In [1]:
```
import numpy as np
import pandas as pd

import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.metrics import euclidean_distances
from scipy.spatial.distance import cdist

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

In [2]:
```
data = pd.read_csv(r"C:\Users\kunal perane\Downloads\data.csv.zip")
genre_data = pd.read_csv(r"C:\Users\kunal perane\Downloads\data_by_genres.csv"
year_data = pd.read_csv(r"C:\Users\kunal perane\Downloads\data_by_year.csv")
```

In [3]:
```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170653 entries, 0 to 170652
Data columns (total 19 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   valence           170653 non-null  float64
 1   year              170653 non-null  int64  
 2   acousticness      170653 non-null  float64
 3   artists           170653 non-null  object 
 4   danceability      170653 non-null  float64
 5   duration_ms       170653 non-null  int64  
 6   energy            170653 non-null  float64
 7   explicit          170653 non-null  int64  
 8   id                170653 non-null  object 
 9   instrumentalness  170653 non-null  float64
 10  key               170653 non-null  int64  
 11  liveness          170653 non-null  float64
 12  loudness          170653 non-null  float64
 13  mode              170653 non-null  int64  
 14  name              170653 non-null  object 
 15  popularity        170653 non-null  int64  
 16  release_date      170653 non-null  object 
 17  speechiness       170653 non-null  float64
 18  tempo             170653 non-null  float64
dtypes: float64(9), int64(6), object(4)
memory usage: 24.7+ MB
None
```

In [4]:
```
print(genre_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2973 entries, 0 to 2972
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   mode              2973 non-null   int64  
 1   genres            2973 non-null   object 
 2   acousticness      2973 non-null   float64
 3   danceability      2973 non-null   float64
 4   duration_ms       2973 non-null   float64
 5   energy            2973 non-null   float64
 6   instrumentalness  2973 non-null   float64
 7   liveness          2973 non-null   float64
 8   loudness          2973 non-null   float64
 9   speechiness       2973 non-null   float64
 10  tempo             2973 non-null   float64
 11  valence           2973 non-null   float64
 12  popularity        2973 non-null   float64
 13  key               2973 non-null   int64  
dtypes: float64(11), int64(2), object(1)
memory usage: 325.3+ KB
None
```

In [5]:
```
print(year_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   mode              100 non-null    int64  
 1   year              100 non-null    int64  
 2   acousticness      100 non-null    float64
 3   danceability      100 non-null    float64
 4   duration_ms       100 non-null    float64
 5   energy            100 non-null    float64
 6   instrumentalness  100 non-null    float64
 7   liveness          100 non-null    float64
 8   loudness          100 non-null    float64
 9   speechiness       100 non-null    float64
 10  tempo             100 non-null    float64
 11  valence           100 non-null    float64
 12  popularity        100 non-null    float64
 13  key               100 non-null    int64  
dtypes: float64(11), int64(3)
memory usage: 11.1 KB
None
```

In [6]:
```
data.head()
```

Out[6]:

	valence	year	acousticness	artists	danceability	duration_ms	energy	explicit	
0	0.0594	1921	0.982	['Sergei Rachmaninoff', 'James Levine', 'Berli...	0.279	831667	0.211	0	4I
1	0.9630	1921	0.732	['Dennis Day']	0.819	180533	0.341	0	7xI
2	0.0394	1921	0.961	['KHP Kridhamardawa Karaton Ngayogyakarta Hadi...	0.328	500062	0.166	0	1
3	0.1650	1921	0.967	['Frank Parker']	0.275	210000	0.309	0	3ftB
4	0.2530	1921	0.957	['Phil Regan']	0.418	166693	0.193	0	4d6H

In [7]:
```
genre_data.head()
```

Out[7]:

	mode	genres	acousticness	danceability	duration_ms	energy	instrumentalness	livenes
0	1	21st century classical	0.979333	0.162883	1.602977e+05	0.071317	0.606834	0.36160
1	1	432hz	0.494780	0.299333	1.048887e+06	0.450678	0.477762	0.13100
2	1	8-bit	0.762000	0.712000	1.151770e+05	0.818000	0.876000	0.12600
3	1	[]	0.651417	0.529093	2.328809e+05	0.419146	0.205309	0.21869
4	1	a cappella	0.676557	0.538961	1.906285e+05	0.316434	0.003003	0.17225

In [8]:
```
year_data.head()
```

Out[8]:

	mode	year	acousticness	danceability	duration_ms	energy	instrumentalness	liveness
0	1	1921	0.886896	0.418597	260537.166667	0.231815	0.344878	0.205710
1	1	1922	0.938592	0.482042	165469.746479	0.237815	0.434195	0.240720
2	1	1923	0.957247	0.577341	177942.362162	0.262406	0.371733	0.227462
3	1	1924	0.940200	0.549894	191046.707627	0.344347	0.581701	0.235219
4	1	1925	0.962607	0.573863	184986.924460	0.278594	0.418297	0.237668

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
valence             0
year                0
acousticness        0
artists             0
danceability        0
duration_ms         0
energy              0
explicit            0
id                  0
instrumentalness    0
key                 0
liveness            0
loudness            0
mode                0
name                0
popularity          0
release_date        0
speechiness         0
tempo               0
dtype: int64
```

In [10]:
```
genre_data.isnull().sum()
```

Out[10]:
```
mode                0
genres              0
acousticness        0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
liveness            0
loudness            0
speechiness         0
tempo               0
valence             0
popularity          0
key                 0
dtype: int64
```

In [11]:
```
year_data.isnull().sum()
```

Out[11]:
```
mode              0
year              0
acousticness      0
danceability      0
duration_ms       0
energy            0
instrumentalness  0
liveness          0
loudness          0
speechiness       0
tempo             0
valence           0
popularity        0
key               0
dtype: int64
```

In [12]:
```
data.shape
```

Out[12]:
```
(170653, 19)
```

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

Out[13]:
```
(2973, 14)
```

In [14]:
```
year_data.shape
```

Out[14]:
```
(100, 14)
```

In [15]:
```
data.dtypes
```

Out[15]:
```
valence           float64
year                int64
acousticness      float64
artists            object
danceability      float64
duration_ms         int64
energy            float64
explicit            int64
id                 object
instrumentalness  float64
key                 int64
liveness          float64
loudness          float64
mode                int64
name               object
popularity          int64
release_date       object
speechiness       float64
tempo             float64
dtype: object
```

In [16]:
```
genre_data.dtypes
```

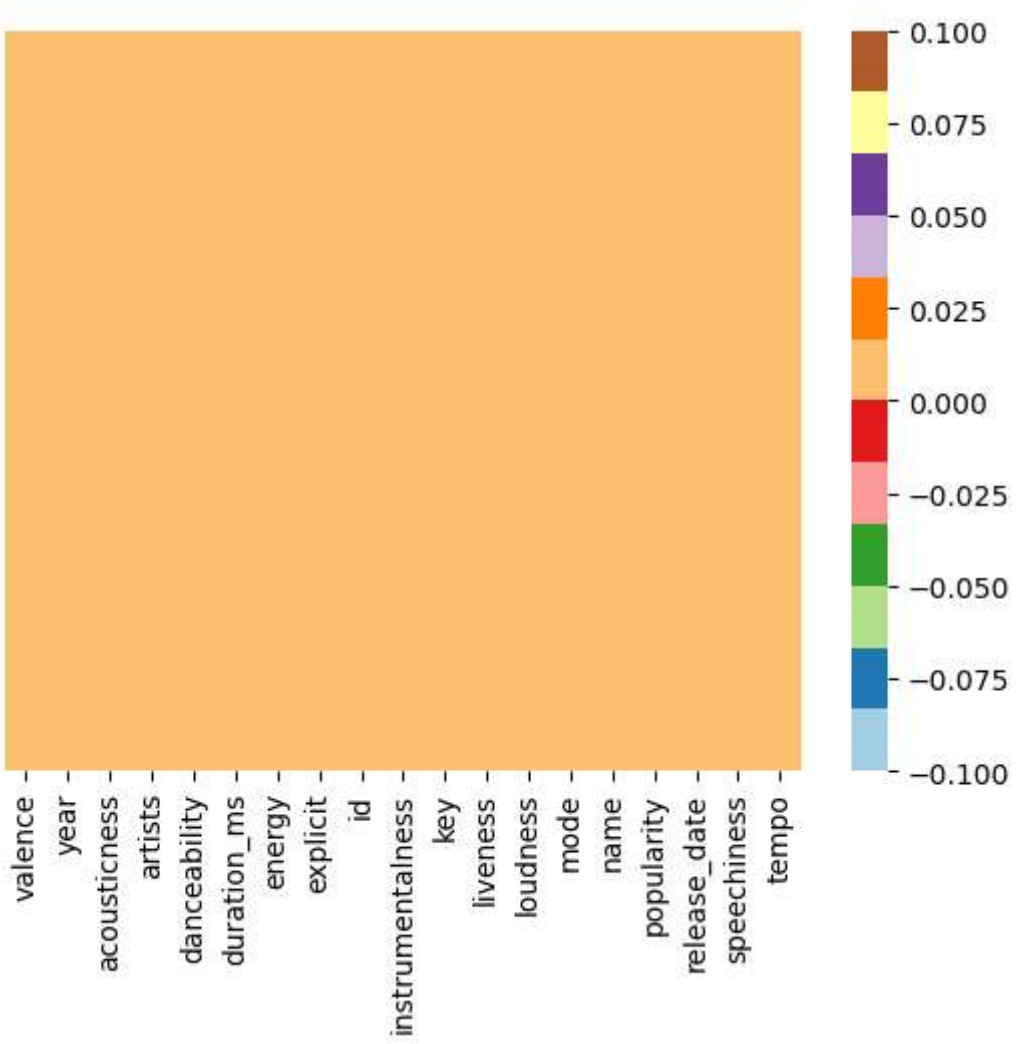
Out[16]:
```
mode                int64
genres             object
acousticness      float64
danceability      float64
duration_ms       float64
energy            float64
instrumentalness  float64
liveness          float64
loudness          float64
speechiness       float64
tempo             float64
valence           float64
popularity        float64
key                 int64
dtype: object
```

In [17]:
```
year_data.dtypes
```

Out[17]:
```
mode                int64
year                int64
acousticness      float64
danceability      float64
duration_ms       float64
energy            float64
instrumentalness  float64
liveness          float64
loudness          float64
speechiness       float64
tempo             float64
valence           float64
popularity        float64
key                 int64
dtype: object
```

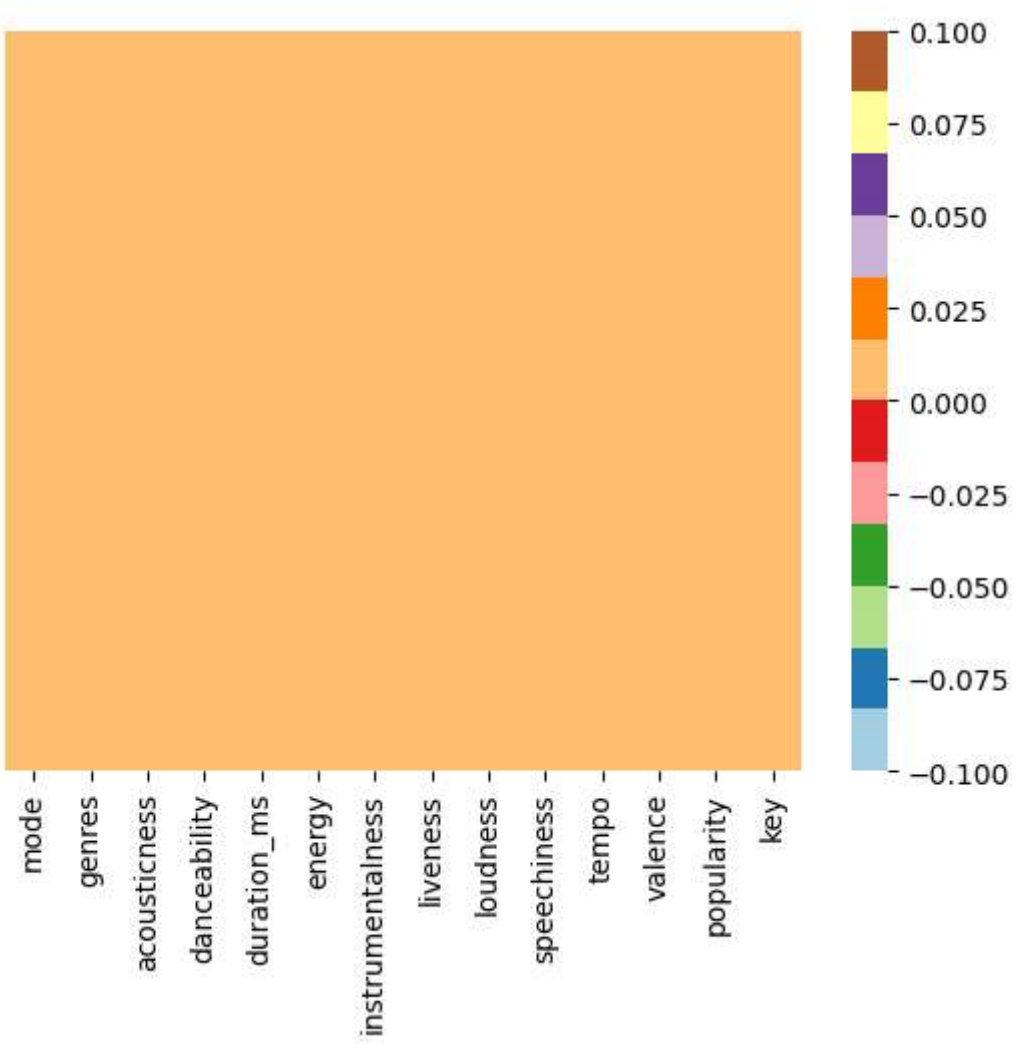
In [18]:
```
sns.heatmap(data.isnull(),yticklabels=False,cmap='Paired')
```

