Spotify Song Popularity

A correlation analysis of music popularity

Group 5 : Yuqing Huang, Uchenna Nwagbara, Alex Tucker

Our Hypothesis

- **Business Claim:**We want to help Spotify and other streaming platforms alike accurately set the price per song based on popularity.

- **Null Hypothesis:** The Valence of the song has the strongest correlation with popularity.

- **Alternate Hypothesis:** The Valence of the song has the weakest correlation with popularity.

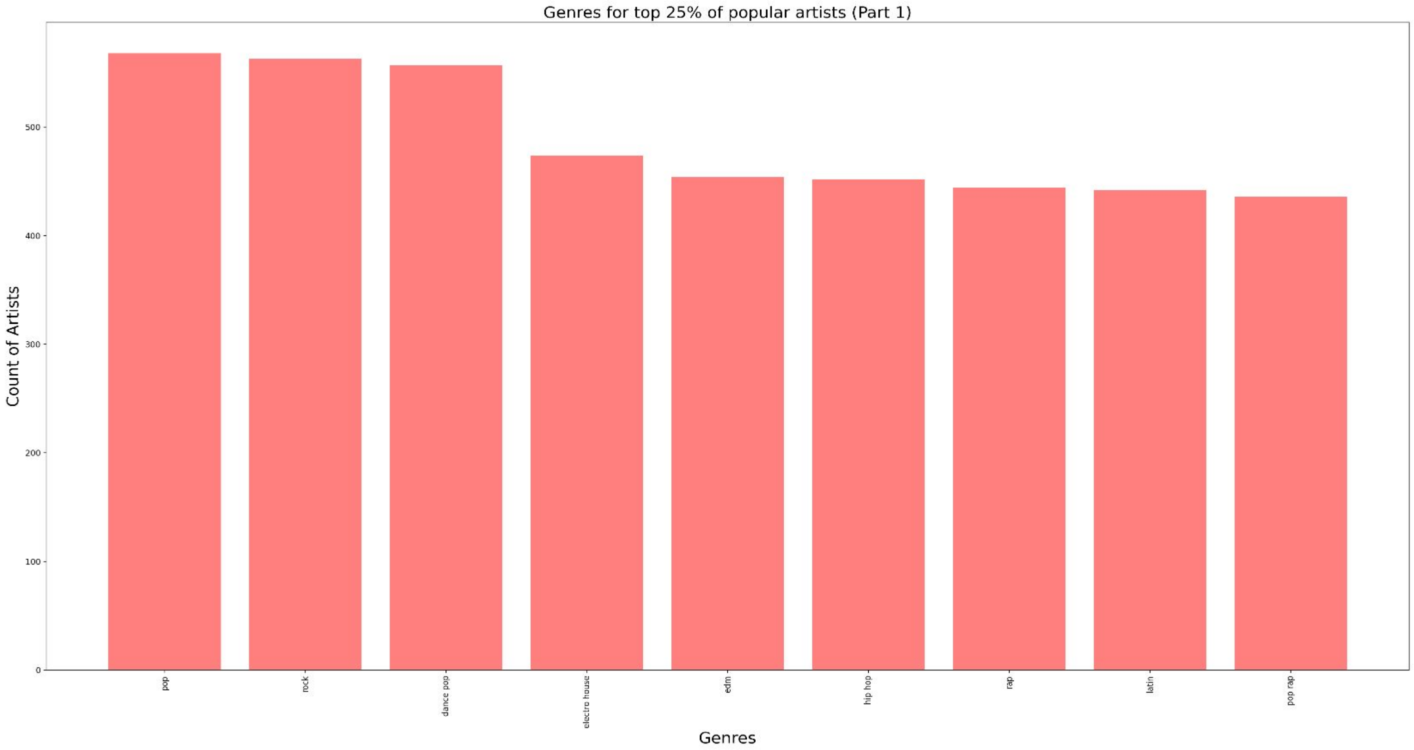
Key Terms and Definitions

- **Danceability-** Describes how suitable a track is for dancing based on a combination of musical elements such as tempo, rhythm, and beat strength.

- **Energy-** A measure of 0.0 to 1.0 representing a perceptual measure of intensity and activity. Typically energetic tracks feel fast, loud, and noisy.

- **Valence-** A measure of 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence tend to be more happy and cheerful, while tracks with low valence sound more negative, sad, or angry.

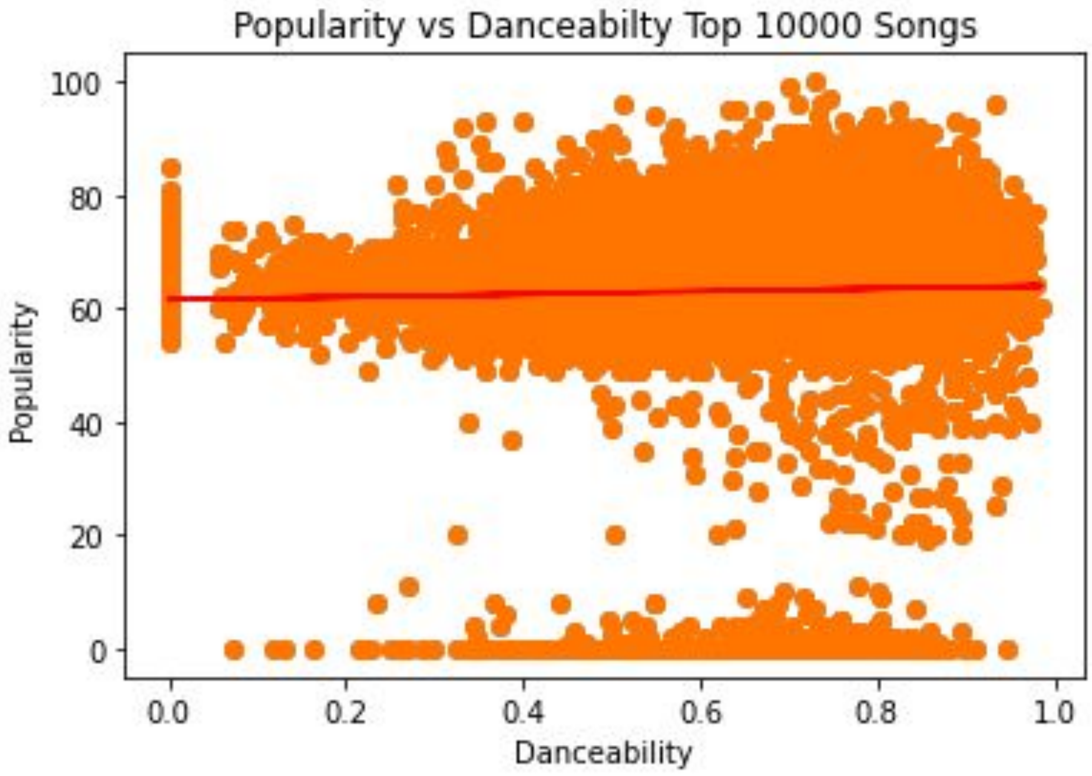
Genres for top 25% of popular artists



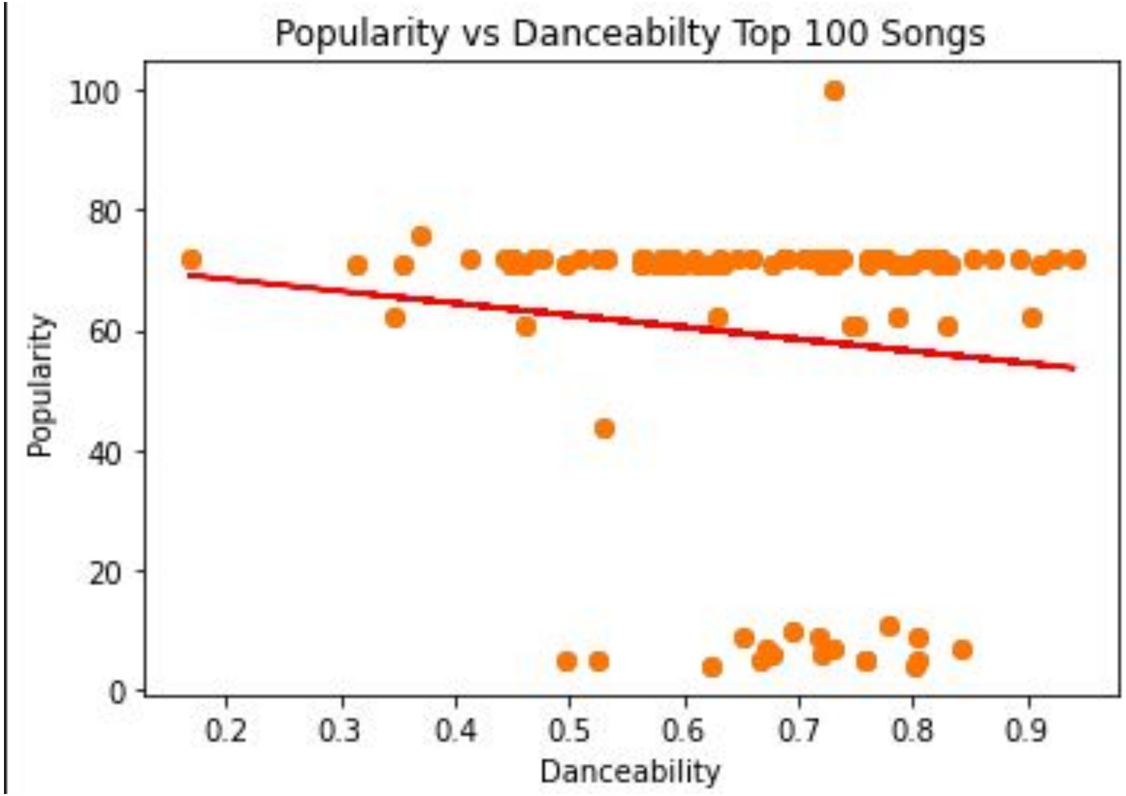
Examples From the Data

	Popularity	Danceability	Energy	Valence
Bad Bunny- Dakiti	**100**	**.731**	**.573**	**.145**
Ariana Grande- Positions	**96**	**.737**	**.802**	**.682**
Pop Smoke- For the Night	**95**	**.823**	**.586**	**.114**

Popularity vs Danceability Top 10000 Songs



Popularity vs Danceability Top 100 Songs

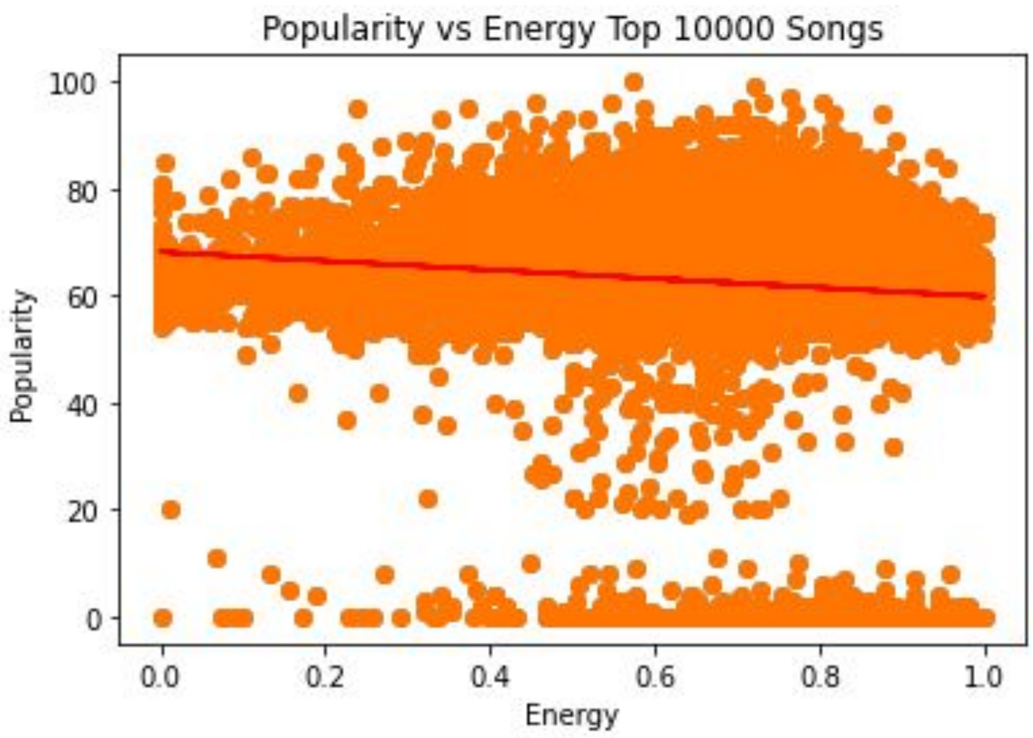


Danceability Conclusion

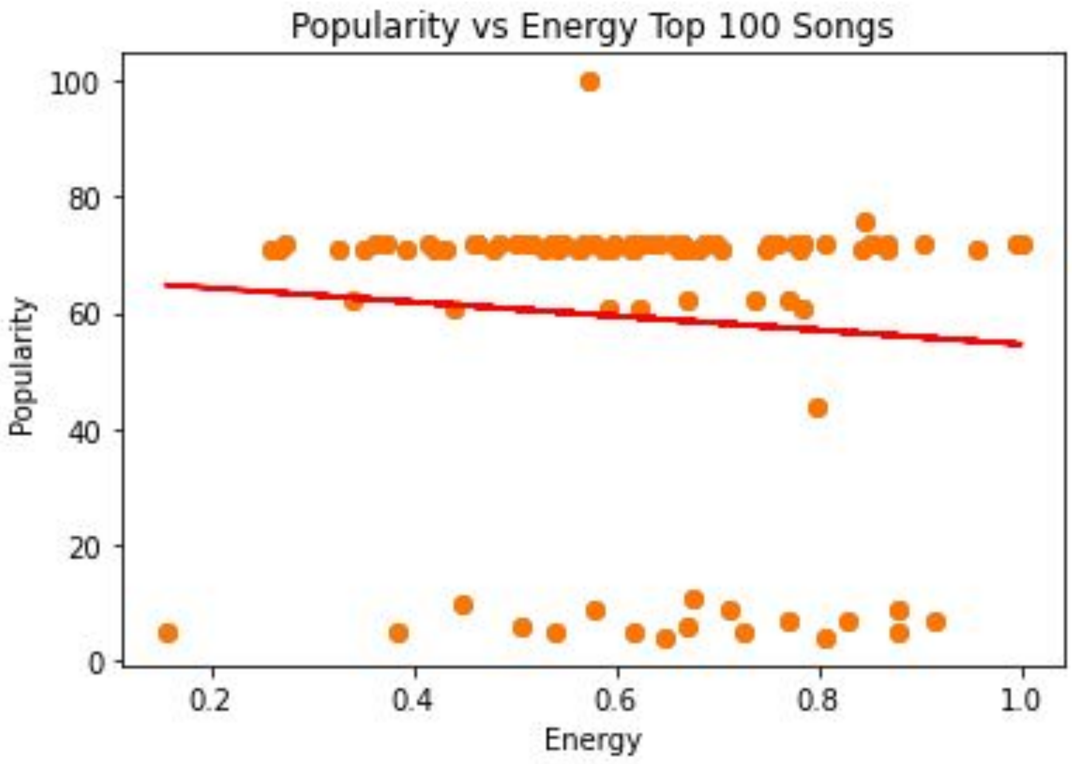
The correlation between danceability and popularity is to weak to use as a price determinant.

	popularity	danceability
popularity	1.000000	0.199606
danceability	0.199606	1.000000

Popularity vs Energy Top 10000 Songs



Popularity vs Energy Top 100 Songs

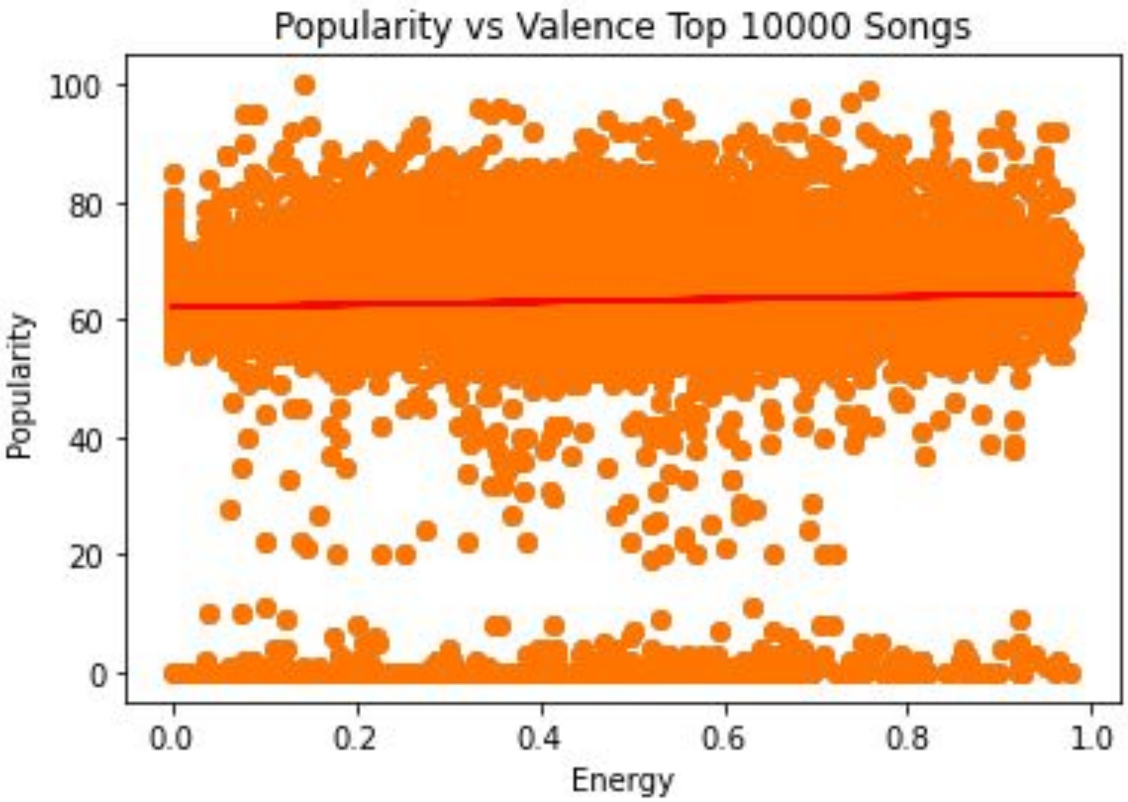


Energy Conclusion

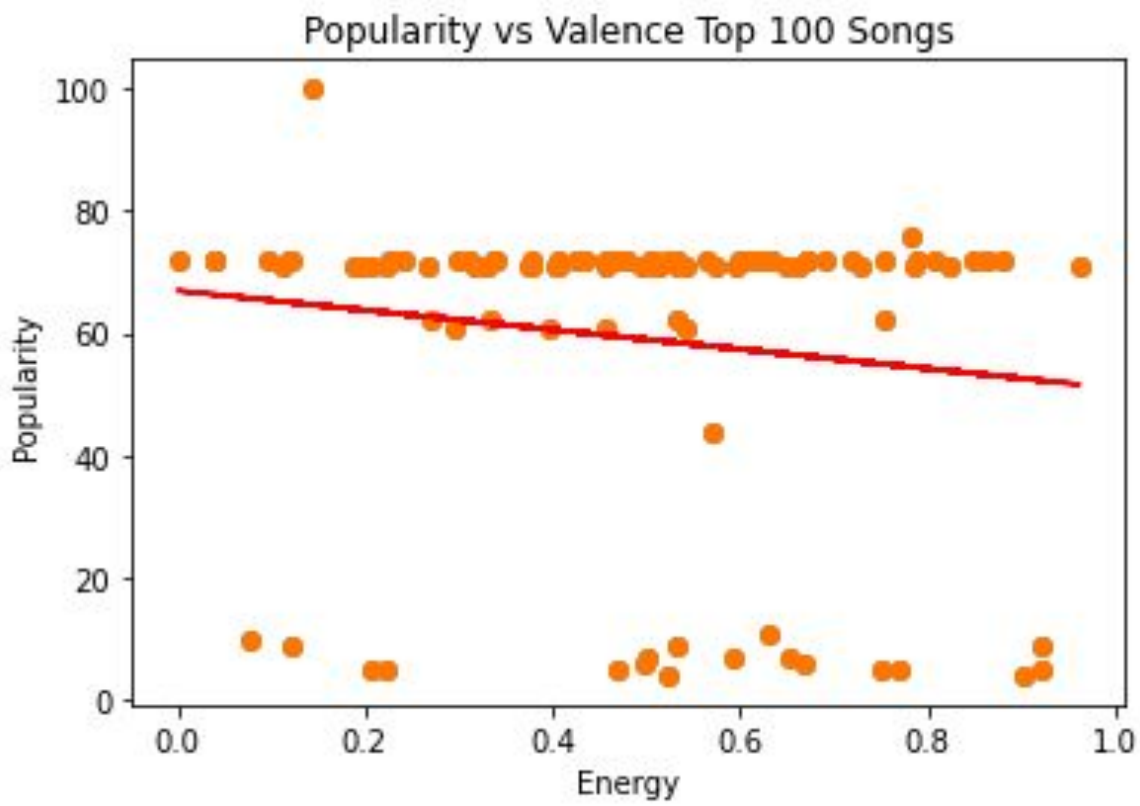
Although the trend is consistently negative regardless of the sample size, the correlation between energy and popularity is not strong enough to justify using as a price gauge.

	popularity	energy
popularity	1.000000	0.485005
energy	0.485005	1.000000

Popularity vs Valence Top 10000 Songs



Popularity vs Valence Top 100 Songs



Valence Conclusion

Valence has the weakest correlation with popularity of out the three parameters. This would not be an ideal parameter to using to determine price.

	popularity	valence
popularity	1.000000	0.014200
valence	0.014200	1.000000

Analysis Conclusion and Limitations

Conclusion

- Based on our analysis we will reject our null hypothesis, because out of the three parameters we chose, danceability did not have the strongest correlation with popularity.

Limitations

- Spotify was created in 2011, so the dataset does not take into account songs made before 2011.
- There are other popular streaming platforms such as Apple Music and Soundcloud.

Business Solution

- Based on our analysis the data is not conclusive enough for Spotify to make a decision on what prices to charge per song based on popularity.

Our Code

Import Dependencies and clean data

```
In [ ]: import pandas as pd
        import matplotlib.pyplot as plt
        from scipy.stats import linregress
        import numpy as np
        import seaborn as sn
        artists_df=pd.read_csv('artists.csv')
        data_df = pd.read_csv('data_o.csv')

        data_df.head()
```

```
In [ ]: top_100_songs = data_df.sort_values(by='popularity',ascending=False)

        top_100_songs.head(10)
```

```
In [ ]: artists_df=pd.read_csv('artists.csv')
        artists_df['expanded genres']=artists_df['genres'].str.strip("[]'")
        artists_df['expanded genres']=artists_df['expanded genres'].apply(lambda s:s.replace("'",""))
        expanded_genres=artists_df['expanded genres'].str.split(',\s+',expand=True).stack().value_counts()
        expanded_genres.to_frame()
        top_25_perc_pop=artists_df[artists_df['popularity']>41]
        genres_count=top_25_perc_pop['expanded genres'].str.split(',\s+',expand=True).stack().value_counts().to_frame().rename(
```

```
In [ ]: top_quart_genres_ct=genres_count[genres_count['Counts of Genres']>22]
        genres_top_100=top_quart_genres_ct.iloc[0:99 , :]
        genres_top_10=top_quart_genres_ct.iloc[0:9 , :]
        genres_middle_10=top_quart_genres_ct.iloc[20:30 , :]
```

```
In [ ]: x_axis = np.arange(len(genres_top_10))
        tick_locations = [value for value in x_axis]
        plt.figure(figsize=(30,15))
        plt.bar(x_axis, genres_top_10["Counts of Genres"], color='r', alpha=0.5, align="center")
        plt.xticks(tick_locations, genres_top_10.index, rotation="vertical")
        plt.title("Genres for top 25% of popular artists (Part 1)",fontsize=20)
        plt.xlabel("Genres",fontsize=20)
        plt.ylabel("Count of Artists",fontsize=20)
```

