[Spotify Song & Genre Analysis, Popularity Prediction and Building a Simple Recommendation System - 9]

Group Members:

Sude Buket Kipri 28368

Umut Şahin Önder 28248

Rüya Buse Dinmezel 28195

Introduction

As the dataset provides many songs with pre-calculated features along with the genre information, there is a huge potential to analyze how these features change with different genres. In the exploratory data analysis part, we will be interpreting the relationships among features with the help of visual explanations -graphs- and analyze the most popular songs. As for the hypothesis testing part, we will apply statistical tests to see if there are significant differences between various features and genres over time. Then we will be building a simple recommendation model based on feature similarities of songs in the dataset. As the dataset spans over a long time, how the features of genres change over time will be analyzed. Since music is a high demand sector and we're all interested in it, we decided to work on this topic especially with Spotify data in order to gain more knowledge on various features.

In [ ]:

```
from google.colab import drive
drive.mount('./drive', force_remount=True)

path_prefix = './drive/My Drive'
```

```
Mounted at ./drive
```

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from os.path import join
from os import listdir
import seaborn as sns
import re


sns.set_style("darkgrid")

import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

In [ ]:

```
filename = "data.csv"
df = pd.read_csv(join(path_prefix, filename))
```

In [ ]:

```
df['artists'] = df['artists'].apply(lambda x: re.findall(r'[\'"]([\w\s-]+)[\'"]', x))
df = df.explode('artists')
```

In [ ]:

```
df.isna().sum()
```

Out[ ]:

```
acousticness            0
artists             10719
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
valence                 0
year                    0
dtype: int64
```

In [ ]:

```
df.describe()
```

Out[ ]:

	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liv
count	229540.000000	229540.000000	2.295400e+05	229540.000000	229540.000000	229540.000000	229540.000000	229540.0
mean	0.549971	0.519929	2.392904e+05	0.450814	0.068785	0.221009	5.178431	0.2
std	0.385812	0.184887	1.563314e+05	0.277079	0.253089	0.349366	3.511733	0.1
min	0.000000	0.000000	4.937000e+03	0.000000	0.000000	0.000000	0.000000	0.0
25%	0.120750	0.386000	1.653138e+05	0.212000	0.000000	0.000001	2.000000	0.1
50%	0.644000	0.530000	2.075330e+05	0.416000	0.000000	0.000953	5.000000	0.1
75%	0.943000	0.661000	2.727902e+05	0.680000	0.000000	0.413000	8.000000	0.2
max	0.996000	0.988000	5.338302e+06	1.000000	1.000000	1.000000	11.000000	1.0

In [ ]:

```
df.shape
```

Out[ ]:

```
(229540, 19)
```

In [ ]:

```
df.dtypes
```

Out[ ]:

```
acousticness        float64
artists              object
danceability        float64
duration_ms           int64
energy              float64
explicit              int64
id                   object
instrumentalness    float64
key                   int64
liveness            float64
loudness            float64
mode                  int64
```

```
mode                int64
name               object
popularity          int64
release_date       object
speechiness       float64
tempo             float64
valence           float64
year                int64
dtype: object
```

In [ ]:

```
df.tail()
```

Out[ ]:

	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key
174384	0.00917	Tony T	0.792	147615	0.866	0	46LhBf6TvYjZU2SMvGZAbn	0.00006	6
174385	0.79500	Alessia Cara	0.429	144720	0.211	0	7tue2Wemjd0FZzRtDrQFZd	0.00000	4
174386	0.80600	Roger Fly	0.671	218147	0.589	0	48Qj61hOdYmUCFJbpQ29Ob	0.92000	4
174387	0.92000	Taylor Swift	0.462	244000	0.240	1	1gcyHQpBQ1lfXGdhZmWrHP	0.00000	0
174388	0.23900	Roger Fly	0.677	197710	0.460	0	57tgYkWQTNHVFEt6xDKKZj	0.89100	7

In [ ]:

```
filename1 = "data_by_genres.csv"
dfByGenre = pd.read_csv(join(path_prefix, filename1))
```

In [ ]:

```
dfByGenre.describe()
```

Out[ ]:

	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	
count	3232.000000	3232.000000	3.232000e+03	3232.000000	3232.000000	3232.000000	3232.000000	3232.000000	3232.
mean	0.368161	0.540810	2.614567e+05	0.583474	0.254888	0.204010	-10.396208	0.084999	119.
std	0.319697	0.149353	1.255810e+05	0.232931	0.283907	0.103440	5.024042	0.081157	17.
min	0.000001	0.063200	3.094600e+04	0.000953	0.000000	0.016500	-40.637000	0.024300	61.
25%	0.082914	0.450250	2.068497e+05	0.419648	0.013383	0.142317	-12.212486	0.046431	109.
50%	0.280119	0.550334	2.418786e+05	0.622982	0.128343	0.185618	-9.097583	0.060769	120.
75%	0.629265	0.645470	2.873741e+05	0.756599	0.443383	0.233807	-7.038634	0.091795	128.
max	0.996000	0.940000	3.478338e+06	0.999000	0.984000	0.944000	-0.862000	0.956000	211.

In [ ]:

```
dfByGenre.dtypes
```

Out[ ]:

```
genres               object
acousticness        float64
danceability        float64
duration_ms         float64
energy              float64
instrumentalness    float64
liveness            float64
```

```
loudness              float64
speechiness           float64
tempo                 float64
valence               float64
popularity            float64
key                     int64
mode                    int64
dtype: object
```

In [ ]:

```
genre_group=dfByGenre.groupby(by="genres")
genre_group["popularity"].mean()
```

Out[ ]:

```
genres
21st century classical     6.600000
432hz                     41.200000
8-bit                      0.000000
[]                        12.350770
a cappella                39.086248
                              ...
zim urban groove           9.000000
zolo                      31.108254
zouk                      32.555556
zurich indie               0.000000
zydeco                    27.703810
Name: popularity, Length: 3232, dtype: float64
```

In [ ]:

```
filename2 = "data_by_artist.csv"
dfByArtist = pd.read_csv(join(path_prefix, filename2))
```

In [ ]:

```
dfByArtist.describe()
```

Out[ ]:

	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness
count	32539.000000	32539.000000	3.253900e+04	32539.000000	32539.000000	32539.000000	32539.000000	32539.000000
mean	0.455682	0.552737	2.485982e+05	0.525908	0.233281	0.209035	-11.008171	0.092047
std	0.383678	0.176445	1.581808e+05	0.267402	0.334487	0.151781	5.568630	0.111257
min	0.000000	0.000000	8.042000e+03	0.000000	0.000000	0.000000	-60.000000	0.000000
25%	0.055155	0.436487	1.804616e+05	0.299000	0.000012	0.109000	-13.792500	0.039700
50%	0.391833	0.563000	2.215070e+05	0.537375	0.009550	0.162000	-9.887000	0.052822
75%	0.876000	0.684500	2.815870e+05	0.747039	0.451149	0.259500	-6.944500	0.090800
max	0.996000	0.987000	4.696690e+06	1.000000	1.000000	0.986000	3.367000	0.971000

In [ ]:

```
dfByArtist.dtypes
```

Out[ ]:

```
artists              object
acousticness        float64
danceability        float64
duration_ms         float64
energy              float64
instrumentalness    float64
liveness            float64
loudness            float64
speechiness         float64
```

```
tempo               float64
valence             float64
popularity          float64
key                   int64
mode                  int64
count                 int64
dtype: object
```

In [ ]:

```
filename3 = "data_by_year.csv"
dfByYear = pd.read_csv(join(path_prefix, filename3))
```

In [ ]:

```
dfByYear.describe()
```

Out[ ]:

	year	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechi
count	102.000000	102.000000	102.000000	102.000000	102.000000	102.000000	102.000000	102.000000	102.00
mean	1970.500000	0.548818	0.536939	228698.383547	0.455189	0.214148	0.212475	-12.192694	0.11
std	29.588849	0.271056	0.049568	28440.517398	0.164968	0.109135	0.016892	3.048535	0.10
min	1920.000000	0.189632	0.415141	140135.140496	0.208856	0.099986	0.162309	-20.840083	0.05
25%	1945.250000	0.298426	0.501910	211933.800750	0.280230	0.124572	0.200305	-14.298878	0.06
50%	1970.500000	0.458340	0.541315	238300.393513	0.494563	0.182022	0.212563	-11.849556	0.08
75%	1995.750000	0.843186	0.573307	249445.435116	0.595108	0.283295	0.222198	-9.756256	0.10
max	2021.000000	0.962702	0.655929	284759.933638	0.694245	0.581701	0.262480	-7.376558	0.62

In [ ]:

```
filename4 = "data_w_genres.csv"
df_genre = pd.read_csv(join(path_prefix, filename4))
```

In [ ]:

```
df_genre.describe()
```

Out[ ]:

	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness
count	32539.000000	32539.000000	3.253900e+04	32539.000000	32539.000000	32539.000000	32539.000000	32539.000000
mean	0.455682	0.552737	2.485982e+05	0.525908	0.233281	0.209035	-11.008171	0.092047
std	0.383678	0.176445	1.581808e+05	0.267402	0.334487	0.151781	5.568630	0.111257
min	0.000000	0.000000	8.042000e+03	0.000000	0.000000	0.000000	-60.000000	0.000000
25%	0.055155	0.436487	1.804616e+05	0.299000	0.000012	0.109000	-13.792500	0.039700
50%	0.391833	0.563000	2.215070e+05	0.537375	0.009550	0.162000	-9.887000	0.052822
75%	0.876000	0.684500	2.815870e+05	0.747039	0.451149	0.259500	-6.944500	0.090800
max	0.996000	0.987000	4.696690e+06	1.000000	1.000000	0.986000	3.367000	0.971000

In [ ]:

```
df_genre2=df_genre.drop(["acousticness", "danceability","duration_ms","energy", "instrume
ntalness", "liveness", "loudness", "speechiness", "tempo", "valence","popularity", "key"
, "mode", "count"], axis=1)
df_genre2
```

Out[ ]:

	artists	genres
0	"Cats" 1981 Original London Cast	['show tunes']
1	"Cats" 1983 Broadway Cast	[]
2	"Fiddler On The Roof" Motion Picture Chorus	[]
3	"Fiddler On The Roof" Motion Picture Orchestra	[]
4	"Joseph And The Amazing Technicolor Dreamcoat"...	[]
...	...	...
32534	김효근	[]
32535	나율	[]
32536	미스티	[]
32537	시온 Zion & 한해 Hanhae of 팬텀 Phantom	[]
32538	조정현	['classic korean pop']

32539 rows × 2 columns

In [ ]:

```
dfmerged = pd.merge(df, df_genre2, on="artists")
DfMerged = dfmerged
#DfMerged = merged.drop(["acousticness_y", "danceability_y","duration_ms_y","energy_y","instrumentalness_y","liveness_y","loudness_y","speechiness_y","tempo_y","valence_y","popularity_y","key_y","mode_y"], axis=1)
```

In [ ]:

```
DfMerged.head()
```

Out[ ]:

	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	live
0	0.991	Mamie Smith	0.598	168333	0.2240	0	0cS0A1fUEUd1EW3FcF8AEI	0.000522	5	0
1	0.993	Mamie Smith	0.647	163827	0.1860	0	11m7laMUgmOKql3oYzuhne	0.000018	0	0
2	0.992	Mamie Smith	0.782	195200	0.0573	0	5DlCyqLyX2AOVDTjjkDZ8x	0.000002	5	0
3	0.995	Mamie Smith	0.482	198000	0.2290	0	0lqEx4vktZP1y9hnwfF27Y	0.000061	7	0
4	0.992	Mamie Smith	0.574	189800	0.1380	0	4HYmmG8uHL2hP4zSFWavKF	0.000492	3	0

In [ ]:

```
DfMerged['genres'] = DfMerged['genres'].apply(lambda x: re.findall(r'[\'](\w\s-]+)[\']', x))
```

In [ ]:

```
DfMerged = DfMerged.explode('genres')
```

In [ ]:

```
DfMerged.shape
```

Out[ ]:

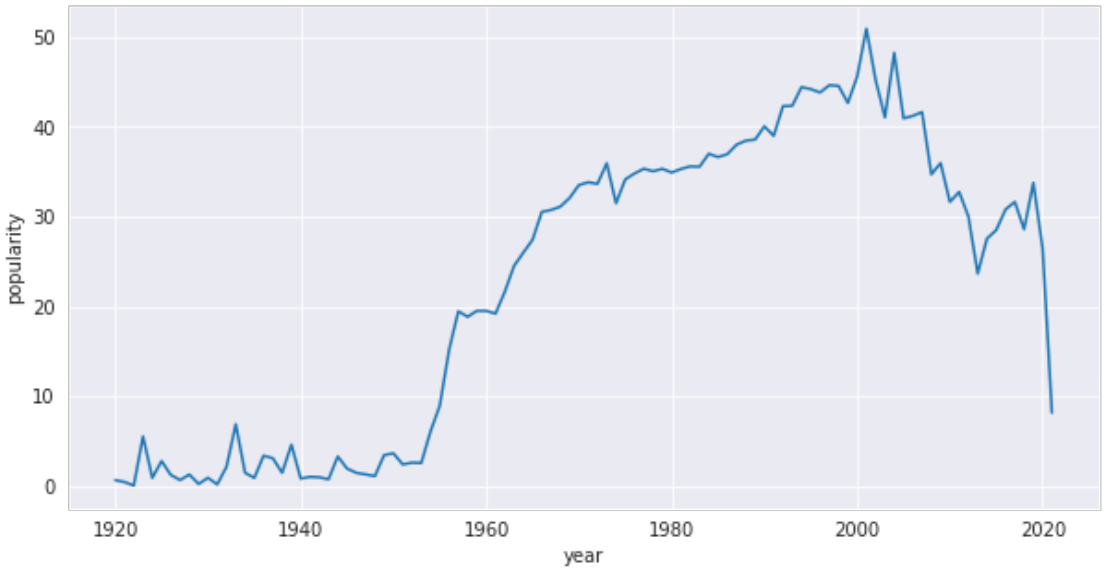
```
(840058, 20)
```

Exploratory Data Analysis

Visualizations Descriptive Statistics

In [ ]:

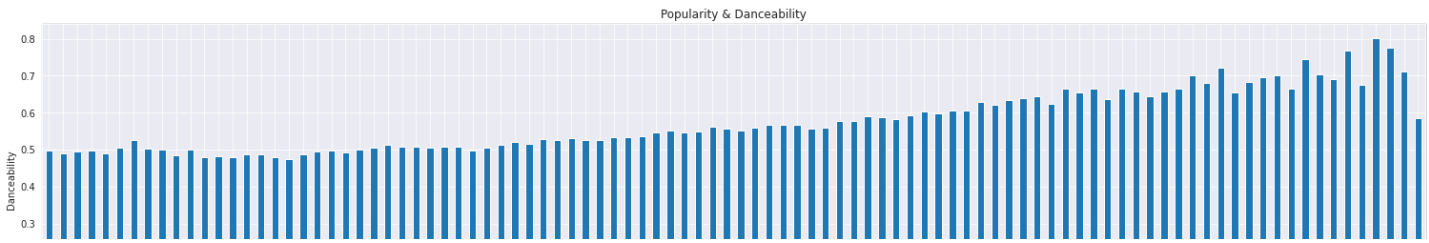
```
fig, ax = plt.subplots(1, 1, figsize=(10, 5))
df.groupby("year").popularity.mean().plot()
plt.ylabel("popularity");
```

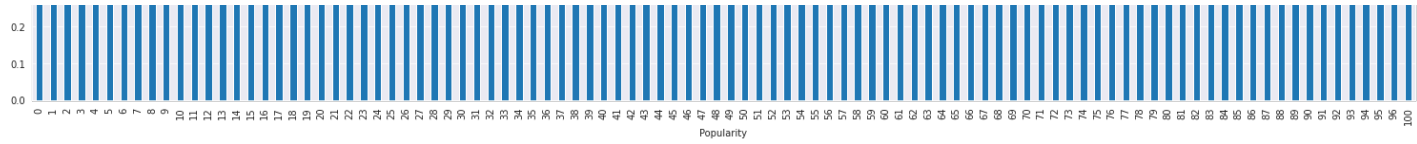


> **This figure Shows the Average popularity of songs according to year. In early 2000's there is a peak this would suggest that those year's music are more poupular however there might exist a recency bias.**

In [ ]:

```
fig, ax = plt.subplots(1, 1, figsize=(26, 6))

ax = df.groupby(by = "popularity").mean().sort_values(by = "popularity")['danceability']
.plot.bar()

plt.title("Popularity & Danceability")
plt.ylabel("Danceability")
plt.xlabel("Popularity")
plt.show();
```



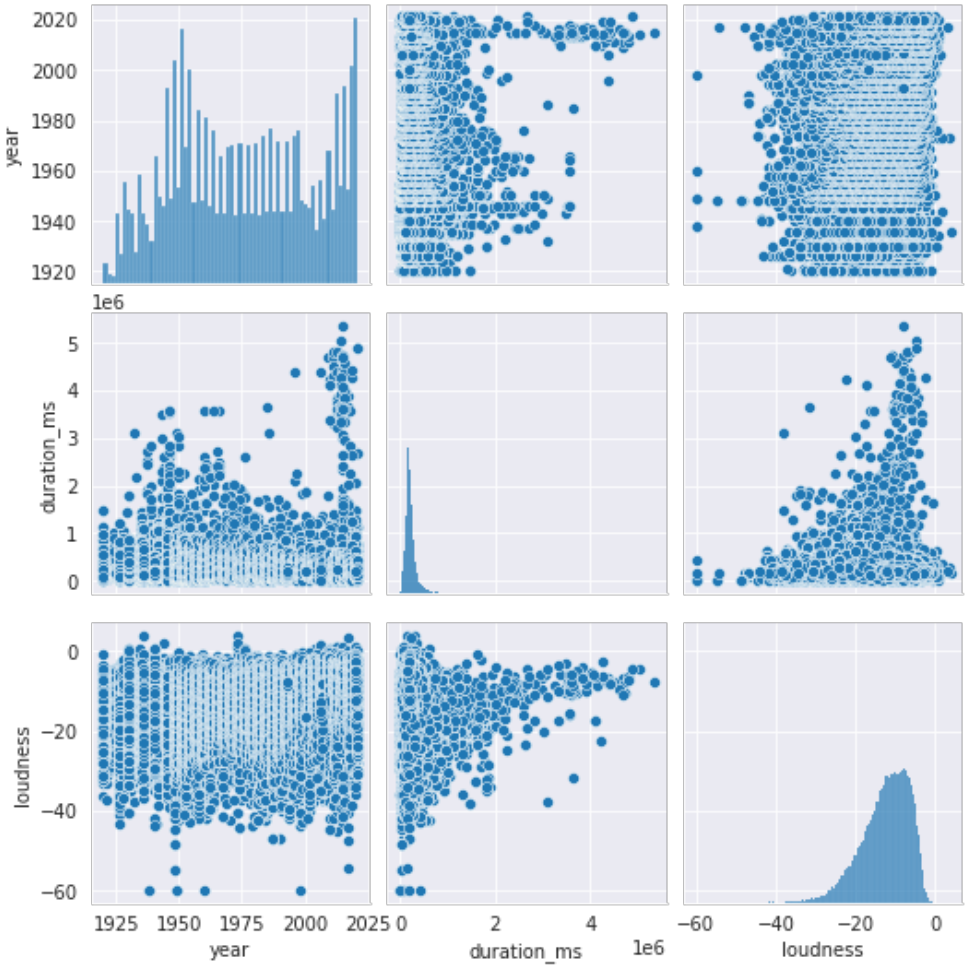


> **This figure represents the relationship between song's popularity and it's danceability feature. The bar chart suggests positive correlation between Popularity & Danceability. And danceability peaks at 0.988000 when there is 94% popularity.**

