Spotify

Description of the Dataset:

The dataset titled "Spotify Data: Popular Hip-hop Artists and Tracks" provides a curated collection of approximately 500 entries showcasing the vibrant realm of hip-hop music. These entries meticulously compile the most celebrated hip-hop tracks and artists, reflecting their significant influence on the genre's landscape. Each entry not only highlights the popularity and musical composition of the tracks but also underscores the creative prowess of the artists and their profound impact on global listeners.

In [1]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [11]:
```
#Load the import Labraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df=pd.read_csv("Spotify.csv")
df.head(5)
```

Out[3]:

	Artist	Track Name	Popularity	Duration (ms)	Track ID
0	Drake	Rich Baby Daddy (feat. Sexyy Red & SZA)	92	319191	1yeB8MUNeLo9Ek1UEpsyz6
1	Drake	One Dance	91	173986	1zi7xx7UVEFkmKfv06H8x0
2	Drake	IDGAF (feat. Yeat)	90	260111	2YSzYUF3jWqb9YP9VXmpjE
3	Drake	First Person Shooter (feat. J. Cole)	88	247444	7aqfrAY2p9BUSiupwk3svU
4	Drake	Jimmy Cooks (feat. 21 Savage)	88	218364	3F5CgOj3wFlRv51JsHbxhe

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Artist', 'Track Name', 'Popularity', 'Duration (ms)', 'Track ID'], dtype='object')
```

About Columns:

- Artist: The name of the artist, providing direct attribution to the creative mind behind the track.
- Track Name: The title of the track, encapsulating its identity and essence.
- Popularity: A numeric score reflecting the track's reception and appeal among Spotify listeners.
- Duration (ms): The track's length in milliseconds, detailing the temporal extent of the musical experience.
- Track ID: A unique identifier within Spotify's ecosystem, enabling direct access to the track for further

exploration.

1.Load the dataframe and ensure data quality by checking for missing values and duplicate rows. Handle missing values and remove duplicate rows if necessary.

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Artist          0
Track Name      0
Popularity      0
Duration (ms)   0
Track ID        0
dtype: int64
```

Their is no missing values in the DataFrame of spotify

In [6]:
```
df.shape
```

Out[6]:
```
(440, 5)
```

In [7]:
```
# check the unique Artist
len(df['Artist'].unique())
```

Out[7]:
```
115
```

In [8]:
```
# Check the duplicate rows
df.duplicated().sum()

# Remove duplicates rows from the DataFrame
df= df.drop_duplicates()
```

In [9]:
```
df.shape
```

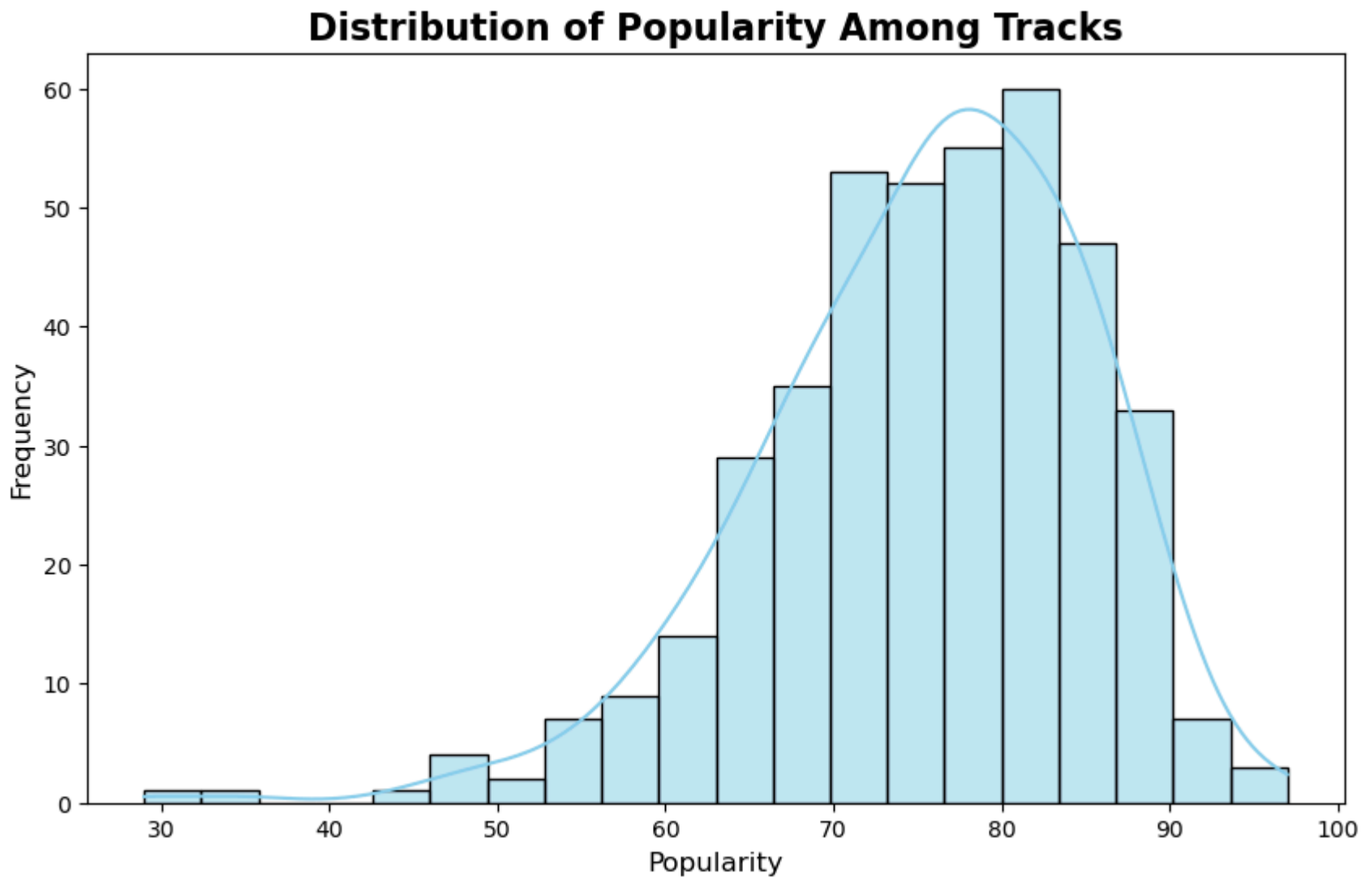
Out[9]:
```
(413, 5)
```

In [10]:
```
df.duplicated().sum()
```

Out[10]:
```
0
```

2.What is the distribution of popularity among the tracks in the dataset? Visualize it using a histogram

In [16]:
```
# Plot the distribution of popularity
plt.figure(figsize=(10, 6))
sns.histplot(df['Popularity'], bins=20, kde=True, color='skyblue')

# Add titles and labels
plt.title('Distribution of Popularity Among Tracks', fontsize=16, fontweight='bold')
plt.xlabel('Popularity', fontsize=12)
plt.ylabel('Frequency', fontsize=12)

# Show the plot
plt.show()
```



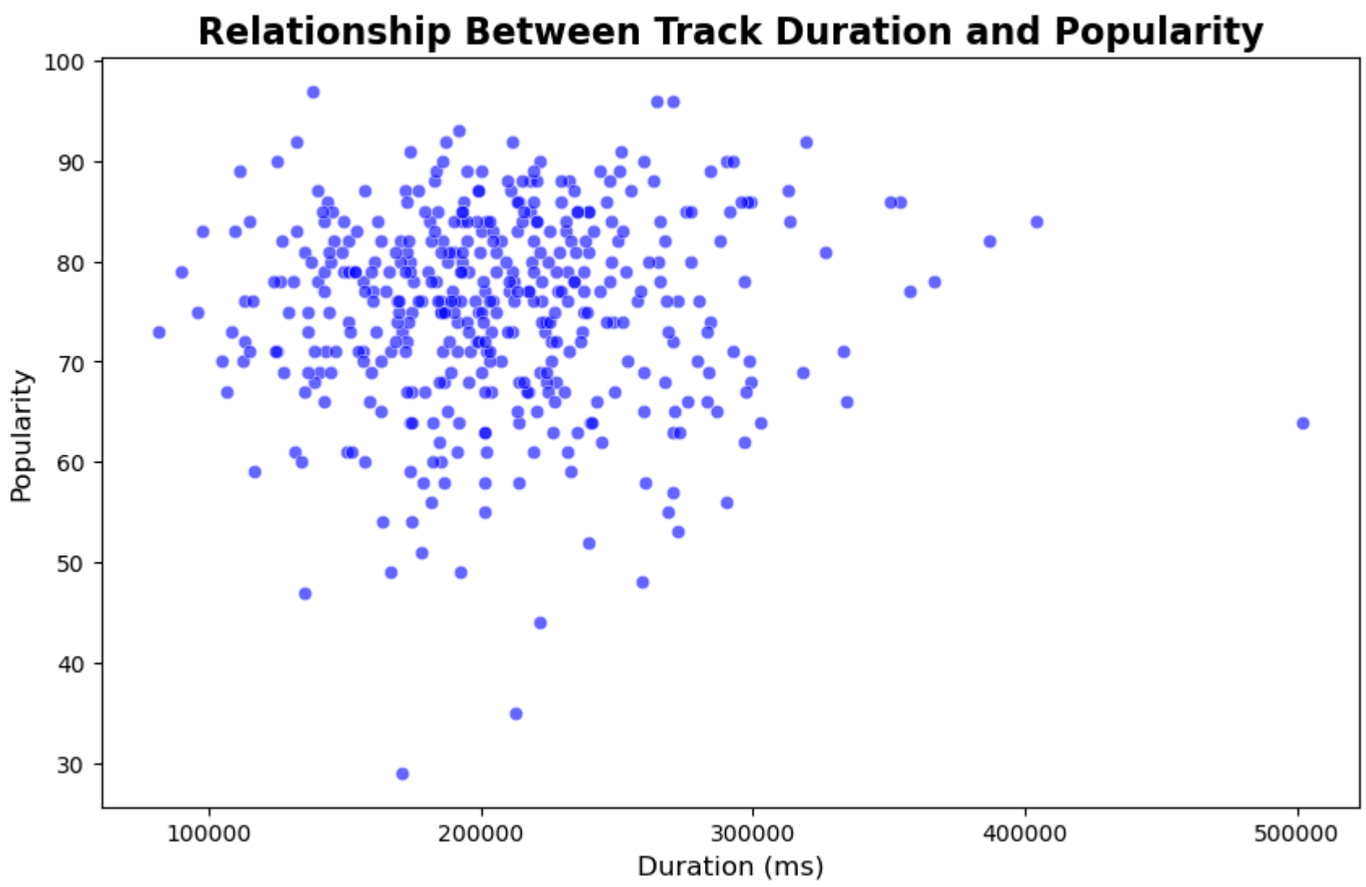
Here are some insights based on the histogram showing the distribution of popularity among tracks:

- Normal Distribution: The distribution of popularity appears to be roughly normal, with a peak around the 80 mark. This suggests that most tracks have a popularity score in this range.
- Frequency Peaks: The highest frequency of tracks is observed in the 70-80 popularity range, indicating that a significant number of tracks are quite popular.
- Lower Popularity Scores: There are fewer tracks with popularity scores below 50, suggesting that very low popularity is less common among the dataset.
- High Popularity Scores: While there are tracks with popularity scores above 90, they are less frequent, indicating that extremely high popularity is achieved by fewer tracks.
- Spread of Data: The distribution shows a gradual decline in frequency as popularity scores move away from the peak, indicating that while many tracks are popular, fewer achieve very high popularity.
- Outliers: There are some outliers on the lower end (below 40), but they are minimal compared to the overall distribution.

Overall, the data suggests that most tracks tend to have moderate to high popularity, with a notable concentration around the 70-80 range.

3.Is there any relationship between the popularity and the duration of tracks? Explore this using a scatter plot.

In [15]:
```
#Plot the relationship between popularity and duration
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Duration (ms)', y='Popularity', data=df, color='blue', alpha=0.6)

# Add titles and labels
plt.title('Relationship Between Track Duration and Popularity', fontsize=16, fontweight='bold')
plt.xlabel('Duration (ms)', fontsize=12)
plt.ylabel('Popularity', fontsize=12)

# Show the plot
plt.show()
```



The scatter plot illustrates the relationship between track duration (in milliseconds) and popularity. Here are some insights based on the visual data:

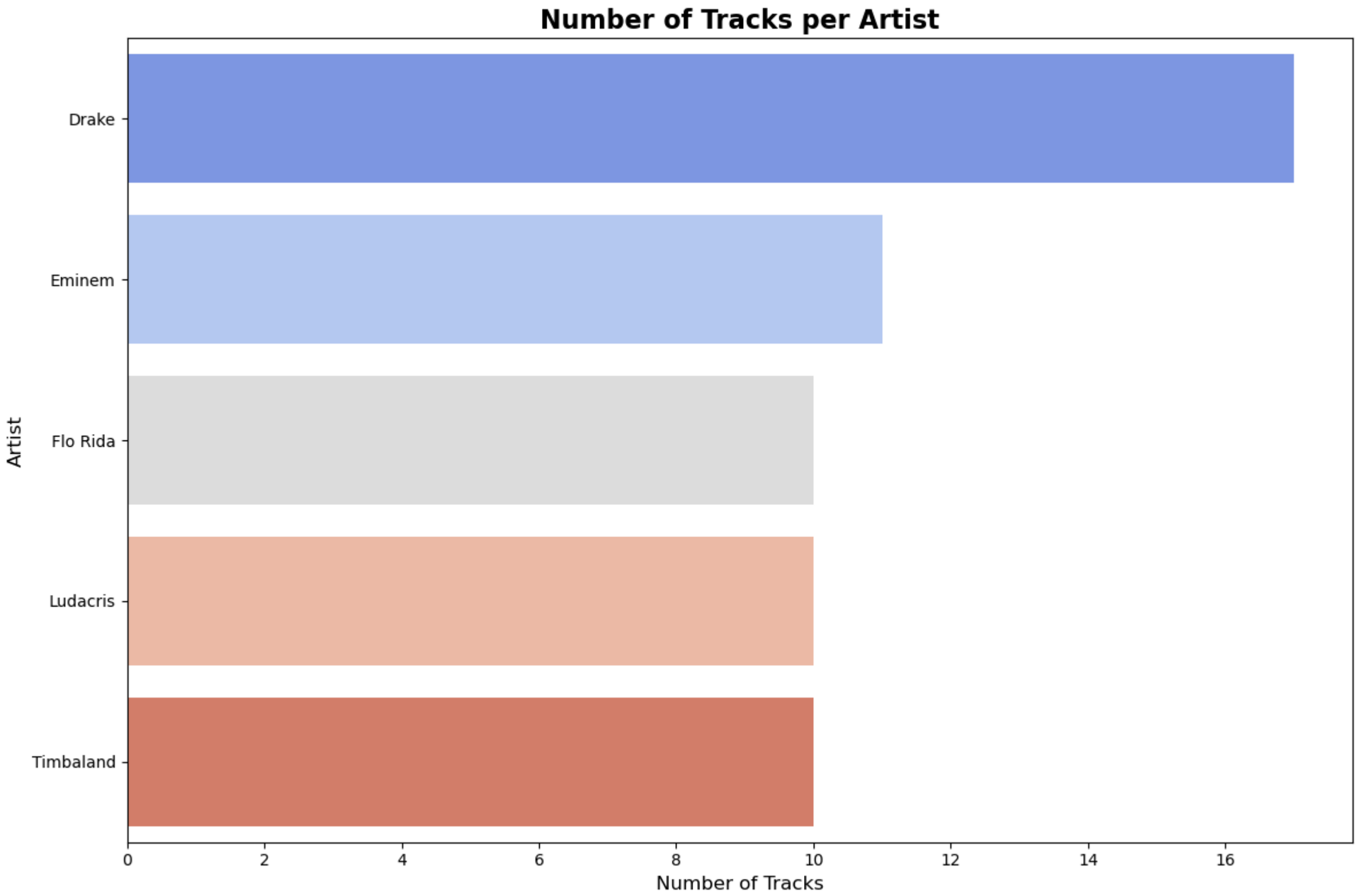
- General Trend: There appears to be a slight positive correlation between track duration and popularity. As the duration increases, the popularity tends to increase as well, although the relationship is not very strong.
- Popularity Range: Most tracks have a popularity score ranging from about 50 to 90, regardless of their duration. This suggests that many tracks achieve a similar level of popularity.
- Outliers: There are a few outliers with high popularity scores (above 90) that have varying durations. This indicates that some tracks can be very popular regardless of their length.
- Duration Distribution: The majority of tracks seem to cluster around the 200,000 ms mark (approximately 3 minutes and 20 seconds), which is a common length for popular songs.
- Variability: There is significant variability in popularity for tracks of similar durations, indicating that factors other than duration likely influence a track's popularity.

4.Which artist has the highest number of tracks in the dataset? Display the count of tracks for each artist using a countplot.

In [20]:
```
# Count the number of tracks for each artist
artist_counts = df['Artist'].value_counts().head(5)

# Find the artist with the highest number of tracks
top_artist = artist_counts.idxmax()
top_tracks = artist_counts.max()

print(f"The artist with the highest number of tracks is {top_artist} with {top_tracks} tracks.")
```

```
# Plot the count of tracks for each artist using a countplot
plt.figure(figsize=(12, 8))
sns.countplot(y='Artist', data=df, order=artist_counts.index, palette='coolwarm')

# Add title and labels
plt.title('Number of Tracks per Artist', fontsize=16, fontweight='bold')
plt.xlabel('Number of Tracks', fontsize=12)
plt.ylabel('Artist', fontsize=12)

# Show the plot
plt.tight_layout()
plt.show()
```

```
The artist with the highest number of tracks is Drake with 17 tracks.
```



5.What are the top 5 least popular tracks in the dataset? Provide the artist name and track name for each.

In [21]:
```
# Sort the dataset by the 'Popularity' column in ascending order to get the least popular tracks
least_popular_tracks = df[['Artist', 'Track Name', 'Popularity']].sort_values(by='Popularity').head(5)

# Display the top 5 least popular tracks (Artist and Track Name)
print("Top 5 Least Popular Tracks:")
print(least_popular_tracks)
```

```
Top 5 Least Popular Tracks:
            Artist                     Track Name  Popularity
207          Pressa  Attachments (feat. Coi Leray)          29
231   Justin Bieber                     Intentions          35
413  French Montana                Splash Brothers          44
225       Lil Baby                  On Me - Remix          47
407    Wyclef Jean      911 (feat. Mary J. Blige)          48
```

6.Among the top 5 most popular artists, which artist has the highest popularity on average? Calculate and display the average popularity for each artist.

In [23]:
```
#Group by 'Artist' and calculate the average 'Popularity'
artist_avg_popularity = df.groupby('Artist')['Popularity'].mean().reset_index()

# Round the 'Popularity' values to two decimal places
artist_avg_popularity['Popularity'] = artist_avg_popularity['Popularity'].round(2)

# Sort by 'Popularity' in descending order to get the most popular artists
top_5_artists = artist_avg_popularity.sort_values(by='Popularity', ascending=False).head(5)

# Display the top 5 artists and their average popularity
print("Top 5 Most Popular Artists by Average Popularity:")
print(top_5_artists)
```

```
Top 5 Most Popular Artists by Average Popularity:
          Artist  Popularity
113         cassö       92.00
104        Trueno       89.00
24   David Guetta       87.00
103  Travis Scott       86.56
114            ¥$       85.10
```

7.For the top 5 most popular artists, what are their most popular tracks? List the track name for each artist.

In [24]:
```
# Group by 'Artist' and calculate the average 'Popularity'
artist_avg_popularity = df.groupby('Artist')['Popularity'].mean().reset_index()

# Sort by 'Popularity' in descending order to get the most popular artists
top_5_artists = artist_avg_popularity.sort_values(by='Popularity', ascending=False).head(5)['Artist']

# Filter the dataset for only the top 5 artists
top_5_artists_df = df[df['Artist'].isin(top_5_artists)]

# Find the most popular track for each of the top 5 artists
most_popular_tracks = top_5_artists_df.loc[top_5_artists_df.groupby('Artist')['Popularity'].idxmax()]

# Select only the 'Artist' and 'Track Name' columns
most_popular_tracks = most_popular_tracks[['Artist', 'Track Name', 'Popularity']].sort_values(by='Popularity', ascending=False)

# Display the result
print("Most Popular Tracks for the Top 5 Most Popular Artists:")
print(most_popular_tracks)
```

```
Most Popular Tracks for the Top 5 Most Popular Artists:
          Artist                    Track Name  Popularity
260            ¥$                     CARNIVAL          96
30   Travis Scott    FE!N (feat. Playboi Carti)          93
```

```
140         cassö                          Prada          92
241        Trueno  Mamichula - con Nicki Nicole          89
200  David Guetta             Baby Don't Hurt Me          87
```