Bar Chart and Scatter plot

In [ ]:
```
x_axis = np.arange(len(genres_middle_10))
tick_locations = [value for value in x_axis]
plt.figure(figsize=(30,15))
plt.bar(x_axis, genres_middle_10["Counts of Genres"], color='r', alpha=0.5, align="center")
plt.xticks(tick_locations, genres_middle_10.index, rotation="vertical")
plt.title("Genres for top 25% of popular artists (Part 2)",fontsize=20)
plt.xlabel("Genres",fontsize=20)
plt.ylabel("Count of Artists",fontsize=20)
```

In [ ]:
```
new_songs= top_100_songs.sort_values(by='year',ascending=False).head(100)
```

In [ ]:
```
x_values = new_songs['danceability']
y_values = new_songs['popularity']
plt.scatter(x_values,y_values)
plt.title('Popularity vs Danceabilty Top 10000 Songs')
plt.xlabel('Danceability')
plt.ylabel('Popularity')
(slope, intercept, rvalue, pvalue, stderr) = linregress(x_values, y_values)
regress_values = x_values * slope + intercept
line_eq = "y = " + str(round(slope,2)) + "x + " + str(round(intercept,2))
plt.scatter(x_values,y_values)
plt.plot(x_values,regress_values,"r-")
plt.annotate(line_eq,(10,10),fontsize=15,color="red")
plt.show()
```

In [ ]:
```
x_values = new_songs['energy']
y_values = new_songs['popularity']
plt.scatter(x_values,y_values)
plt.title('Popularity vs Energy Top 10000 Songs')
plt.xlabel('Energy')
plt.ylabel('Popularity')
(slope, intercept, rvalue, pvalue, stderr) = linregress(x_values, y_values)
regress_values = x_values * slope + intercept
line_eq = "y = " + str(round(slope,2)) + "x + " + str(round(intercept,2))
plt.scatter(x_values,y_values)
plt.plot(x_values,regress_values,"r-")
plt.show()
```

Linear Regression and correlation

```
In [ ]: x_values = new_songs['year']
        y_values = new_songs['popularity']
        plt.scatter(x_values,y_values)
        plt.title('Popularity vs Year Top 10000 Songs')
        plt.xlabel('Year')
        plt.ylabel('Popularity')
        (slope, intercept, rvalue, pvalue, stderr) = linregress(x_values, y_values)
        regress_values = x_values * slope + intercept
        line_eq = "y = " + str(round(slope,2)) + "x + " + str(round(intercept,2))
        plt.scatter(x_values,y_values)
        plt.plot(x_values,regress_values,"r-")
        plt.show()
```

```
In [ ]: x_values = new_songs['valence']
        y_values = new_songs['popularity']
        plt.scatter(x_values,y_values)
        plt.title('Popularity vs Valence Top 10000 Songs')
        plt.xlabel('Energy')
        plt.ylabel('Popularity')
        (slope, intercept, rvalue, pvalue, stderr) = linregress(x_values, y_values)
        regress_values = x_values * slope + intercept
        line_eq = "y = " + str(round(slope,2)) + "x + " + str(round(intercept,2))
        plt.scatter(x_values,y_values)
        plt.plot(x_values,regress_values,"r-")
        plt.figure(figsize=(10, 10))
        plt.show()
```

```
In [ ]: corrMatrix = top_100_songs[['popularity','danceability','energy','valence','loudness','mode','speechiness','tempo','exp
        sn.heatmap(corrMatrix, annot=True)
        plt.figure(figsize=(10, 10))
        plt.show()
```

```
In [ ]: corr = top_100_songs.corr()
        corr.style.background_gradient(cmap='coolwarm')
```