Visualizations of Aggregated Forms Based on Features

In [ ]:

```
sns.pairplot(df[['year', 'duration_ms', 'loudness']]);
```

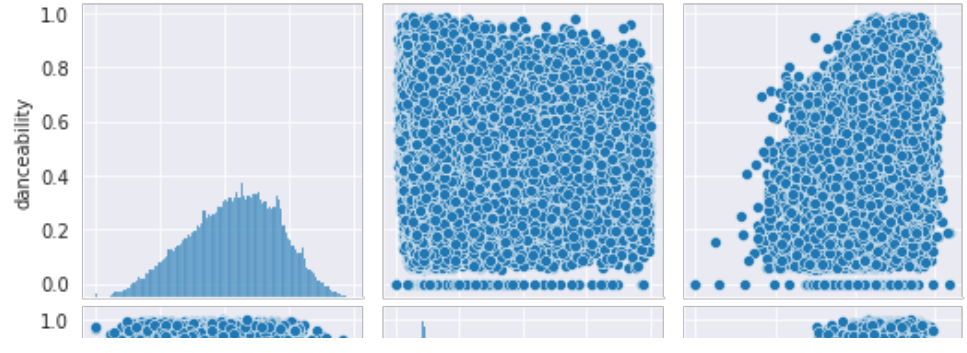


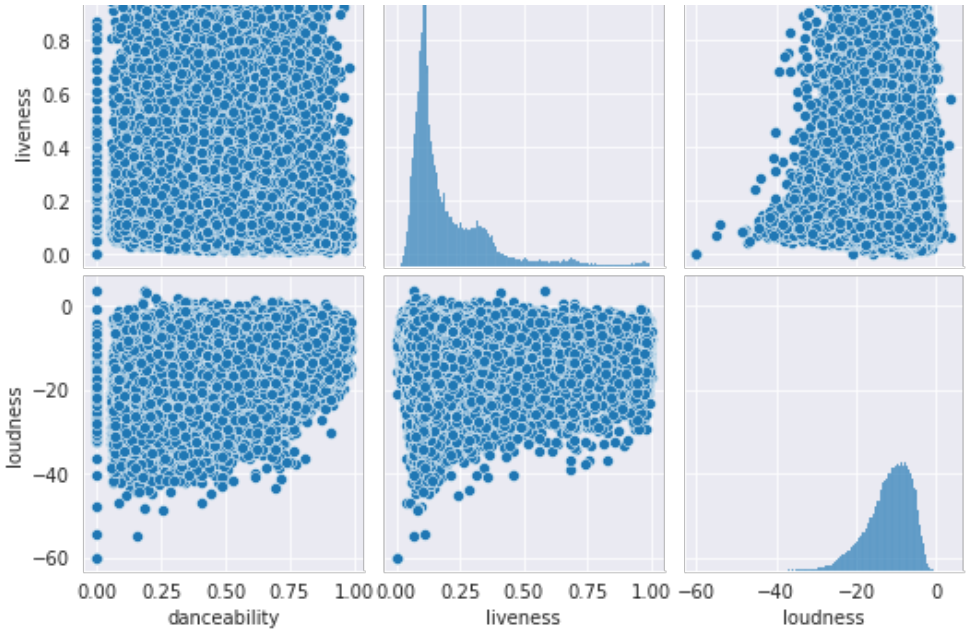
In [ ]:

```
sns.pairplot(df[['danceability', 'liveness', 'loudness']]);
```

Out[ ]:

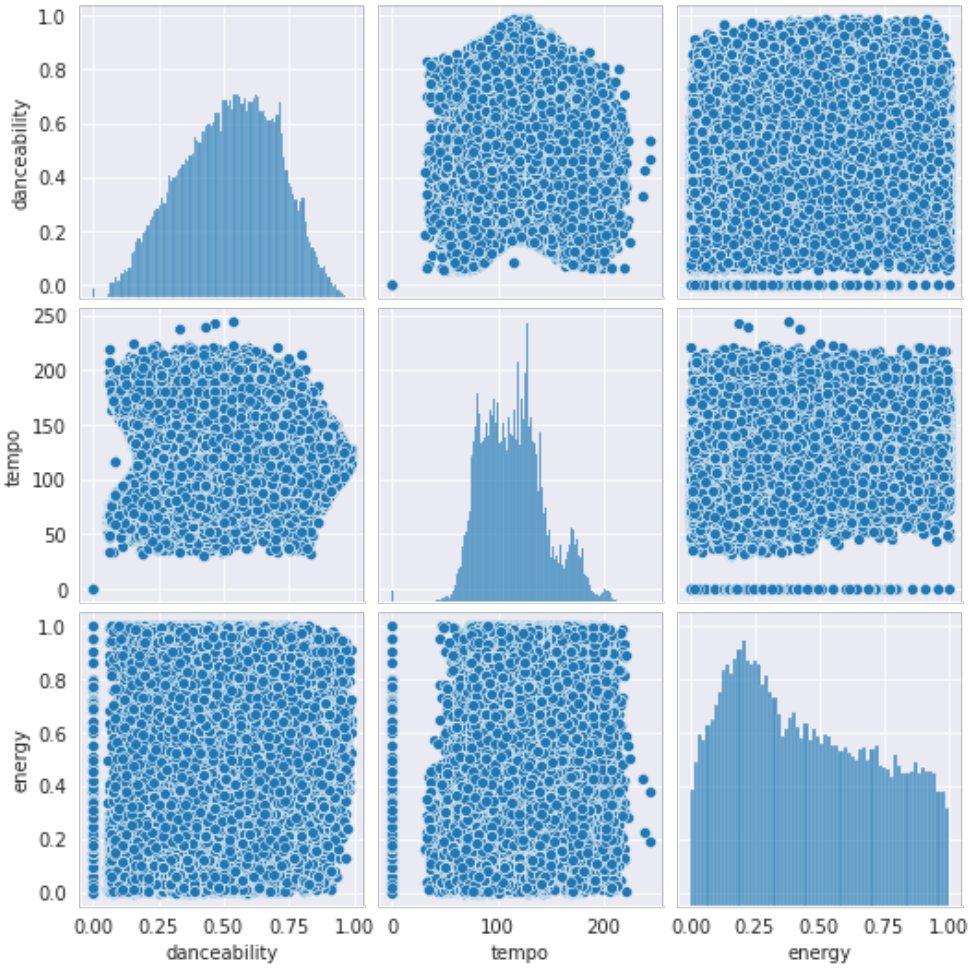
```
<seaborn.axisgrid.PairGrid at 0x7ff3985ae0d0>
```





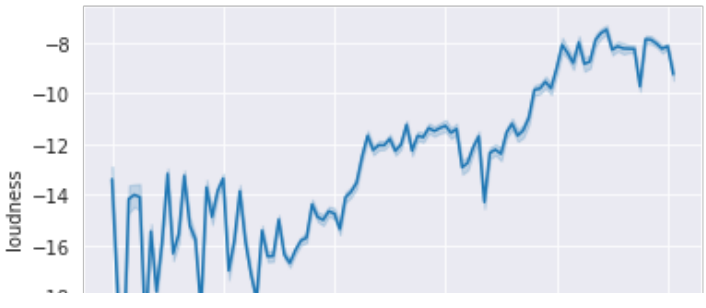
In [ ]:

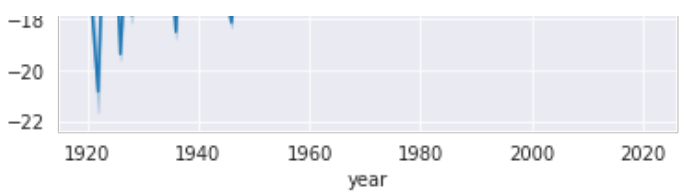
```
sns.pairplot(df[['danceability', 'tempo', 'energy']]);
```



In [ ]:

```
sns.lineplot(x="year", y="loudness", data=df);
```

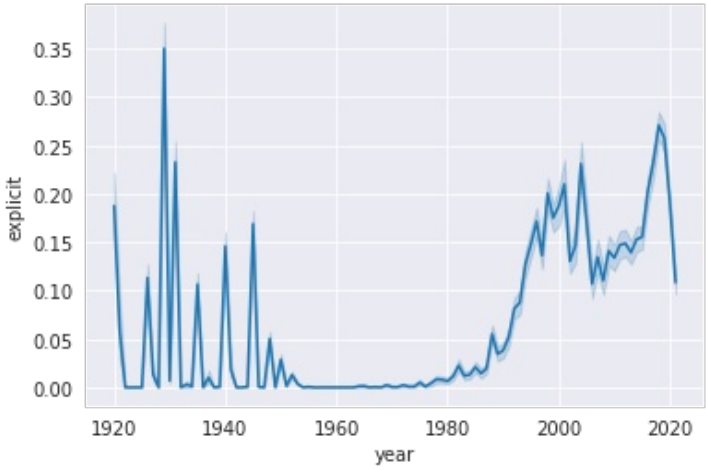




> **This Plot shows the correlation beetween Years and The Mean of Loudness. It can be seen that there is a positive correlation, As the years went on loudness usually increased. This would be caused by many reasons such as the taste in music or advancement in music recording.**

In [ ]:

```
sns.lineplot(x="year", y="explicit", data=df);
```



> **This line chart show how mean measurement explicity of songs changes over years. In 1930s there is a peak in explicity and between the years of 1960 and 1980 explicity is relatively stable.**

In [ ]:

```
dfByGenre.groupby(by="genres").mean()
```

Out[ ]:

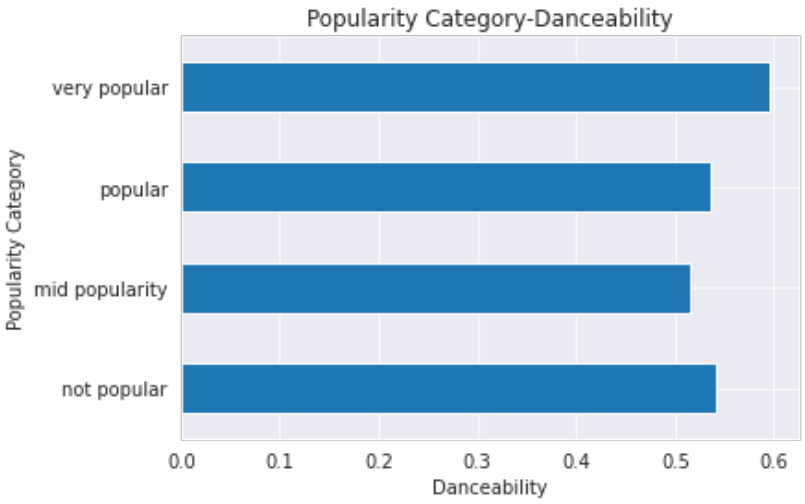
genres	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo
21st century classical	0.754600	0.284100	3.525932e+05	0.159580	0.484374	0.168580	-22.153400	0.062060	91.351000
432hz	0.485515	0.312000	1.047430e+06	0.391678	0.477250	0.265940	-18.131267	0.071717	118.900933
8-bit	0.028900	0.673000	1.334540e+05	0.950000	0.630000	0.069000	-7.899000	0.292000	192.816000
[]	0.535793	0.546937	2.495312e+05	0.485430	0.278442	0.220970	-11.624754	0.101511	116.068980
a cappella	0.694276	0.516172	2.018391e+05	0.330533	0.036080	0.222983	-12.656547	0.083627	105.506031
...	...	...	...	...	...	...	...	...	...
zim urban groove	0.003910	0.553000	4.267200e+04	0.942000	0.961000	0.113000	-8.004000	0.039900	134.995000
zolo	0.208648	0.533837	2.641016e+05	0.620470	0.163334	0.201430	-10.878906	0.061828	126.765194

	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo
zouk	0.272928	0.641889	4.416418e+05	0.695778	0.257604	0.166011	-9.518889	0.050511	105.848889
zurich genres indie	0.993000	0.705667	1.984173e+05	0.172667	0.468633	0.179667	-11.453333	0.348667	91.278000
zydeco	0.381496	0.625963	1.840881e+05	0.572417	0.019234	0.262545	-10.643750	0.072440	130.137242

3232 rows × 13 columns

In [ ]:

```
def categorize_popularity(pop):
  if pop < 15:
    return "not popular"
  elif pop< 30:
    return "mid popularity"
  elif pop < 50:
    return "popular"
  elif pop < 100:
    return "very popular"
  return "not popular"

dfByGenre["Popularity Category"] = dfByGenre["popularity"].apply(categorize_popularity)

ax = dfByGenre.groupby("Popularity Category").mean()["danceability"][["not popular", "mid popularity","popular", "very popular"]].plot.barh(grid=True)
ax.set_xlabel("Danceability")
ax.set_title("Popularity Category-Danceability");
```



> **This Graph Categorizes songs according to their mean popularity and then compares their mean Danceability. It can be observed that Popular and Not Popular songs have more dancability. This might be due to the fact that Danceability in songs are a hit or miss situation.**

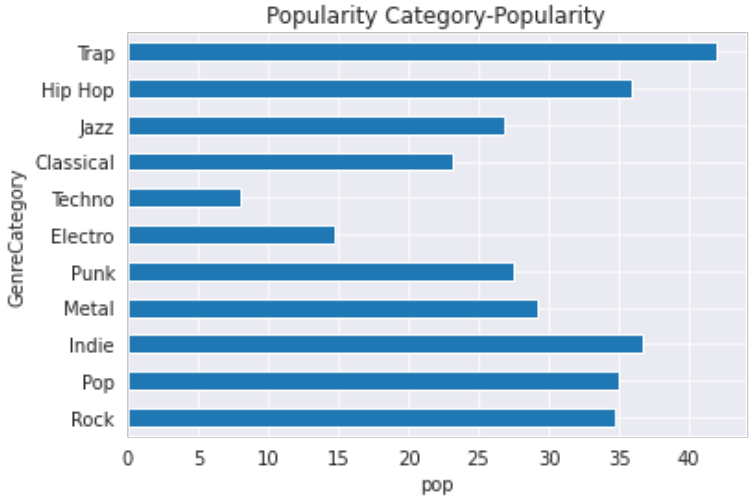
In [ ]:

```
def categorize_genre(genre):
  if (genre.find("rock")!=-1):
    return "Rock"
  elif (genre.find("pop")!=-1):
    return "Pop"
  elif (genre.find("indie")!=-1):
    return "Indie"
  elif (genre.find("metal")!=-1):
    return "Metal"
  elif (genre.find("punk")!=-1):
    return "Punk"
  elif (genre.find("electro")!=-1):
    return "Electro"
  elif (genre.find("techno")!=-1):
```

```
    return "Techno"
  elif (genre.find("classical")!=-1):
    return "Classical"
  elif (genre.find("jazz")!=-1):
    return "Jazz"
  elif (genre.find("hip hop")!=-1):
    return "Hip Hop"
  elif (genre.find("trap")!=-1):
    return "Trap"



dfByGenre["GenreCategory"] = dfByGenre["genres"].apply(categorize_genre)
```

In [ ]:

```
ax = dfByGenre.groupby("GenreCategory").mean()["popularity"][["Rock", "Pop","Indie", "Metal","Punk","Electro","Techno","Classical","Jazz","Hip Hop","Trap"]].plot.barh(grid=True)
ax.set_xlabel("pop")
ax.set_title("Popularity Category-Popularity");
```



> **This Graph Compares The Popularity of Songs from different genres. We can see that Hip Hop and its Subgenre Trap is the highest ranked 1st and 3rd genres. This may show us that it is a pretty mainstream type of music.**

Analysis of the Most Popular Artists and Songs

In [ ]:

```
#if categorize_popularity(pop) == "popular":
df_new = df.loc[df["popularity"] > 90]
df_new.groupby(by = "popularity").mean().sort_values(by = "popularity", ascending = False)
#fig, ax = plt.subplots(1, 1, figsize=(20, 5))
#df_new.groupby("artists").popularity.mean().plot(kind = "bar")
```

Out[ ]:

popularity	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness	loudness	
100	0.721000	0.585000	242014.000000	0.436000	1.000000	1.310000e-05	10.000000	0.105000	-8.761000	1.(
96	0.344500	0.718500	156425.500000	0.762000	1.000000	0.000000e+00	3.500000	0.182550	-4.164500	0.5
95	0.306500	0.797000	192018.500000	0.619500	1.000000	2.726000e-04	2.500000	0.108000	-7.108500	0.(

	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness	loudness	0.(
94	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	4.200[illegible]	[illegible]	-6.823800	
popularity										
93	0.718000	0.675500	164633.000000	0.461000	0.500000	7.950000e-07	7.000000	0.111800	-8.939500	1.(
92	0.326733	0.764167	172124.666667	0.519667	0.500000	3.314688e-03	5.166667	0.101600	-6.937000	0.5
91	0.231208	0.654154	199612.846154	0.634154	0.307692	5.515385e-06	4.923077	0.221946	-6.208538	0.5

In [ ]:

```
df_new = df.loc[df["popularity"] > 90]
df_new.groupby(by = ["name","artists","popularity"]).mean().sort_values(by = "popularity
", ascending = False)
```

Out[ ]:

name	artists	popularity	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness
drivers license	['Olivia Rodrigo']	100	0.72100	0.585	242014	0.436	1	0.000013	10	0.1050
positions	['Ariana Grande']	96	0.46800	0.737	172325	0.802	1	0.000000	0	0.0931
Mood (feat. iann dior)	['24kGoldn', 'iann dior']	96	0.22100	0.700	140526	0.722	1	0.000000	7	0.2720
BICHOTA	['KAROL G']	95	0.21200	0.863	178947	0.666	1	0.000493	1	0.1030
DÁKITI	['Bad Bunny', 'Jhay Cortez']	95	0.40100	0.731	205090	0.573	1	0.000052	4	0.1130
34+35	['Ariana Grande']	94	0.23700	0.830	173711	0.585	1	0.000000	0	0.2480
LA NOCHE DE ANOCHE	['Bad Bunny', 'ROSALÍA']	94	0.03030	0.856	203201	0.618	0	0.000000	7	0.0866
Whoopty	['CJ']	94	0.19000	0.711	123429	0.598	1	0.000000	3	0.1500
WITHOUT YOU	['The Kid LAROI']	94	0.21300	0.662	161385	0.413	1	0.000000	0	0.1340
Therefore I Am	['Billie Eilish']	94	0.21800	0.889	174321	0.340	0	0.130000	11	0.0550
What You Know Bout Love	['Pop Smoke']	93	0.65000	0.709	160000	0.548	1	0.000002	10	0.1330
you broke me first	['Tate McRae']	93	0.78600	0.642	169266	0.374	0	0.000000	4	0.0906
Hecha Pa' Mi	['Boza']	92	0.36200	0.725	186133	0.756	0	0.000685	4	0.1030
Lonely (with benny blanco)	['Justin Bieber', 'benny blanco']	92	0.86400	0.631	149297	0.239	1	0.000000	11	0.1160
WAP (feat. Megan Thee Stallion)	['Cardi B', 'Megan Thee Stallion']	92	0.01940	0.935	187541	0.454	1	0.000000	1	0.0824
HOLIDAY	['Lil Nas X']	92	0.12000	0.810	154998	0.511	1	0.000000	5	0.0832
The Business	['Tiësto']	92	0.41400	0.798	164000	0.620	0	0.019200	8	0.1120

name	artists	popularity	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness
Anyone	['Justin Bieber']	92	0.18100	0.686	190779	0.598	0	0.000003	2	0.1130
Monster (Shawn Mendes & Justin Bieber)	['Shawn Mendes', 'Justin Bieber']	91	0.06760	0.652	178994	0.383	0	0.000000	2	0.0828
Head & Heart (feat. MNEK)	['Joel Corry', 'MNEK']	91	0.16800	0.734	166028	0.874	0	0.000011	8	0.0489
ROCKSTAR (feat. Roddy Ricch)	['DaBaby', 'Roddy Ricch']	91	0.24700	0.746	181733	0.690	1	0.000000	11	0.1010
Snowman	['Sia']	91	0.48300	0.716	165907	0.512	0	0.000000	1	0.0928
Heather	['Conan Gray']	91	0.58400	0.357	198040	0.425	0	0.000000	5	0.3220
La Nota	['Manuel Turizo', 'Rauw Alejandro', 'Myke Towers']	91	0.18000	0.736	216107	0.632	0	0.000000	11	0.3490
Good Days	['SZA']	91	0.49900	0.436	279204	0.655	1	0.000008	1	0.6880
Watermelon Sugar	['Harry Styles']	91	0.12200	0.548	174000	0.816	0	0.000000	0	0.3350
For The Night (feat. Lil Baby & DaBaby)	['Pop Smoke', 'Lil Baby', 'DaBaby']	91	0.11400	0.823	190476	0.586	1	0.000000	6	0.1930
DÁKITI	['Bad Bunny', 'Jhay Cortez']	91	0.40100	0.731	205090	0.573	1	0.000052	4	0.1130
Dynamite	['BTS']	91	0.01120	0.746	199054	0.765	0	0.000000	6	0.0936
Bandido	['Myke Towers', 'Juhn']	91	0.12200	0.713	232853	0.617	0	0.000000	8	0.0962
Life Goes On	['BTS']	91	0.00691	0.566	207481	0.716	0	0.000000	1	0.3700

MOST POPULAR SONGS:

100- *drivers license* - Olivia Rodrigo

96- *positions* - Ariana Grande

96- *Mood (feat. iann dior)* - 24kGoldn, iann dior

In [ ]:

```
dfByArtist_new = dfByArtist.loc[dfByArtist["popularity"] > 85]
dfByArtist_new = dfByArtist_new.drop(["acousticness", "danceability","duration_ms","energy", "instrumentalness", "liveness", "loudness", "speechiness", "tempo", "valence", "key", "mode", "count"], axis=1)
#dfByArtist_new.groupby(by = ["artists","popularity"]).mean().sort_values(by = "popularity", ascending = False)
dfByArtist_new.groupby(by = ["artists"]).mean().sort_values(by = "popularity", ascending = False)
```

Out[ ]:

artists	popularity

artists	popularity
CJ	94.000000
Boza	92.000000
Joel Corry	91.000000
Juhn	91.000000
Lele Pons	89.000000
Los Legendarios	89.000000
Ritt Momney	89.000000
KIDDO	87.000000
DJ Nelson	86.000000
Emilee	86.000000
Jay Wheeler	86.000000
Jerry Di	86.000000
Nuka	86.000000
Surf Mesa	86.000000
Chencho Corleone	85.666667
Tones And I	85.500000

MOST POPULAR ARTISTS:

94- CJ

92- Boza

91- Joel Corry

91- Juhn

89- Lele Pons

How Songs of Different Genres Change Over Time

In [ ]:

```
###
def categorize_years(year):
  if  year< 1930:
    return "1920s"
  elif year< 1940:
    return "1930s"
  elif year< 1950:
    return "1940s"
  elif year< 1960:
    return "1950s"
  elif year< 1970:
    return "1960s"
  elif year < 1980:
    return "1970s"
  elif year < 1990:
    return "1980s"
  elif year < 2000:
    return "1990s"
  elif year < 2010:
    return "2000s"
  elif year < 2020:
    return "2010s"
  elif year > 2020:
    return "2020-"

dfByYear["Years Category"] = dfByYear["year"].apply(categorize_years)
```

In [ ]:

```
dfByYear.groupby(by = ["Years Category","year","popularity"]).mean().sort_values(by = "year", ascending = False)
```

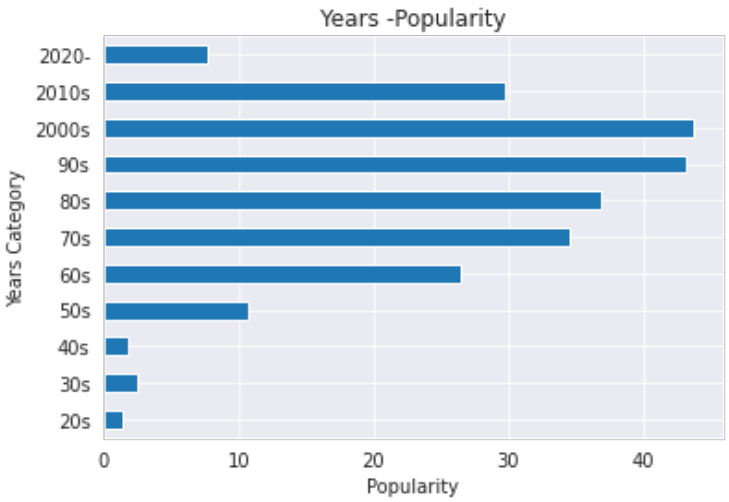
Out[ ]:

Years Category	year	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speech
2020-	2021	7.750543	0.340253	0.652488	222760.028261	0.578896	0.356765	0.162309	-9.547752	0.1
2010s	2019	32.346930	0.261344	0.603101	225463.326320	0.629781	0.221123	0.218501	-8.527674	0.1
	2018	27.599484	0.233836	0.602731	227073.140015	0.659552	0.229874	0.231009	-8.173434	0.1
	2017	31.662338	0.207413	0.580475	243402.249536	0.686392	0.247068	0.231513	-8.026319	0.0
	2016	29.212431	0.226944	0.581583	258001.186462	0.656547	0.203332	0.221417	-8.002876	0.0
...	...	...	...	...	...	...	...	...	...	
20s	1924	0.661017	0.940200	0.549894	191046.707627	0.344347	0.581701	0.235219	-14.231343	0.0
	1923	5.205405	0.957247	0.577341	177942.362162	0.262406	0.371733	0.227462	-14.129211	0.0
	1922	0.090909	0.828934	0.575620	140135.140496	0.226173	0.254776	0.256662	-20.840083	0.4
	1921	0.391026	0.862105	0.432171	257891.762821	0.241136	0.337158	0.205219	-16.811660	0.0
	1920	0.610315	0.631242	0.515750	238092.997135	0.418700	0.354219	0.216049	-12.654020	0.0

101 rows × 12 columns

In [ ]:

```
ax = dfByYear.groupby("Years Category").mean()["popularity"][["20s","30s","40s","50s", "60s","70s", "80s","90s","2000s" ,"2010s","2020-"]].plot.barh(grid=True)
ax.set_xlabel("Popularity")
ax.set_title("Years -Popularity");
```

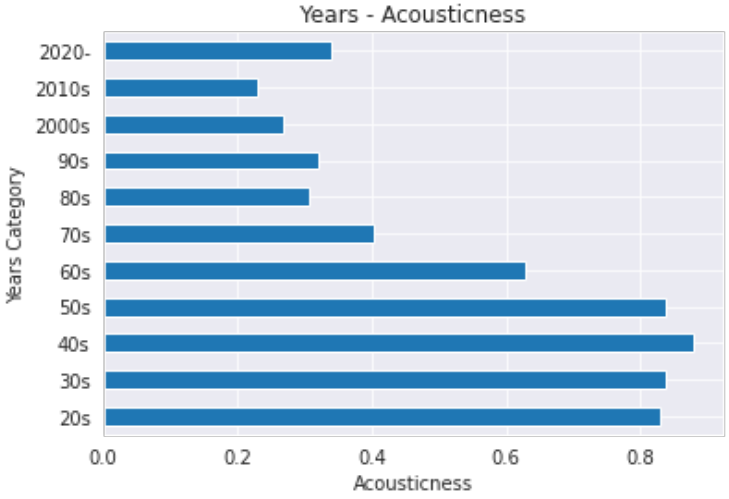


> **Above chart show the relationship between years and Popularity**

In [ ]:

```
ax = dfByYear.groupby("Years Category").mean()["acousticness"][["20s","30s","40s","50s",
```

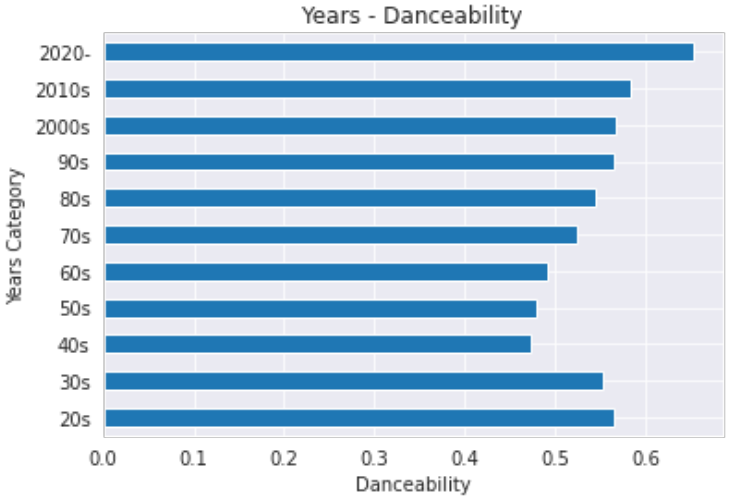
```
"60s","70s", "80s","90s","2000s" ,"2010s","2020-"]].plot.barh(grid=True)
ax.set_xlabel("Acousticness")
ax.set_title("Years - Acousticness");
```



> **Above chart show the relationship between years and Acousticness**

In [ ]:

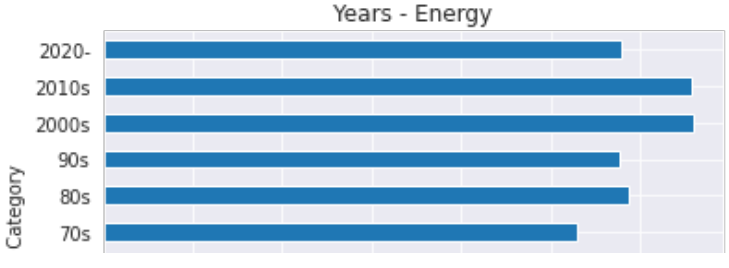
```
ax = dfByYear.groupby("Years Category").mean()["danceability"][["20s","30s","40s","50s",
"60s","70s", "80s","90s","2000s" ,"2010s","2020-"]].plot.barh(grid=True)
ax.set_xlabel("Danceability")
ax.set_title("Years - Danceability");
```

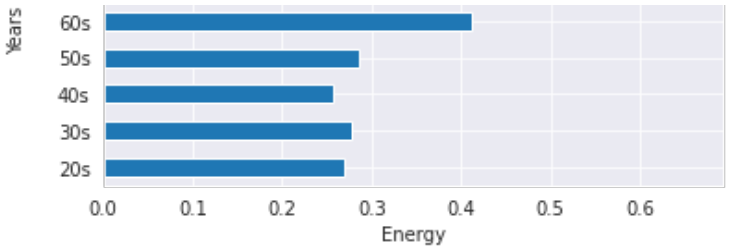


> **Above chart show the relationship between years and Danceability**

In [ ]:

```
ax = dfByYear.groupby("Years Category").mean()["energy"][["20s","30s","40s","50s", "60s"
,"70s", "80s","90s","2000s" ,"2010s","2020-"]].plot.barh(grid=True)
ax.set_xlabel("Energy")
ax.set_title("Years - Energy");
```

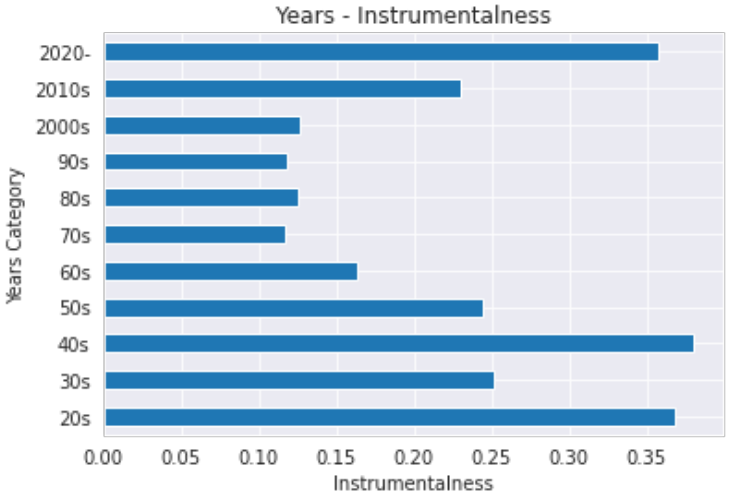




> **Above chart show the relationship between years and Energy**

In [ ]:

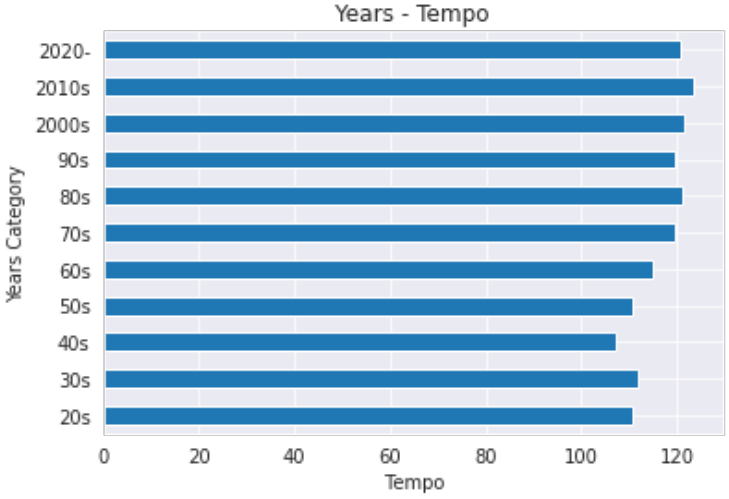
```
ax = dfByYear.groupby("Years Category").mean()["instrumentalness"][["20s","30s","40s","50s", "60s","70s", "80s","90s","2000s" ,"2010s","2020-"]].plot.barh(grid=True)
ax.set_xlabel("Instrumentalness")
ax.set_title("Years - Instrumentalness");
```



> **Above chart show the relationship between years and Instrumentalness.**
> **Instrumentalnessflavtuates a lot, peaks at 40s and has trough between 60s and 2000s.**

In [ ]:

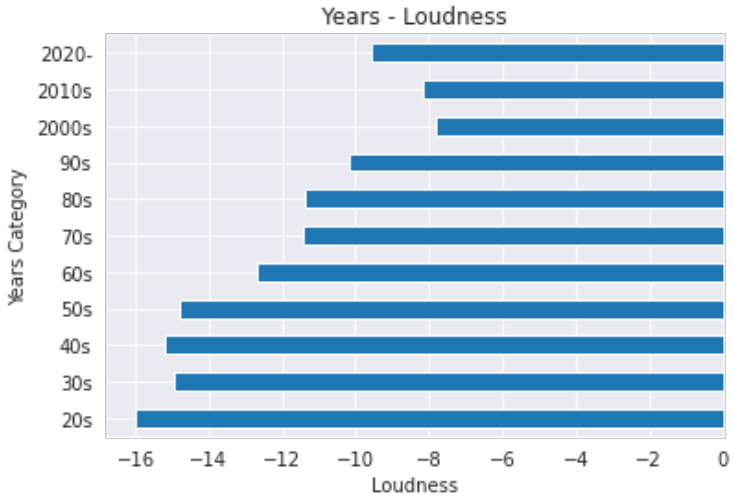
```
ax = dfByYear.groupby("Years Category").mean()["tempo"][["20s","30s","40s","50s", "60s", "70s", "80s","90s","2000s" ,"2010s","2020-"]].plot.barh(grid=True)
ax.set_xlabel(" Tempo")
ax.set_title("Years - Tempo");
```



Above bar chart represents how tempo changes over years.It averages between 100 and 120 and peaks at 2010s.

