['danceability' 'valence' 'instrumentalness']
1966 ['danceability' 'instrumentalness' 'valence']
1965 ['danceability' 'loudness' 'valence']
1964 ['energy' 'speechiness' 'acousticness']
```

Finding:

- We see that the popularly of a song for different year is due to dirrerent reasons, som have slightly similar reason

Check the featuers that are highly related to popularity.

In [204…
```
df=pd.DataFrame(corelation_arr)
df.value_counts()
```

```
Out[204]:loudness 23
         danceability 14
          duration_ms 14
               energy 11
         acousticness 10
           speechiness 8
              valence 8
             liveness 7
     instrumentalness 5
                tempo 3
         track_number 2
dtype: int64
```

Finding:

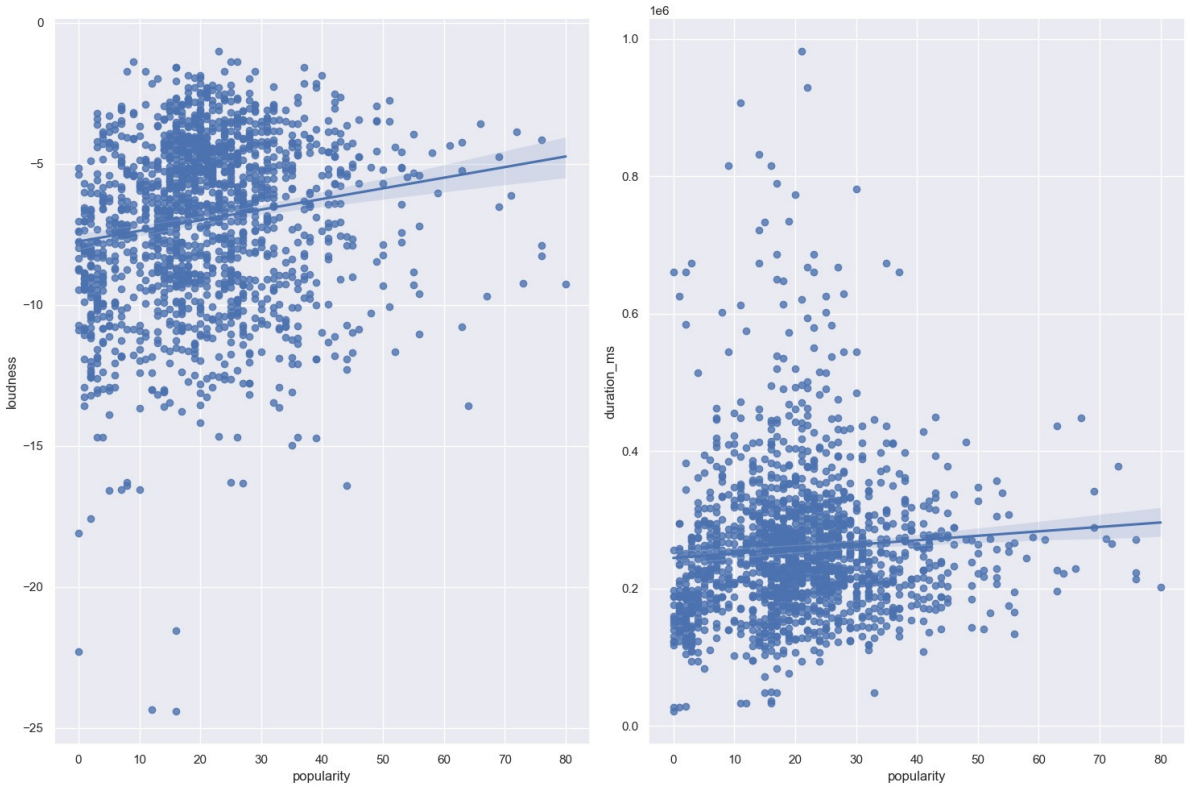
- Overall loudness is more related to the popularity of a song

Some of the most important features to make the song popular.

- Loudness of the song is the most import features to become popular in the sense that a song must have some deep voice to become popular and touch the hearth of the listener.
  Duration of the song is also important feature to become popular as if as song was only 10
- second long no one's gonna like it
  Danceability is aslo an important as if the song is good at dancing most of the people
- like the song , even though they cannot dance as they are admire by the beautiful dance performed by the dancers or enthusiast
  Energy is also important as a song which give energy is mostly want by everyone
- despite of age

In [205…

```
# scatter plots with linear regression line
fig, (ax1, ax2) = plt.subplots(ncols=2, sharex=True, figsize=(15,10));
sns.regplot(x=song_dat['popularity'], y=song_dat['loudness'],ax=ax1);
sns.regplot(x=song_dat['popularity'], y=song_dat['duration_ms'],ax=ax2);
plt.tight_layout()
save_plots('Scatter1')
```

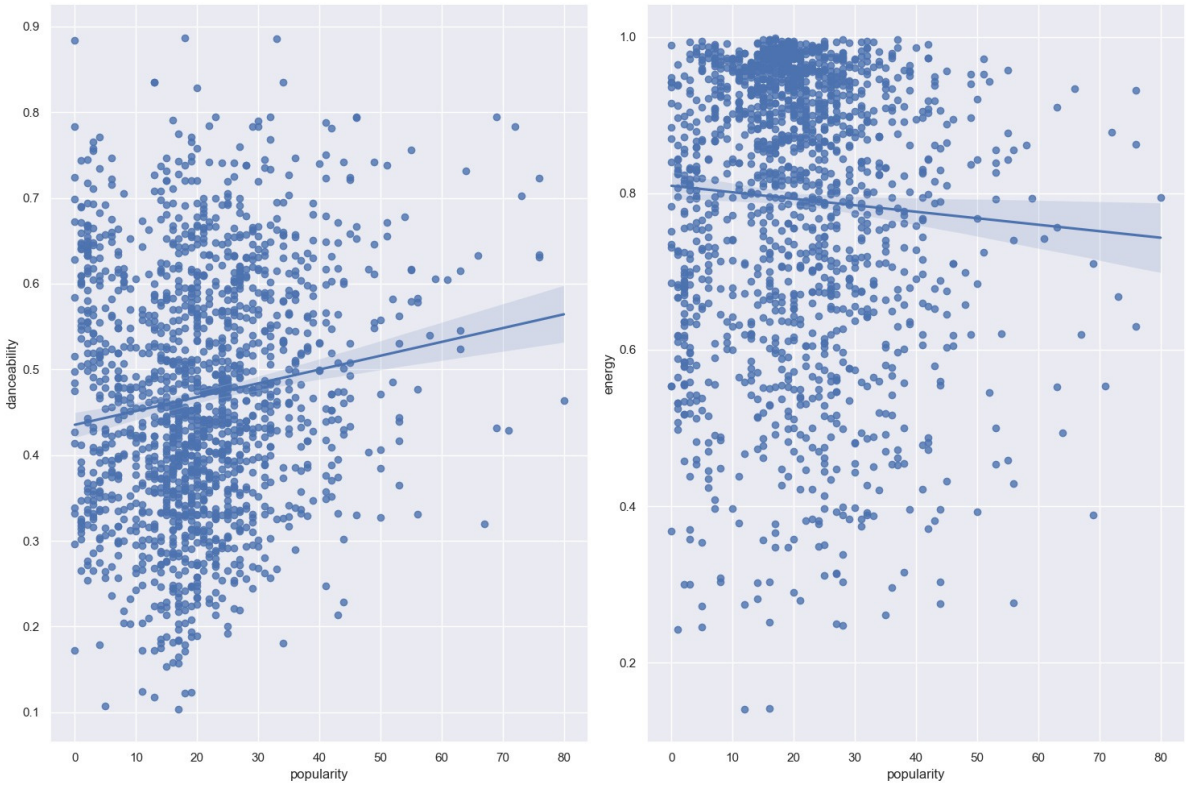


Finding:

- Most popular song have loudness in the range of 0,-10 there are some outlier present in it Most of the popular song have duration in the range of 1-4, high popular song have duration
- near 2

In [206…

