eda

June 20, 2024

0.1 Import Libraries

[117]:
```
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
import seaborn as sns
```

0.2 Pre-processing

0.2.1 Set file path

[118]:
```
file_path = 'spotify_songs.csv'
```

0.2.2 Load data

[119]:
```
df = pd.read_csv(file_path)
```

0.2.3 Overview of the raw dataframe

View some rows of the dataframe.

[120]:
```
df.head(4)
```

[120]:

	track_id	track_name	track_artist	track_popularity	track_album_id	track_album_name	track_album_release_date
0	6f807x0ima9a1j3VPbc7VN	I Don't Care (with Justin Bieber) - Loud Luxur…	Ed Sheeran	66	2oCs0DGTsRO98Gh5ZSl2Cx	I Don't Care (with Justin Bieber) [Loud Luxury…	2019-06-14
1	0r7CVbZTWZgbTCYdfa2P31	Memories - Dillon Francis Remix	Maroon 5	67	63rPSO264uRjW1X5E6cWv6		
2	1z1Hg7Vb0AhHDiEmnDE79l	All the Time - Don Diablo Remix	Zara Larsson	70	1HoSmj2eLcsrR0vE9gThr4		
3	75FpbthrwQmzHlBJLuGdC7	Call You Mine - Keanu Silva Remix	The Chainsmokers	60	1nqYsOef1yKKuGOVchbsk6		

```
1                    Memories (Dillon Francis Remix)              2019-12-13
2                    All the Time (Don Diablo Remix)              2019-07-05
3                        Call You Mine - The Remixes              2019-07-19

  playlist_name            playlist_id playlist_genre  … key  loudness  \
0     Pop Remix  37i9dQZF1DXcZDD7cfEKhW            pop  …   6    -2.634
1     Pop Remix  37i9dQZF1DXcZDD7cfEKhW            pop  …  11    -4.969
2     Pop Remix  37i9dQZF1DXcZDD7cfEKhW            pop  …   1    -3.432
3     Pop Remix  37i9dQZF1DXcZDD7cfEKhW            pop  …   7    -3.778

   mode  speechiness  acousticness  instrumentalness  liveness  valence  \
0     1       0.0583        0.1020          0.000000    0.0653    0.518
1     1       0.0373        0.0724          0.004210    0.3570    0.693
2     0       0.0742        0.0794          0.000023    0.1100    0.613
3     1       0.1020        0.0287          0.000009    0.2040    0.277

     tempo  duration_ms
0  122.036       194754
1   99.972       162600
2  124.008       176616
3  121.956       169093

[4 rows x 23 columns]
```

View the statistics of the dataframe.

[121]:
```
df.describe()
```

[121]:
```
       track_popularity  danceability        energy           key  \
count      32833.000000  32833.000000  32833.000000  32833.000000
mean          42.477081      0.654850      0.698619      5.374471
std           24.984074      0.145085      0.180910      3.611657
min            0.000000      0.000000      0.000175      0.000000
25%           24.000000      0.563000      0.581000      2.000000
50%           45.000000      0.672000      0.721000      6.000000
75%           62.000000      0.761000      0.840000      9.000000
max          100.000000      0.983000      1.000000     11.000000

           loudness          mode   speechiness  acousticness  \
count  32833.000000  32833.000000  32833.000000  32833.000000
mean      -6.719499      0.565711      0.107068      0.175334
std        2.988436      0.495671      0.101314      0.219633
min      -46.448000      0.000000      0.000000      0.000000
25%       -8.171000      0.000000      0.041000      0.015100
50%       -6.166000      1.000000      0.062500      0.080400
75%       -4.645000      1.000000      0.132000      0.255000
max        1.275000      1.000000      0.918000      0.994000
```

```
       instrumentalness      liveness       valence         tempo  \
count      32833.000000  32833.000000  32833.000000  32833.000000
mean           0.084747      0.190176      0.510561    120.881132
std            0.224230      0.154317      0.233146     26.903624
min            0.000000      0.000000      0.000000      0.000000
25%            0.000000      0.092700      0.331000     99.960000
50%            0.000016      0.127000      0.512000    121.984000
75%            0.004830      0.248000      0.693000    133.918000
max            0.994000      0.996000      0.991000    239.440000

         duration_ms
count   32833.000000
mean   225799.811622
std     59834.006182
min      4000.000000
25%    187819.000000
50%    216000.000000
75%    253585.000000
max    517810.000000
```

[122]:
```
df.describe(include='object')
```

[122]:
```
                     track_id track_name   track_artist  \
count                   32833      32828          32828
unique                  28356      23449          10692
top     7BKLCZ1jbUBVqRi2FVlTVw     Poison  Martin Garrix
freq                       10         22            161

                track_album_id track_album_name track_album_release_date  \
count                    32833            32828                    32833
unique                   22545            19743                     4530
top     5L1xcowSxwzFUSJzvyMp48    Greatest Hits               2020-01-10
freq                        42              139                      270

          playlist_name             playlist_id playlist_genre  \
count             32833                   32833          32833
unique              449                     471              6
top     Indie Poptimism  4JkkvMpVl4lSioqQjeAL0q            edm
freq                308                     247           6043

               playlist_subgenre
count                      32833
unique                        24
top     progressive electro house
freq                       1809
```

It can be seen that there are some missing values in the dataframe.

