Analysis on Spotify Dataset

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

1. Imported libraries which will be requiring while doing the analysis.
2. Imported pandas as pd to work on the csv file by converting it into the DataFrame.
3. Imported numpy as np to perform a wide variety of mathematical operations on arrays.
4. Imported matplotlib and seaborn to perform graphical representation of analysis. To make graphs we can use matplotlib and seaborn.

Load the Dataset

In [2]:
```
df2=pd.read_csv('tracks.csv')
df1=pd.read_csv('artists.csv')
```

In [3]:
```
df2.head()
```

Out[3]:

	id	name	popularity	duration_ms	explicit	artists	id_artists	release_date	danceability	energy	key	l
0	35iwgR4jXetI318WEWsa1Q	Carve	6	126903	0	['Uli']	['45tIt06XoI0Iio4LBEVpls']	1922-02-22	0.645	0.4450	0	
1	021ht4sdgPcrDgSk7JTbKY	Capítulo 2.16 - Banquero Anarquista	0	98200	0	['Fernando Pessoa']	['14jtPCOoNZwquk5wd9DxrY']	1922-06-01	0.695	0.2630	0	
2	07A5yehtSnoedViJAZkNnc	Vivo para Quererte - Remasterizado	0	181640	0	['Ignacio Corsini']	['5LiOoJbxVSAMkBS2fUm3X2']	1922-03-21	0.434	0.1770	1	
3	08FmqUhxtyLTn6pAh6bk45	El Prisionero - Remasterizado	0	176907	0	['Ignacio Corsini']	['5LiOoJbxVSAMkBS2fUm3X2']	1922-03-21	0.321	0.0946	7	
4	08y9GfoqCWfOGsKdwojr5e	Lady of the Evening	0	163080	0	['Dick Haymes']	['3BiJGZsyX9sJchTqcSA7Su']	1922	0.402	0.1580	3	

In [4]:
```
df1.head()
```

Out[4]:

	id	followers	genres	name	popularity
0	0DheY5irMjBUeLybbCUEZ2	0.0	[]	Armid & Amir Zare Pashai feat. Sara Rouzbehani	0
1	0DlhY15l3wsrnlfGio2bjU	5.0	[]	ปูนา ภาวิณี	0
2	0DmRESX2JknGPQyO15yxg7	0.0	[]	Sadaa	0
3	0DmhnbHjm1qw6NCYPeZNgJ	0.0	[]	Tra'gruda	0
4	0Dn11fWM7vHQ3rinvWEI4E	2.0	[]	Ioannis Panoutsopoulos	0

In [61]:
```
df2.tail()
```

Out[61]:

	id	name	popularity	duration_ms	explicit	artists	id_artists	release_date	danceability	energy
586667	5rgu12WBIHQtvej2MdHSH0	云与海	50	258267	0	['阿YueYue']	['1QLBXKM5GCpyQQSVMNZqrZ']	2020-09-26	0.560	0.518
586668	0NuWgxEp51CutD2pJoF4OM	blind	72	153293	0	['ROLE MODEL']	['1dy5WNgIKQU6ezkpZs4y8z']	2020-10-21	0.765	0.663
586669	27Y1N4Q4U3EfDU5Ubw8ws2	What They'll Say About Us	70	187601	0	['FINNEAS']	['37M5pPGs6V1fchFJSgCguX']	2020-09-02	0.535	0.314
586670	45XJsGpFTyzbzeWK8VzR8S	A Day At A Time	58	142003	0	['Gentle Bones', 'Clara Benin']	['4jGPdu95icCKVF31CcFKbS', '5ebPSE9YI5aLeZ1Z2g...	2021-03-05	0.696	0.615
586671	5Ocn6dZ3BJFPWh4ylwFXtn	Mar de Emociones	38	214360	0	['Afrosound']	['0i4Qda0k4nf7jnNHmSNpYv']	2015-07-01	0.686	0.723

5 rows × 21 columns

1. read_csv() function of pandas reads the csv file.
2. head() function gives first 5 records from the given dataset.
3. tail() function gives last 5 records from the given dataset.

In [5]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 586672 entries, 0 to 586671
Data columns (total 20 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   id                586672 non-null  object 
 1   name              586601 non-null  object 
 2   popularity        586672 non-null  int64  
 3   duration_ms       586672 non-null  int64  
 4   explicit          586672 non-null  int64  
 5   artists           586672 non-null  object 
 6   id_artists        586672 non-null  object 
 7   release_date      586672 non-null  object 
 8   danceability      586672 non-null  float64
 9   energy            586672 non-null  float64
 10  key               586672 non-null  int64  
 11  loudness          586672 non-null  float64
 12  mode              586672 non-null  int64  
 13  speechiness       586672 non-null  float64
 14  acousticness      586672 non-null  float64
 15  instrumentalness  586672 non-null  float64
 16  liveness          586672 non-null  float64
 17  valence           586672 non-null  float64
 18  tempo             586672 non-null  float64
 19  time_signature    586672 non-null  int64  
dtypes: float64(9), int64(6), object(5)
memory usage: 89.5+ MB
```

In [6]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1104349 entries, 0 to 1104348
Data columns (total 5 columns):
 #   Column      Non-Null Count    Dtype  
---  ------      --------------    -----  
 0   id          1104349 non-null  object 
 1   followers   1104336 non-null  float64
 2   genres      1104349 non-null  object 
 3   name        1104349 non-null  object 
 4   popularity  1104349 non-null  int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 42.1+ MB
```

1. There are 586672 entries in tracks dataset.
2. Total 20 columns are there in the dataset.
3. info() function gives the information about the dataset such as column counts, entries, data types, etc.

In [7]:
```
df2.columns
```

Out[7]:
```
Index(['id', 'name', 'popularity', 'duration_ms', 'explicit', 'artists',
       'id_artists', 'release_date', 'danceability', 'energy', 'key',
       'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness',
       'liveness', 'valence', 'tempo', 'time_signature'],
      dtype='object')
```

We will look at the columns:

Release date : This will indicate that on what date the song gad released.

Popularity : This will indicate the popularity of a song.

Acoustics: A confidence measure from 0.0 to 1.0 of whether the track is acoustic.1.0 represents high confidence the track is acoustic.

