Data analysis and visualizations

visualization is the process of putting data into a chart, graph, or other visual format that helps inform analysis and interpretation

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
```

In [3]:
```
df=pd.read_csv(r"C:\Users\bhava\Downloads\spotify dataset.csv")
df
```

Out[3]:

	track_id	track_name	track_artist	track_popularity	track_album_id	track_album_name	tra
0	6f807x0ima9a1j3VPbc7VN	I Don't Care (with Justin Bieber) - Loud Luxur...	Ed Sheeran	66	2oCs0DGTsRO98Gh5ZSl2Cx	I Don't Care (with Justin Bieber) [Loud Luxury...	
1	0r7CVbZTWZgbTCYdfa2P31	Memories - Dillon Francis Remix	Maroon 5	67	63rPSO264uRjW1X5E6cWv6	Memories (Dillon Francis Remix)	
2	1z1Hg7Vb0AhHDiEmnDE79l	All the Time - Don Diablo Remix	Zara Larsson	70	1HoSmj2eLcsrR0vE9gThr4	All the Time (Don Diablo Remix)	
3	75FpbthrwQmzHlBJLuGdC7	Call You Mine - Keanu Silva Remix	The Chainsmokers	60	1nqYsOef1yKKuGOVchbsk6	Call You Mine - The Remixes	
4	1e8PAfcKUYoKkxPhrHqw4x	Someone You Loved - Future Humans Remix	Lewis Capaldi	69	7m7vv9wlQ4i0LFuJiE2zsQ	Someone You Loved (Future Humans Remix)	
...	...	...	...	...	...	...	
32828	7bxnKAamR3snQ1VGLuVfC1	City Of Lights - Official Radio Edit	Lush & Simon	42	2azRoBBWEEEYhqV6sb7JrT	City Of Lights (Vocal Mix)	
32829	5Aevni09Em4575077nkWHz	Closer - Sultan & Ned Shepard Remix	Tegan and Sara	20	6kD6KLxj7s8eCE3ABvAyf5	Closer Remixed	
32830	7ImMqPP3Q1yfUHvsdn7wEo	Sweet Surrender - Radio Edit	Starkillers	14	0ltWNSY9JgxoIZO4VzuCa6	Sweet Surrender (Radio Edit)	
32831	2m69mhnfQ1Oq6lGtXuYhgX	Only For You - Maor Levi Remix	Mat Zo	15	1fGrOkHnHJcStl14zNx8Jy	Only For You (Remixes)	
