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

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

In [2]:
```
df = pd.read_csv("song_data.csv")
df.head()
```

Out[2]:

	song_name	song_popularity	song_duration_ms	acousticness	danceability	energy	instrumenta
0	Boulevard of Broken Dreams	73	262333	0.005520	0.496	0.682	0.0
1	In The End	66	216933	0.010300	0.542	0.853	0.0
2	Seven Nation Army	76	231733	0.008170	0.737	0.463	0.4
3	By The Way	74	216933	0.026400	0.451	0.970	0.0
4	How You Remind Me	56	223826	0.000954	0.447	0.766	0.0

In [3]:
```
df.shape
```

Out[3]:
```
(18835, 15)
```

Data Cleaning

In [4]:
```
df.isna().sum()
```

Out[4]:
```
song_name           0
song_popularity     0
song_duration_ms    0
acousticness        0
danceability        0
energy              0
instrumentalness    0
key                 0
liveness            0
loudness            0
audio_mode          0
speechiness         0
tempo               0
time_signature      0
audio_valence       0
dtype: int64
```

Exploiratory Data Analysis

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18835 entries, 0 to 18834
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   song_name         18835 non-null  object 
 1   song_popularity   18835 non-null  int64  
 2   song_duration_ms  18835 non-null  int64  
 3   acousticness      18835 non-null  float64
 4   danceability      18835 non-null  float64
 5   energy            18835 non-null  float64
 6   instrumentalness  18835 non-null  float64
 7   key               18835 non-null  int64  
 8   liveness          18835 non-null  float64
 9   loudness          18835 non-null  float64
 10  audio_mode        18835 non-null  int64  
 11  speechiness       18835 non-null  float64
 12  tempo             18835 non-null  float64
