Spotify

Home About Content Others

SPOTIFY

Presentation

Guillermo David
James Latanna
Matthias Mangold

Smash Mouth - All Stars

CONTENTS:

HOW DOES THE DURATION OF A TRACK AFFECT ITS POPULARITY?

DOES FOLLOWERS AFFECT ARTIST'S POPULARITY IN THE TOP 100?

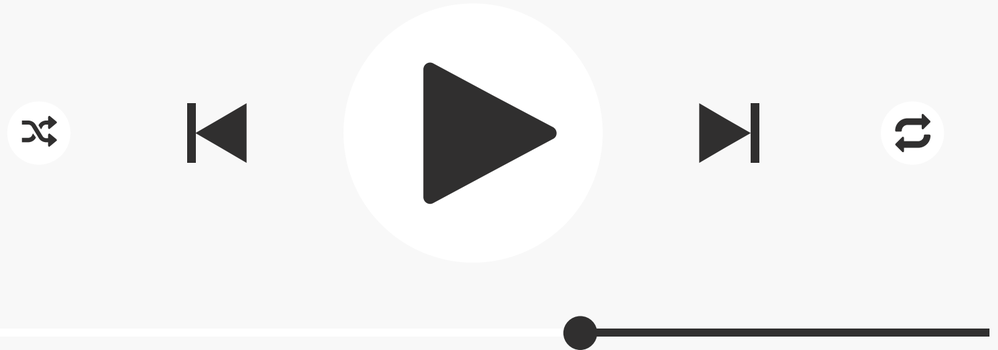
DOES THE DANCEABILITY AFFECTS THE POPULARITY OF A SONG?

Darude - Sandstorm

RESEARCH QUESTION 1 : HOW DOES THE DURATION OF A TRACK AFFECT ITS POPULARITY?

H_0: THERE IS NO SIGNIFICANT DIFFERENCE IN POPULARITY BETWEEN TRACKS OF DIFFERENT LENGTHS (SHORT, MODERATE, AND LONG).

H_1: TRACKS WITH MODERATE LENGTH (3 TO 4 MINUTES) ARE MORE POPULAR THAN SHORTER OR LONGER TRACKS.



RESEARCH QUESTION: HOW DOES THE DURATION OF A TRACK AFFECT ITS POPULARITY?

1. COLLECT DATA FROM SPOTIFY WEB API

```
df.to_csv('artist_tracks.csv', index=False)
```

track_id	artist_id	track_name	artist_name	duration_ms	popularity
2GujK1FWxxOZ118PaWNgbZ	6s22t5Y3prQHyaHWUN1R1C	World's Smallest Violin	AJR	180742	64
4SQLQfcR0vhyIN4uPBlc0d	6s22t5Y3prQHyaHWUN1R1C	Bang!	AJR	170858	67
2QKLrgXNQtK2c8QbLRO3XC	6s22t5Y3prQHyaHWUN1R1C	Burn The House Down	AJR	212373	67
2sVcBaVtBPM9vIFk1Jnbw8	6s22t5Y3prQHyaHWUN1R1C	Weak	AJR	201159	63
3LDNcikQd7Zui9gJCISTtR	6s22t5Y3prQHyaHWUN1R1C	Yes I'm A Mess	AJR	164973	65
4rnyUV17cSZGsz18xJNdjL	6s22t5Y3prQHyaHWUN1R1C	100 Bad Days	AJR	212783	62
1UDhFEt0NlOFKwWRQJTgHe	6s22t5Y3prQHyaHWUN1R1C	Way Less Sad	AJR	206108	62
7Gk8IPlFTLUcoIH3mtASI0	6s22t5Y3prQHyaHWUN1R1C	Record Player	AJR	149538	60
5tf8NV1rHghYymjIqeKMus	6s22t5Y3prQHyaHWUN1R1C	Sober Up (feat. Rivers Cuon	AJR	218763	59
44B9xspO3RycWjEJ3D8cKl	6s22t5Y3prQHyaHWUN1R1C	The Good Part	AJR	227267	55
51ZQ1vr10ffzbwIjDCwqm4	66CXWjxzNUsdJxJ2JdwvnR	we can't be friends (wait fo	Ariana Grande	228639	87
0Lmbke3KNVFXtoH2mMSHCw	66CXWjxzNUsdJxJ2JdwvnR	the boy is mine	Ariana Grande	173639	83
2o1pb13quMReXZqE7jWsgq	66CXWjxzNUsdJxJ2JdwvnR	intro (end of the world)	Ariana Grande	92400	82
1oFAF1hdPOickyHgbuRjyX	66CXWjxzNUsdJxJ2JdwvnR	Save Your Tears (Remix) (w	Ariana Grande	191013	79
7xoUc6faLbCqZO6fQEYprd	66CXWjxzNUsdJxJ2JdwvnR	One Last Time	Ariana Grande	197266	82
6ocbgoVGwYJhOv1GgI9NsF	66CXWjxzNUsdJxJ2JdwvnR	7 rings	Ariana Grande	178626	82

2. PREPROCESS DATA

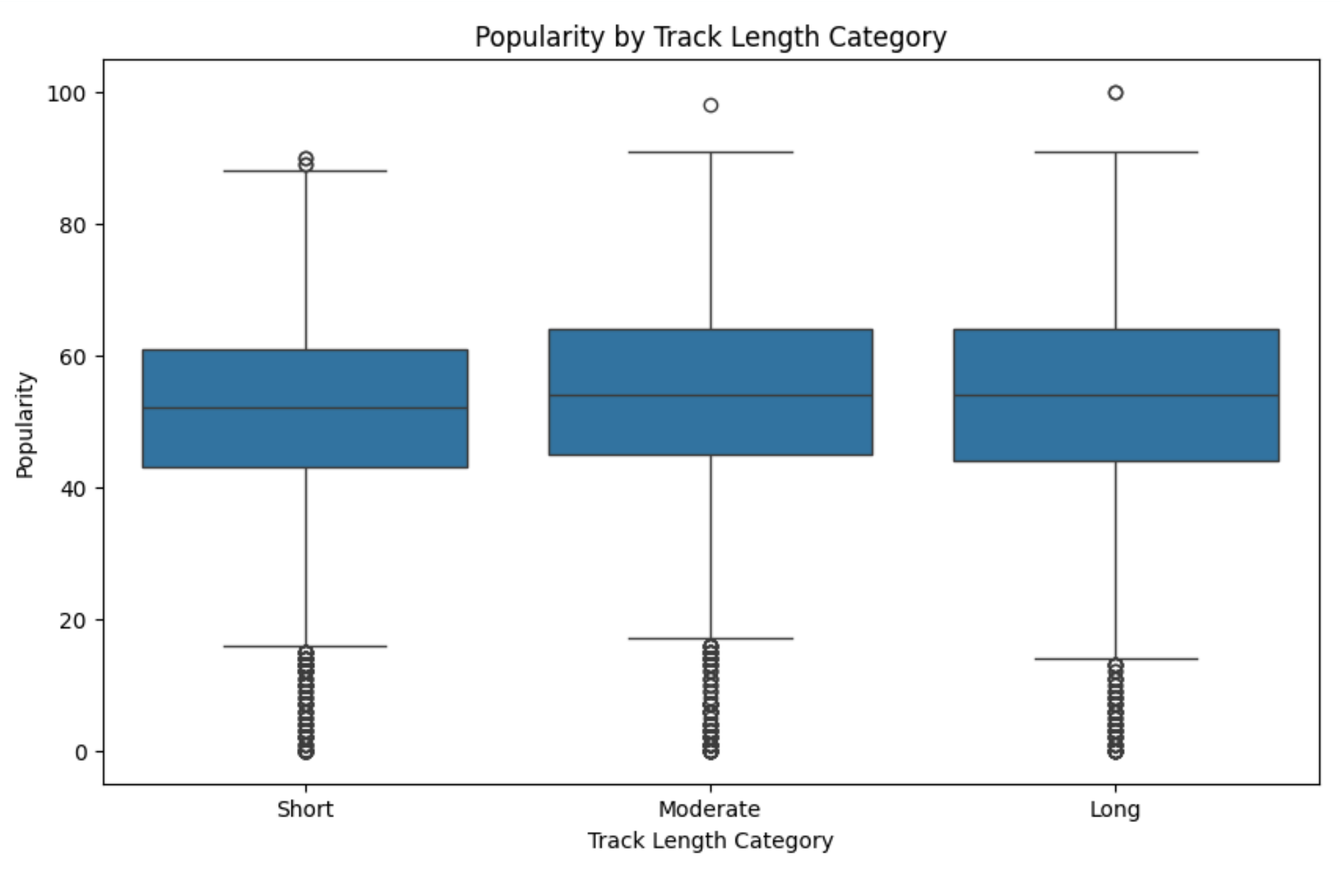
```
df = pd.read_csv('artist_tracks.csv')

df['duration_min'] = df['duration_ms'] / 60000  # Convert milliseconds to minutes

def categorize_length(duration):
    if duration < 3:
        return 'Short'
    elif 3 <= duration <= 4:
        return 'Moderate'
    else:
        return 'Long'

df['length_category'] = df['duration_min'].apply(categorize_length)
```

```
import time
import pandas as pd

def create_dataframe(artists, max_artists=1000):  # Add max_artists parameter
    df_list = []
    count = 0
    for i in range(min(max_artists, len(artists))):  # Iterate through rows
        artist = artists.iloc[i]  # Access row data using iloc
        artist_id = artist['id']
        artist_name = artist['name']
        # Retrieve top tracks for the artist
        top_tracks = get_artist_top_tracks(artist_id)
        time.sleep(1)  # Wait 1 second before fetching top tracks
        if top_tracks:
            for track in top_tracks[:10]:
                track_data = {
                    "track_id": track['id'],
                    "artist_id": artist_id,
                    "track_name": track['name'],
                    "artist_name": artist_name,
                    "duration_ms": track['duration_ms'],
                    "popularity": track['popularity']
                }
                df_list.append(track_data)
                count += 1
                if count % 50 == 0:
                    time.sleep(1)  # Wait an extra second after every 50 tracks

    return pd.DataFrame(df_list)


# Process up to 1000 artists to potentially get 10000 entries
df = create_dataframe(artists, max_artists=1000)
df
```

✓ 21m 44s completed at 11:31 AM

RESEARCH QUESTION: HOW DOES THE DURATION OF A TRACK AFFECT ITS POPULARITY?

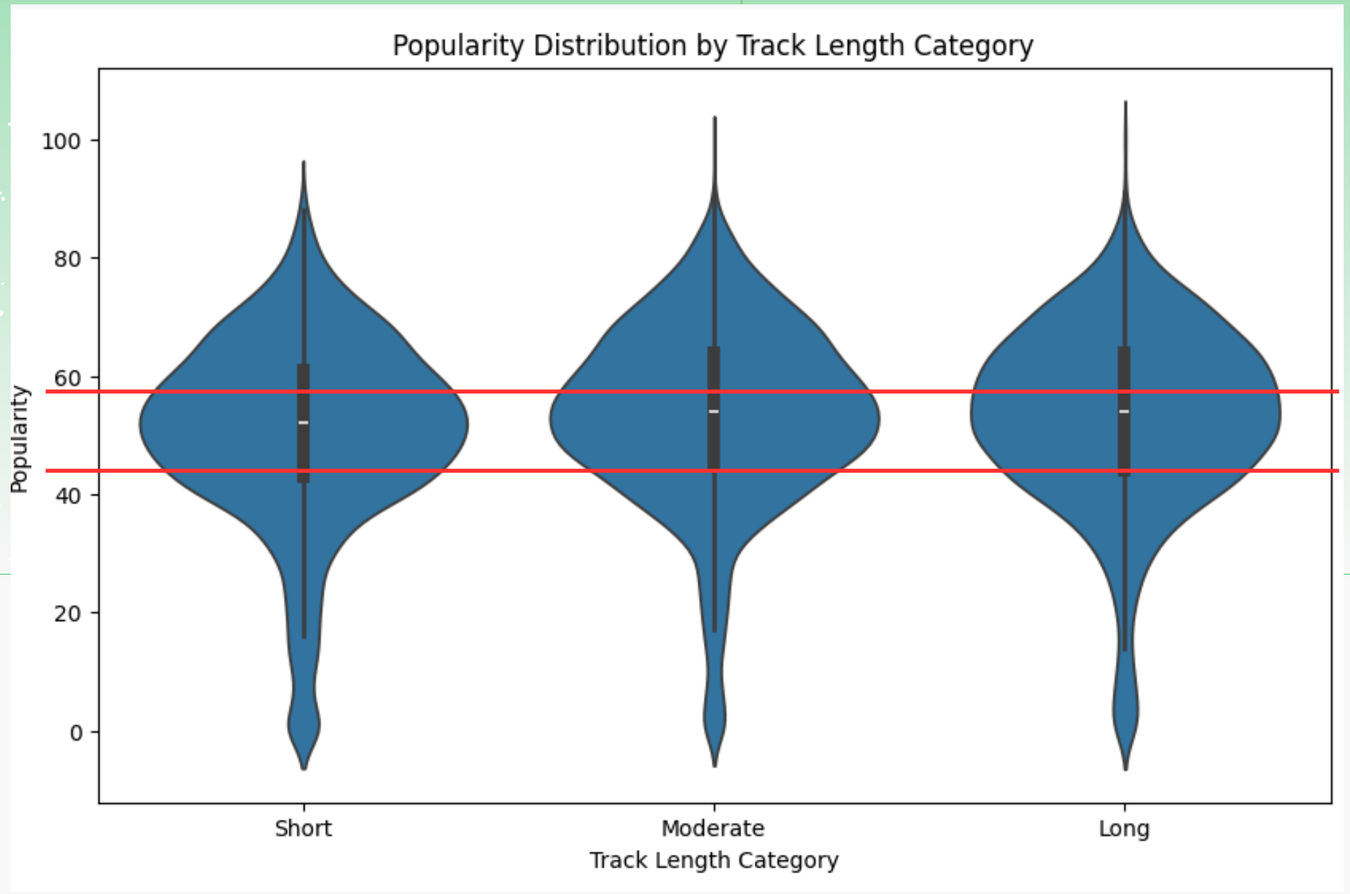
3. EXPLORATORY DATA ANALYSIS



- The median popularity score for all track length categories (short, moderate, and long) is quite similar, around 55 to 60.
- All three categories contain a few outliers above the 80th percentile, indicating that some tracks achieve a high popularity score, regardless of length.
- The lower and upper whiskers show that the popularity distribution spans a wide range in each category, with minimum values close to 0 and maximum values near 100. The range of popularity is similar for all track lengths.

RESEARCH QUESTION: HOW DOES THE DURATION OF A TRACK AFFECT ITS POPULARITY?

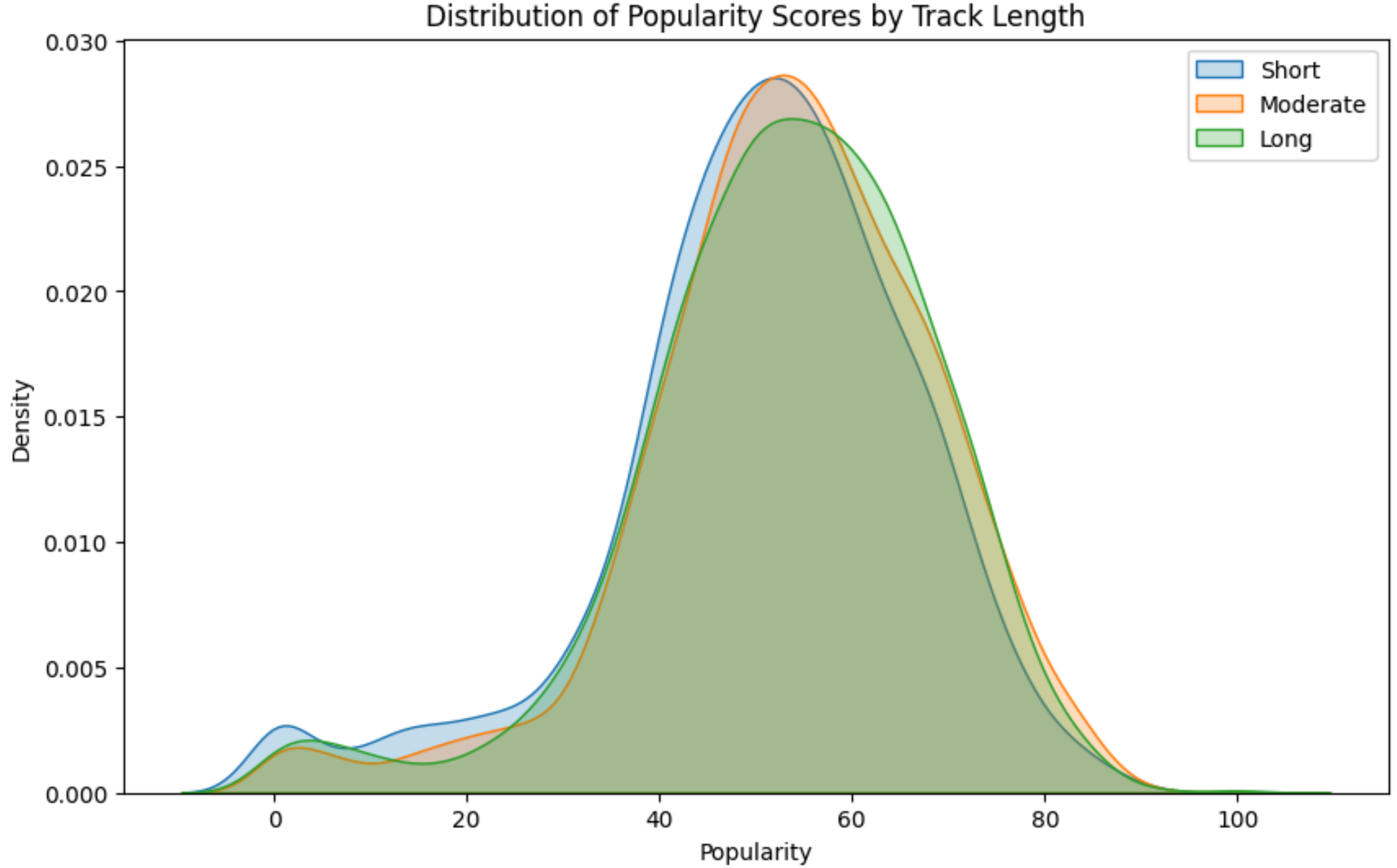
3. EXPLORATORY DATA ANALYSIS



- All three track length categories have a similar shape and a concentration of tracks with popularity scores around 45 to 55.
- The median popularity is approximately the same for all categories, slightly above 50, and there are no significant visible differences in terms of central tendency between short, moderate, and long tracks.

RESEARCH QUESTION: HOW DOES THE DURATION OF A TRACK AFFECT ITS POPULARITY?

3. EXPLORATORY DATA ANALYSIS



- The peaks for all categories occur in the 50-60 popularity range, with moderate-length tracks showing a very slight shift toward the higher popularity range compared to the short and long tracks.
- The curves largely overlap, indicating minimal differences between the distributions.

3. EXPLORATORY DATA ANALYSIS

Conclusion of EDA:

Based on the box, violin, and density plots, **the data does not provide strong evidence to support the alternative hypothesis** (H_1) that tracks of moderate length (3 to 4 minutes) are more popular than shorter or longer tracks.

The null hypothesis (H_0), which states that there is no significant difference in popularity between tracks of different lengths, **appears to hold**.

RESEARCH QUESTION: HOW DOES THE DURATION OF A TRACK AFFECT ITS POPULARITY?

4. ANOVA TEST

Hypothesis:

H_0: There is no significant difference in popularity between tracks of different lengths (short, moderate, and long).

H_1: Tracks with moderate length (3 to 4 minutes) are more popular than shorter or longer tracks.

Before, with large dataset:

```
# Perform ANOVA test
f_stat, p_value = f_oneway(popularity_short, popularity_moderate, popularity_long)

print(f"F-statistic: {f_stat}")
print(f"P-value: {p_value}")
```

```
F-statistic: 32.039079353959835
P-value: 1.3520694955788396e-14
```

We would reject the null hypothesis since the p-value is much smaller than 0.05.

Possible Reason:
Large Sample Size: ANOVA is sensitive to sample size, and large samples can produce low p-values even when group means are very close.

Now, with smaller dataset:

```
# Perform ANOVA test
f_stat, p_value = f_oneway(popularity_short, popularity_moderate, popularity_long)

print(f"F-statistic: {f_stat}")
print(f"P-value: {p_value}")
```

```
F-statistic: 1.0128338627951912
P-value: 0.36357008956578263
```

Fail to Reject H_0:
Using a smaller sample size, the p-value is now greater than 0.05. we fail to reject the null hypothesis H_0.

This means there's no statistically significant difference in popularity between tracks of different lengths based on this sample.

Thynk Unlimited

RQ 2 : DOES FOLLOWERS AFFECT ARTIST'S POPULARITY IN THE TOP 100?

H_0:
THE NUMBER OF FOLLOWERS DOES NOT SIGNIFICANTLY AFFECT THE ARTIST'S POPULARITY

H_1:
THE NUMBER OF FOLLOWERS HAS A SIGNIFICANT IMPACT ON THE ARTIST'S POPULARITY

Continuum

```python
def get_access_token():
    client_id = "4b180c9573244a5892f11df40ecad4d6"
    client_secret = "03d2128bf1b74dd29ac62f488aa5ead8"

    url = "https://accounts.spotify.com/api/token"
    headers = {
        "Authorization": "Basic " + base64.b64encode(f"{client_id}:{client_secret}".encode()).decode(),
        "Content-Type": "application/x-www-form-urlencoded"
    }
    data = {
        "grant_type": "client_credentials"
    }

    response = requests.post(url, headers=headers, data=data)
    return response.json().get("access_token")
```

The function sends a POST request to Spotify's server to obtain the access token from the JSON response.
The client ID and secret are combined and encoded in Base64 to ensure security before being sent to Spotify.
This encoding protects the sensitive information.

```python
[5] access_token = get_access_token()

    # Print the access token to see it
    print("Access Token:", access_token)
```

```
Access Token: BQBnIPAJW3zwu-jofLN49A9iuSgKDI4-6JGyj5wO4BCpb5GrIeLXMiPZ-2Fv0A47y5ddEEbGfyUeJCeely05Aezbx4sIW7UnJKllFlNayy1QjHqRra0
```

```python
def search_all_artists(limit=50):
    token = get_access_token()
    if not token:
        return []

    all_artists = []
    for letter in range(65, 91):  # Loop through A-Z
        offset = 0
        while True:
            url = f"https://api.spotify.com/v1/search?q={chr(letter)}&type=artist&limit={limit}&offset={offset}"
            response = requests.get(url, headers={"Authorization": f"Bearer {token}"})
            if not response.ok:
                break

            artists = response.json().get("artists", {}).get("items", [])
            if not artists:
                break
            all_artists.extend(artists)
            offset += limit

    return all_artists
```

A GET request is then sent to the Spotify API, with the authorization header containing the access token.
The loop continues to request more data until there are no more results for a given letter.

```python
artist_data = search_all_artists()

artist_df = pd.DataFrame(artist_data)

artist_df
```

	external_urls	followers	genres	href	id	images	name	popularity	type	
0	{'spotify': 'https://open.spotify.com/artist/6...	{'href': None, 'total': 3460480}	[pov: indie]	https://api.spotify.com/v1/artists/6s22t5Y3prQ...	6s22t5Y3prQHyaHWUN1R1C	[{'height': 640, 'url': 'https://i.scdn.co/ima...	AJR	73	artist	spotify:artist:6s22t5Y3prQHyaHV
1	{'spotify': 'https://open.spotify.com/artist/6...	{'href': None, 'total': 10029160}	[pop]	https://api.spotify.com/v1/artists/66CXWjxzNUs...	66CXWjxzNUsdJxJ2JdwvnR	[{'height': 640, 'url': 'https://i.scdn.co/ima...	Ariana Grande	92	artist	spotify:artist:66CXWjxzNUsdJx...
2	{'spotify': 'https://open.spotify.com/artist/0...	{'href': None, 'total': 4548436}	[contemporary r&b, dance pop, hip pop, r&b, ur...	https://api.spotify.com/v1/artists/0urTpYCsixq...	0urTpYCsixqZwgNTkPJOJ4	[{'height': 640, 'url': 'https://i.scdn.co/ima...	Aaliyah	69	artist	spotify:artist:0urTpYCsixqZwgN
3	{'spotify': 'https://open.spotify.com/artist/1...	{'href': None, 'total': 15065573}	[east coast hip hop, hip hop, rap]	https://api.spotify.com/v1/artists/13ubrt8QOOC...	13ubrt8QOOCPljQ2FL1Kca	[{'height': 640, 'url': 'https://i.scdn.co/ima...	A$AP Rocky	86	artist	spotify:artist:13ubrt8QOOCPljQ
4	{'spotify': 'https://open.spotify.com/artist/7...	{'href': None, 'total': 29544827}	[australian rock, hard rock, rock]	https://api.spotify.com/v1/artists/711MCceyCBc...	711MCceyCBcFnzjGY4Q7Un	[{'height': 640, 'url': 'https://i.scdn.co/ima...	AC/DC	84	artist	spotify:artist:711MCceyCBcFnzj

```python
top100 = new_df.head(100)
top100
```

	followers	genres	id	name	popularity
5	123299629	[pop]	06HL4z0CvFAxyc27GXpf02	Taylor Swift	100
3162	92414786	[canadian contemporary r&b, canadian pop, pop]	1Xyo4u8uXC1ZmMpatF05PJ	The Weeknd	96
262	100535598	[art pop, pop]	6qqNVTkY8uBg9cP3Jd7DAH	Billie Eilish	96
3001	92458597	[canadian hip hop, canadian pop, hip hop, pop ...	3TVXtAsR1Inumwj472S9r4	Drake	96
2028	13457565	[pop]	74KM79TiuVKeVCqs8QtB0B	Sabrina Carpenter	95
...	...	...	...	...	...
3660	16367137	[pop, pop r&b]	6LuN9FCkKOj5PcnpouEgny	Khalid	85
185	32811732	[r&b, soul]	3fMbdgg4jU18AjLCKBhRSm	Michael Jackson	85
5014	967607	[reggaeton chileno]	7CvTknweLr9feJtRGrpDBy	FloyyMenor	85
10016	5730095	[australian hip hop]	2tIP7SsRs7vjlcLrU85W8J	The Kid LAROI	85
1666	7511789	[big room, brostep, dutch edm, edm, house, pop...	2o5jDhtHVPhrJdv3cEQ99Z	Tiësto	85

100 rows × 5 columns

Filter it to get the top 100 popular artists

```
df['name'].duplicated().sum()
```

```
6936
```

CHECK IF THE DF HAS ANY DUPLICATED ITEMS

```
df.describe()
```

	popularity
count	26000.000000
mean	54.739500
std	17.306422
min	0.000000
25%	48.000000
50%	57.000000
75%	65.000000
max	100.000000

```
new_df = df.drop_duplicates(subset='name').sort_values(by='popularity', ascending=False)
new_df
```

	followers	genres	id	name	popularity
5	123299629	[pop]	06HL4z0CvFAxyc27GXpf02	Taylor Swift	100
3162	92414786	[canadian contemporary r&b, canadian pop, pop]	1Xyo4u8uXC1ZmMpatF05PJ	The Weeknd	96
262	100535598	[art pop, pop]	6qqNVTkY8uBg9cP3Jd7DAH	Billie Eilish	96
3001	92458597	[canadian hip hop, canadian pop, hip hop, pop ...	3TVXtAsR1Inumwj472S9r4	Drake	96
2028	13457565	[pop]	74KM79TiuVKeVCqs8QtB0B	Sabrina Carpenter	95
...	...	...	...	...	...
24096	51	[]	5K62S65kurpFZeRHp8T7U9	Ÿ	0
21019	2	[]	35X3x3lq5bnjTiY5ECdcFH	VAGOI	0
18087	11	[]	6i7lzGFpDm0za0n1XXc4dG	Š	0
111	8	[]	7EKU6tinAyODDKWhcKBa1n	Ã	0
94	15	[]	5FnG9lLegyiFtLtZG2XLQ4	Ã	0

19064 rows × 5 columns

DROP ANY DUPLICATED ITEMS AND
SORT THE DF BY POPULARITY

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26000 entries, 0 to 25999
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   external_urls  26000 non-null  object
 1   followers      26000 non-null  object
 2   genres         26000 non-null  object
 3   href           26000 non-null  object
 4   id             26000 non-null  object
 5   images         26000 non-null  object
 6   name           26000 non-null  object
 7   popularity     26000 non-null  int64
 8   type           26000 non-null  object
 9   uri            26000 non-null  object
dtypes: int64(1), object(9)
memory usage: 2.0+ MB
```

```
[12] df.shape
```

```
(26000, 10)
```

```
[14] df.columns
```

```
Index(['external_urls', 'followers', 'genres', 'href', 'id', 'images', 'name',
       'popularity', 'type', 'uri'],
      dtype='object')
```

```
[15] df.drop(['external_urls', 'href', 'images', "uri", "type"], axis=1, inplace=True)
```

```
df
```

	followers	genres	id	name	popularity
0	{'href': None, 'total': 3460480}	[pov: indie]	6s22t5Y3prQHyaHWUN1R1C	AJR	73
1	{'href': None, 'total': 100291604}	[pop]	66CXWjxzNUsdJxJ2JdwvnR	Ariana Grande	92
2	{'href': None, 'total': 4548436}	[contemporary r&b, dance pop, hip pop, r&b, ur...	0urTpYCsixqZwgNTkPJOJ4	Aaliyah	69
3	{'href': None, 'total': 15065573}	[east coast hip hop, hip hop, rap]	13ubrt8QOOCPljQ2FL1Kca	A$AP Rocky	86
4	{'href': None, 'total': 29544827}	[australian rock, hard rock, rock]	711MCceyCBcFnzjGY4Q7Un	AC/DC	84
...	...	...	...	...	...
25995	{'href': None, 'total': 10241}	[]	1kqmfPNXoed2ueFvyMfNP7	Z Berg	34
25996	{'href': None, 'total': 31167}	[chinese singer-songwriter, mainland chinese pop]	1p8mNyT18G4coJooY8NTGN	金志文	39
25997	{'href': None, 'total': 270}	[calming instrumental]	1rXR5cwxxippMLTtaeAa6y	Leo Zimmerman	50
25998	{'href': None, 'total': 1437}	[]	7DHVVPqZYKCrro216FSMzq	Zarech	31
25999	{'href': None, 'total': 3265}	[glitchcore]	3pdRi1JLxXHjxBiyLLTmli	Zootzie	32

26000 rows × 5 columns

DROP COLUMNS THAT ARE UNRELATED TO OBJECTIVES

Thynk Unlimited

RQ 2 : DOES FOLLOWERS AFFECT ARTIST'S POPULARITY IN THE TOP 1000?

1. Check the coorelation between popularity and followers

```
testdata = top100[['popularity', 'followers']].copy()

testdata.dropna(inplace=True)
correlation = testdata.corr().iloc[0, 1]

print(f"Correlation between popularity and followers: {correlation}")
```

```
Correlation between popularity and followers: 0.5774766417590353
```

THIS INDICATES A MODERATE POSITIVE RELATIONSHIP BETWEEN FOLLOWERS AND POPULARITY.
SO WHEN THE FOLLOWERS INCREASE THE POPULARITY WILL ALSO INCREASE

2. Use regression model to further find out information about it

```
import statsmodels.api as sm

# Define the independent and dependent variables
X = top100['popularity']  # Independent variable
y = top100['followers']  # Dependent variable

# Add a constant to the independent variable for the intercept
X = sm.add_constant(X)

# Fit the regression model
model = sm.OLS(y, X).fit()

# Get the summary of the regression results
summary = model.summary()

# Display the summary
summary
```

OLS Regression Results

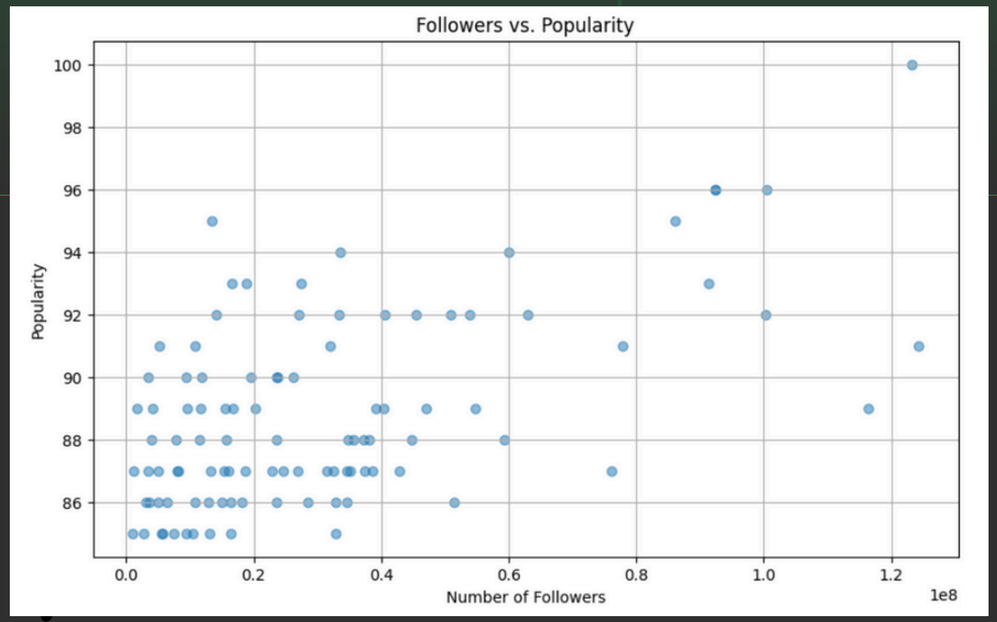
Dep. Variable:	followers	**R-squared:**	0.333
Model:	OLS	**Adj. R-squared:**	0.327
Method:	Least Squares	**F-statistic:**	49.03
Date:	Sun, 13 Oct 2024	**Prob (F-statistic):**	3.18e-10
Time:	06:57:19	**Log-Likelihood:**	-1838.0
No. Observations:	100	**AIC:**	3680.
Df Residuals:	98	**BIC:**	3685.
Df Model:	1		
Covariance Type:	nonrobust		

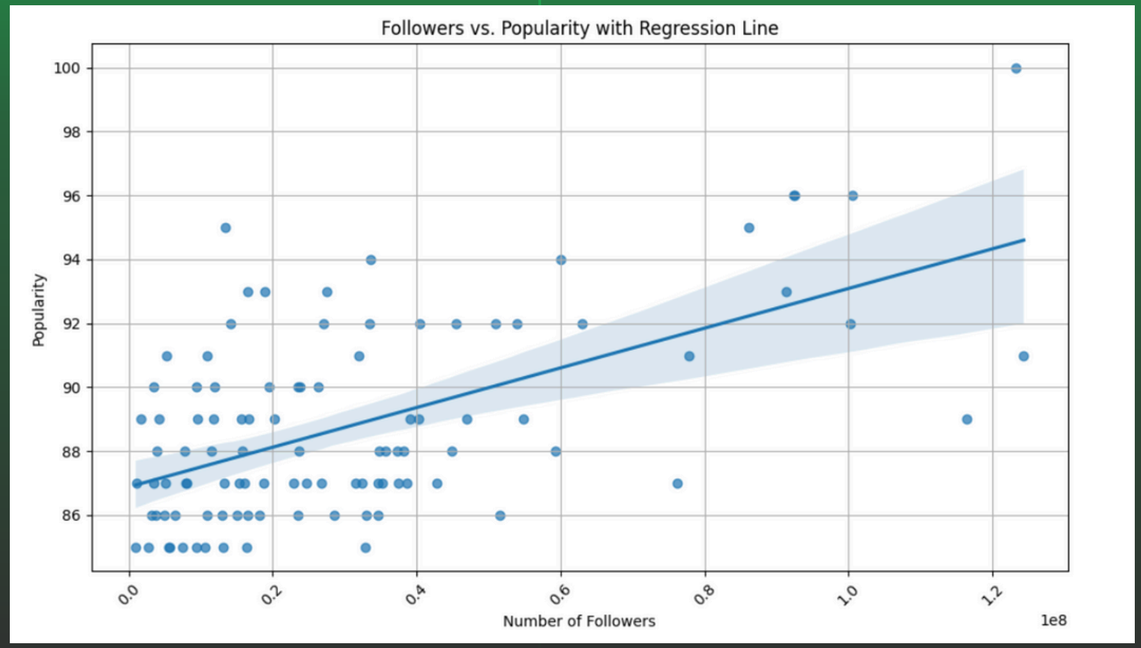
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-4.462e+08	6.82e+07	-6.543	0.000	-5.82e+08	-3.11e+08
popularity	5.373e+06	7.67e+05	7.002	0.000	3.85e+06	6.9e+06

Omnibus:	20.521	**Durbin-Watson:**	2.309
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	29.983
Skew:	0.944	**Prob(JB):**	3.08e-07
Kurtosis:	4.906	**Cond. No.**	2.58e+03

RQ : DOES FOLLOWERS AFFECT ARTIST'S POPULARITY IN THE TOP 100?

3. Visualize the data





To better see the results, use a graph to visualize i

To better see the results, we can add regression line to see that it is in fact inclining

RQ : DOES FOLLOWERS AFFECT ARTIST'S POPULARITY IN THE TOP 1000?

4. NULL HYPOTHESIS

```python
from scipy import stats

follower_threshold = top100['followers'].median()

high_followers_popularity = top100[top100['followers'] > follower_threshold]['popularity']
low_followers_popularity = top100[top100['followers'] <= follower_threshold]['popularity']

t_stat, p_value = stats.ttest_ind(high_followers_popularity, low_followers_popularity, alternative='greater')

print(f"T-statistic: {t_stat}")
print(f"P-value: {p_value}")

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis.")
    print("There is evidence to suggest that artists with more followers have higher popularity.")
else:
    print("Fail to reject the null hypothesis.")
    print("There is not enough evidence to suggest that artists with more followers have higher popularity.")
```

```
T-statistic: 3.540743200810276
P-value: 0.0003061901643848149
Reject the null hypothesis.
There is evidence to suggest that artists with more followers have higher popularity.
```

RQ 3 : DOES THE DANCEABILITY AFFECTS THE POPULARITY OF A SONG?

H0

- Danceability does not have a significant effect on the popularity of a song.

H1

Danceability has a significant effect on the popularity of a song.

GETTING DATA

```python
def get_album_tracks(album_id):
    tracks = sp.album_tracks(album_id)
    tracks = tracks['items']
    tracks = pd.DataFrame(tracks)
    tracks = tracks[['name', 'id']]
    return tracks
```

```python
album_test = get_album_tracks('https://open.spotify.com/i
album_test.head()
```

	name	id
0	Moscow Mule	6Xom58OOXk2SoU711L2IXO
1	Después de la Playa	1dm6z1fWB0cErMszU25dy2
2	Me Porto Bonito	6Sq7ltF9Qa7SNFBsV5Cogx
3	Tití Me Preguntó	1IHWl5LamUGEuP4ozKQSXZ
4	Un Ratito	5CzixCxDkRXX9mScCmah8O

```python
album_list = ['https://open.spotify.com/intl-es/album/3RQ
```

```python
all_songs_df = pd.DataFrame()
for i in album_list:
    temporal_df = get_album_tracks(i)
    all_songs_df = pd.concat([all_songs_df, temporal_df])
all_songs_df = all_songs_df.reset_index(drop=True)
```

```python
def get_audio_features(track_id):
    features = sp.audio_features(track_id)[0]
    return {
        'popularity': sp.track(track_id)['popularity'],
        'danceability': features['danceability'],
        'energy': features['energy'],
        'valence': features['valence'],
        'tempo': features['tempo'],
        'acousticness': features['acousticness'],
        'instrumentalness': features['instrumentalness']
    }
```

```python
all_songs_df = all_songs_df.drop(62).reset_index(drop=True)
```

```python
audio_features = all_songs_df['id'].apply(get_audio_features)
```

```python
audio_features_df = pd.DataFrame(audio_features.tolist())
all_songs_df = pd.concat([all_songs_df, audio_features_df], axis=1)
```

DATA

```
[12] audio_features_df = pd.DataFrame(audio_features.tolist())
     all_songs_df = pd.concat([all_songs_df, audio_features_df], axis=1)
     all_songs_df.head()
```

	name	id	popularity	danceability	energy	valence	tempo
0	Moscow Mule	6Xom58OOXk2SoU711L2IXO	80	0.804	0.674	0.292	99.968
1	Después de la Playa	1dm6z1fWB0cErMszU25dy2	72	0.564	0.903	0.607	78.293
2	Me Porto Bonito	6Sq7ltF9Qa7SNFBsV5Cogx	82	0.911	0.712	0.425	92.005
3	Tití Me Preguntó	1IHWl5LamUGEuP4ozKQSXZ	80	0.650	0.715	0.187	106.672
4	Un Ratito	5CzixCxDkRXX9mScCmah8O	73	0.787	0.546	0.222	93.050

Próximos pasos: Generar código con all_songs_df | Ver gráficos recomendados | New interactive sheet

```
[20] all_songs_df.shape
```

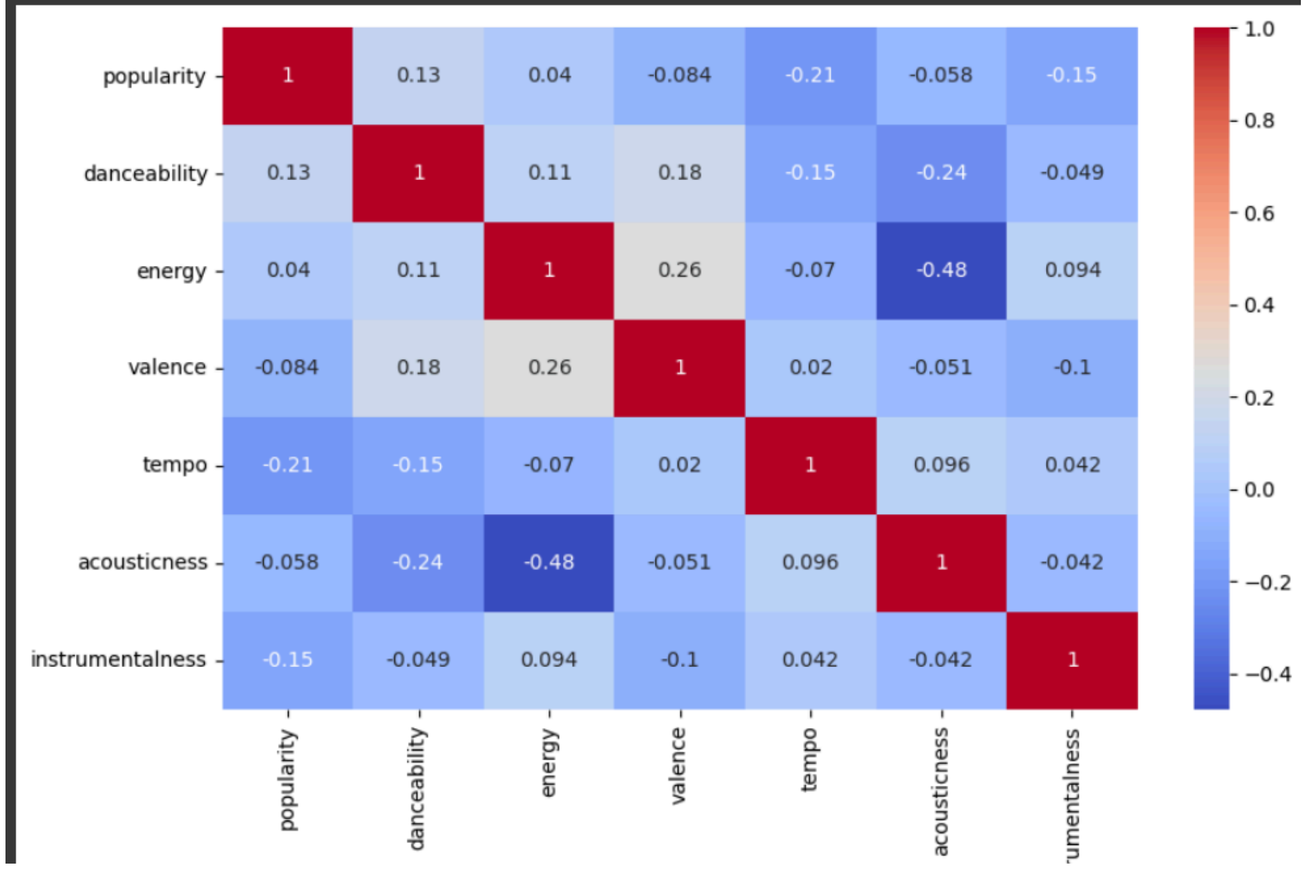
```
(163, 9)
```

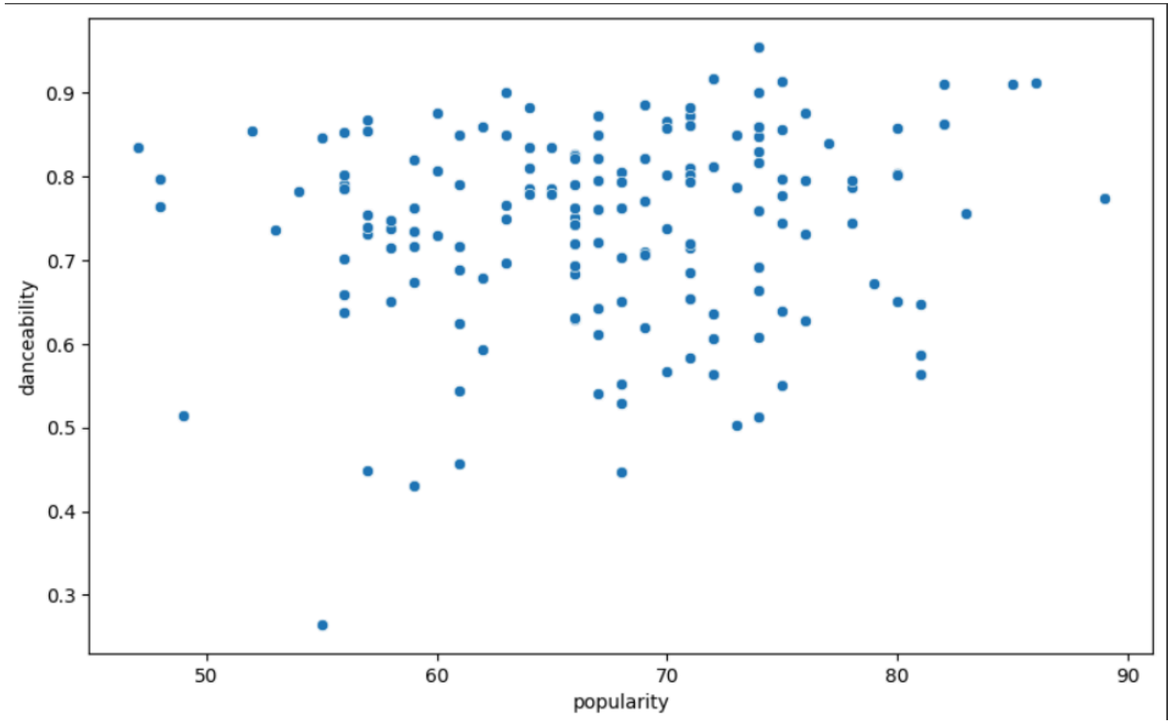
```
[21] all_songs_df.describe()
```

	popularity	danceability	energy	valence	tempo	acousticness	instrumentalness
count	163.000000	163.000000	163.000000	163.000000	163.000000	163.000000	163.000000
mean	67.398773	0.744405	0.669429	0.510813	122.515387	0.267137	0.009693
std	8.138872	0.116308	0.120830	0.218809	30.460094	0.221614	0.071352
min	47.000000	0.265000	0.331000	0.050800	76.993000	0.002890	0.000000
25%	61.000000	0.680500	0.597500	0.339500	96.013500	0.091050	0.000000
50%	67.000000	0.766000	0.677000	0.519000	119.935000	0.212000	0.000003

```
high_danceability = all_songs_df[all_songs_df['danceability'] > 0.6]['popularity']
low_danceability = all_songs_df[all_songs_df['danceability'] <= 0.6]['popularity']
```

```
plt.figure(figsize=(10, 6))
numeric_df = all_songs_df.select_dtypes(include=[float, int])
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm')
plt.show()
```



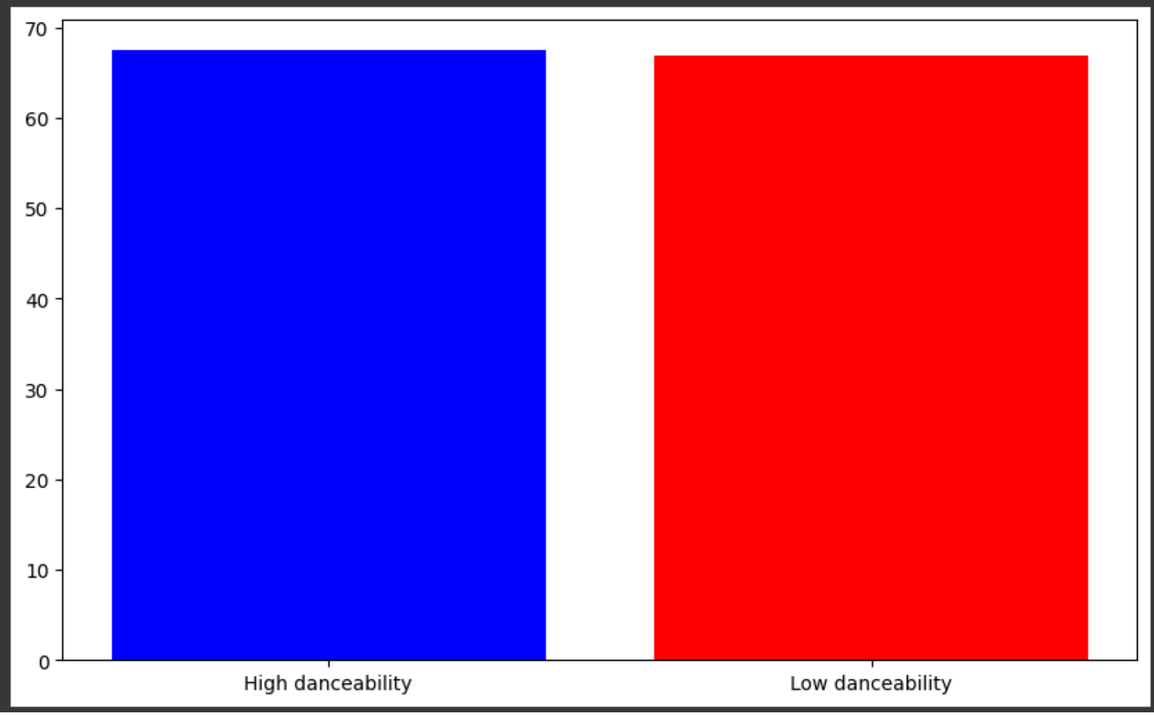


COMPARING THE AVERAGE

```
[29] high_danceability_avg_popularity = high_danceability.mean()
     low_danceability_avg_popularity = low_danceability.mean()
     print(f"Average Popularity for High Danceability: {high_danceability_avg_popularity}")
     print(f"Average Popularity for Low Danceability: {low_danceability_avg_popularity}")
```

```
Average Popularity for High Danceability: 67.45833333333333
Average Popularity for Low Danceability: 66.94736842105263
```

```
plt.figure(figsize=(10, 6))
plt.bar(['High danceability', 'Low danceability'], [high_danceability_avg_popularity, low_danceability_avg_popularity], color=['blue', 'red'])
plt.show()
```



RESULT

```
] t_stat, p_value = stats.ttest_ind(high_danceability, low_danceability)

  print(f"t-statistic: {t_stat}")
  print(f"p-value: {p_value}")

  if p_value < 0.05:
      print("There is a significant difference in popularity between high and low danceability.
  else:
      print("There is no significant difference in popularity between high and low danceability

  t-statistic: 0.25646921744847884
  p-value: 0.7979159744784049
  There is no significant difference in popularity between high and low danceability.
```

Fail to reject the H0

Conclusion

IN OUR DATA THERE ARE A LOT OF FACTORS THAT AFFECTS AN ARTISTS OR A TRACK POPULARITY

The first case was there is no statistically significant difference in popularity between tracks of different lengths based on this sample.

SECOND CASE IS THAT THERE IS EVIDENCE TO SUPPORT THE CLAIM THAT ARTISTS IN THE TOP 100, ONES WHO HAS MORE FOLLOWERS ALSO HAVE HIGHER POPULARITY

THIRD CASE IS WAS NO STATISTICALLY SIGNIFICANT DIFFERENCE BETWEEN THE SONGS WITH HIGH DANCEABILITY AND LOW DANCEABILITY. We need more samples

References:

Welcome to Spotipy! — spotipy 2.0 documentation. (n.d.). Spotipy.readthedocs.io. https://spotipy.readthedocs.io/en/2.24.0/

Spotify. (n.d.). Web API | Spotify for Developers. Developer.spotify.com. https://developer.spotify.com/documentation/web-api