32832	29zWqhca3zt5NsckZqDf6c	Typhoon - Original Mix	Julian Calor	27	0X3mUOm6MhxR7PzxG95rAo	Typhoon/Storm	

32833 rows × 23 columns

In [5]:
```
df.info()
```

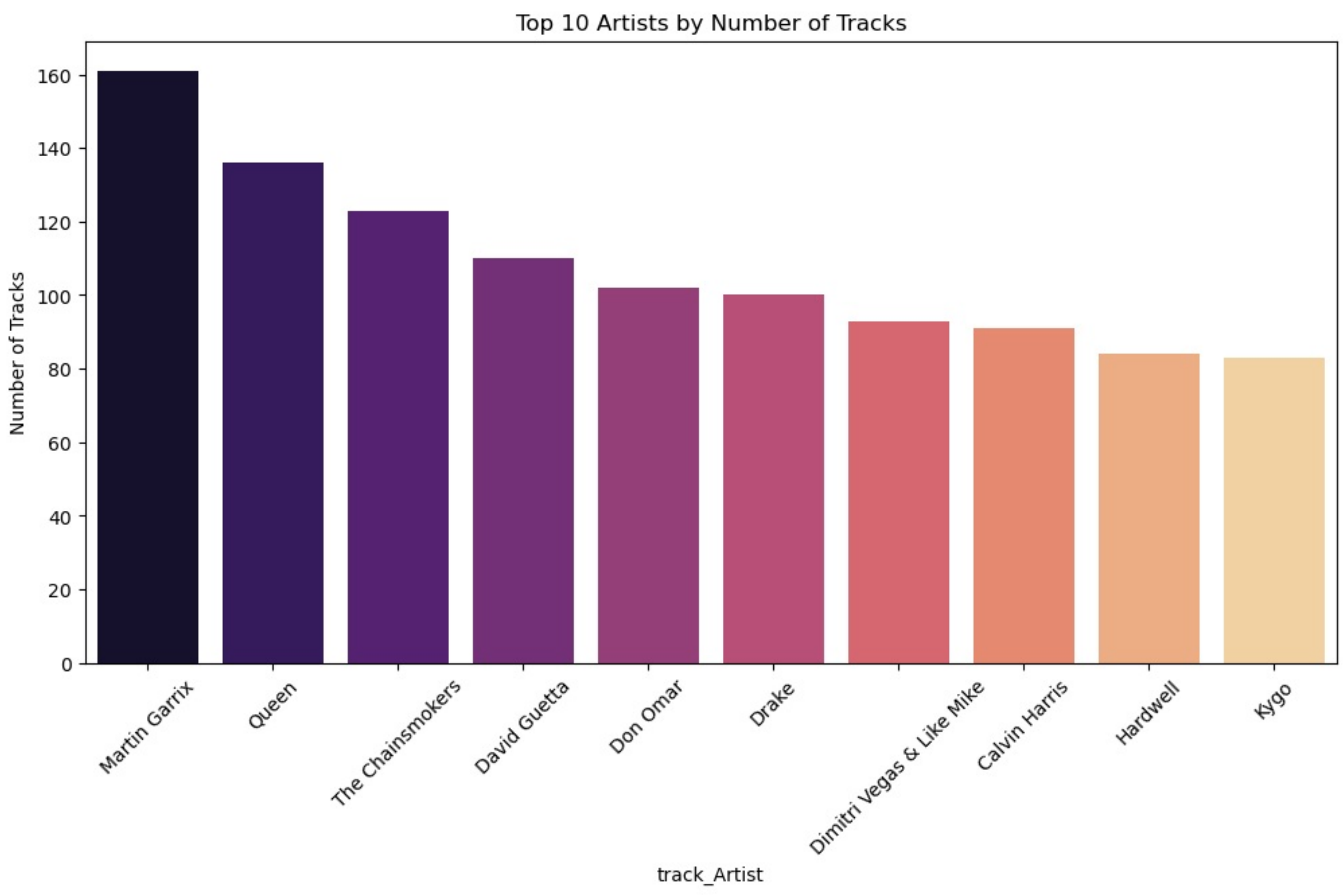
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32833 entries, 0 to 32832
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   track_id                  32833 non-null  object 
 1   track_name                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
```

In [7]:
```
df.describe()
```

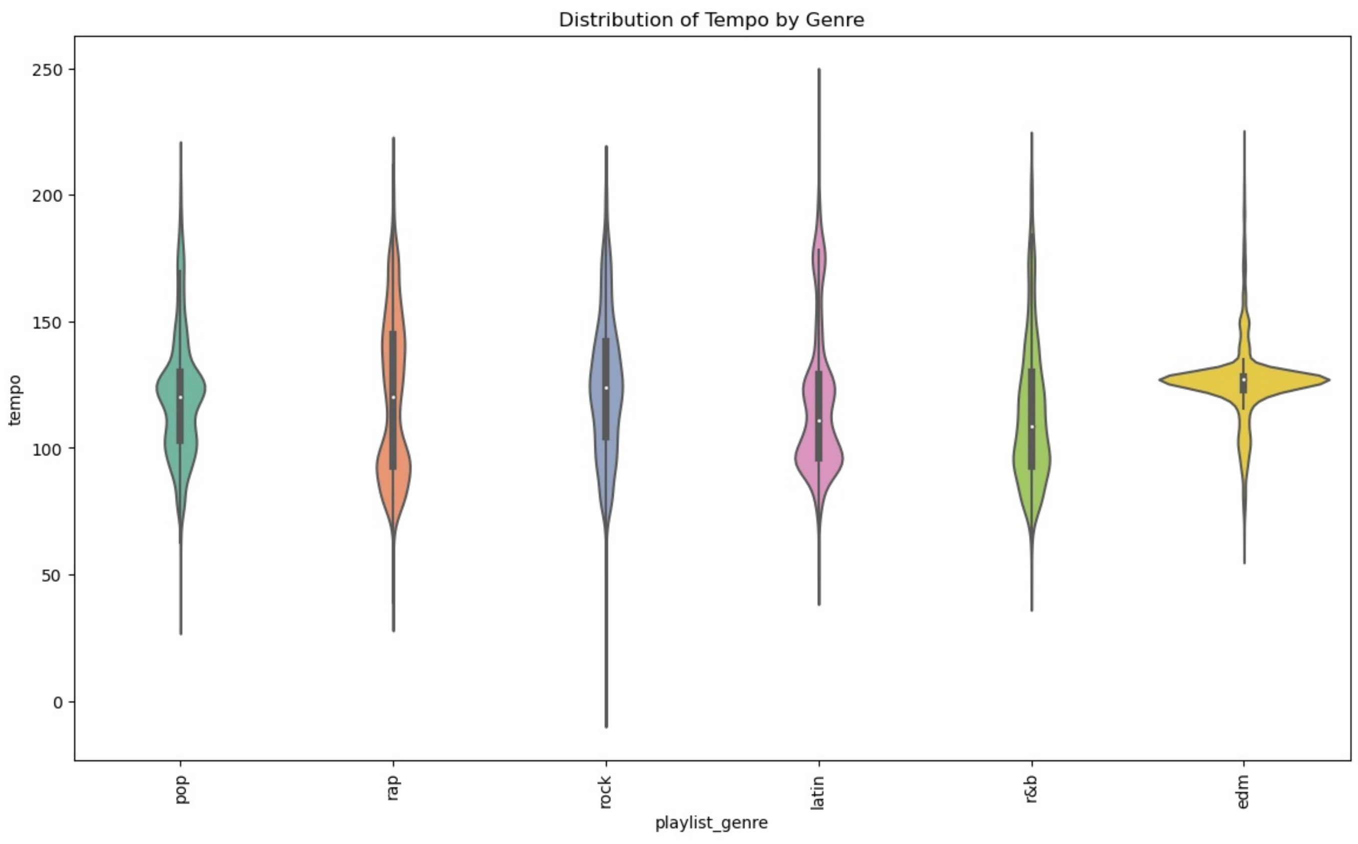
Out[7]:

	track_popularity	danceability	energy	key	loudness	mode	speechiness	acousticness	instru
count	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32
mean	42.477081	0.654850	0.698619	5.374471	-6.719499	0.565711	0.107068	0.175334	
std	24.984074	0.145085	0.180910	3.611657	2.988436	0.495671	0.101314	0.219633	
min	0.000000	0.000000	0.000175	0.000000	-46.448000	0.000000	0.000000	0.000000	
25%	24.000000	0.563000	0.581000	2.000000	-8.171000	0.000000	0.041000	0.015100	
50%	45.000000	0.672000	0.721000	6.000000	-6.166000	1.000000	0.062500	0.080400	
75%	62.000000	0.761000	0.840000	9.000000	-4.645000	1.000000	0.132000	0.255000	
max	100.000000	0.983000	1.000000	11.000000	1.275000	1.000000	0.918000	0.994000	

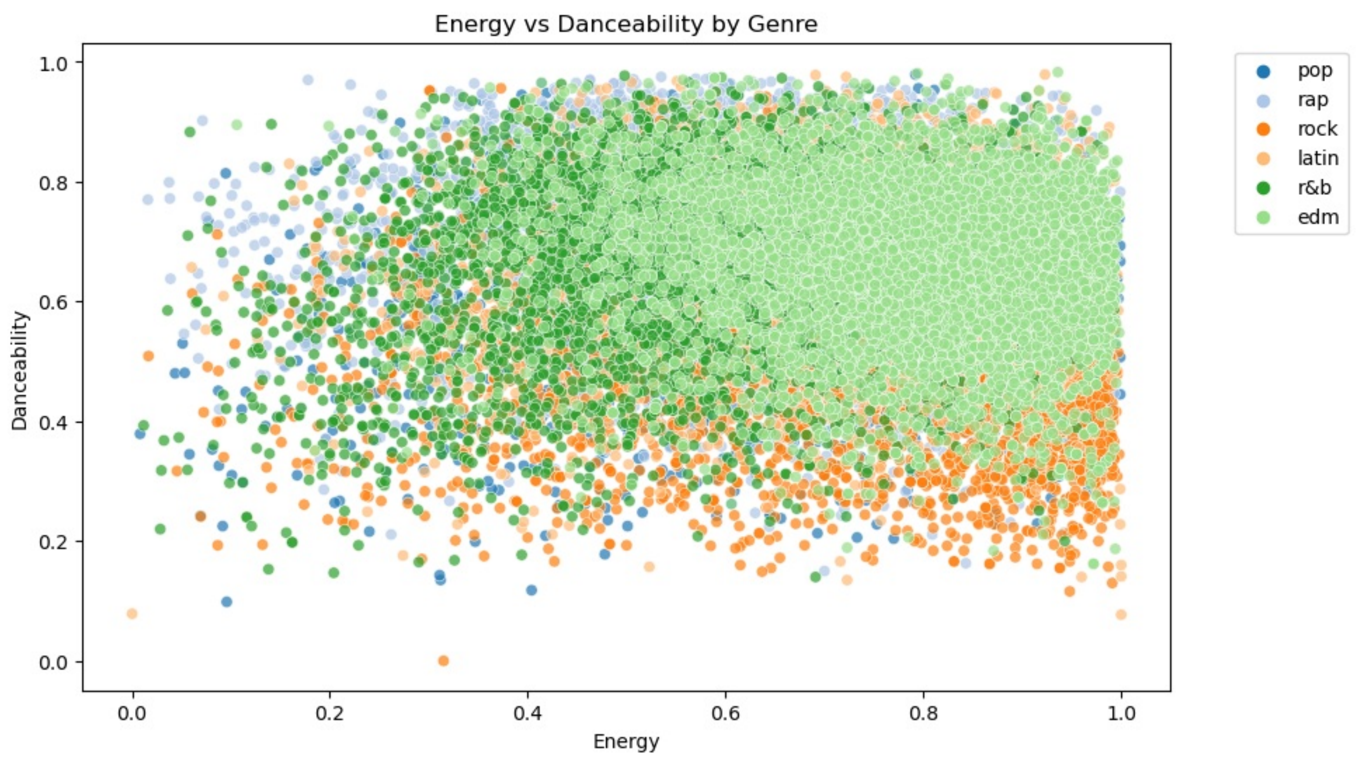
In [22]:
```
plt.figure(figsize=(12, 6))
top_artists = df['track_artist'].value_counts().head(10)
sns.barplot(x=top_artists.index, y=top_artists.values, palette='magma')
plt.title('Top 10 Artists by Number of Tracks')
plt.xlabel('track_Artist')
plt.ylabel('Number of Tracks')
plt.xticks(rotation=45)
plt.show()
```



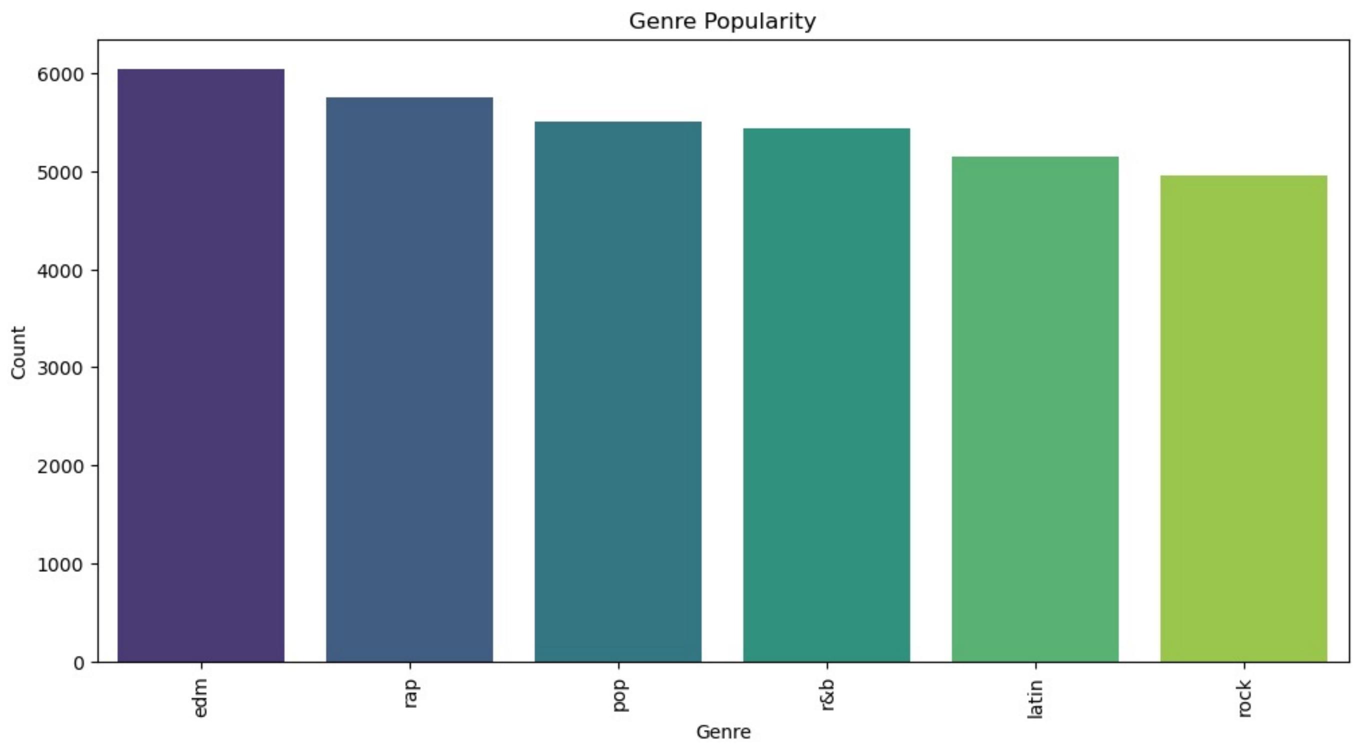
```
In [28]: plt.figure(figsize=(14, 8))
         sns.violinplot(x='playlist_genre', y='tempo', data=df, palette='Set2')
         plt.title('Distribution of Tempo by Genre')
         plt.xticks(rotation=90)
         plt.show()
```



```
In [38]: plt.figure(figsize=(10, 6))
         sns.scatterplot(data=df, x='energy', y='danceability', hue='playlist_genre', palette='tab20', alpha=0.7)
         plt.title('Energy vs Danceability by Genre')
         plt.xlabel('Energy')
         plt.ylabel('Danceability')
         plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
         plt.show()
```

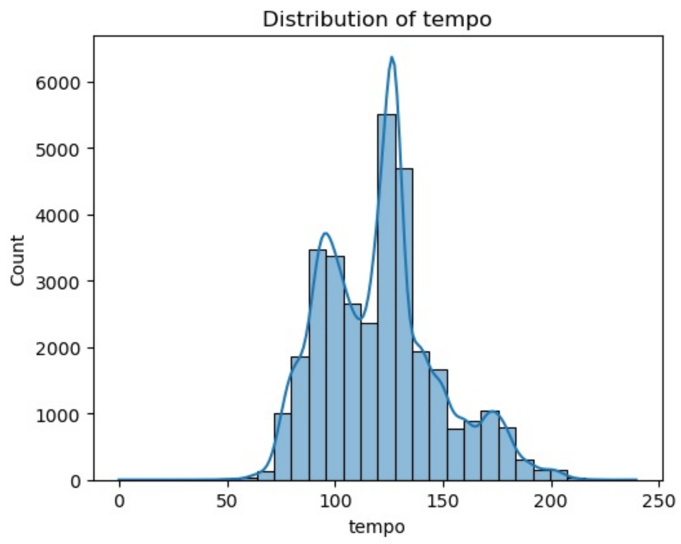


```
In [42]: plt.figure(figsize=(12, 6))
         genre_count = df['playlist_genre'].value_counts()
         sns.barplot(x=genre_count.index, y=genre_count.values, palette='viridis')