Out[18]: <Axes: >



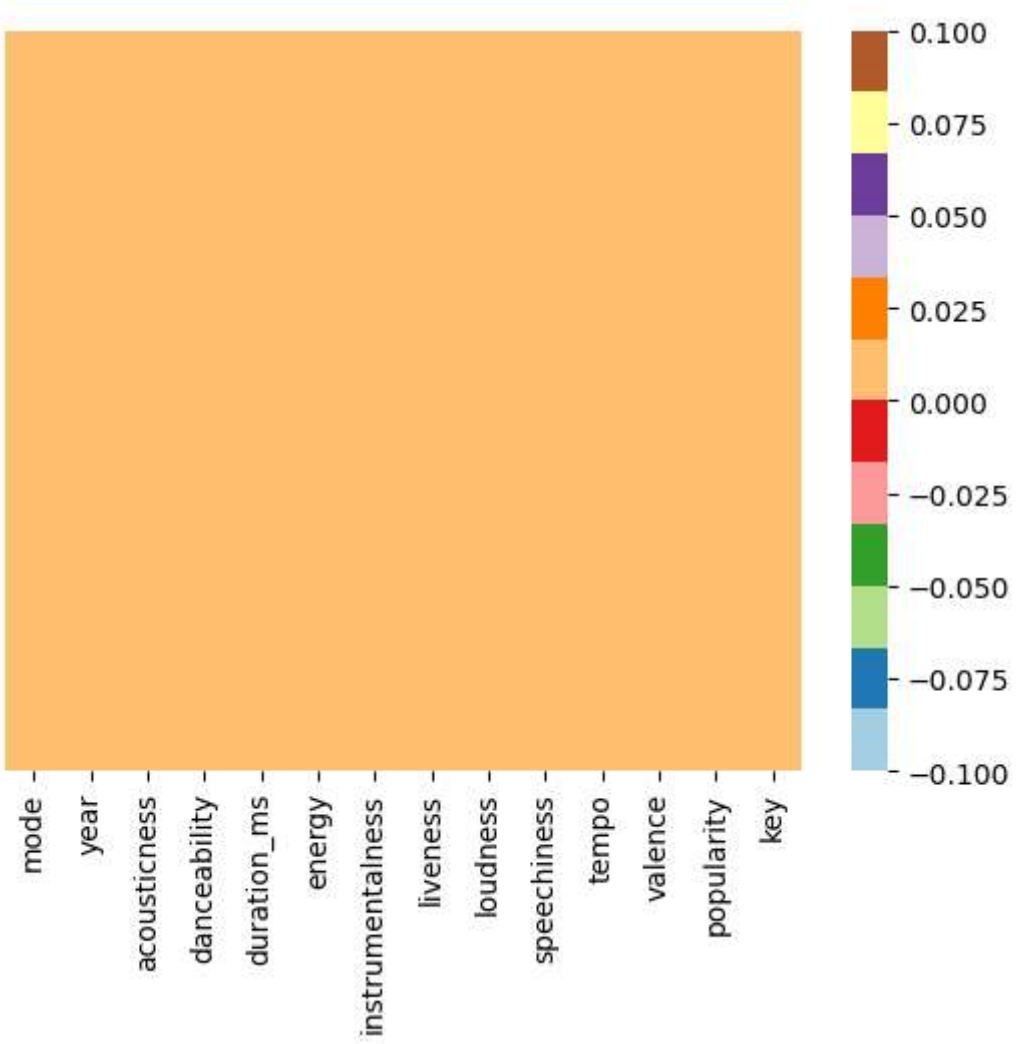
In [19]:
```
sns.heatmap(genre_data.isnull(),yticklabels=False,cmap='Paired')
```

Out[19]: <Axes: >



In [20]:
```
sns.heatmap(year_data.isnull(),yticklabels=False,cmap='Paired')
```

Out[20]: <Axes: >

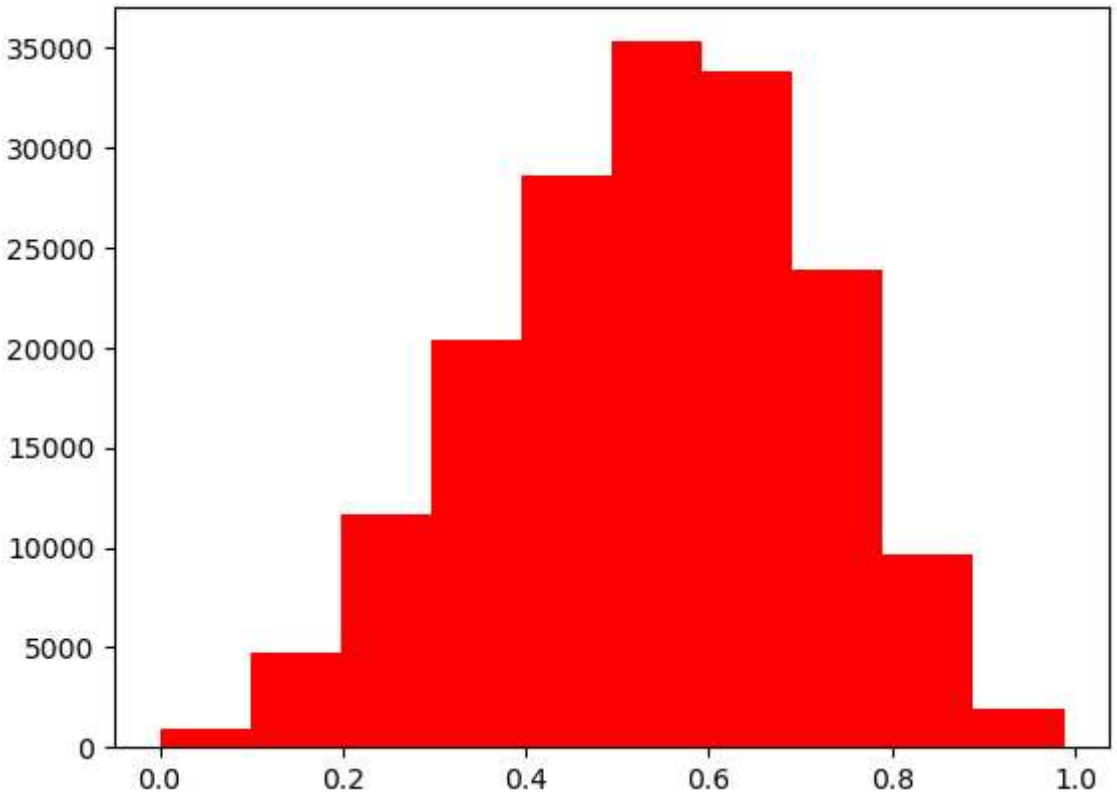


In [21]:
```
top10_genres = genre_data.nlargest(10, 'popularity')

fig = px.bar(top10_genres, x='genres', y=['valence', 'energy', 'danceability',
fig.show()
```