Danceability: A Danceability measure from 0.0 to 1.0 describes how suitable a track is for dancing.

Energy: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.

Instrumentalness: It predicts whether a track contains no vocals. The closer the instrumentals value is to 1.0

Liveness: Detects the presence of an audience in the recording. A value above 0.8 provides a strong likelihood that the track is live.

Loudness: The overall loudness of a track in decibels(dB)

Mode: It indicates the modality(major or minor) of the song. 1.0 represents major mode and 0 represents minor.

Speechiness: Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audiobook, poetry), the closer to 1.0 the attribute value.

Tempo: The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, the tempo is the speed or pace of a given piece and derives directly from the average beat duration.

Time_signature: An estimated overall time signature of a track.

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

Name: The name of the song.

Artists: The singer of the song.

In [8]:
```
df2.drop(columns=['duration_ms','key'])  #removing the columns that are not required.
```

Out[8]:

	id	name	popularity	explicit	artists	id_artists	release_date	danceability	energy	loudness
0	35iwgR4jXetI318WEWsa1Q	Carve	6	0	['Uli']	['45tIt06XoI0Iio4LBEVpls']	1922-02-22	0.645	0.4450	-13.338
1	021ht4sdgPcrDgSk7JTbKY	Capítulo 2.16 - Banquero Anarquista	0	0	['Fernando Pessoa']	['14jtPCOoNZwquk5wd9DxrY']	1922-06-01	0.695	0.2630	-22.136
2	07A5yehtSnoedViJAZkNnc	Vivo para Quererte - Remasterizado	0	0	['Ignacio Corsini']	['5LiOoJbxVSAMkBS2fUm3X2']	1922-03-21	0.434	0.1770	-21.180
3	08FmqUhxtyLTn6pAh6bk45	El Prisionero - Remasterizado	0	0	['Ignacio Corsini']	['5LiOoJbxVSAMkBS2fUm3X2']	1922-03-21	0.321	0.0946	-27.961
4	08y9GfoqCWfOGsKdwojr5e	Lady of the Evening	0	0	['Dick Haymes']	['3BiJGZsyX9sJchTqcSA7Su']	1922	0.402	0.1580	-16.900
...	...	...	...	...	...	...	...	...	...	..
586667	5rgu12WBIHQtvej2MdHSH0	云与海	50	0	['阿YueYue']	['1QLBXKM5GCpyQQSVMNZqrZ']	2020-09-26	0.560	0.5180	-7.471
586668	0NuWgxEp51CutD2pJoF4OM	blind	72	0	['ROLE MODEL']	['1dy5WNgIKQU6ezkpZs4y8z']	2020-10-21	0.765	0.6630	-5.223
586669	27Y1N4Q4U3EfDU5Ubw8ws2	What They'll Say About Us	70	0	['FINNEAS']	['37M5pPGs6V1fchFJSgCguX']	2020-09-02	0.535	0.3140	-12.823
586670	45XJsGpFTyzbzeWK8VzR8S	A Day At A Time	58	0	['Gentle Bones', 'Clara Benin']	['4jGPdu95icCKVF31CcFKbS', '5ebPSE9YI5aLeZ1Z2g...	2021-03-05	0.696	0.6150	-6.212
586671	5Ocn6dZ3BJFPWh4ylwFXtn	Mar de Emociones	38	0	['Afrosound']	['0i4Qda0k4nf7jnNHmSNpYv']	2015-07-01	0.686	0.7230	-7.067

586672 rows × 18 columns

In [9]:
```
df1.shape
```

Out[9]: (1104349, 5)

In [10]:
```
df2.shape
```

Out[10]: (586672, 20)

1. The output for the above code is (1104349,5) and (586672, 20) respectively.
2. So there are 1104349 rows in the artists data and 586672 rows in the tracks data.

In [12]:
```
df1_new = df1.drop_duplicates()  #dropping the duplicate values from df1.
```

In [13]:
```
df1_new.shape
```

Out[13]: (1104349, 5)

In [14]:
```
df2_new = df2.drop_duplicates() #dropping the duplicate values from df2
```

In [15]:
```
df2_new.shape
```

Out[15]: (586672, 20)

Checked for duplicate values. There are no duplicate values in both the dataset.

In [17]:
```
df2_new.isnull().sum()
```

Out[17]:
```
id                  0
name               71
popularity          0
duration_ms         0
explicit            0
artists             0
id_artists          0
release_date        0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
time_signature      0
dtype: int64
```

In [18]:
```
df2=df2_new.dropna()
df2.head()
```

Out[18]:

	id	name	popularity	duration_ms	explicit	artists	id_artists	release_date	danceability	energy	key	l
0	35iwgR4jXetI318WEWsa1Q	Carve	6	126903	0	['Uli']	['45tIt06XoI0Iio4LBEVpls']	1922-02-22	0.645	0.4450	0	
1	021ht4sdgPcrDgSk7JTbKY	Capítulo 2.16 - Banquero Anarquista	0	98200	0	['Fernando Pessoa']	['14jtPCOoNZwquk5wd9DxrY']	1922-06-01	0.695	0.2630	0	
2	07A5yehtSnoedViJAZkNnc	Vivo para Quererte - Remasterizado	0	181640	0	['Ignacio Corsini']	['5LiOoJbxVSAMkBS2fUm3X2']	1922-03-21	0.434	0.1770	1	
3	08FmqUhxtyLTn6pAh6bk45	El Prisionero - Remasterizado	0	176907	0	['Ignacio Corsini']	['5LiOoJbxVSAMkBS2fUm3X2']	1922-03-21	0.321	0.0946	7	
4	08y9GfoqCWfOGsKdwojr5e	Lady of the Evening	0	163080	0	['Dick Haymes']	['3BiJGZsyX9sJchTqcSA7Su']	1922	0.402	0.1580	3	

In [19]:
```
df2.isnull().sum()
```

Out[19]:
```
id                  0
name                0
popularity          0
duration_ms         0
explicit            0
artists             0
id_artists          0
release_date        0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
time_signature      0
dtype: int64
```

In [20]:
```
df1_new.isnull().sum()
```

Out[20]:
```
id            0
followers    13
genres        0
name          0
popularity    0
dtype: int64
```