         plt.title('Genre Popularity')
         plt.xlabel('Genre')
         plt.ylabel('Count')
         plt.xticks(rotation=90)
         plt.show()
```



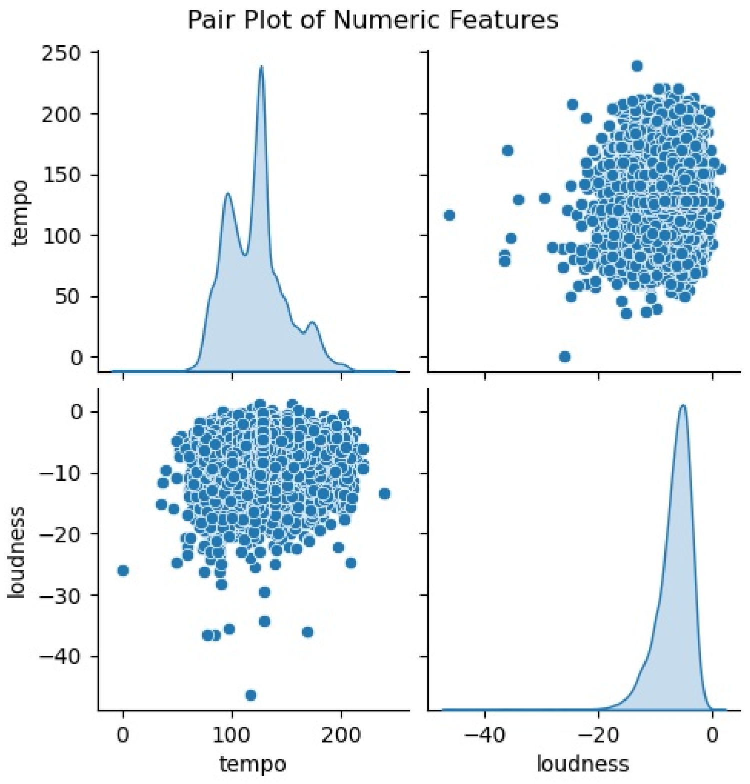
```
In [58]: plt.figure(figsize=(16, 12))
         for i, feature in enumerate(numeric_features):
             plt.subplot(3, 3, i+1)
             sns.histplot(df[feature], kde=True, bins=30)
             plt.title(f'Distribution of {feature}')
         plt.tight_layout()
         plt.show()
```

```
C:\Users\bhava\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\bhava\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



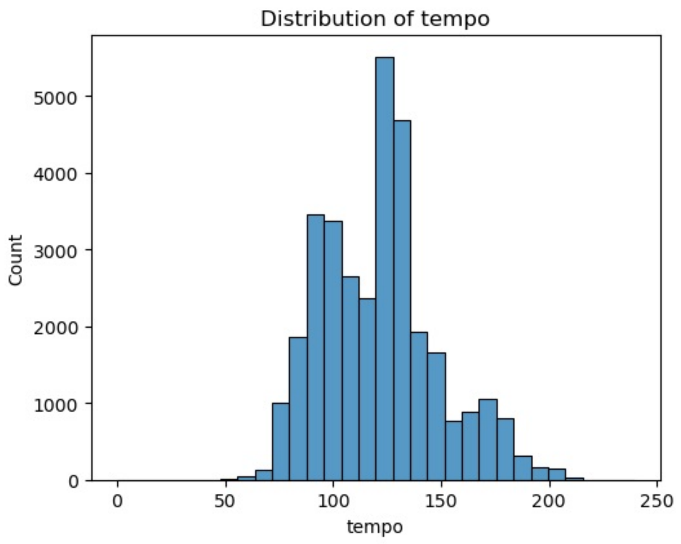
```
In [56]: sns.pairplot(df[numeric_features], diag_kind='kde')
         plt.suptitle('Pair Plot of Numeric Features', y=1.02)
         plt.show()
```

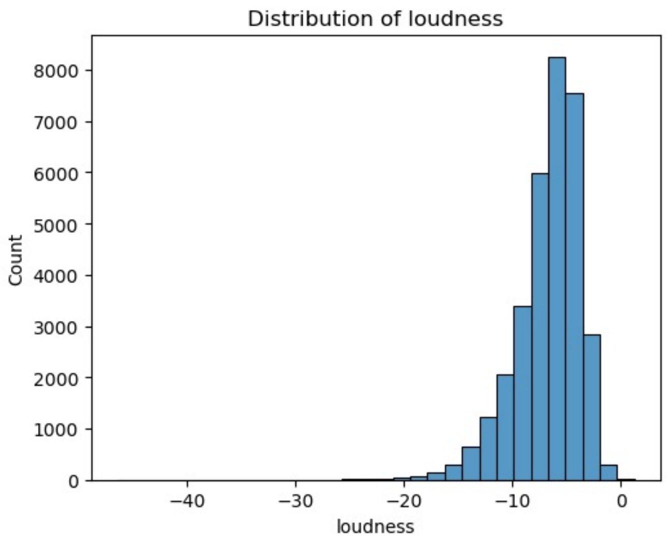
```
C:\Users\bhava\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\bhava\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



```
In [74]: plt.figure(figsize=(16, 12))
         for i, feature in enumerate(numeric_features):
             plt.subplot(3, 3, i+1)
             sns.histplot(df[feature],  bins=30)
             plt.title(f'Distribution of {feature}')
         plt.tight_layout()
         plt.show()
```

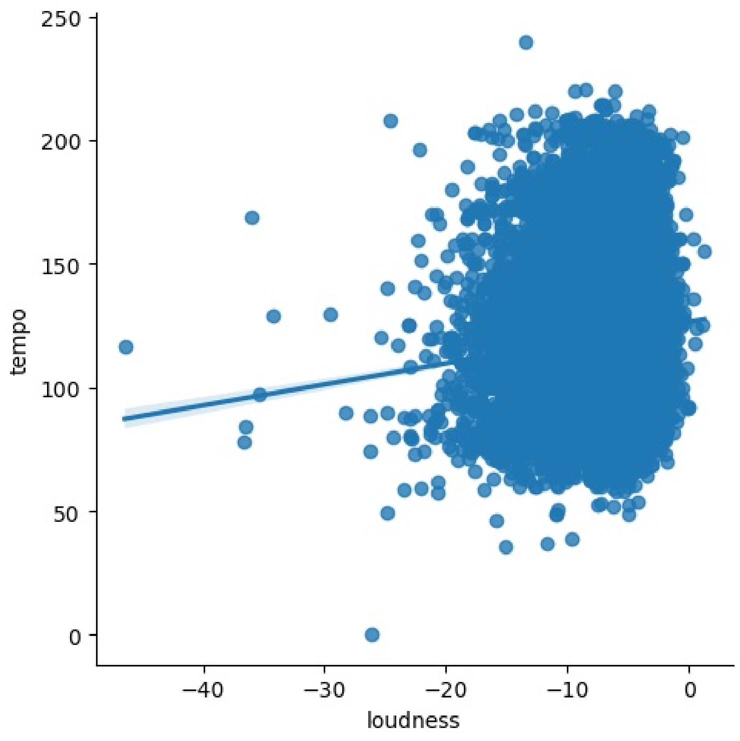
```
C:\Users\bhava\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\bhava\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```





```
In [80]: sns.lmplot(x="loudness",y="tempo",data=df)

Out[80]: <seaborn.axisgrid.FacetGrid at 0x1a020681810>
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js