```
fig, (ax1, ax2) = plt.subplots(ncols=2, sharex=True, figsize=(15,10));
sns.regplot(x=song_dat['popularity'], y=song_dat['danceability'],ax=ax1);
sns.regplot(x=song_dat['popularity'], y=song_dat['energy'],ax=ax2);
plt.tight_layout()
save_plots('Scatter2')
```



Finding:

- Danceability vary a lot and most popular song are less danceable
- Enery vary as well

In [207…

```
fig, (ax1, ax2) = plt.subplots(ncols=2, sharex=True, figsize=(15,10));
sns.regplot(x=song_dat['popularity'], y=song_dat['acousticness'],ax=ax1);
sns.regplot(x=song_dat['popularity'], y=song_dat['speechiness'],ax=ax2);
plt.tight_layout()
save_plots('Scatter3')
```

Finding:

- Acousticness vary as lot but speechiness does not vary a lot
- Like when popularity increase speechiness decrease

Comment on the importance of dimensionality reduction techniques, share your ideas and explain your observations.

As the number of features in our data increases, traditional analysis methods suffer from the "curse of dimensionality." This means models require more data, become computationally expensive, and are prone to overfitting. Dimensionality reduction alleviates this by condensing the information into a lower-dimensional space while preserving the essential patterns.

Pca is mostly commonly used dimensionality reduction technique to reduce the number of features when we have a dataset with large number of features.

Perform Cluster Analysis:

Identify the right number of clusters.

In [208…
```
model_data=song_dat.copy()
```

In [209…
```
def preprocessing_data(data,drop_cols,scale_type=StandardScaler,pca_n_component=0.9
'''
 def: function to preprocess the data which include drop the unwanted column,
 standardize the data and perform pca to give the final result. It return two va
 of the pca result.

args:
 arg1:data-The dataframe to work with
 arg2:drop_cols-The column to drop from the data
 arg3:scale_type-The standardization type
 arg4:pca_n_component-The number of component of the pca
 '''
 # Drop the unwanted columns
 X=data.drop(drop_cols,axis=1)

# Standardization the data
 scaler=scale_type()
 scale_song_df=pd.DataFrame(scaler.fit_transform(X),columns=X.columns)
 scale_song=scaler.fit_transform(X)

# Performing PCA
 pca=PCA(n_components=pca_n_component)
 result_arr=pca.fit_transform(scale_song)

# Find the number of columns
 n_cols=result_arr.shape[1]

# Generating the column name for the PCA dataframe
 comp_name=['Component '+str(x) for x in range(n_cols)]
result_df=pd.DataFrame(result_arr,columns=comp_name)
 return result_df,result_arr
```

In [210…
```
drop_cols=['track_number','name','album','release_date','id','uri']
```

In [211…
```
model_pca_df,model_pca_arr=preprocessing_data(model_data,drop_cols,MinMaxScaler,2)
```

In [212…
```
model_pca_df.to_csv(CSV_PATH+'/model_pca.csv')
```

In [213…
```
model_pca_df.head()
```

Out[213]:

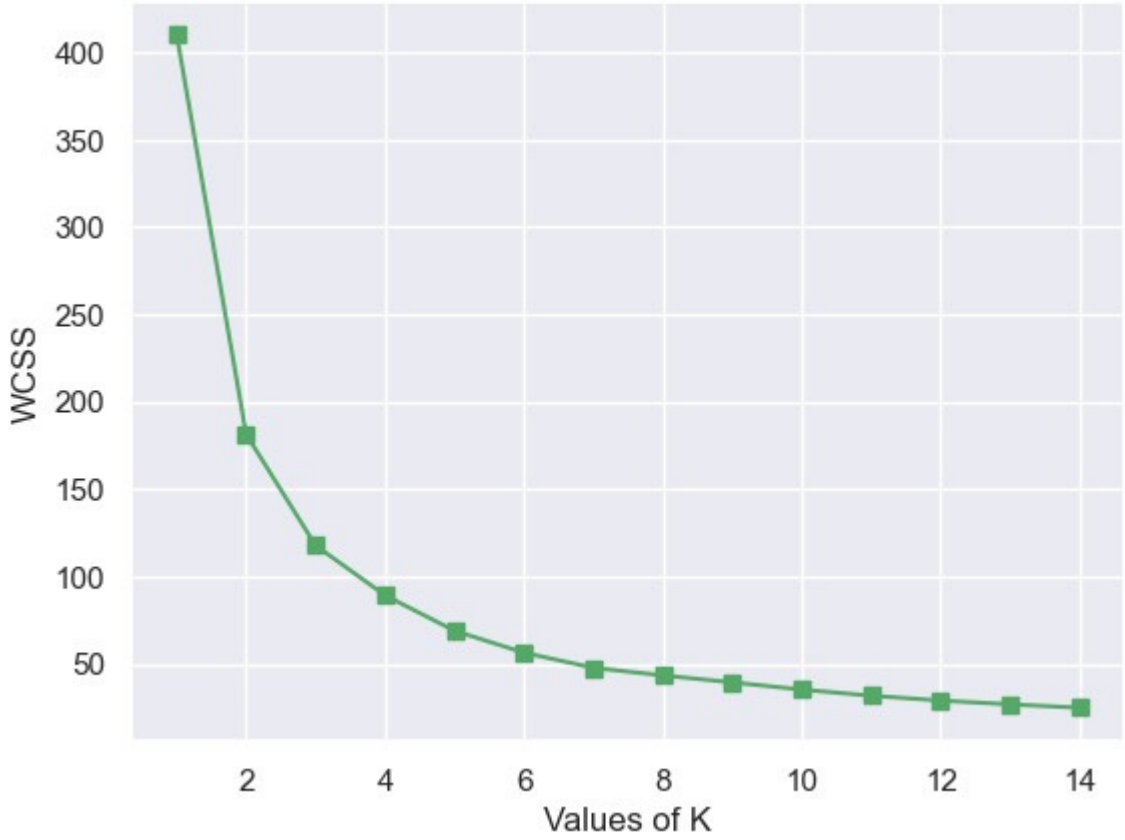
	Component 0	Component 1
0	-0.562865	0.560630
1	-0.566803	-0.032554
2	-0.553333	0.113999
3	-0.555718	-0.320689
4	-0.622233	-0.226714

In [214…
```
model_pca_arr[:2]
```

Out[214]:
```
array([[-0.56286482, 0.56063 ],
 [-0.56680332, -0.03255373]])
```

In [215…
```
# Check the number of right cluster using Elbow graph
def find_num_cluster(df):
 wcss={}

 for i in range(1,15):
                kmeans=KMeans(n_clusters=i,init='k-means++',random_state=1)
 kmeans.fit(df)
 wcss[i]=kmeans.inertia_

 plt.plot(wcss.keys(),wcss.values(),'gs-')
 plt.xlabel('Values of K')
 plt.ylabel('WCSS')
 save_plots('Elbow graph')
 plt.show()
```

In [216…
```
# scale_song_df,scale_song_arr=preprocessing_data(song_dat)
find_num_cluster(model_pca_df)
plt.show()
save_plots('Elbow K Means')
```




```
<Figure size 640x480 with 0 Axes>
```

```
In [217…# cross-validation: find optimal number of clusters k by mean silouette score
scores = []
for k in range(2,15):
 kmeans = KMeans(n_clusters=k,init='k-means++',random_state=7)
 kmeans.fit(model_pca_df)
          score = silhouette_score(model_pca_df, kmeans.labels_, metric='euclidean')
 print(k, score)
```

```
2 0.5095438173403372
3 0.5113451870633491
4 0.49105937082912654
5 0.4374107406900147
6 0.41321312699116136
7 0.39777664558407116
8 0.3644920753129331
9 0.36840066341454586
10 0.36908696351747317
11 0.37088233164556467
12 0.3656790740713023
13 0.3622239827134698
14 0.3659029265879628
```

```
In [218…# determine best clustering solution by mean silhouette score
silhouette_score(model_pca_df, kmeans.labels_, metric='euclidean')
```