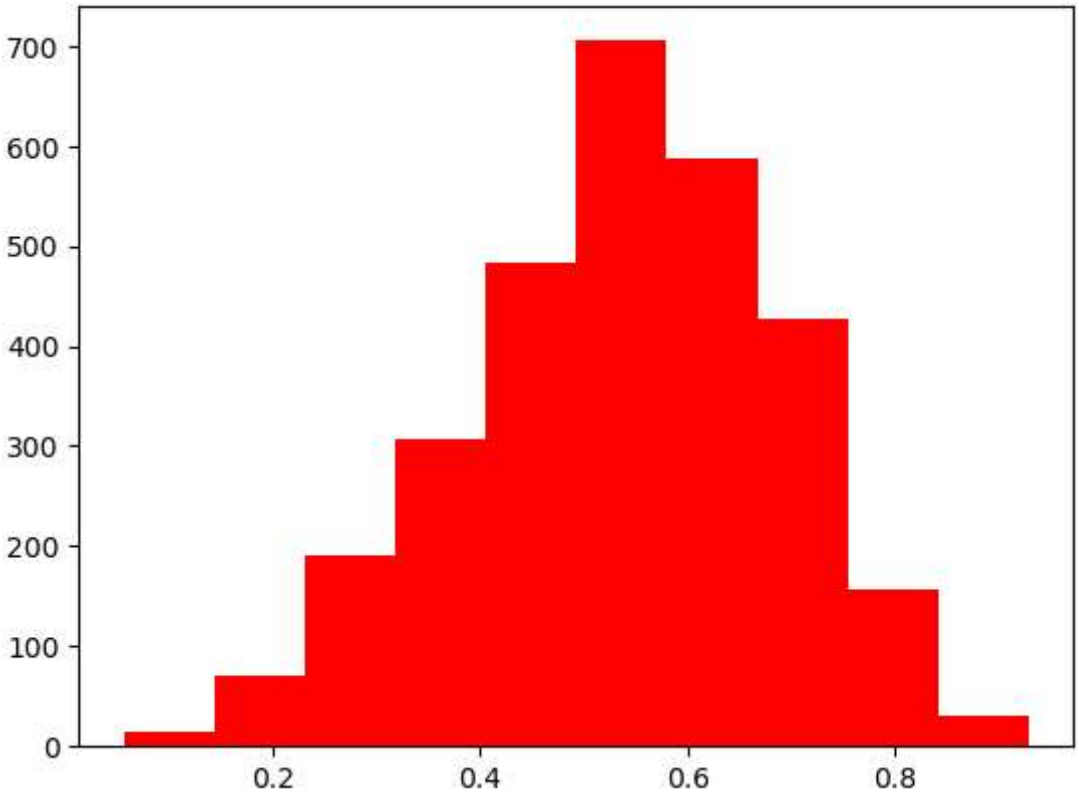
In [22]:
```
plt.hist(data['danceability'],bins=10,color='red')
```

Out[22]:
```
(array([  898.,  4725., 11673., 20360., 28545., 35282., 33768., 23868.,
         9650.,  1884.]),
 array([0.   , 0.0988, 0.1976, 0.2964, 0.3952, 0.494 , 0.5928, 0.6916,
        0.7904, 0.8892, 0.988 ]),
 <BarContainer object of 10 artists>)
```



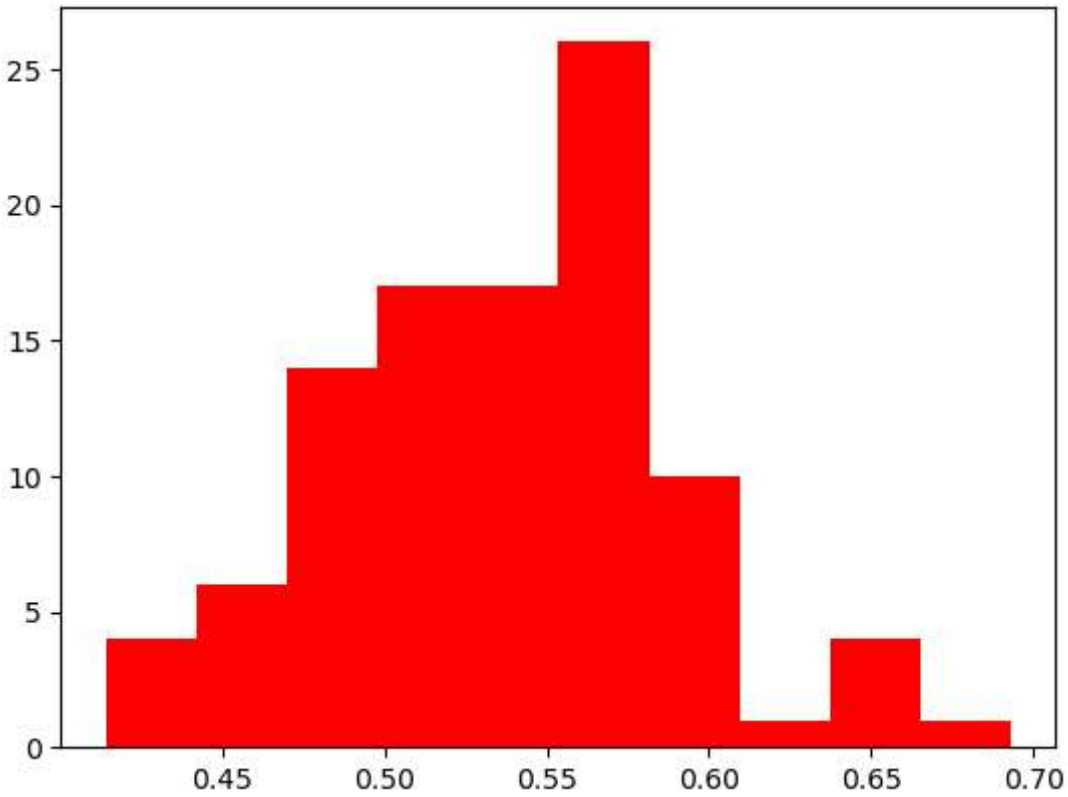
In [23]:
```
plt.hist(genre_data['danceability'],bins=10,color='red')
```

Out[23]:
```
(array([ 15.,  70., 191., 307., 484., 705., 588., 427., 156.,  30.]),
 array([0.0569 , 0.14411, 0.23132, 0.31853, 0.40574, 0.49295, 0.58016,
        0.66737, 0.75458, 0.84179, 0.929  ]),
 <BarContainer object of 10 artists>)
```