 13  time_signature    18835 non-null  int64  
 14  audio_valence     18835 non-null  float64
dtypes: float64(9), int64(5), object(1)
memory usage: 2.2+ MB
```

In [6]:
```
df.describe()
```

Out[6]:

	song_popularity	song_duration_ms	acousticness	danceability	energy	instrumentalr
count	18835.000000	1.883500e+04	18835.000000	18835.000000	18835.000000	18835.000
mean	52.991877	2.182116e+05	0.258539	0.633348	0.644995	0.078
std	21.905654	5.988754e+04	0.288719	0.156723	0.214101	0.22
min	0.000000	1.200000e+04	0.000001	0.000000	0.001070	0.000
25%	40.000000	1.843395e+05	0.024100	0.533000	0.510000	0.000
50%	56.000000	2.113060e+05	0.132000	0.645000	0.674000	0.000
75%	69.000000	2.428440e+05	0.424000	0.748000	0.815000	0.002
max	100.000000	1.799346e+06	0.996000	0.987000	0.999000	0.997

In [7]:
```
df["popularity"] = [ 1 if i>=66.5 else 0 for i in df.song_popularity ]
df["popularity"].value_counts()
```

Out[7]:
```
0    13386
1     5449
Name: popularity, dtype: int64
```

- Checking the popularity rating of songs that have been popular in the last 10 years on Spotify and took the mean value of them (66.5) . According to this value, the songs that has rating above this value and could remain on the top of the lists for a long time. If song_popularity is

higher than 66.5 (this is about 30% percent of data) we labeled it "1" and if it is not we labeled it "0". So we have "1" for the popular songs and "0" for the unpopular ones.

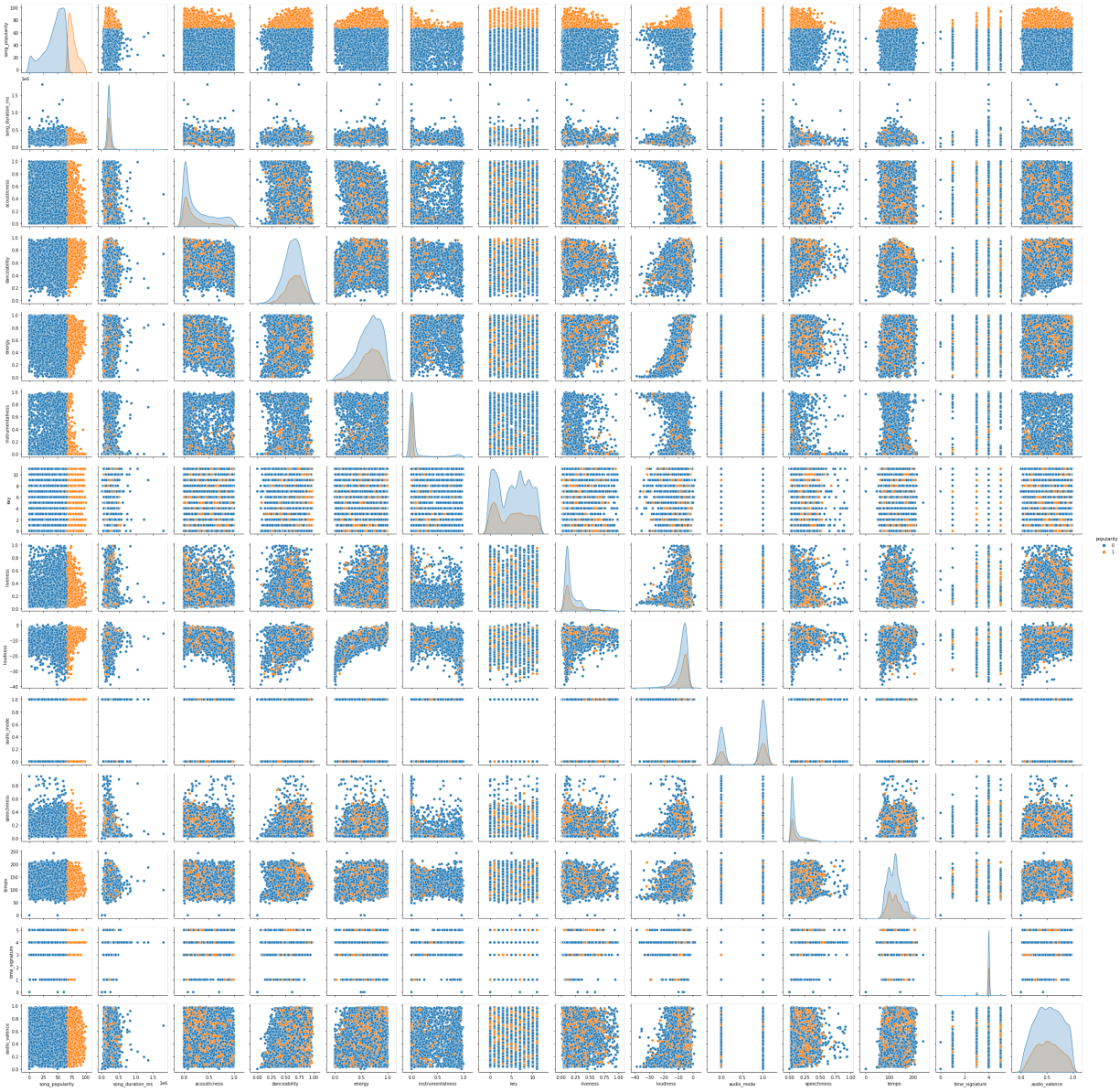
In [8]:
```
#getting the data of the popular songs
a=df[df["popularity"]==1]
a.describe()
```

Out[8]:

	song_popularity	song_duration_ms	acousticness	danceability	energy	instrumentalnes
count	5449.000000	5449.000000	5449.000000	5449.000000	5449.000000	5449.0000
mean	76.992292	218539.555515	0.210354	0.659758	0.658601	0.0223
std	8.068717	48620.048311	0.246079	0.147652	0.187495	0.1155
min	67.000000	67000.000000	0.000009	0.072200	0.002890	0.0000
25%	71.000000	190185.000000	0.026300	0.562000	0.541000	0.0000
50%	75.000000	212429.000000	0.106000	0.668000	0.680000	0.0000
75%	82.000000	240533.000000	0.300000	0.765000	0.802000	0.0001
max	100.000000	547733.000000	0.996000	0.978000	0.997000	0.9680

In [9]:
```
sns.pairplot(data=df, hue="popularity")
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x8e5bbf33a0>
```

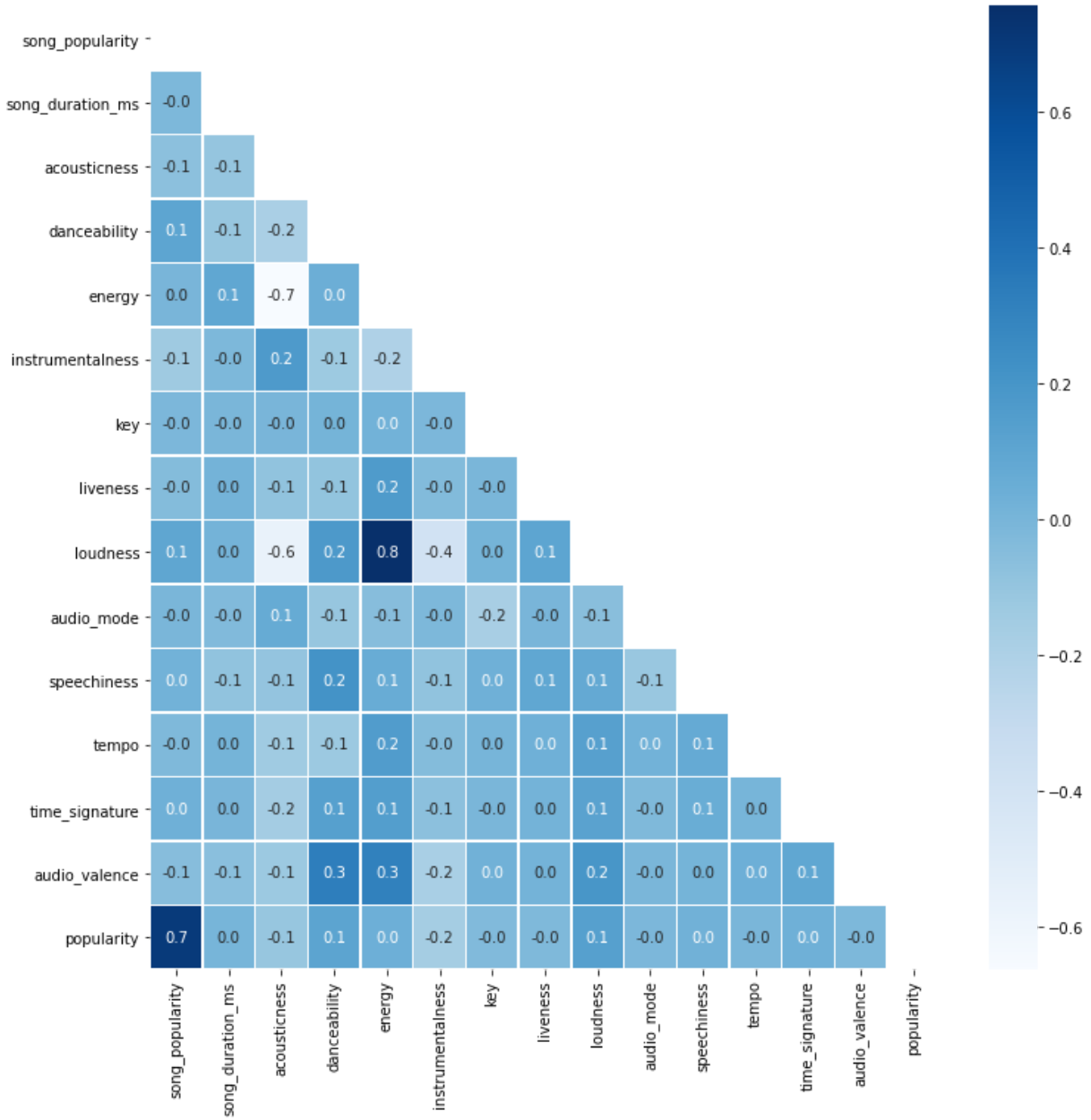


In [10]:
```
#checking for correaltion and using background colors
df.corr().style.background_gradient()
```

Out[10]:

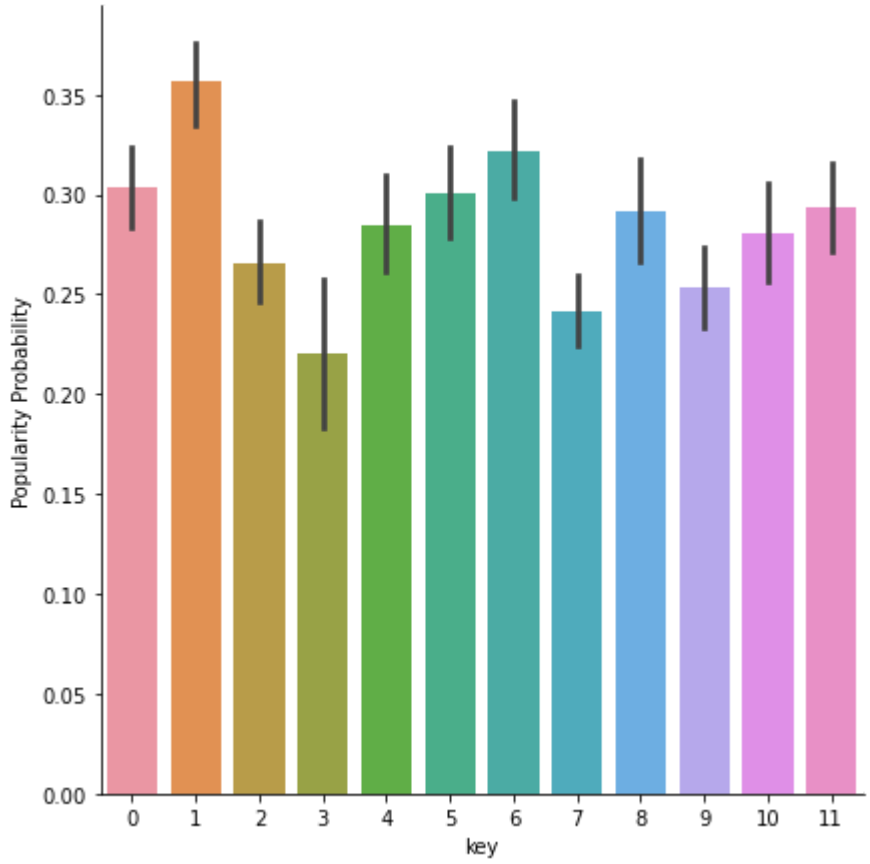
	song_popularity	song_duration_ms	acousticness	danceability	energy	instru
song_popularity	1.000000	-0.018899	-0.065181	0.104290	0.001365	
song_duration_ms	-0.018899	1.000000	-0.097882	-0.104985	0.092688	
acousticness	-0.065181	-0.097882	1.000000	-0.182500	-0.662639	
danceability	0.104290	-0.104985	-0.182500	1.000000	0.044373	
energy	0.001365	0.092688	-0.662639	0.044373	1.000000	
instrumentalness	-0.130907	-0.015188	0.173098	-0.130483	-0.205914	
key	-0.013160	-0.004615	-0.002025	0.007723	0.020416	
liveness	-0.038937	0.016086	-0.082537	-0.090694	0.167143	
loudness	0.099442	0.016469	-0.557744	0.177639	0.755516	
audio_mode	-0.004969	-0.026328	0.064100	-0.106539	-0.051605	
speechiness	0.021479	-0.083447	-0.093574	0.213728	0.062053	
tempo	-0.022672	0.012791	-0.136421	-0.121286	0.162448	
time_signature	0.034983	0.000264	-0.157177	0.136391	0.146265	
audio_valence	-0.052895	-0.063421	-0.121670	0.332014	0.316742	
popularity	0.699048	0.003494	-0.106483	0.107519	0.040548	

In [11]:
```
plt.subplots(figsize=(12, 12))
mask = np.zeros_like(df.corr())
mask[np.triu_indices_from(mask)] = True
sns.heatmap(df.corr(), annot=True, linewidths=0.4,linecolor="white", fmt= '.1f',c
plt.show()
```

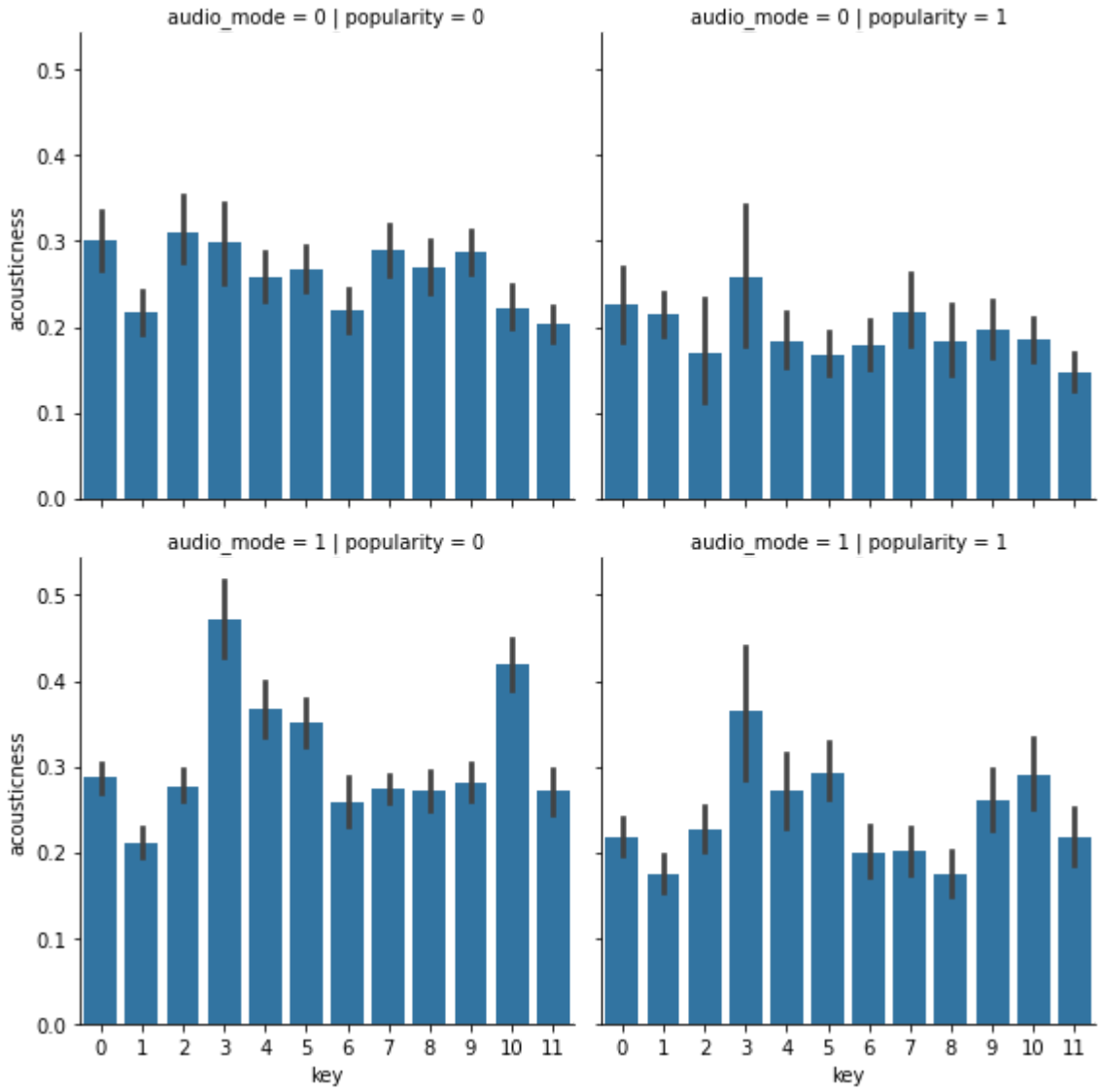


- Correlation between loudness and energy is 0.8 which is strong and correlation between loudness and accusticness is 0.6 which is moderate. Except two of them all the correlations are quite low. When we compare the correlation between song_popularity and all other features, we don't see a strong correlation (a linear relationship) that gives us a clear information about popularity. Accusticness,danceability and loudness seems to have correlation with popularity feature(0.10) and istrumentalness has 0.20.

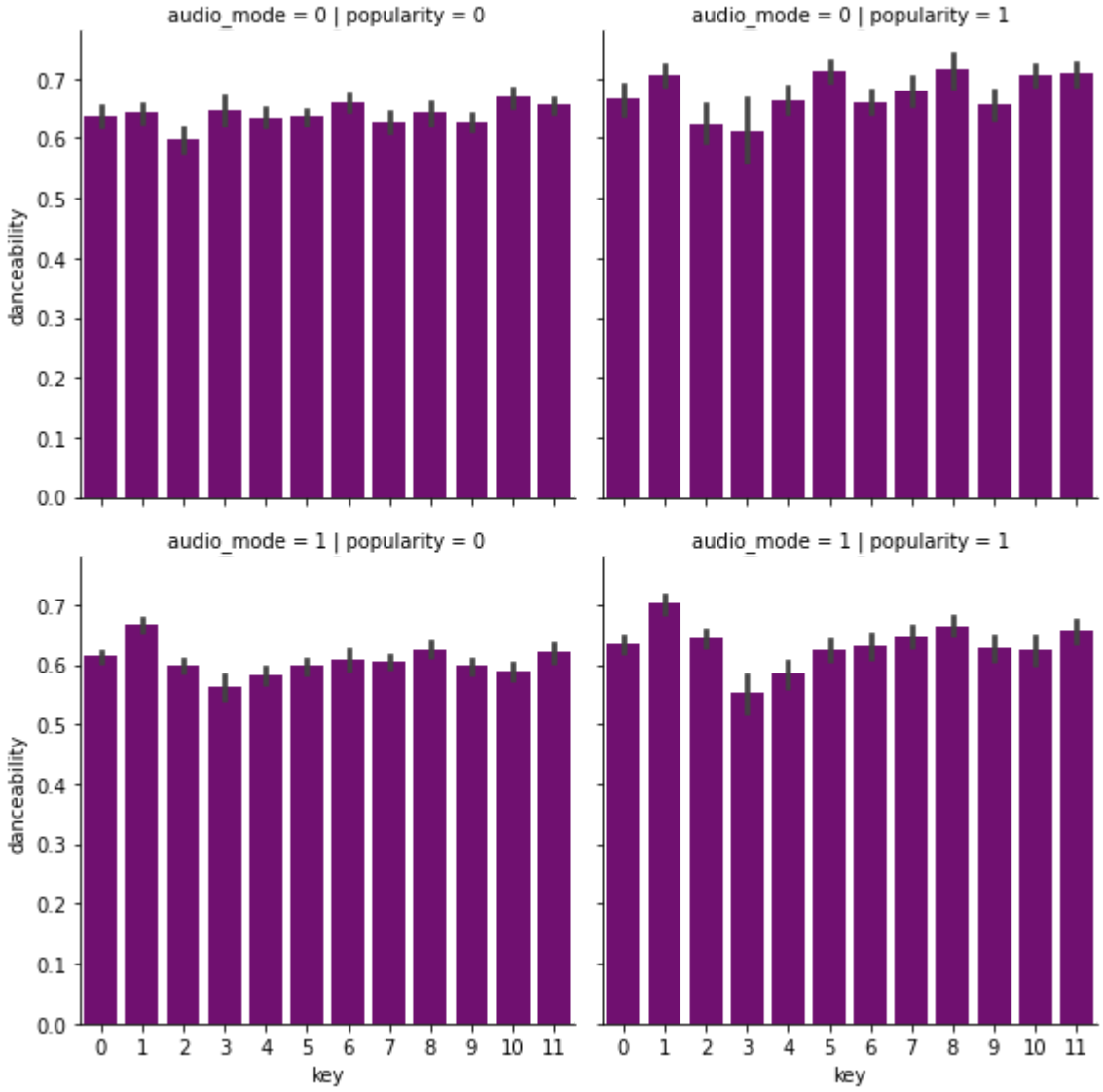
In [12]:
```
g = sns.factorplot(x = "key", y = "popularity", data = df, kind = "bar", size = 6
g.set_ylabels("Popularity Probability")
plt.show()
```



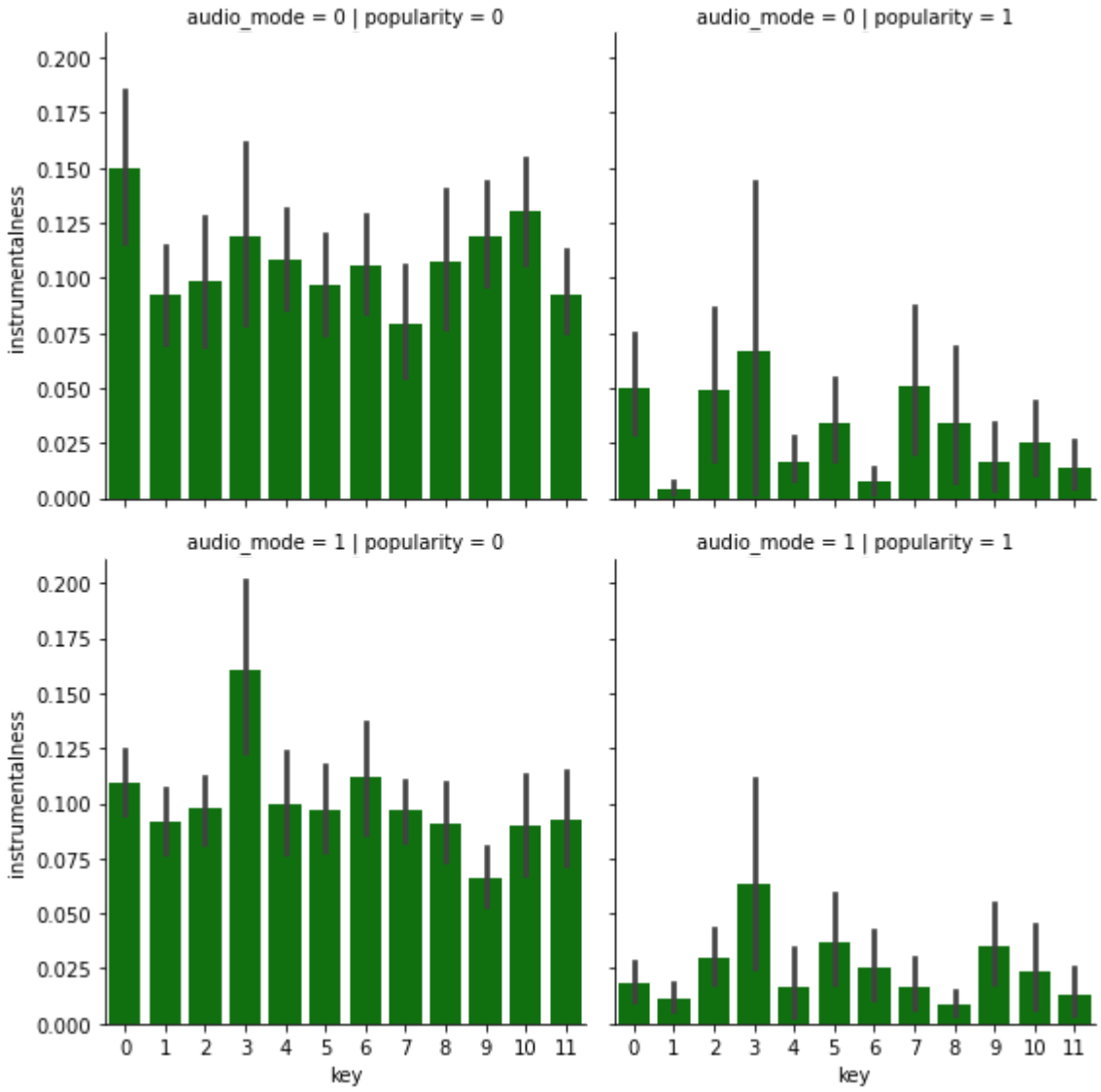
In [13]:
```
g = sns.FacetGrid(df, row = "audio_mode", col = "popularity", size = 4)
g.map(sns.barplot, "key", "acousticness")
g.add_legend()
plt.show()
```



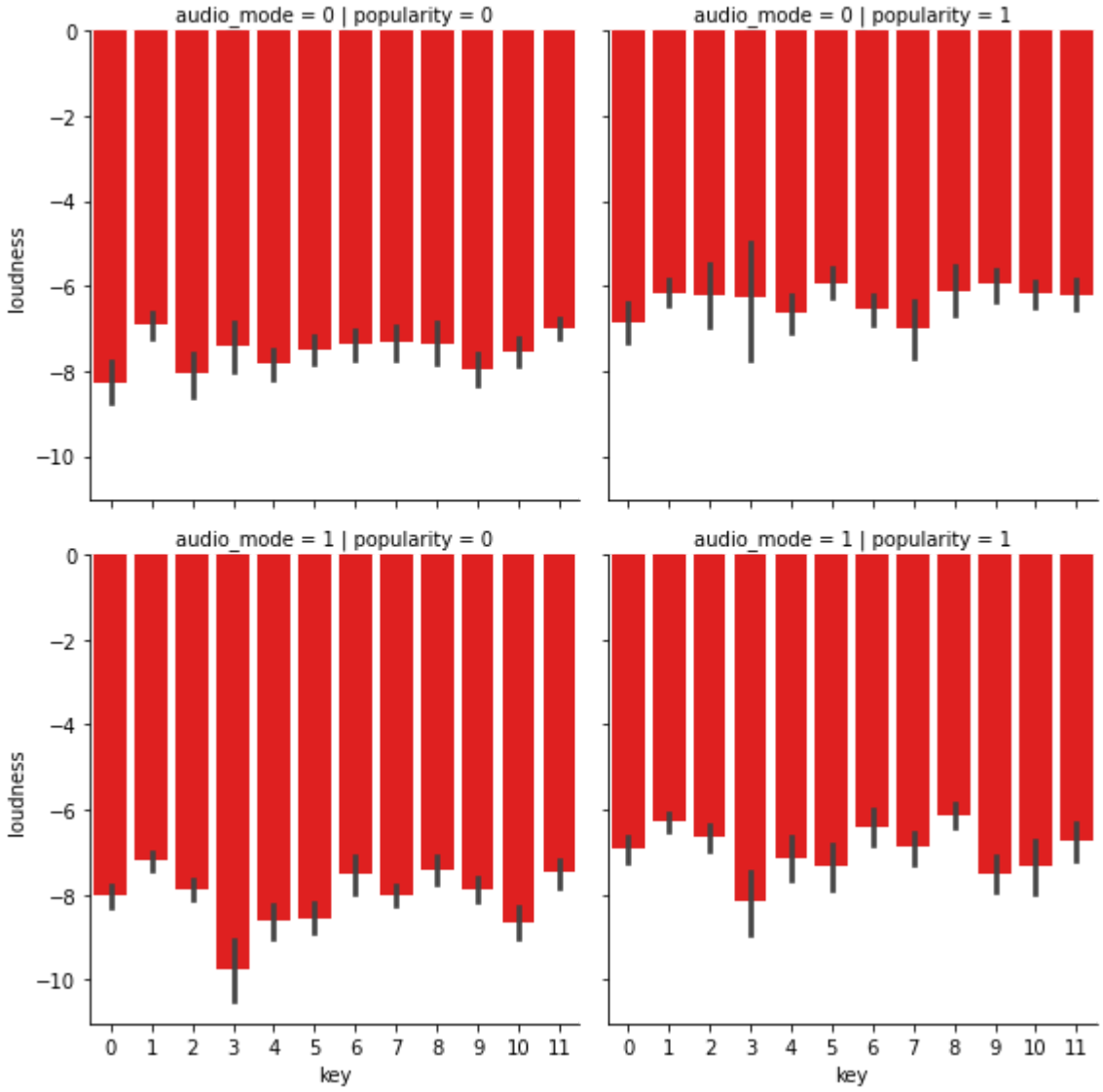
In [14]:
```
g = sns.FacetGrid(df, row = "audio_mode", col = "popularity", size = 4)
g.map(sns.barplot, "key", "danceability",color="purple")
g.add_legend()
plt.show()
```



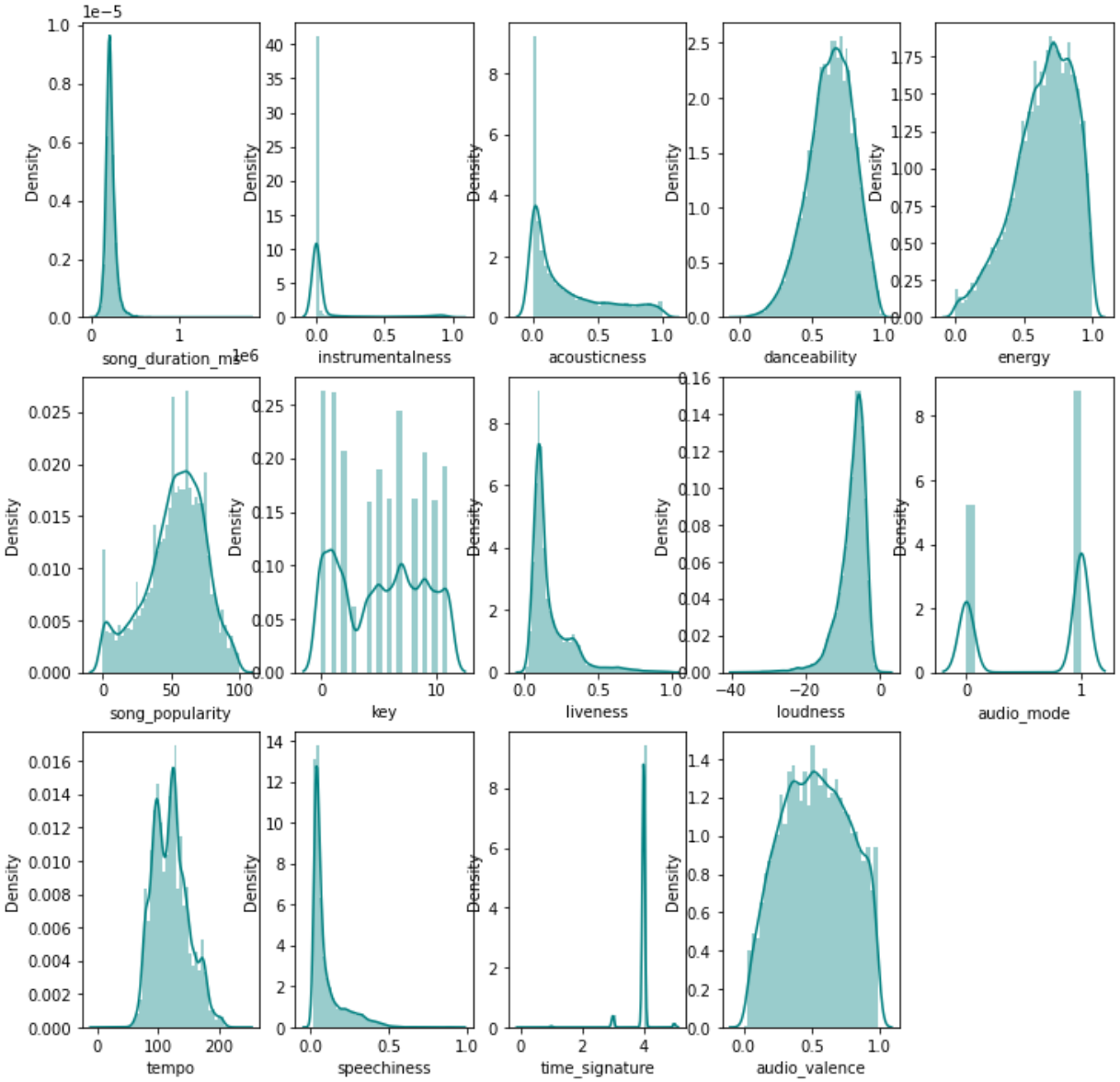
In [15]:
```
g = sns.FacetGrid(df, row = "audio_mode", col = "popularity", size = 4)
g.map(sns.barplot, "key", "instrumentalness",color="green")
g.add_legend()
plt.show()
```



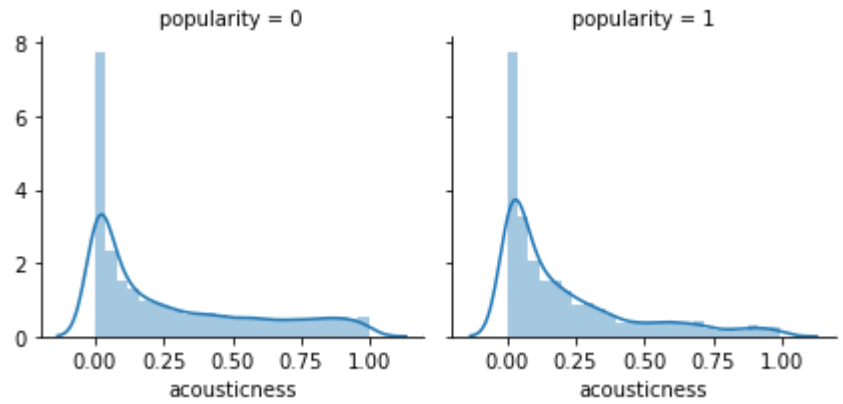
In [16]:
```
g = sns.FacetGrid(df, row = "audio_mode", col = "popularity", size = 4)
g.map(sns.barplot, "key", "loudness",color="red")
g.add_legend()
plt.show()
```



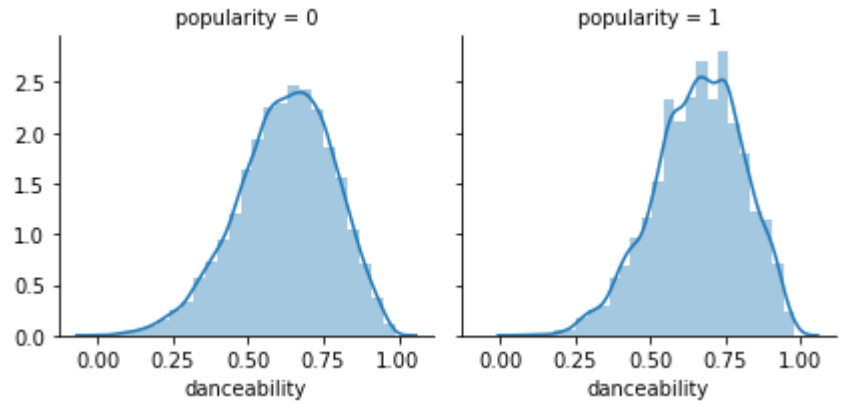
In [17]:
```
f, axes = plt.subplots(3, 5, figsize=(12, 12))
sns.distplot( df["song_duration_ms"] , color="teal", ax=axes[0, 0])
sns.distplot( df["instrumentalness"] , color="teal", ax=axes[0, 1])
sns.distplot( df["acousticness"] , color="teal", ax=axes[0, 2])
sns.distplot( df["danceability"] , color="teal", ax=axes[0, 3])
sns.distplot( df["energy"] , color="teal", ax=axes[0, 4])
sns.distplot( df["song_popularity"] , color="teal", ax=axes[1, 0])
sns.distplot( df["key"] , color="teal", ax=axes[1, 1])
sns.distplot( df["liveness"] , color="teal", ax=axes[1, 2])
sns.distplot( df["loudness"] , color="teal", ax=axes[1, 3])
sns.distplot( df["audio_mode"] , color="teal", ax=axes[1, 4])
sns.distplot( df["tempo"] , color="teal", ax=axes[2, 0])
sns.distplot( df["speechiness"] , color="teal", ax=axes[2, 1])
sns.distplot( df["time_signature"] , color="teal", ax=axes[2, 2])
sns.distplot( df["audio_valence"] , color="teal", ax=axes[2, 3])
f.delaxes(axes[2][4])
plt.show()
```



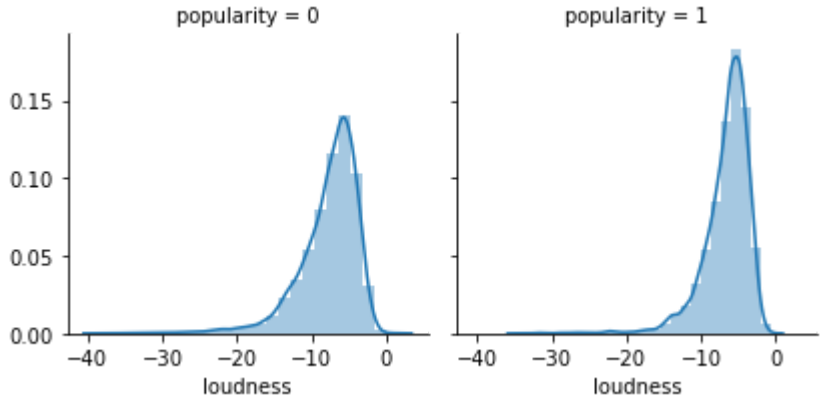
In [18]:
```
g = sns.FacetGrid(df, col = "popularity")
g.map(sns.distplot, "acousticness", bins = 25)
plt.show()
```



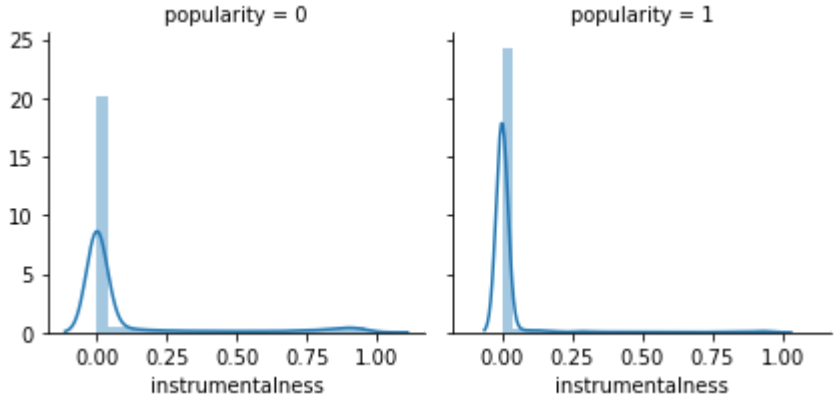
In [19]:
```
g = sns.FacetGrid(df, col="popularity")
g.map(sns.distplot, "danceability", bins = 25)
plt.show()
```



In [20]:
```
g = sns.FacetGrid(df, col = "popularity")
g.map(sns.distplot, "loudness", bins = 25)
plt.show()
```



In [21]:
```
g = sns.FacetGrid(df, col = "popularity")
g.map(sns.distplot, "instrumentalness", bins = 25)
plt.show()
```



- Data distribution of songs display today's songs features like dancebility, energy, loudness and tempo are quite high. People like fast and loud music.
- According to instrumentalness, liveness and speechness, most of the songs are not live performances and they have lyrics.
- Keys like 0,1,5,6 and 11 seems more effective in songs. And if key== 0 or 1 or 6 song has more chance to be populer.
- Time_signure is mostly 4 and 5 in both populer and general data.
- If danceability>0.6 song has more chance to be popular.
- If loudness > -10 song has more chance to be popular.

In [22]:
```
df.head()
```

Out[22]:

	song_name	song_popularity	song_duration_ms	acousticness	danceability	energy	instrumenta
0	Boulevard of Broken Dreams	73	262333	0.005520	0.496	0.682	0.0
1	In The End	66	216933	0.010300	0.542	0.853	0.0
2	Seven Nation Army	76	231733	0.008170	0.737	0.463	0.4
3	By The Way	74	216933	0.026400	0.451	0.970	0.0
4	How You Remind Me	56	223826	0.000954	0.447	0.766	0.0

In [42]:
```
df_cat = df.select_dtypes(object)
df_num = df.select_dtypes(["int64", "float64"])
#dividing the data into catagorical and numerical data
```

In [24]:
```
df_cat
```

Out[24]:

	song_name
0	Boulevard of Broken Dreams
1	In The End
2	Seven Nation Army
3	By The Way
4	How You Remind Me
...	...
18830	Let It Breathe
18831	Answers
18832	Sudden Love (Acoustic)
18833	Gentle on My Mind
18834	Up to Me

18835 rows × 1 columns

In [25]:
```
df_num
```

Out[25]:

	song_popularity	song_duration_ms	acousticness	danceability	energy	instrumentalness	k
0	73	262333	0.005520	0.496	0.682	0.000029	
1	66	216933	0.010300	0.542	0.853	0.000000	
2	76	231733	0.008170	0.737	0.463	0.447000	
3	74	216933	0.026400	0.451	0.970	0.003550	
4	56	223826	0.000954	0.447	0.766	0.000000	
...	...	...	...	...	...	...	
18830	60	159645	0.893000	0.500	0.151	0.000065	
18831	60	205666	0.765000	0.495	0.161	0.000001	
18832	23	182211	0.847000	0.719	0.325	0.000000	
18833	55	352280	0.945000	0.488	0.326	0.015700	
18834	60	193533	0.911000	0.640	0.381	0.000254	

18835 rows × 15 columns

In [26]:
```
from sklearn.preprocessing import LabelEncoder
```

In [27]:
```
for cols in df_cat:
    le = LabelEncoder()
    df_cat[cols] = le.fit_transform(df_cat[cols])
```

In [28]:
```
df_cat
```

Out[28]:

	song_name
0	1561
1	5541
2	9638
3	1760
4	4988
...	...
18830	6322
18831	687
18832	10447
18833	4131
18834	11775

18835 rows × 1 columns

In [29]:
```
df = pd.concat([df_cat,df_num], axis=1)
```

In [30]:
```
df.head()
```

Out[30]:

	song_name	song_popularity	song_duration_ms	acousticness	danceability	energy	instrumenta
0	1561	73	262333	0.005520	0.496	0.682	0.0
1	5541	66	216933	0.010300	0.542	0.853	0.0
2	9638	76	231733	0.008170	0.737	0.463	0.4
3	1760	74	216933	0.026400	0.451	0.970	0.0
4	4988	56	223826	0.000954	0.447	0.766	0.0

dividing the dataset in x and y

In [31]:
```
x = df.iloc[:, :-1]
y = df.iloc[:,-1]
```

In [32]:
```
x
```

Out[32]:

	song_name	song_popularity	song_duration_ms	acousticness	danceability	energy	instrun
0	1561	73	262333	0.005520	0.496	0.682	
1	5541	66	216933	0.010300	0.542	0.853	
2	9638	76	231733	0.008170	0.737	0.463	
3	1760	74	216933	0.026400	0.451	0.970	
4	4988	56	223826	0.000954	0.447	0.766	
...	...	...	...	...	...	...	...
18830	6322	60	159645	0.893000	0.500	0.151	
18831	687	60	205666	0.765000	0.495	0.161	
18832	10447	23	182211	0.847000	0.719	0.325	
18833	4131	55	352280	0.945000	0.488	0.326	
18834	11775	60	193533	0.911000	0.640	0.381	

18835 rows × 15 columns

In [33]:
```
y
```

Out[33]:
```
0        1
1        0
2        1
3        1
4        0
        ..
18830    0
18831    0
18832    0
18833    0
18834    0
Name: popularity, Length: 18835, dtype: int64
```

creating & testing the model and aslo dividing them itno training and testing data

In [34]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.3, random_state=
```

In [35]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matr
logreg = LogisticRegression()
knn = KNeighborsClassifier()
```

In [36]:
```
def mymodel(model):
    model.fit(xtrain, ytrain)
    ypred = model.predict(xtest)
    print(classification_report(ytest, ypred))
```

In [37]:
```
mymodel(logreg)
```

```
              precision    recall  f1-score   support

           0       0.91      0.90      0.90      4045
           1       0.75      0.78      0.77      1606

    accuracy                           0.86      5651
   macro avg       0.83      0.84      0.84      5651
weighted avg       0.87      0.86      0.87      5651
```

In [38]:
```
mymodel(knn)
```

```
              precision    recall  f1-score   support

           0       0.80      0.83      0.81      4045
           1       0.52      0.47      0.50      1606

    accuracy                           0.73      5651
   macro avg       0.66      0.65      0.66      5651
weighted avg       0.72      0.73      0.72      5651
```

In [39]:
```
from sklearn.model_selection import cross_val_score

cvs = cross_val_score(logreg, x,y, cv=15, scoring="accuracy")
print(f"Avg. Accuracy-: {cvs.mean()}")
```

```
Avg. Accuracy-: 0.8714639531047782
```

HyperParameter Tuning

In [40]:
```
logreg = LogisticRegression(solver="liblinear")
logreg.fit(xtrain, ytrain)
ypred = logreg.predict(xtest)
print(accuracy_score(ytest, ypred))
print()
print(classification_report(ytest, ypred))
print()
print(confusion_matrix(ytest, ypred))
```

```
0.8635639709785878

              precision    recall  f1-score   support

           0       0.90      0.91      0.91      4045
           1       0.77      0.74      0.76      1606

    accuracy                           0.86      5651
   macro avg       0.83      0.83      0.83      5651
weighted avg       0.86      0.86      0.86      5651


[[3686  359]
 [ 412 1194]]
```

In [41]:
```
cvs = cross_val_score(logreg, x,y, cv=10, scoring="accuracy")
print(f"Avg. Accuracy-: {cvs.mean()}\nStandard Deviation -: {cvs.std()}")
```

```
Avg. Accuracy-: 0.8604216912299455
Standard Deviation -: 0.027256269642031036
```

Conclusion

- we saw that the correlation between loudness and energy was high and correlation between loudness and accousticness was moderate rest all the correlation were quite low
- also found that nowadays people like more loud and energetic songs and accorduing to these which song might be on the top list of spotify for a long time
- if the loudness of the song is greater than -10dB than there is a chance of that song getting popular in the today's generation
- and also got to know that nowadays people only listen to song that are about 3 to 4 mins long. if the song is above that time duration they simply dont listen or they just skip the song
- after all the eda and model training we get the the conclusion is that the logistic regression model gives the best accuracy for this data set.
- with the accuracy of 86% logistic regression givs the best accuracy

In [ ]: