# Song_Cluster_Analysis

February 7, 2021

# 1 Spotify Song Cluster Analysis

## 1.1 Package Installations

```
In [1]: # !pip install nbconvert
```

```
In [2]: # !pip install latex
```

```
In [3]: # !pip install plotly.express
```

```
In [4]: # !pip install pingouin
```

```
In [5]: import numpy as np
        import pandas as pd

        import re
        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
        from sklearn.metrics import silhouette_score
        from pingouin import pairwise_ttests
        import statsmodels
        from scipy import stats

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

## 1.2 Project Overview

We used song data sourced from Spotify to assess what characteristics of a song contribute to its popularity. With this knowledge, artists can be more intentional about the type of music they release to ensure that it is popular on spotify. Our hypothesis being that certain audio features are more common in popular songs. The strongest relationship we found in the data was recency, or the year the song was released, but other audio features also contributed significantly to song popularity, those features being duration, speechiness, loudness, and valence. Since we found shorter durations and increased positivity to be contributing when songs were analyzed individually and as clusters, we would give the most attention to those features when creating a song, however all of the aforementioned were significantly associated.

## 1.3 Project Background

What makes up a popular song? We set out to analyze the components of popular songs on Spotify to see what music trends existed and how they have changed over time. Using the information we gather, we could make predictions about how popular a song could be expected to be given its audio features.

The nature of the data took our analysis down another path, that is, to examine how popular songs were when the data for our project was sourced and look for commonalities between them in terms of audio features to determine what types of songs were most popular. This information can still be used by artist to curate songs with characteristics similar to the most popular songs which they might be reasonably able to expect to be independently popular **and/or** become popular after being recommended to a listener by a recommendation engine given their similar audio profiles.

## 1.4 Data Details

The dataset contains a nearly 14k song subset of a Kaggle dataset sourced from Spotify's web API. Songs in the dataset were released between 2014 and 2020 . Spotify songs are rated for their audio features which help with create recommendations of songs a user may like based on their current selection. These audio features are included as variables for the songs in our dataset.

**Primary:** - **id**: Unique track id assigned by Spotify

**Numerical:** - **acousticness**: 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**: Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
- **duration_ms**: The duration of the track in milliseconds; typically ranging from 200k to 300k.
- **energy**: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy
- **instrumentalness**: Predicts whether a track contains no vocals. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
- **liveness**: Detects the presence of an audience in the recording.

- **loudness**: The overall loudness of a track in decibels. Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.
- **popularity** The popularity of the track. The value will be between 0 and 100, with 100 being the most popular. Based on the total number of plays and how recent the plays are.
- **speechiness**: Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording, the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
- **tempo**: The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
- **valence**: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive, while tracks with low valence sound more negative.
- **year**: The year the track was released

**Indicator:** - **mode**: Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.

- **explicit**: Whether or not the track has explicit lyrics

**Categorical:** - **key**: The key the track is in using standard Pitch Class notation.

- **artists**: Artist name
- **release_date**: Release date of the song
- **name**: Name of the song

### 1.4.1 Reading Spotify in Datasets

There were five datasets in the Kaggle dataset.

**1 - All Songs**
2 - All Songs by Artist
3 - All Songs by Genre
4 - All Songs by Year
5 - All Songs with the Genre

We decided to use the "All Songs" dataset as it offered the most expansive dataset with the most useful data. We found that the song genres were too granular for the purposes of our analysis.

In [6]:
```python
#Reading in spotify datasets limiting to 2014 and after where year is available
all_songs = pd.read_csv('data.csv')
all_songs = all_songs[all_songs['year'] > 2013]
all_songs.reset_index(inplace=True)
del all_songs['index']

#removing brackets from artists name's
all_songs['artists'] = all_songs['artists'].str.replace(r"\['","")
all_songs['artists'] = all_songs['artists'].str.replace(r"\']","")
all_songs.head()
```

Out[6]:
```
   valence  year  acousticness  \
0    0.591  2014        0.0489
1    0.463  2014        0.3010
2    0.510  2014        0.4310
3    0.584  2014        0.0751
4    0.211  2014        0.2200

                                             artists  danceability  \
0                                      Ariana Grande         0.525
1                                            J. Cole         0.692
2                                          Vance Joy         0.484
3                                            J. Cole         0.517
4  Ty Dolla $ign', 'The Weeknd', 'Wiz Khalifa', '...         0.805

   duration_ms  energy  explicit                      id  instrumentalness  \
0       204093   0.621         0  0lizgQ7Qw35od7CYaoMBZb               0.0
1       292987   0.521         1  62vpWI1CHwFy7tMIcSStl8               0.0
2       204280   0.731         0  3JvrhDOgAt6p7K8mDyZwRd               0.0
3       239320   0.705         1  6Ius4TC0L3cN74HT7ENE6e               0.0
4       242983   0.330         1  7t2bFihaDvhIrd2gn2CWJO               0.0

   key  liveness  loudness  mode  \
0    7    0.2940    -7.364     1
1   10    0.0565    -8.465     0
2    1    0.1510    -6.694     1
3    6    0.1280    -8.205     0
4    1    0.1050    -8.712     0

                                                name  popularity release_date  \
0                                      Santa Tell Me          86   2014-11-24
1                                     No Role Modelz          84   2014-12-09
2                                            Riptide          78   2014-09-09
3                                         Wet Dreamz          79   2014-12-09
4  Or Nah (feat. The Weeknd, Wiz Khalifa & DJ Mus...          80   2014-06-10

   speechiness    tempo
0       0.1160  191.900
```

```
1       0.3300  100.450
2       0.0379  101.654
3       0.3640  175.906
4       0.1000  121.970
```

```
In [7]: #Validating that we only have 2014 through 2020 data
        all_songs['year'].unique()
```

```
Out[7]: array([2014, 2015, 2016, 2017, 2018, 2019, 2020])
```

# 2 Data Exploration & Cleaning

```
In [8]: #Checking each dataframe for null values and printing descriptive statistics
        print(all_songs.isna().sum())
```

```
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
```

### 2.0.1 Descriptive Statistics

```
In [9]: all_songs.describe()
```

```
Out[9]:           valence          year  acousticness  danceability    duration_ms  \
        count  13850.000000  13850.000000  13850.000000  13850.000000   13850.000000
        mean       0.450527   2017.023899      0.262342      0.629006  213673.695523
        std        0.234194      2.009061      0.285212      0.170337   61273.411870
        min        0.000000   2014.000000      0.000000      0.000000   30579.000000
        25%        0.268000   2015.000000      0.031100      0.526000  179892.500000
        50%        0.437000   2017.000000      0.144000      0.645000  208073.500000
```

```
75%        0.620000   2019.000000      0.417750      0.753000  238133.000000
max        0.993000   2020.000000      0.996000      0.985000  875307.000000

             energy     explicit  instrumentalness           key  \
count  13850.000000  13850.00000      13850.000000  13850.000000
mean       0.612610      0.35148          0.074021      5.220000
std        0.210238      0.47745          0.226839      3.598993
min        0.000020      0.00000          0.000000      0.000000
25%        0.487000      0.00000          0.000000      2.000000
50%        0.633000      0.00000          0.000001      5.000000
75%        0.768000      1.00000          0.000424      8.000000
max        1.000000      1.00000          1.000000     11.000000

           liveness      loudness          mode    popularity   speechiness  \
count  13850.000000  13850.000000  13850.000000  13850.000000  13850.000000
mean       0.182998     -7.493299      0.623321     61.178917      0.112462
std        0.148379      4.581983      0.484571     15.819325      0.114144
min        0.000000    -54.837000      0.000000      0.000000      0.000000
25%        0.097300     -8.514750      0.000000     57.000000      0.039000
50%        0.122000     -6.486500      1.000000     63.000000      0.060700
75%        0.217000     -4.994000      1.000000     69.000000      0.141000
max        0.987000      1.023000      1.000000    100.000000      0.918000

              tempo
count  13850.000000
mean     120.725859
std       30.588292
min        0.000000
25%       97.011250
50%      120.076000
75%      140.793500
max      220.099000
```

Fortunately, our datset was pretty clean as it was. All variables were complete with no NA or missing values. The minimum and maximum values also fell within the expected ranges as per the documentation.

#### 2.0.2 Finding Popular Songs

```
In [10]: print('The threshold for the 75th percentile of popularirty is a popularity rating of
```

```
The threshold for the 75th percentile of popularirty is a popularity rating of  69.0 and the t
```

```
In [11]: all_songs[all_songs['popularity']>=69].describe()
```

```
Out[11]:            valence          year  acousticness  danceability    duration_ms  \
         count  3853.000000   3853.000000   3853.000000   3853.000000    3853.000000
         mean      0.473454   2018.257462      0.255294      0.664993  203231.753439
```

```
std       0.226407     1.717193      0.261291      0.151925   46421.185487
min       0.000000  2014.000000      0.000000      0.000000   57074.000000
25%       0.298000  2017.000000      0.042100      0.573000  176547.000000
50%       0.463000  2019.000000      0.162000      0.682000  201000.000000
75%       0.641000  2020.000000      0.398000      0.772000  226160.000000
max       0.980000  2020.000000      0.994000      0.980000  632625.000000

            energy     explicit  instrumentalness         key     liveness  \
count  3853.000000  3853.000000       3853.000000  3853.00000  3853.000000
mean      0.612346     0.413185          0.025658     5.25305     0.175399
std       0.182644     0.492469          0.130344     3.55962     0.136232
min       0.000020     0.000000          0.000000     0.00000     0.000000
25%       0.500000     0.000000          0.000000     2.00000     0.096200
50%       0.630000     0.000000          0.000000     5.00000     0.120000
75%       0.744000     1.000000          0.000038     8.00000     0.206000
max       1.000000     1.000000          1.000000    11.00000     0.953000

          loudness         mode   popularity  speechiness        tempo
count  3853.000000  3853.000000  3853.000000  3853.000000  3853.000000
mean     -6.744038     0.593304    74.492863     0.116558   121.279041
std       3.344613     0.491281     4.978173     0.113299    30.808471
min     -40.449000     0.000000    69.000000     0.000000     0.000000
25%      -7.833000     0.000000    71.000000     0.041700    97.054000
50%      -6.146000     1.000000    73.000000     0.067600   120.001000
75%      -4.799000     1.000000    77.000000     0.149000   142.689000
max       0.457000     1.000000   100.000000     0.894000   220.099000
```

In [12]:
```python
all_songs[all_songs['popularity']>=75].describe()
```

Out[12]:
```
           valence         year  acousticness  danceability    duration_ms  \
count  1561.000000  1561.000000   1561.000000   1561.000000    1561.000000
mean      0.473298  2018.540038      0.248980      0.676274  200430.385010
std       0.221702     1.643509      0.259878      0.141682   41655.686083
min       0.000000  2014.000000      0.000010      0.000000   78681.000000
25%       0.305000  2018.000000      0.040600      0.589000  175918.000000
50%       0.463000  2019.000000      0.153000      0.692000  199387.000000
75%       0.640000  2020.000000      0.388000      0.774000  221429.000000
max       0.969000  2020.000000      0.985000      0.980000  467587.000000

            energy     explicit  instrumentalness          key     liveness  \
count  1561.000000  1561.000000       1561.000000  1561.000000  1561.000000
mean      0.613942     0.404228          0.016379     5.312620     0.166191
std       0.176604     0.490899          0.099445     3.580882     0.121679
min       0.000020     0.000000          0.000000     0.000000     0.021500
25%       0.511000     0.000000          0.000000     2.000000     0.094900
50%       0.632000     0.000000          0.000000     5.000000     0.119000
75%       0.741000     1.000000          0.000024     8.000000     0.194000
max       0.979000     1.000000          1.000000    11.000000     0.908000
```

