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

In [2]:
```
Data=pd.read_csv(r"C:\Users\faij2\Downloads\DATA SET OF PROJECTS\spotify\data.
```

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

Out[3]:

	valence	year	acousticness	artists	danceability	duration_ms	energ
0	0.0594	1921	0.982	['Sergei Rachmaninoff', 'James Levine', 'Berli...	0.279	831667	0.21
1	0.9630	1921	0.732	['Dennis Day']	0.819	180533	0.34
2	0.0394	1921	0.961	['KHP Kridhamardawa Karaton Ngayogyakarta Hadi...	0.328	500062	0.16
3	0.1650	1921	0.967	['Frank Parker']	0.275	210000	0.30
4	0.2530	1921	0.957	['Phil Regan']	0.418	166693	0.19

In [4]:
```
GENRES=pd.read_csv(r"C:\Users\faij2\Downloads\DATA SET OF PROJECTS\spotify\dat
YEAR=pd.read_csv(r"C:\Users\faij2\Downloads\DATA SET OF PROJECTS\spotify\data_
ARTIST=pd.read_csv(r"C:\Users\faij2\Downloads\DATA SET OF PROJECTS\spotify\dat
```

In [5]:
```
YEAR.head()
```

Out[5]:

	mode	year	acousticness	danceability	duration_ms	energy	instrumenta
0	1	1921	0.886896	0.418597	260537.166667	0.231815	0.3
1	1	1922	0.938592	0.482042	165469.746479	0.237815	0.4
2	1	1923	0.957247	0.577341	177942.362162	0.262406	0.3
3	1	1924	0.940200	0.549894	191046.707627	0.344347	0.5
4	1	1925	0.962607	0.573863	184986.924460	0.278594	0.4

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

Out[6]:

	genres	artists	acousticness	danceability	duration_ms	energy	ins
0	['show tunes']	"Cats" 1981 Original London Cast	0.590111	0.467222	250318.555556	0.394003	
1	[]	"Cats" 1983 Broadway Cast	0.862538	0.441731	287280.000000	0.406808	
2	[]	"Fiddler On The Roof" Motion Picture Chorus	0.856571	0.348286	328920.000000	0.286571	
3	[]	"Fiddler On The Roof" Motion Picture Orchestra	0.884926	0.425074	262890.962963	0.245770	
4	[]	"Joseph And The Amazing Technicolor Dreamcoat"...	0.510714	0.467143	270436.142857	0.488286	

In [7]:

```
ARTIST.head()
```

Out[7]:

	mode	count	acousticness	artists	danceability	duration_ms	energ
0	1	9	0.590111	"Cats" 1981 Original London Cast	0.467222	250318.555556	0.3940
1	1	26	0.862538	"Cats" 1983 Broadway Cast	0.441731	287280.000000	0.4068
2	1	7	0.856571	"Fiddler On The Roof" Motion Picture Chorus	0.348286	328920.000000	0.2865
3	1	27	0.884926	"Fiddler On The Roof" Motion Picture Orchestra	0.425074	262890.962963	0.2457
4	1	7	0.510714	"Joseph And The Amazing Technicolor Dreamcoat"...	0.467143	270436.142857	0.4882

Data Collection and Preprocessin

In [8]:
```
ARTIST.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28680 entries, 0 to 28679
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   mode              28680 non-null  int64  
 1   count             28680 non-null  int64  
 2   acousticness      28680 non-null  float64
 3   artists           28680 non-null  object 
 4   danceability      28680 non-null  float64
 5   duration_ms       28680 non-null  float64
 6   energy            28680 non-null  float64
 7   instrumentalness  28680 non-null  float64
 8   liveness          28680 non-null  float64
 9   loudness          28680 non-null  float64
 10  speechiness       28680 non-null  float64
 11  tempo             28680 non-null  float64
 12  valence           28680 non-null  float64
 13  popularity        28680 non-null  float64
 14  key               28680 non-null  int64  
dtypes: float64(11), int64(3), object(1)
memory usage: 3.3+ MB
```

In [9]:
```
GENRES.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28680 entries, 0 to 28679
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   genres            28680 non-null  object 
 1   artists           28680 non-null  object 
 2   acousticness      28680 non-null  float64
 3   danceability      28680 non-null  float64
 4   duration_ms       28680 non-null  float64
 5   energy            28680 non-null  float64
 6   instrumentalness  28680 non-null  float64
 7   liveness          28680 non-null  float64
 8   loudness          28680 non-null  float64
 9   speechiness       28680 non-null  float64
 10  tempo             28680 non-null  float64
 11  valence           28680 non-null  float64
 12  popularity        28680 non-null  float64
 13  key               28680 non-null  int64  
 14  mode              28680 non-null  int64  
 15  count             28680 non-null  int64  
dtypes: float64(11), int64(3), object(2)
memory usage: 3.5+ MB
```

In [10]:
```
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
```

In [11]:
```
YEAR.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
```

In [12]:
```
Data.drop_duplicates()
```

```
YEAR.drop_duplicates()
GENRES.drop_duplicates()
ARTIST.drop_duplicates()
```

Out[12]:

	mode	count	acousticness	artists	danceability	duration_ms	
0	1	9	0.590111	"Cats" 1981 Original London Cast	0.467222	250318.555556	0.
1	1	26	0.862538	"Cats" 1983 Broadway Cast	0.441731	287280.000000	0.
2	1	7	0.856571	"Fiddler On The Roof" Motion Picture Chorus	0.348286	328920.000000	0.
3	1	27	0.884926	"Fiddler On The Roof" Motion Picture Orchestra	0.425074	262890.962963	0.
4	1	7	0.510714	"Joseph And The Amazing Technicolor Dreamcoat"...	0.467143	270436.142857	0.
...	...	...	...	...	...	...	
28675	1	2	0.512000	麥志誠	0.356000	198773.000000	0.
28676	0	2	0.541000	黃品源	0.578000	293840.000000	0.
28677	1	11	0.785455	黃國隆	0.570818	174582.727273	0.
28678	1	2	0.381000	黑豹	0.353000	316160.000000	0.
28679	1	2	0.568000	조정현	0.447000	237688.000000	0.

28680 rows × 15 columns

In [13]:

```
Data.isnull().sum()
```

Out[13]:
```
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 [14]:
```
Data.describe()
```

Out[14]:

	valence	year	acousticness	danceability	duration_m
count	170653.000000	170653.000000	170653.000000	170653.000000	1.706530e+0
mean	0.528587	1976.787241	0.502115	0.537396	2.309483e+0
std	0.263171	25.917853	0.376032	0.176138	1.261184e+0
min	0.000000	1921.000000	0.000000	0.000000	5.108000e+0
25%	0.317000	1956.000000	0.102000	0.415000	1.698270e+0
50%	0.540000	1977.000000	0.516000	0.548000	2.074670e+0
75%	0.747000	1999.000000	0.893000	0.668000	2.624000e+0
max	1.000000	2020.000000	0.996000	0.988000	5.403500e+0

In [15]:
```
Data.select_dtypes(include='object').describe()
```

Out[15]:

	artists	id	name	release_date
count	170653	170653	170653	170653
unique	34088	170653	133638	11244
top	['Эрнест Хемингуэй']	4BJqT0PrAfrxzMOxytFOIz	White Christmas	1945
freq	1211	1	73	1446

Data Analysis

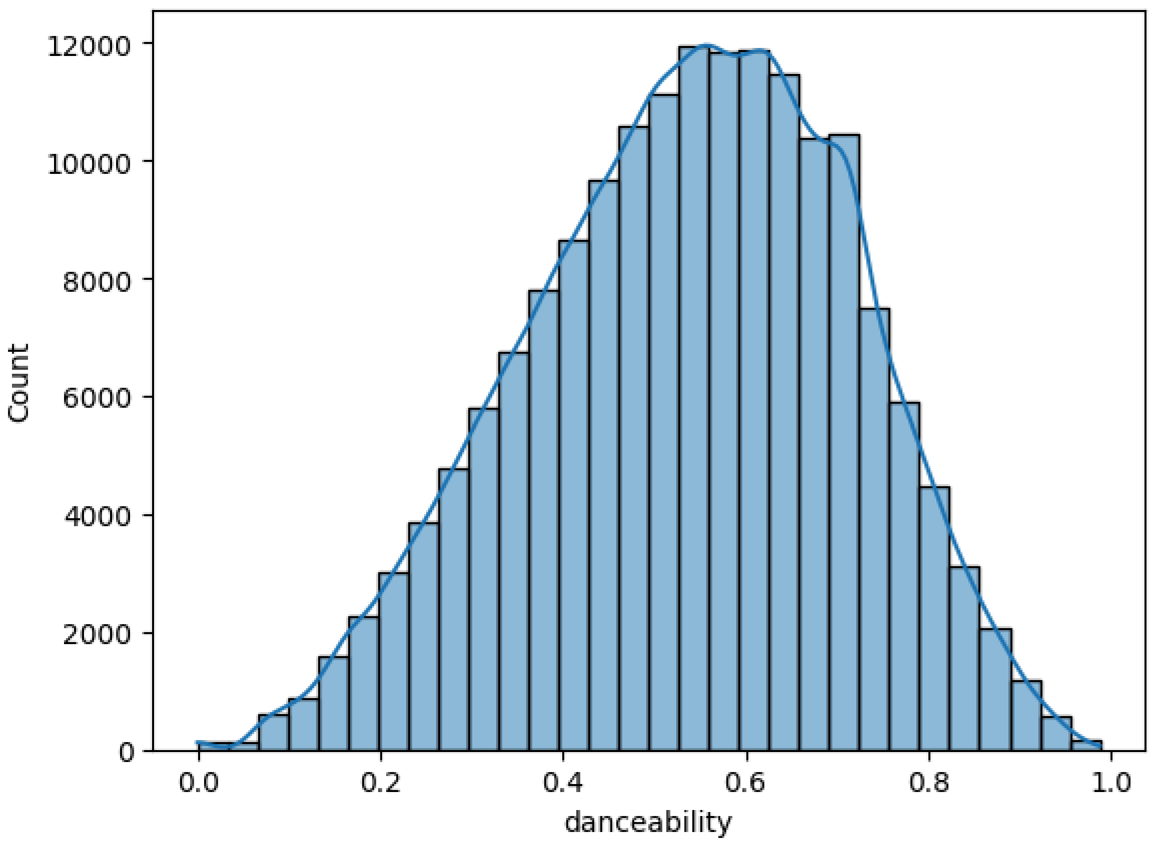
Analyze the distribution of various features

In [16]:
```
print(Data[['danceability', 'energy', 'tempo']].describe())
```

```
        danceability         energy          tempo
count  170653.000000  170653.000000  170653.000000
mean        0.537396       0.482389     116.861590
std         0.176138       0.267646      30.708533
min         0.000000       0.000000       0.000000
25%         0.415000       0.255000      93.421000
50%         0.548000       0.471000     114.729000
75%         0.668000       0.703000     135.537000
max         0.988000       1.000000     243.507000
```

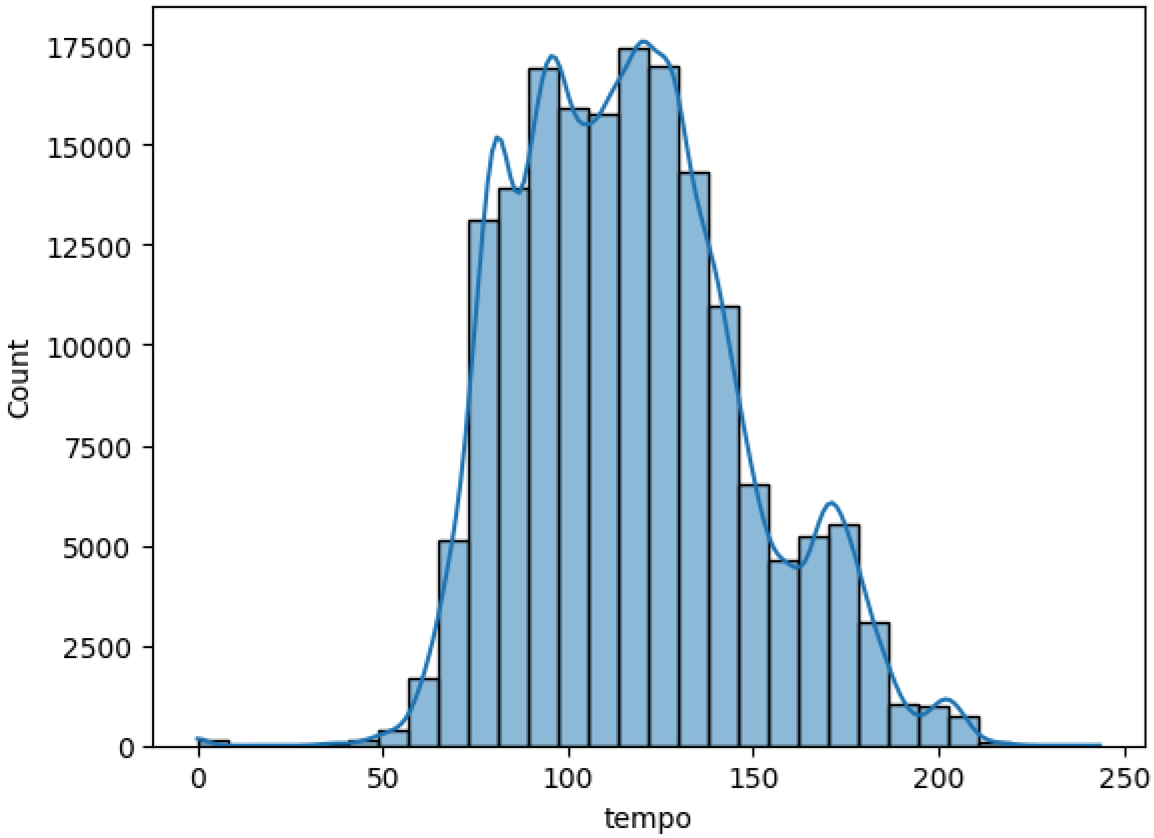
In [17]:
```
#Data['danceability'].hist()
sns.histplot(Data['danceability'], bins=30, kde=True)
# data danceability is like normal distribution
```

Out[17]:
```
<Axes: xlabel='danceability', ylabel='Count'>
```



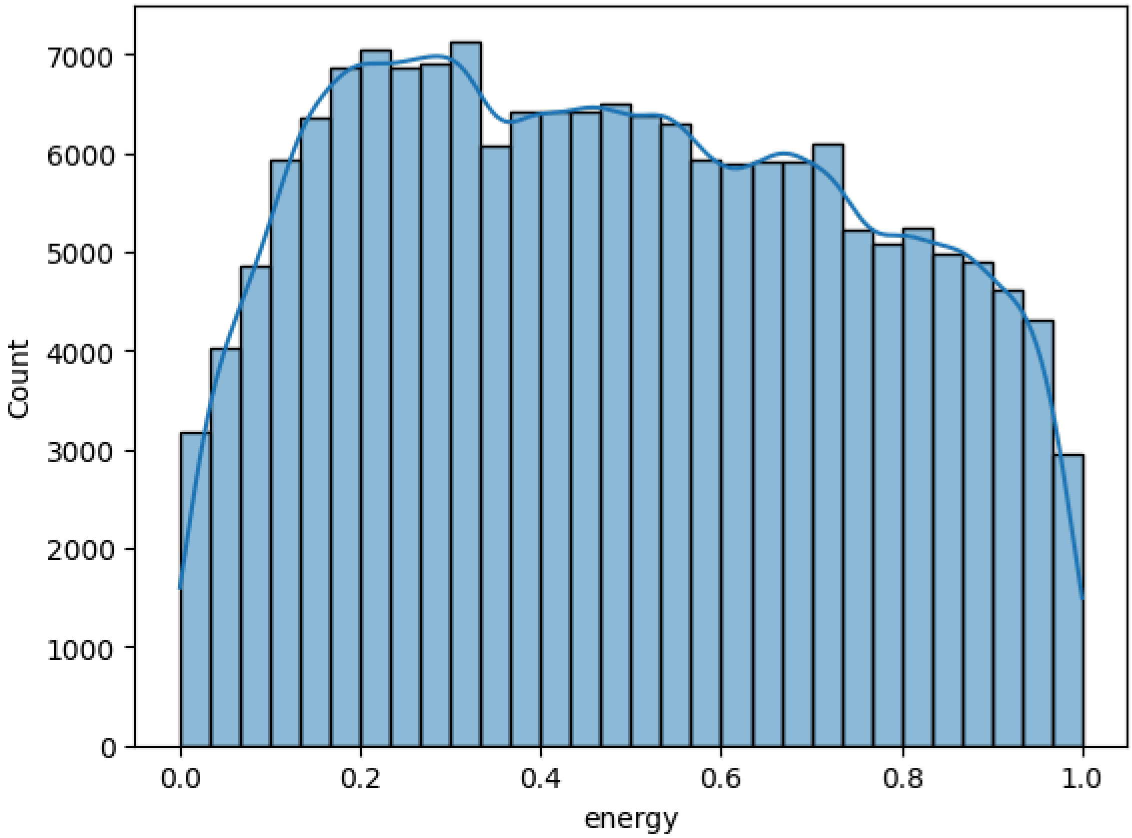
In [18]:
```
#Data['tempo'].hist()
sns.histplot(Data['tempo'], bins=30, kde=True)
# its right skew (0.4497406161103406)
```

Out[18]: <Axes: xlabel='tempo', ylabel='Count'>



In [19]:
```
#Data['energy'].hist()
sns.histplot(Data['energy'], bins=30, kde=True)
energy_std=Data['energy'].std()
print("energy_std=",energy_std)
```

```
energy_std= 0.2676457045730614
```

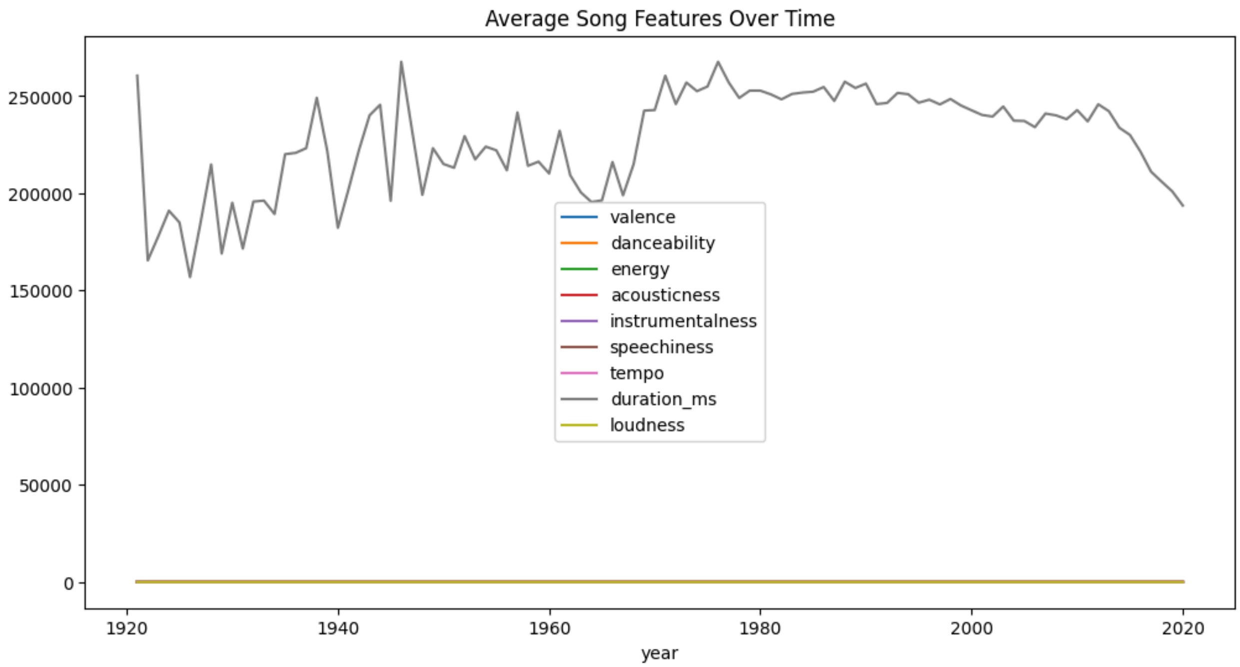


Identify trends over time (e.g., popularity of genres, changes in song features

In [20]:
```
yearly_avg = Data.groupby('year')[['valence', 'danceability', 'energy', 'acous
```

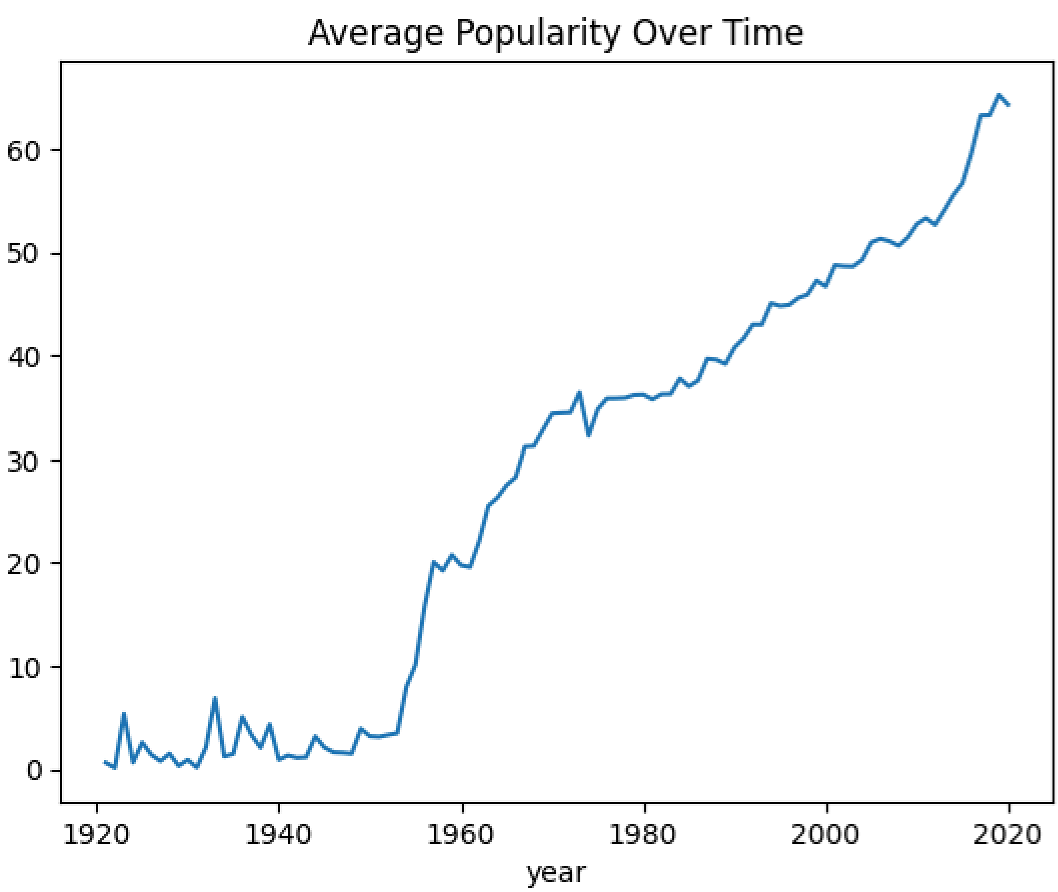
In [21]:
```
yearly_avg.plot(figsize=(12, 6), title='Average Song Features Over Time')
```

Out[21]:
```
<Axes: title={'center': 'Average Song Features Over Time'}, xlabel='year'>
```



In [22]:
```
pop_trend = Data.groupby('year')['popularity'].mean()
pop_trend.plot(title='Average Popularity Over Time')
```

Out[22]: <Axes: title={'center': 'Average Popularity Over Time'}, xlabel='year'>



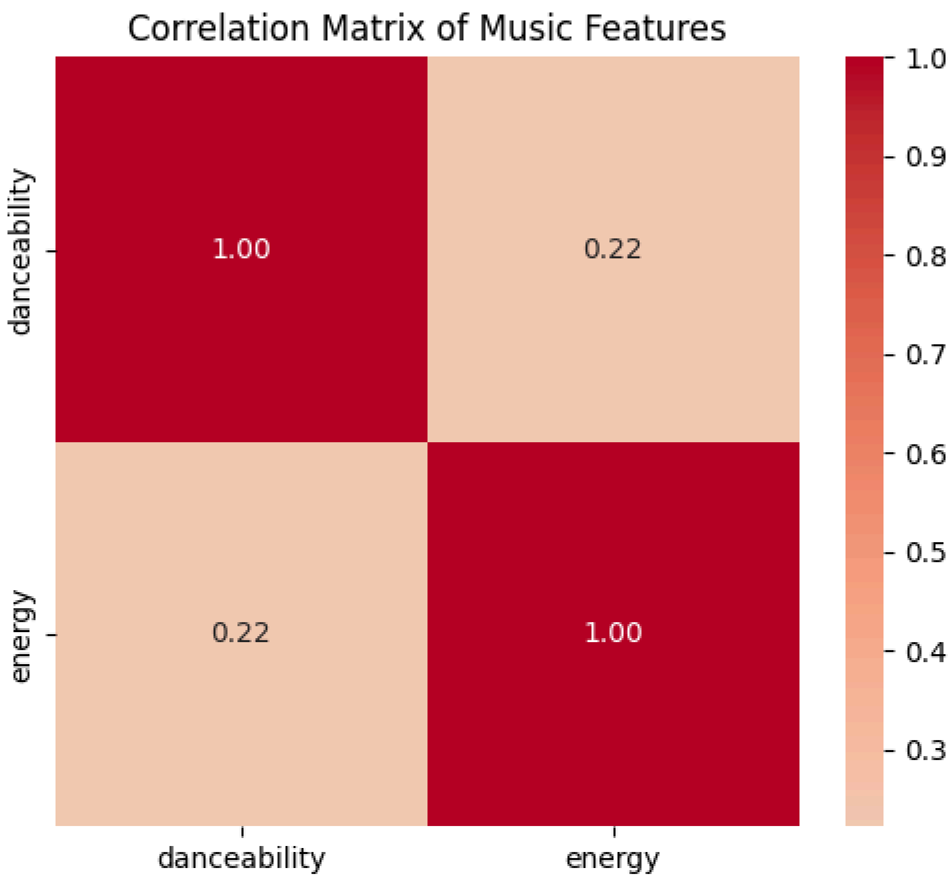
Examine correlations between different features (e.g., energy vs. danceability

In [23]:
```
features = [
    'acousticness', 'danceability', 'duration_ms',  'instrumentalness',
    'liveness', 'loudness', 'speechiness', 'tempo', 'valence', 'popularity', '
]
df_numeric = Data[features]