In [21]:
```
df1=df1_new.dropna()
df1.head()
```

Out[21]:

	id	followers	genres	name	popularity
0	0DheY5irMjBUeLybbCUEZ2	0.0	[]	Armid & Amir Zare Pashai feat. Sara Rouzbehani	0
1	0DlhY15l3wsrnlfGio2bjU	5.0	[]	ปูนา ภาวิณี	0
2	0DmRESX2JknGPQyO15yxg7	0.0	[]	Sadaa	0
3	0DmhnbHjm1qw6NCYPeZNgJ	0.0	[]	Tra'gruda	0
4	0Dn11fWM7vHQ3rinvWEl4E	2.0	[]	Ioannis Panoutsopoulos	0

In [22]:
```
df1.isnull().sum()
```

Out[22]:
```
id            0
followers     0
genres        0
name          0
popularity    0
dtype: int64
```

1. Empty cells are the most common and anticipated data cleaning problem.
2. So after eliminating the duplicate rows, checking for any missing data points is the most important step.
3. In python dataframe, check for empty cells can be performed using isnull() , further using the sum() function to get an overall sum of the empty values.

Data Analysis

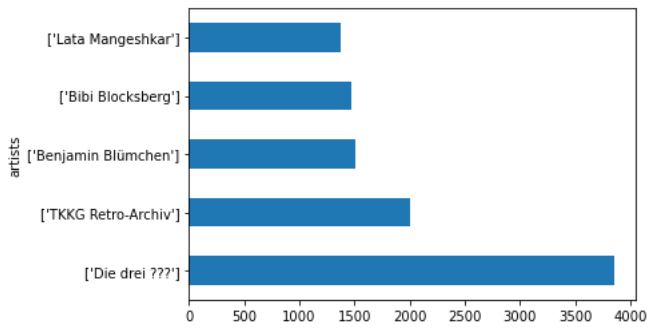
Analysis of a data is defined as a process of inspecting, cleaning, transforming and modeling data with the goal of discovering useful information, informing conclusions and supporting decision-making.

1.Top 5 most popular artists.

In [25]:
```
top_five_artists=df2.groupby('artists').count().sort_values('name',ascending=False)['name'][:5]
top_five_artists
```

Out[25]:
```
artists
['Die drei ???']         3856
['TKKG Retro-Archiv']    2006
['Benjamin Blümchen']    1503
['Bibi Blocksberg']      1472
['Lata Mangeshkar']      1373
Name: name, dtype: int64
```

In [26]:
```
top_five_artists.plot.barh()
plt.show()
```



From above graph it shows that, 'Die drei' is the most popular artist. Thses are the top 5 most popular artist.

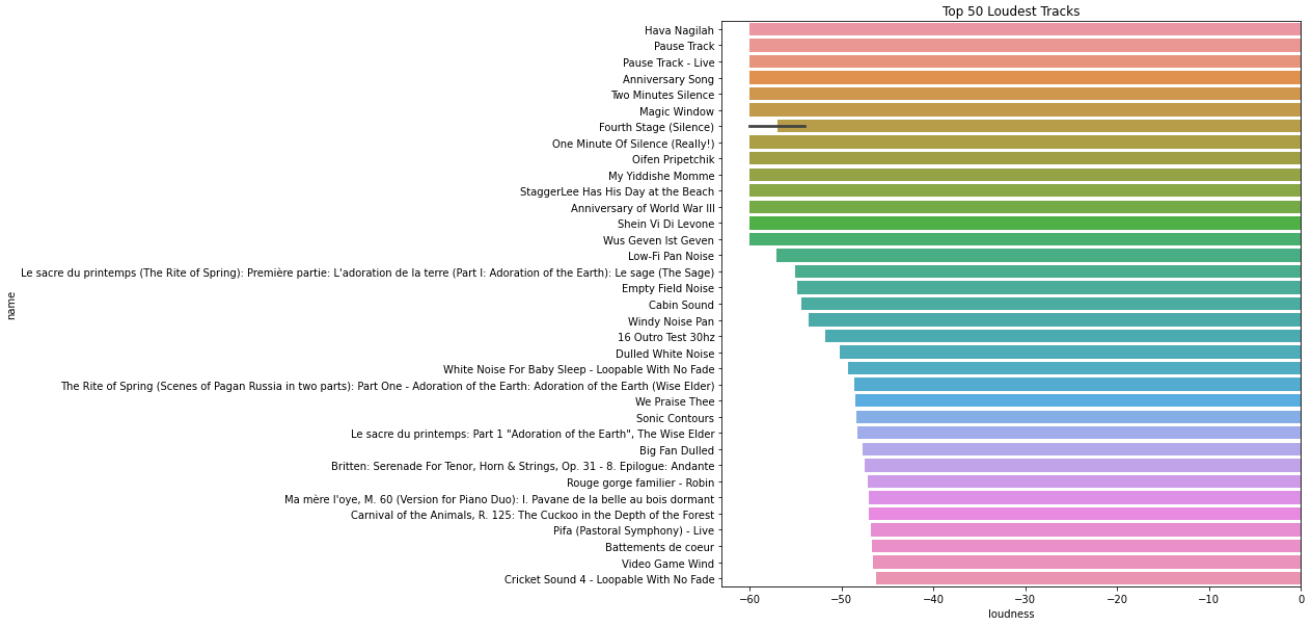
2.Top 50 loudest track

In [28]:
```
top_f_ltracks=df2[['loudness','name']].sort_values(by='loudness',ascending=True)[:50]
top_f_ltracks.head()
#values between -60 and 0db(decibels) ---> -60 is the most loudest value.
```

Out[28]:

	loudness	name
382566	-60.0	Hava Nagilah
1580	-60.0	Pause Track
12627	-60.0	Pause Track - Live
1575	-60.0	Pause Track
353485	-60.0	Anniversary Song

In [63]:
```
plt.figure(figsize=(10,10))
sns.barplot(x='loudness',y='name',data=top_f_ltracks)
plt.title('Top 50 Loudest Tracks')
plt.show()
```



This graph shows the loudness of top 50 songs with their names.

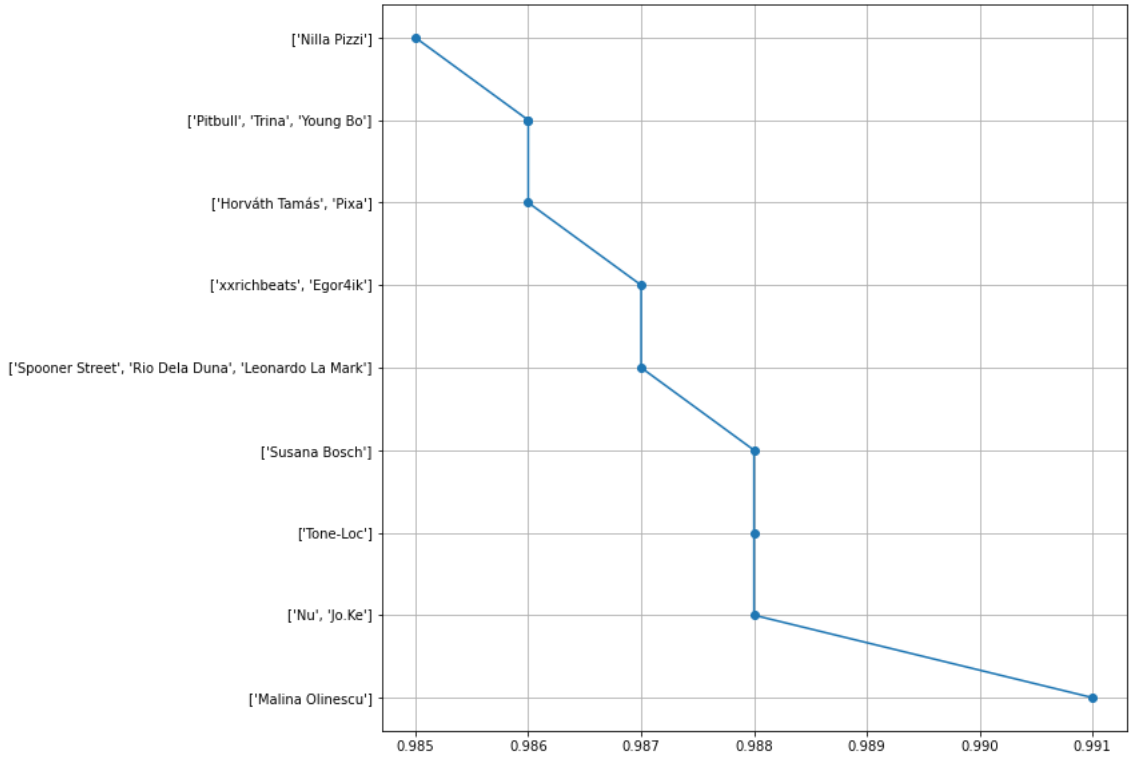
3.Artist with the most danceability song.

In [30]:
```
top_artist_danceable_songs=df2[['danceability','name','artists']].sort_values(by='danceability',ascending=False)[:10]
top_artist_danceable_songs
```

Out[30]:

	danceability	name	artists
418558	0.991	Puisorul cafeniu	['Malina Olinescu']
156664	0.988	Who Loves The Sun feat. Jo.Ke - Edit	['Nu', 'Jo.Ke']
62569	0.988	Funky Cold Medina	['Tone-Loc']
252256	0.988	Tío Mario	['Susana Bosch']
356102	0.987	Cool - Leonardo La Mark Remix	['Spooner Street', 'Rio Dela Duna', 'Leonardo ...
74928	0.987	New Year (2021)	['xxrichbeats', 'Egor4ik']
509977	0.986	BABÁM	['Horváth Tamás', 'Pixa']
175899	0.986	Go Girl	['Pitbull', 'Trina', 'Young Bo']
303940	0.986	Go Girl	['Pitbull', 'Trina', 'Young Bo']
131805	0.985	O mama mama - Remix 2014	['Nilla Pizzi']

In [31]:
```
plt.figure(figsize=(10,10))
x=top_artist_danceable_songs['danceability']
y=top_artist_danceable_songs['artists']
plt.plot(x,y,marker='o')
plt.grid()
plt.show()
```



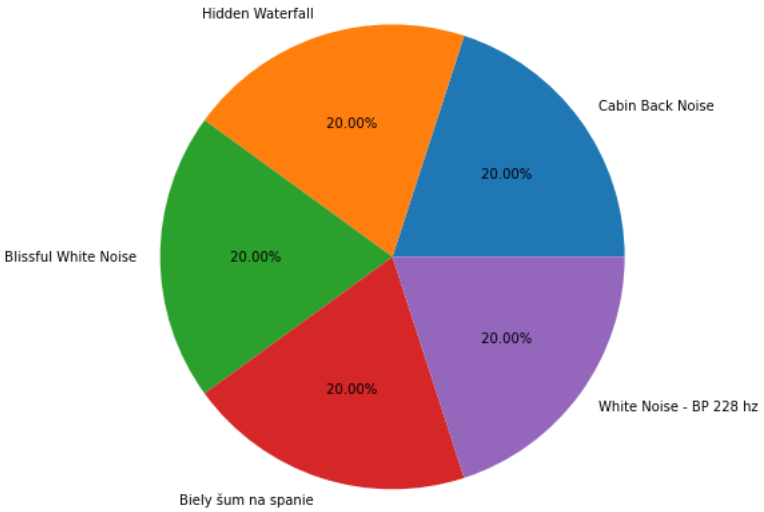
4.Top 5 instrumentalness tracks.

In [35]:
```
top_ten_instrumental_tracks=df2[['instrumentalness','name','artists']].sort_values(by='instrumentalness',ascending=False)[:5]
top_ten_instrumental_tracks
```

Out[35]:

	instrumentalness	name	artists
91647	1.0	Cabin Back Noise	['High Altitude Samples']
586440	1.0	Hidden Waterfall	['White Noise Therapy']
323794	1.0	Blissful White Noise	['Sea of Noise']
323753	1.0	Biely šum na spanie	['Biely Šum']
278997	1.0	White Noise - BP 228 hz	['Granular']

In [36]:
```
plt.figure(figsize=(10,8))
plt.pie(x='instrumentalness',data=top_ten_instrumental_tracks,autopct='% 1.2f%%',labels=top_ten_instrumental_tracks.name)
plt.show()
```



1. This pie chart shows the top 5 instrumentalness tracks with name respectively.
2. We use autopct which give values in percentage, and the syntax is '% 1.2f%%' where % sign at the start and end and the f works as pacemaker i.e we need to put it by deafult and the 1 shows that 1 integer value we want and after that point(.) and 2 number indicates that we want 2 numbers after the decimal point(we can write 3,4 etc how many numbers we want after the point).

5.Top 100 energetic tracks with their loudness and their correlation with mode.

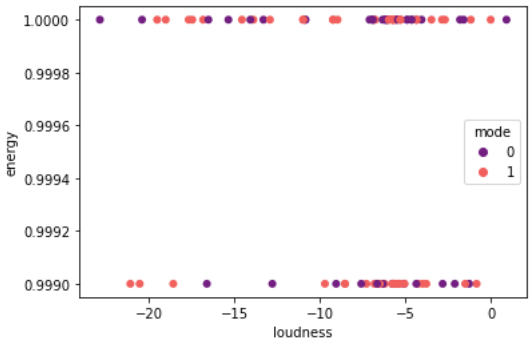
In [37]:
```
top_t_energetic_tracks=df2[['energy','name','artists','loudness','mode']].sort_values(by='energy',ascending=False)[:100]
top_t_energetic_tracks.head()
```

Out[37]:

	energy	name	artists	loudness	mode
91478	1.0	I Feel Love (Mixed) - Omar Sherif Remix	['CRØW', 'Omar Sherif']	-5.595	1
187938	1.0	Rain Sounds: Rain on Big Roof - Loopable	['Rainfall For Sleep']	-10.856	0
68344	1.0	Transilvanian Hunger - Studio	['Darkthrone']	-4.920	0
380632	1.0	Proper Order [Mix Cut] - Original Mix	['Sneijder', 'Bryan Kearney']	-6.102	0
255673	1.0	Soggy Afternoon	['Outside Broadcast Recordings']	-16.847	1

In [38]:
```
sns.scatterplot(x="loudness", y="energy",
                hue="mode",
                palette="magma",
                sizes=(1, 8), linewidth=0,
                data=top_t_energetic_tracks)
```

Out[38]: <AxesSubplot:xlabel='loudness', ylabel='energy'>



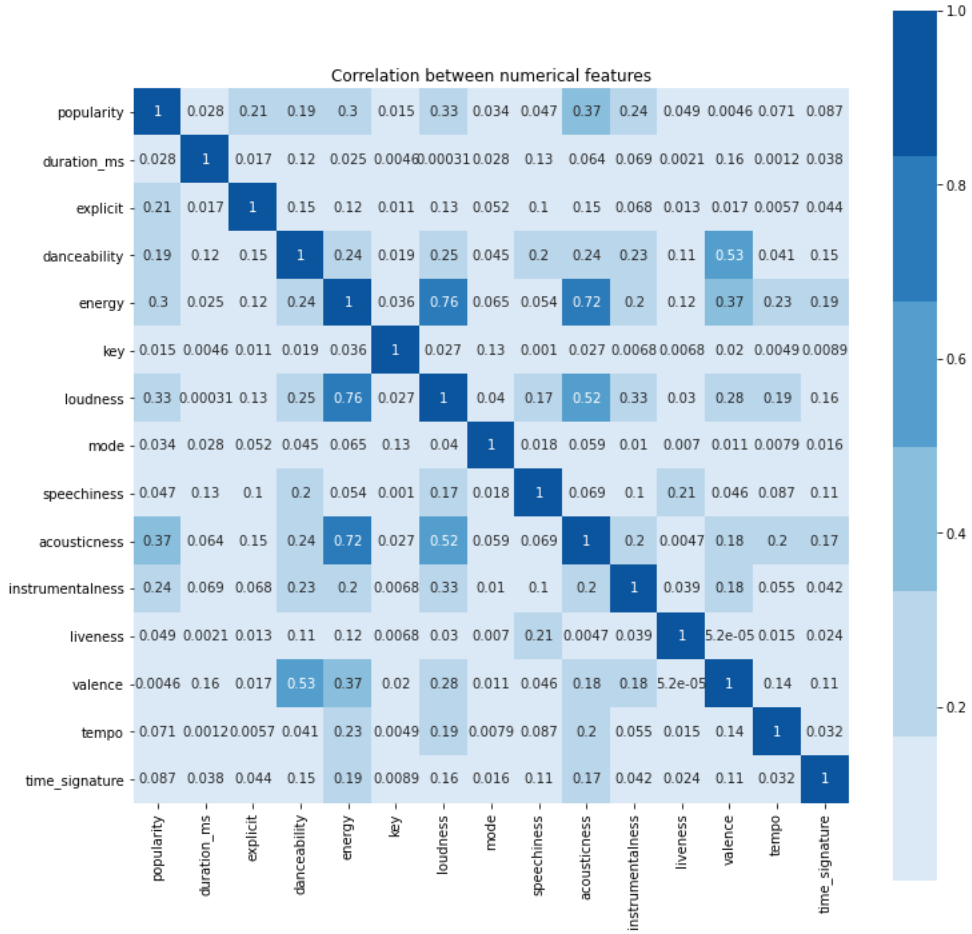
Scatterplot:

1. Scatterplots are also a great way to determine a relationship between two variables.

2. So I will be plotting it for the variables for which we found the correlation.
3. Additionally, I will also add Mode in the plot to get the analysis as per major and minor modes.

6.Finding the correlation between all of the numerical features.

In [40]:
```
numeric_columns = df2.columns[df2.dtypes != 'object']
numeric_df2 = pd.DataFrame(data=df2, columns=numeric_columns, index=df2.index)

corr = np.abs(numeric_df2.corr())
fig, ax = plt.subplots(figsize=(12,12))
cmap = sns.color_palette("Blues")
sns.heatmap(corr, cmap=cmap, square=True, annot=True)
plt.title('Correlation between numerical features')
plt.show()
```



In [ ]:
```
#Heatmap:
#There are many characteristic features of songs in the dataset.
#If there is any relationship between the song features or not can be determined using a heatmap.
#This will help to better understand music peculiarities.
```

In [ ]:
```
#Dark blue color shows there is strong relation between two variables.
#Color between light and dark blue shows the weak relation.
#Light blue color shows there is no relation between two variables.
```

7. Top 10 artists in terms of average energy per song and compare the results with their average acousticness values.

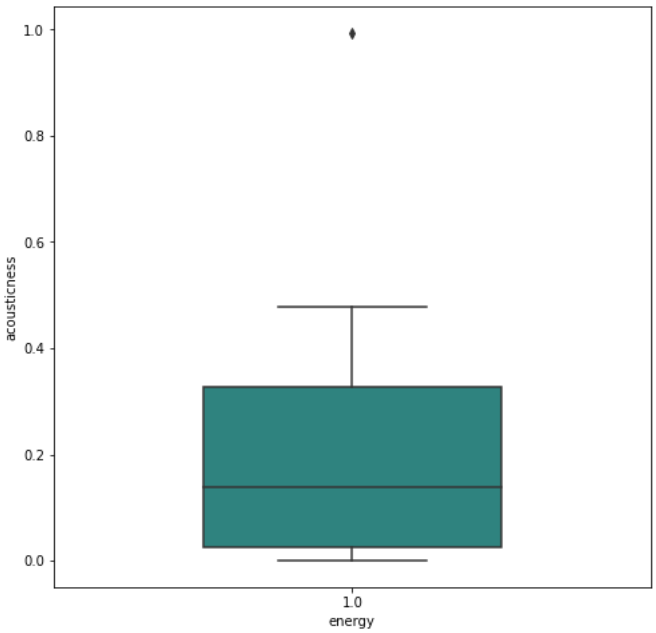
In [41]:
```
a=df2[['artists','energy','acousticness']].groupby('artists').mean().sort_values(by='energy', ascending=False)[:10]
a
```

Out[41]:

artists	energy	acousticness
['DJ Düse']	1.0	0.103000
['The Brick Slayer']	1.0	0.000250
['Outside Broadcast Recordings']	1.0	0.358000
['Komprex']	1.0	0.000443
['Tinnitus']	1.0	0.150000
['White Noise Research']	1.0	0.994000
['Palis']	1.0	0.478000
['Nature Sounds Nature Music']	1.0	0.236000
['Epic Soundscapes']	1.0	0.128000
['Starpicker', 'Ray Reverse']	1.0	0.000304

In [42]:
```
plt.figure(figsize=(8,8))
sns.boxplot(data=a,x='energy',y='acousticness',palette='viridis',width=0.5)
plt.plot()
```

Out[42]: []



1. Boxplot is also used for detect the outlier in data set.
2. Boxplot summarizes a sample data using 25th, 50th and 75th percentiles.
3. These percentiles are also known as the lower quartile, median and upper quartile.
4. There is one outlier in the above graph.
5. Outlier is defined as the data point which is extremely different from the other data points.

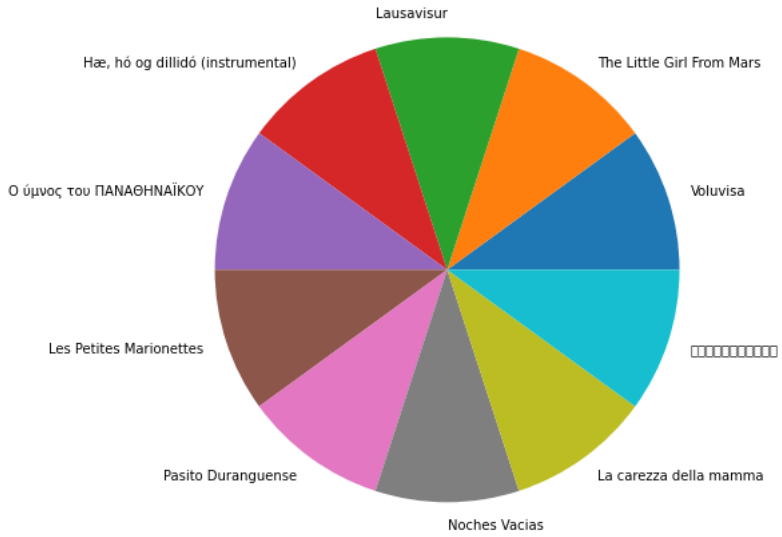
8.Top 10 tracks with the most valence.

In [44]:
```
top_valence=df2[['valence','name']].sort_values(by='valence',ascending=False)[:10]
top_valence
```

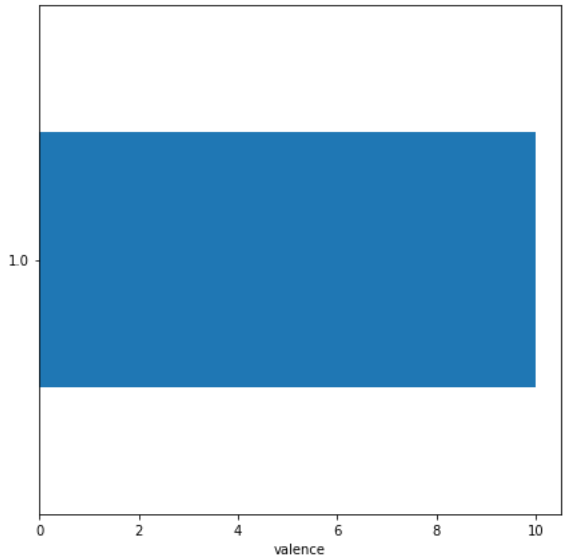
Out[44]:

	valence	name
322066	1.0	Voluvisa
541260	1.0	The Little Girl From Mars
322093	1.0	Lausavisur
320551	1.0	Hæ, hó og dillidó (instrumental)
288680	1.0	Ο ύμνος του ΠΑΝΑΘΗΝΑΪΚΟΥ
51030	1.0	Les Petites Marionettes
236547	1.0	Pasito Duranguense
206971	1.0	Noches Vacias
397802	1.0	La carezza della mamma
561584	1.0	やさしさに包まれたなら

In [45]:
```
plt.figure(figsize=(10,8))
plt.pie(x='valence',data=top_valence,labels=top_valence.name)
plt.show()
```



In [50]:
```
plt.figure(figsize=(7,7))
top_valence['valence'].value_counts().plot(kind='barh')
plt.xlabel('valence')
plt.show()
```