```
       loudness         mode   popularity  speechiness        tempo
count  1561.000000  1561.000000  1561.000000  1561.000000  1561.000000
mean     -6.478233     0.569507    79.338245     0.111196   121.320165
std       2.963855     0.495304     4.191980     0.104563    30.339365
min     -40.449000     0.000000    75.000000     0.000000     0.000000
25%      -7.617000     0.000000    76.000000     0.042800    97.972000
50%      -6.003000     1.000000    78.000000     0.065700   120.028000
75%      -4.681000     1.000000    81.000000     0.137000   142.053000
max      -1.339000     1.000000   100.000000     0.884000   207.970000
```

In [13]:
```
all_songs[all_songs['popularity']>=75].groupby('year').size()
```

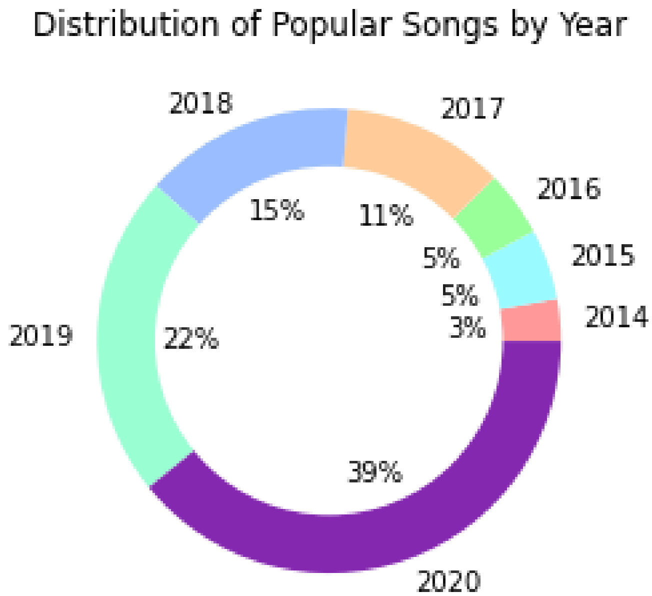
Out[13]:
```
year
2014     45
2015     77
2016     71
2017    177
2018    229
2019    351
2020    611
dtype: int64
```

About 28% of songs had a popularity rating at or above the 75th percentile and 11% of songs had a popularity rating in the 90th percentile. We decided that a song would be considered popular if it scored within the 90th percentile for popularity.

# 3 Descriptive Analysis

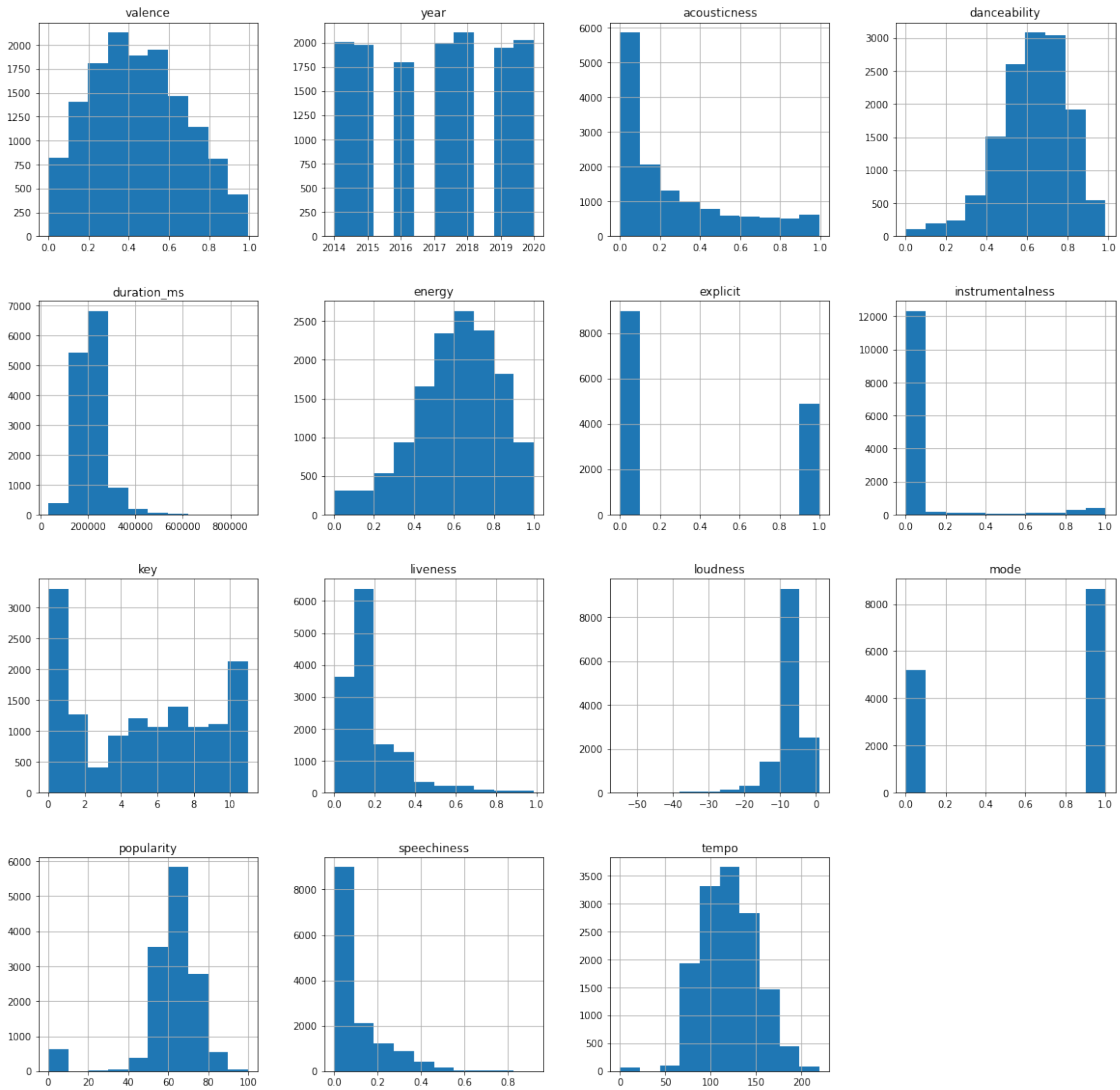
## 3.1 Plotting Variables

In [14]:
```
#Creating Donut Chart displaying distribution of popular songs by year
pie_chart = all_songs[all_songs['popularity']>=75].groupby('year').size()
labels = ['2014','2015','2016','2017','2018','2019','2020']
colors = ['#ff9999','#99faff','#99ff99','#ffcc99', '#99bdff', '#99ffd3', '#8528b0']
plt.pie(pie_chart, labels = labels, autopct='%1.f%%', colors = colors )

#Adding donut to center
circle = plt.Circle(xy=(0,0), radius=0.75, facecolor='white')
plt.gca().add_artist(circle)
plt.title('Distribution of Popular Songs by Year')
plt.show()
```




Nearly 40% of popular songs were released in 2020 which was not surprising since song popularity was determined in part by recency of play. Each prior year saw a reduction in the percent contribution to popular songs for songs released in that year.

```
In [15]: all_songs.hist(figsize=(20,20))
         plt.show()
```




**Skewed Left**

***Instrumentalness***, and to a lesser extent, ***Duration***, are almost completely distributed around the minimum value.

***Acousticness, Liveliness, and Speechiness*** have longtail distributions.

**Skewed Right**

***Loudness*** is skewed toward the right.

**Normal Distribution**

***Danceability, Energy, Tempo*** and ***Valence*** are pretty normally distributed; Popularity is somewhat normally distributed.

**Indicators**

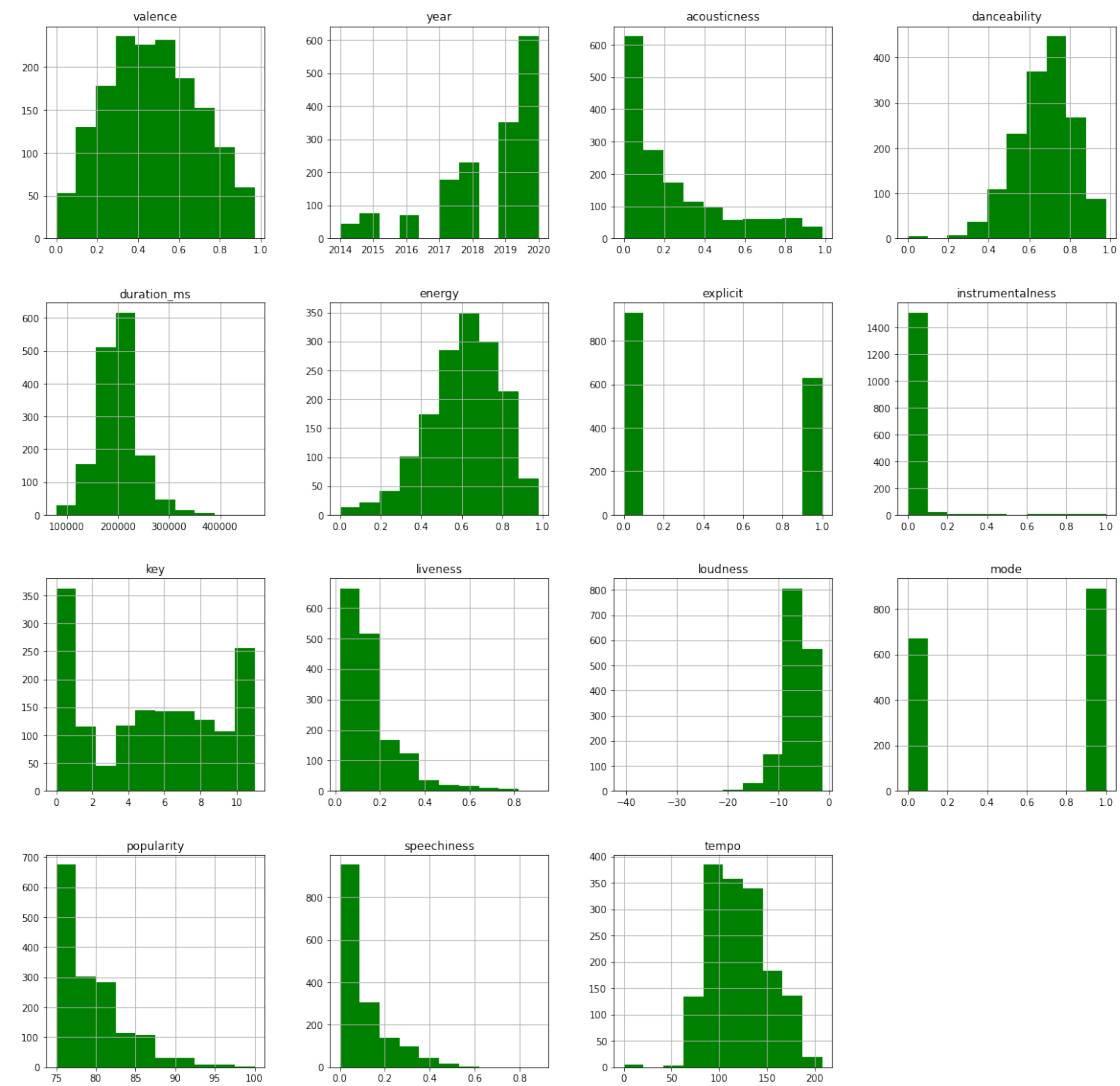
***Explicitness***, and ***Mode*** are Indicators with binary values

**Other**

There is no clear distribution for ***Key***.

Songs to be roughly equally represented across all ***Years*** in the dataset.

In [16]:
```
all_songs[all_songs['popularity']>=75].hist(figsize=(20,20), color = 'green')
plt.show()
```




Popular songs were similar in distribution to all songs for most variables, the most notable were that popular songs tended to be more loud, they were less likely to be live and while tempo did remain centrally distributed, there was slightly less spread. As noted previously, popular songs also were much more likely to be released near the time the data was sourced.

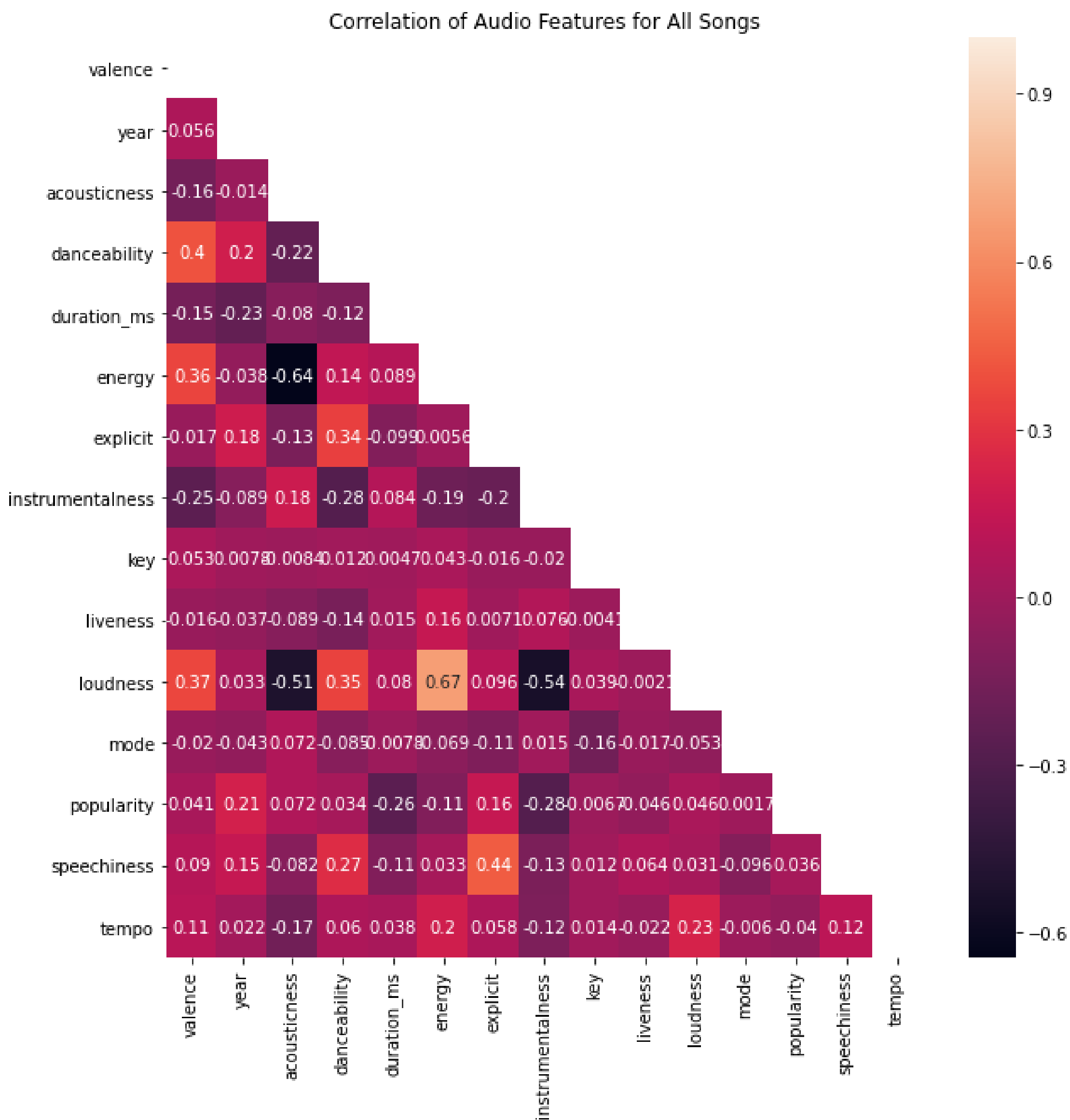
## 3.2 Interactions

In [17]:
```
#Displaying correlations for variables in the dataframe
all_songs.corr()['popularity']
```

Out[17]:
```
valence             0.040537
year                0.212880
```

```
acousticness        0.071848
danceability        0.034076
duration_ms        -0.255084
energy             -0.110988
explicit            0.158173
instrumentalness   -0.283240
key                -0.006720
liveness           -0.046066
loudness            0.045696
mode                0.001747
popularity          1.000000
speechiness         0.035795
tempo              -0.039747
Name: popularity, dtype: float64
```

In [19]:
```
# Create mask for upper triangle
mask = np.triu(np.ones_like(all_songs.corr(), dtype=bool))

# Plot heat map to show correlation
plt.figure(figsize = (10,10))
sns.heatmap(all_songs.corr(), annot = True, mask=mask)
plt.title("Correlation of Audio Features for All Songs")
plt.show()
```

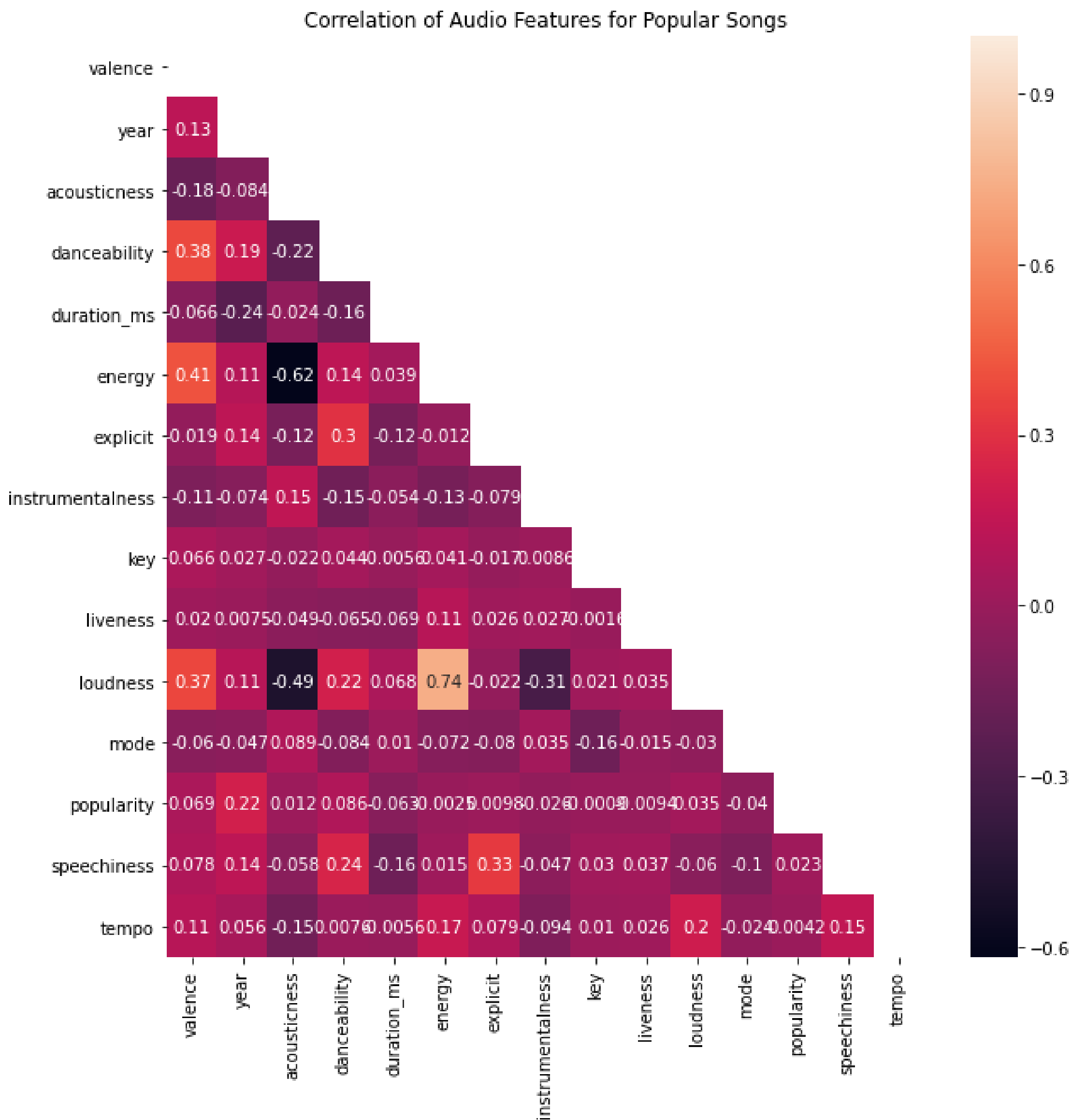



```
In [20]: # Popular songs only
         all_songs[all_songs['popularity']>=75].corr()['popularity']
```

```
Out[20]: valence             0.068596
         year                0.219569
         acousticness        0.012081
         danceability        0.085917
         duration_ms        -0.063154
         energy             -0.002490
         explicit            0.009833
         instrumentalness   -0.026469
         key                -0.000899
```

```
liveness             -0.009410
loudness              0.035037
mode                 -0.040351
popularity            1.000000
speechiness           0.022743
tempo                 0.004184
Name: popularity, dtype: float64
```

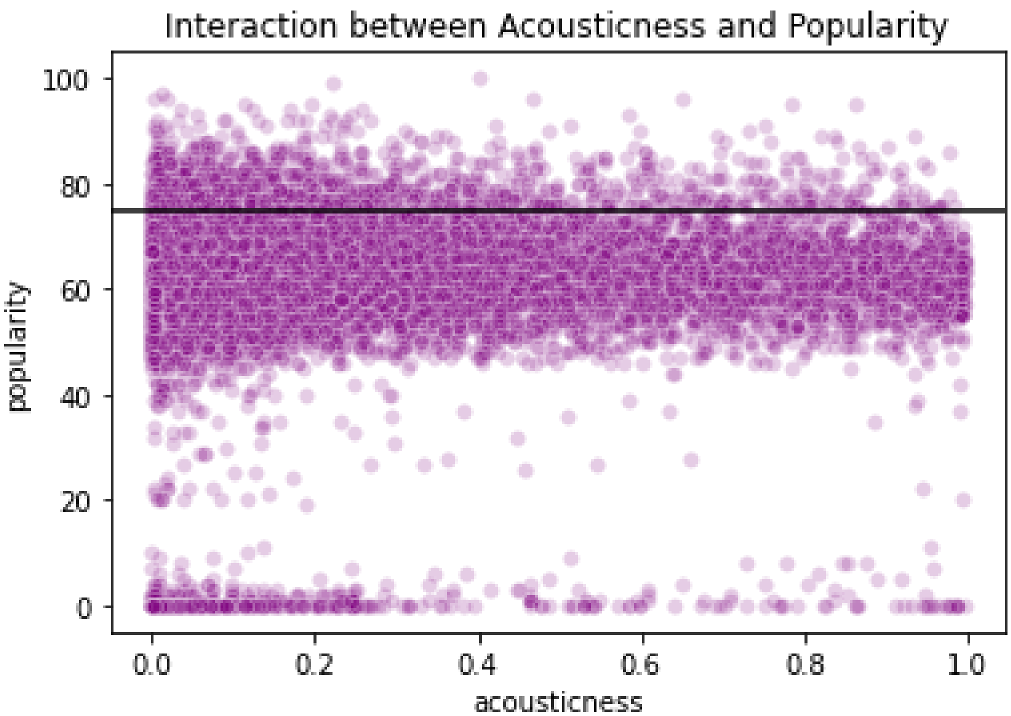
In [21]:
```python
mask = np.triu(np.ones_like(all_songs[all_songs['popularity']>=75].corr(), dtype=bool

# Plot heat map to show correlation
plt.figure(figsize = (10,10))
sns.heatmap(all_songs[all_songs['popularity']>=75].corr(), annot = True, mask=mask)
plt.title("Correlation of Audio Features for Popular Songs")
plt.show()
```



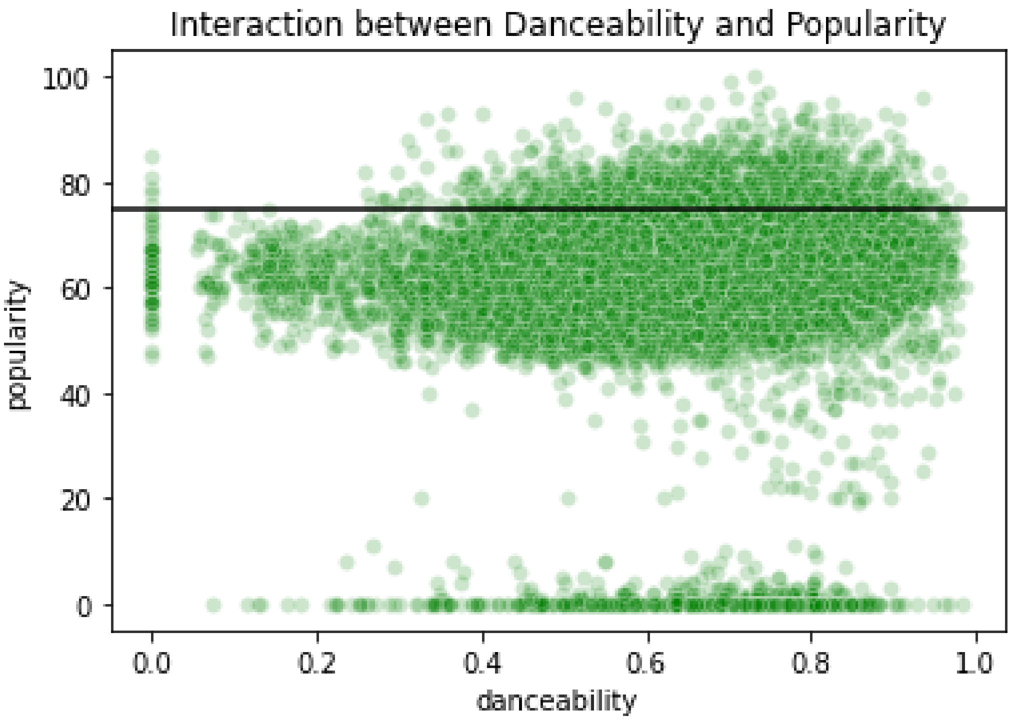
Popularity is most highly correlated with the song's year, instrumentalness and duration (a negative correlation).

For popular songs only, the features most highly correlated with popularity were increasing danceability and valence as well as decreasing duration.

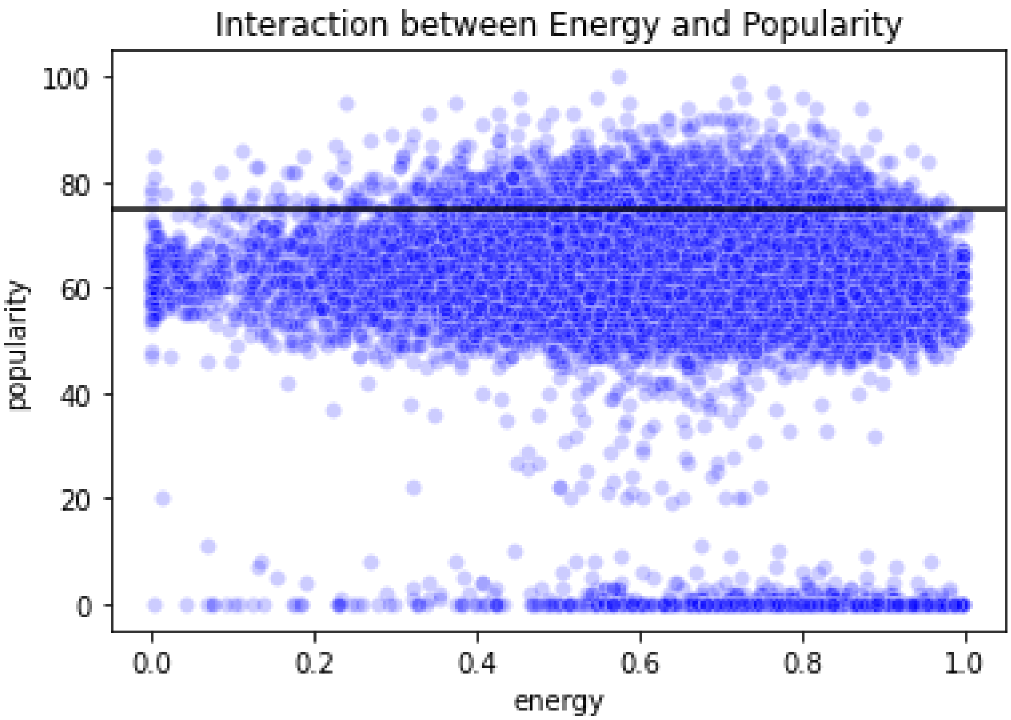
```
In [22]: #Creating scatter plots for variables related to popularity
         #Popular song threshold shown
         sns.scatterplot(x = 'acousticness', y = 'popularity', data = all_songs, alpha = 0.2, c
         plt.title('Interaction between Acousticness and Popularity')
         plt.axhline(y=75, color = 'black')
         plt.show()
```



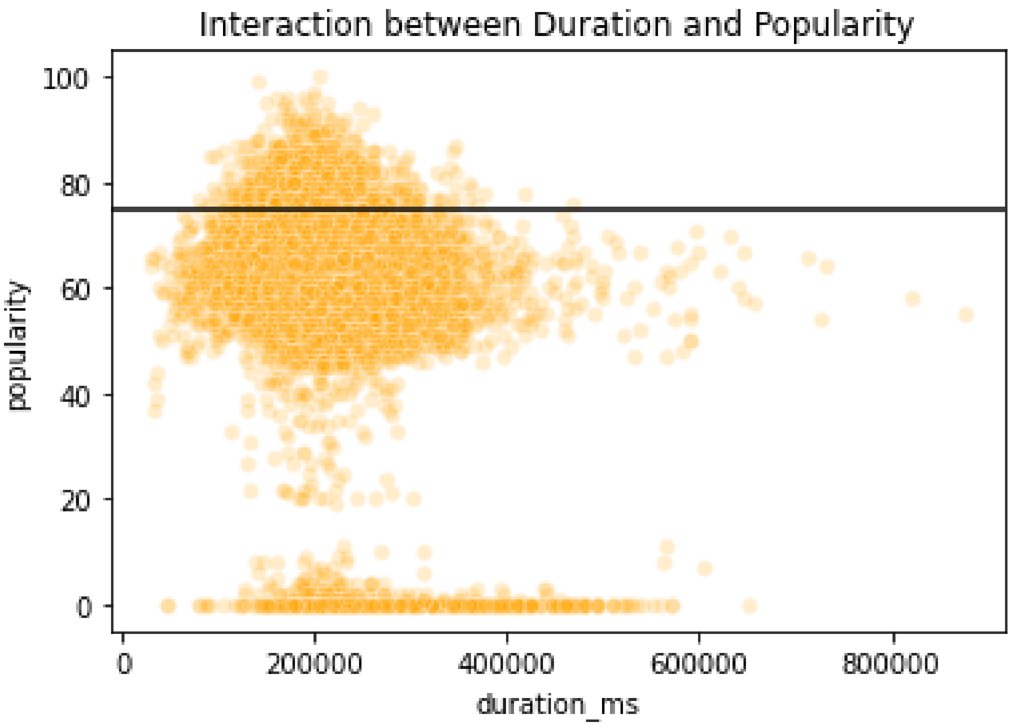

```
In [23]: sns.scatterplot(x = 'danceability', y = 'popularity', data = all_songs, alpha = 0.2, c
         plt.title('Interaction between Danceability and Popularity')
         plt.axhline(y=75, color = 'black')
         plt.show()
```



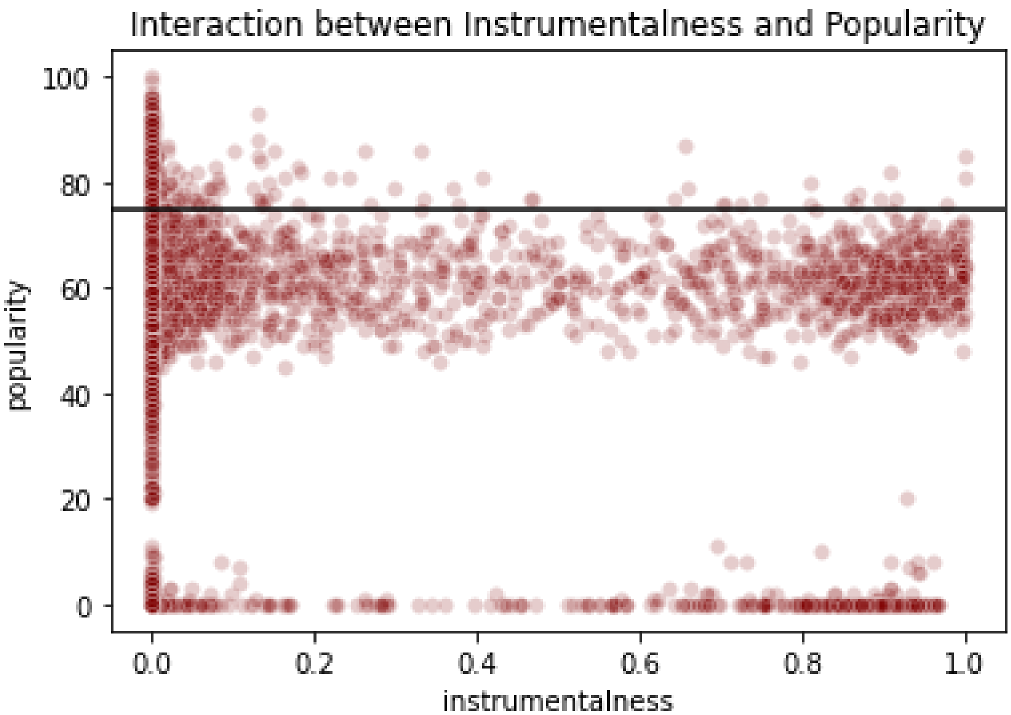

```
In [24]: sns.scatterplot(x = 'energy', y = 'popularity', data = all_songs, alpha = 0.2, color =
         plt.title('Interaction between Energy and Popularity')
         plt.axhline(y=75, color = 'black')
         plt.show()
```



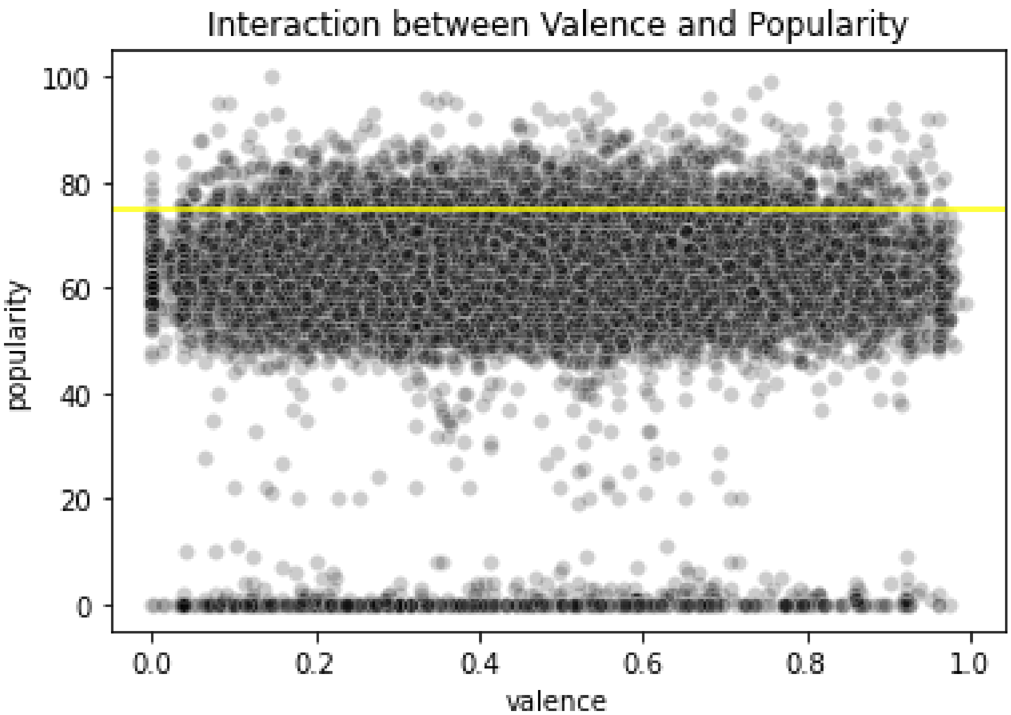

```
In [25]: sns.scatterplot(x = 'duration_ms', y = 'popularity', data = all_songs, alpha = 0.2, co
         plt.title('Interaction between Duration and Popularity')
         plt.axhline(y=75, color = 'black')
         plt.show()
```



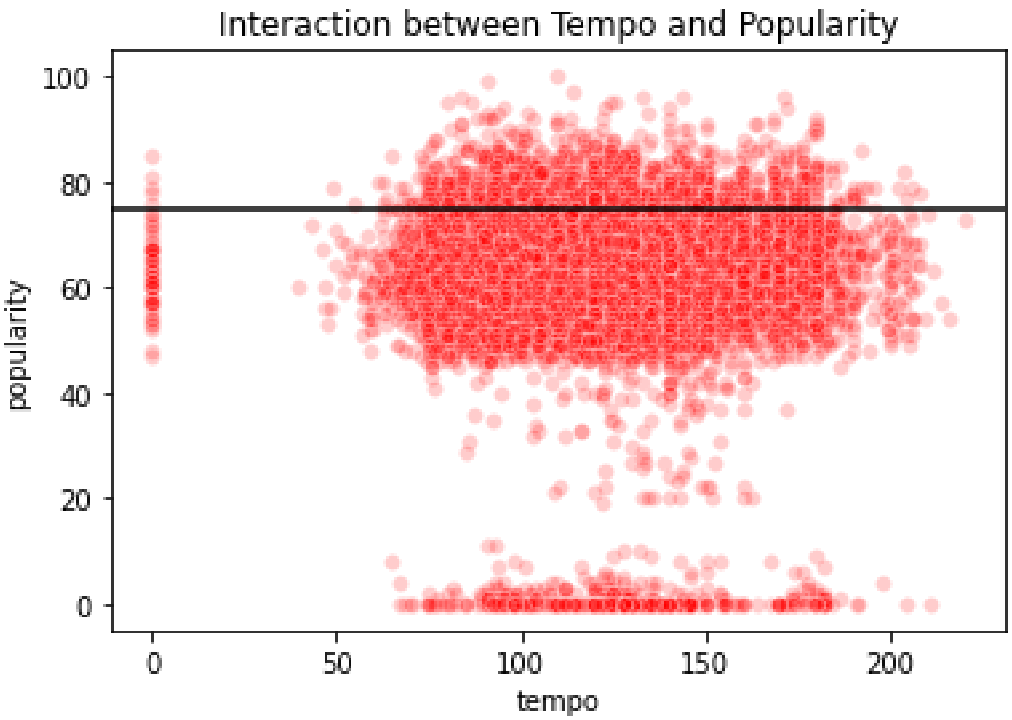

```
In [26]: sns.scatterplot(x = 'instrumentalness', y = 'popularity', data = all_songs, alpha = 0
         plt.title('Interaction between Instrumentalness and Popularity')
         plt.axhline(y=75, color = 'black')
         plt.show()
```



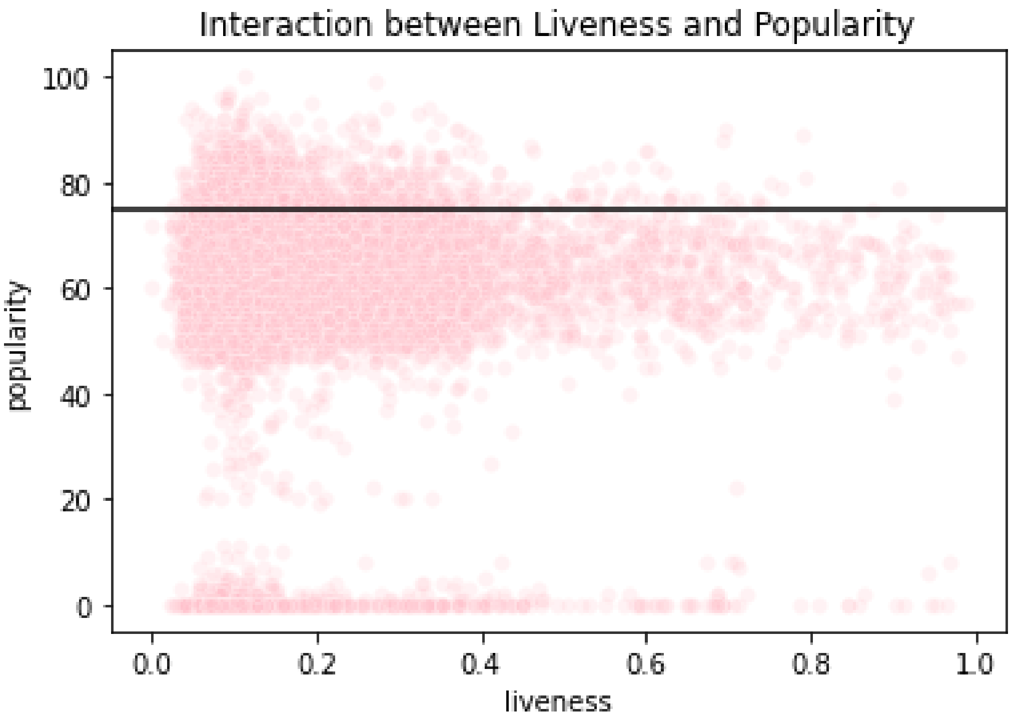

```
In [27]: sns.scatterplot(x = 'valence', y = 'popularity', data = all_songs, alpha = 0.2, color
         plt.title('Interaction between Valence and Popularity')
         plt.axhline(y=75, color = 'yellow')
         plt.show()
```




```
In [28]: sns.scatterplot(x = 'tempo', y = 'popularity', data = all_songs, alpha = 0.2, color =
         plt.title('Interaction between Tempo and Popularity')
         plt.axhline(y=75, color = 'black')
         plt.show()
```

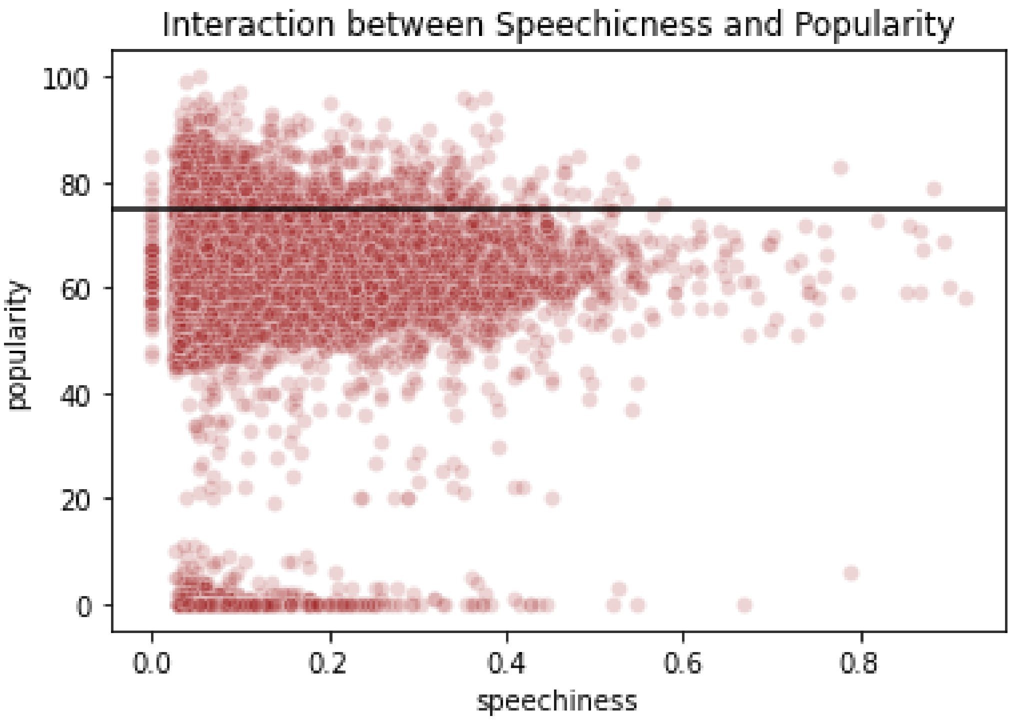



```
In [29]: sns.scatterplot(x = 'liveness', y = 'popularity', data = all_songs, alpha = 0.2, colo
         plt.title('Interaction between Liveness and Popularity')
         plt.axhline(y=75, color = 'black')
         plt.show()
```

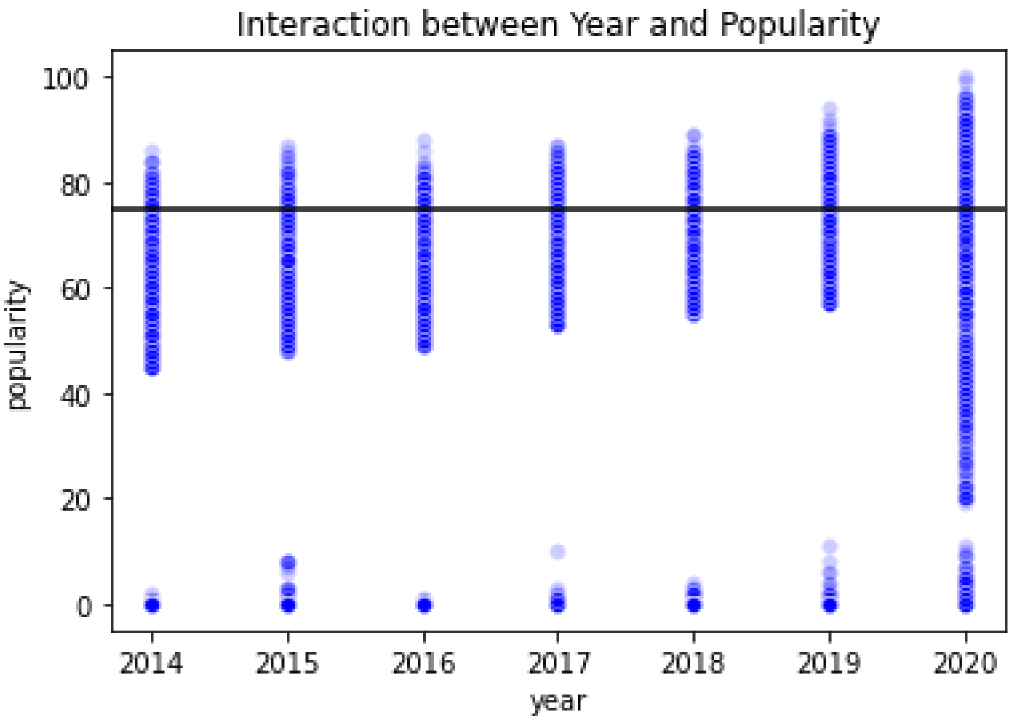



```
In [30]: sns.scatterplot(x = 'loudness', y = 'popularity', data = all_songs, alpha = 0.2, colo
         plt.title('Interaction between Loudness and Popularity')
         plt.axhline(y=75, color = 'red')
         plt.show()
```


```
In [31]: sns.scatterplot(x = 'speechiness', y = 'popularity', data = all_songs, alpha = 0.2, co
         plt.title('Interaction between Speechicness and Popularity')
         plt.axhline(y=75, color = 'black')
         plt.show()
```




```
In [32]: sns.scatterplot(x = 'year', y = 'popularity', data = all_songs, alpha = 0.2, color =
         plt.title('Interaction between Year and Popularity')
         plt.axhline(y=75, color = 'black')
         plt.show()
```




### 3.3 Takeaways

As a result of our descriptive analysis, we determined that the most important feature of a song that determined how popular it would be was the year in which the song was released. After referencing the documentation on what contributes to the popularity score, this result was expected.

There were some other audio features which appeared to also contribute to a songs popularity like increasing danceability where popular songs ***seemed*** to have higher scores than all songs (popular songs also seemed to have lower duration and accousticness but these were a little less clear).

From here, we ran a cluster analysis to group songs based on their audio features to determine which of them appeared to be most important for popularity.

## 4 Models & Statistical Analysis

### 4.1 Cluster Analysis

#### 4.1.1 Finding the Optimal Number - The Silhouette

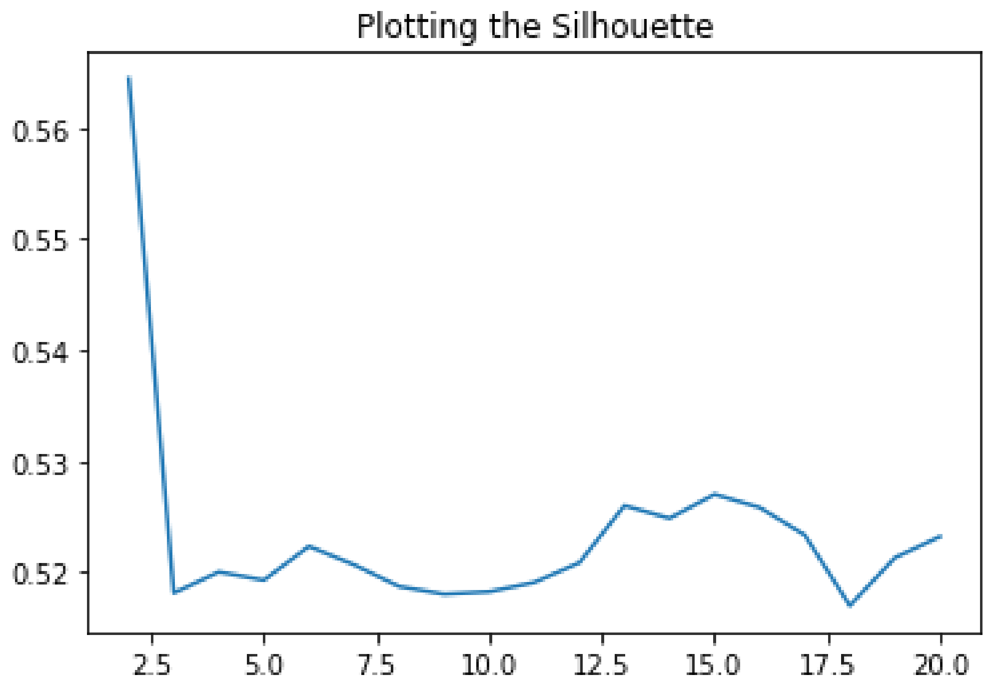
The Silhouette method is used to measure how similar a point is to its own cluster compared to other clusters. Using this method, we can determine the optimal number of cluster for the analysis by choosing the number of clusters that correspond to where k peaks on the Silhouette plot.

```
In [33]: X = all_songs.select_dtypes(np.number)
         sil = []
```

```
kmax = 20

for k in range(2, kmax+1):
    kmeans = KMeans(n_clusters=k).fit(X)
    labels = kmeans.labels_
    sil.append(silhouette_score(X, labels, metric = 'euclidean'))

k = [k for k in range(2,21)]
k
```

Out[33]: [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

In [34]:
```
plt.plot(k,sil)
plt.title("Plotting the Silhouette")
plt.show()
```




In [35]:
```
#Creating the song clusters
song_cluster_pipeline = Pipeline([('scaler', StandardScaler()),
                                  ('kmeans', KMeans(n_clusters=3,
                                   n_jobs=4))])

number_cols = list(X.columns)
song_cluster_pipeline.fit(X)
song_cluster_labels = song_cluster_pipeline.predict(X)
all_songs['cluster_label'] = song_cluster_labels
all_songs['cluster_label'].unique()
```

Out[35]: array([2, 1, 0], dtype=int32)

```
In [36]: #Checking descriptive stats for each cluster
         all_songs[all_songs['cluster_label']==0].describe()
```

Out[36]:

```
             valence         year  acousticness  danceability    duration_ms  \
count    2164.000000  2164.000000   2164.000000   2164.000000    2164.000000
mean        0.248086  2016.666359      0.711323      0.461512  208339.485675
std         0.181591     1.831192      0.261742      0.197682   75229.424602
min         0.000000  2014.000000      0.000000      0.000000   30583.000000
25%         0.102000  2015.000000      0.598500      0.343750  162780.750000
50%         0.221000  2017.000000      0.790500      0.484000  205105.000000
75%         0.360000  2018.000000      0.912250      0.603000  244396.250000
max         0.976000  2020.000000      0.996000      0.945000  820853.000000

              energy     explicit  instrumentalness          key     liveness  \
count    2164.000000  2164.000000       2164.000000  2164.000000  2164.000000
mean        0.307117     0.076248          0.282318     4.920980     0.180105
std         0.197301     0.265455          0.394107     3.538801     0.172218
min         0.000020     0.000000          0.000000     0.000000     0.000000
25%         0.178750     0.000000          0.000001     2.000000     0.099875
50%         0.300500     0.000000          0.002045     5.000000     0.112000
75%         0.406250     0.000000          0.750250     8.000000     0.158000
max         1.000000     1.000000          1.000000    11.000000     0.987000

            loudness         mode   popularity  speechiness        tempo  \
count    2164.000000  2164.000000  2164.000000  2164.000000  2164.000000
mean      -14.127390     0.739372    61.212107     0.060278   105.909484
std         7.444881     0.439079    13.348413     0.065693    35.396282
min       -54.837000     0.000000     0.000000     0.000000     0.000000
25%       -16.243000     0.000000    58.000000     0.032900    81.419500
50%       -11.827500     1.000000    63.000000     0.040950   104.087500
75%        -9.229250     1.000000    67.000000     0.057400   129.979250
max        -2.939000     1.000000    95.000000     0.789000   215.669000

       cluster_label
count         2164.0
mean             0.0
std              0.0
min              0.0
25%              0.0
50%              0.0
75%              0.0
max              0.0
```

```
In [37]: all_songs[all_songs['cluster_label']==1].describe()
```

Out[37]:

```
             valence         year  acousticness  danceability    duration_ms  \
count    4820.000000  4820.000000   4820.000000   4820.000000    4820.000000
```

```
mean       0.455847  2017.708299      0.203268      0.724194  201437.410166
std        0.210967     1.890764      0.208991      0.130896   54017.197576
min        0.035600  2014.000000      0.000038      0.216000   30579.000000
25%        0.294000  2016.000000      0.038650      0.644000  167544.000000
50%        0.445000  2018.000000      0.128000      0.741000  197707.000000
75%        0.607000  2019.000000      0.307000      0.820000  229477.750000
max        0.985000  2020.000000      0.992000      0.985000  727107.000000

            energy     explicit  instrumentalness          key     liveness  \
count  4820.000000  4820.000000       4820.000000  4820.000000  4820.000000
mean      0.613621     0.940041          0.006845     5.188797     0.185104
std       0.143643     0.237435          0.051946     3.677049     0.142965
min       0.156000     0.000000          0.000000     0.000000     0.022100
25%       0.516000     1.000000          0.000000     1.000000     0.101000
50%       0.616000     1.000000          0.000000     5.000000     0.126000
75%       0.715000     1.000000          0.000008     8.000000     0.219000
max       0.992000     1.000000          0.906000    11.000000     0.939000

          loudness         mode   popularity  speechiness        tempo  \
count  4820.000000  4820.000000  4820.000000  4820.000000  4820.000000
mean     -6.841523     0.530705    65.170124     0.195736   123.333177
std       2.331149     0.499108     9.873095     0.141355    28.974571
min     -20.188000     0.000000     0.000000     0.024200    39.497000
25%      -8.108250     0.000000    60.000000     0.071600    98.223250
50%      -6.562000     1.000000    65.000000     0.165000   124.258500
75%      -5.247750     1.000000    71.000000     0.291000   144.018500
max       0.457000     1.000000   100.000000     0.918000   220.099000

       cluster_label
count         4820.0
mean             1.0
std              0.0
min              1.0
25%              1.0
50%              1.0
75%              1.0
max              1.0
```

```
In [38]: all_songs[all_songs['cluster_label']==2].describe()
```

```
Out[38]:          valence         year  acousticness  danceability    duration_ms  \
count  6866.000000  6866.000000   6866.000000   6866.000000    6866.000000
mean      0.510597  2016.656132      0.162304      0.614974  223944.905185
std       0.229052     2.017950      0.189350      0.137077   59354.293437
min       0.033000  2014.000000      0.000002      0.099300   40000.000000
25%       0.332250  2015.000000      0.014900      0.524000  190998.500000
50%       0.508000  2016.000000      0.083400      0.618000  213934.500000
75%       0.684000  2018.000000      0.251000      0.717000  241693.000000
```

```
max        0.993000  2020.000000     0.941000     0.983000  875307.000000

             energy     explicit  instrumentalness          key     liveness  \
count   6866.000000  6866.000000       6866.000000  6866.000000  6866.000000
mean       0.708185     0.025051          0.055530     5.336149     0.182432
std        0.154475     0.156291          0.189169     3.556805     0.143929
min        0.213000     0.000000          0.000000     0.000000     0.013400
25%        0.591000     0.000000          0.000000     2.000000     0.093400
50%        0.717500     0.000000          0.000002     6.000000     0.124000
75%        0.833000     0.000000          0.000461     8.000000     0.227000
max        0.999000     1.000000          0.976000    11.000000     0.979000

           loudness         mode   popularity  speechiness        tempo  \
count   6866.000000  6866.000000  6866.000000  6866.000000  6866.000000
mean      -5.859945     0.651762    58.366589     0.070450   123.565265
std        2.057934     0.476446    19.004689     0.058355    28.659956
min      -17.473000     0.000000     0.000000     0.022600    49.452000
25%       -7.023000     0.000000    55.000000     0.036000   100.020000
50%       -5.632500     1.000000    62.000000     0.048800   122.048000
75%       -4.446000     1.000000    69.000000     0.079100   142.126250
max        1.023000     1.000000    97.000000     0.521000   210.715000

       cluster_label
count         6866.0
mean             2.0
std              0.0
min              2.0
25%              2.0
50%              2.0
75%              2.0
max              2.0
```

In [39]:
```
#Plotting the clusters
pca_pipeline = Pipeline([('scaler', StandardScaler()), ('PCA', PCA(n_components=2))])
song_embedding = pca_pipeline.fit_transform(X)
projection = pd.DataFrame(columns=['x', 'y'], data=song_embedding)
projection['title'] = all_songs['name']
projection['cluster'] = all_songs['cluster_label']

fig = px.scatter(projection, x='x', y='y', color='cluster', hover_data=['x', 'y', 'ti
fig.show()
```

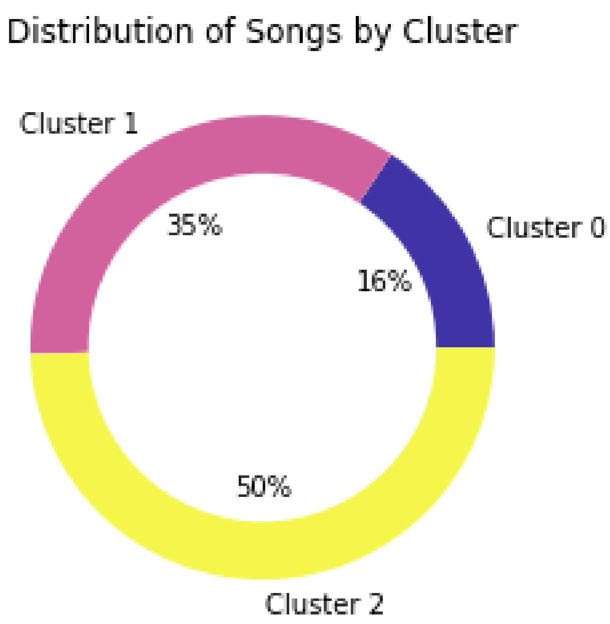
Ideally, we would have clusters that were close to each other and far from one another. Cluster 0 is distinct from both Clusters 2 and 1 but there is a little more spread. Clusters 2 and 1 overlap each other a little but they are tighter clusters than Cluster 0.

## 4.2 Describe Clusters

In [40]:
```
all_songs.groupby('cluster_label').size()
```

Out[40]: cluster_label
```
0    2164
1    4820
2    6866
dtype: int64
```

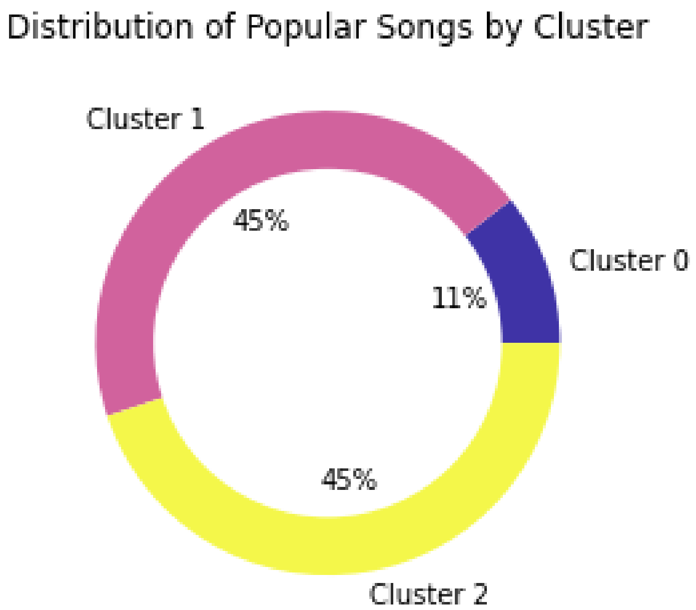
In [70]:
```
#Creating Donut Chart for Song Clusters and Adding Donut to Center
cluster_all = all_songs.groupby('cluster_label').size()
cluster_labels = ['Cluster 0','Cluster 1','Cluster 2']
colors_cluster = ['#3f33a6','#d1629d','#f4f74a']
plt.pie(cluster_all, labels = cluster_labels, autopct='%1.f%%', colors = colors_clust


circle = plt.Circle(xy=(0,0), radius=0.75, facecolor='white')
plt.gca().add_artist(circle)
plt.title('Distribution of Songs by Cluster')
plt.show()
```




In [42]:
```
all_songs[all_songs['popularity']>=75].groupby('cluster_label').size()
```

Out[42]: cluster_label
```
0    165
1    695
2    701
dtype: int64
```

In [43]:
```
#Creating Donut Chart for Song Clusters Limited to Popular Songs and Adding Donut to c
clusters_pop = all_songs[all_songs['popularity']>=75].groupby('cluster_label').size()
plt.pie(clusters_pop, labels = cluster_labels, autopct='%1.f%%', colors = colors_clus

circle = plt.Circle(xy=(0,0), radius=0.75, facecolor='white')
plt.gca().add_artist(circle)
plt.title('Distribution of Popular Songs by Cluster')
plt.show()
```




In [44]:
```
all_songs.groupby('cluster_label').mean()
```

Out[44]:
```
                valence         year  acousticness  danceability  \
cluster_label
0              0.248086  2016.666359      0.711323      0.461512
1              0.455847  2017.708299      0.203268      0.724194
2              0.510597  2016.656132      0.162304      0.614974

                  duration_ms    energy  explicit  instrumentalness       key  \
cluster_label
0              208339.485675  0.307117  0.076248          0.282318  4.920980
1              201437.410166  0.613621  0.940041          0.006845  5.188797
2              223944.905185  0.708185  0.025051          0.055530  5.336149

               liveness   loudness      mode  popularity  speechiness  \
cluster_label
0              0.180105 -14.127390  0.739372   61.212107     0.060278
```

```
1               0.185104  -6.841523  0.530705   65.170124     0.195736
2               0.182432  -5.859945  0.651762   58.366589     0.070450

                     tempo
cluster_label
0               105.909484
1               123.333177
2               123.565265
```

Nearly half of the songs were assigned to Cluster 2, one third to Cluster 1 and the remaining to Cluster 0. Popular songs seem to be over-represented in Cluster 1 (+10%) and under-represented in Clusters 0 & 2 (-5%) if we were to assume they would be evenly distributed amongst all clusters. Cluster 1 also has the highest average popularity among all clusters. We will test to see if there is a significant difference between the popularity of songs in Cluster 0 and the other clusters.

### 4.3 ANOVA Test

The Analysis of Variance (ANOVA) test is used to determine if there is a significant difference between three or more groups along some numeric value. We will use this test to determine if popularity differs significantly between the song clusters as it appears to based on distributions.

**Null Hypothesis**: There is no difference in popularity between the three song clusters.

**Alternative Hypothesis**: At least one of the clusters has differs in popularity from the others.

**P-value**: 0.05/6 –> 0.0083

```
In [45]: cluster_test_1 = all_songs.groupby(['cluster_label'])
         cluster_names = all_songs['cluster_label'].unique()
         print("\t\t\tstatistic\t\tpvalue")
         for i in range(len(cluster_names)):
             for j in range(i+1, len(cluster_names)):
                 cluster1 = cluster_test_1[['popularity']].get_group(cluster_names[i])
                 cluster2 = cluster_test_1[['popularity']].get_group(cluster_names[j])
                 stat, pvalue = stats.ttest_ind(cluster1, cluster2, equal_var = False)
                 print('Cluster ' + str(cluster_names[i]) + " vs. " + 'Cluster ' +  str(cluste
```

```
                        statistic               pvalue
Cluster 2 vs. Cluster 1        -25.21081458494421        2.4037167362948075e-136
Cluster 2 vs. Cluster 0        -7.746178081933237        1.1329239993381899e-14
Cluster 1 vs. Cluster 0        12.359032899765955        2.5156178431365245e-34
```

```
In [71]: #Finding Correlations to determine which characteristics of Cluster 0 Might Contribut
         all_songs[all_songs['cluster_label']==1].corr()['popularity']

Out[71]: valence             0.049856
         year                0.370702
         acousticness        0.028634
         danceability        0.022490
         duration_ms        -0.108151
         energy             -0.010146
```

```
explicit            -0.018221
instrumentalness    -0.021723
key                  0.001024
liveness            -0.052745
loudness             0.082234
mode                -0.003545
popularity           1.000000
speechiness         -0.108464
tempo                0.018844
cluster_label             NaN
Name: popularity, dtype: float64
```

```
In [47]: clusters_pct = round((pd.crosstab(all_songs['cluster_label'], all_songs['year'], norma
         clusters_pct
```

```
Out[47]: year           2014  2015  2016  2017  2018  2019  2020
         cluster_label
         0              15.2  16.1  15.9  19.7  12.7  13.4   7.0
         1               7.2   9.5  10.6  13.8  18.8  17.0  23.0
         2              19.4  17.0  13.7  13.1  13.4  12.2  11.2
```

Cluster 1 has many almost twice as many songs released in 2020 as other clusters. We will want to focus on other audio features that might impact song popularity. The next five most highly correlated features were **duration_ms**(-), **speechiness**(-), **loudness**(+), **liveness**(-), **valence**(+).

```
In [48]: cluster_test_2 = all_songs.groupby(['cluster_label'])
         print('Testing for significant differences in duration')
         print("\t\t\tstatistic\t\tpvalue")
         for i in range(len(cluster_names)):
             for j in range(i+1, len(cluster_names)):
                 cluster1 = cluster_test_1[['duration_ms']].get_group(cluster_names[i])
                 cluster2 = cluster_test_1[['duration_ms']].get_group(cluster_names[j])
                 stat, pvalue = stats.ttest_ind(cluster1, cluster2, equal_var = False)
                 print('Cluster ' + str(cluster_names[i]) + " vs. " + 'Cluster ' +  str(cluste
```

```
Testing for significant differences in duration
                        statistic               pvalue
Cluster 2 vs. Cluster 1        21.282193264247905       1.6262214419474703e-98
Cluster 2 vs. Cluster 0        8.822990477292679       1.836778436561522e-18
Cluster 1 vs. Cluster 0        -3.845991219891225       0.00012240306732907107
```

```
In [49]: cluster_test_3 = all_songs.groupby(['cluster_label'])
         print('Testing for significant differences in speechiness')
         print("\t\t\tstatistic\t\tpvalue")
         for i in range(len(cluster_names)):
             for j in range(i+1, len(cluster_names)):
                 cluster1 = cluster_test_1[['speechiness']].get_group(cluster_names[i])
                 cluster2 = cluster_test_1[['speechiness']].get_group(cluster_names[j])
```

```
            stat, pvalue = stats.ttest_ind(cluster1, cluster2, equal_var = False)
            print('Cluster ' + str(cluster_names[i]) + " vs. " + 'Cluster ' +  str(cluste
```

```
Testing for significant differences in speechiness
                        statistic               pvalue
Cluster 2 vs. Cluster 1        -58.15362833650543        0.0
Cluster 2 vs. Cluster 0        6.445667995117201        1.3179860016153327e-10
Cluster 1 vs. Cluster 0        54.667358789058355        0.0
```

```
In [50]: cluster_test_4 = all_songs.groupby(['cluster_label'])
         print('Testing for significant differences in loudness')
         print("\t\t\tstatistic\t\tpvalue")
         for i in range(len(cluster_names)):
             for j in range(i+1, len(cluster_names)):
                 cluster1 = cluster_test_1[['loudness']].get_group(cluster_names[i])
                 cluster2 = cluster_test_1[['loudness']].get_group(cluster_names[j])
                 stat, pvalue = stats.ttest_ind(cluster1, cluster2, equal_var = False)
                 print('Cluster ' + str(cluster_names[i]) + " vs. " + 'Cluster ' +  str(cluste
```

```
Testing for significant differences in loudness
                        statistic               pvalue
Cluster 2 vs. Cluster 1        23.50278901747312        8.732922737045655e-119
Cluster 2 vs. Cluster 0        51.04753374344391        0.0
Cluster 1 vs. Cluster 0        44.55516341432732        5.279149884e-315
```

```
In [51]: cluster_test_5 = all_songs.groupby(['cluster_label'])
         print('Testing for significant differences in liveness')
         print("\t\t\tstatistic\t\tpvalue")
         for i in range(len(cluster_names)):
             for j in range(i+1, len(cluster_names)):
                 cluster1 = cluster_test_1[['liveness']].get_group(cluster_names[i])
                 cluster2 = cluster_test_1[['liveness']].get_group(cluster_names[j])
                 stat, pvalue = stats.ttest_ind(cluster1, cluster2, equal_var = False)
                 print('Cluster ' + str(cluster_names[i]) + " vs. " + 'Cluster ' +  str(cluste
```

```
Testing for significant differences in liveness
                        statistic               pvalue
Cluster 2 vs. Cluster 1        -0.9917787540315631        0.3213285045722127
Cluster 2 vs. Cluster 0        0.5689303510430173        0.5694436900759304
Cluster 1 vs. Cluster 0        1.179899941987292        0.23811894458095892
```

```
In [52]: cluster_test_6 = all_songs.groupby(['cluster_label'])
         print('Testing for significant differences in valence')
         print("\t\t\tstatistic\t\tpvalue")
         for i in range(len(cluster_names)):
             for j in range(i+1, len(cluster_names)):
```

```
            cluster1 = cluster_test_1[['valence']].get_group(cluster_names[i])
            cluster2 = cluster_test_1[['valence']].get_group(cluster_names[j])
            stat, pvalue = stats.ttest_ind(cluster1, cluster2, equal_var = False)
            print('Cluster ' + str(cluster_names[i]) + " vs. " + 'Cluster ' +  str(cluste
```

```
Testing for significant differences in valence
                        statistic                pvalue
Cluster 2 vs. Cluster 1        13.32794723904964        3.29074543641859e-40
Cluster 2 vs. Cluster 0        54.88144951591811        0.0
Cluster 1 vs. Cluster 0        41.998062867420224        0.0
```

The results of the ANOVA test allow us to reject the hypothesis that there is no difference in the average popularity, duration, speechiness, loudness, and valence among the groups. We fail to reject a difference among the groups in liveness.

## 5 Conclusions & Recommendations

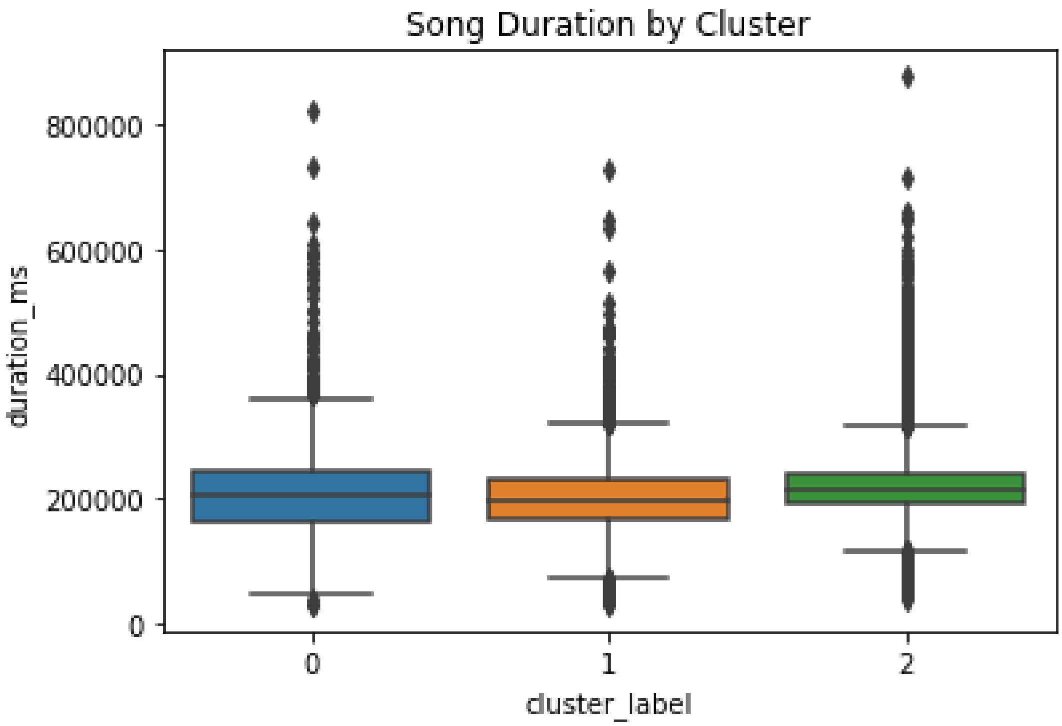
As a result of our analysis, we found that the audio features that had the highest impact on a song's popularity were the songs danceability and positivity/valence; where increased ratings led to higher popularity ratings. Additionally song duration was negatively correlated as people seemed to prefer shorter songs. Our recommendation would be to release a danceable song with a positive message but it should be a little shorter of a song.

For artists looking to increase exposure, or gain popularity, through recommendations on the platform, we recommend making songs that have similar make-ups to Cluster 1. These songs have a good mix of words and music but leaning toward more music (average rating 0.2 , lowest among clusters), are on the louder side (average loudness -6.8), are moderately positive (average valence 0.5) and are on the shorter side (average duration less than 3.5 minutes which was lowest among the clusters). These were the audio features that were most highly correlated with popularity.

```
In [53]: round((all_songs.groupby('cluster_label')['duration_ms'].mean()/60000),1)
```

```
Out[53]: cluster_label
         0    3.5
         1    3.4
         2    3.7
         Name: duration_ms, dtype: float64
```

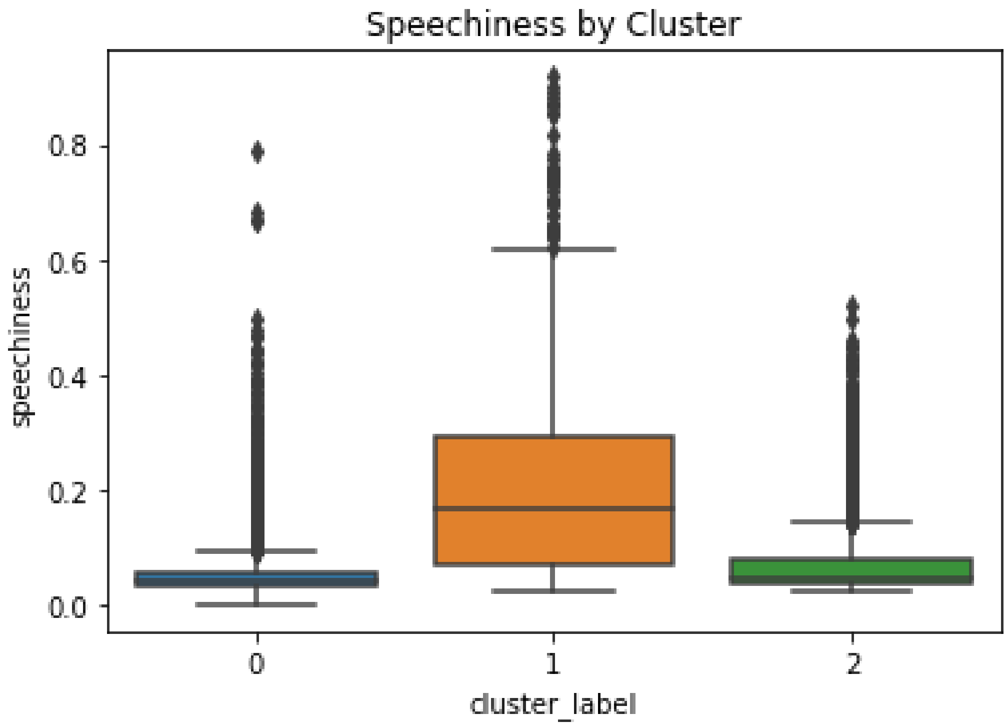
```
In [63]: sns.boxplot(x = 'cluster_label', y = 'duration_ms', data = all_songs)
         plt.title('Song Duration by Cluster')
         plt.show()
```




```
In [55]: all_songs.groupby('cluster_label')['speechiness'].mean()
```

```
Out[55]: cluster_label
         0    0.060278
         1    0.195736
         2    0.070450
         Name: speechiness, dtype: float64
```

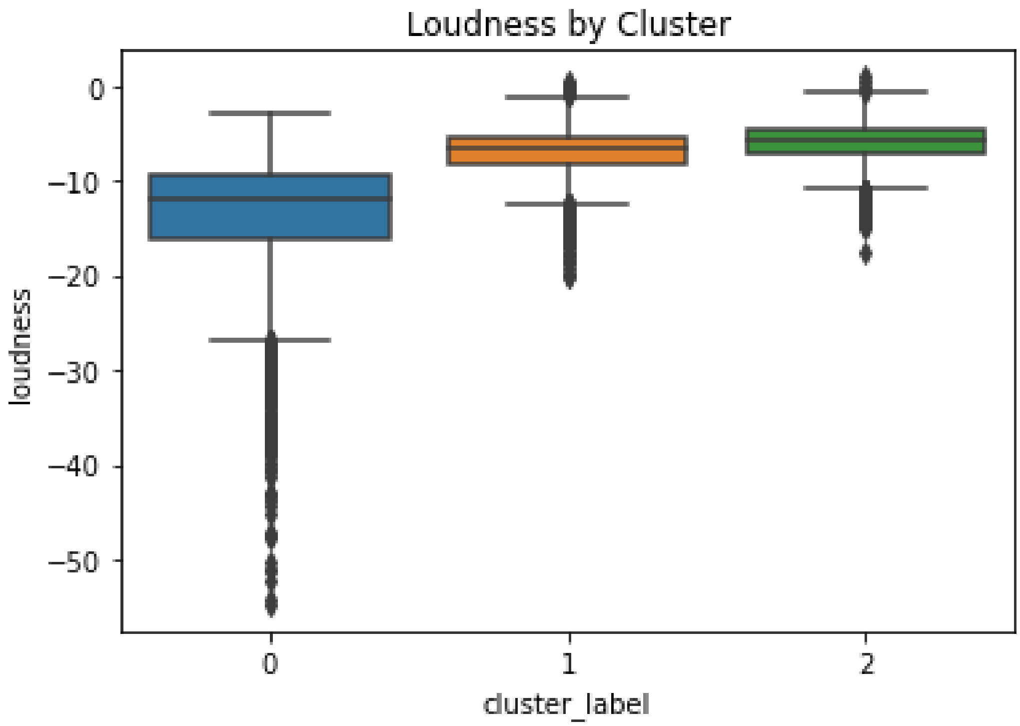
```
In [64]: sns.boxplot(x = 'cluster_label', y = 'speechiness', data = all_songs)
         plt.title('Speechiness by Cluster')
         plt.show()
```




In [57]:
```
all_songs.groupby('cluster_label')['loudness'].mean()
```

Out[57]:
```
cluster_label
0   -14.127390
1    -6.841523
2    -5.859945
Name: loudness, dtype: float64
```

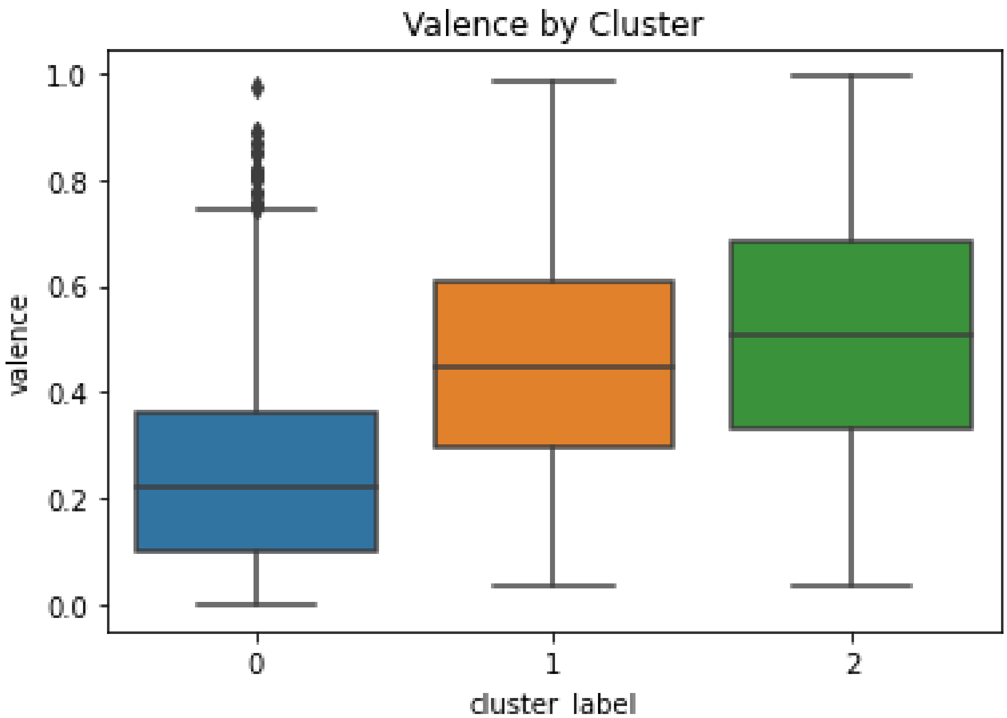
In [65]:
```
sns.boxplot(x = 'cluster_label', y = 'loudness', data = all_songs)
plt.title('Loudness by Cluster')
plt.show()
```




In [59]:
```
all_songs.groupby('cluster_label')['valence'].mean()
```

Out[59]:
```
cluster_label
0    0.248086
1    0.455847
2    0.510597
Name: valence, dtype: float64
```

In [66]:
```
sns.boxplot(x = 'cluster_label', y = 'valence', data = all_songs)
plt.title('Valence by Cluster')
plt.show()
```




## 6 References

Amplifying Artist Input in Your Personalized Recommendations - Reference information on how to increase exposure with Spotify's recommendation system.

Spotify dataset - Source dataset for analysis.

Spotify Web API Resources - Reference documentation on audio features and Spotify's web API.

The Silhouette Method - Background information on K-means cluster analysis and the Silhouette method.

```
In [67]: !jupyter nbconvert --to pdf Song_Cluster_Analysis.ipynb
```

```
[NbConvertApp] Converting notebook Song_Cluster_Analysis.ipynb to pdf
/opt/conda/lib/python3.6/site-packages/nbconvert/filters/datatypefilter.py:41: UserWarning: Yo
  mimetypes=output.keys())
/opt/conda/lib/python3.6/site-packages/nbconvert/filters/datatypefilter.py:41: UserWarning: Yo
  mimetypes=output.keys())
[NbConvertApp] Support files will be in Song_Cluster_Analysis_files/
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
```

```
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Making directory Song_Cluster_Analysis_files
[NbConvertApp] Writing 119864 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 872753 bytes to Song_Cluster_Analysis.pdf
```

```
In [68]: all_songs.to_csv(r'all_songs.csv', index = True)
```