correlation_danceability = df_numeric.corr()
```

In [24]:
```
features = [
    'acousticness', 'energy', 'duration_ms',  'instrumentalness',
    'liveness', 'loudness', 'speechiness', 'tempo', 'valence', 'popularity', '
]
df_numeric = Data[features]

correlation_energy = df_numeric.corr()
```

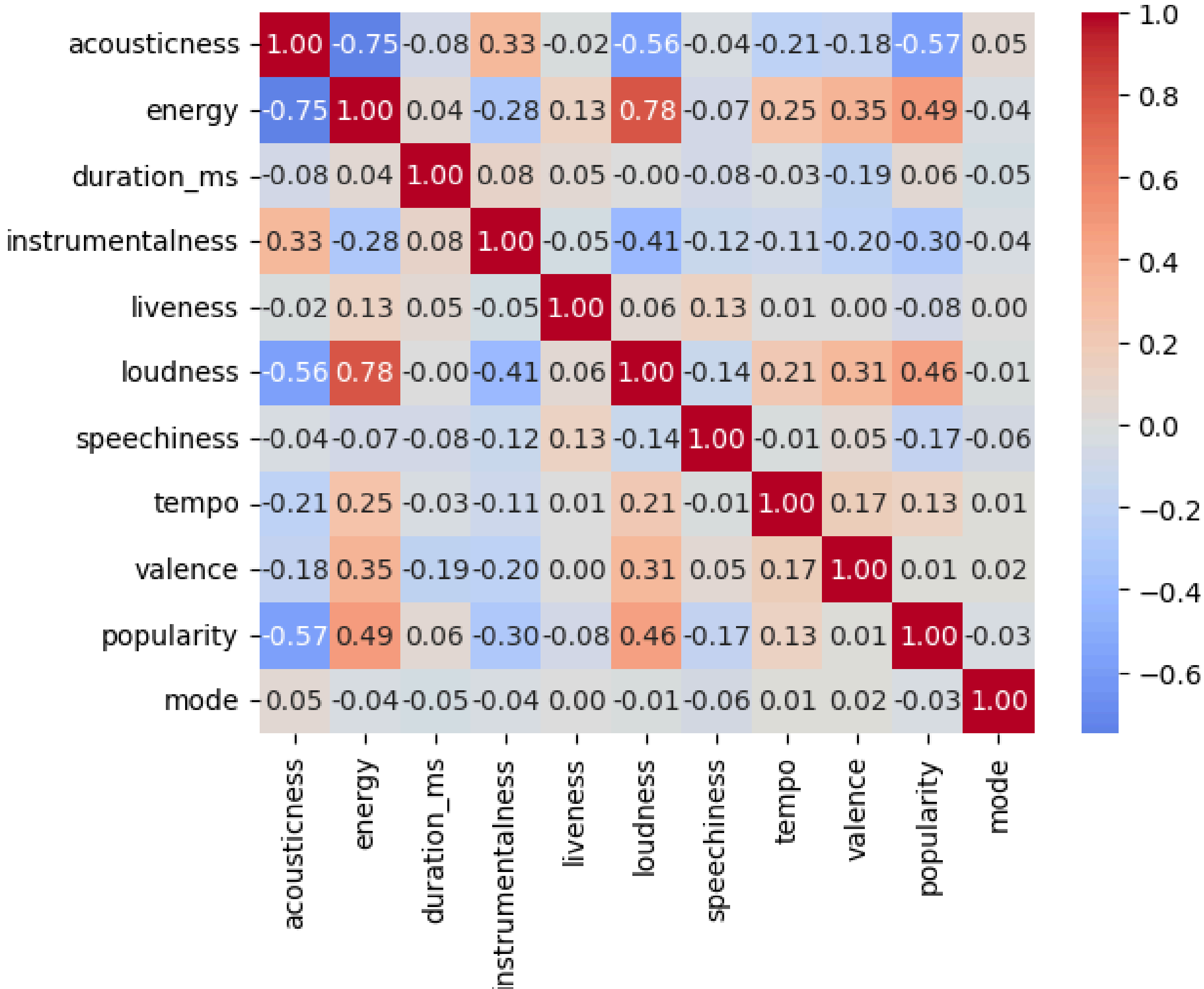
In [25]:
```
features = [
    'danceability',  'energy']
f_numeric = Data[features]

