```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
import plotly.graph_objects as go
import numpy as np
import plotly.express as px   #importing plotly
```

▶ Pre-processing the data

```
data = pd.read_csv("/content/drive/MyDrive/spotify/tracks.csv")
data_artist = pd.read_csv("/content/drive/MyDrive/spotify/artists.csv")
data
data_artist
```

id followers genres name popularity

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 586672 entries, 0 to 586671
Data columns (total 20 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   id                586672 non-null  object 
 1   name              586601 non-null  object 
 2   popularity        586672 non-null  int64  
 3   duration_ms       586672 non-null  int64  
 4   explicit          586672 non-null  int64  
 5   artists           586672 non-null  object 
 6   id_artists        586672 non-null  object 
 7   release_date      586672 non-null  object 
 8   danceability      586672 non-null  float64
 9   energy            586672 non-null  float64
 10  key               586672 non-null  int64  
 11  loudness          586672 non-null  float64
 12  mode              586672 non-null  int64  
 13  speechiness       586672 non-null  float64
 14  acousticness      586672 non-null  float64
 15  instrumentalness  586672 non-null  float64
 16  liveness          586672 non-null  float64
 17  valence           586672 non-null  float64
 18  tempo             586672 non-null  float64
 19  time_signature    586672 non-null  int64  
dtypes: float64(9), int64(6), object(5)
memory usage: 89.5+ MB
```

```
# missing data
data.isnull().sum()
```

```
id                  0
name               71
popularity          0
duration_ms         0
```

```
explicit            0
artists             0
id_artists          0
release_date        0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
time_signature      0
dtype: int64
```

```
# summary
data.describe().transpose()
```

	count	mean	std	min	25%	50%	75%
popularity	586672.0	27.570053	18.370642	0.0	13.0000	27.000000	41.00000
duration_ms	586672.0	230051.167286	126526.087418	3344.0	175093.0000	214893.000000	263867.00000
explicit	586672.0	0.044086	0.205286	0.0	0.0000	0.000000	0.00000
danceability	586672.0	0.563594	0.166103	0.0	0.4530	0.577000	0.68600
energy	586672.0	0.542036	0.251923	0.0	0.3430	0.549000	0.74800

✓ we should change the release_date to a date type and then put months and years into separate columns.

loudness	586672.0	-10.206067	5.089328	-60.0	-12.8910	-9.243000	-6.48200

```
data[["year", "month", "day"]] = data["release_date"].str.split("-", expand = True)
data[["year", "month", "day"]]
```

year month day

✅ What are the most popular songs right now?

```
most_popular = data.query('popularity>90', inplace=False).sort_values('popularity', ascending=False)
most_popular[:10]
```

	id	name	popularity	duration_ms	explicit	artists	
93802	4iJyoBOLtHqaGxP12qzhQI	Peaches (feat. Daniel Caesar & Giveon)	100	198082	1	['Justin Bieber', 'Daniel Caesar', 'Giveon']	['1uNFoZAH '20wkVl
93803	7lPN2DXiMsVn7XUKtOW1CS	drivers license	99	242014	1	['Olivia Rodrigo']	['1McMsnEEl
		Astronaut					

✓ Sort the filtered values and show the columns of interest

```
pop_date = most_popular.sort_values('release_date', ascending=False)
pop_date[['name', 'popularity', 'explicit','release_date']][:20]
```

	name	popularity	explicit	release_date
93802	Peaches (feat. Daniel Caesar & Giveon)	100	1	2021-03-19
93805	Leave The Door Open	96	0	2021-03-05
93815	What's Next	91	1	2021-03-05
93811	Hold On	92	0	2021-03-05
93816	We're Good	91	0	2021-02-11
93813	911	91	1	2021-02-05
93809	Up	92	1	2021-02-05
93806	Fiel	94	0	2021-02-04

☹ March 2020 the world went under a complete lockdown because of the Covid-19.

✓ We try to know songs that released in March 2020 and their popularity

93810	Goosebumps - Remix	92	1	2021-01-15

```
most_popular_march_20 = data.query('(popularity > 80) and (year in ["2020"]) and (month in ["03"])')
most_popular_march_20[['id', 'name', 'explicit','popularity','year','month']][:20]
```

	id	name	explicit	popularity	year	month
92810	5QO79kh1waicV47BqGRL3g	Save Your Tears	1	97	2020	03
92813	0VjIjW4GlUZAMYd2vXMi3b	Blinding Lights	0	96	2020	03
92816	3FAJ6O0NOHQV8Mc5Ri6ENp	Heartbreak Anniversary	0	94	2020	03
92853	4xqrdfXkTW4T0RauPLv3WA	Heather	0	89	2020	03
92867	5nujrmhLynf4yMoMtj8AQF	Levitating (feat. DaBaby)	0	89	2020	03
92927	7szuecWAPwGoV1e5vGu8tl	In Your Eyes	1	86	2020	03
92951	6KfoDhO4XUWSbnyKjNp9c4	Maniac	0	86	2020	03
92961	3PfIrDoz19wz7qK7tYeu62	Don't Start Now	0	85	2020	03
92995	5m5aY6S9ttfIG157xli2Rs	Alô Ambev (Segue Sua Vida) - Ao Vivo	0	84	2020	03
93021	527k23H0A4Q0UJN3vGs0Da	After Party	1	84	2020	03

✅How do different features of a song impact its popularity?

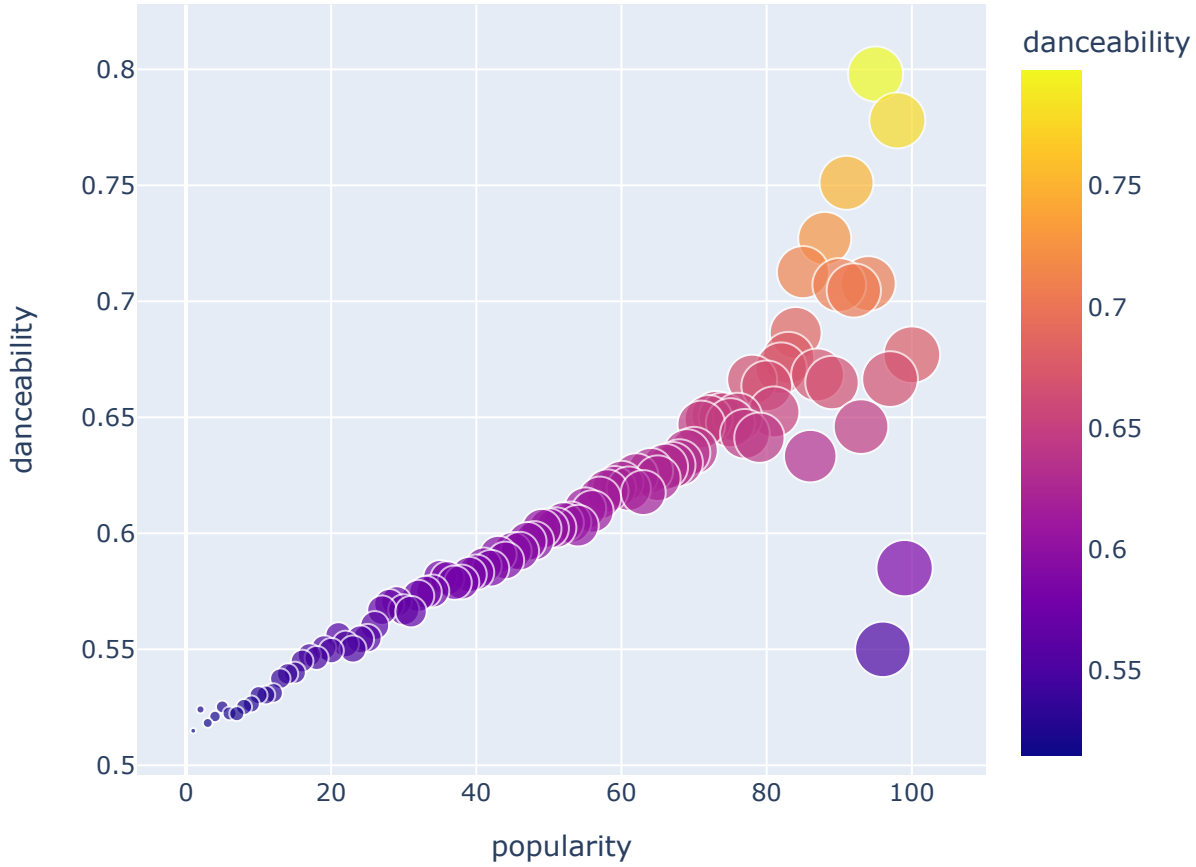
93071	1iaTO3ngY3oAAYvCThIvnM	WHATS POPPIN	1	83	2020	03

```
data1=data.groupby('popularity')['danceability'].mean().sort_values(ascending=[False]).reset_index()
data1.head()
```

	popularity	danceability
0	95	0.798000
1	98	0.778000
2	91	0.751091
3	88	0.727105
4	85	0.712600

✓ This dataframe will have the popularity for different songs grouped by the mean of the danceability score.

```
fig1 = px.scatter(data1, x="popularity", y="danceability", color="danceability",size='popularity')
fig1.show()
```



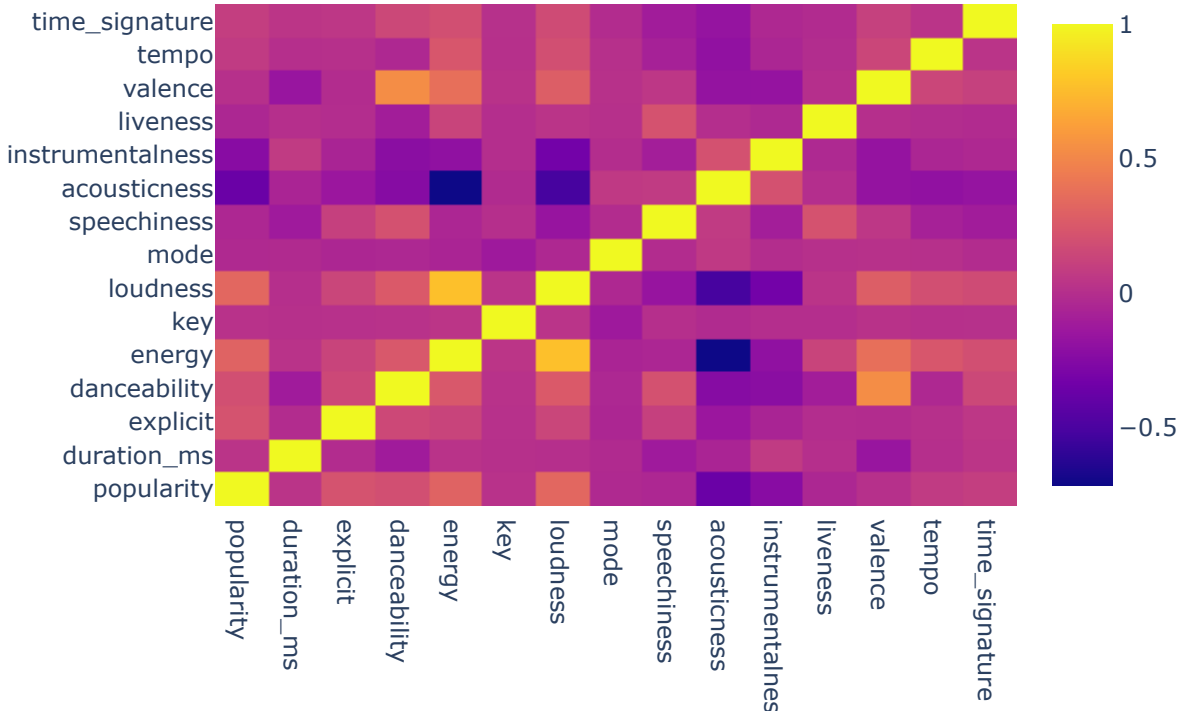
✓ We release that 'popularity' and 'danceability' are positively correlated, which implies that, as the popularity of the song increase, the danceability score for that song also increases.

✅ Calculate Pearson's Correlation Constant 'r' for two different features. The following are the three conditions for the Pearson's Correlation Coefficient 'r':- r>0, implies, positive correlation r=0, implies, no correlation r<0, implies, negative correlation.

```
data_1 = data1['popularity']
data_2 = data1['danceability']

# calculate Pearson's correlation
corr, _ = pearsonr(data_1, data_2)
print('Pearsons correlation: %.3f' % corr)
```

```
Pearsons correlation: 0.899
```

```
matrix=data.corr()  #returns a matrix with correlation of all features
x_list=['popularity','duration_ms','explicit',
        'danceability','energy','key','loudness',
        'mode','speechiness','acousticness','instrumentalness',
        'liveness','valence','tempo','time_signature']

fig_heatmap = go.Figure(data=go.Heatmap(
                   z=matrix,
                   x=x_list,
                   y=x_list,
                   hoverongaps = False))
fig_heatmap.update_layout(margin = dict(t=200,r=200,b=200,l=200),
    width = 800, height = 650,
    autosize = False )

fig_heatmap.show()
```



✅ We observe that there is no significant positive correlation between popularity and a song's feature. The most positive correlation occurs between popularity, danceability, loudness, and energy.

✅MOST POPULAR ARTIST

```
artists_popular = data_artist.sort_values(by=['popularity'], ascending=False).reset_index()
artists_popular[:10]
```

	index	id	followers	genres	name	popularity
0	144481	1uNFoZAHBGtllmzznpCI3s	44606973.0	['canadian pop', 'pop', 'post-teen pop']	Justin Bieber	100
1	115489	4q3ewBCX7sLwd24euuV69X	32244734.0	['latin', 'reggaeton', 'trap latino']	Bad Bunny	98
2	126338	06HL4z0CvFAxyc27GXpf02	38869193.0	['pop', 'post-teen pop']	Taylor Swift	98
3	313676	3TVXtAsR1Inumwj472S9r4	54416812.0	['canadian hip hop', 'canadian pop', 'hip hop'...	Drake	98
4	144484	3Nrfpe0tUJi4K4DXYWgMUX	31623813.0	['k-pop', 'k-pop boy group']	BTS	96
5	115490	4MCBfE4596Uoi2O4DtmEMz	16996777.0	['chicago rap', 'melodic rap']	Juice WRLD	96
6	144483	1Xyo4u8uXC1ZmMpatF05PJ	31308207.0	['canadian contemporary r&b', 'canadian pop', ...	The Weeknd	96
					Ariana	

▶ Analyzing the Genres

```
data_artist[data_artist["genres"]=='[]']
df_genre=data_artist[data_artist["genres"]!='[]']
df_genre.head()
```

▶ We observe that the column 'genres' has a list passed as value. Let's split these lists into individual values.

```
df_sort_genres=pd.DataFrame(df_genre.assign(genres=df_genre.genres.str.split(",")).explode('genres'))
df_sort_genres.tail()
```

	id	followers	genres	name	popularity
1104328	1q9C5XlekzXbRLIuLCDTre	90087.0	'teen pop']	Brent Rivera	33
1104331	4fh2BIKYPFvXFsQLhaeVJp	309.0	['la indie']	Lone Kodiak	20
1104334	7akMsd2vb4xowNTehv3gsY	774.0	['indie rockism']	The Str!ke	0
1104336	35m7AJrUCtHYHyIUhCzmgi	205.0	['indie rockism']	Hunter Fraser	6
1104345	1IjurfXKPlGncNdW3J8zJ8	2123.0	['deep acoustic pop']	Right the Stars	18

```
df_sort_genres['genres']=df_sort_genres.genres.str.replace('[',' ')
df_sort_genres['genres']=df_sort_genres.genres.str.replace(']',' ')
```

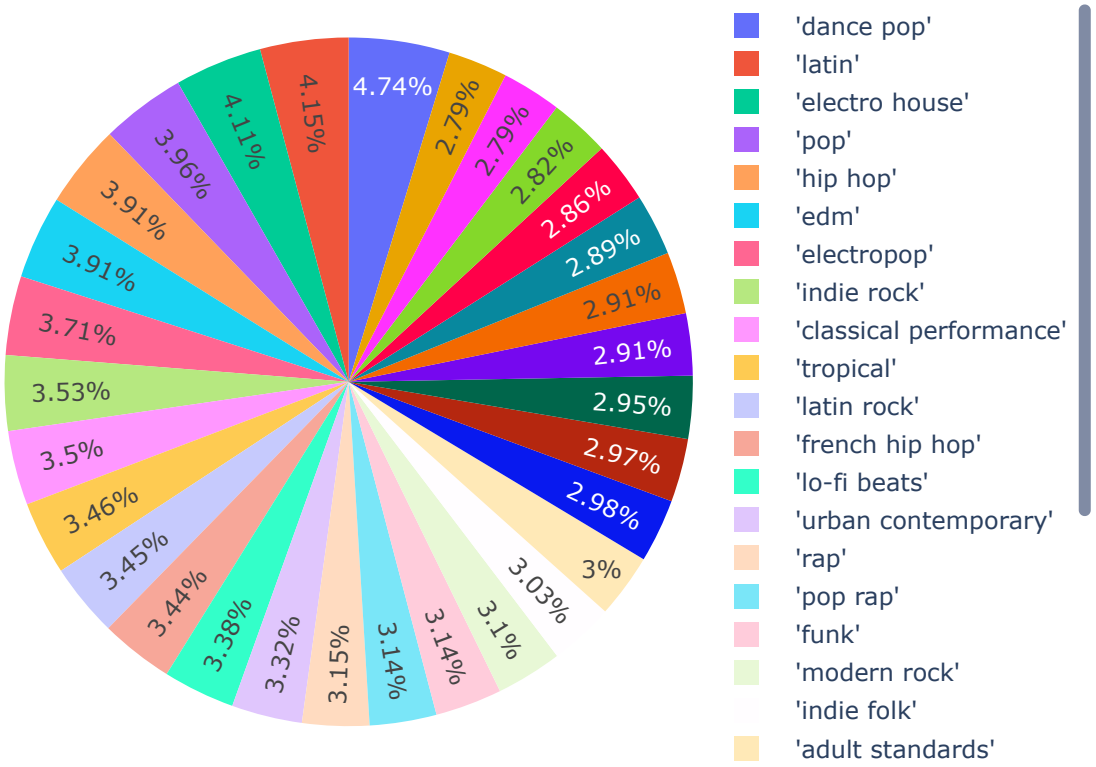
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:

The default value of regex will change from True to False in a future version. In addition, single character regular expression

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning:

The default value of regex will change from True to False in a future version. In addition, single character regular expression
```

```
# get top 30 most commom genres
n = 30
top_30=pd.DataFrame(df_sort_genres['genres'].value_counts()[:n]).reset_index()
top_30.rename(columns = {'index':'Genres','genres':'Total_Count'}, inplace = True)
top_30
```

	Genres	Total_Count
0	'dance pop'	551
1	'latin'	483
2	'electro house'	478
3	'pop'	461
4	'hip hop'	455
5	'edm'	455
6	'electropop'	432
7	'indie rock'	411
8	'classical performance'	407
9	'tropical'	402
10	'latin rock'	401
11	'french hip hop'	400
12	'lo-fi beats'	393
13	'urban contemporary'	386
14	'rap'	366
15	'pop rap'	365
16	'funk'	365
17	'modern rock'	361
18	'indie folk'	353
19	'adult standards'	349
20	'pop dance'	347
21	'country rock'	346

22	'uk hip hop'	343
23	'corrido'	339
24	'stomp and holler'	338
25	'art rock'	336

```
fig3 = px.pie(top_30, values='Total_Count', names='Genres')
fig3.show()
```



✓ 0 sn. tamamlanma zamanı: 15:05