In [ ]:

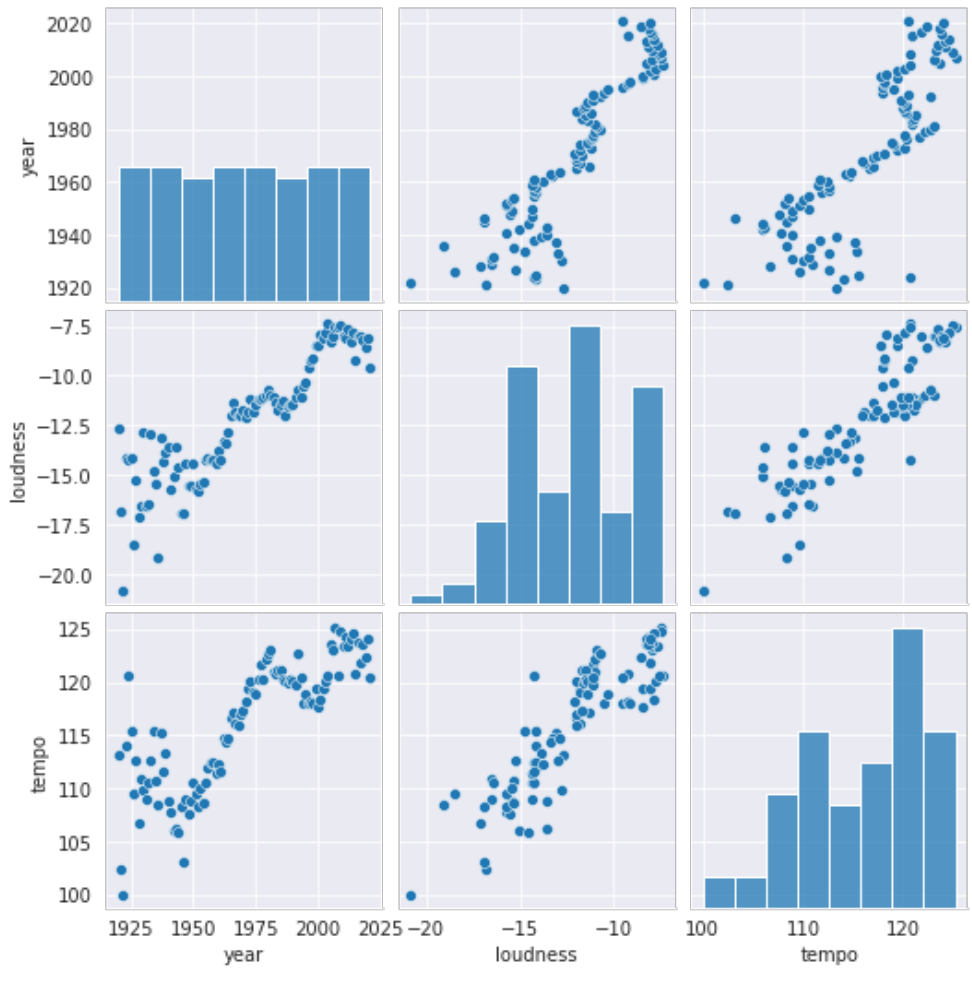
```
ax = dfByYear.groupby("Years Category").mean()["loudness"][["20s","30s","40s","50s", "60s","70s", "80s","90s","2000s" ,"2010s","2020-"]].plot.barh(grid=True)
ax.set_xlabel("Loudness")
ax.set_title("Years - Loudness");
```



Above bar chart show the loudness of songs over years. Loudness is between -16 and 0. It decreases over years but then also starts to increase after 2000s.

In [ ]:

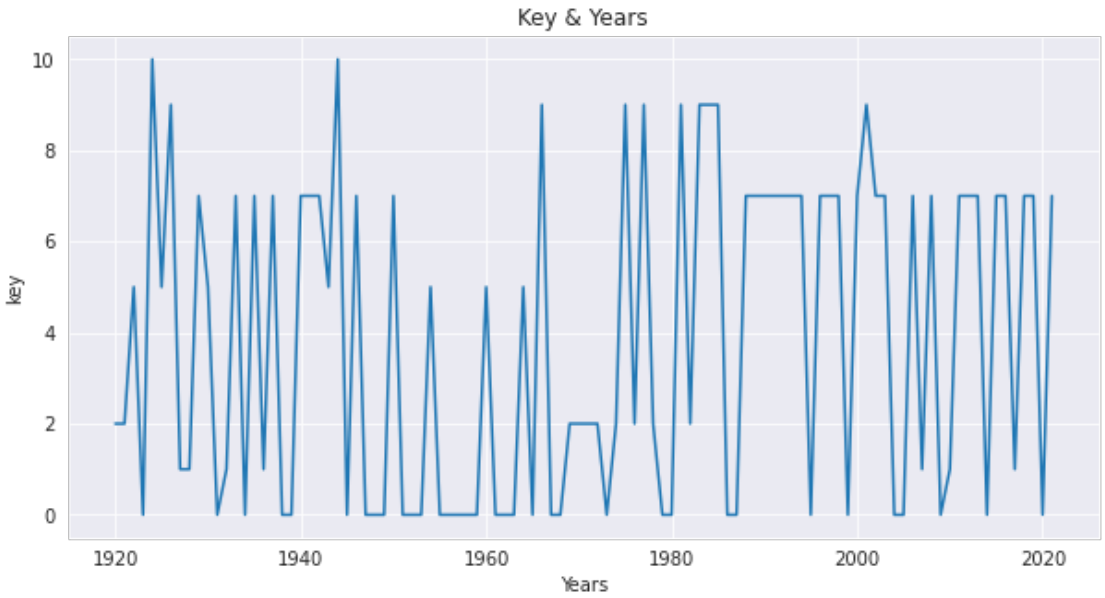
```
sns.pairplot(dfByYear[["year", 'loudness', 'tempo']]);
```



> **The Figures above show some correlation between different parameters**

In [ ]:

```
fig, ax = plt.subplots(1, 1, figsize=(10, 5))
dfByYear.groupby("year").key.mean().plot()
plt.title("Key & Years")
plt.ylabel("key")
plt.xlabel("Years");
```



> **Above figure show how the mean of keys changes over years. The keys flactuates alot has many ups and downs.**

In [ ]:

```
###
def popular_genres_cat(genres):

  if (genres==("rock")):
    return "Rock"
  elif (genres==("pop")):
    return "Pop"
  elif (genres==("rap")):
    return "Rap"
  elif (genres==("classical")):
    return "Classical Music"
  elif (genres==("alternative rock")):
    return "Alternative Rock"
  elif (genres==("hip hop")):
    return "Hip-Hop"


DfMerged["Some Genres"] = DfMerged["genres"].apply(popular_genres_cat)
DfMerged["Years Category"] = DfMerged["year"].apply(categorize_years)
```

> **Below table shows some specific genres *(Rap, Rock,Pop,Classical Music,Alternative Rock,Hip-Hop)* and their features , how they change over time.**

In [ ]:

```
DfMerged.groupby(by = ["Years Category","Some Genres"]).mean().sort_values(by = "year", ascending = False)
```

Out[ ]:

Years Category	Some Genres	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness	lo
2020-	Rock	0.432153	0.624167	190790.000000	0.692917	0.000000	0.000524	1.833333	0.213692	-6
	Rap	0.176014	0.701435	211835.177419	0.626565	0.935484	0.000004	2.887097	0.167777	-7
	Pop	0.346131	0.627225	198679.437500	0.539644	0.325000	0.002431	5.537500	0.181006	-7
	Hip-Hop	0.147439	0.664188	217216.531250	0.645219	0.906250	0.000005	2.843750	0.128006	-7
	Classical Music	0.923058	0.413519	350638.730769	0.255337	0.000000	0.755346	4.826923	0.160658	16
	Alternative Rock	0.003945	0.592000	298274.000000	0.722333	0.000000	0.057533	5.666667	0.189667	-9
2010s	Rap	0.169978	0.695039	236859.526570	0.640182	0.877741	0.010902	5.050539	0.202459	-6
	Pop	0.233774	0.625333	223189.012137	0.636916	0.272054	0.020328	5.190053	0.176850	-6
	Classical Music	0.892247	0.357131	328089.467172	0.266880	0.000000	0.703949	4.944444	0.562757	19
	Hip-Hop	0.177662	0.681504	246771.885847	0.646693	0.841942	0.015845	5.077479	0.214407	-6
	Rock	0.159982	0.514582	242865.865823	0.699787	0.048101	0.077558	5.173840	0.225533	-7
	Alternative Rock	0.137683	0.503872	280271.893004	0.675362	0.032922	0.233340	5.469136	0.237023	-8
2000s	Pop	0.182758	0.634144	244018.194082	0.679991	0.115753	0.022433	5.375979	0.200351	-6
	Rap	0.127904	0.706851	252916.716010	0.717222	0.776366	0.010263	5.592757	0.221548	-5
	Classical Music	0.918809	0.383615	296870.680653	0.244653	0.000000	0.741993	4.536131	0.546053	21
	Hip-Hop	0.141923	0.706910	250713.337371	0.710338	0.732283	0.015215	5.579043	0.225563	-5
	Rock	0.148740	0.494035	251381.966423	0.736896	0.068613	0.100616	5.187591	0.219483	-6
	Alternative Rock	0.140711	0.496151	247556.991758	0.742229	0.048077	0.152483	5.324176	0.209765	-6
1990s	Hip-Hop	0.142458	0.747946	255827.509200	0.650710	0.643974	0.033943	5.660534	0.226171	-8
	Rap	0.129732	0.755190	252770.850932	0.668877	0.717674	0.031807	5.602484	0.229243	-8
	Pop	0.255302	0.643085	258801.355786	0.624257	0.065630	0.023590	5.442142	0.185378	-8
	Alternative Rock	0.153115	0.470870	244022.581182	0.687978	0.090591	0.174242	5.340481	0.193611	-8
	Classical Music	0.922547	0.256810	349070.732919	0.095928	0.000000	0.706015	4.993789	0.129654	24
	Rock	0.138670	0.475283	270277.694014	0.716796	0.105439	0.122255	5.205105	0.225049	-8
1980s	Hip-Hop	0.096507	0.798521	251506.332168	0.662266	0.307692	0.035380	5.520979	0.204370	10
	Rap	0.097070	0.792208	250095.244526	0.655234	0.277372	0.034337	5.459854	0.200909	11
	Pop	0.277633	0.681226	275111.384921	0.632698	0.000000	0.060490	5.742063	0.165054	-9
	Alternative Rock	0.139676	0.472722	212800.471090	0.728324	0.054976	0.224156	5.276777	0.222592	-9
	Classical Music	0.936965	0.301139	294783.891156	0.126387	0.000000	0.673721	4.884354	0.153631	23
	Rock	0.143668	0.506297	263647.340659	0.721380	0.019674	0.105138	5.267281	0.251553	-9
1970s	Alternative Rock	0.151774	0.460397	209353.390164	0.709185	0.019672	0.192798	5.144262	0.199438	-8
	Classical									

	Music	0.924590	0.290248	290512.653944	0.153417	0.000000	0.615614	5.005089	0.184546	20
Years	Some Rock Genres	0.296577	0.493909	259261.096973	0.595665	0.003643	0.102849	5.007988	0.242150	10
	Hip-Hop	0.228272	0.612886	309945.971429	0.598573	0.000000	0.129882	6.014286	0.169030	10
	Rap	0.242612	0.606043	321883.242857	0.608059	0.000000	0.129726	6.057143	0.156317	11
	Pop	0.349030	0.574867	236504.758333	0.623707	0.000000	0.032576	4.541667	0.202088	-9
1960s	Alternative Rock	0.132385	0.376111	236369.037037	0.597444	0.000000	0.143253	5.185185	0.175981	10
	Pop	0.467188	0.561538	203208.307692	0.505677	0.000000	0.081965	5.123077	0.157705	10
	Hip-Hop	0.225100	0.673250	393850.000000	0.562750	0.000000	0.010782	4.250000	0.135175	12
	Rock	0.390742	0.488284	210826.456367	0.551363	0.000323	0.117066	5.279897	0.218650	10
	Rap	0.323733	0.600000	369973.333333	0.454000	0.000000	0.016745	5.000000	0.121467	14
	Classical Music	0.927372	0.293593	328233.123578	0.162589	0.000000	0.688255	4.655800	0.172628	21
1950s	Rock	0.679115	0.611325	167056.950000	0.606625	0.000000	0.164390	3.800000	0.182782	-9
	Pop	0.872556	0.390444	262970.333333	0.224411	0.000000	0.193976	4.666667	0.130489	14
	Classical Music	0.948408	0.329509	285241.532962	0.187980	0.000000	0.455135	5.041783	0.215168	19
	Alternative Rock	0.081300	0.435500	225433.000000	0.602500	0.000000	0.000023	7.500000	0.225500	-9
1940s	Hip-Hop	0.993000	0.572000	194817.000000	0.206000	0.000000	0.011600	9.000000	0.063400	-9
	Classical Music	0.951817	0.343762	284694.050165	0.167170	0.000000	0.611269	5.090224	0.191706	19
	Rock	0.938083	0.522417	176814.416667	0.304017	0.000000	0.352121	6.333333	0.325183	13
	Pop	0.728390	0.463954	181843.734043	0.385382	0.000000	0.052870	4.617021	0.217821	11
1930s	Hip-Hop	0.514000	0.600500	180678.000000	0.358350	0.000000	0.214300	6.500000	0.093200	15
	Classical Music	0.935505	0.321742	370612.102347	0.173647	0.000000	0.587529	5.017840	0.216469	18
	Rap	0.034000	0.844000	175800.000000	0.618000	0.000000	0.065600	2.000000	0.060400	16
	Pop	0.404034	0.537200	219839.760000	0.649840	0.000000	0.135226	5.840000	0.232356	-7
1920s	Hip-Hop	0.968000	0.502000	184692.000000	0.200000	0.000000	0.000000	7.000000	0.345000	12
	Classical Music	0.981548	0.338035	298072.004065	0.131515	0.000000	0.807815	5.524390	0.185778	21
	Pop	0.528364	0.719455	203707.636364	0.400636	1.000000	0.000007	5.363636	0.133518	10

> **Below table shows the most popular genres and their characteristics over time by years.**

In [ ]:

```
popular_genres= DfMerged.sort_values("popularity", ascending=False).head(500)
```

```
popular_genres.groupby(by = ["Years Category","genres"]).mean().sort_values(by = "year",
ascending = False)
```

Out[ ]:

Years Category	genres	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness
2020-	post-teen pop	0.451000	0.635500	216396.500000	0.487000	0.50	0.000008	6.000000	0.109000
	pop	0.451000	0.635500	216396.500000	0.487000	0.50	0.000008	6.000000	0.109000
	canadian pop	0.181000	0.686000	190779.000000	0.538000	0.00	0.000003	2.000000	0.113000
2010s	german dance	0.223000	0.789000	166794.000000	0.720000	0.00	0.000000	8.000000	0.129000
	reggaeton colombiano	0.295000	0.803000	200960.000000	0.715000	0.00	0.000134	2.000000	0.057400
	reggaeton	0.295000	0.803000	200960.000000	0.715000	0.00	0.000134	2.000000	0.057400
	queens hip hop	0.295000	0.803000	200960.000000	0.715000	0.00	0.000134	2.000000	0.057400
	pop rock	0.837000	0.764000	189486.000000	0.320000	0.00	0.000000	11.000000	0.082200
	pop rap	0.295000	0.803000	200960.000000	0.715000	0.00	0.000134	2.000000	0.057400
	pop edm	0.223000	0.789000	166794.000000	0.720000	0.00	0.000000	8.000000	0.129000
	trap	0.104000	0.896000	196653.000000	0.586000	1.00	0.000000	10.000000	0.790000
	trap latino	0.295000	0.803000	200960.000000	0.715000	0.00	0.000134	2.000000	0.057400
	latin	0.295000	0.803000	200960.000000	0.715000	0.00	0.000134	2.000000	0.057400
	hip pop	0.295000	0.803000	200960.000000	0.715000	0.00	0.000134	2.000000	0.057400
	scandipop	0.223000	0.789000	166794.000000	0.720000	0.00	0.000000	8.000000	0.129000
	dfw rap	0.192000	0.695000	215280.000000	0.762000	0.00	0.002440	0.000000	0.086300
	tropical house	0.223000	0.789000	166794.000000	0.720000	0.00	0.000000	8.000000	0.129000
	edm	0.223000	0.789000	166794.000000	0.720000	0.00	0.000000	8.000000	0.129000
	bedroom pop	0.598000	0.744000	188387.000000	0.619000	0.00	0.003720	0.000000	0.231000
	pop dance	0.333667	0.769333	177887.000000	0.649000	0.00	0.000045	3.666667	0.093067
	melodic rap	0.209980	0.691600	211528.000000	0.627000	0.80	0.000555	3.600000	0.290980
	electropop	0.631000	0.526000	197137.000000	0.360500	0.00	0.065000	5.500000	0.097500
	australian pop	0.587500	0.770000	187672.500000	0.550000	0.00	0.000052	3.500000	0.120900
	chicago rap	0.251300	0.622333	215235.666667	0.595667	1.00	0.000112	2.666667	0.192867
	rap	0.108825	0.757500	207872.750000	0.625750	0.75	0.000611	6.000000	0.308825
	post-teen pop	0.256433	0.594500	205871.000000	0.671333	0.00	0.000046	4.166667	0.168067
	australian dance	0.483000	0.716000	165907.000000	0.512000	0.00	0.000000	1.000000	0.092800
	pop	0.457244	0.580375	205379.875000	0.542688	0.00	0.018567	5.312500	0.126500
	dance pop	0.262475	0.708250	184438.500000	0.642000	0.00	0.000034	4.500000	0.143300
	uk pop	0.726000	0.406000	195494.000000	0.445000	0.00	0.000000	3.500000	0.088800
	miami hip hop	0.469000	0.872000	119133.000000	0.391000	1.00	0.000004	0.000000	0.297000
	emo rap	0.469000	0.872000	119133.000000	0.391000	1.00	0.000004	0.000000	0.297000
	talent show	0.701000	0.311000	208827.000000	0.485000	0.00	0.000000	6.000000	0.072600
	modern alternative	0.058650	0.600000	250286.500000	0.664000	0.00	0.083350	10.000000	0.112000

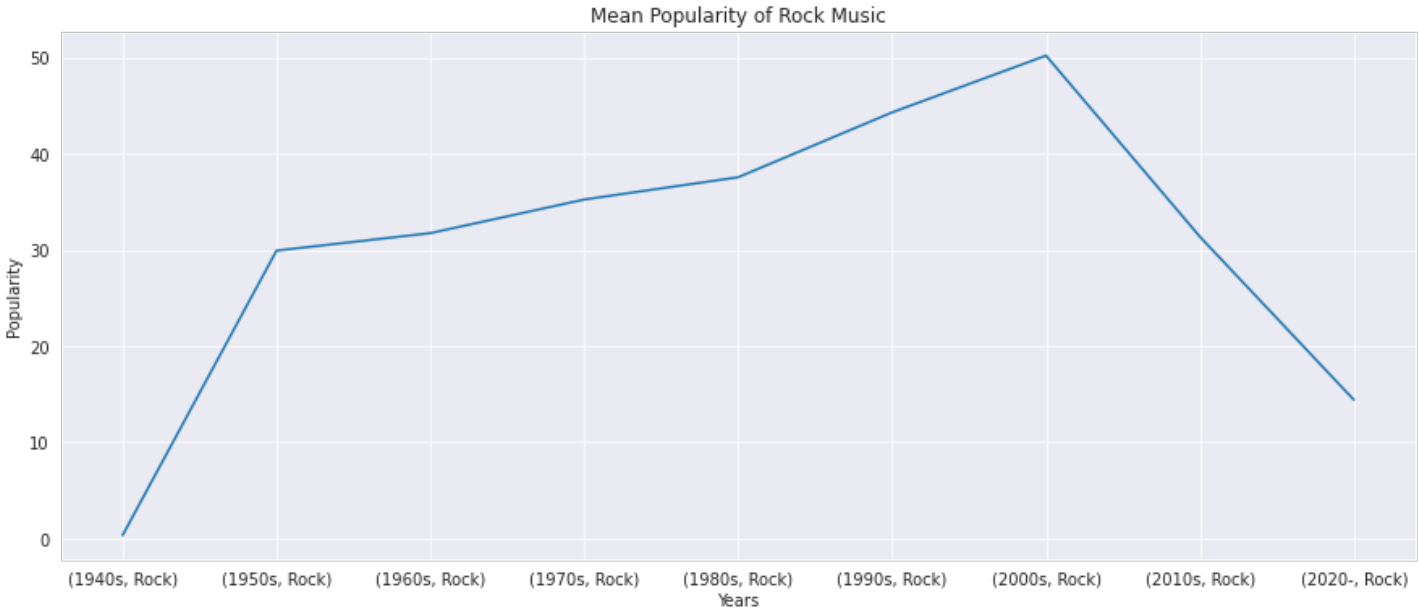
Years Category	rock	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness
	modern rock	0.058650	0.600000	250286.500000	0.664000	0.00	0.083350	10.000000	0.112000
1950s	genres urban contemporary	0.164000	0.336000	241107.000000	0.627000	0.00	0.000000	7.000000	0.070800
	pop	0.164000	0.336000	241107.000000	0.627000	0.00	0.000000	7.000000	0.070800
	dance pop	0.164000	0.336000	241107.000000	0.627000	0.00	0.000000	7.000000	0.070800
1960s	brill building pop	0.614000	0.589000	126267.000000	0.472000	0.00	0.000000	8.000000	0.505000
	vocal jazz	0.614000	0.589000	126267.000000	0.472000	0.00	0.000000	8.000000	0.505000
	adult standards	0.614000	0.589000	126267.000000	0.472000	0.00	0.000000	8.000000	0.505000