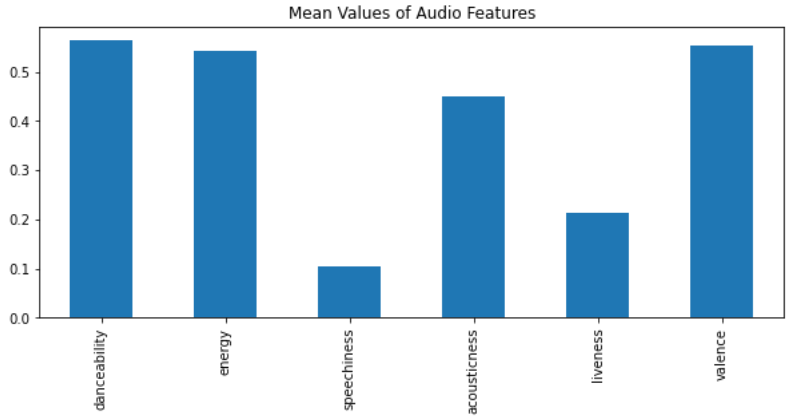
9. Mean values of audio features.

In [51]:
```
small = df2[['danceability', 'energy', 'speechiness', 'acousticness', 'liveness', 'valence']]
small.head()
```

Out[51]:

	danceability	energy	speechiness	acousticness	liveness	valence
0	0.645	0.4450	0.4510	0.674	0.151	0.127
1	0.695	0.2630	0.9570	0.797	0.148	0.655
2	0.434	0.1770	0.0512	0.994	0.212	0.457
3	0.321	0.0946	0.0504	0.995	0.104	0.397
4	0.402	0.1580	0.0390	0.989	0.311	0.196

In [52]:
```
plt.figure(figsize=(10,4))
small.mean().plot.bar()
plt.title('Mean Values of Audio Features')
plt.show()
```



1. Bar graph represents the data using bars either in Horizontal or Vertical directions.
2. Bar graphs are used to show two or more values and typically the x-axis should be categorical data.
3. Use for comparing different groups.
4. It is a bivariate graph.

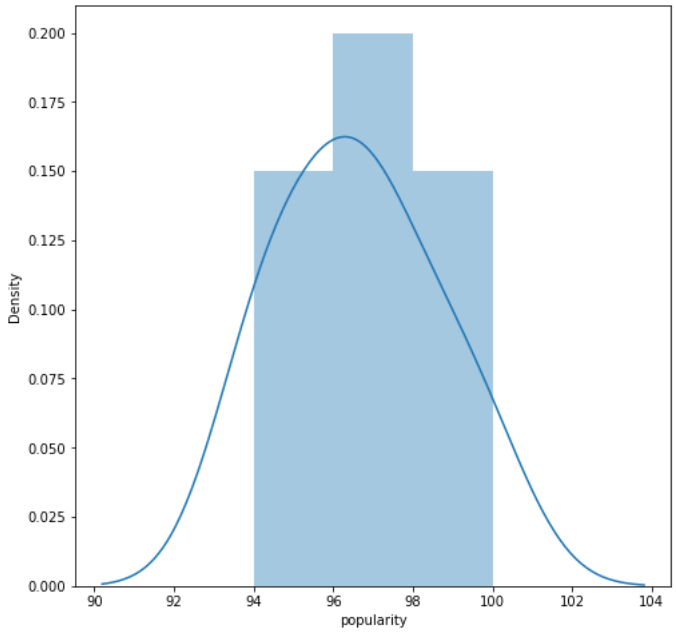
10.10 Most Popular Songs

In [57]:
```
most_popular = df2.query('popularity>90', inplace=False).sort_values('popularity', ascending=False)[:10]
most_popular
```

Out[57]:

	id	name	popularity	duration_ms	explicit	artists	id_artists	release_date	danceability	energy	key
93802	4iJyoBOLtHqaGxP12qzhQI	Peaches (feat. Daniel Caesar & Giveon)	100	198082	1	['Justin Bieber', 'Daniel Caesar', 'Giveon']	['1uNFoZAHBGtllmzznpCI3s', '20wkVLutqVOYrc0kxF...	2021-03-19	0.677	0.696	0
93803	7lPN2DXiMsVn7XUKtOW1CS	drivers license	99	242014	1	['Olivia Rodrigo']	['1McMsnEElThX1knmY4oliG']	2021-01-08	0.585	0.436	10
93804	3Ofmpyhv5UAQ70mENzB277	Astronaut In The Ocean	98	132780	0	['Masked Wolf']	['1uU7g3DNSbsu0QjSEqZtEd']	2021-01-06	0.778	0.695	4
92810	5QO79kh1waicV47BqGRL3g	Save Your Tears	97	215627	1	['The Weeknd']	['1Xyo4u8uXC1ZmMpatF05PJ']	2020-03-20	0.680	0.826	0
92811	6tDDoYIxWvMLTdKpjFkc1B	telepatía	97	160191	0	['Kali Uchis']	['1U1el3k54VvEUzo3ybLPlM']	2020-12-04	0.653	0.524	11
92813	0VjIjW4GlUZAMYd2vXMi3b	Blinding Lights	96	200040	0	['The Weeknd']	['1Xyo4u8uXC1ZmMpatF05PJ']	2020-03-20	0.514	0.730	1
93805	7MAibcTli4lisCtbHKrGMh	Leave The Door Open	96	242096	0	['Bruno Mars', 'Anderson .Paak', 'Silk Sonic']	['0du5cEVh5yTK9QJze8zA0C', '3jK9MiCrA42lLAdMGU...	2021-03-05	0.586	0.616	5
92814	6f3Slt0GbA2bPZlz0aIFXN	The Business	95	164000	0	['Tiësto']	['2o5jDhtHVPhrJdv3cEQ99Z']	2020-09-16	0.798	0.620	8
91866	60ynsPSSKe6O3sfwRnIBRf	Streets	94	226987	1	['Doja Cat']	['5cj0lLjcoR7YOSnhnX0Po5']	2019-11-07	0.749	0.463	11
92816	3FAJ6O0NOHQV8Mc5Ri6ENp	Heartbreak Anniversary	94	198371	0	['Giveon']	['4fxd5Ee7UefO4CUXgwJ7IP']	2020-03-27	0.449	0.465	0

In [58]:
```
plt.figure(figsize=(8,8))
sns.distplot(most_popular['popularity'])
plt.show()
```



1. Histogram shows the DATA DISTRIBUTION.
2. Above graph shows the normal distribution of the data.
3. It has zero skewness, hence it is bell-shaped.

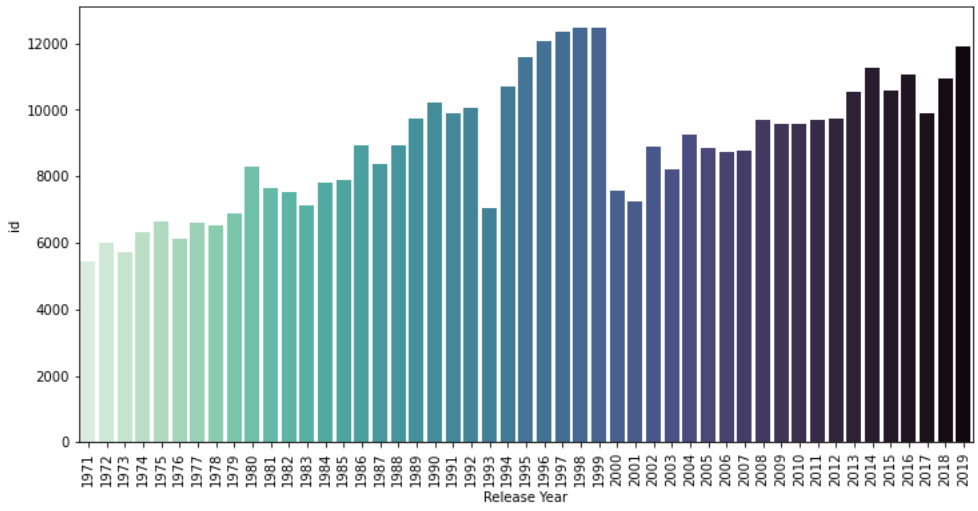
11.Songs by year.

In [59]:
```
# fetch release dates of the songs
df2['release_date'] = pd.to_datetime(df2['release_date'])
df2['year'] = df2['release_date'].apply(lambda x: x.year)

# year df
idx = pd.DataFrame(range(1971,2020),columns=['Release Year'])

# then merge the idx with df "release_year" that is just created
release_year = pd.merge(idx, df2[['id', 'year']], how='left',left_on='Release Year',right_on = 'year', copy=False)

# grouping the songs by year to get the sum
release_year = release_year.groupby('Release Year',as_index=False)['id'].count()
release_year.head()
```

Out[59]:

	Release Year	id
0	1971	5453
1	1972	5998
2	1973	5713
3	1974	6340
4	1975	6624

In [60]:
```
plt.figure(figsize=(12,6))
sns.barplot(data=release_year, x='Release Year', y='id', palette='mako_r')
plt.xticks(rotation=90)
plt.show()
```

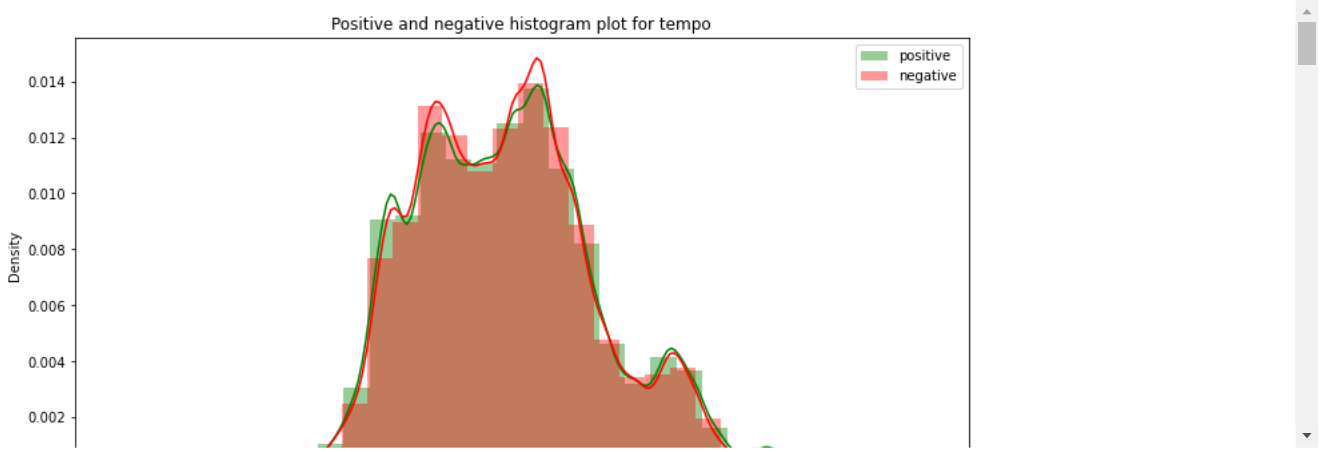


1. The merge() method updates the content of two DataFrame by merging them together, using the specified method(s).
2. Syntax : dataframe.merge(right, how, on, left_on, right_on, left_index, right_index, sort, suffixes, copy, indicator, validate)
3. how ---> 'left','right','outer','inner','cross' ---> Optional. Default 'inner'. Specifies how to merge
4. left_on---> Optional. Specifies in what level to do the merging on the DataFrame to the left.
5. right_on---> Optional. Specifies in what level to do the merging on the DataFrame to the right.
6. copy---> Optional. Default True. Specifies whether to keep copies or not.

12.Multiple feature plots

In [48]:
```
i_feature_cols=['tempo','loudness','speechiness','acousticness','danceability','energy','valence','instrumentalness']
```

In [56]:
```
for feature_col in i_feature_cols:
    p_data=df2[df2['mode']==1][feature_col]
    n_data=df2[df2['mode']==0][feature_col]

    plt.figure(figsize=(12,6))

    sns.distplot(p_data,bins=30,label='positive',color='green')
    sns.distplot(n_data,bins=30,label='negative',color='red')

    plt.legend(loc='upper right')

    plt.title(f'Positive and negative histogram plot for {feature_col}')
```



In [ ]:
```
These are the different histograms of the different features with their mode.
```