```
Out[218]: 0.3659029265879628
```

Finding:

- From the above Elbow graph and silhouetee score the right number of cluster is 8. So I perform K means with 8 clusters

Use appropriate clustering algorithm.

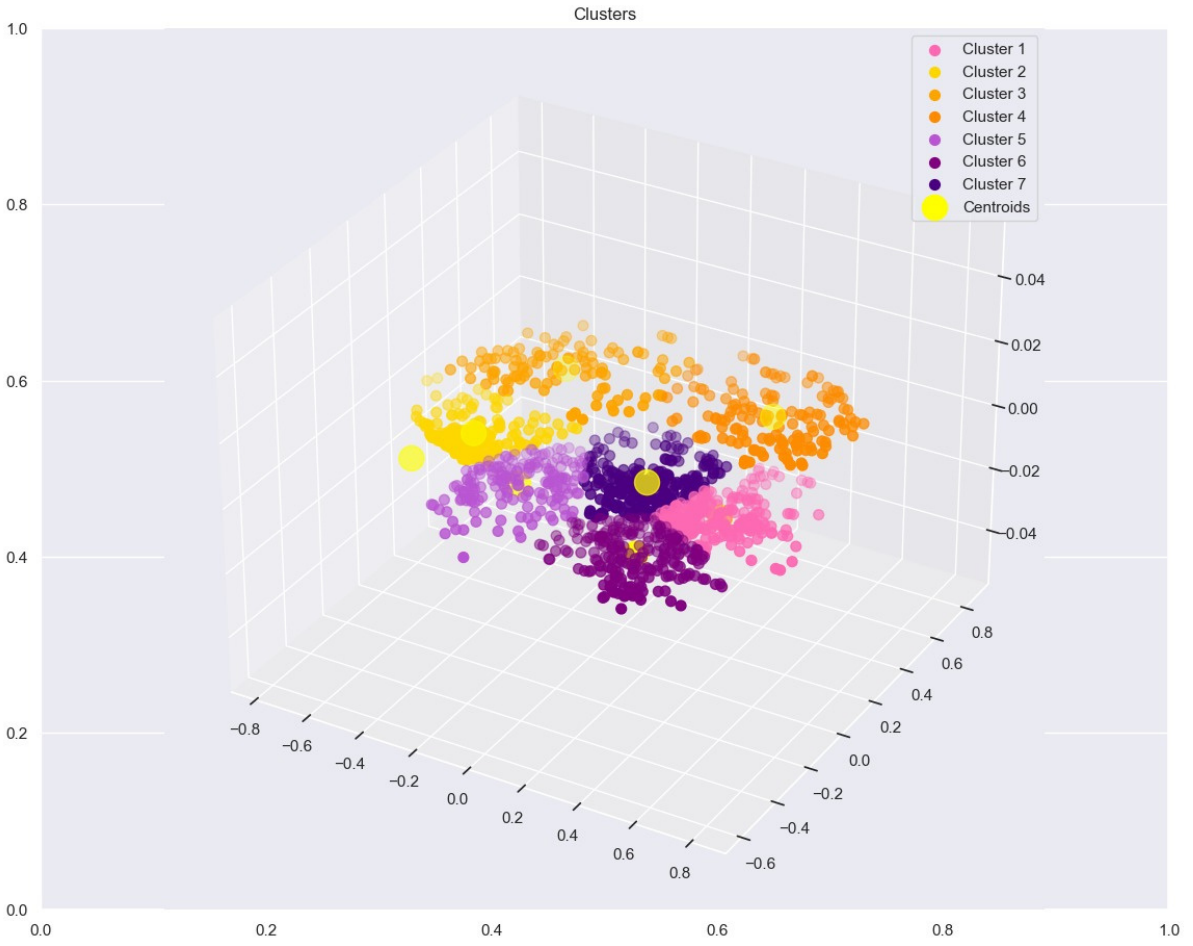
In [219…
```
# Train the model
kmeans=KMeans(n_clusters = 8, init = 'k-means++', max_iter = 300, n_init = 10, rand
y_kmeans=kmeans.fit_predict(model_pca_arr)
```

In [220…
```
y_kmeans
```

Out[220]:
```
array([2, 1, 1, ..., 0, 5, 0])
```

In [221…
```
def plot_cluster_3d(df,model,y_kmeans,num_cluster):
# colors=['red','blue','green','magenta','gray','purple','orange']
 colors=['#FF69B4','#FFD700','#FFA500','#FF8C00','#BA55D3','#800080','#4B0082','
 # Visualizing clusters
 fig,ax=plt.subplots(figsize=(14,11))
 ax=fig.add_subplot(111,projection='3d')

for i in range(num_cluster-1):
 plt.scatter(df[y_kmeans==i,0],df[y_kmeans==i,1],s=50,c=colors[i],label=f'Cl
 plt.scatter(model.cluster_centers_[:, 0], model.cluster_centers_[:,1], s= 300,
plt.title('Clusters')
 plt.legend()
 save_plots('Clusters')
 save_plots('Cluster 3d')
 plt.show()
```

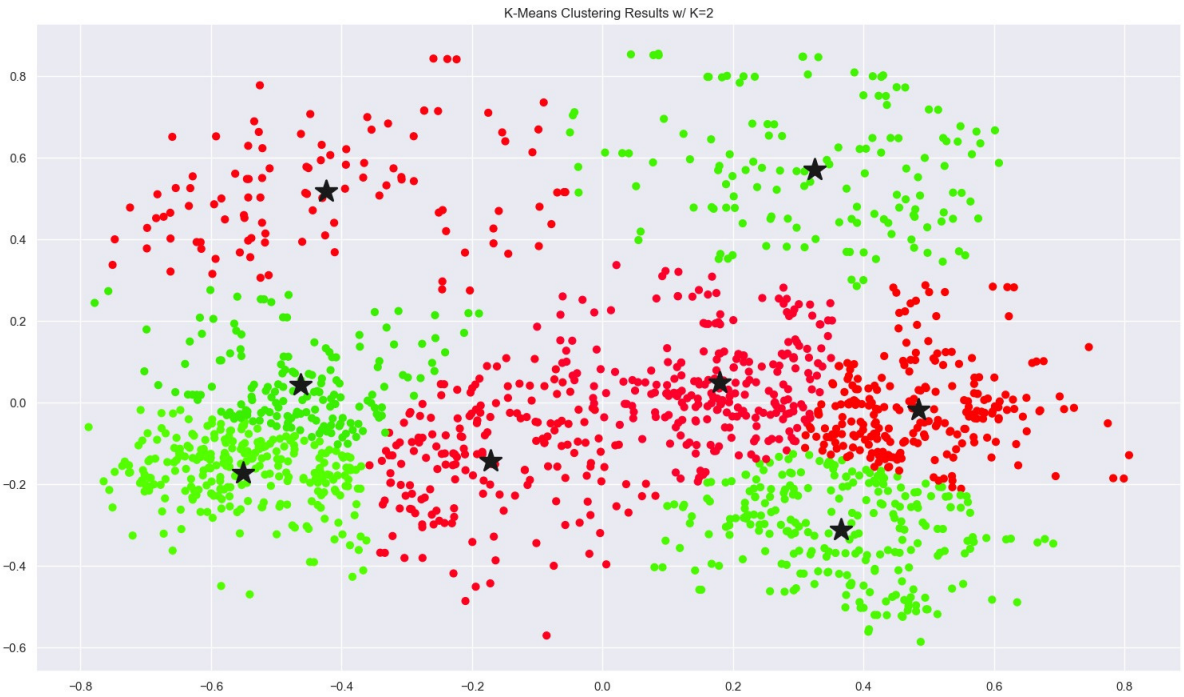
In [222…
```
plot_cluster_3d(model_pca_arr,kmeans,y_kmeans,8)
```



In [223…
```
def plot_cluster_2d(df,model):
 # visualize clusters on principle components
 plt.title('K-Means Clustering Results w/ K=2')
```

```
 plt.scatter(df[:,0], df[:,1], s=40, c=model.labels_, cmap=plt.cm.prism)
 plt.scatter(model.cluster_centers_[:,0], model.cluster_centers_[:,1], marker='*

save_plots('Cluster 2d')
 plt.gcf().set_size_inches(18.5, 10.5)
```

In [224...
```
plot_cluster_2d(model_pca_arr,kmeans)
```



In [225...
```
pkl.dump(kmeans,open(PKL_PATH+'/model.pkl','wb'))
with open(PKL_PATH+'/model.pkl','wb') as file:
 pkl.dump(kmeans,file)
```

Define each cluster based on the features.

In [226...
```
song_dat_cluster=song_dat.copy()
```

In [227...
```
song_dat_cluster['cluster']=y_kmeans
```

In [228...
```
song_dat_cluster.to_csv(CSV_PATH+'/song data with cluster.csv')
```

In [229...
```
song_dat_cluster.head()
```

Out[229]:

	name	album	release_date	track_number	id	
0	Intro - Concert Music - Live	Licked Live In NYC	2022-06-10	1	2IEkywL J4ykbhi1yRQvmsT	spotify:track:2IEkywL J
1	Fighting Man - Street Live	Licked Live In NYC	2022-06-10	2	6GVgVJBKkGJoRfarYRvGTU	spotify:track:6GVgVJBK
2	Start Me Up - Live	Licked Live In NYC	2022-06-10	3	1Lu761pZ0dBTGpzxaQoZNW	spotify:track:1Lu761pZ0d
3	Can't If You Rock Me - Live	Licked Live In NYC	2022-06-10	4	1agTQzOTUnGNggyckEqiDH	spotify:track:1agTQzOTU
4	Donâ€™t Stop - Live	Licked Live In NYC	2022-06-10	5	7piGJR8YndQBQWVXv6KtQw	spotify:track:7piGJR8Ynd

In [230…

```
px.scatter_3d(song_dat_cluster,x='danceability',y='energy',z='tempo',color='cluster
```

In [231…

```
# popularity mean by cluster
song_dat_cluster.groupby(['cluster']).popularity.mean().sort_values(ascending=False
```

Out[231]: cluster
5 24.555118
0 23.194215
6 22.701613
3 21.774648
2 19.093458
4 18.194595
1 17.742331
7 17.085502
Name: popularity, dtype: float64

In [232…
```
# checking number of songs in each cluster
song_dat_cluster['cluster'].value_counts()
```

Out[232]:7 269
5 254
6 248
0 242
4 185
1 163
3 142
2 107
Name: cluster, dtype: int64

In [233…
```
# Checking which cluster was most popular
song_dat_cluster.groupby('cluster')['popularity'].mean().sort_values(ascending=Fals
```

Out[233]: cluster
5 24.555118
0 23.194215
6 22.701613
3 21.774648
2 19.093458
4 18.194595
1 17.742331
7 17.085502
Name: popularity, dtype: float64

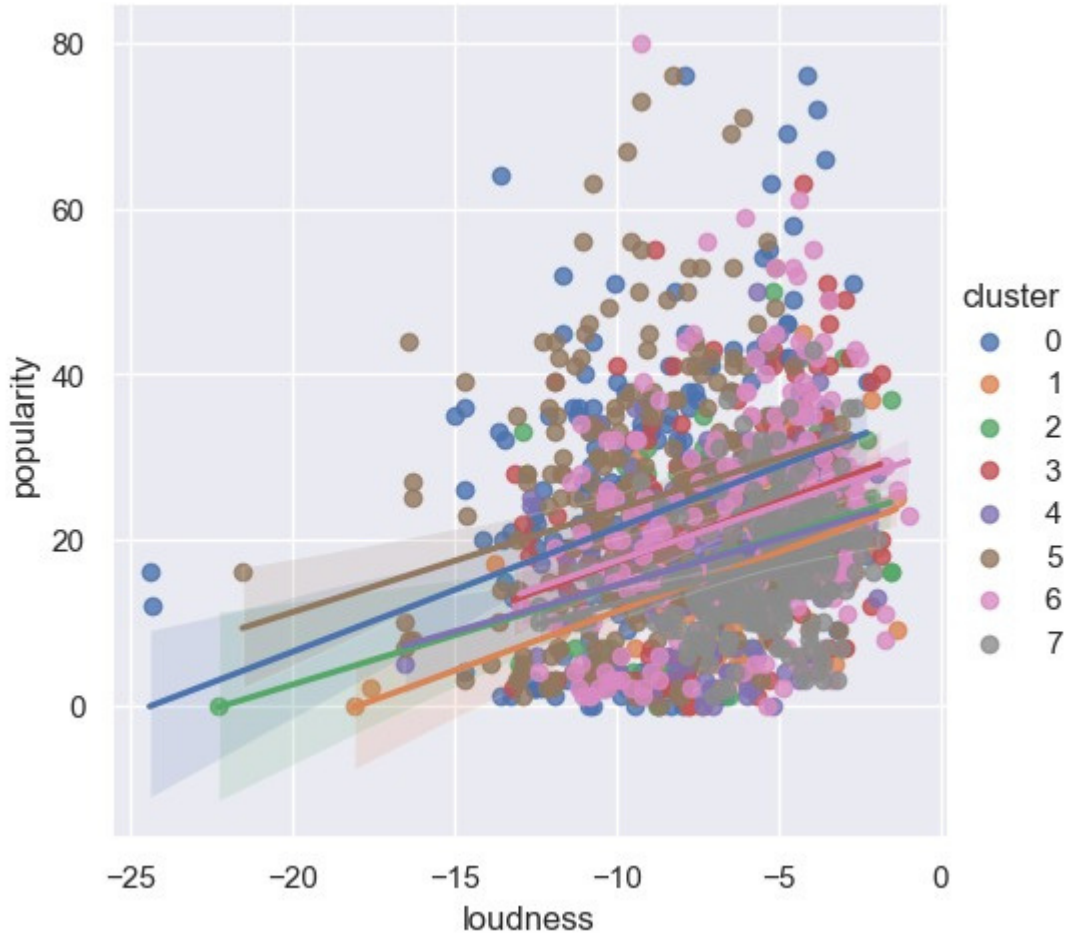
In [234…
```
# Checking the song for cluster 5 as it has the highly most popular song
song_dat_cluster.loc[song_dat_cluster['cluster'] == 5][:10].head()
```

Out[234]:

	name	album	release_date	track_number	id	
52	Worried About You	Tattoo -You Remastered (Super Deluxe)	2021-10-22 2021	7	0Vb7dqiZ89IQngeX52jLaO	spotify:track:0Vb7
54	Heaven -	Tattoo Remastered You 2021 (Super Deluxe)	2021-10-22	9	7f1HoWerlIfVpwgsOu0Mib	spotify:track:7f1Ho
55	No Use In Crying -	Tattoo You Remastered (Super Deluxe)	2021-10-22 2021	10	4hH58AvAjRwnWQnC5IXhzR	spotify:track:4hH58A
61	Drift Away	Tattoo You (Super Deluxe)	2021-10-22	5	100ccVerC10XT80e9qKcNb	spotify:track:100cc
64	Fast Talking, Walking	Tattoo You Slow (Super Deluxe)	2021-10-22	8	1a4BmdKkz56SERLEoWJNc8	spotify:track:1a4Bm

In [235…

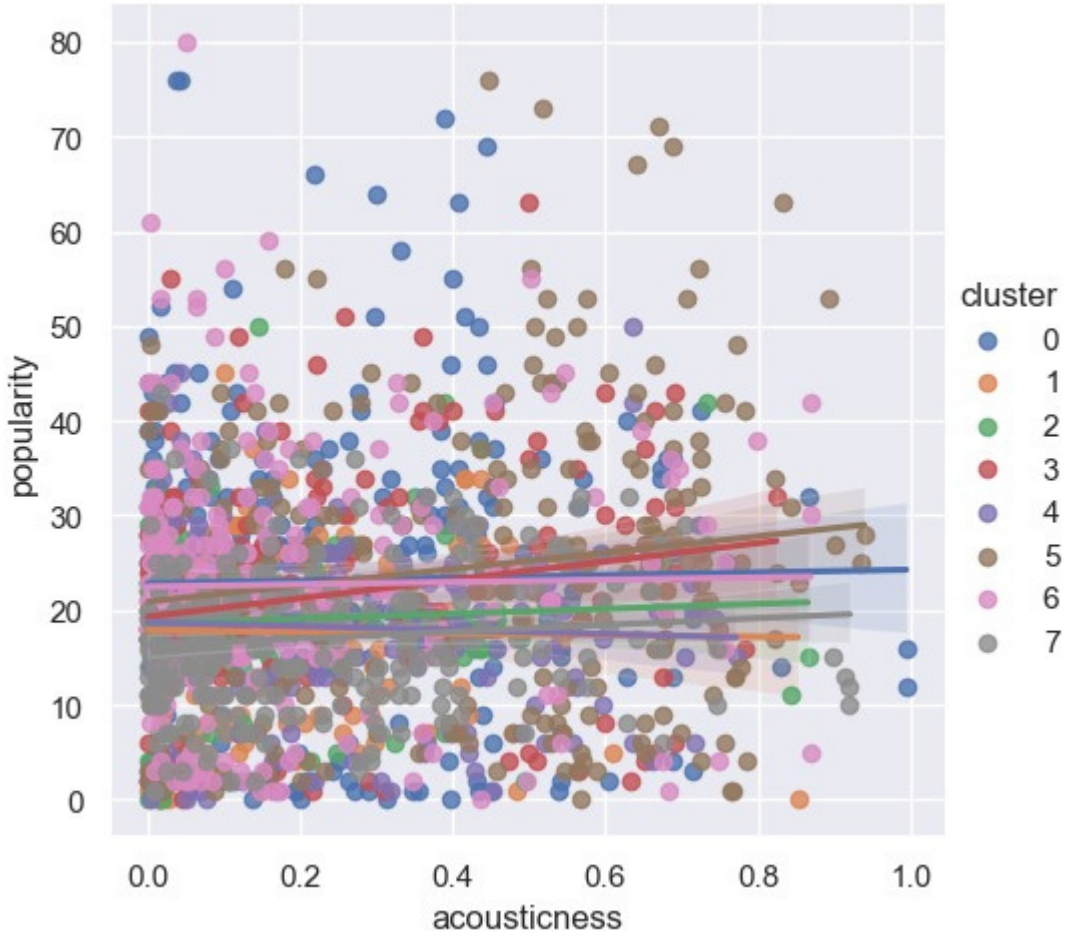
```
# Showing how loudliness affect the popularity
sns.lmplot(data=song_dat_cluster,x='loudness',y='popularity',hue='cluster');
save_plots('Loudness popularity')
plt.show()
```

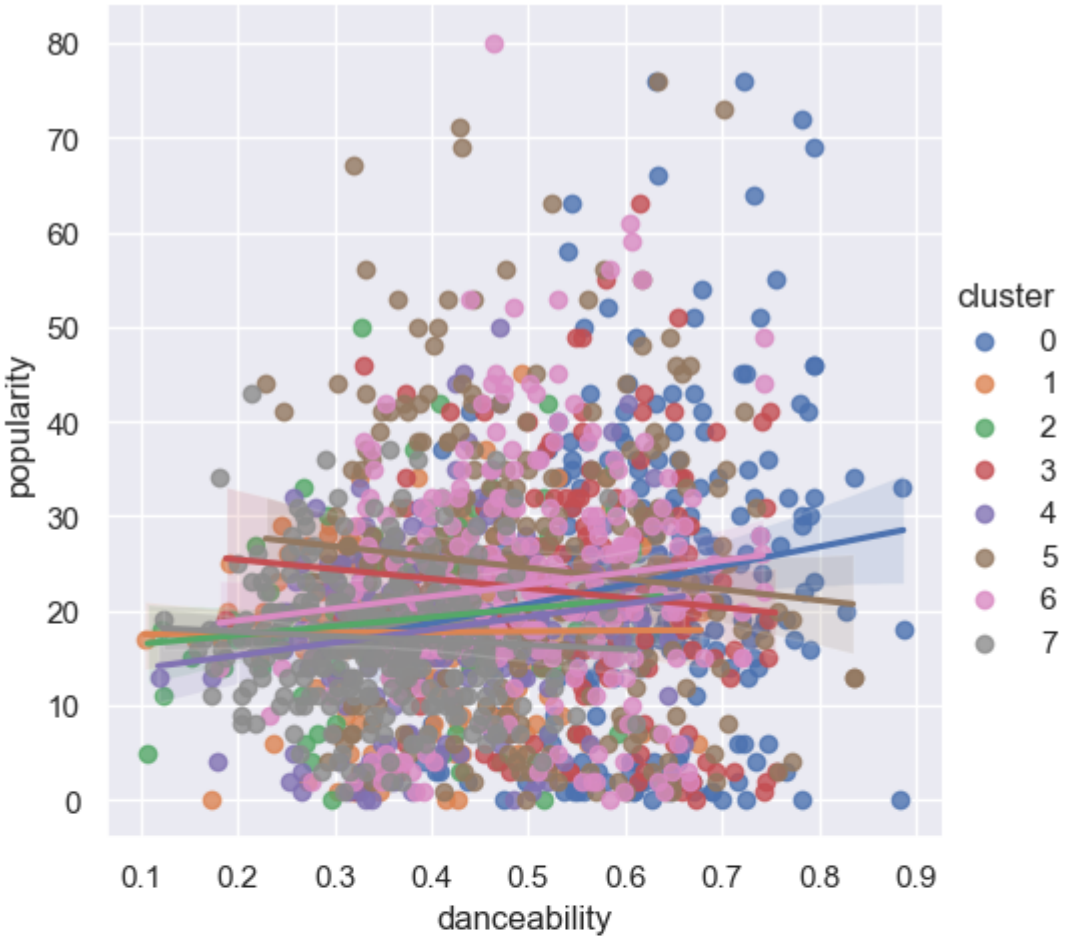


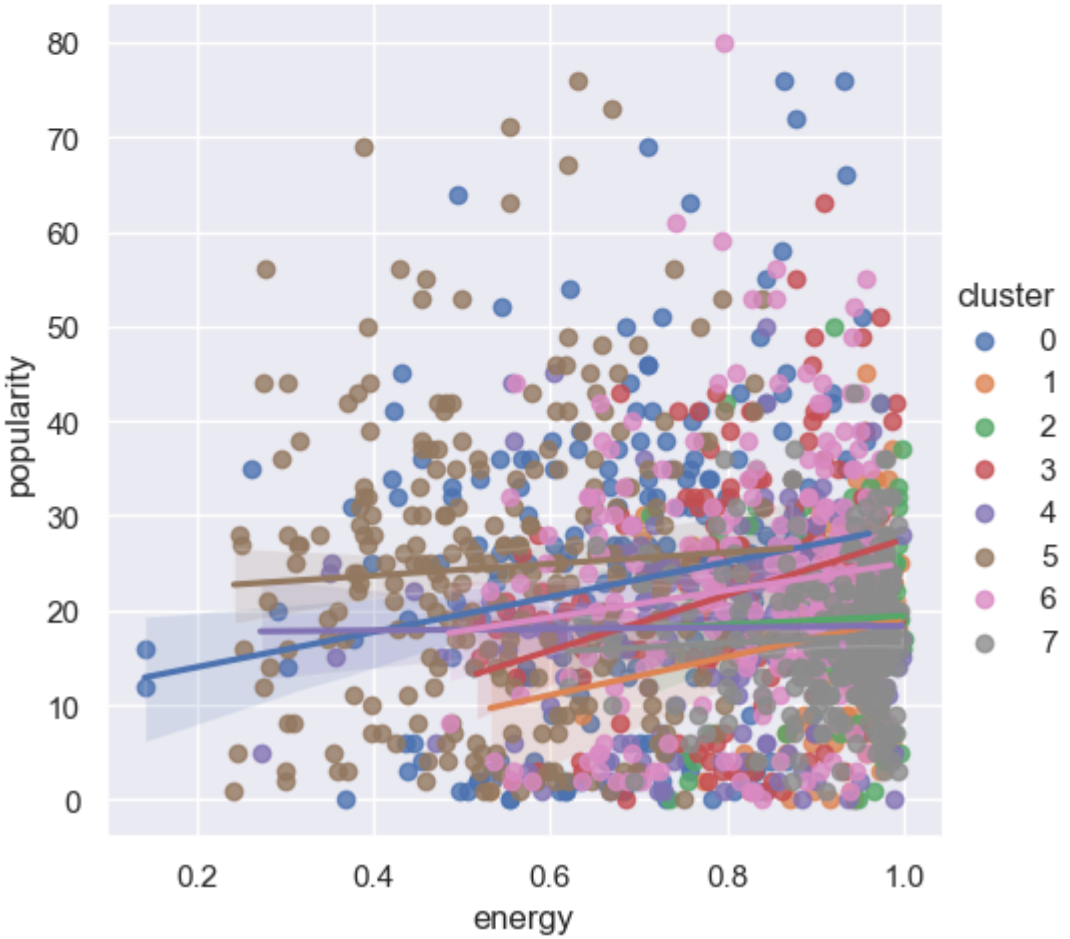
In [236…
```
song_dat_cluster.iloc[:,6:].columns
```

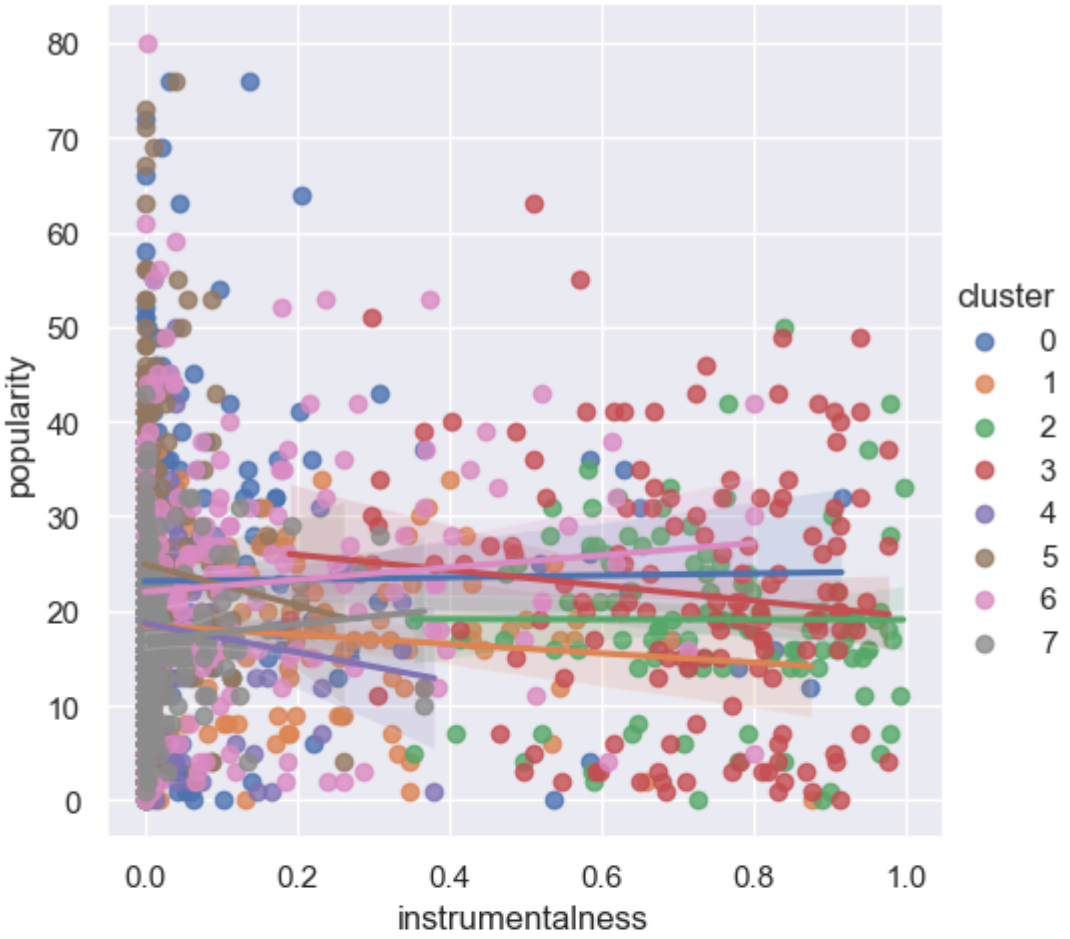
Out[236]:
```
Index(['acousticness', 'danceability', 'energy', 'instrumentalness',
 'liveness', 'loudness', 'speechiness', 'tempo', 'valence', 'popularity',
 'duration_ms', 'cluster'],
 dtype='object')
```

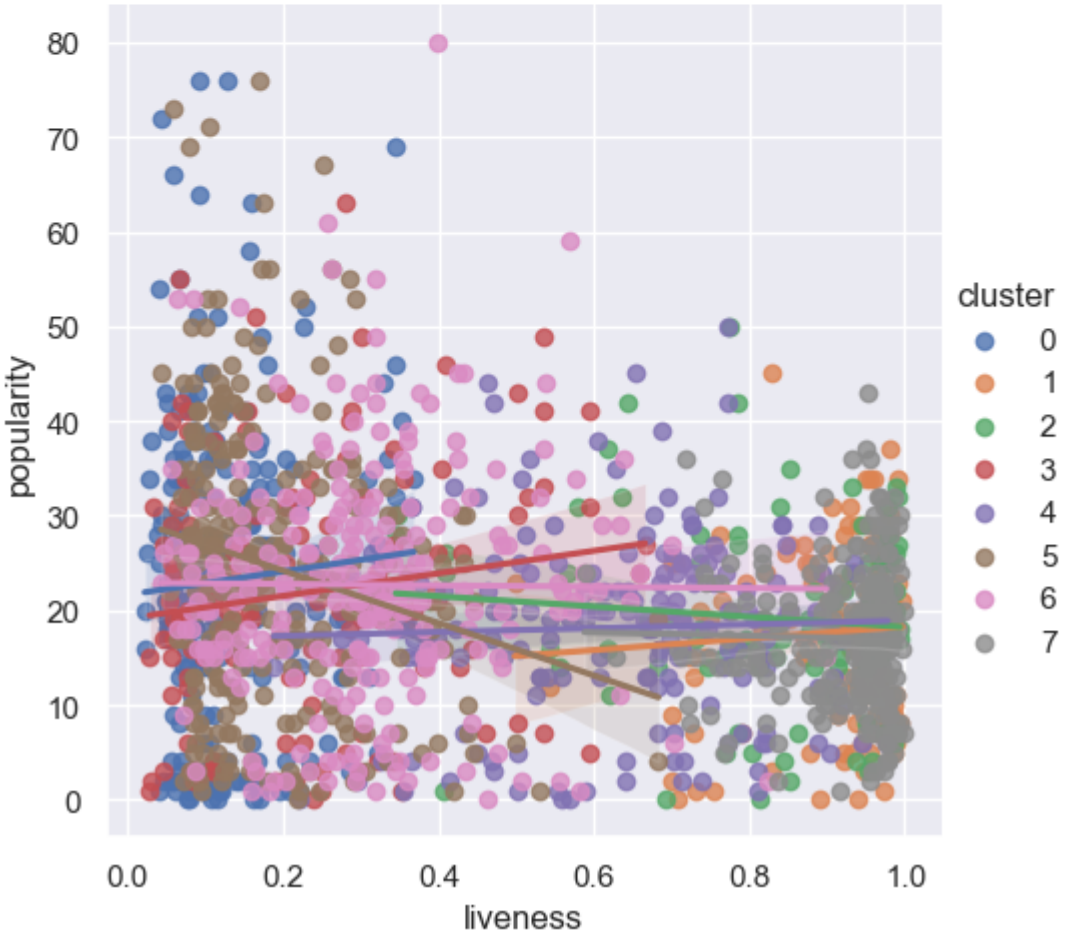
In [237…
```
for col in ['acousticness', 'danceability', 'energy', 'instrumentalness','liveness'
    sns.lmplot(data=song_dat_cluster,x=col,y='popularity',hue='cluster');
 save_plots(f'Lmplots {col} vs popularity')