In [ ]:

```
###
def rockgenre(genres):

  if (genres==("rock")):
    return "Rock"
DfMerged["Rock"] = DfMerged["genres"].apply(rockgenre)
```

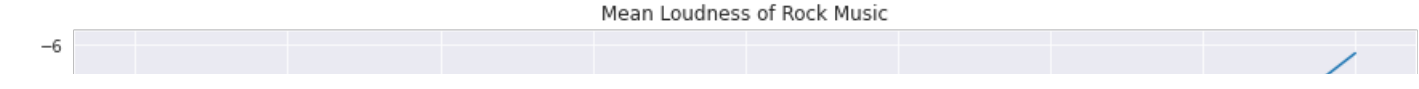
In [ ]:

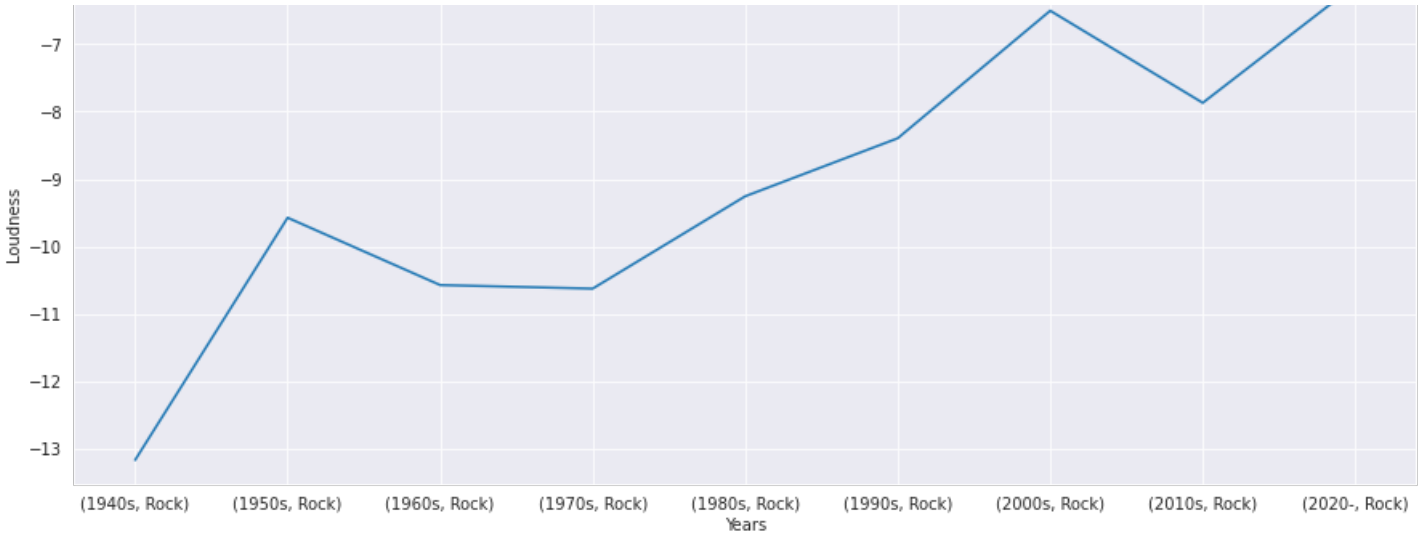
```
fig, ax = plt.subplots(1, 1, figsize=(15, 6))

DfMerged.groupby(by = ["Years Category","Rock"]).popularity.mean().plot(label="Popularit
y");#blue

plt.title("Mean Popularity of Rock Music")
plt.xlabel("Years");
plt.ylabel("Popularity");
```



In [ ]:

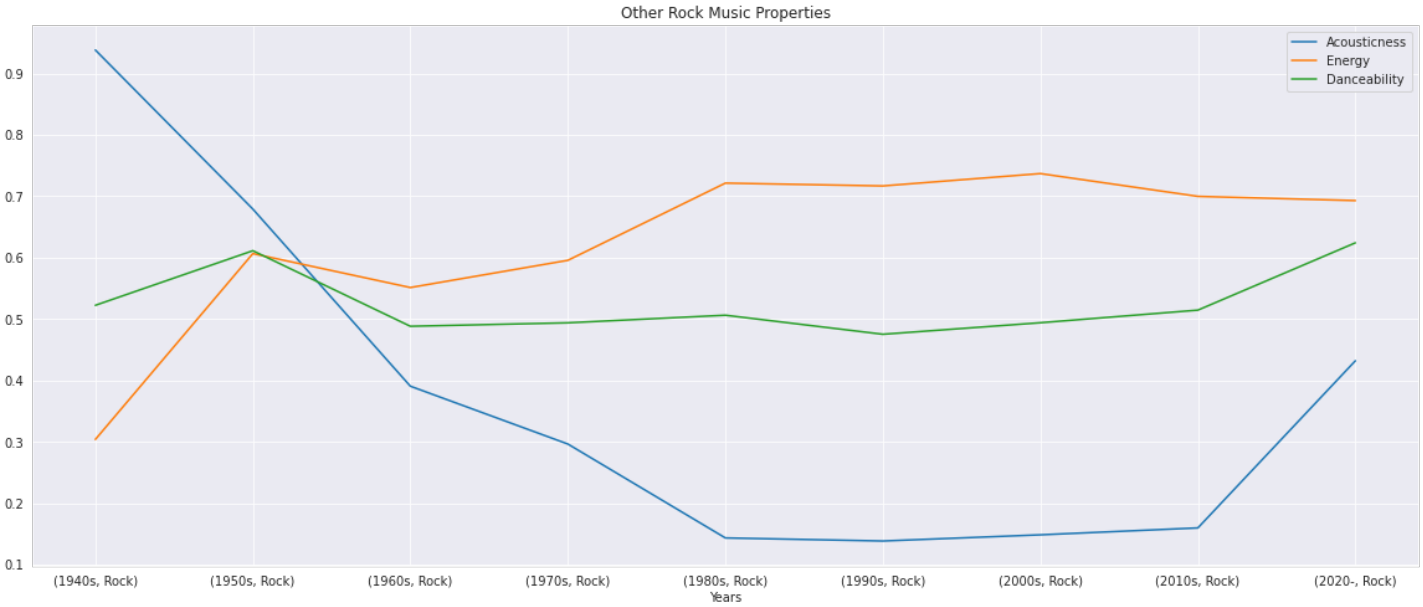
```
fig, ax = plt.subplots(1, 1, figsize=(15, 6))

DfMerged.groupby(by = ["Years Category","Rock"]).loudness.mean().plot(label="Loudness");
#red

plt.title("Mean Loudness of Rock Music")
plt.ylabel("Loudness")
plt.xlabel("Years");
```





In [ ]:

```
fig, ax = plt.subplots(1, 1, figsize=(20, 8))

DfMerged.groupby(by = ["Years Category","Rock"]).acousticness.mean().plot(label="Acousticness");#yellow
DfMerged.groupby(by = ["Years Category","Rock"]).energy.mean().plot(label="Energy") ;#purple
DfMerged.groupby(by = ["Years Category","Rock"]).danceability.mean().plot(label="Danceability"); #green

plt.legend()
plt.title("Other Rock Music Properties")
plt.xlabel("Years");
```



> **These figures show some Rock Music Features changes over time.**

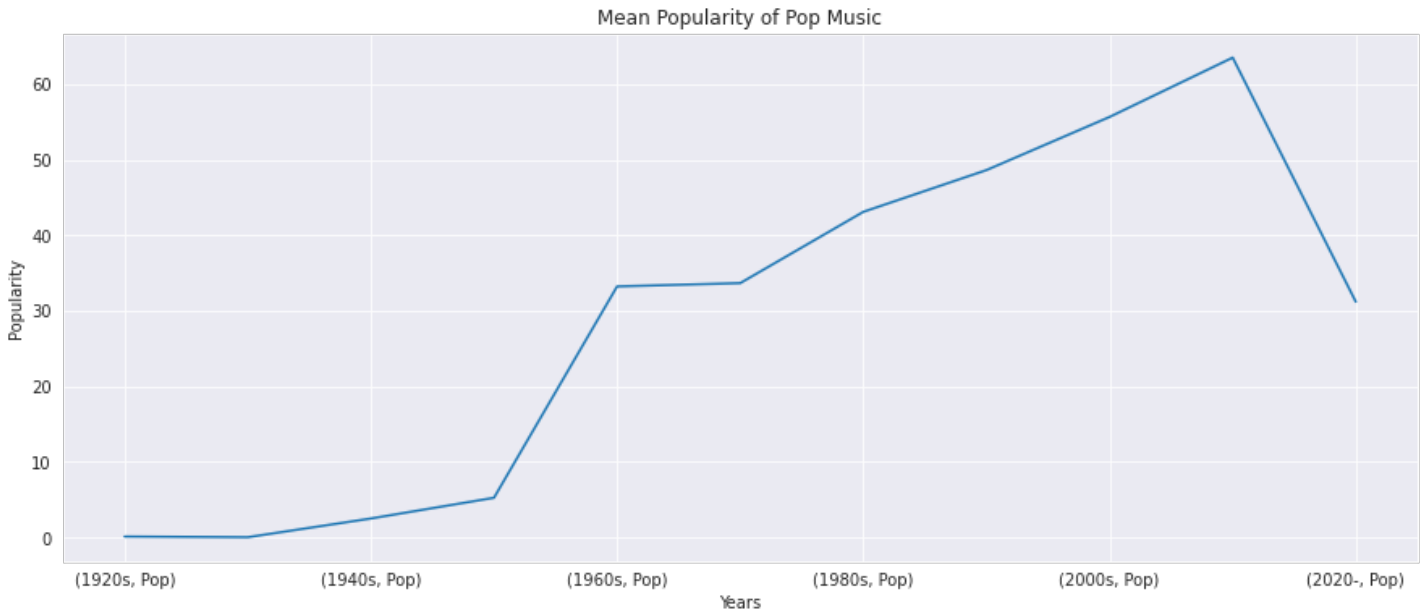
- **Popularity decreased between 2000s - 2020s other than that its popularity have been increasing. *It's Loudness is positively correlated with the Years, As the Years went on Loudness is relativly increased**

In [ ]:

```
###
def Popgenre(genres):

  if (genres==("pop")):
    return "Pop"
DfMerged["Pop"] = DfMerged["genres"].apply(Popgenre)
```

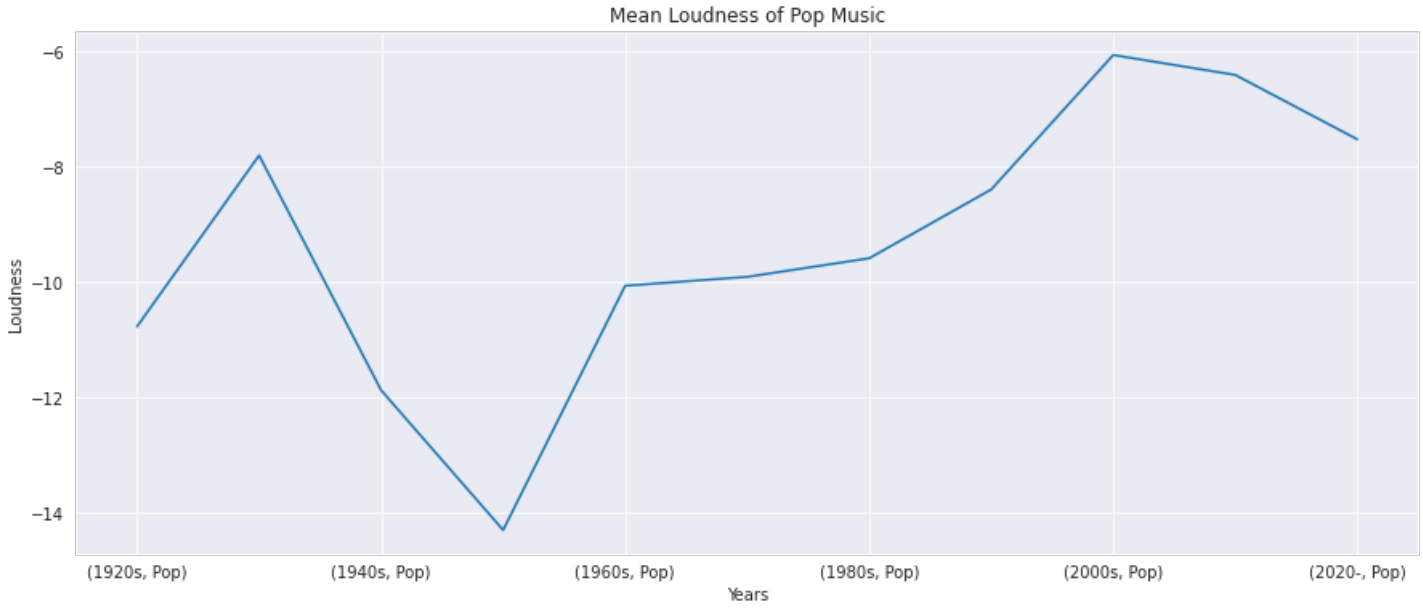
In [ ]:

```
fig, ax = plt.subplots(1, 1, figsize=(15, 6))
DfMerged.groupby(by = ["Years Category","Pop"]).popularity.mean().plot(label="Popularity
") #blue
plt.title("Mean Popularity of Pop Music")

plt.ylabel("Popularity");
plt.xlabel("Years");
```



In [ ]:

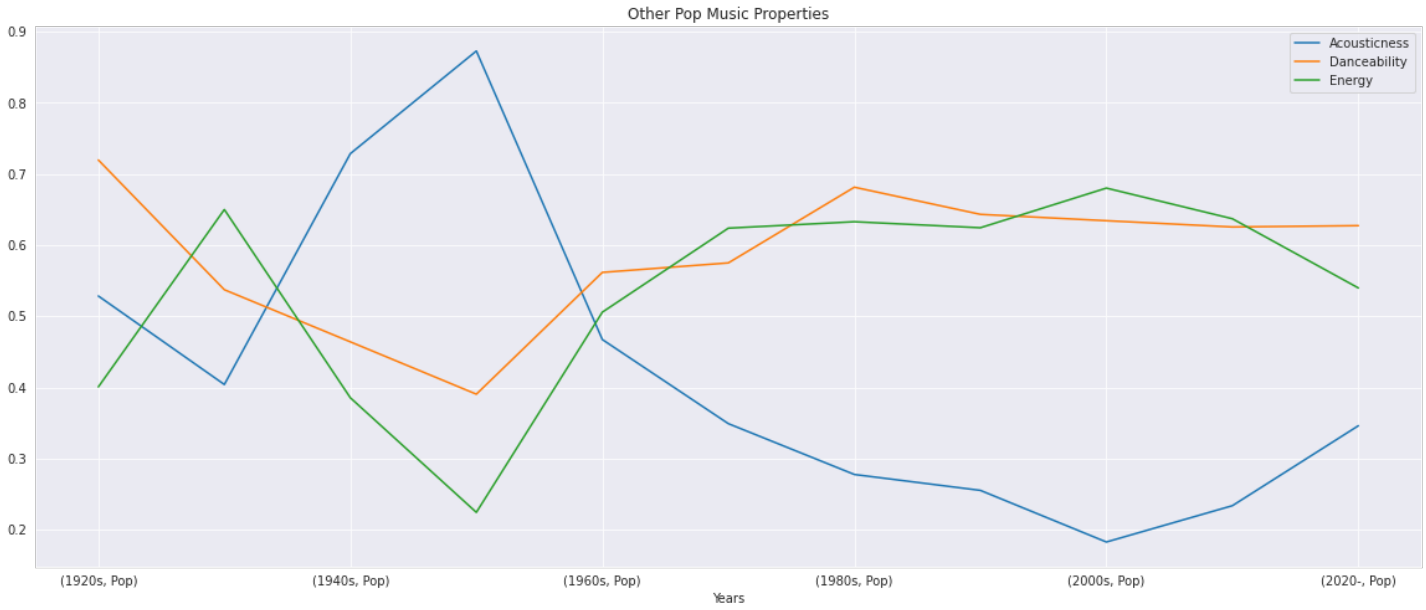
```
fig, ax = plt.subplots(1, 1, figsize=(15, 6))
DfMerged.groupby(by = ["Years Category","Pop"]).loudness.mean().plot(label="Loudness") #
red

plt.title("Mean Loudness of Pop Music")
plt.ylabel("Loudness")
plt.xlabel("Years");
```



In [ ]:

```
fig, ax = plt.subplots(1, 1, figsize=(20, 8))
DfMerged.groupby(by = ["Years Category","Pop"]).acousticness.mean().plot(label="Acoustic
ness")#yellow
DfMerged.groupby(by = ["Years Category","Pop"]).danceability.mean().plot(label="Danceabi
lity") #green
DfMerged.groupby(by = ["Years Category","Pop"]).energy.mean().plot(label="Energy") #purp
le
```

```
plt.legend()
plt.title("Other Pop Music Properties")
plt.xlabel("Years");
```



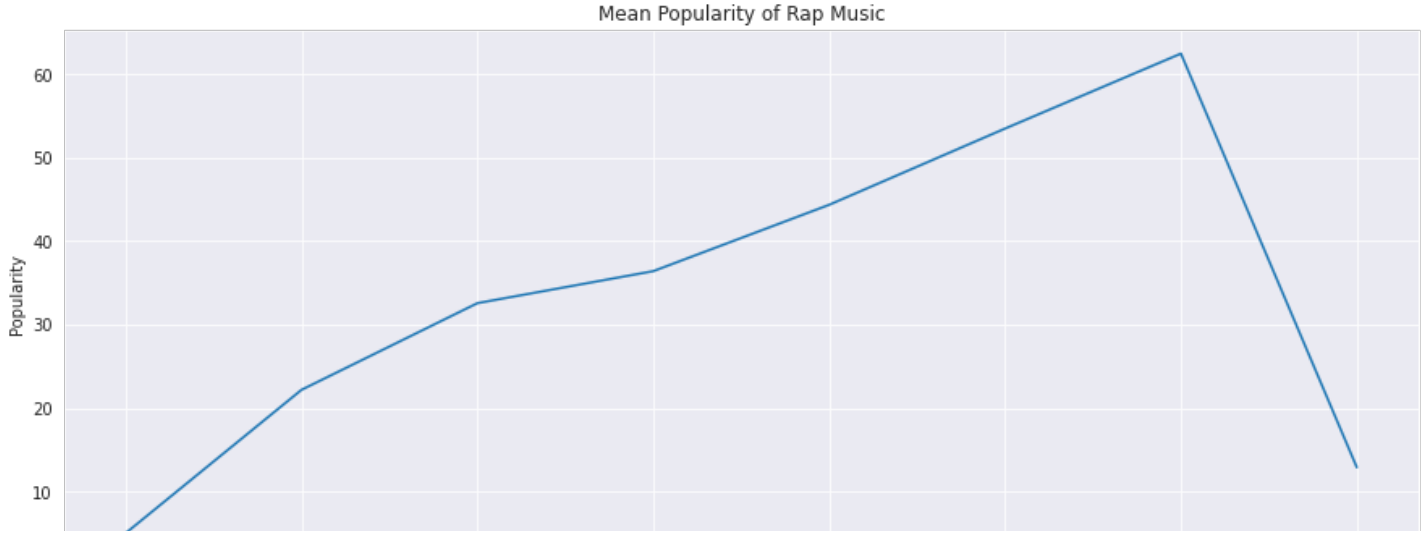
> **These figures show some Pop Music Features changes over time.**
>
> - **It's popularity kept on increasing over the passing years untill 2010s.**
> - **Loudness kept on increasing except from 30's to 50's and 2000's to 2020's *It's acousticness changes rapidly while the other stats are fairly stable.**

In [ ]:

```
###
def rapgenre(genres):

  if (genres==("rap")):
    return "Rap"
DfMerged["Rap"] = DfMerged["genres"].apply(rapgenre)
```

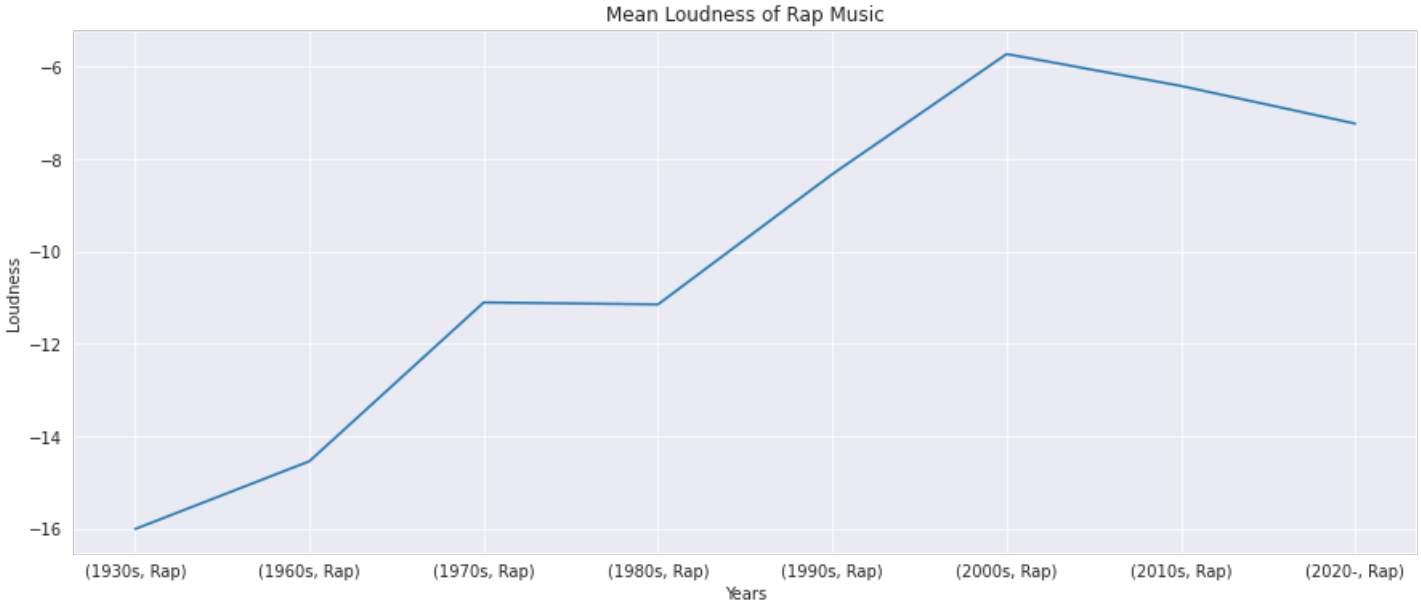
In [ ]:

```
fig, ax = plt.subplots(1, 1, figsize=(15, 6))
DfMerged.groupby(by = ["Years Category","Rap"]).popularity.mean().plot(label="Popularity
")#blue

plt.title("Mean Popularity of Rap Music")
plt.ylabel("Popularity")
plt.xlabel("Years");
```



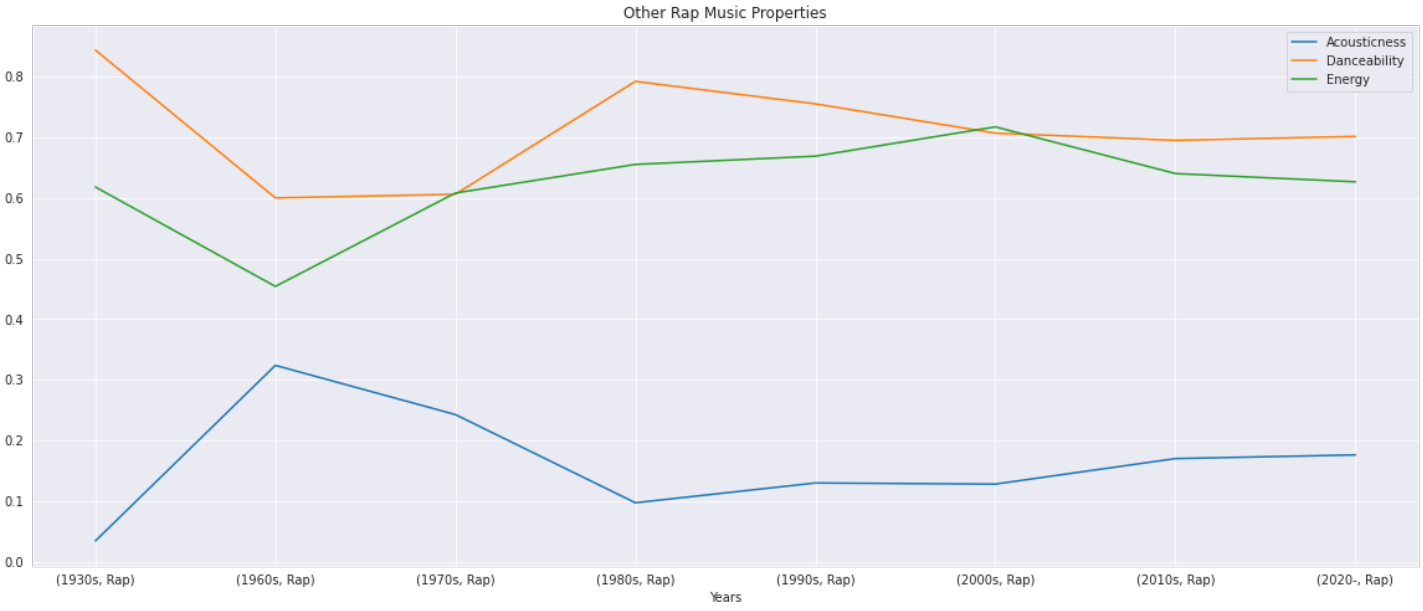
In [ ]:

```
fig, ax = plt.subplots(1, 1, figsize=(15, 6))
DfMerged.groupby(by = ["Years Category","Rap"]).loudness.mean().plot(label="Loudness") #
red

plt.title("Mean Loudness of Rap Music")
plt.ylabel("Loudness")
plt.xlabel("Years");
```



In [ ]:

```
fig, ax = plt.subplots(1, 1, figsize=(20, 8))
DfMerged.groupby(by = ["Years Category","Rap"]).acousticness.mean().plot(label="Acoustic
ness")#yellow
DfMerged.groupby(by = ["Years Category","Rap"]).danceability.mean().plot(label="Danceabi
lity") #green
DfMerged.groupby(by = ["Years Category","Rap"]).energy.mean().plot(label="Energy") #purp
le

plt.legend()
plt.title("Other Rap Music Properties")
plt.xlabel("Years");
```



These figures show some Rap Music Features changes over time

These figures show some Rap Music Features changes over time.

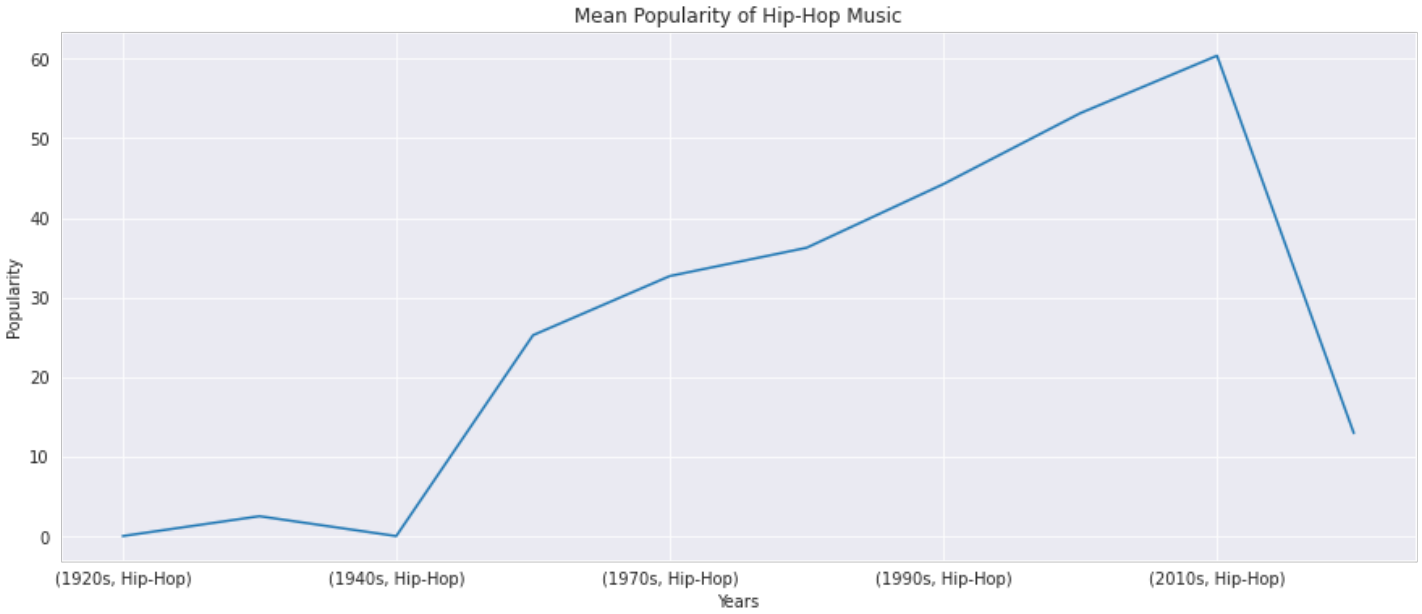
- **It's popularity was stabily rasing Until 2020's. After 2010's there is a sharp decrase *It's Loudness had been on an increase until 2000's after that part there is a slight decrease.*It's Energy has been slightly increasing over the years. The other stats are fairly stable.**

In [ ]:

```
###
def hiphopgenre(genres):

  if (genres==("hip hop")):
    return "Hip-Hop"
DfMerged["Hip-Hop"] = DfMerged["genres"].apply(hiphopgenre)
```

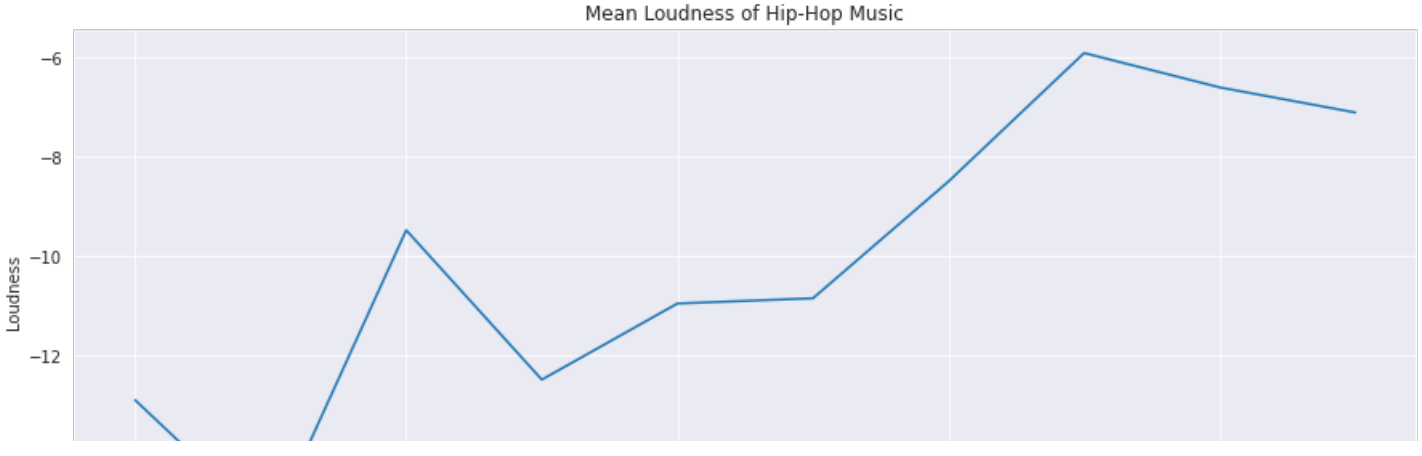
In [ ]:

```
fig, ax = plt.subplots(1, 1, figsize=(15, 6))
DfMerged.groupby(by = ["Years Category","Hip-Hop"]).popularity.mean().plot(label="Popula
rity")#blue

plt.title("Mean Popularity of Hip-Hop Music")
plt.ylabel("Popularity");
plt.xlabel("Years");
```



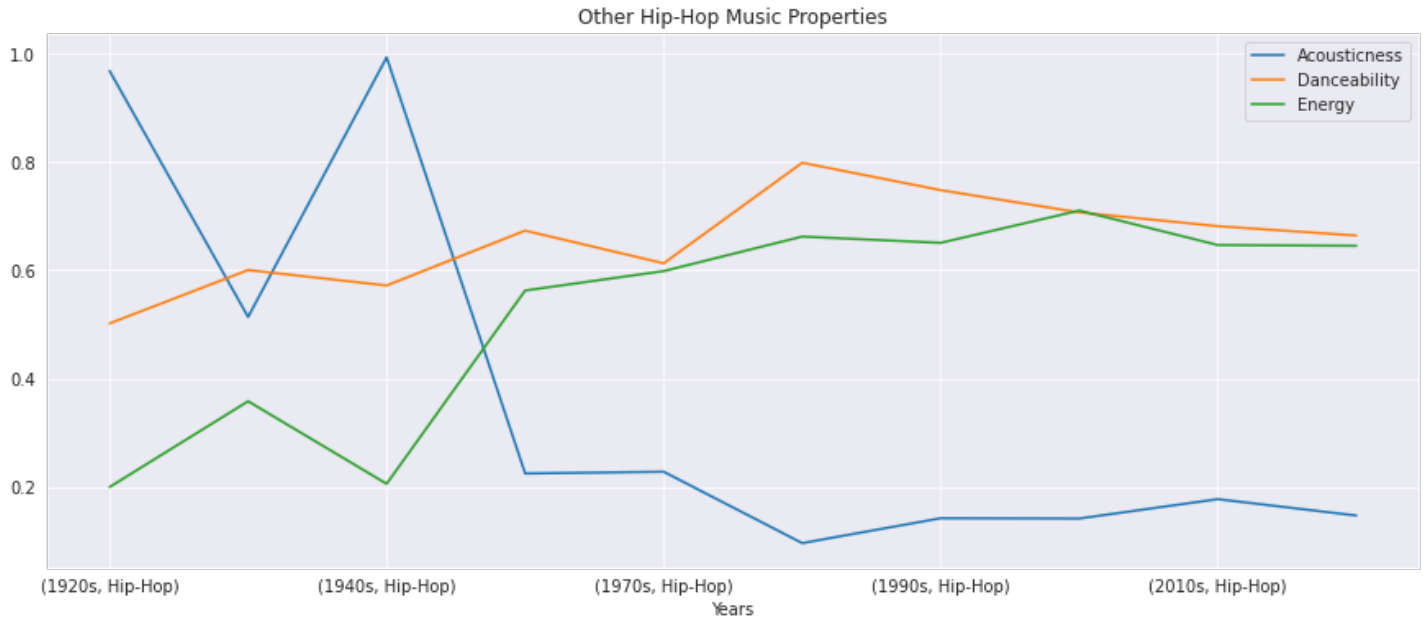
In [ ]:

```
fig, ax = plt.subplots(1, 1, figsize=(15, 6))
DfMerged.groupby(by = ["Years Category","Hip-Hop"]).loudness.mean().plot(label="Loudness
") #red

plt.title("Mean Loudness of Hip-Hop Music")
plt.ylabel("Loudness");
plt.xlabel("Years");
```



In [ ]:

```
fig, ax = plt.subplots(1, 1, figsize=(20, 8))

DfMerged.groupby(by = ["Years Category","Hip-Hop"]).acousticness.mean().plot(label="Acou
sticness")#yellow
DfMerged.groupby(by = ["Years Category","Hip-Hop"]).danceability.mean().plot(label="Danc
eability") #green
DfMerged.groupby(by = ["Years Category","Hip-Hop"]).energy.mean().plot(label="Energy") #
purple

plt.legend()
plt.title("Other Hip-Hop Music Properties")
plt.xlabel("Years");
```



> **These figures show some of Hip-Hop Features changes over time.**

- **Popularity has a positive slope most of the time , peaks at 2010s. Overall has a mid level of popularity most of the time.**
- **Acoustiveness also flactuates a lot and seems to be negatively correlated with ppularity**
- **Danceability and energy are positively correlated slightly and both has 2 troughs at 10s and 50s .**
- **Loudness peaks at 2010s and seems to be positively realted with popularity and energy.**