correlation_matrix = f_numeric.corr()
```

In [26]:
```
plt.figure(figsize=(6, 5))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', center
plt.title('Correlation Matrix of Music Features')
plt.show()
```



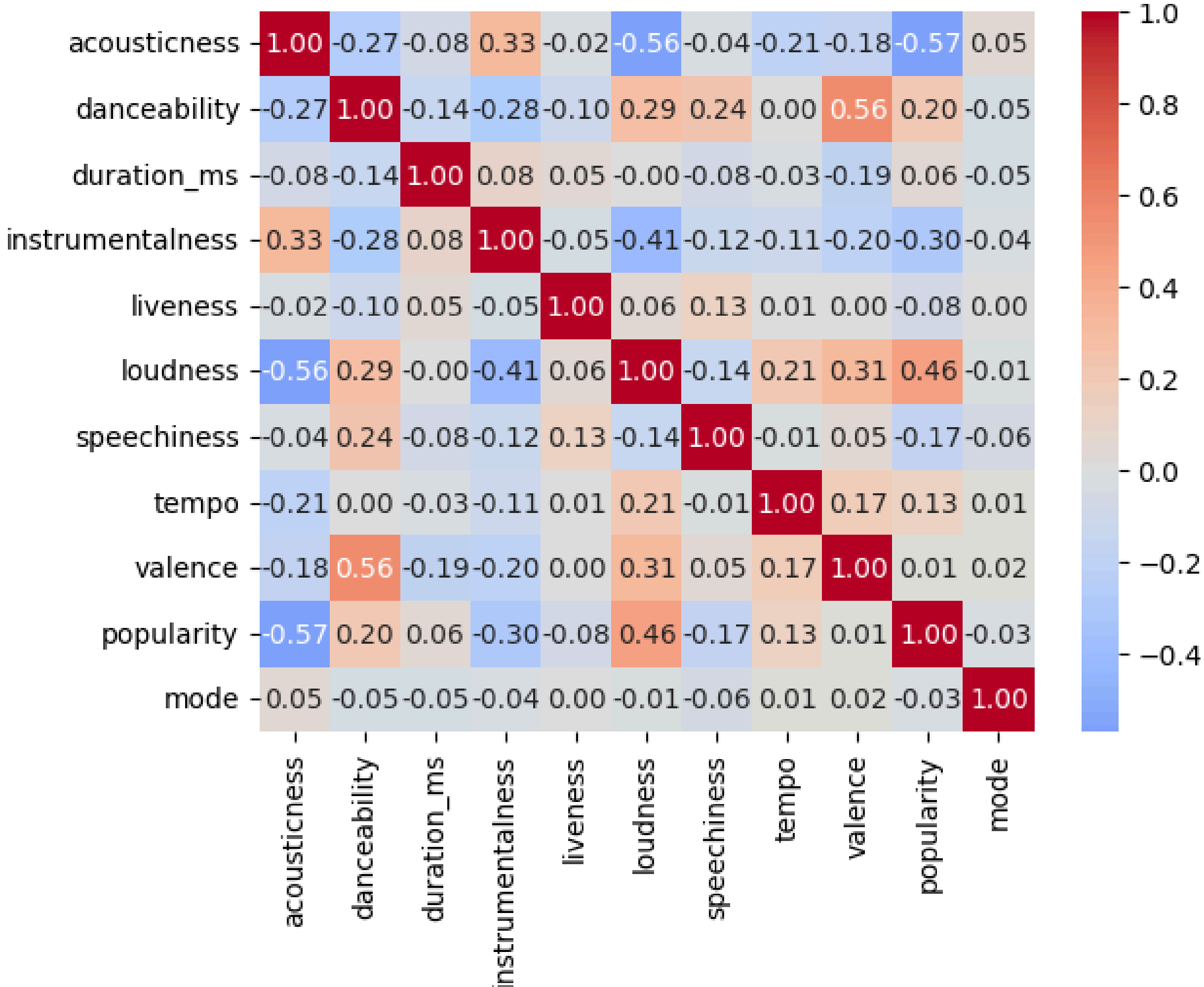
In [27]:
```
sns.heatmap(correlation_energy, annot=True, fmt=".2f", cmap='coolwarm', center
```

Out[27]: <Axes: >



In [28]:
```
sns.heatmap(correlation_danceability, annot=True, fmt=".2f", cmap='coolwarm',
```

Out[28]: <Axes: >



Visualization

Create visualizations to represent key findings (e.g., bar charts, line graphs, scatter plots).

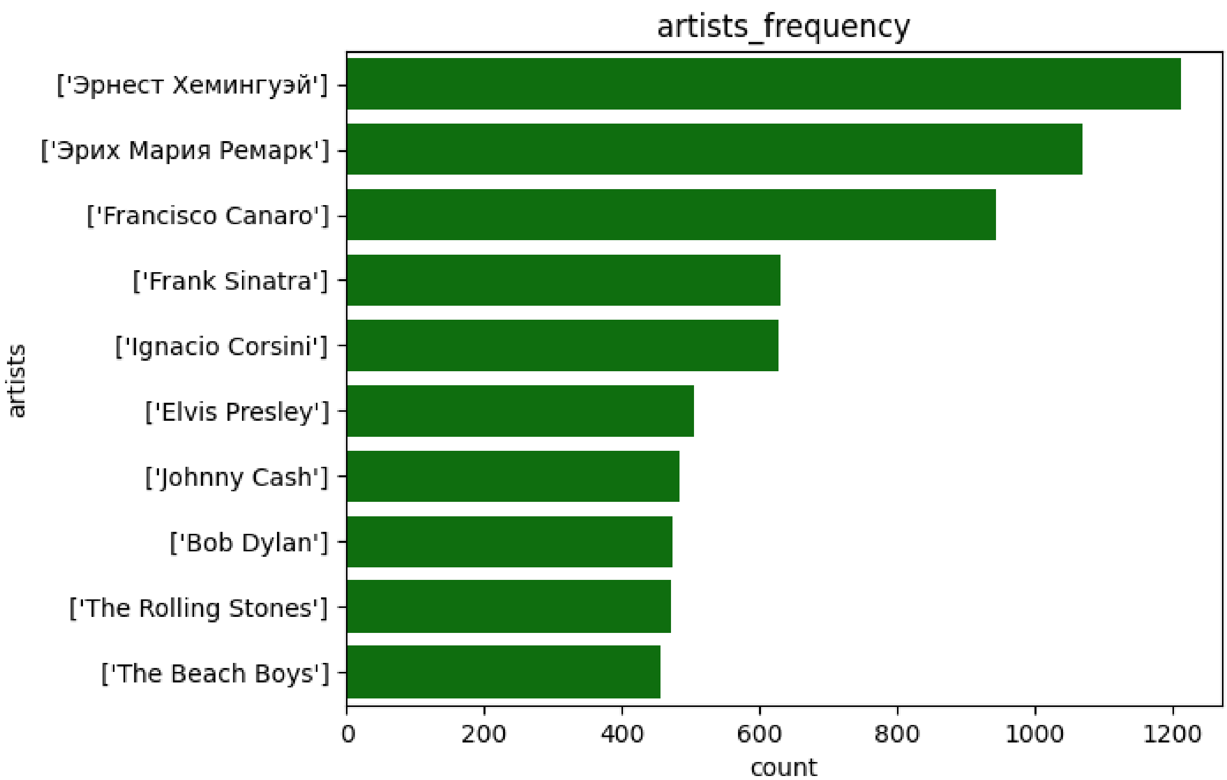
In [29]:
```
df=Data['artists'].value_counts().reset_index()
df = df.head(10)
df
```

Out[29]:

	artists	count
0	['Эрнест Хемингуэй']	1211
1	['Эрих Мария Ремарк']	1068
2	['Francisco Canaro']	942
3	['Frank Sinatra']	630
4	['Ignacio Corsini']	628
5	['Elvis Presley']	504
6	['Johnny Cash']	484
7	['Bob Dylan']	474
8	['The Rolling Stones']	471
9	['The Beach Boys']	455

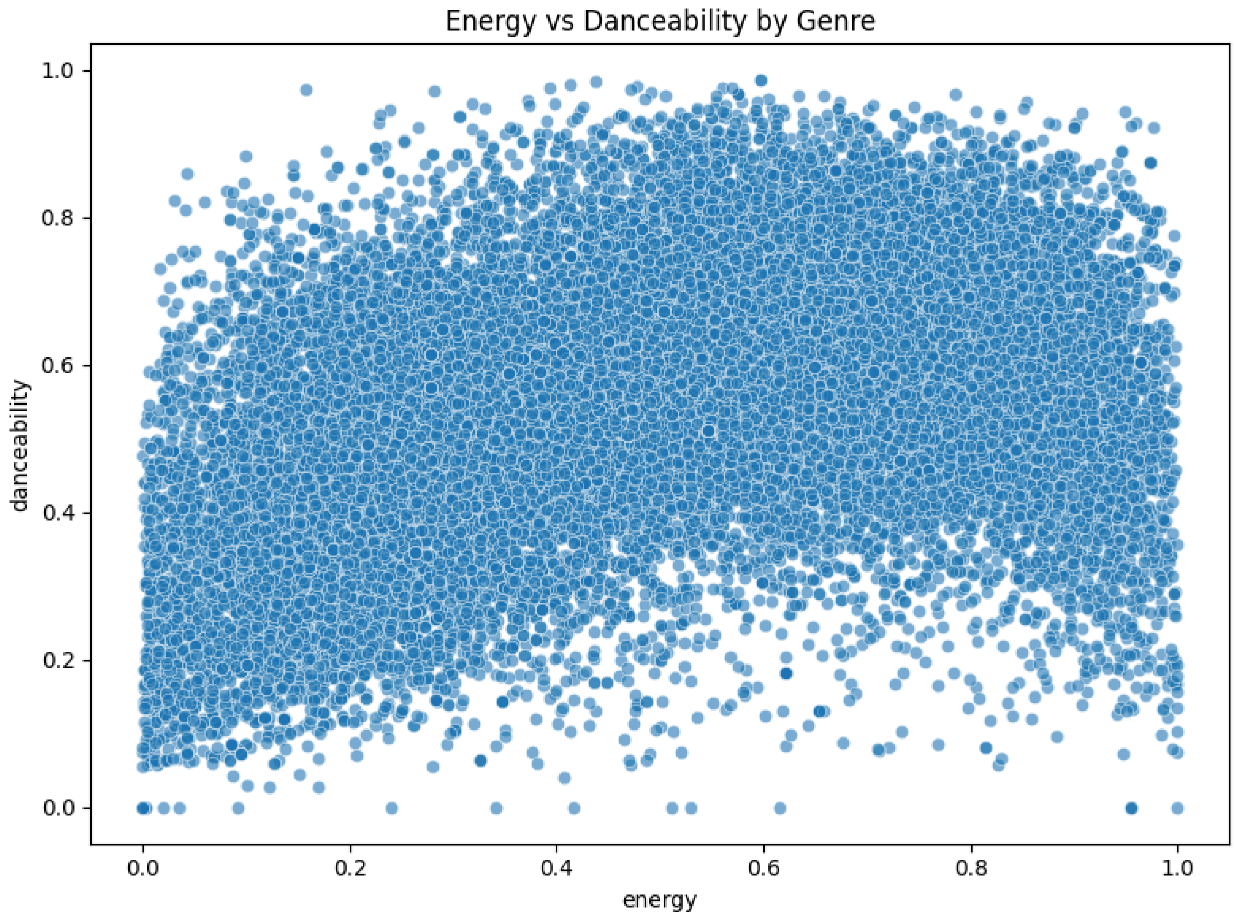
In [30]:

```
sns.barplot(y='artists',x="count",color='green',data=df)
plt.title('artists_frequency')
plt.show()
```

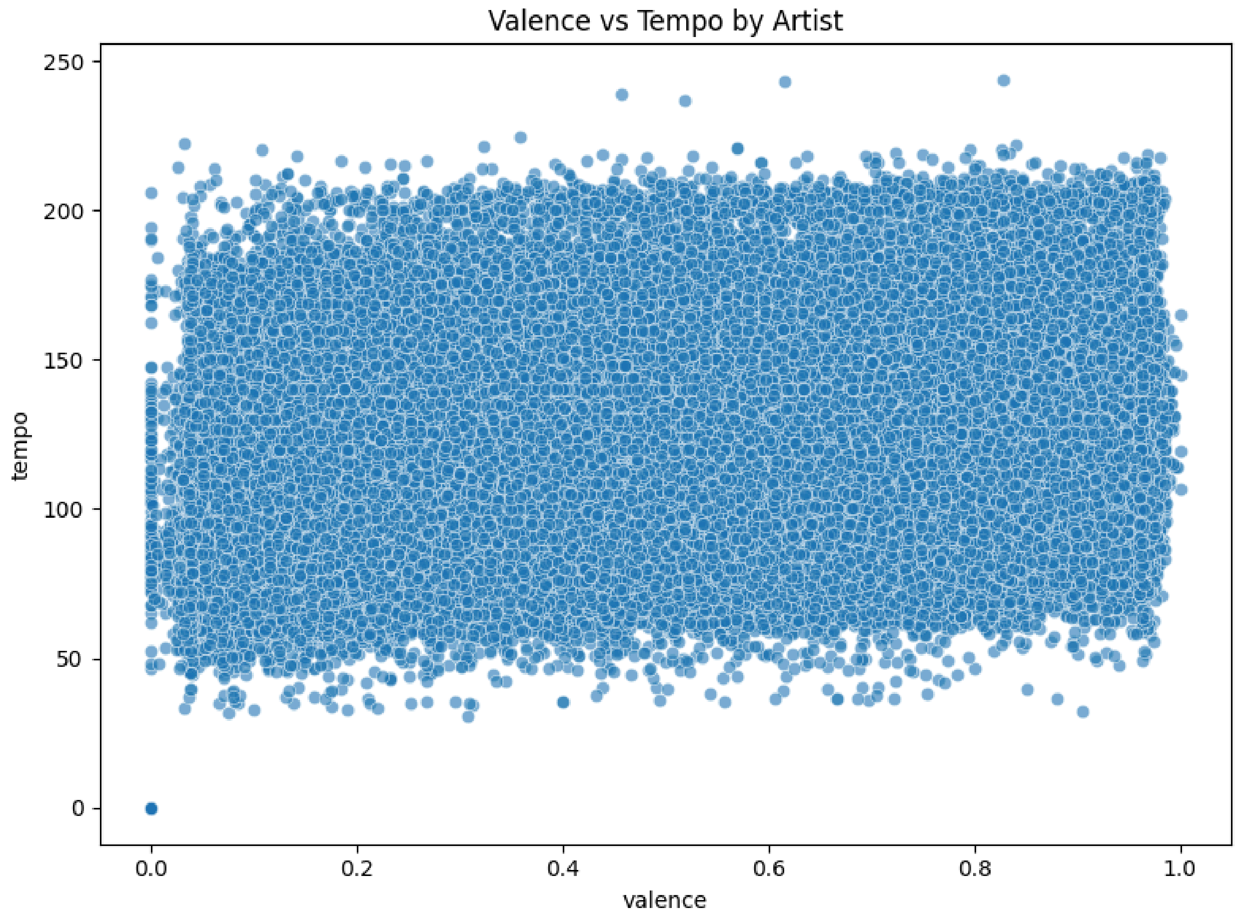


In [31]:

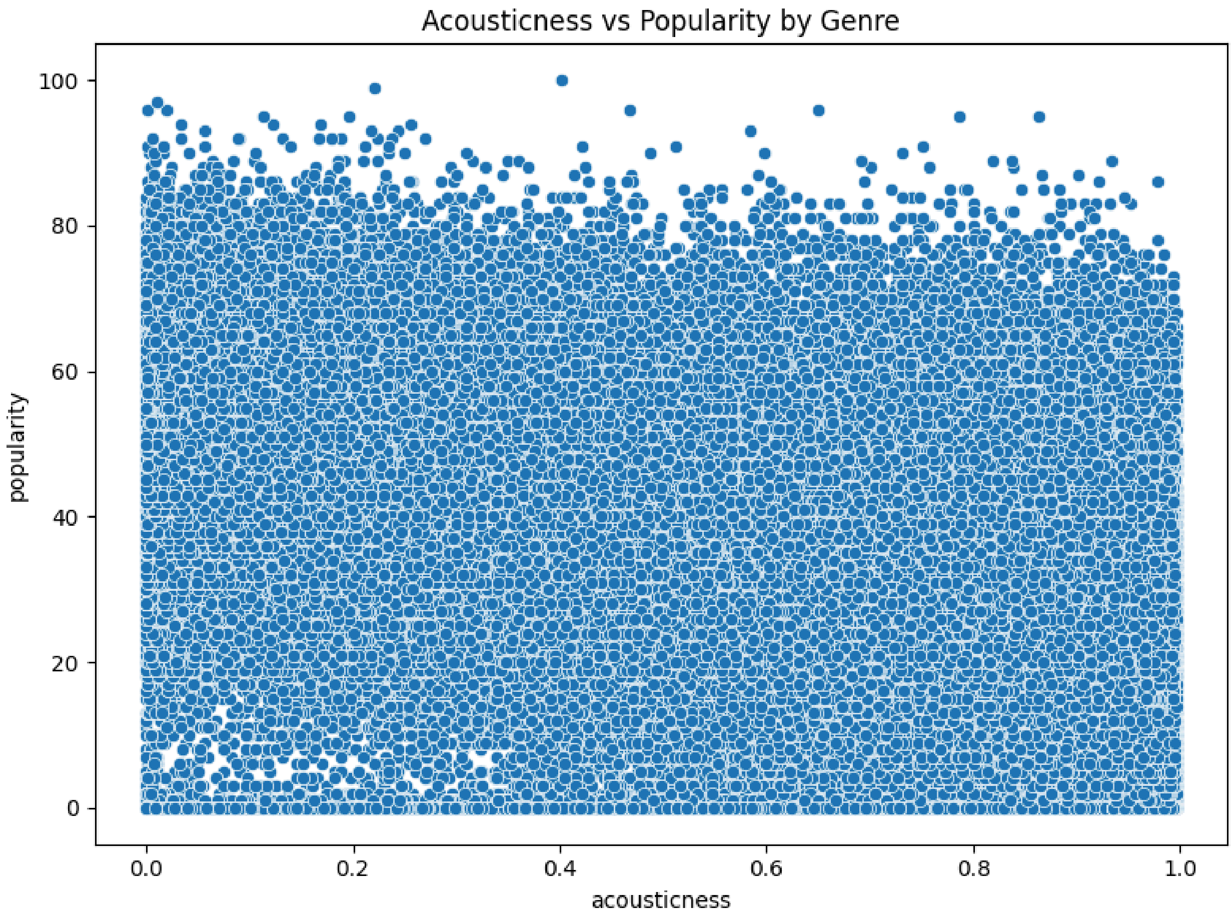
```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=GENRES, x='energy', y='danceability', alpha=0.6)
plt.title('Energy vs Danceability by Genre')
plt.tight_layout()
plt.show()
```



In [32]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=Data, x='valence', y='tempo', alpha=0.6)
plt.title('Valence vs Tempo by Artist')
plt.tight_layout()
plt.show()
```



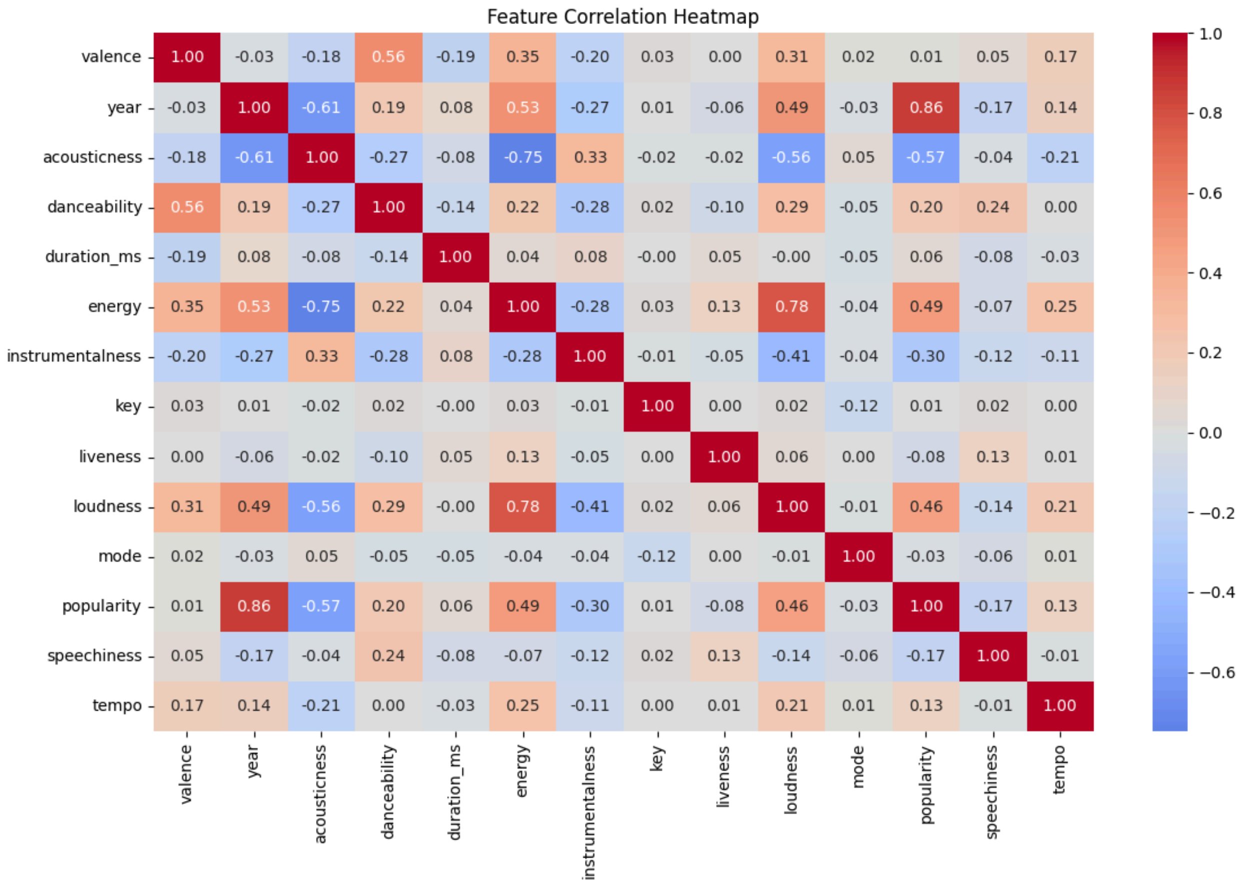
In [33]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=Data, x='acousticness', y='popularity',  alpha=1)
plt.title('Acousticness vs Popularity by Genre')
plt.tight_layout()
plt.show()
```



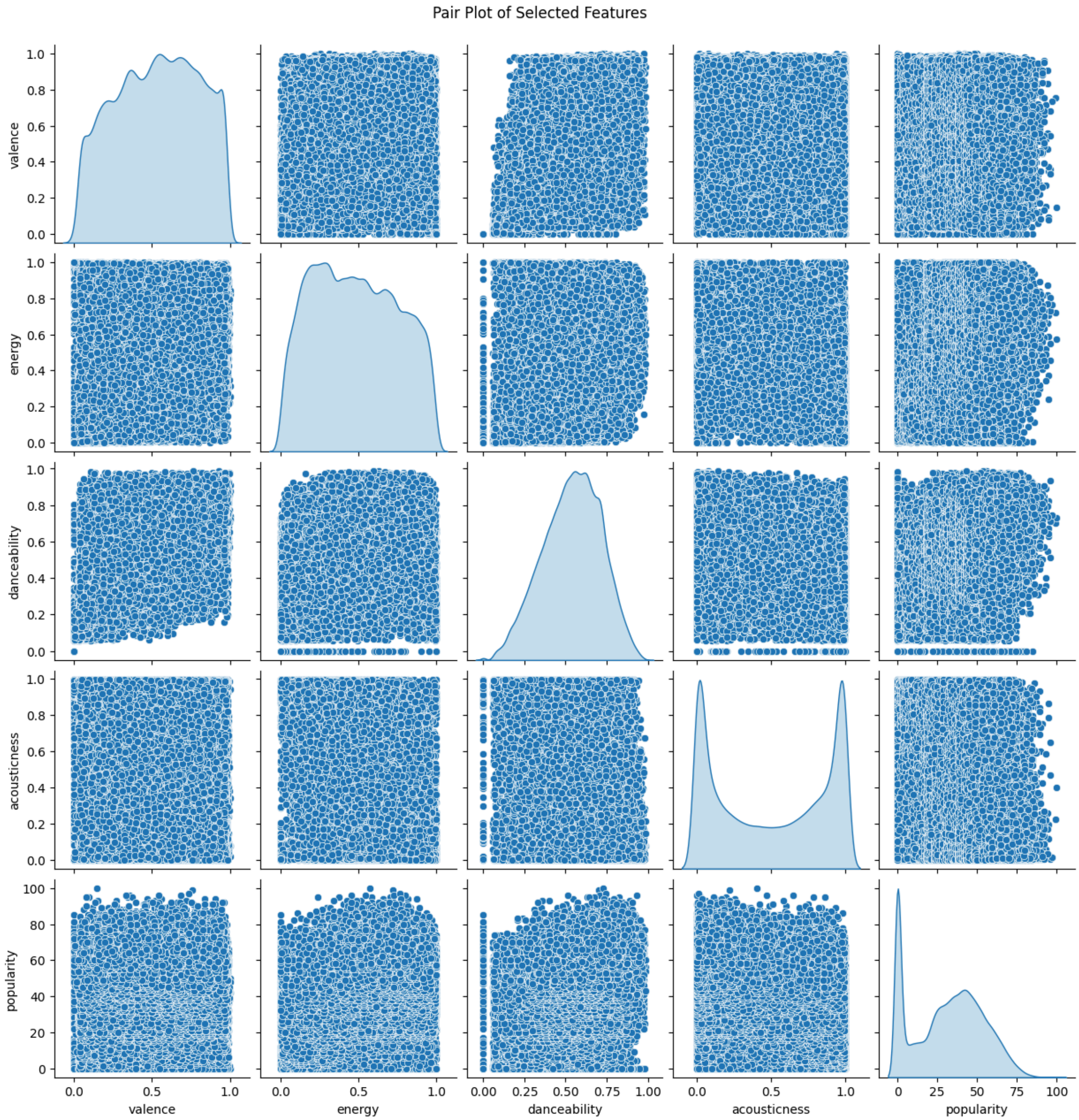
Use advanced visualization techniques (e.g., heatmaps, pair plots) to uncover deeper insights.

In [34]:

```
# Select only numeric columns
numeric_cols = ['valence', 'year', 'acousticness', 'danceability', 'duration_m
                'energy', 'instrumentalness', 'key', 'liveness', 'loudness',
                'mode', 'popularity', 'speechiness', 'tempo']

# Compute correlation matrix
corr_matrix = Data[numeric_cols].corr()

# Plot heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', center=0)
plt.title('Feature Correlation Heatmap')
plt.tight_layout()
plt.show()
```



In [35]:
```
selected_features = ['valence', 'energy', 'danceability', 'acousticness', 'pop

sns.pairplot(Data[selected_features], diag_kind='kde')
plt.suptitle('Pair Plot of Selected Features', y=1.02)
plt.show()
```



Modeling and Predictions

Build predictive models to forecast song popularity.

Evaluate the performance of different models (e.g., linear regression, decision trees).

Task 5.3: Fine-tune models to improve accuracy.

In [36]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
```

In [37]:
```
features = ['valence', 'energy', 'danceability', 'acousticness', 'tempo', 'spe
X = Data[features]
y = Data['popularity']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

In [38]:
```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor

lr_model = LinearRegression()
dt_model = DecisionTreeRegressor()

lr_model.fit(X_train, y_train)
dt_model.fit(X_train, y_train)
```

Out[38]:

▾ DecisionTreeRegressor

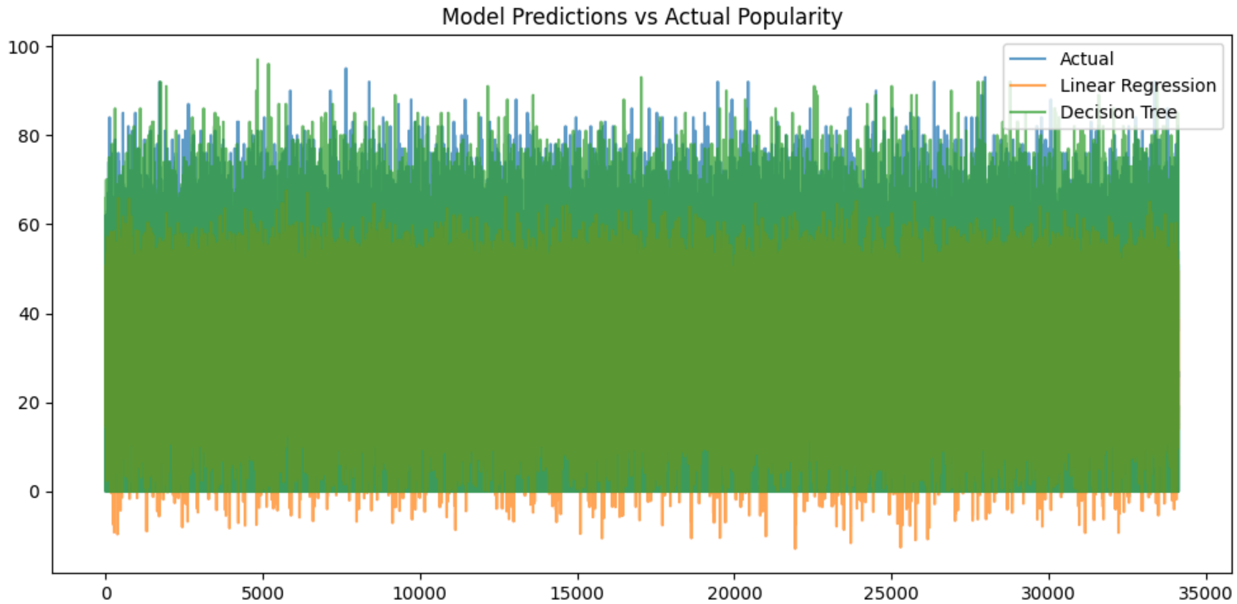
Parameters

	criterion	'squared_error'
	splitter	'best'
	max_depth	None
	min_samples_split	2
	min_samples_leaf	1
	min_weight_fraction_leaf	0.0
	max_features	None
	random_state	None
	max_leaf_nodes	None
	min_impurity_decrease	0.0
	ccp_alpha	0.0
	monotonic_cst	None

In [39]:
```
lr_preds = lr_model.predict(X_test)
dt_preds = dt_model.predict(X_test)
```

In [40]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
plt.plot(y_test.values, label='Actual', alpha=0.7)
plt.plot(lr_preds, label='Linear Regression', alpha=0.7)
plt.plot(dt_preds, label='Decision Tree', alpha=0.7)
plt.legend()
```

```
plt.title('Model Predictions vs Actual Popularity')
plt.tight_layout()
plt.show()
```



Conclusion

Positive

show a high positive correlation, suggesting that louder tracks tend to be more energetic.

moderately correlated, indicating that happier-sounding tracks may also be more danceable.

Negative

correlated with both energy and loudness, implying that acoustic tracks are generally softer and less energetic.

popularity, suggesting instrumental tracks may be less favored by mainstream listeners

In [ ]:
```

```