View the data types of the dataframe.

[123]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32833 entries, 0 to 32832
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   track_id                  32833 non-null  object
 1   track_name                32828 non-null  object
 2   track_artist              32828 non-null  object
 3   track_popularity          32833 non-null  int64
 4   track_album_id            32833 non-null  object
 5   track_album_name          32828 non-null  object
 6   track_album_release_date  32833 non-null  object
 7   playlist_name             32833 non-null  object
 8   playlist_id               32833 non-null  object
 9   playlist_genre            32833 non-null  object
 10  playlist_subgenre         32833 non-null  object
 11  danceability              32833 non-null  float64
 12  energy                    32833 non-null  float64
 13  key                       32833 non-null  int64
 14  loudness                  32833 non-null  float64
 15  mode                      32833 non-null  int64
 16  speechiness               32833 non-null  float64
 17  acousticness              32833 non-null  float64
 18  instrumentalness          32833 non-null  float64
 19  liveness                  32833 non-null  float64
 20  valence                   32833 non-null  float64
 21  tempo                     32833 non-null  float64
 22  duration_ms               32833 non-null  int64
dtypes: float64(9), int64(4), object(10)
memory usage: 5.8+ MB
```

0.3 Data cleaning and wrangling

0.3.1 Check for missing values

[124]:
```
df.isnull().sum()
```

[124]:
```
track_id                  0
track_name                5
track_artist              5
track_popularity          0
track_album_id            0
track_album_name          5
```

```
track_album_release_date    0
playlist_name               0
playlist_id                 0
playlist_genre              0
playlist_subgenre           0
danceability                0
energy                      0
key                         0
loudness                    0
mode                        0
speechiness                 0
acousticness                0
instrumentalness            0
liveness                    0
valence                     0
tempo                       0
duration_ms                 0
dtype: int64
```

There are 3 columns with missing values. The missing values are in the following columns 'track_name', 'track_artist', and 'track_album_name'.

0.3.2 Handle Missing Values

[125]:
```
df[df.isnull().any(axis=1)]
```

[125]:

	track_id	track_name	track_artist	track_popularity	\
8151	69gRFGOWY9OMpFJgFol1u0	NaN	NaN	0	
9282	5cjecvXOCmC9gKOLaf5EMQ	NaN	NaN	0	
9283	5TTzhRSWQS4Yu8xTgAuq6D	NaN	NaN	0	
19568	3VKFip3OdAvv4OfNTgFWeQ	NaN	NaN	0	
19811	69gRFGOWY9OMpFJgFol1u0	NaN	NaN	0	

	track_album_id	track_album_name	track_album_release_date	\
8151	717UG2du6utFe7CdmpuUe3	NaN	2012-01-05	
9282	3luHJEPw434tvNbme3SP8M	NaN	2017-12-01	
9283	3luHJEPw434tvNbme3SP8M	NaN	2017-12-01	
19568	717UG2du6utFe7CdmpuUe3	NaN	2012-01-05	
19811	717UG2du6utFe7CdmpuUe3	NaN	2012-01-05	

	playlist_name	playlist_id	playlist_genre	…	key	\
8151	HIP&HOP	5DyJsJZOpMJh34WvUrQzMV	rap	…	6	
9282	GANGSTA Rap	5GA8GDo7RQC3JEanT81B3g	rap	…	11	
9283	GANGSTA Rap	5GA8GDo7RQC3JEanT81B3g	rap	…	10	
19568	Reggaeton viejito	0si5tw70PIgPkY1Eva6V8f	latin	…	11	
19811	latin hip hop	3nH8aytdqNeRbcRCg3dw9q	latin	…	6	

```
       loudness  mode  speechiness  acousticness  instrumentalness  liveness  \
8151     -7.635     1       0.1760        0.0410           0.00000    0.1160
9282     -5.364     0       0.3190        0.0534           0.00000    0.5530
9283     -5.907     0       0.3070        0.0963           0.00000    0.0888
19568    -6.075     0       0.0366        0.0606           0.00653    0.1030
19811    -7.635     1       0.1760        0.0410           0.00000    0.1160

       valence    tempo  duration_ms
8151     0.649   95.999       282707
9282     0.191  146.153       202235
9283     0.505   86.839       206465
19568    0.726   97.017       252773
19811    0.649   95.999       282707

[5 rows x 23 columns]
```

In the dataframe above, there are missing concurrent values in 'track_name', 'track_artist', and 'track_album_name'. Therefore, we will drop the rows with the missing values above.

[126]:
```
df.dropna(inplace=True)
```

Then, we will check if there are still missing values in the dataframe.

[127]:
```
df.isnull().sum()
```

[127]:
```
track_id                  0
track_name                0
track_artist              0
track_popularity          0
track_album_id            0
track_album_name          0
track_album_release_date  0
playlist_name             0
playlist_id               0
playlist_genre            0
playlist_subgenre         0
danceability              0
energy                    0
key                       0
loudness                  0
mode                      0
speechiness               0
acousticness              0
instrumentalness          0
liveness                  0
valence                   0
tempo                     0
duration_ms               0
```

```
dtype: int64
```

0.3.3 Convert data types

Because the data types of the columns are normalized, we don't need to convert any data types.

0.3.4 Check the consistency of the dataframe

Check for duplicated data

[128]:
```
df.duplicated().sum()
```

[128]:
```
0
```

There is no duplicated data in the dataframe.

0.3.5 Rename the specific column

Because the column names are normalized, we don't need to rename any columns.

0.3.6 Remove the specific column

We will remove the track_id, track_album_id, and playlist_id columns because they are not needed for the analysis.

[129]:
```
df.drop(['track_id', 'track_album_id', 'playlist_id'], axis=1, inplace=True)
```

0.3.7 Review the cleaned dataframe

View the first few rows of the cleaned dataframe.

[130]:
```
df.head()
```

[130]:

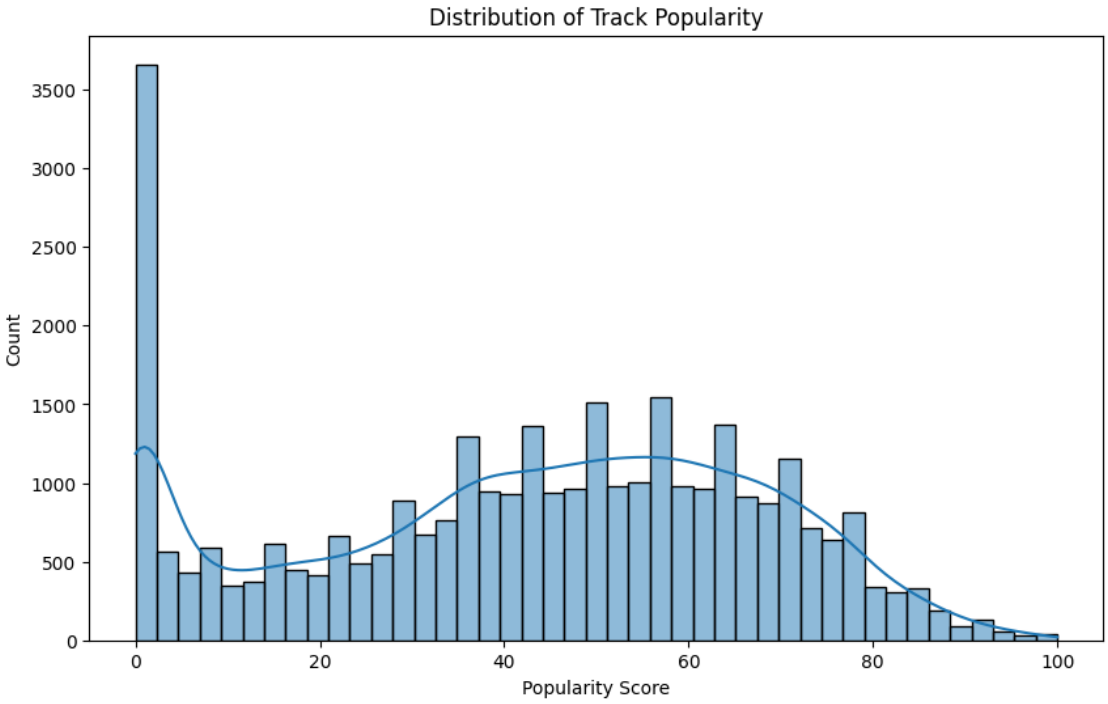
	track_name	track_artist	track_popularity	track_album_name	track_album_release_date	playlist_name	playlist_genre	playlist_subgenre
0	I Don't Care (with Justin Bieber) - Loud Luxur…	Ed Sheeran	66	I Don't Care (with Justin Bieber) [Loud Luxury…	2019-06-14	Pop Remix	pop	dance pop
1	Memories - Dillon Francis Remix	Maroon 5	67	Memories (Dillon Francis Remix)	2019-12-13	Pop Remix	pop	dance pop
2	All the Time - Don Diablo Remix	Zara Larsson	70	All the Time (Don Diablo Remix)				
3	Call You Mine - Keanu Silva Remix	The Chainsmokers	60	Call You Mine - The Remixes				
4	Someone You Loved - Future Humans Remix	Lewis Capaldi	69	Someone You Loved (Future Humans Remix)				

```
2               2019-07-05     Pop Remix            pop         dance pop
3               2019-07-19     Pop Remix            pop         dance pop
4               2019-03-05     Pop Remix            pop         dance pop

   danceability  energy  key  loudness  mode  speechiness  acousticness  \
0         0.748   0.916    6    -2.634     1       0.0583        0.1020
1         0.726   0.815   11    -4.969     1       0.0373        0.0724
2         0.675   0.931    1    -3.432     0       0.0742        0.0794
3         0.718   0.930    7    -3.778     1       0.1020        0.0287
4         0.650   0.833    1    -4.672     1       0.0359        0.0803

   instrumentalness  liveness  valence    tempo  duration_ms
0          0.000000    0.0653    0.518  122.036       194754
1          0.004210    0.3570    0.693   99.972       162600
2          0.000023    0.1100    0.613  124.008       176616
3          0.000009    0.2040    0.277  121.956       169093
4          0.000000    0.0833    0.725  123.976       189052
```

0.4 Data visualization

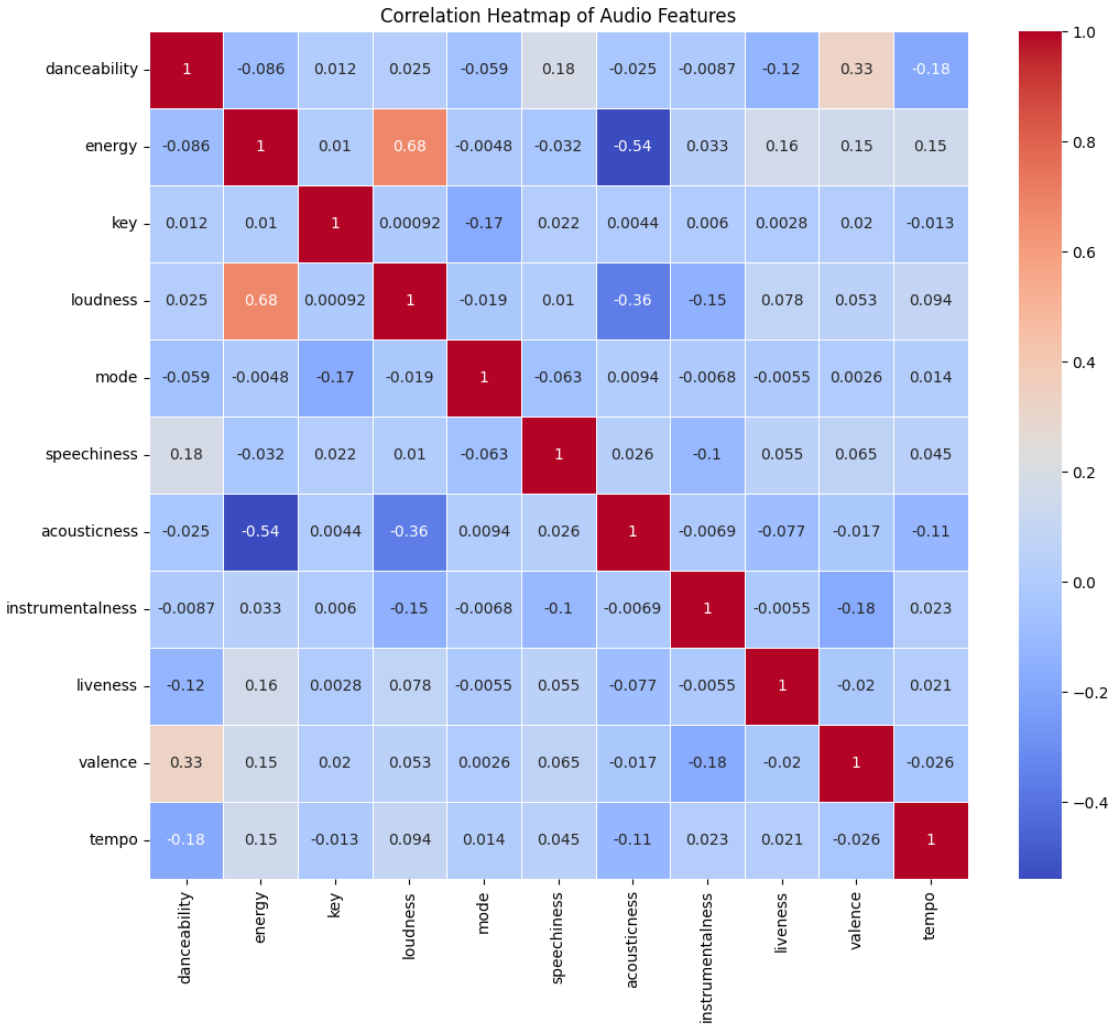
0.4.1 1. Distribution of Track Popularity

[131]:
```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='track_popularity', kde=True)
plt.title('Distribution of Track Popularity')
plt.xlabel('Popularity Score')
plt.ylabel('Count')
plt.show()
```



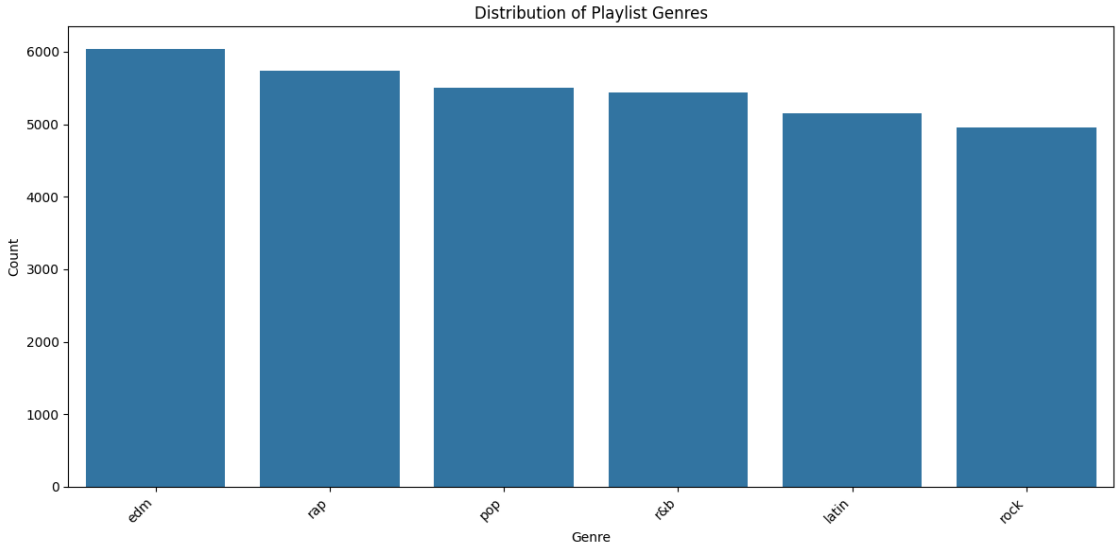
0.4.2 2. Correlation Heatmap of Audio Features

[132]:
```
audio_features = ['danceability', 'energy', 'key', 'loudness', 'mode',␣
 ↪'speechiness',
                  'acousticness', 'instrumentalness', 'liveness', 'valence',␣
 ↪'tempo']

plt.figure(figsize=(12, 10))
sns.heatmap(df[audio_features].corr(), annot=True, cmap='coolwarm',␣
 ↪linewidths=0.5)
plt.title('Correlation Heatmap of Audio Features')
plt.show()
```



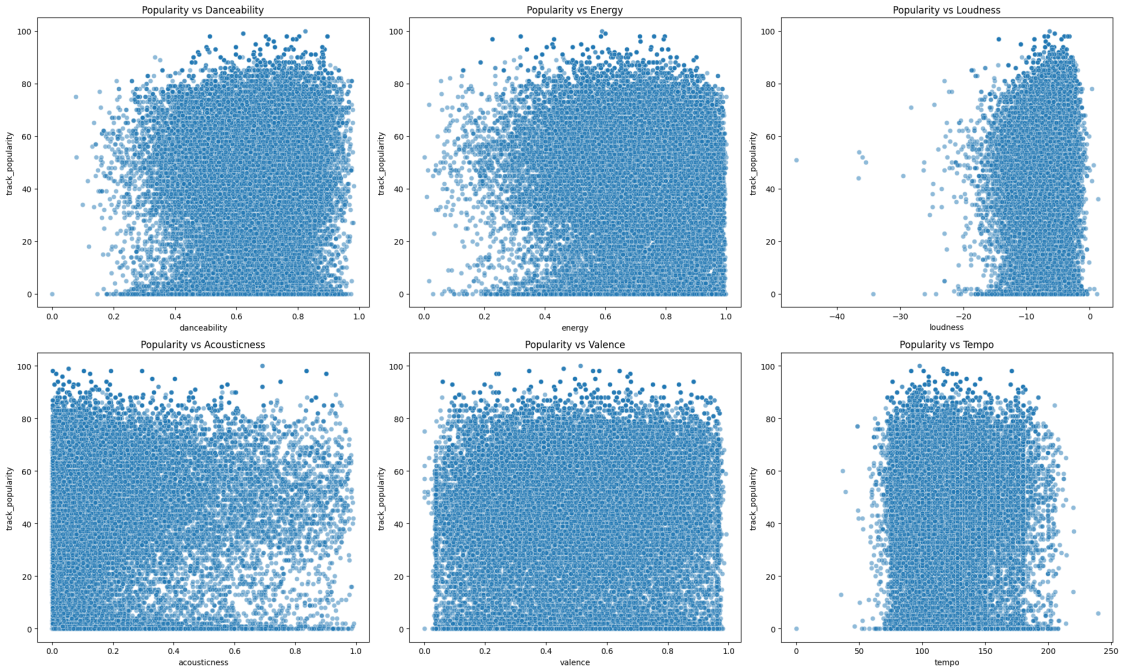
0.4.3 3. Genre Distribution

[133]:
```
plt.figure(figsize=(12, 6))
genre_counts = df['playlist_genre'].value_counts()
sns.barplot(x=genre_counts.index, y=genre_counts.values)
plt.title('Distribution of Playlist Genres')
plt.xlabel('Genre')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



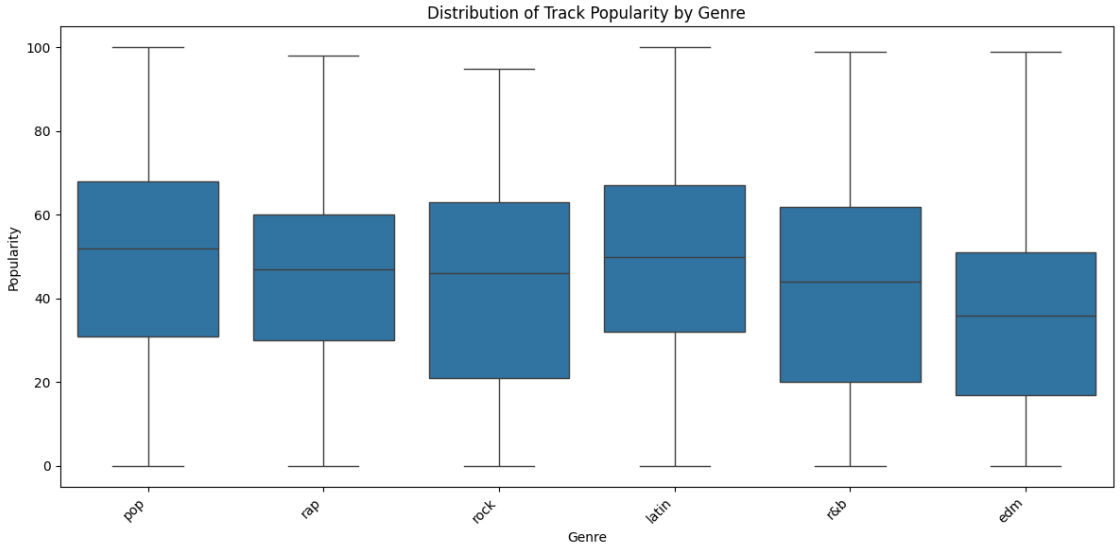
0.4.4 4. Popularity vs. Audio Features

[134]:
```
fig, axs = plt.subplots(2, 3, figsize=(20, 12))
axs = axs.flatten()

features_to_plot = ['danceability', 'energy', 'loudness', 'acousticness',␣
 ↪'valence', 'tempo']

for i, feature in enumerate(features_to_plot):
    sns.scatterplot(data=df, x=feature, y='track_popularity', alpha=0.5,␣
 ↪ax=axs[i])
    axs[i].set_title(f'Popularity vs {feature.capitalize()}')

plt.tight_layout()
plt.show()
```



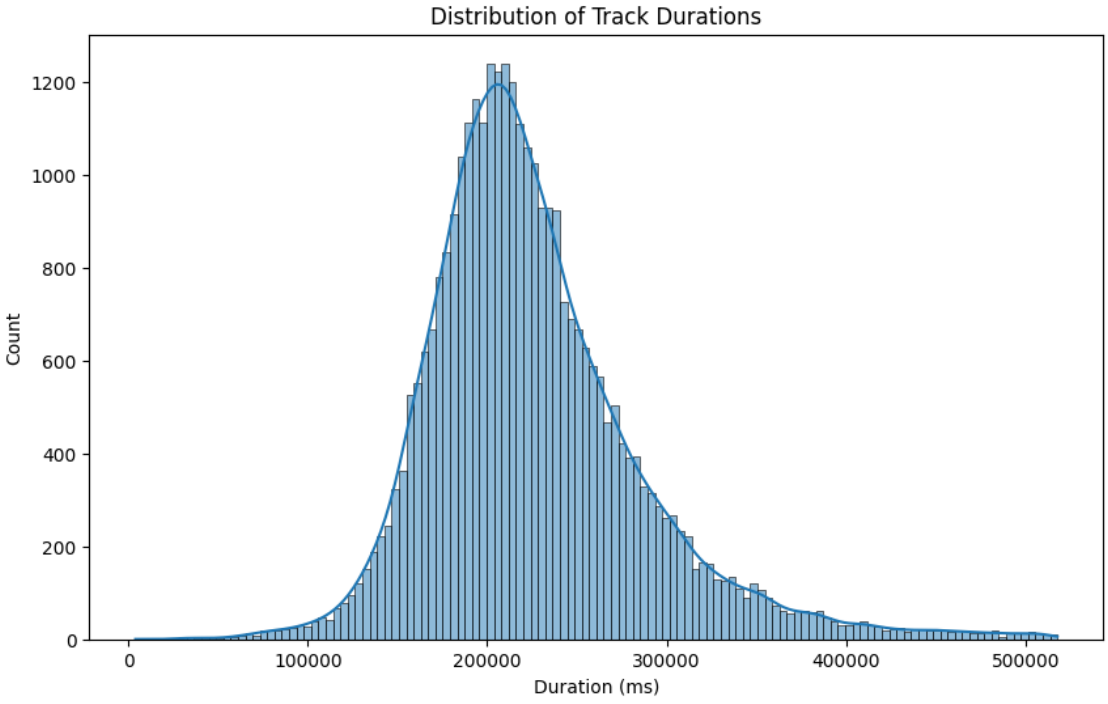
0.4.5 5. Box Plot of Popularity by Genre

[135]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x='playlist_genre', y='track_popularity')
plt.title('Distribution of Track Popularity by Genre')
plt.xlabel('Genre')
plt.ylabel('Popularity')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



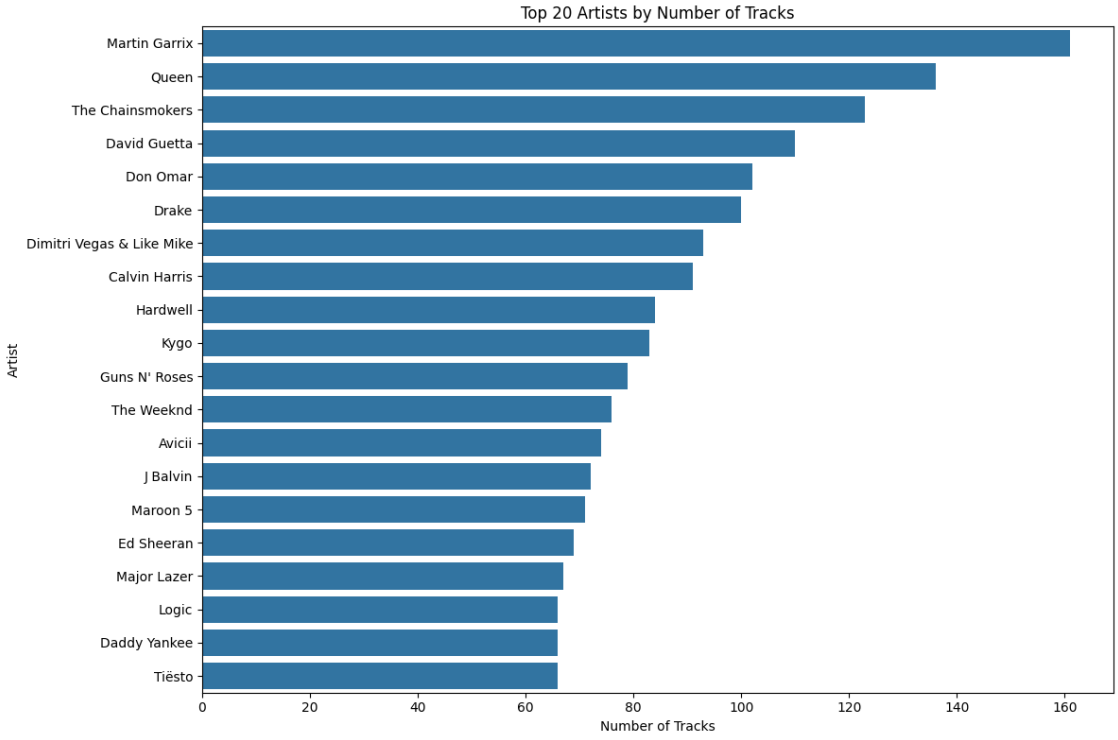
0.4.6 6. Duration Distribution

[136]:
```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='duration_ms', kde=True)
plt.title('Distribution of Track Durations')
plt.xlabel('Duration (ms)')
plt.ylabel('Count')
plt.show()
```



0.4.7 7. Top 20 Artists by Track Count

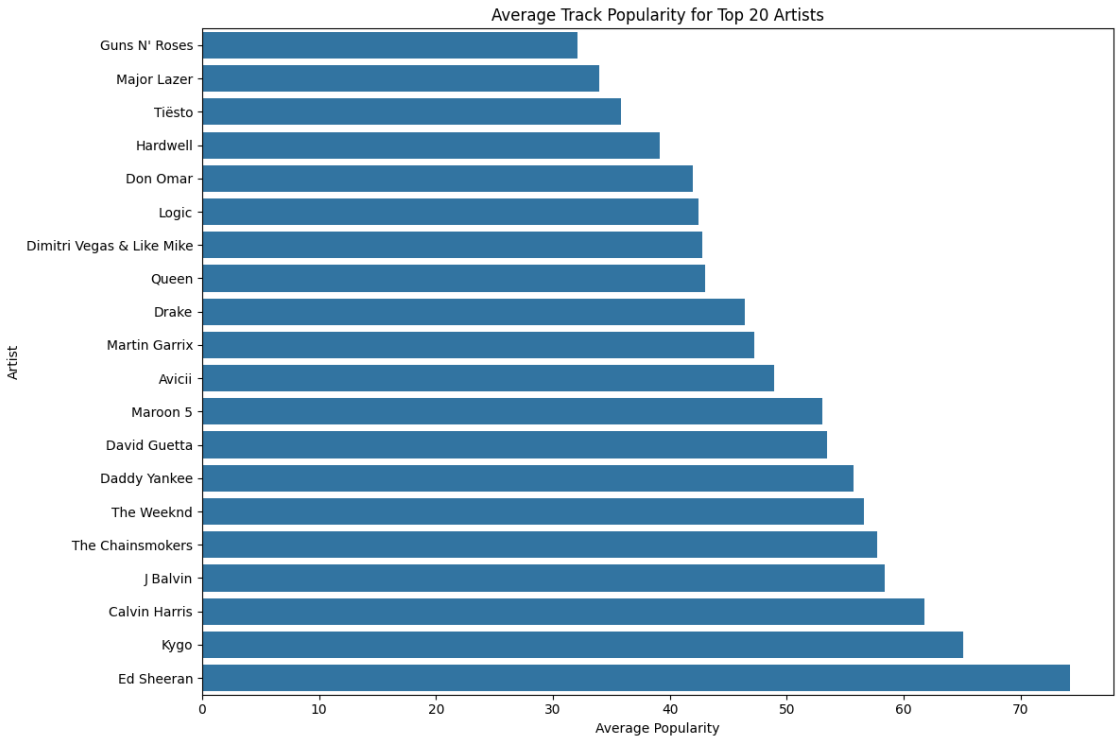
[137]:
```
plt.figure(figsize=(12, 8))
top_20_artists = df['track_artist'].value_counts().nlargest(20)
sns.barplot(x=top_20_artists.values, y=top_20_artists.index)
plt.title('Top 20 Artists by Number of Tracks')
plt.xlabel('Number of Tracks')
plt.ylabel('Artist')
plt.tight_layout()
plt.show()
```



0.4.8 8. Average Popularity by Top 20 Artists

[138]:
```
top_20_artists = df['track_artist'].value_counts().nlargest(20).index
artist_popularity = df[df['track_artist'].isin(top_20_artists)].
 ↪groupby('track_artist')['track_popularity'].mean().
 ↪sort_values(ascending=True)

plt.figure(figsize=(12, 8))
sns.barplot(x=artist_popularity.values, y=artist_popularity.index)
```

```
plt.title('Average Track Popularity for Top 20 Artists')
plt.xlabel('Average Popularity')
plt.ylabel('Artist')
plt.tight_layout()
plt.show()
```



0.4.9 9. Artist Diversity Across Genres

[139]:
```
# Get top 10 artists by track count
top_10_artists = df['track_artist'].value_counts().nlargest(10).index

# Create a pivot table
genre_artist_pivot = df[df['track_artist'].isin(top_10_artists)].pivot_table(
    index='track_artist',
    columns='playlist_genre',
    values='track_name',
    aggfunc='count',
    fill_value=0
)

# Normalize the data
genre_artist_pivot_norm = genre_artist_pivot.div(genre_artist_pivot.
 ↪sum(axis=1), axis=0)
```

```
# Plot
ax = genre_artist_pivot_norm.plot(kind='barh', stacked=True, figsize=(12, 8))
plt.title('Genre Distribution for Top 10 Artists')
plt.xlabel('Proportion of Tracks')
plt.ylabel('Artist')
plt.legend(title='Genre', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```