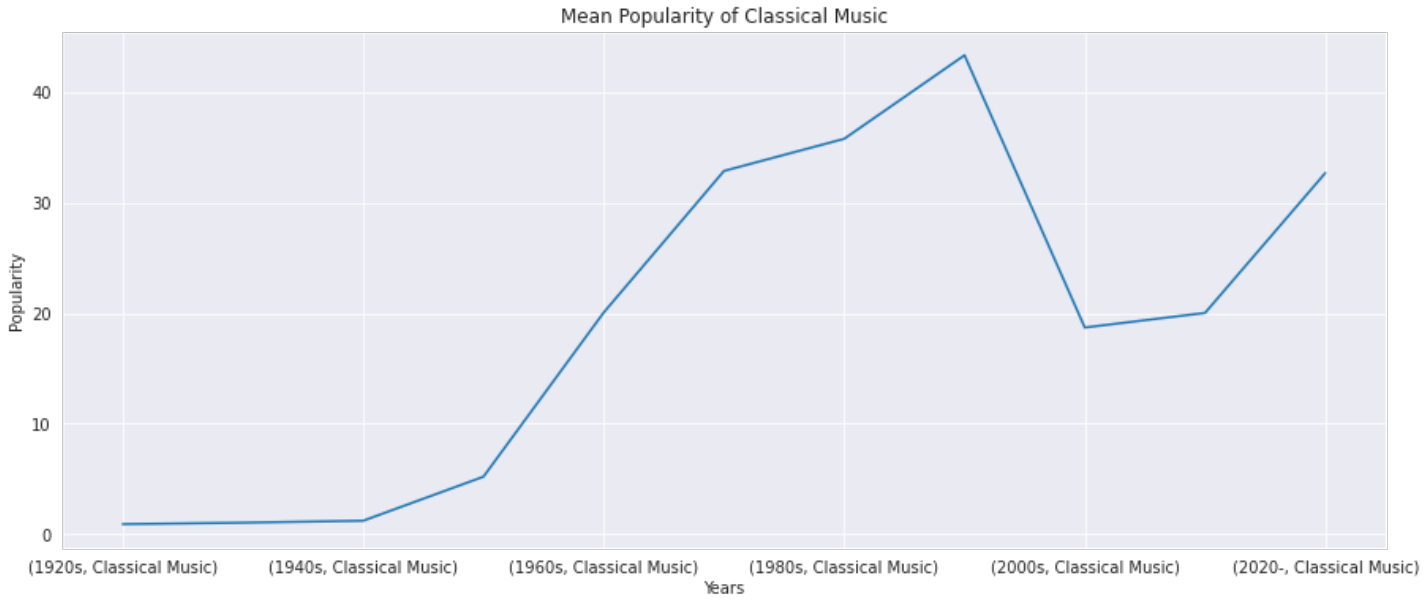
In [ ]:

```
###
def classicalgenre(genres):

  if (genres==("classical")):
    return "Classical Music"
DfMerged["Classical Music"] = DfMerged["genres"].apply(classicalgenre)
```

In [ ]:

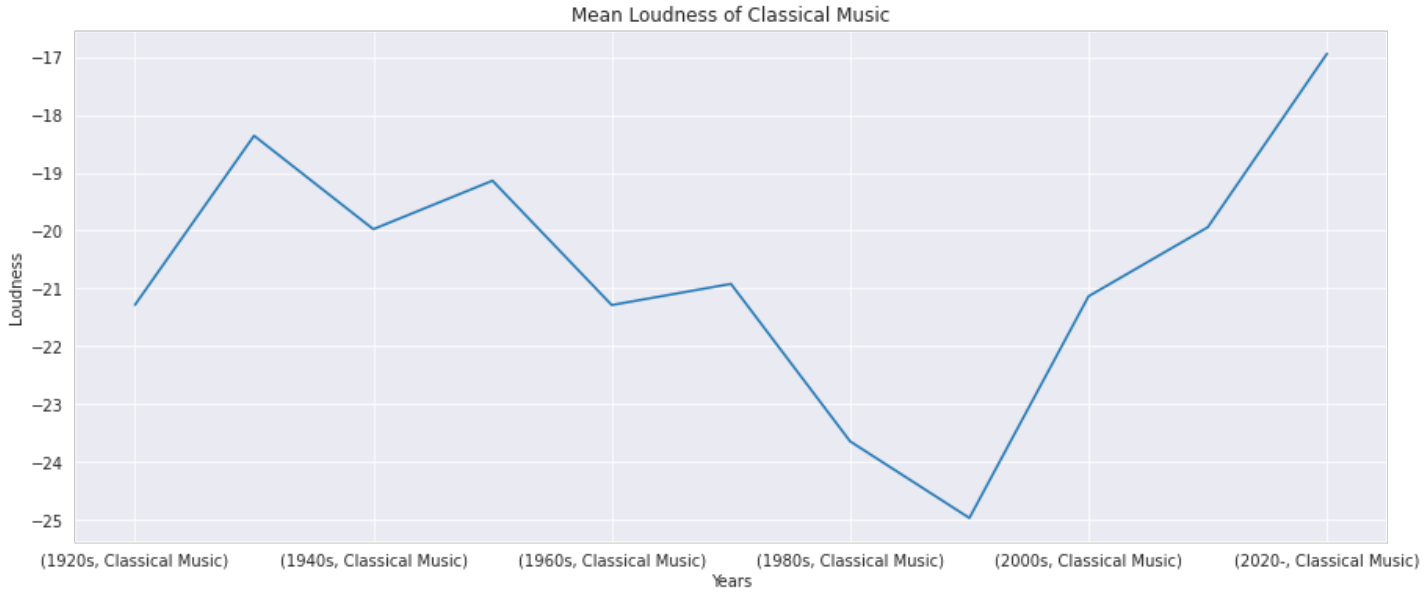
```
fig, ax = plt.subplots(1, 1, figsize=(15, 6))

DfMerged.groupby(by = ["Years Category","Classical Music"]).popularity.mean().plot(label
="Popularity")#blue
```

```
plt.title("Mean Popularity of Classical Music")
plt.ylabel("Popularity");
plt.xlabel("Years");
```



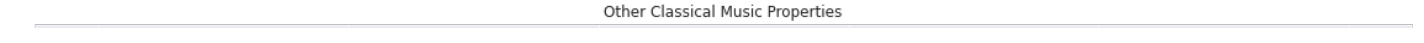
In [ ]:

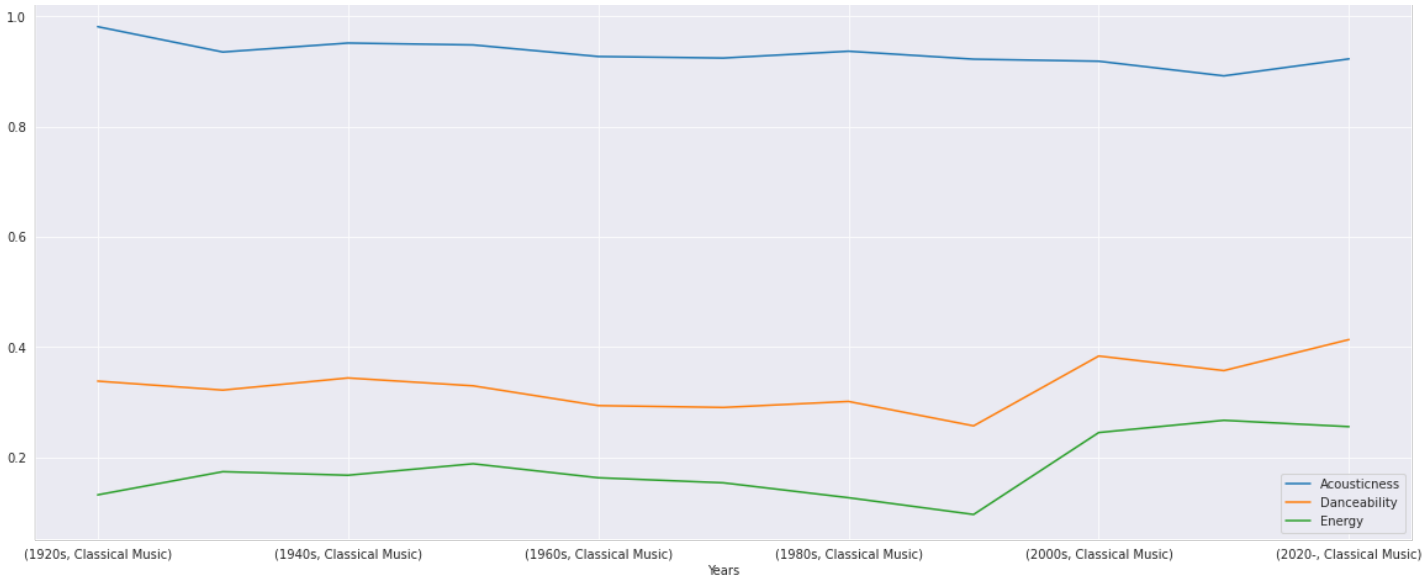
```
fig, ax = plt.subplots(1, 1, figsize=(15, 6))

DfMerged.groupby(by = ["Years Category","Classical Music"]).loudness.mean().plot(label="
Loudness") #red

plt.title("Mean Loudness of Classical Music")
plt.ylabel("Loudness");
plt.xlabel("Years");
```



In [ ]:

```
fig, ax = plt.subplots(1, 1, figsize=(20, 8))

DfMerged.groupby(by = ["Years Category","Classical Music"]).acousticness.mean().plot(lab
el="Acousticness")#yellow
DfMerged.groupby(by = ["Years Category","Classical Music"]).danceability.mean().plot(lab
el="Danceability") #green
DfMerged.groupby(by = ["Years Category","Classical Music"]).energy.mean().plot(label="En
ergy") #purple

plt.legend()
plt.title("Other Classical Music Properties")
plt.xlabel("Years");
```





> **This figure shows some of Classical Music Features changes over time.**

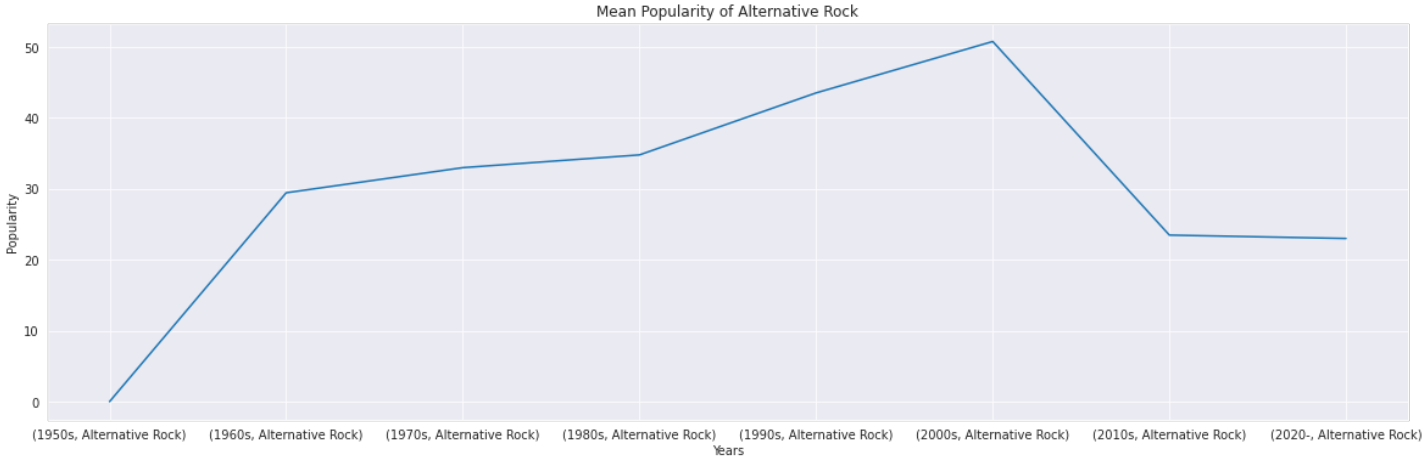
- **Popularity hits peak at 90. Has a lot of flactuations ,through and peaks. There is a big increase popularity of Classical Music between 50s and 90s.But mostly not very popular.**
- **Acoustiveness is relatively stable.**
- **Danceability and energy are highly positively correlated and both has through at 1930s so in this period Danceability and energy of classical music is lower than other years .**
- **Loudness peaks at 2020s and 90s.**

In [ ]:

```
###
def altrockgenre(genres):

  if (genres==("alternative rock")):
    return "Alternative Rock"
DfMerged["Alternative Rock"] = DfMerged["genres"].apply(altrockgenre)
```

In [ ]:

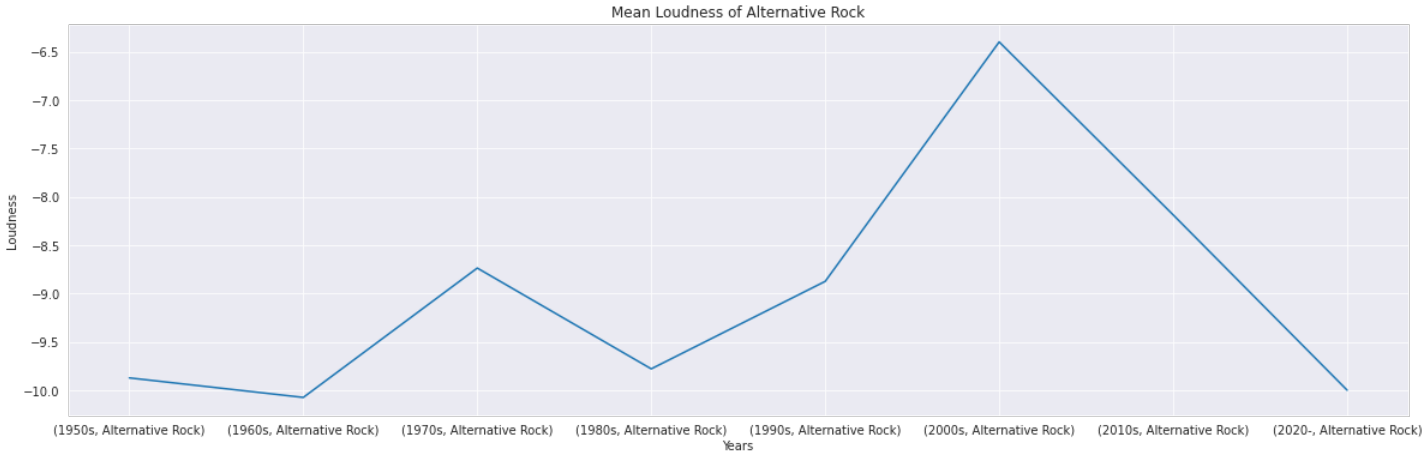
```
fig, ax = plt.subplots(1, 1, figsize=(20, 6))

DfMerged.groupby(by = ["Years Category","Alternative Rock"]).popularity.mean().plot( lab
el="Popularity")#blue

plt.title("Mean Popularity of Alternative Rock")
plt.xlabel("Years");
plt.ylabel("Popularity");
```



In [ ]:

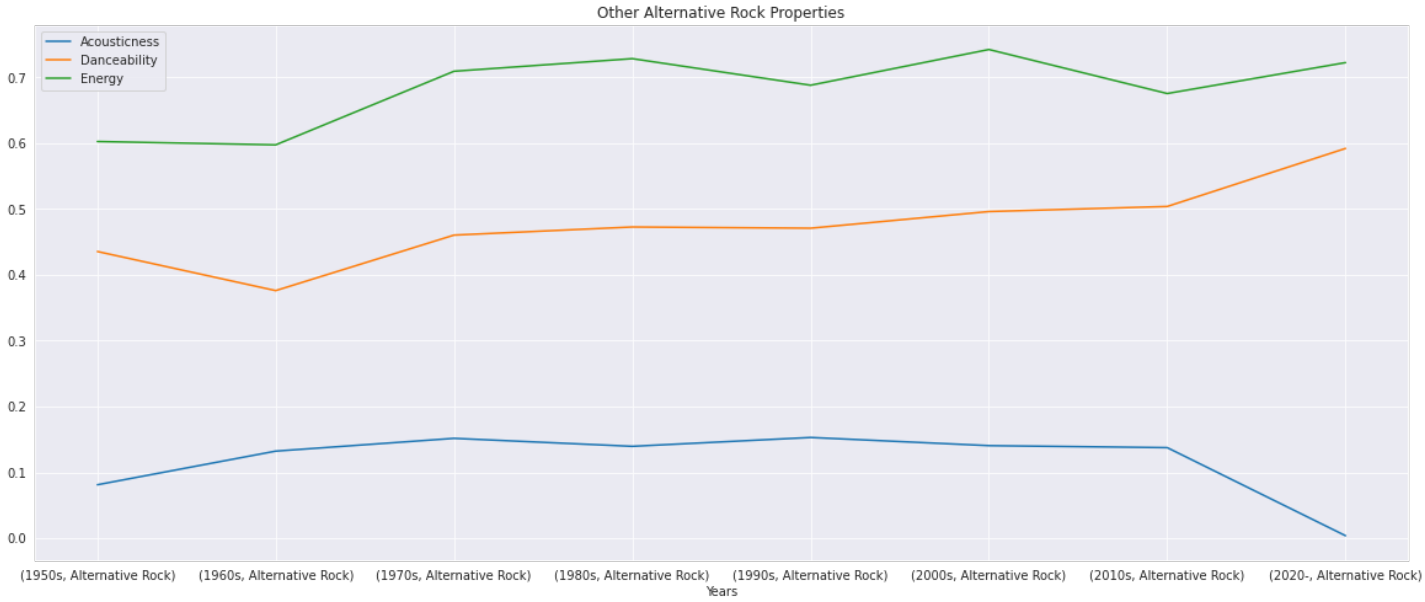
```
fig, ax = plt.subplots(1, 1, figsize=(20, 6))
```

```
fig, ax = plt.subplots(1, 1, figsize=(20, 8))

DfMerged.groupby(by = ["Years Category","Alternative Rock"]).loudness.mean().plot(label=
"Loudness") #red

plt.title("Mean Loudness of Alternative Rock")
plt.xlabel("Years");
plt.ylabel("Loudness");
```



In [ ]:

```
fig, ax = plt.subplots(1, 1, figsize=(20, 8))

DfMerged.groupby(by = ["Years Category","Alternative Rock"]).acousticness.mean().plot(la
bel="Acousticness")#yellow
DfMerged.groupby(by = ["Years Category","Alternative Rock"]).danceability.mean().plot(la
bel="Danceability") #green
DfMerged.groupby(by = ["Years Category","Alternative Rock"]).energy.mean().plot(label="E
nergy") #purple

plt.legend()

plt.title("Other Alternative Rock Properties")
plt.xlabel("Years");
```



> **These figures show some Alternative Rock Music Features changes over time.**

- **Popularity hits bottom at 50s and hits peak at 2000s, popularity of alternative rock decreases after 2000s this maybe because there are more new genres.**
- **Acoustiveness is relatively stable.**
- **Danceability and energy are highly positively correlated and both peaks at 2020.**
- **Loudness peaks at 2020s and 60s with -10 and and seem to be slightly positively correlated with popularity.**

Hypothesis Testing

Hypothesis Test 1: We want to test whether there is significant differences in terms of popularity for different values of explicity (0 and 1).

Null Hypothesis (
H_0**): Means of** `popularity` **samples for all explicity values are same (e.g.** p_1 **denotes popularity of 1st sample).**

$H_0 : \mu_{p_1}$
$= \mu_{p_2}$

Alternative Hypothesis (
H_A**): Means of** `popularity` **samples for all explicity values are different.**

H_A **: Means** μ_{p_1}, μ_{p_2} **are not same.**

Significance level: As most of hypothesis tests assume significance level as `0.05`**, we are setting it as** `0.05` **for our test too.**

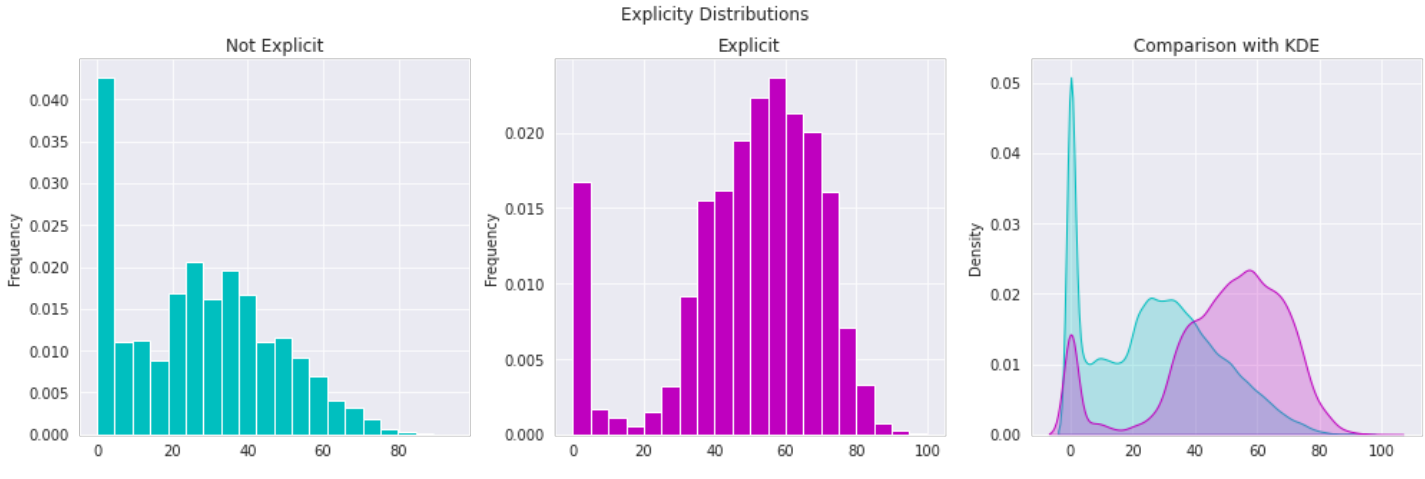
In [ ]:

```
# Creating samples based on target attribute
sample_1, sample_2 = [DfMerged[DfMerged["explicit"] == i] for i in range(0,2)]
samplettest1=DfMerged[DfMerged["explicit"] == 0]["popularity"]
samplettest2=DfMerged[DfMerged["explicit"] == 1]["popularity"]
```

Before applying an appropriate test, we can also visualize these samples to observe the possible statistical difference.

In [ ]:

```
fig, ax = plt.subplots(1, 3, figsize=(14,5))

sample_1["popularity"].plot(kind="hist", ax=ax[0], bins=20, label="completed", color="c"
, density=True)
ax[0].set_title("Not Explicit")

sample_2["popularity"].plot(kind="hist", ax=ax[1], bins=20, label="none", color="m", den
sity=True)
ax[1].set_title("Explicit")

sns.kdeplot(sample_1["popularity"], shade=True, label="Not Explicit", ax=ax[2], color="c
")
sns.kdeplot(sample_2["popularity"], shade=True, label="Explicit", ax=ax[2], color="m")
ax[2].set_title("Comparison with KDE")

plt.suptitle("Explicity Distributions")
# To avoid suptitle and titles of ax titles colliding
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



Since we want to test 2 samples simultaneously, Two-Sample T-test is a test that we can apply here. From here we calculate the p value with t-test.

In [ ]:

```
stats.ttest_ind(samplettest1, samplettest2, equal_var=False)
```

Out[ ]:

```
Ttest_indResult(statistic=-249.36246600880153, pvalue=0.0)
```

Interpreting results:

> **~0.0 < 0.05**
>
> **As p-value we obtained was too small for Python to calculate it is smaller than the threshold significance level 0.05, we can conclude that means of popularity samples are not the same. Here, we reject the null hypothesis.**

Hypothesis Test 2: We want to test whether there is significant differences in terms of popularity for different keys (samples are taken from key 5 and key 6).

Null Hypothesis (
H_0**): Means of** `popularity` **samples for all keys are same (e.g.** p_1 **denotes popularity of 1st sample).**

$H_0 : \mu_{p_1}$
$= \mu_{p_2}$

Alternative Hypothesis (
H_A**): Means of** `popularity` **samples for all keys are different.**

H_A **: Means** μ_{p_1}, μ_{p_2} **are not same.**

Significance level: As most of hypothesis tests assume significance level as `0.05`**, we are setting it as** `0.05` **for our test too.**

In [ ]:

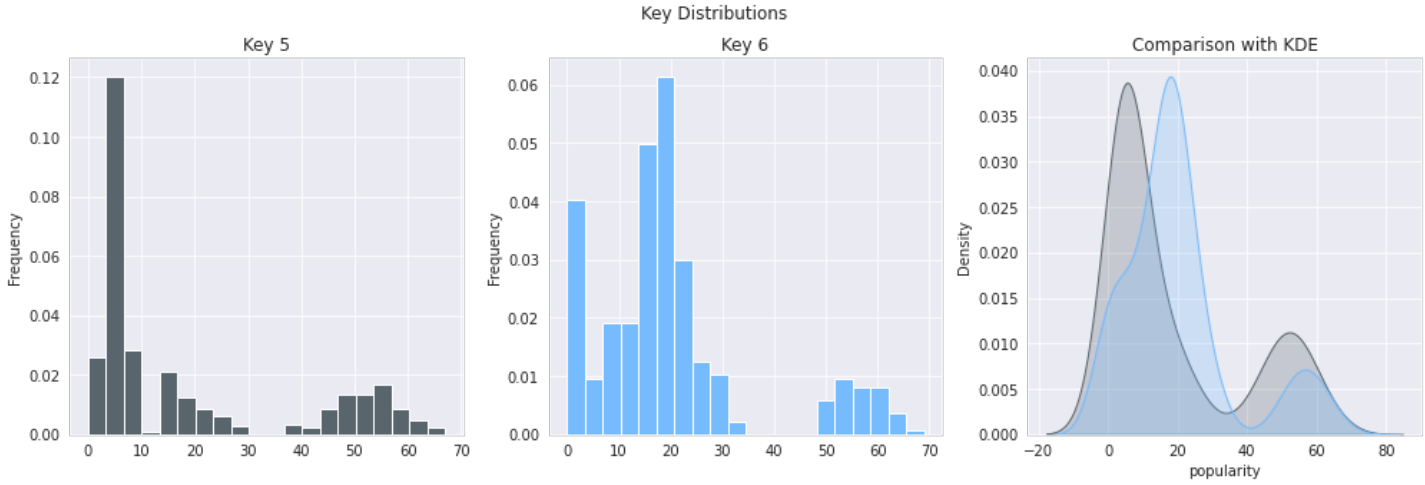
```
sample0, sample1 = [DfMerged[DfMerged["key"] == i] for i in range(5,7)]
samplettest1=DfMerged[DfMerged["key"] == 5]["popularity"]
samplettest2=DfMerged[DfMerged["key"] == 6]["popularity"]
```

Before applying an appropriate test, we can also visualize these samples to observe the possible statistical difference.

In [ ]:

```
fig, ax = plt.subplots(1, 3, figsize=(14,5))

sample0["popularity"].plot(kind="hist", ax=ax[0], bins=20, label="completed", color="xkcd:slate grey", density=True)
ax[0].set_title("Key 5")

sample1["popularity"].plot(kind="hist", ax=ax[1], bins=20, label="none", color="xkcd:sky blue", density=True)
ax[1].set_title("Key 6")



sns.kdeplot(sample0["popularity"], shade=True, label="Key 5", ax=ax[2], color="xkcd:slate grey")
sns.kdeplot(sample1["popularity"], shade=True, label="Key 6", ax=ax[2], color="xkcd:sky blue")
```

```
ax[2].set_title("Comparison with KDE")

plt.suptitle("Key Distributions")
# To avoid suptitle and titles of ax titles colliding
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



Since we want to test 2 samples simultaneously, Two-Sample T-test is a test that we can apply here. From here we calculate the p value with t-test.

In [ ]:

```
stats.ttest_ind(samplettest1, samplettest2, equal_var=False)
```

Out[ ]:

```
Ttest_indResult(statistic=-37.151234973375125, pvalue=2.32488893902949e-299)
```

Interpreting results:

> **2.32488893902949e-299 < 0.05**
>
> **As p-value we obtained is smaller than the threshold significance level 0.05, we can conclude that means of popularity samples are not the same nor close to each other. Hence, we reject the null hypothesis.**

Hypothesis Test 3: We want to test whether there is significant differences in terms of loudness for 70s and 80s Rap Music (1st sample denotes 70s Rap Music, 2nd sample denotes 80s Rap Music).

Null Hypothesis (
H_0**): Means of** `loudness` **samples for 70s and 80s Rap Music are same (e.g.** l_1 **denotes loudness of 1st sample).**

$H_0 : \mu_{l_1}$
$= \mu_{l_2}$

Alternative Hypothesis (
H_A**): Means of** `loudness` **samples for 70s and 80s Rap Music are different.**

H_A **: Means** μ_{l_1}, μ_{l_2} **are not same.**

Significance level: As most of hypothesis tests assume significance level as `0.05`**, we are setting it as** `0.05` **for our test too.**

In [ ]:

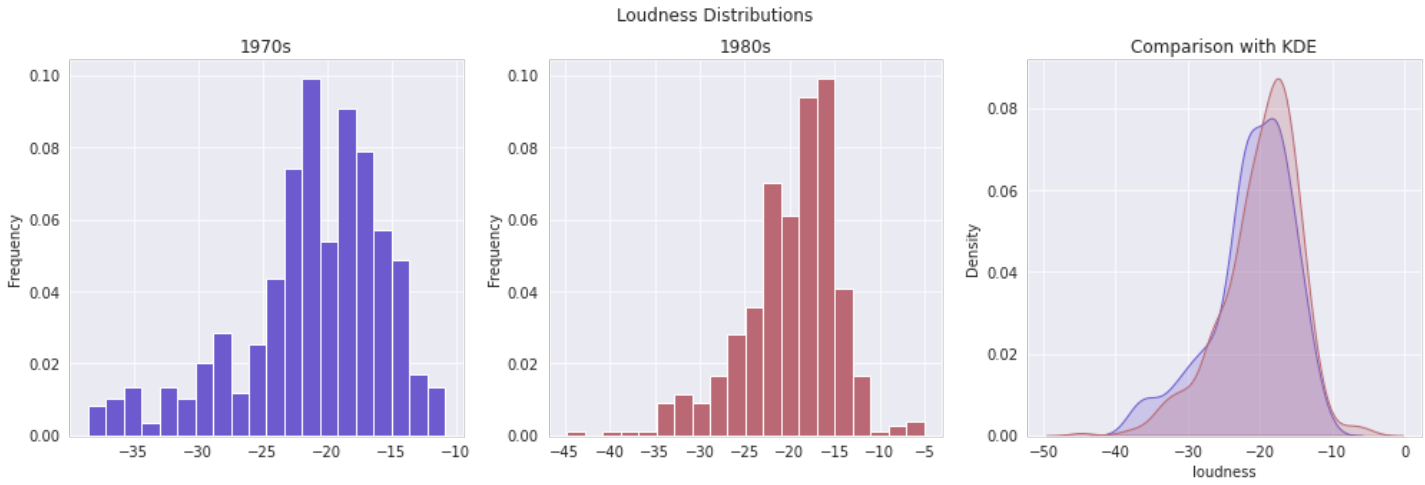
```
sample0 = DfMerged[DfMerged["Years Category"] == "1970s"]
sample0 = sample0[sample0["Rap"] == "Rap"]
```

```
sample1 = DfMerged[DfMerged["Years Category"] == "1980s"]
sample1 = sample1[sample1["Rap"] == "Rap"]
```

Before applying an appropriate test, we can also visualize these samples to observe the possible statistical difference.

In [ ]:

```
fig, ax = plt.subplots(1, 3, figsize=(14,5))

sample0["loudness"].plot(kind="hist", ax=ax[0], bins=20, label="completed", color="xkcd:light indigo", density=True)
ax[0].set_title("1970s")

sample1["loudness"].plot(kind="hist", ax=ax[1], bins=20, label="none", color="xkcd:dusky rose", density=True)
ax[1].set_title("1980s")



sns.kdeplot(sample0["loudness"], shade=True, label="1970s", ax=ax[2], color="xkcd:light indigo")
sns.kdeplot(sample1["loudness"], shade=True, label="1980s", ax=ax[2], color="xkcd:dusky rose")
ax[2].set_title("Comparison with KDE")

plt.suptitle("Loudness Distributions")
# To avoid suptitle and titles of ax titles colliding
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



Since we want to test 2 samples simultaneously, ANOVA is a test that we can apply here. Below, we introduce our samples to ANOVA test.

In [ ]:

```
from scipy.stats import f_oneway
f_stats, p_values = f_oneway(sample0["loudness"].values, sample1["loudness"].values)
p_values
```

Out[ ]:

```
0.9296732081671646
```

Interpreting results:

> **0.929 > 0.05**
>
> **As p-value we obtained is greater than the threshold significance level 0.05, we can conclude that means of loudness samples of 1970's and 1980's Rap are the same or very close to each other. Hence, we have failed to reject the null hypothesis.**

Hypothesis Test 4: We want to test whether there is significant differences in terms of popularity for 2000s and 2010s Classical Music (1st sample denotes 2000s Classical Music, 2nd sample denotes 2010s Classical Music).

Null Hypothesis (H_0): Means of `popularity` samples for 2000s and 2010s Classical Music are same (e.g. p_1 denotes popularity of 1st sample).

$H_0 : \mu_{p_1}$
$= \mu_{p_2}$

Alternative Hypothesis (H_A): Means of `popularity` samples for 2000s and 2010s Classical Music are different.

H_A **: Means** μ_{p_1}, μ_{p_2} **are not same.**

Significance level: As most of hypothesis tests assume significance level as `0.05`, we are setting it as `0.05` for our test too.

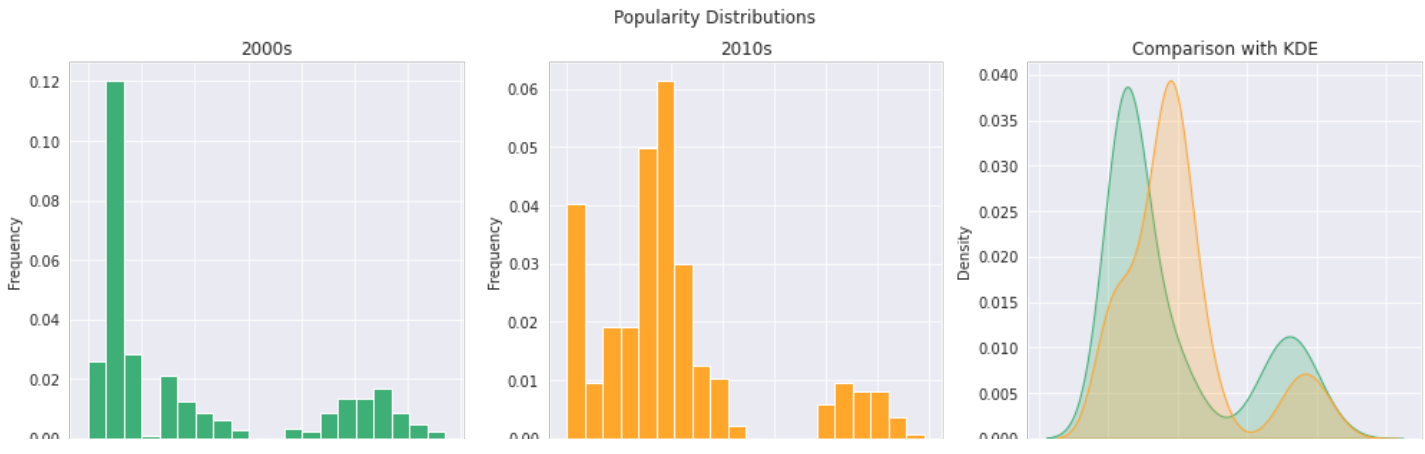
In [ ]:

```
sample0 = DfMerged[DfMerged["Years Category"] == "2000s"]
sample0 = sample0[sample0["Classical Music"] == "Classical Music"]
sample1 = DfMerged[DfMerged["Years Category"] == "2010s"]
sample1 = sample1[sample1["Classical Music"] == "Classical Music"]
```

Before applying an appropriate test, we can also visualize these samples to observe the possible statistical difference.

In [ ]:

```
fig, ax = plt.subplots(1, 3, figsize=(14,5))

sample0["popularity"].plot(kind="hist", ax=ax[0], bins=20, label="completed", color="xkcd:dark seafoam green", density=True)
ax[0].set_title("2000s")

sample1["popularity"].plot(kind="hist", ax=ax[1], bins=20, label="none", color="xkcd:mango", density=True)
ax[1].set_title("2010s")



sns.kdeplot(sample0["popularity"], shade=True, label="2000s", ax=ax[2], color="xkcd:dark seafoam green")
sns.kdeplot(sample1["popularity"], shade=True, label="2010s", ax=ax[2], color="xkcd:mango")
ax[2].set_title("Comparison with KDE")

plt.suptitle("Popularity Distributions")
# To avoid suptitle and titles of ax titles colliding
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



Since we want to test 2 samples simultaneously, ANOVA is a test that we can apply here. Below, we introduce our samples to ANOVA test.

In [ ]:

```
from scipy.stats import f_oneway
f_stats, p_values = f_oneway(sample0["popularity"].values, sample1["popularity"].values)
p_values
```

Out[ ]:

```
0.29711595762485876
```

Interpreting results:

> **0.29711595762485876 > 0.05**
>
> **As p-value we obtained is greater than the threshold significance level 0.05, we can conclude that means of popularity samples are the same or close to each other. Hence, we failed to reject the null hypothesis.**

Future Work

Our project has one remaining part, namely Machine Learning. In the light of the information we got from the previous parts, we will create a simple song recommendation system using similarity metrics and Nearest Neighbors methods. The song recommendation system will recommend songs similar to the ones given in the input. Additively, we will predict the popularity of songs with some machine learning models and show efforts on hyper tuning for the purpose of increasing the performance of those models.
****