 plt.show()
plt.tight_layout()
```

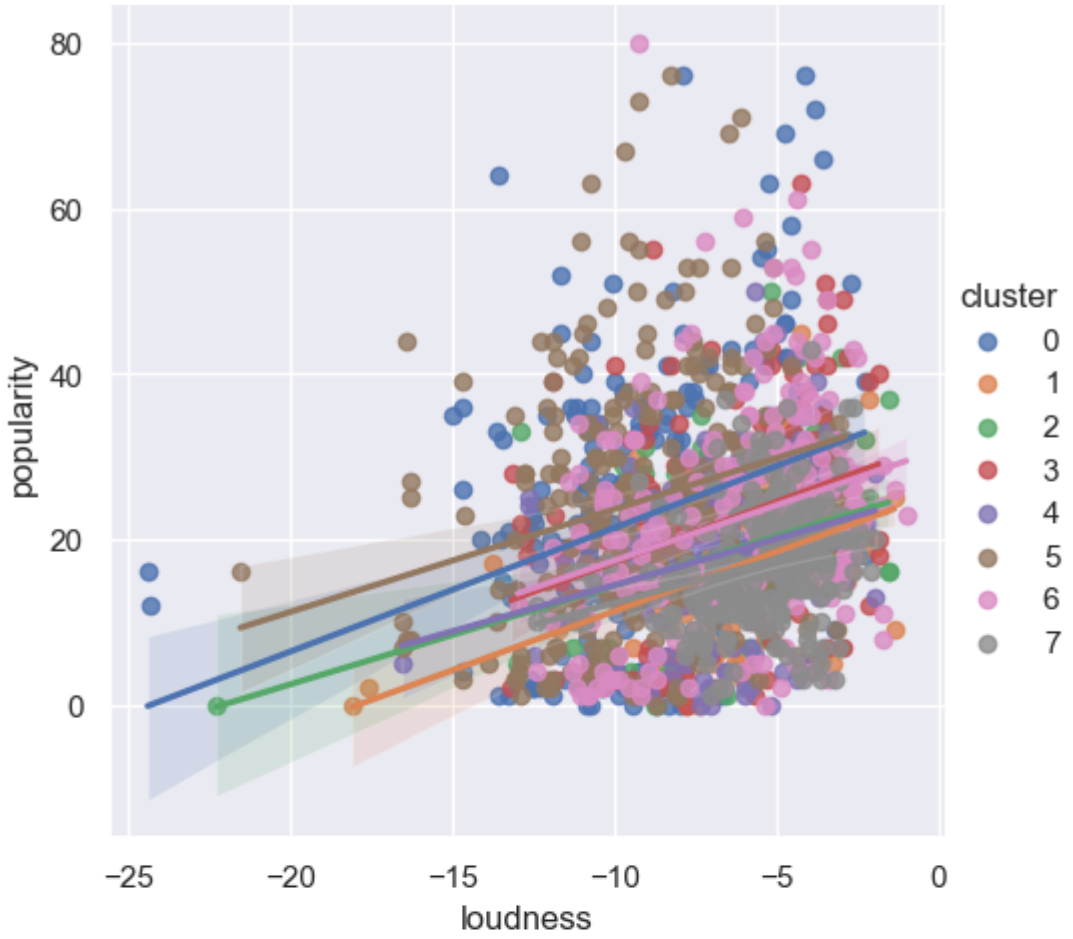


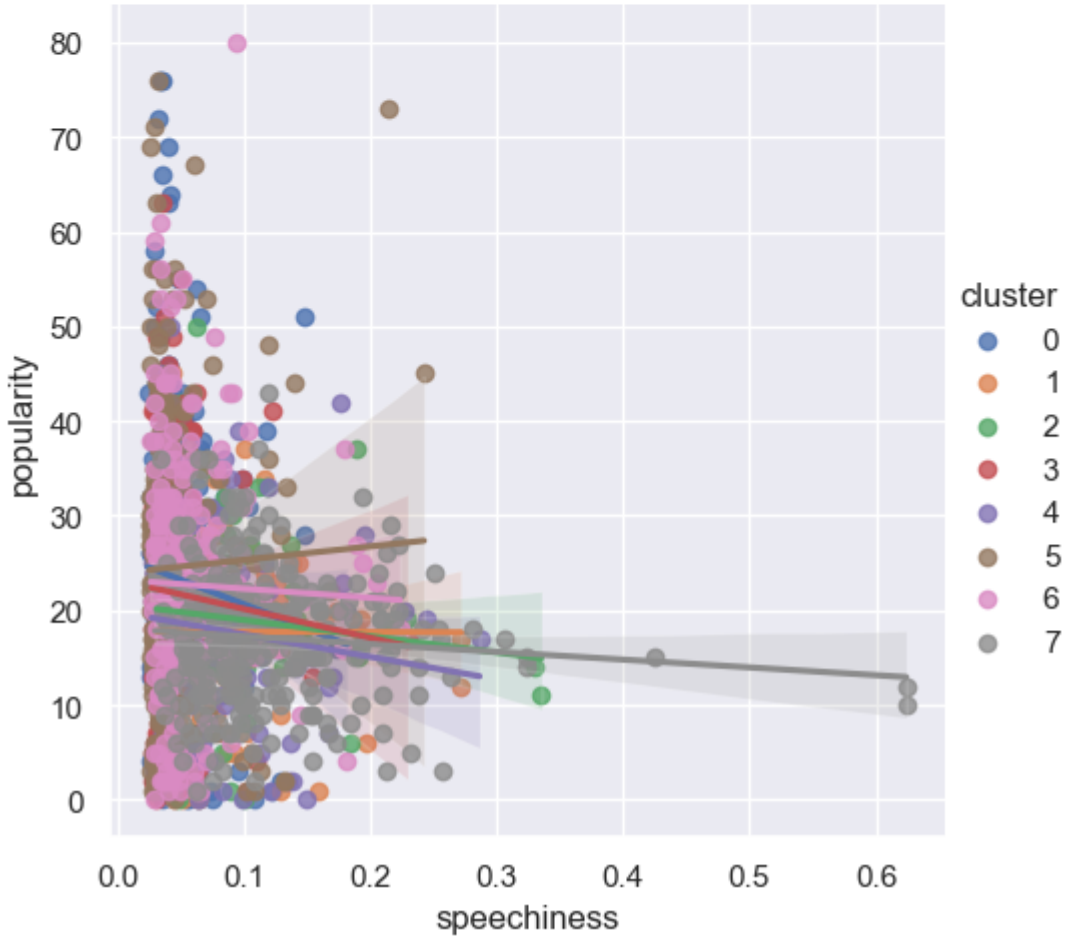

popularity
danceability
cluster
0
1
2
3
4
5
6
7

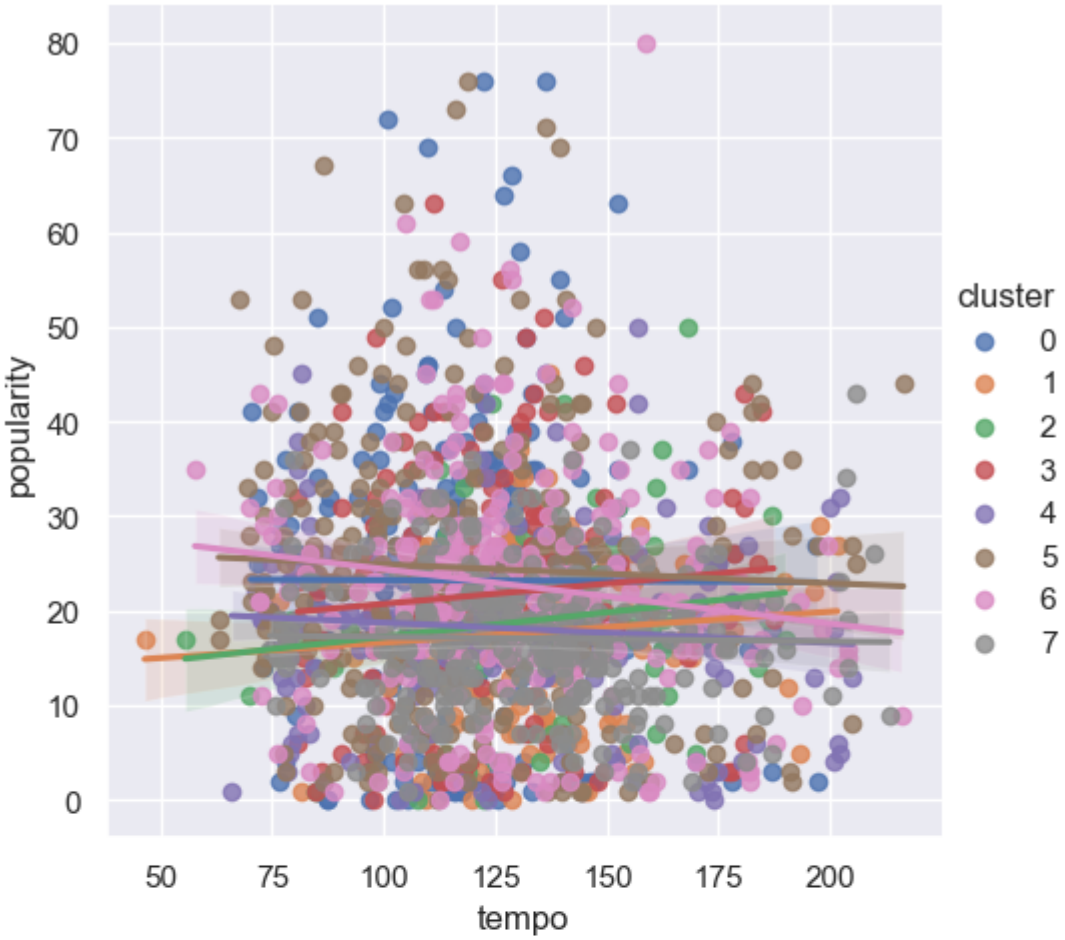

popularity
energy
cluster
0
1
2
3
4
5
6
7

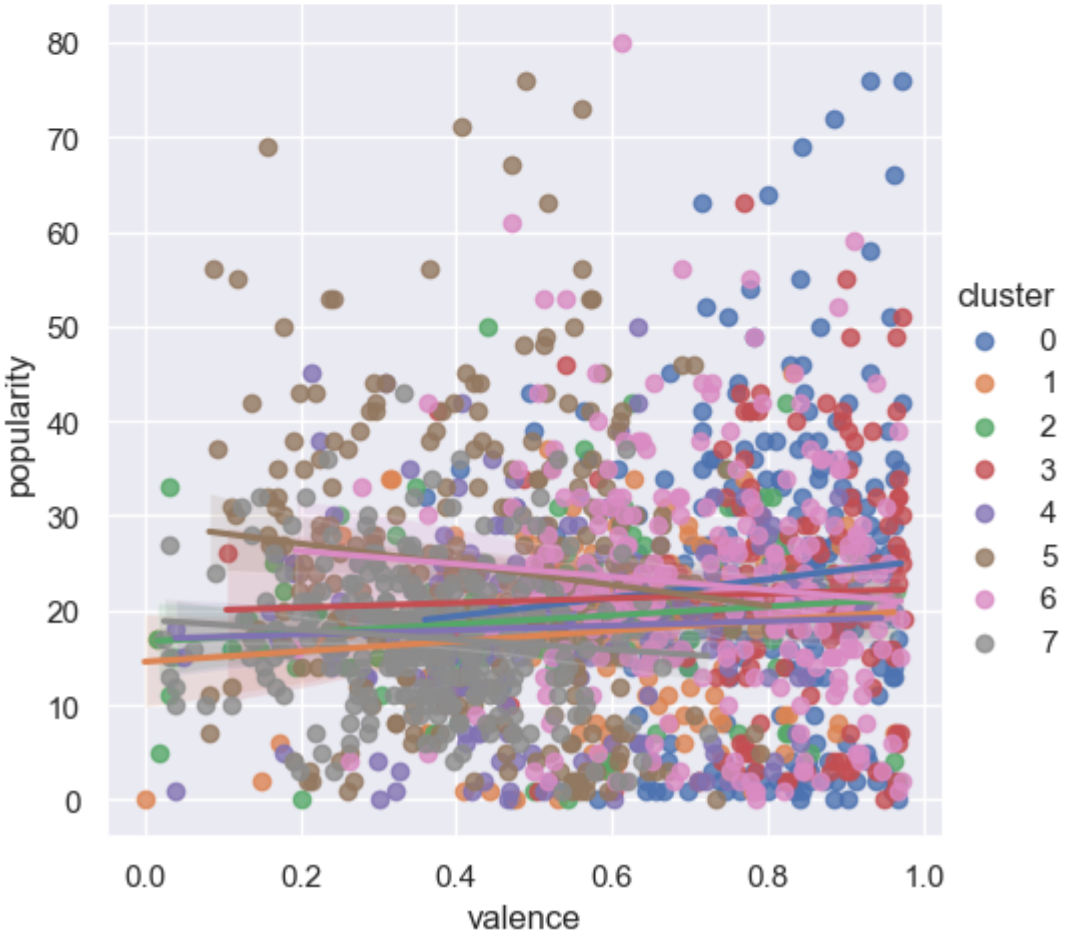

popularity
instrumentalness
cluster
0
1
2
3
4
5
6
7

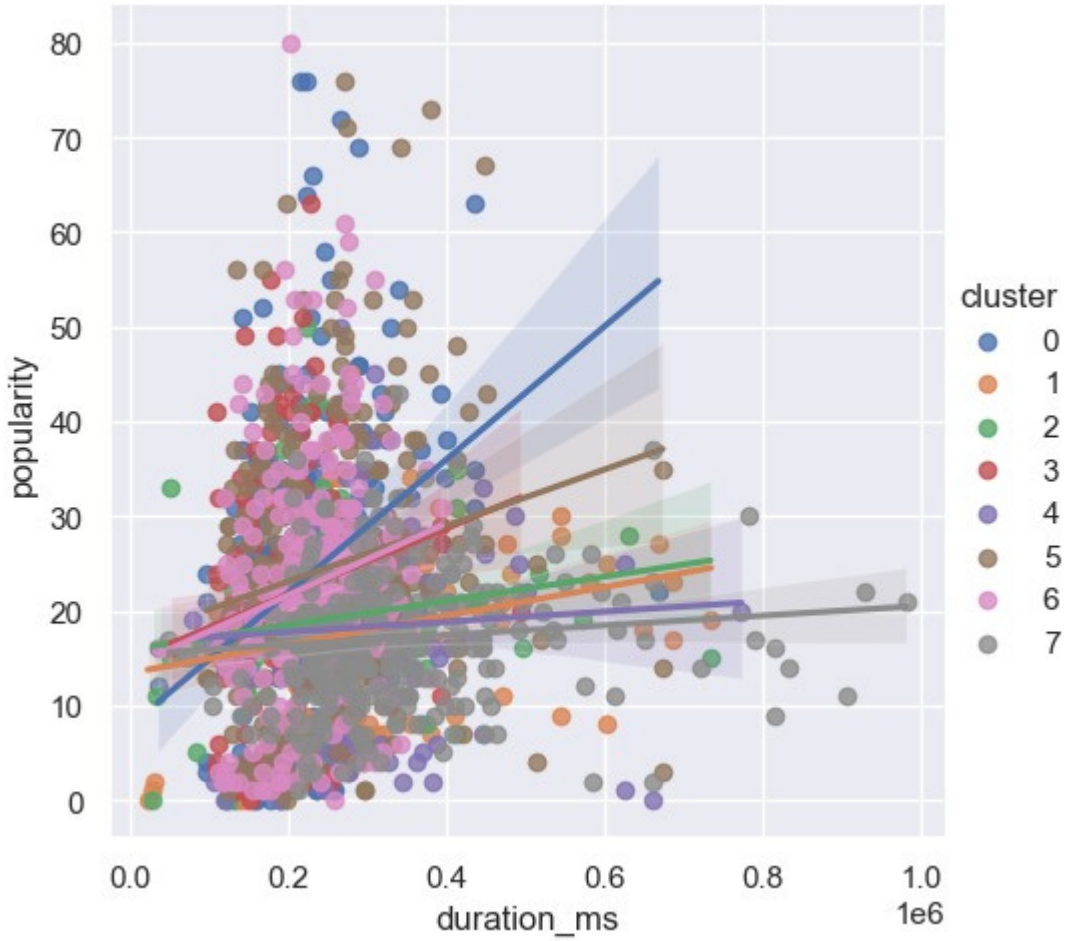

popularity
liveness
cluster
0
1
2
3
4
5
6
7


popularity
loudness
cluster
0
1
2
3
4
5
6
7


popularity
speechiness
cluster
0
1
2
3
4
5
6
7


popularity
tempo
cluster
0
1
2
3
4
5
6
7


popularity
valence
cluster
0
1
2
3
4
5
6
7




```
<Figure size 640x480 with 0 Axes>
```

Finding:

- The song belong to the cluster 5 was the most popular song For
- each cluster there is around 200 songs

Thank You