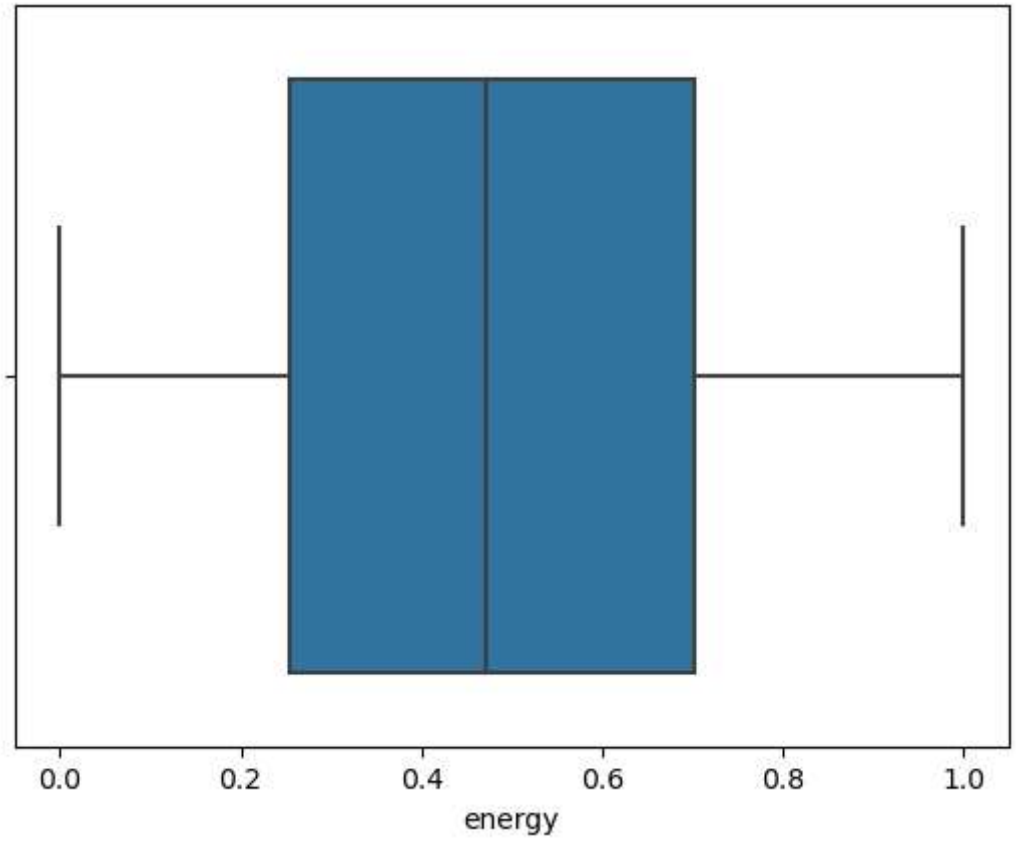


In [24]:
```
plt.hist(year_data['danceability'],bins=10,color='red')
```

Out[24]:
```
(array([ 4.,  6., 14., 17., 17., 26., 10.,  1.,  4.,  1.]),
 array([0.41444501, 0.44229094, 0.47013688, 0.49798281, 0.52582874,
        0.55367467, 0.58152061, 0.60936654, 0.63721247, 0.6650584 ,
        0.69290433]),
 <BarContainer object of 10 artists>)
```

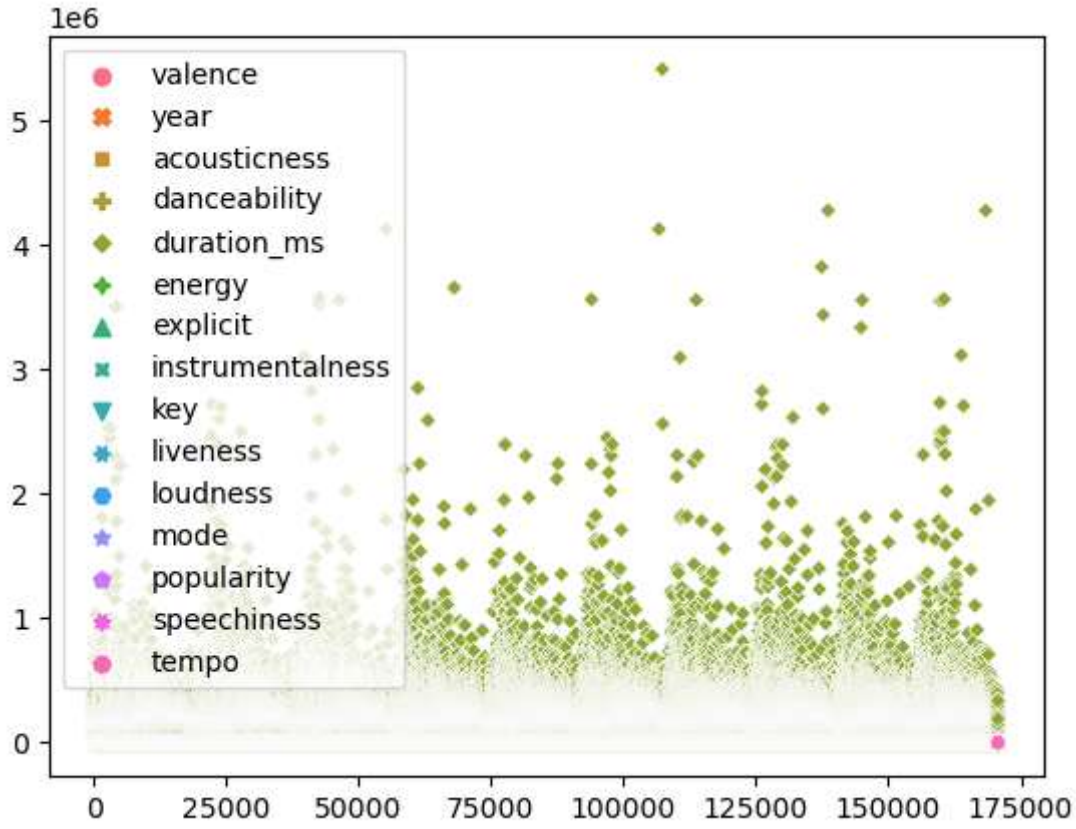


In [25]:
```
sns.boxplot(data=data, x='energy')
plt.show()
```



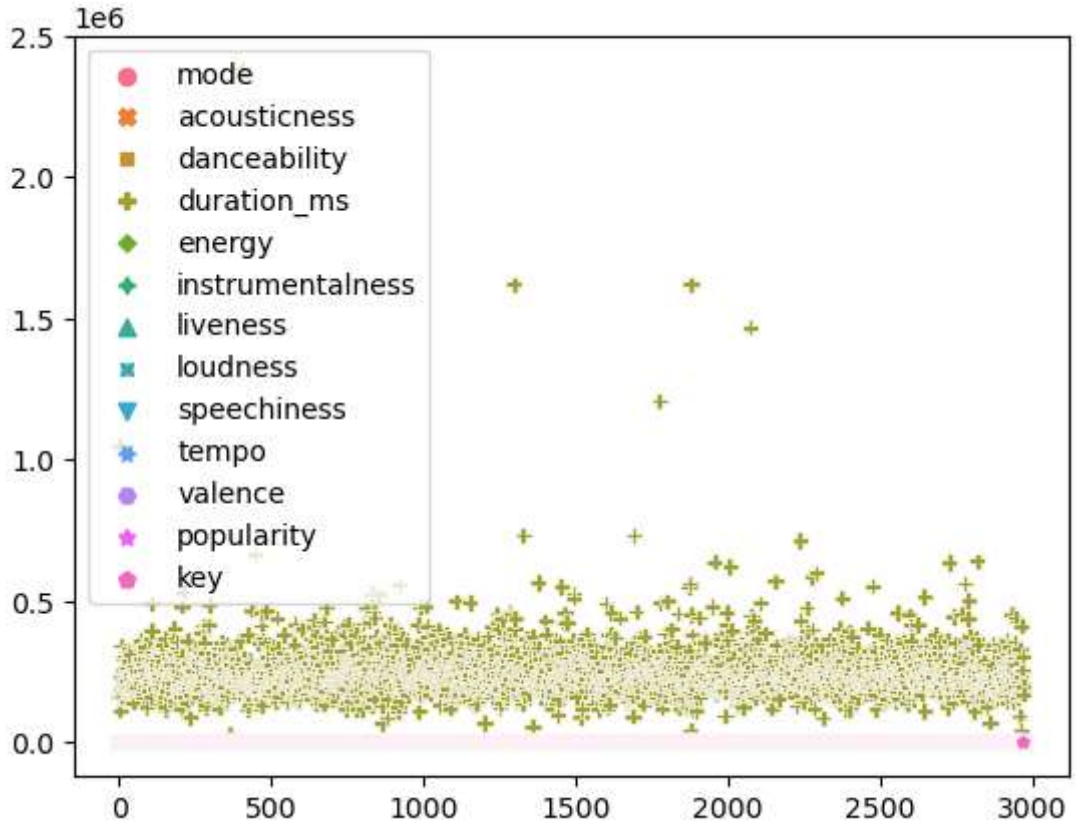
In [26]:
```
sns.scatterplot(data)
```

Out[26]: <Axes: >



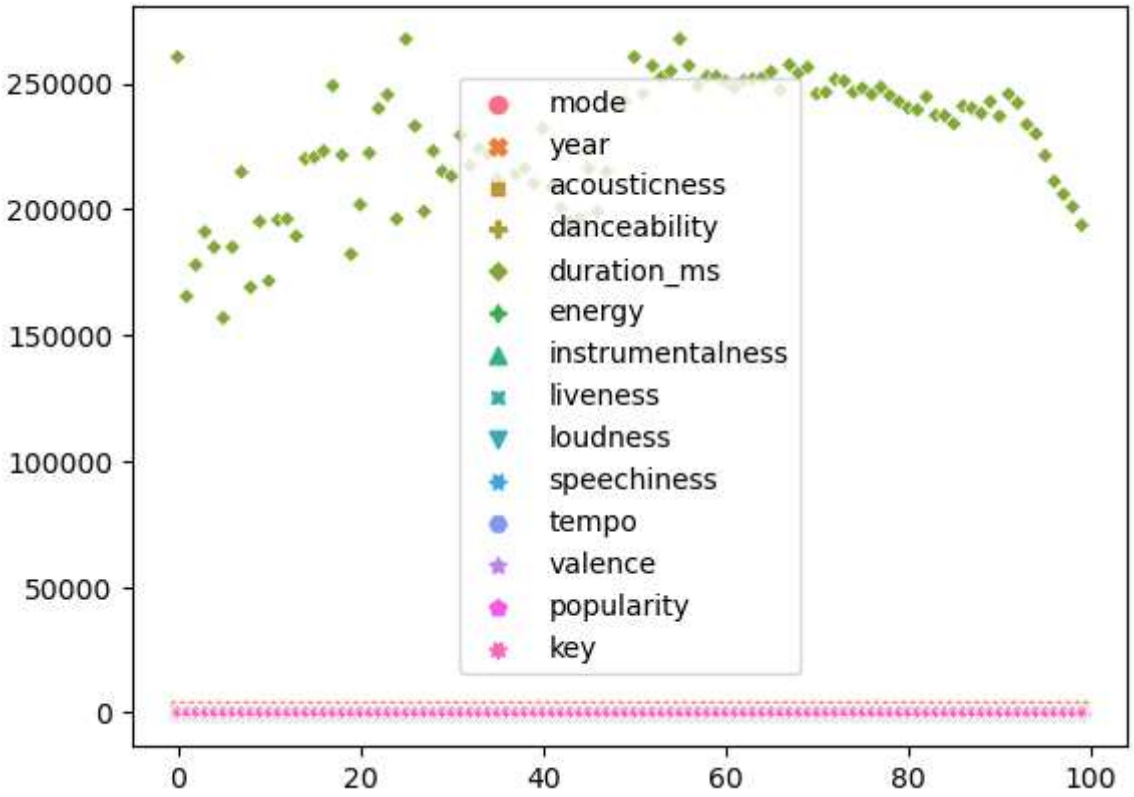
In [27]:
```
sns.scatterplot(genre_data)
```

Out[27]: <Axes: >



In [28]:
```
sns.scatterplot(year_data)
```

Out[28]: <Axes: >



In [29]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

cluster_pipeline = Pipeline([('scaler', StandardScaler()), ('kmeans', KMeans(n
X = genre_data.select_dtypes(np.number)
cluster_pipeline.fit(X)
genre_data['cluster'] = cluster_pipeline.predict(X)
```

In [31]:
```
from sklearn.preprocessing import StandardScaler
scale = StandardScaler().fit(X)
dff = scale.transform(X)
features_scaled = pd.DataFrame( X, columns= X.columns)
features_scaled.head()
```

Out[31]:

	mode	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudn
0	1	0.979333	0.162883	1.602977e+05	0.071317	0.606834	0.361600	-31.514
1	1	0.494780	0.299333	1.048887e+06	0.450678	0.477762	0.131000	-16.854
2	1	0.762000	0.712000	1.151770e+05	0.818000	0.876000	0.126000	-9.180
3	1	0.651417	0.529093	2.328809e+05	0.419146	0.205309	0.218696	-12.288
4	1	0.676557	0.538961	1.906285e+05	0.316434	0.003003	0.172254	-12.479

In [32]:
```
from sklearn.cluster import KMeans
```

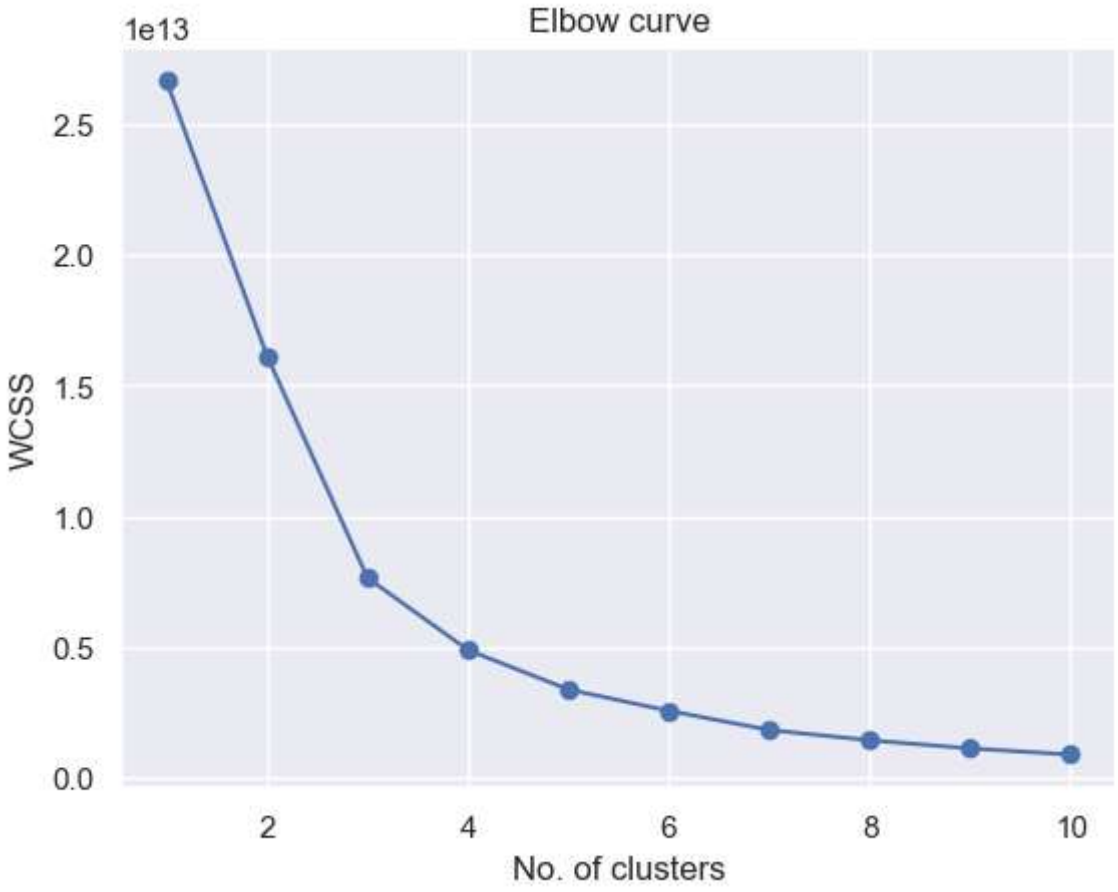
In [33]:
```
wcss = []
for i in range(1,11):
 kmeans=KMeans(n_clusters=i,init='k-means++',random_state=25)
 kmeans.fit(features_scaled)

 wcss.append(kmeans.inertia_)
```

In [34]:
```
wcss
```

Out[34]:
```
[26628885290978.64,
 16134098152145.098,
 7650517770472.112,
 4896217028175.5625,
 3385307401863.378,
 2571261948532.4434,
 1838672914613.7983,
 1445577756386.238,
 1135894838044.8372,
 900920947698.4186]
```

In [35]:
```
sns.set()
plt.plot(range(1,11),wcss, marker="o")
plt.title('Elbow curve')
plt.xlabel('No. of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [36]:
```
kmeans=KMeans(n_clusters=4,init='k-means++',random_state=0)
y=kmeans.fit_predict(features_scaled)
```

In [37]:
```
print(y)
```

```
[1 3 1 ... 2 1 1]
```

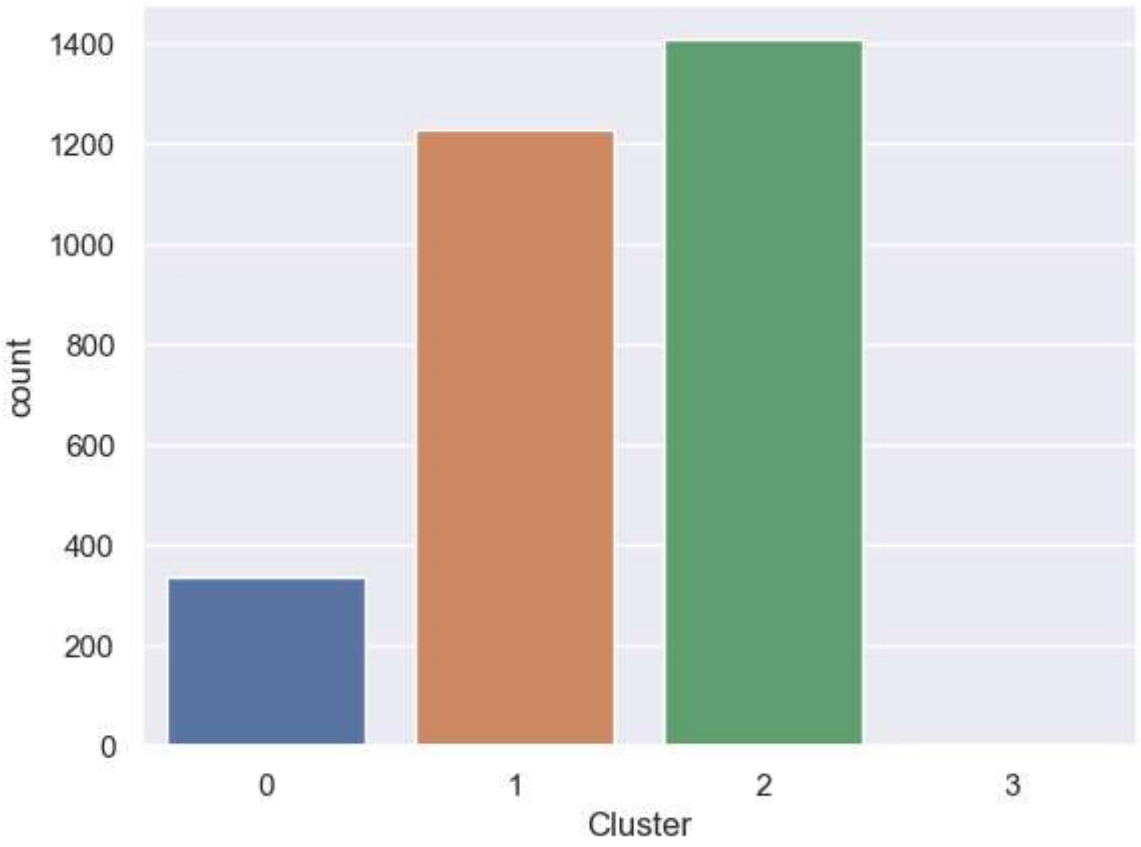
In [38]:
```
data_output = features_scaled.copy(deep = True)
data_output['Cluster'] = kmeans.labels_
data_output.head()
```

Out[38]:

	mode	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudn
0	1	0.979333	0.162883	1.602977e+05	0.071317	0.606834	0.361600	-31.514
1	1	0.494780	0.299333	1.048887e+06	0.450678	0.477762	0.131000	-16.854
2	1	0.762000	0.712000	1.151770e+05	0.818000	0.876000	0.126000	-9.180
3	1	0.651417	0.529093	2.328809e+05	0.419146	0.205309	0.218696	-12.288
4	1	0.676557	0.538961	1.906285e+05	0.316434	0.003003	0.172254	-12.479

In [39]:
```
sns.countplot(x='Cluster',data=data_output)
```

Out[39]:
```
<Axes: xlabel='Cluster', ylabel='count'>
```



In [40]:
```
np.unique(kmeans.labels_, return_counts=True)
```

Out[40]:
```
(array([0, 1, 2, 3]), array([ 335, 1226, 1406,    6], dtype=int64))
```

In [41]:
```
from sklearn.metrics import silhouette_score, calinski_harabasz_score,davies_b
silhouette_avg = silhouette_score(features_scaled, y)
print(f"Silhouette Score: {silhouette_avg}")
```

```
Silhouette Score: 0.5129622261630266
```

In [42]:
```
calinski_harabasz_index = calinski_harabasz_score(features_scaled, y)
print(f"Calinski-Harabasz Index: {calinski_harabasz_index}")
```

```
Calinski-Harabasz Index: 4392.784805169155
```

In [43]:
```
davies_bouldin_index = davies_bouldin_score(features_scaled, y)
print(f"Davies-Bouldin Index: {davies_bouldin_index}")
```

```
Davies-Bouldin Index: 0.5389046422916128
```

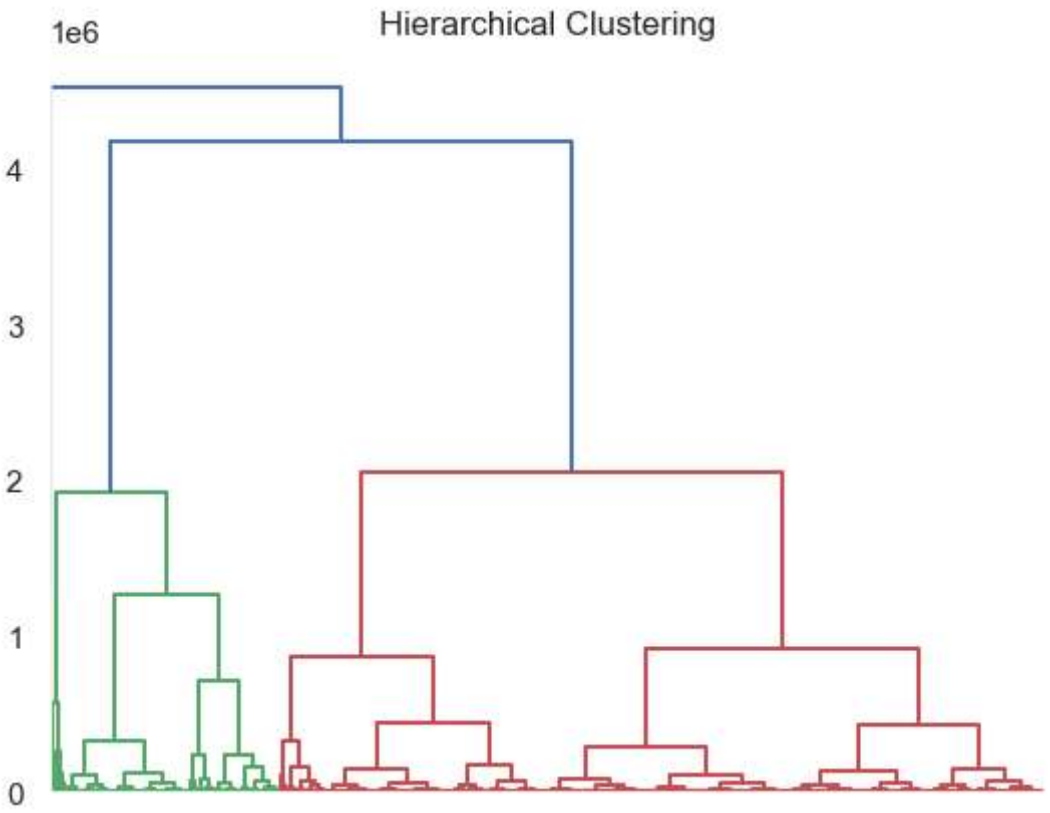
In [44]:
```
import scipy.cluster.hierarchy as sch
from sklearn.preprocessing import scale as s
from scipy.cluster.hierarchy import dendrogram, linkage
```

In [45]:
```
Z = sch.linkage(features_scaled,method='ward')
Z
```

Out[45]:
```
array([[2.65000000e+02, 1.27500000e+03, 0.00000000e+00, 2.00000000e+00],
       [1.80000000e+01, 1.74700000e+03, 0.00000000e+00, 2.00000000e+00],
       [1.05100000e+03, 1.84600000e+03, 0.00000000e+00, 2.00000000e+00],
       ...,
       [5.93700000e+03, 5.93800000e+03, 2.05562752e+06, 2.28300000e+03],
       [5.94100000e+03, 5.94200000e+03, 4.17258373e+06, 2.96700000e+03],
       [5.94000000e+03, 5.94300000e+03, 4.53279122e+06, 2.97300000e+03]])
```

In [46]:
```
den = sch.dendrogram(Z)
plt.tick_params(
 axis='x',
 which='both',
 bottom=False,
 top=False,
 labelbottom=False)
plt.title('Hierarchical Clustering')
```

Out[46]: Text(0.5, 1.0, 'Hierarchical Clustering')



In [47]:
```
from sklearn.cluster import AgglomerativeClustering
```

In [49]:
```
hc_model = AgglomerativeClustering(n_clusters = 2, affinity = 'euclidean', lin
```

In [50]:
```
y_cluster = hc_model.fit_predict(features_scaled)
```

In [51]:
```
y_cluster
```

Out[51]: array([0, 1, 0, ..., 0, 0, 0], dtype=int64)

In [52]:
```
data_out = features_scaled.copy(deep = True)
data_out['Cluster'] = hc_model.labels_
data_out.head()
```

Out[52]:

	mode	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudn
0	1	0.979333	0.162883	1.602977e+05	0.071317	0.606834	0.361600	-31.514
1	1	0.494780	0.299333	1.048887e+06	0.450678	0.477762	0.131000	-16.854
2	1	0.762000	0.712000	1.151770e+05	0.818000	0.876000	0.126000	-9.180
3	1	0.651417	0.529093	2.328809e+05	0.419146	0.205309	0.218696	-12.288
4	1	0.676557	0.538961	1.906285e+05	0.316434	0.003003	0.172254	-12.479

In [53]:
```
np.unique(hc_model.labels_, return_counts=True)
```

Out[53]:
```
(array([0, 1], dtype=int64), array([2967,    6], dtype=int64))
```

In [54]:
```
silhouette_avg = silhouette_score(features_scaled, y_cluster)
print(f"Silhouette Score: {silhouette_avg}")
```

```
Silhouette Score: 0.9411156753351567
```

In [55]:
```
calinski_harabasz_index = calinski_harabasz_score(features_scaled, y_cluster)
print(f"Calinski-Harabasz Index: {calinski_harabasz_index}")
```

```
Calinski-Harabasz Index: 1866.0902278992346
```

In [56]:
```
davies_bouldin_index = davies_bouldin_score(features_scaled, y_cluster)
print(f"Davies-Bouldin Index: {davies_bouldin_index}")
```

```
Davies-Bouldin Index: 0.28064368021260866
```

In [ ]:
```

```