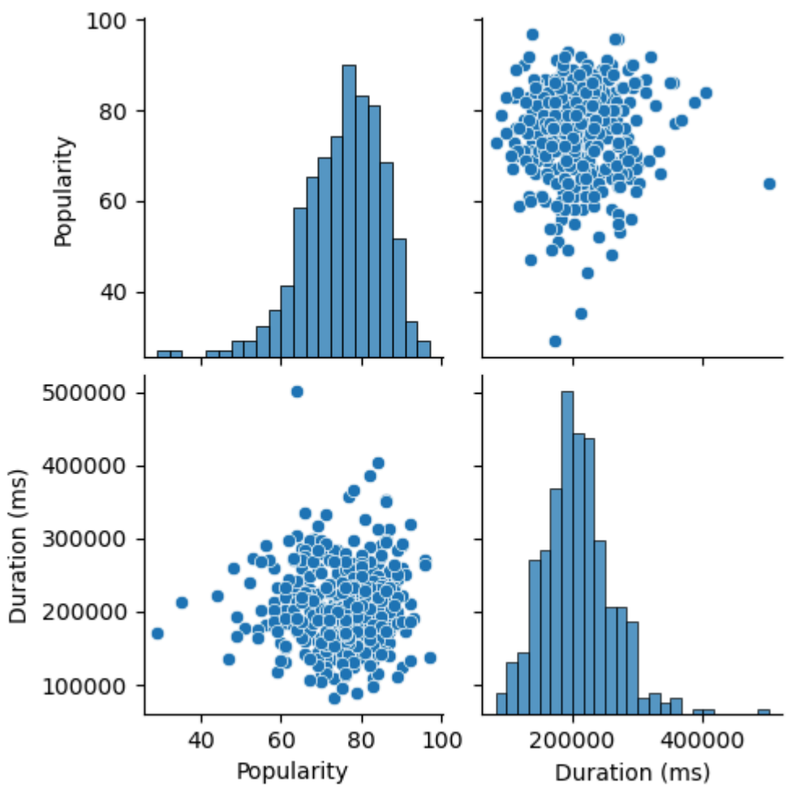
In [25]:
```
df.columns
```

Out[25]:
```
Index(['Artist', 'Track Name', 'Popularity', 'Duration (ms)', 'Track ID'], dtype='object')
```

8.Visualize relationships between multiple numerical variables simultaneously using a pair plot

In [26]:
```
# Select numerical columns for pair plot
numerical_cols= ['Popularity','Duration (ms)']

#create a pair plot
sns.pairplot(df[numerical_cols])
```

Out[26]:
```
<seaborn.axisgrid.PairGrid at 0x24948fc7e10>
```

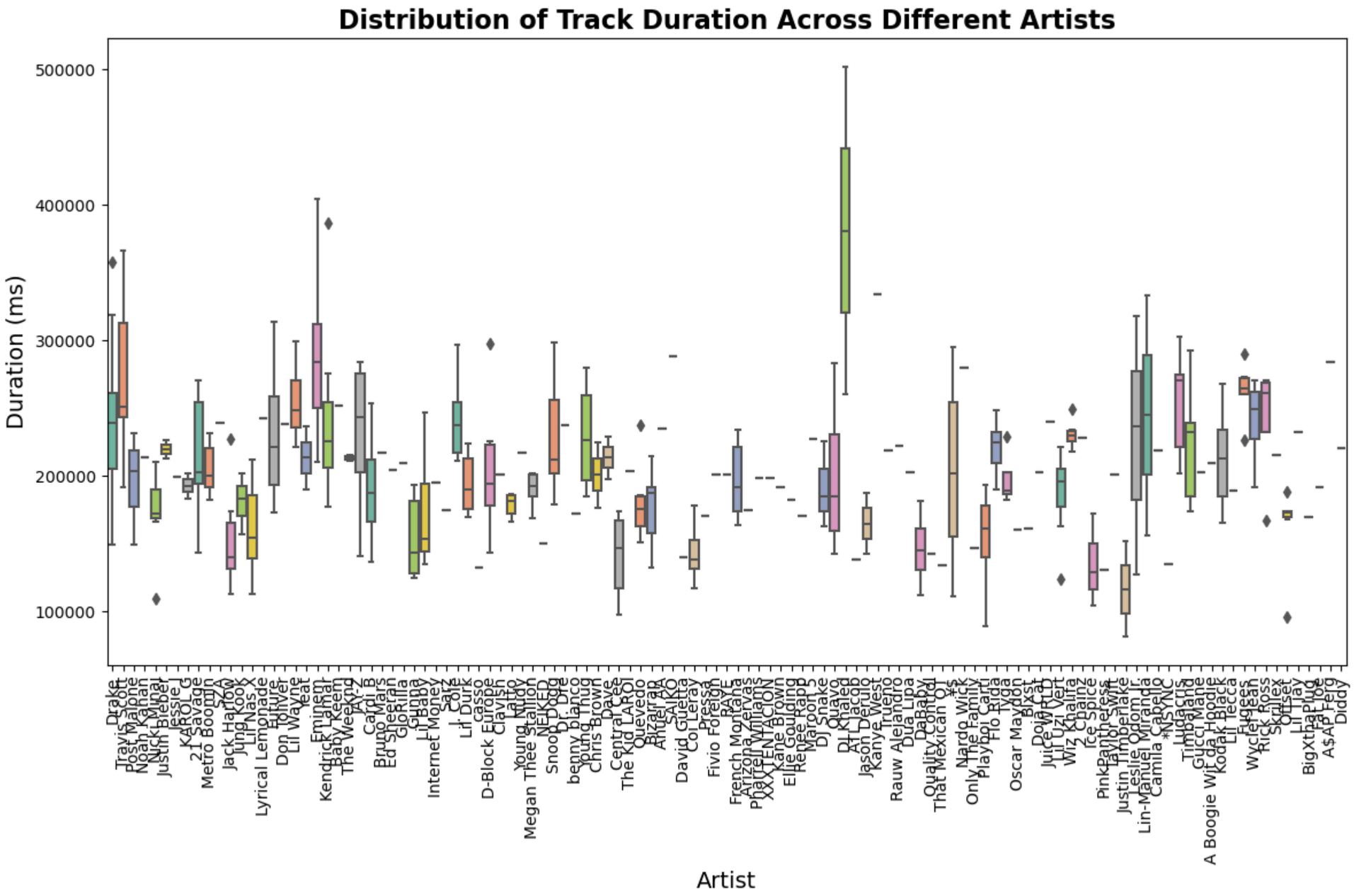


9.Does the duration of tracks vary significantly across different artists? Explore this visually using a box plot or violin plot.

Using a Box Plot

- A box plot shows the distribution of track durations across different artists, including median, quartiles, and potential outliers.

In [27]:
```
# Create a box plot to visualize the distribution of track durations across different artists
plt.figure(figsize=(12, 8))
sns.boxplot(x='Artist', y='Duration (ms)', data=df, palette='Set2')

# Add title and labels
plt.title('Distribution of Track Duration Across Different Artists', fontsize=16, fontweight='bold')
plt.xlabel('Artist', fontsize=14)
plt.ylabel('Duration (ms)', fontsize=14)
plt.xticks(rotation=90)  # Rotate x-axis labels if needed

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

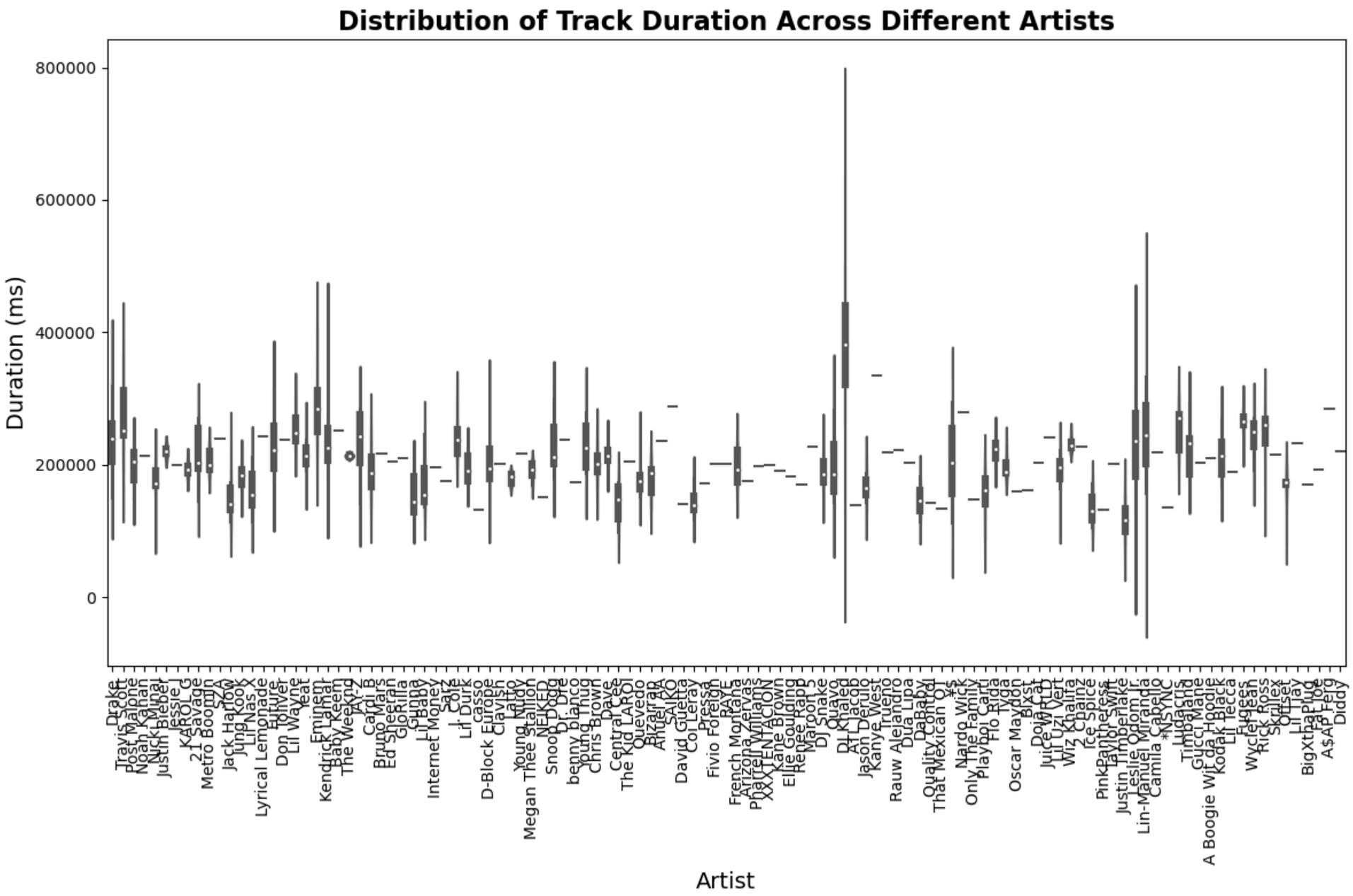


Using a Violin Plot

- A violin plot provides a combination of a box plot and a kernel density plot, showing the distribution of track durations and its density across different artists.

In [28]:
```
# Create a violin plot to visualize the distribution of track durations across different artists
plt.figure(figsize=(12, 8))
sns.violinplot(x='Artist', y='Duration (ms)', data=df, palette='Set2')

# Add title and labels
plt.title('Distribution of Track Duration Across Different Artists', fontsize=16, fontweight='bold')
```

```
plt.xlabel('Artist', fontsize=14)
plt.ylabel('Duration (ms)', fontsize=14)
plt.xticks(rotation=90)  # Rotate x-axis labels if needed

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

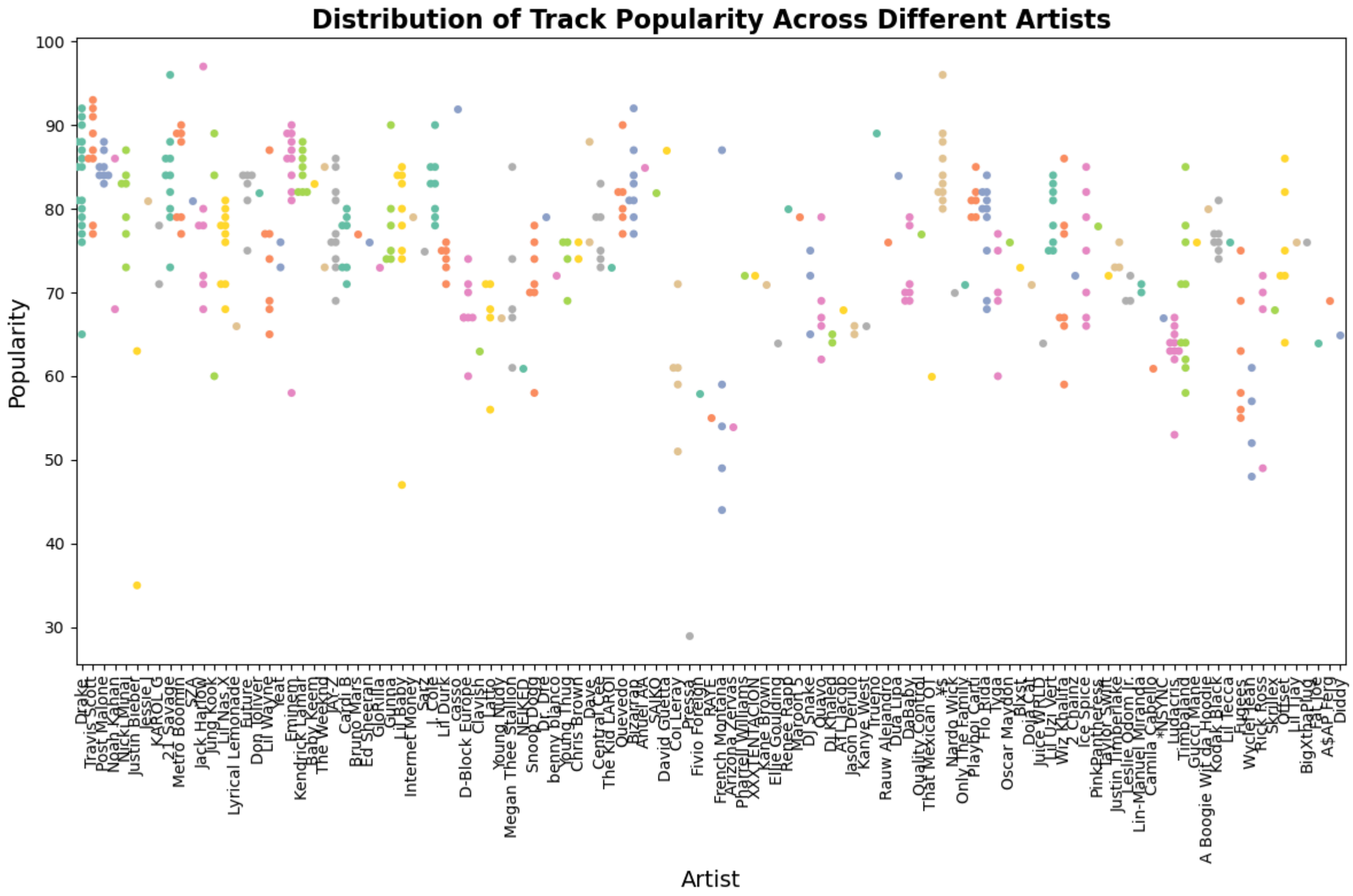


10.How does the distribution of track popularity vary for different artists? Visualize this using a swarm plot or a violin plot.

Using a Swarm Plot

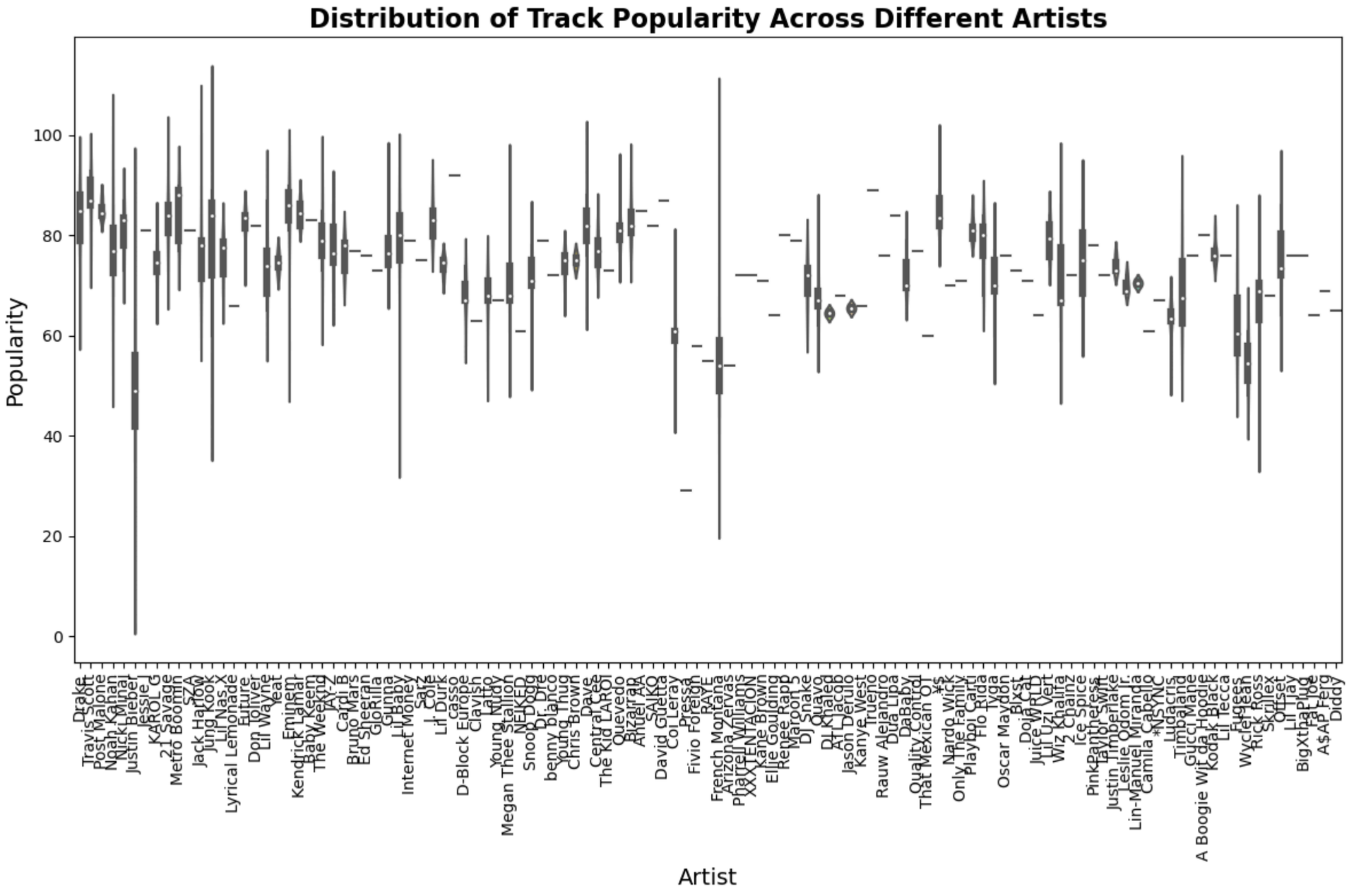
- A swarm plot shows individual data points and their distribution, providing a clear view of how popularity scores are distributed for each artist.

In [29]:
```
# Create a swarm plot to visualize the distribution of track popularity across different artists
plt.figure(figsize=(12, 8))
sns.swarmplot(x='Artist', y='Popularity', data=df, palette='Set2')

# Add title and labels
plt.title('Distribution of Track Popularity Across Different Artists', fontsize=16, fontweight='bold')
plt.xlabel('Artist', fontsize=14)
plt.ylabel('Popularity', fontsize=14)
plt.xticks(rotation=90)  # Rotate x-axis labels if needed

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



In [30]:
```
# Create a violin plot to visualize the distribution of track popularity across different artists
plt.figure(figsize=(12, 8))
sns.violinplot(x='Artist', y='Popularity', data=df, palette='Set2')

# Add title and labels
```

```
plt.title('Distribution of Track Popularity Across Different Artists', fontsize=16, fontweight='bold')
plt.xlabel('Artist', fontsize=14)
plt.ylabel('Popularity', fontsize=14)
plt.xticks(rotation=90)  # Rotate x-axis labels if needed

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



In [ ]: