Amazon Sales Data EDA

Description:

This dataset contains information on 1K+ Amazon products, including their ratings, reviews, and other details.

Features:

product_id: Unique identifier for each product

product_name: Name of the product

category: Category of the product

discounted_price: Discounted price of the product

actual_price: Actual price of the product

discount_percentage: Percentage of discount for the product

rating: Rating of the product (1-5)

rating_count: Number of people who voted for the Amazon rating

about_product: Description about the product

user_id: ID of the user who wrote the review

user_name: Name of the user who wrote the review

review_id: ID of the user review

review_title: Short review

review_content: Long review

img_link: Image link of the product

product_link: Official website link of the product

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [52]:

```
data = pd.read_csv('/content/amazon.csv')
data.head()
```

Out[52]:

	product_id	product_name	category	discounted_price	actual_price	discount_
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	₹399	₹1,099	
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	₹199	₹349	

	product_id	product_name	category	discounted_price	actual_price	discount_
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	₹199	₹1,899	
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	Computers&Accessories|Accessories&Peripherals|...	₹329	₹699	
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	Computers&Accessories|Accessories&Peripherals|...	₹154	₹399	

In [39]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   product_id           1465 non-null   object
 1   product_name         1465 non-null   object
 2   category             1465 non-null   object
 3   discounted_price     1465 non-null   object
 4   actual_price         1465 non-null   object
 5   discount_percentage  1465 non-null   object
 6   rating               1465 non-null   object
 7   rating_count         1463 non-null   object
 8   about_product        1465 non-null   object
 9   user_id              1465 non-null   object
 10  user_name            1465 non-null   object
 11  review_id            1465 non-null   object
 12  review_title         1465 non-null   object
 13  review_content       1465 non-null   object
 14  img_link             1465 non-null   object
 15  product_link         1465 non-null   object
dtypes: object(16)
memory usage: 183.2+ KB
```

In [53]:

```
data = data.dropna(subset=['rating_count'])

data.shape
```

Out[53]:

(1463, 16)

In [41]:

```
data.describe()
```

Out[41]:

	product_id	product_name	category	discounted_price	actual_price	disco
count	1463	1463	1463	1463	1463	
unique	1349	1335	211	550	449	
top	B07JW9H4J1	Fire-Boltt Ninja Call Pro Plus 1.83" Smart Wat...	Computers&Accessories|Accessories&Peripherals|...	₹199	₹999	

	product_id	product_name	category	discounted_price	actual_price	disco
freq	3	5	231	52	118	

Conclusion:

1. Dataset consist of multiple types of data such as textual, categorcial and numarical
2. In rating_count, 2 datapoints are missing, performed drop na

1. What is the average rating for each product category

In [42]:

```
data['category'].value_counts()
```

Out[42]:

cat
Computers&Accessories|Accessories&Peripherals|Cables&Accessories|Cables|USBC
Electronics|WearableTechnology|SmartWa
Electronics|Mobiles&Accessories|Smartphones&BasicMobiles|Smartpl
Electronics|HomeTheater,TV&Video|Televisions|SmartTelevi
Electronics|Headphones,Earbuds&Accessories|Headphones|I
Electronics|Cameras&Photography|Accessories|Batteries&Chargers|BatteryCha
Computers&Accessories|NetworkingDevices|DataCards&Do
Electronics|HomeAudio|Speakers|MultimediaSpeakerSys
OfficeProducts|OfficePaperProducts|Paper|Copy&PrintingPaper|Coloured|
Home&Kitchen|Kitchen&HomeAppliances|Vacuum,Cleaning&Ironing|Vacuums&FloorCare|VacuumAccessories|VacuumBags|Handhelc

211 rows × 1 columns

dtype: int64

In [43]:

```
data['rating'].value_counts()
```

Out[43]:

rating	count
4.1	244
4.3	230
4.2	228
4.0	129
3.9	123
4.4	123
3.8	86
4.5	75

rating	count
4	52
3.7	42
3.6	35
3.5	26
4.6	17
3.3	16
3.4	10
4.7	6
3.1	4
4.8	3
3.2	2
2.8	2
3.0	2
5.0	2
2.3	1
|	1
2	1
3	1
2.6	1
2.9	1

dtype: int64

In [44]:

```
data['rating'].dtype
```

Out[44]:

```
dtype('O')
```

In [45]:

```
data['rating'] = pd.to_numeric(data['rating'], errors='coerce')
data['rating'].dtype
```

Out[45]:

```
dtype('float64')
```

In [46]:

```
average_ratings = data.groupby('category')['rating'].mean().reset_index()
average_ratings
```

Out[46]:

	category	rating
0	Car&Motorbike|CarAccessories|InteriorAccessori...	3.800000
1	Computers&Accessories|Accessories&Peripherals|...	4.150000
2	Computers&Accessories|Accessories&Peripherals|...	3.500000
3	Computers&Accessories|Accessories&Peripherals|...	3.600000
4	Computers&Accessories|Accessories&Peripherals|...	4.050000
...	...	...
206	OfficeProducts|OfficePaperProducts|Paper|Stati...	4.250000

<table>
<tr><td>207</td><td>OfficeProducts|OfficePaperProducts|Paper|Stati...</td><td>4.150000</td></tr>
<tr><td>208</td><td>OfficeProducts|OfficePaperProducts|Paper|Stati...</td><td>4.300000</td></tr>
<tr><td>209</td><td>OfficeProducts|OfficePaperProducts|Paper|Stati...</td><td>4.133333</td></tr>
<tr><td>210</td><td>Toys&Games|Arts&Crafts|Drawing&PaintingSupplie...</td><td>4.300000</td></tr>
</table>

211 rows × 2 columns

In [47]:

```
average_ratings = average_ratings.sort_values(by='rating', ascending=False)
average_ratings
```

Out[47]:

<table>
<tr><th></th><th>category</th><th>rating</th></tr>
<tr><td>57</td><td>Computers&Accessories|Tablets</td><td>4.6</td></tr>
<tr><td>48</td><td>Computers&Accessories|NetworkingDevices|Networ...</td><td>4.5</td></tr>
<tr><td>62</td><td>Electronics|Cameras&Photography|Accessories|Film</td><td>4.5</td></tr>
<tr><td>81</td><td>Electronics|HomeAudio|MediaStreamingDevices|St...</td><td>4.5</td></tr>
<tr><td>196</td><td>OfficeProducts|OfficeElectronics|Calculators|B...</td><td>4.5</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
<tr><td>3</td><td>Computers&Accessories|Accessories&Peripherals|...</td><td>3.6</td></tr>
<tr><td>88</td><td>Electronics|HomeTheater,TV&Video|Accessories|3...</td><td>3.5</td></tr>
<tr><td>2</td><td>Computers&Accessories|Accessories&Peripherals|...</td><td>3.5</td></tr>
<tr><td>14</td><td>Computers&Accessories|Accessories&Peripherals|...</td><td>3.4</td></tr>
<tr><td>146</td><td>Home&Kitchen|Kitchen&HomeAppliances|Coffee,Tea...</td><td>3.3</td></tr>
</table>

211 rows × 2 columns

In [48]:

```
average_ratings = average_ratings.sort_values(by='rating', ascending=True)
average_ratings
```

Out[48]:

<table>
<tr><th></th><th>category</th><th>rating</th></tr>
<tr><td>146</td><td>Home&Kitchen|Kitchen&HomeAppliances|Coffee,Tea...</td><td>3.3</td></tr>
<tr><td>14</td><td>Computers&Accessories|Accessories&Peripherals|...</td><td>3.4</td></tr>
<tr><td>88</td><td>Electronics|HomeTheater,TV&Video|Accessories|3...</td><td>3.5</td></tr>
<tr><td>2</td><td>Computers&Accessories|Accessories&Peripherals|...</td><td>3.5</td></tr>
<tr><td>3</td><td>Computers&Accessories|Accessories&Peripherals|...</td><td>3.6</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
<tr><td>194</td><td>HomeImprovement|Electrical|CordManagement</td><td>4.5</td></tr>
<tr><td>123</td><td>Home&Kitchen|CraftMaterials|PaintingMaterials</td><td>4.5</td></tr>
<tr><td>38</td><td>Computers&Accessories|Components|Memory</td><td>4.5</td></tr>
<tr><td>148</td><td>Home&Kitchen|Kitchen&HomeAppliances|Coffee,Tea...</td><td>4.5</td></tr>
<tr><td>57</td><td>Computers&Accessories|Tablets</td><td>4.6</td></tr>
</table>

211 rows × 2 columns

Conclusion:

1. There are 211 product categories
2. Category "Computers&Accessories|Tablets" is highest average rating i.e. 4.6
3. Category "Home&Kitchen|Kitchen&HomeAppliances|Coffee,Tea&Espresso|CoffeeGrinders|ElectricGrinders" is lowest average rarting 3.3.

2. What are the top rating_count products by category?

In [49]:

```
data['rating_count'].dtype
```

Out[49]:

```
dtype('O')
```

In [54]:

```
data['rating_count'].head()
```

Out[54]:

	rating_count
0	24,269
1	43,994
2	7,928
3	94,363
4	16,905

dtype: object

In [56]:

```
data['rating_count'] = data['rating_count'].str.replace(',', '', regex=True)
data['rating_count'] = pd.to_numeric(data['rating_count'])
data['rating_count'].head()
```

Out[56]:

	rating_count
0	24269
1	43994
2	7928
3	94363
4	16905

dtype: int64

In [64]:

```
top_products = (
    data.loc[data.groupby('category')['rating_count'].idxmax()]
    .reset_index(drop=True)
)
```

In [65]:

```
top_products = top_products.sort_values(by='rating_count', ascending=False)
top_products[['category', 'product_name', 'rating_count']].head(5)
```

Out[65]:

	category	product_name	rating_count
89	Electronics|HomeTheater,TV&Video|Accessories|C...	AmazonBasics Flexible Premium HDMI Cable (Blac...	426973
76	Electronics|Headphones,Earbuds&Accessories|Hea...	boAt Bassheads 100 in Ear Wired Earphones with...	363713
117	Electronics|Mobiles&Accessories|Smartphones&Ba...	Redmi 9 Activ (Carbon Black, 4GB RAM, 64GB Sto...	313836
145	Home&Kitchen|Kitchen&Dining|KitchenTools|Manua...	Pigeon Polypropylene Mini Handy and Compact Ch...	270563
42	Computers&Accessories|ExternalDevices&DataStor...	SanDisk Cruzer Blade 32GB USB Flash Drive	253105

Conclusion:

First category with top rating count product category is "Electronics|HomeTheater,TV&Video|Accessories|Cables|HDMICables" with the product name "AmazonBasics Flexible Premium HDMI Cable (Black, 4K@60Hz, 18Gbps), 3-Foot" and 426973 reviews.

3. What is the distribution of discounted prices vs. actual prices?

data cleaning wrt. discounted prices

In [66]:

```
data['discounted_price'].head(2)
```

Out[66]:

	discounted_price
0	₹399
1	₹199

dtype: object

In [68]:

```
data['discounted_price'] = data['discounted_price'].str.replace('₹', '', regex=True)
data['discounted_price'] = data['discounted_price'].str.replace(',', '', regex = True)
data['discounted_price'] = pd.to_numeric(data['discounted_price'])

data['discounted_price'].head(1)
```

Out[68]:

	discounted_price
0	399.0

dtype: float64

data cleaning for actual price

In [70]:

```
data['actual_price'].head(1)
```

Out[70]:

	actual_price

	actual_price
0	₹1,099

dtype: object

In [71]:

```
data['actual_price'] = data['actual_price'].str.replace('₹', '', regex=True)
data['actual_price'] = data['actual_price'].str.replace(',', '', regex = True)
data['actual_price'] = pd.to_numeric(data['actual_price'])

data['actual_price'].head(1)
```

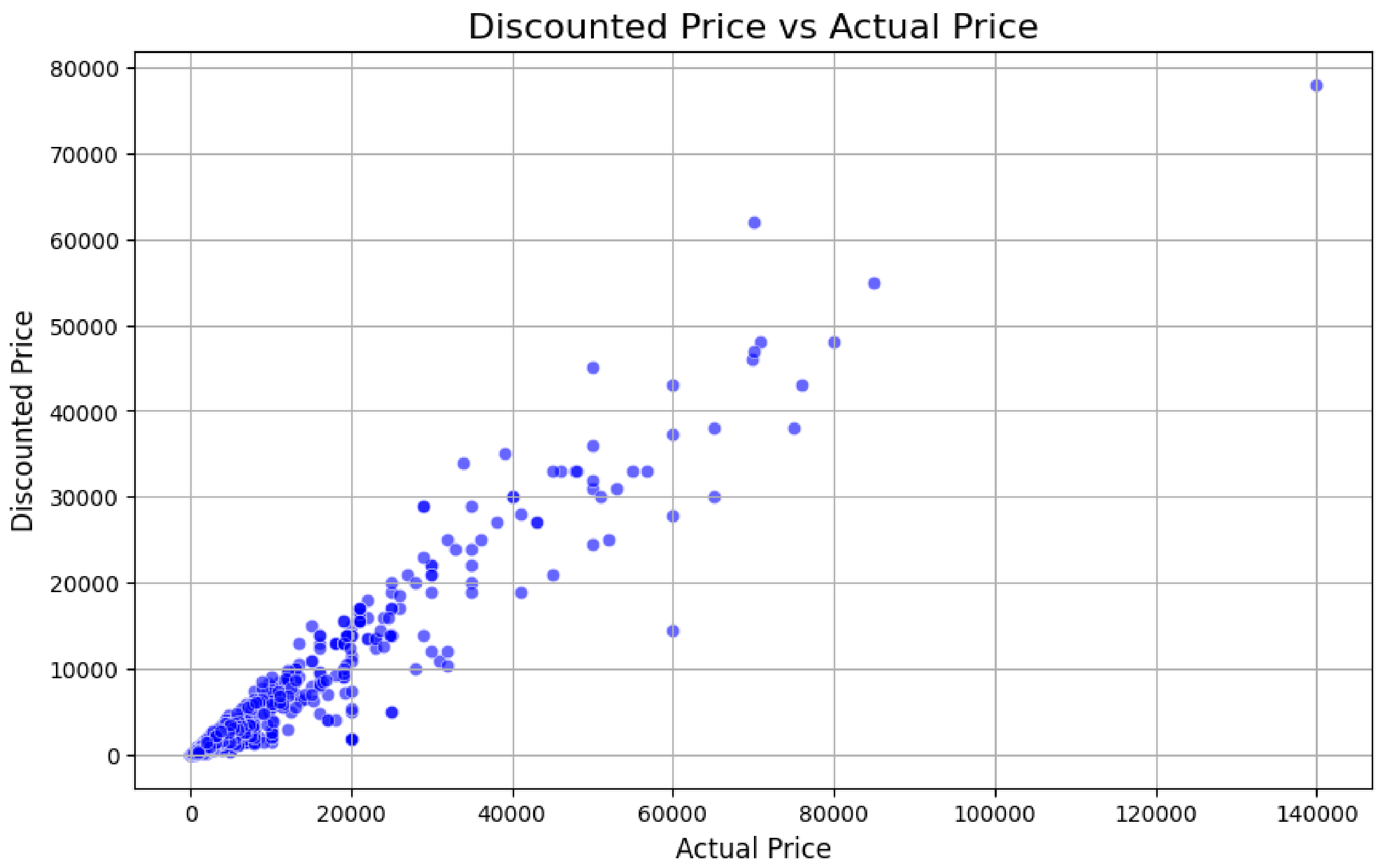
Out[71]:

	actual_price
0	1099.0

dtype: float64

In [72]:

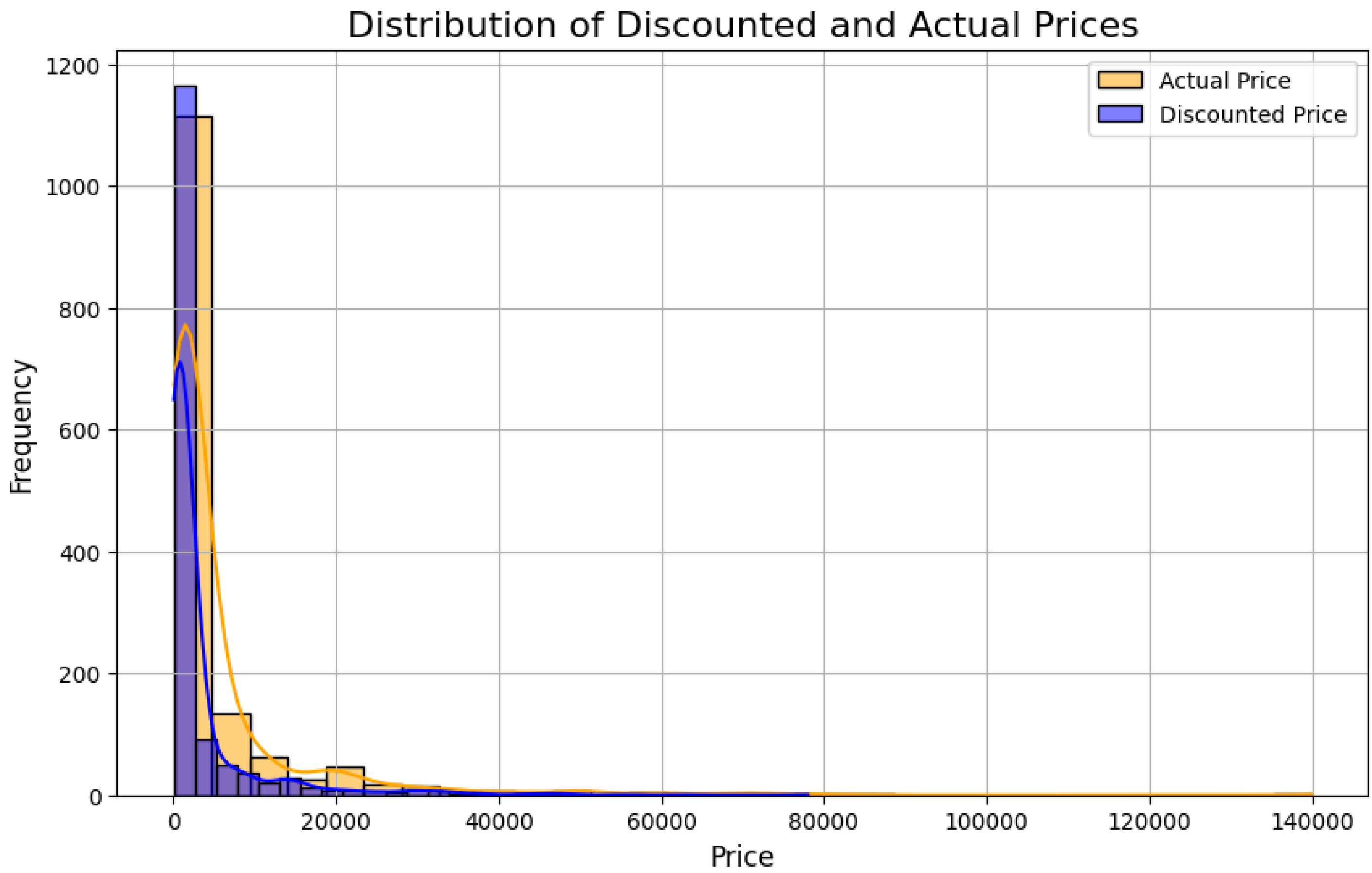
```
plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=data,
    x='actual_price',
    y='discounted_price',
    alpha=0.6,
    color='blue'
)
plt.title('Discounted Price vs Actual Price', fontsize=16)
plt.xlabel('Actual Price', fontsize=12)
plt.ylabel('Discounted Price', fontsize=12)
plt.grid(True)
plt.show()
```



In [73]:

```
plt.figure(figsize=(10, 6))
```

```
sns.histplot(data['actual_price'], label='Actual Price', kde=True, color='orange', bins=3
0)
sns.histplot(data['discounted_price'], label='Discounted Price', kde=True, color='blue',
bins=30)
plt.title('Distribution of Discounted and Actual Prices', fontsize=16)
plt.xlabel('Price', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```



Conclusion:

1. **Distribution seems to be overlapping (same distribution)**
2. **Scatter plot is showing a positive linear relationship between both the features.**

4. How does the average discount percentage vary across categories?

In [74]:

```
data['discount_percentage'].head(1)
```

Out[74]:

	discount_percentage
0	64%

dtype: object

In [75]:

```
data['discount_percentage'] = data['discount_percentage'].str.replace('%', '', regex=Tru
e)
```

```
data['discount_percentage'] = pd.to_numeric(data['discount_percentage'])
data['discount_percentage'].head(1)
```

Out[75]:

	discount_percentage
0	64

dtype: int64

In [77]:

```
avg_discount = data.groupby('category')['discount_percentage'].mean().reset_index()
avg_discount = avg_discount.sort_values(by='discount_percentage', ascending=False)

avg_discount
```

Out[77]:

	category	discount_percentage
106	Electronics|Mobiles&Accessories|MobileAccessor...	90.0
6	Computers&Accessories|Accessories&Peripherals|...	90.0
75	Electronics|Headphones,Earbuds&Accessories|Ear...	90.0
73	Electronics|Headphones,Earbuds&Accessories|Ada...	88.0
14	Computers&Accessories|Accessories&Peripherals|...	87.5
...	...	...
196	OfficeProducts|OfficeElectronics|Calculators|B...	0.0
176	Home&Kitchen|Kitchen&HomeAppliances|SmallKitch...	0.0
81	Electronics|HomeAudio|MediaStreamingDevices|St...	0.0
62	Electronics|Cameras&Photography|Accessories|Film	0.0
210	Toys&Games|Arts&Crafts|Drawing&PaintingSupplie...	0.0

211 rows × 2 columns

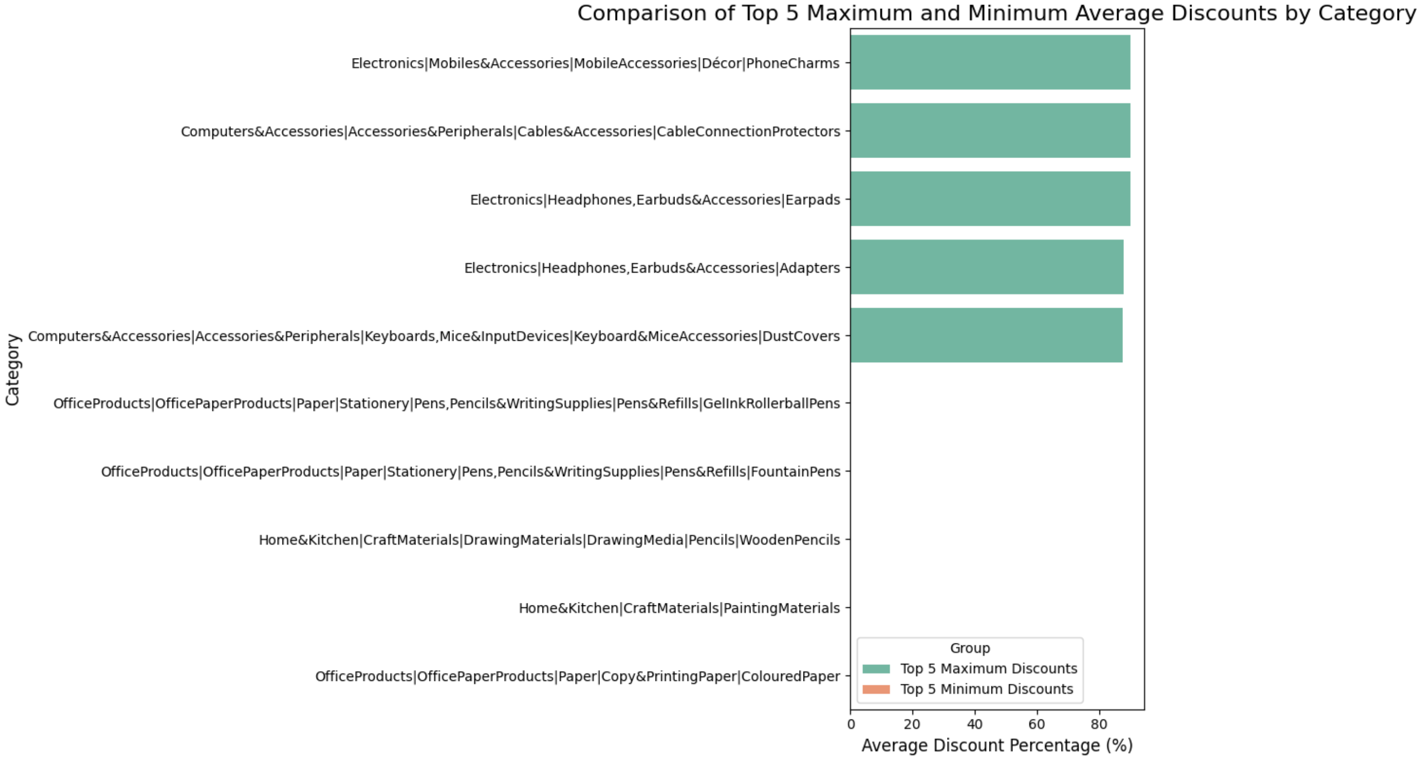
In [80]:

```
top_5_discount = avg_discount.nlargest(5, 'discount_percentage')
bottom_5_discount = avg_discount.nsmallest(5, 'discount_percentage')
```

In [81]:

```
# Combine top and bottom categories into one DataFrame for comparison
combined = pd.concat([
    top_5_discount.assign(Group='Top 5 Maximum Discounts'),
    bottom_5_discount.assign(Group='Top 5 Minimum Discounts')
])

# Plot the combined data
plt.figure(figsize=(12, 8))
sns.barplot(
    data=combined,
    x='discount_percentage',
    y='category',
    hue='Group',
    palette='Set2'
)

# Add titles and labels
plt.title('Comparison of Top 5 Maximum and Minimum Average Discounts by Category', fontsi
ze=16)
plt.xlabel('Average Discount Percentage (%)', fontsize=12)
plt.ylabel('Category', fontsize=12)
```

```
plt.legend(title='Group', fontsize=10)
plt.tight_layout()
plt.show()
```



Conclusion:

1. **Top Category with max. discount of 90% is "Electronics|Mobiles&Accessories|MobileAccessories|Décor|PhoneCharms"**
2. **Categories belongs to the office products, home and kitchen, creaft materials etc. is having least or 0% discount.**

5. What are the most popular product names?

In [82]:

```
data['product_name'].value_counts(ascending = True).head()
```

Out[82]:

product_name	count
AirCase Protective Laptop Bag Sleeve fits Upto 13.3" Laptop/ MacBook, Wrinkle Free, Padded, Waterproof Light Neoprene case Cover Pouch, for Men & Women, Black- 6 Months Warranty	1
USHA Quartz Room Heater with Overheating Protection (3002, Ivory, 800 Watts)	1
Pigeon by Stovekraft Amaze Plus Electric Kettle (14289) with Stainless Steel Body, 1.5 litre, used for boiling Water, making tea and coffee, instant noodles, soup etc. 1500 Watt (Silver)	1
Infinity (JBL Fuze 100, Wireless Portable Bluetooth Speaker with Mic, Deep Bass, Dual Equalizer, IPX7 Waterproof, Rugged Fabric Design (Black)	1
SWAPKART Portable Flexible Adjustable Eye Protection USB LED Desk Light Table Lamp for Reading, Working on PC, Laptop, Power Bank, Bedroom (Multicolour)	1

dtype: int64

6. What are the most popular product keywords?

In [83]:

```python
import re

def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove special characters and numbers
    text = re.sub(r'[^a-z\s]', '', text)
    return text
```

In [84]:

```python
data['cleaned_about_product'] = data['about_product'].apply(preprocess_text)
data['cleaned_about_product'].head(2)
```

Out[84]:

	cleaned_about_product
0	high compatibility compatible with iphone x...
1	compatible with all type c enabled devices be ...

dtype: object

In [85]:

```python
text_data = ' '.join(data['cleaned_about_product'].dropna())
len(text_data)
```

Out[85]:

995573

In [86]:

```python
from wordcloud import WordCloud
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS

wordcloud = WordCloud(stopwords=ENGLISH_STOP_WORDS, background_color="white", width=800,
height=400).generate(text_data)

plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Most Popular Product Keywords', fontsize=16)
plt.axis('off')
plt.show()
```

Most Popular Product Keywords

In [87]:

```
from collections import Counter
words = text_data.split()

filtered_words = [word for word in words if word not in ENGLISH_STOP_WORDS]

word_counts = Counter(filtered_words)

word_counts.most_common(10)
```

Out[87]:

```
[('usb', 1004),
 ('cable', 823),
 ('charging', 620),
 ('warranty', 502),
 ('power', 495),
 ('x', 484),
 ('devices', 476),
 ('design', 472),
 ('use', 460),
 ('easy', 431)]
```

7. What are the most popular product reviews?

In [89]:

```
most_liked_reviews = data.sort_values('rating_count', ascending=False).head(10)
most_liked_reviews[['review_title', 'review_content', 'rating_count']]
```

Out[89]:

	review_title	review_content	rating_count
12	It's quite good and value for money,Works well...	I am using it for 14 days now. The experience ...	426973
65	It's quite good and value for money,Works well...	I am using it for 14 days now. The experience ...	426973
47	It's quite good and value for money,Works well...	I am using it for 14 days now. The experience ...	426973
684	It's quite good and value for money,Works well...	I am using it for 14 days now. The experience ...	426972
400	Best value for money,HEAD PHONE POUCH NOT RECE...	The sound quality of this earphone are really ...	363713
352	Best value for money,HEAD PHONE POUCH NOT RECE...	The sound quality of this earphone are really ...	363713
584	Best value for money,HEAD PHONE POUCH NOT RECE...	The sound quality of this earphone are really ...	363711
370	Best phone for below normal use,Good mobile fo...	If you want a smart phone for just the use of ...	313836
371	Best phone for below normal use,Good mobile fo...	If you want a smart phone for just the use of ...	313836
473	Best phone for below normal use,Good mobile fo...	If you want a smart phone for just the use of ...	313832

In [90]:

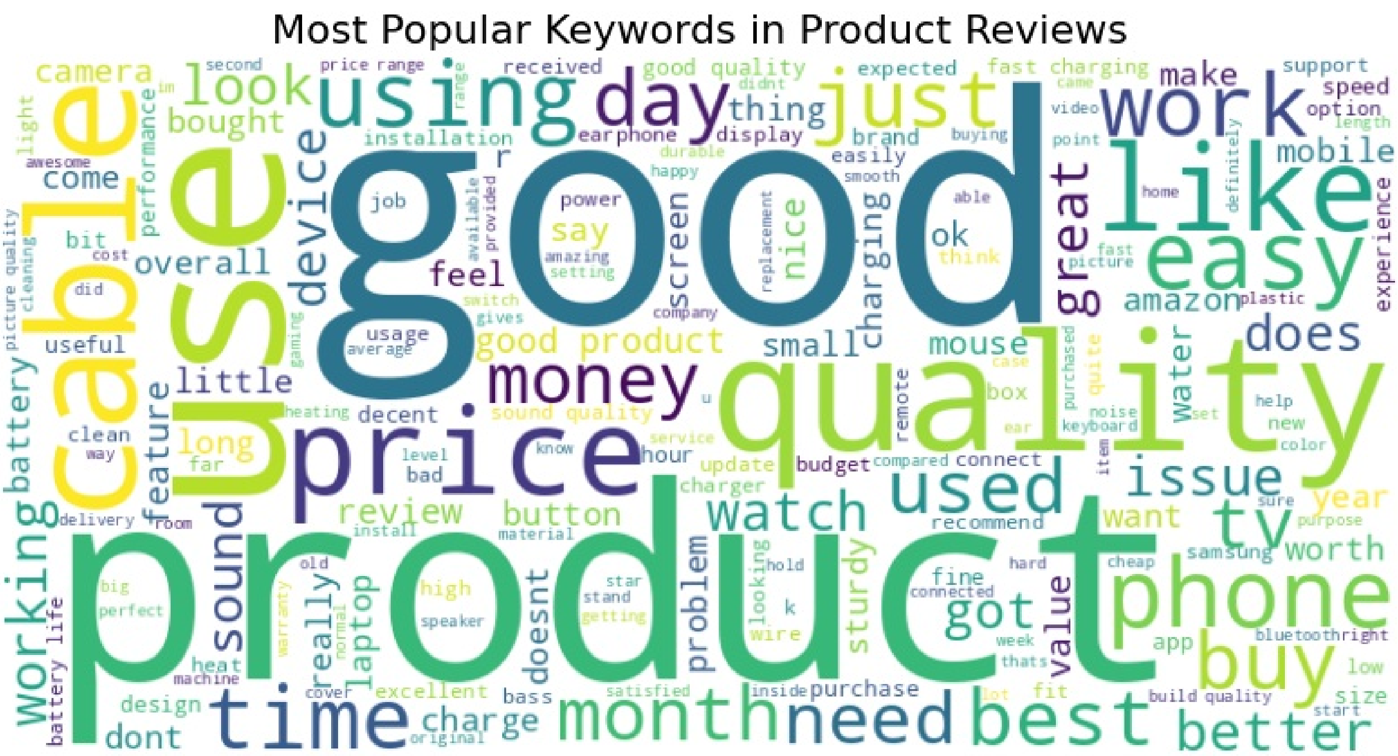
```
all_reviews = ' '.join(data['review_title'].dropna().astype(str)) + ' ' + ' '.join(data[
'review_content'].dropna().astype(str))
all_reviews_cleaned = preprocess_text(all_reviews)
len(all_reviews_cleaned)
```

Out[90]:

```
2188789
```

In [91]:

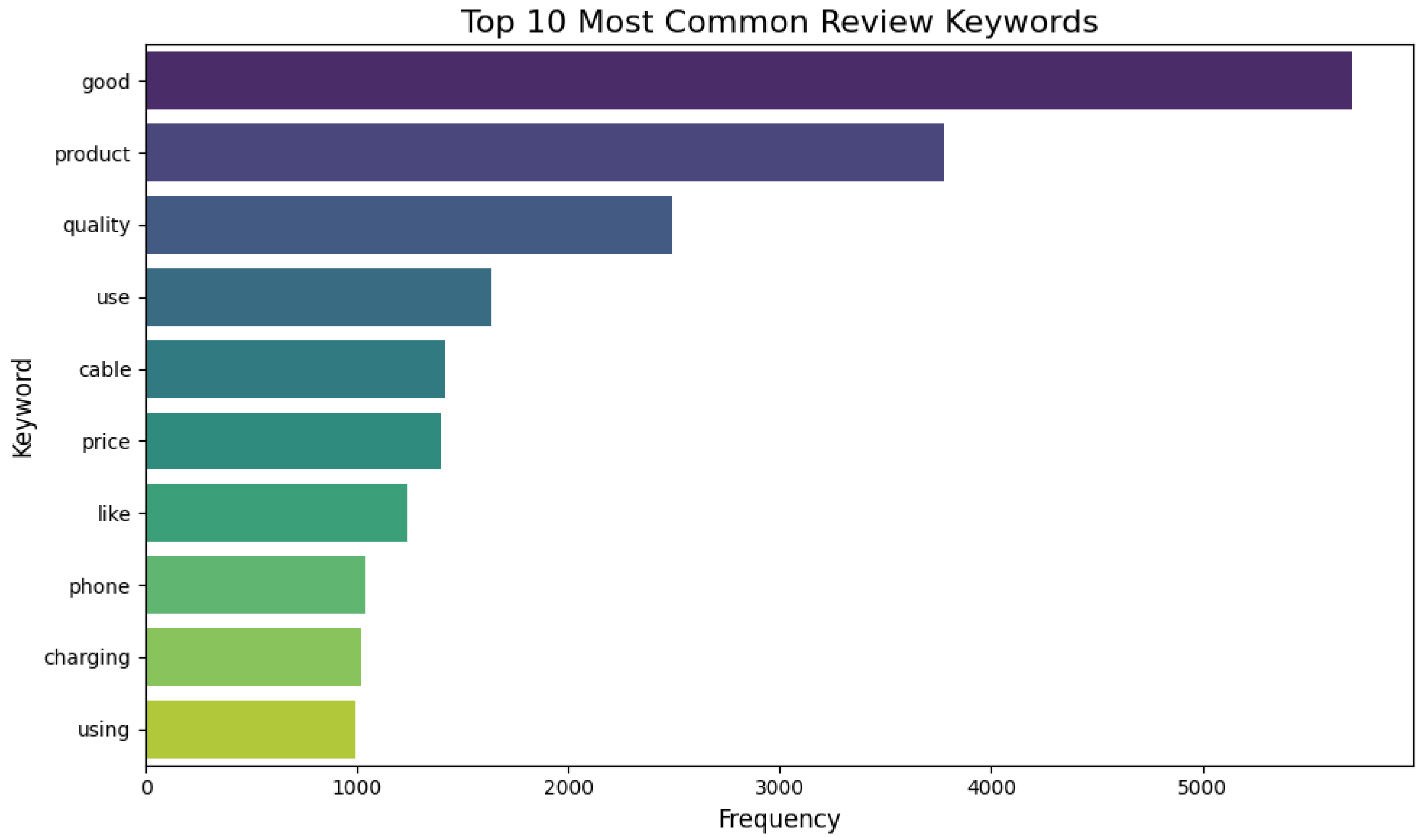
```python
wordcloud = WordCloud(stopwords=ENGLISH_STOP_WORDS, background_color="white", width=800,
height=400).generate(all_reviews_cleaned)
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Most Popular Keywords in Product Reviews', fontsize=16)
plt.axis('off')
plt.show()
```



In [92]:

```python
words = all_reviews_cleaned.split()
filtered_words = [word for word in words if word not in ENGLISH_STOP_WORDS]

review_word_counts = Counter(filtered_words)

top_review_keywords = review_word_counts.most_common(10)

top_review_keywords_df = pd.DataFrame(top_review_keywords, columns=['Keyword', 'Frequenc
y'])
print("Top 10 Most Common Review Keywords:")
print(top_review_keywords_df)

plt.figure(figsize=(10, 6))
sns.barplot(x='Frequency', y='Keyword', data=top_review_keywords_df, palette='viridis')
plt.title('Top 10 Most Common Review Keywords', fontsize=16)
plt.xlabel('Frequency', fontsize=12)
plt.ylabel('Keyword', fontsize=12)
plt.tight_layout()
plt.show()
```

```
Top 10 Most Common Review Keywords:
    Keyword  Frequency
0      good       5710
1   product       3781
2   quality       2489
3       use       1638
4     cable       1413
5     price       1395
6      like       1240
7     phone       1040
8  charging       1021
9     using        992
```



8. What is the correlation between discounted_price and rating?

In [93]:

```
data['discounted_price'].head(2)
```

Out[93]:

	discounted_price
0	399.0
1	199.0

dtype: float64

In [94]:

```
data['rating'].head(2)
```

Out[94]:

	rating
0	4.2
1	4.0

dtype: object

In [95]:

```
data['rating'] = pd.to_numeric(data['rating'], errors='coerce')
data['rating'].head(2)
```

Out[95]:

	rating
0	4.2
1	4.0

dtype: float64

In [97]:

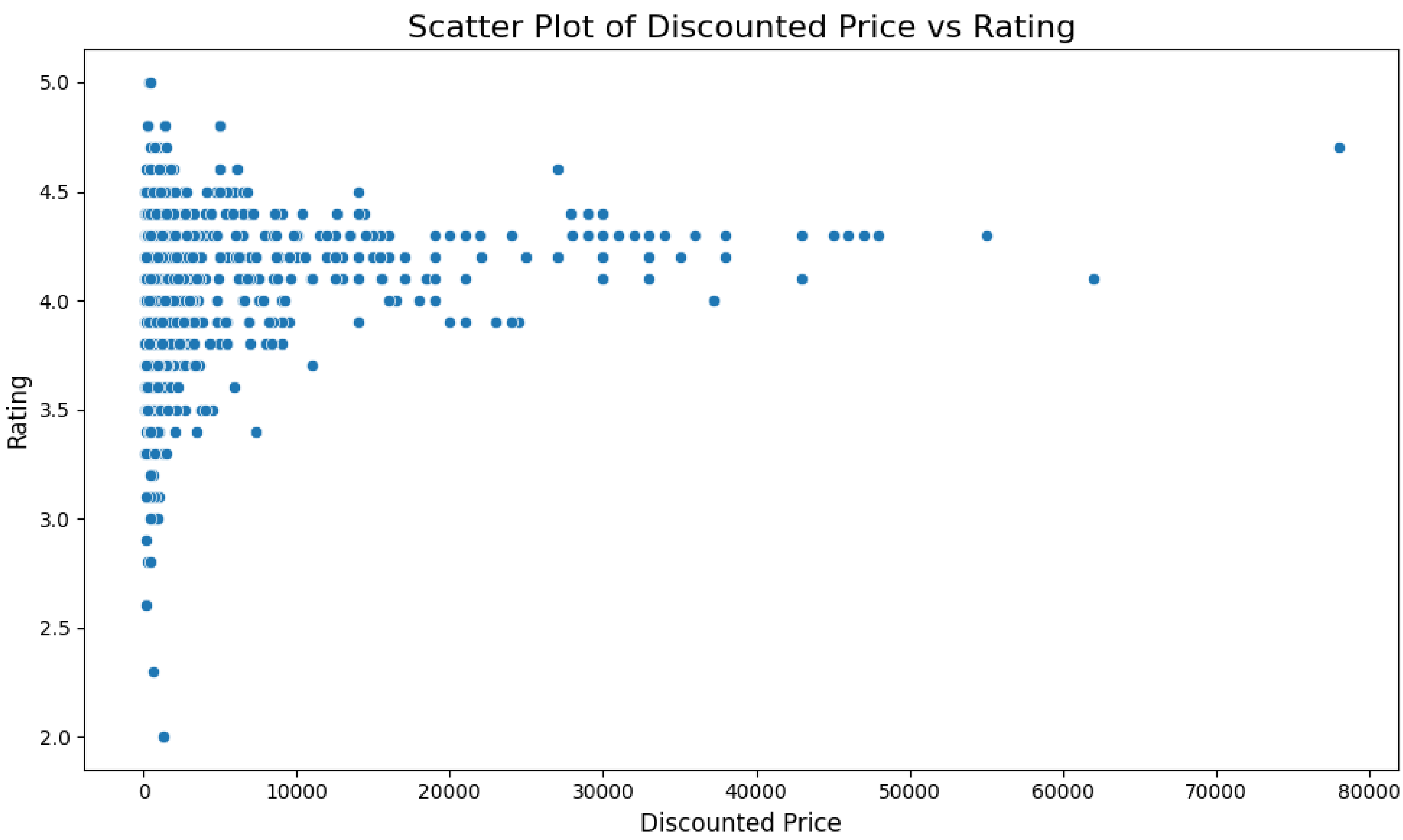
```
data_clean = data.dropna(subset=['discounted_price', 'rating'])

data_clean['discounted_price'].corr(data_clean['rating'])
```

Out[97]:

```
0.12113187526066266
```

In [98]:

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='discounted_price', y='rating', data=data_clean)
plt.title('Scatter Plot of Discounted Price vs Rating', fontsize=16)
plt.xlabel('Discounted Price', fontsize=12)
plt.ylabel('Rating', fontsize=12)
plt.tight_layout()
plt.show()
```



Conclusion:

There is a very weak positive correlation between discounted price and rating, which signifies that the discounted price does not have any good linear impoact on product rating.

9. What are the Top 5 categories based on the highest ratings?

In [99]:

```
data['rating'].head(2)
```

Out[99]:

	rating
0	4.2
1	4.0

dtype: float64

In [100]:

```
average_ratings = data.groupby('category')['rating'].mean().reset_index()
average_ratings.head(2)
```

Out[100]:

	category	rating
0	Car&Motorbike|CarAccessories|InteriorAccessori...	3.80
1	Computers&Accessories|Accessories&Peripherals|...	4.15

In [102]:

```
average_ratings.sort_values(by = 'rating', ascending= False).head(5)
```

Out[102]:

	category	rating
57	Computers&Accessories|Tablets	4.6
48	Computers&Accessories|NetworkingDevices|Networ...	4.5
62	Electronics|Cameras&Photography|Accessories|Film	4.5
81	Electronics|HomeAudio|MediaStreamingDevices|St...	4.5
196	OfficeProducts|OfficeElectronics|Calculators|B...	4.5

10. Identify any potential areas for improvement or optimization based on the data analysis.

Conclusion:

Yes, we can further check corelation between all the features using pair plot and wrt. y. so if any feature is highly correlated or derived from other features, better to drop that feature.(wrt. EDA)

Spotify Data: Popular Hip-hop Artists and Tracks (EDA)

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.

Application in Data Science:

This dataset serves as a valuable resource for various data science explorations. Analysts can delve into trend analysis to discern the popularity dynamics of hit hip-hop tracks over recent years. Additionally, the dataset enables network analysis to uncover collaborative patterns among top artists, shedding light on the genre's evolving collaborative landscape. Furthermore, it facilitates the development of predictive models aimed at

forecasting track popularity based on diverse features, offering insights for artists, producers, and marketers.

Column Descriptors:

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 [103]:

```
data = pd.read_csv('/content/spotify.csv')
data.head()
```

Out[103]:

	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 [104]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440 entries, 0 to 439
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Artist         440 non-null    object
 1   Track Name     440 non-null    object
 2   Popularity     440 non-null    int64 
 3   Duration (ms)  440 non-null    int64 
 4   Track ID       440 non-null    object
dtypes: int64(2), object(3)
memory usage: 17.3+ KB
```

Approcach 1: to track and drop duplicates from the dataframe wrt Track Name

In [118]:

```
duplicates_based_on_track_name = data.duplicated(subset=['Track Name'])
print(f"Duplicate rows based on Track Name: {duplicates_based_on_track_name.sum()}")
```

```
Duplicate rows based on Track Name: 28
```

In [120]:

```
duplicate_tracks = data[duplicates_based_on_track_name]
```

```
print("Duplicated rows based on Track Name:")
duplicate_tracks.head()
```

Duplicated rows based on Track Name:

Out[120]:

	Artist	Track Name	Popularity	Duration (ms)	Track ID
19	Noah Kahan	Dial Drunk (with Post Malone)	68	213817	5TXbpmu45lS8x0YPiUF1jy
39	Travis Scott	MELTDOWN (feat. Drake)	86	246133	67nepsnrcZkowTxMWigSbb
52	Travis Scott	TELEKINESIS (feat. SZA & Future)	86	353754	1i9lZvlaDdWDPyXEE95aiq
72	21 Savage	née-nah	88	220584	2yUzr8Sr6ldG8vmHhZwTnz
73	Drake	Jimmy Cooks (feat. 21 Savage)	88	218364	3F5CgOj3wFIRv51JsHbxhe

In [117]:

```
duplicates_based_on_columns = data.duplicated(subset=['Track Name'])
print(f"Duplicate rows based on Artist and Track Name: {duplicates_based_on_columns.sum()
}")

data[duplicates_based_on_columns == True]
```

Duplicate rows based on Artist and Track Name: 28

Out[117]:

	Artist	Track Name	Popularity	Duration (ms)	Track ID
19	Noah Kahan	Dial Drunk (with Post Malone)	68	213817	5TXbpmu45lS8x0YPiUF1jy
39	Travis Scott	MELTDOWN (feat. Drake)	86	246133	67nepsnrcZkowTxMWigSbb
52	Travis Scott	TELEKINESIS (feat. SZA & Future)	86	353754	1i9lZvlaDdWDPyXEE95aiq
72	21 Savage	née-nah	88	220584	2yUzr8Sr6ldG8vmHhZwTnz
73	Drake	Jimmy Cooks (feat. 21 Savage)	88	218364	3F5CgOj3wFIRv51JsHbxhe
76	Drake	Rich Flex	85	239359	1bDbXMyjaUIooNwFE9wn0N
131	Drake	First Person Shooter (feat. J. Cole)	88	247444	7aqfrAY2p9BUSiupwk3svU
170	Metro Boomin	Trance (with Travis Scott & Young Thug)	89	194786	5wG3HvLhF6Y5KTGlK0IW3J
182	D-Block Europe	Overseas	74	222154	337kcYVjYXdLBltCw9ry3b
210	Post Malone	Sunflower - Spider-Man: Into the Spider-Verse	87	157560	0RiRZpuVRbi7oqRdSMwhQY
212	Metro Boomin	Annihilate (Spider-Man: Across the Spider-Vers...	79	231746	39MK3d3fonIP8Mz9oHCTBB
222	Cardi B	WAP (feat. Megan Thee Stallion)	80	187541	4Oun2ylbjFKMPTiaSbbCih
224	Cardi B	Bongos (feat. Megan Thee Stallion)	78	175099	4YQlmHflXSilMXntcwPkx8
242	Bizarrap	Quevedo: Bzrp Music Sessions, Vol. 52	87	198937	2tTmW7RDtMQtBk7m2rYeSw
270	Lil Durk	All My Life (feat. J. Cole)	74	223878	6T7FXSuXykeGktMLGp8WgE
280	¥$	CARNIVAL	96	264324	3w0w2T288dec0mgeZZqoNN
282	Travis Scott	FE!N (feat. Playboi Carti)	93	191700	42VsgItocQwOQC3XWZ8JNA
290	Lil Baby	Drip Too Hard (Lil Baby & Gunna)	85	145542	78QR3Wp35dqAhFEc2qAGjE
297	Quality Control	Baby (Lil Baby feat. DaBaby)	77	142417	5MPPttjfGap2C6j6eKcO6J
310	Lil Nas X	INDUSTRY BABY (feat. Jack Harlow)	78	212352	5Z9KJZvQzH6PFmb8SNkxuk
331	Nicki Minaj	Everybody (feat. Lil Uzi Vert)	84	180869	5ZJGv7aGdlr9lGpxzSG18T
341	Snoop Dogg	Young, Wild & Free (feat. Bruno Mars)	70	207346	6YbhspuOar1D9WSSnfe7ds
343	Lil Wayne	Sucker for Pain (with Wiz Khalifa, Imagine Dra...	77	243490	4dASQiO1Eoo3RJvt74FtXB
352	Nicki Minaj	Barbie World (with Aqua) [From Barbie The Album]	83	109750	741UUVE2kuITl0c6zuqqbO

	Artist	Track Name	Popularity	Duration (ms)	Track ID
422	Metro Boomin	Annihilate (Spider-Man: Across the Spider-Vers...	79	231746	39MK3d3fonIP8Mz9oHCTBB
430	French Montana	Unforgettable	87	233901	3B54sVLJ402zGa6Xm4YGNe
435	French Montana	Splash Brothers	44	221863	3fBsEOnzwtlkpS0LxXAZhN
439	Rick Ross	Stay Schemin	68	267720	0nq6sfr8z1R5KJ4XUk396e

Approach 2: Custom function to filter duplicate

In [123]:

```
def check_duplicates_by_track_name(data):
    data['duplicated_data'] = False

    for idx, row in data.iterrows():
        track_name = row['Track Name']

        if data[data['Track Name'] == track_name].shape[0] > 1:
            data.at[idx, 'duplicated_data'] = True

    return data
```

In [124]:

```
duplicated_data = check_duplicates_by_track_name(data)

duplicated_data.value_counts()
```

Out[124]:

						count
Artist	**Track Name**	**Popularity**	**Duration (ms)**	**Track ID**	**duplicated_data**	
Metro Boomin	Annihilate (Spider-Man: Across the Spider-Verse) (Metro Boomin & Swae Lee, Lil Wayne, Offset)	79	231746	39MK3d3fonIP8Mz9oHCTBB	True	3
D-Block Europe	Overseas	74	222154	337kcYVjYXdLBltCw9ry3b	True	2
Drake	Jimmy Cooks (feat. 21 Savage)	88	218364	3F5CgOj3wFIRv51JsHbxhe	True	2
Lil Nas X	INDUSTRY BABY (feat. Jack Harlow)	78	212352	5Z9KJZvQzH6PFmb8SNkxuk	True	2
Drake	Rich Flex	85	239359	1bDbXMyjaUlooNwFE9wn0N	True	2
...	...	...	...	...	...	...
Gunna	Drip or Drown	78	126168	6ZthdsKjWtiCxnxbhs74vF	False	1
	Bittersweet	74	191493	7yfRb4seXT7w8zVMW0dXNa	False	1
Gucci Mane	Wake Up in the Sky	76	203161	2G1tXoGBaEMJ7FKGnkf6ud	False	1
GloRilla	Tomorrow 2 (with Cardi B)	73	209811	0WNfQxDGaPTI0yogcMR5v1	False	1
¥$	VULTURES	80	276986	3SIRBp4RRQ2AO5H4NO7xfq	False	1

413 rows × 1 columns

dtype: int64

In [126]:

```
data.shape
```

Out[126]:

(440, 6)

```
(440, 6)
```

In [127]:

```
data = data[data['duplicated_data'] == False]
data.shape
```

Out[127]:

```
(385, 6)
```

Conclusion:

1. **By taking Track Name as the subset, 55 data points came as duplicated.**
2. **By dropping duplicate data points, the dataset got decreased by ~35%.**
3. **No Missing value.**

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

In [128]:

```
data['Popularity'].head(2)
```

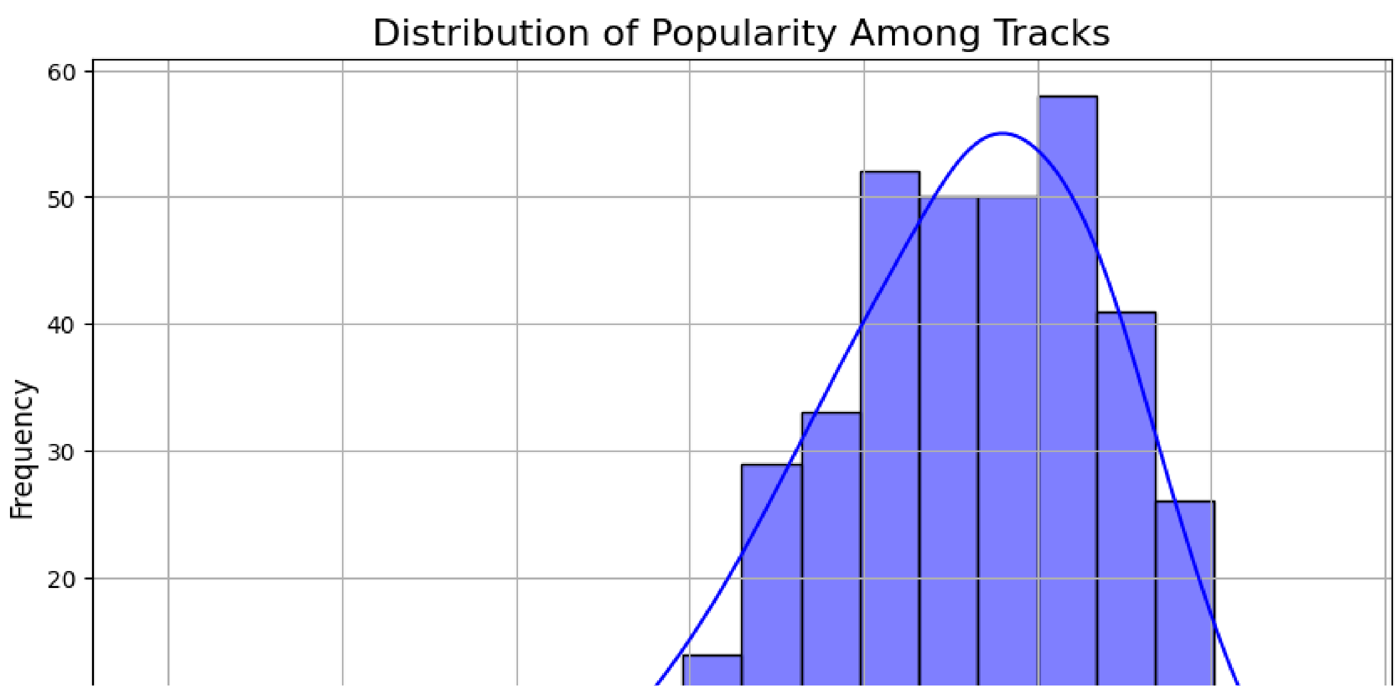
Out[128]:

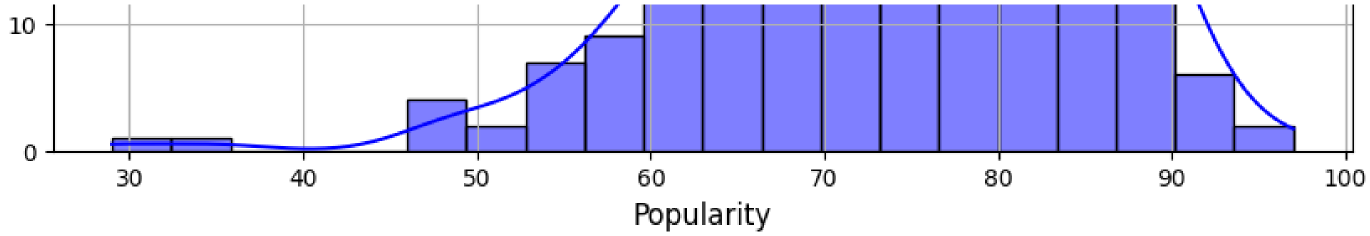
	Popularity
0	92
1	91

dtype: int64

In [129]:

```
plt.figure(figsize=(10, 6))
sns.histplot(data['Popularity'], bins=20, kde=True, color='blue')

plt.title('Distribution of Popularity Among Tracks', fontsize=16)
plt.xlabel('Popularity', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.grid(True)
plt.show()
```





Conclusion:

1. **It seems like normal gaussian but is slighlty left or negative skewed, may be the points less than 40 are outliers which can be determined using box plot etc.**
2. **Majority of popularity is coming in between 75 and 80.**

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

In [130]:

```
data['Duration (ms)'].head(1)
```

Out[130]:

	Duration (ms)
0	319191

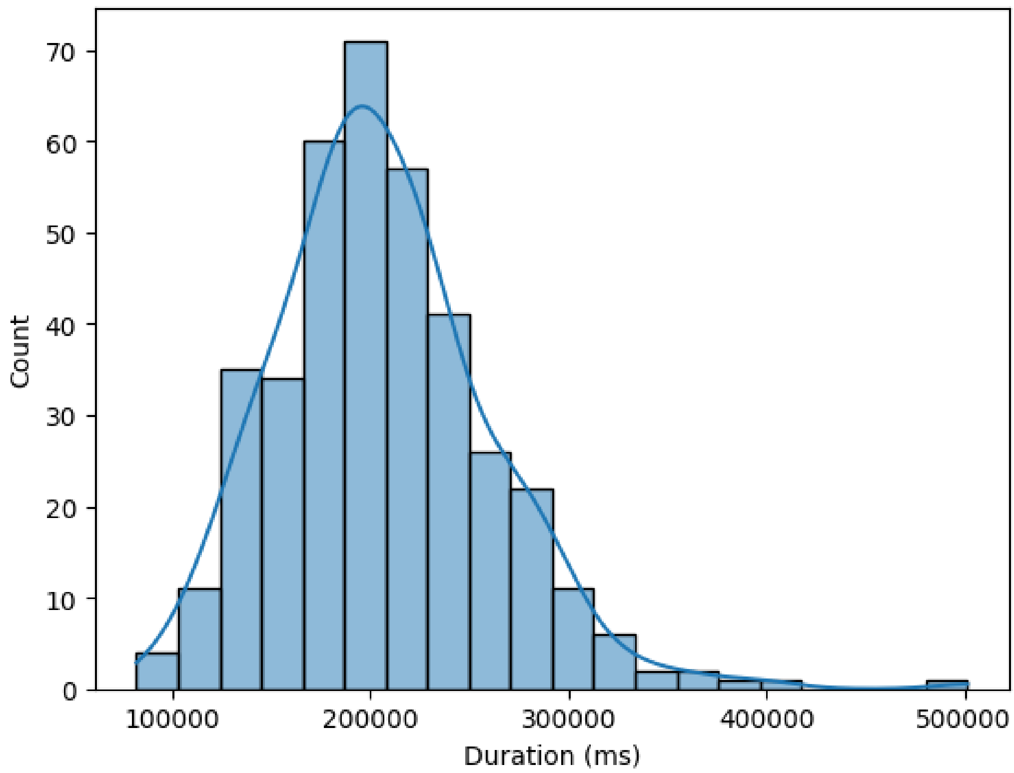
dtype: int64

In [134]:

```
sns.histplot(data['Duration (ms)'], bins=20, kde=True)
```

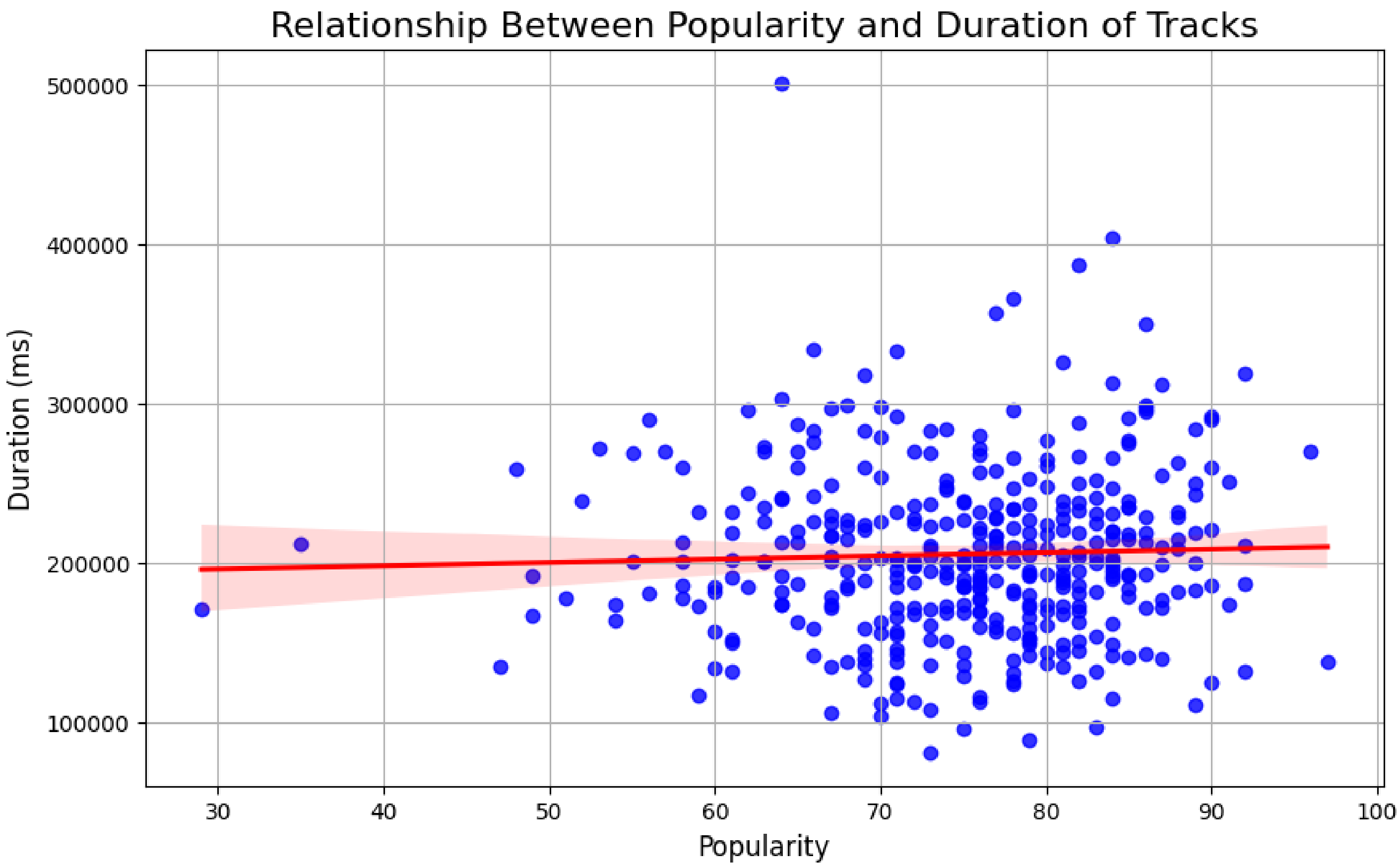
Out[134]:

<Axes: xlabel='Duration (ms)', ylabel='Count'>



In [133]:

```
plt.figure(figsize=(10, 6))
sns.regplot(data=data, x='Popularity', y='Duration (ms)', scatter_kws={'color': 'blue'},
line_kws={'color': 'red'})

plt.title('Relationship Between Popularity and Duration of Tracks', fontsize=16)
plt.xlabel('Popularity', fontsize=12)
plt.ylabel('Duration (ms)', fontsize=12)
plt.grid(True)
plt.show()
```



In [136]:

```
correlation = data['Popularity'].corr(data['Duration (ms)'])
print(f"Pearson correlation coefficient between Popularity and Duration (ms): {correlatio
n}")
```

```
Pearson correlation coefficient between Popularity and Duration (ms): 0.036746897758849 61
```

Conclusion

As the line is almost straight or a very little moving up states that there is no or very slight correlation between both the features.

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

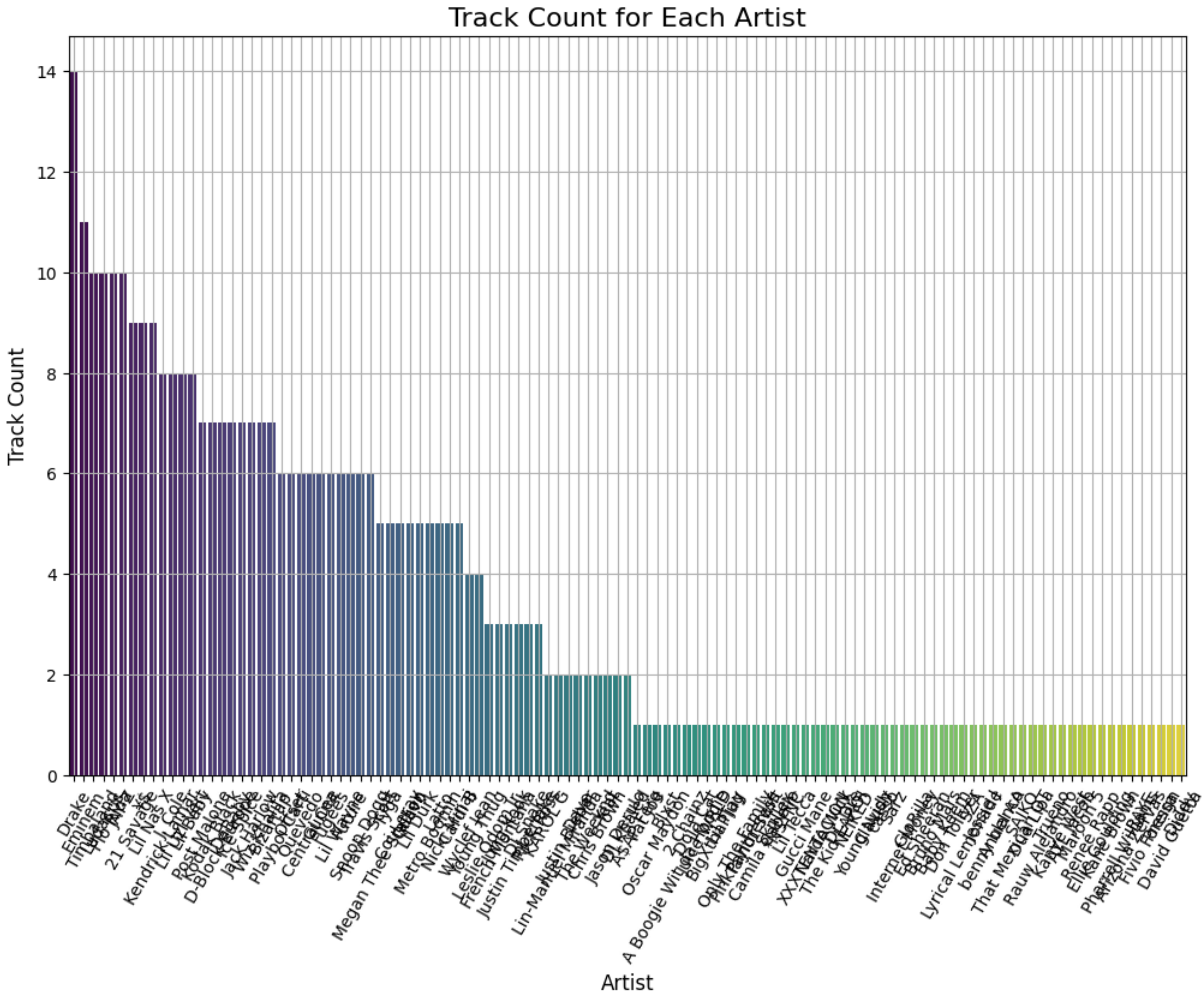
In [137]:

```
artist_track_count = data['Artist'].value_counts().reset_index()
artist_track_count.columns = ['Artist', 'Track Count']

print(f"Artist with the highest number of tracks: {artist_track_count.iloc[0]['Artist']}
({artist_track_count.iloc[0]['Track Count']} tracks)")
```

```
Artist with the highest number of tracks: Drake (14 tracks)
```

In [141]:

```
plt.figure(figsize=(12, 8))
sns.countplot(data=data, x='Artist', order=data['Artist'].value_counts().index, palette=
'viridis')

plt.title('Track Count for Each Artist', fontsize=16)
plt.xlabel('Artist', fontsize=12)
plt.ylabel('Track Count', fontsize=12)
plt.xticks(rotation=60)  # Rotate the x-axis labels for better readability
plt.grid(True)
plt.show()
```



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

In [142]:

```
least_poluar_tracks = data.sort_values(by='Popularity', ascending=True).head(5)
least_poluar_tracks[['Artist', 'Track Name', 'Popularity']]
```

Out[142]:

	Artist	Track Name	Popularity
207	Pressa	Attachments (feat. Coi Leray)	29
231	Justin Bieber	Intentions	35
225	Lil Baby	On Me - Remix	47
407	Wyclef Jean	911 (feat. Mary J. Blige)	48

	Artist	Track Name	Popularity
431	French Montana	Okay	49

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 [146]:

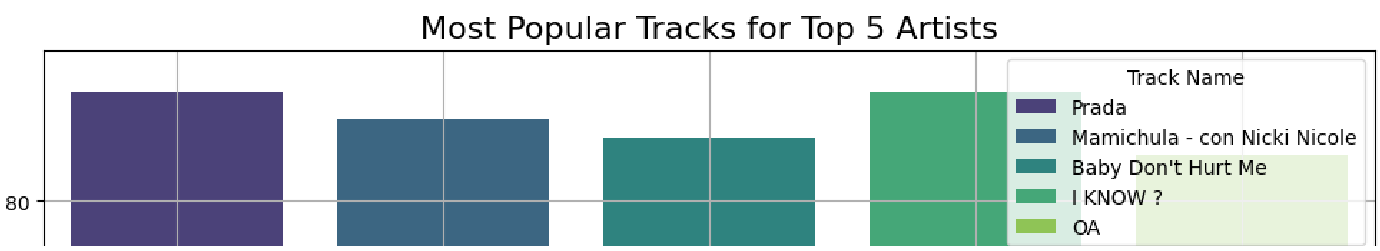
```
artist_popularity = data.groupby('Artist')['Popularity'].mean().reset_index()

top_5_artists = artist_popularity.sort_values(by='Popularity', ascending=False).head(5)

print("Top 5 most popular artists based on average popularity:")
print(top_5_artists)
```

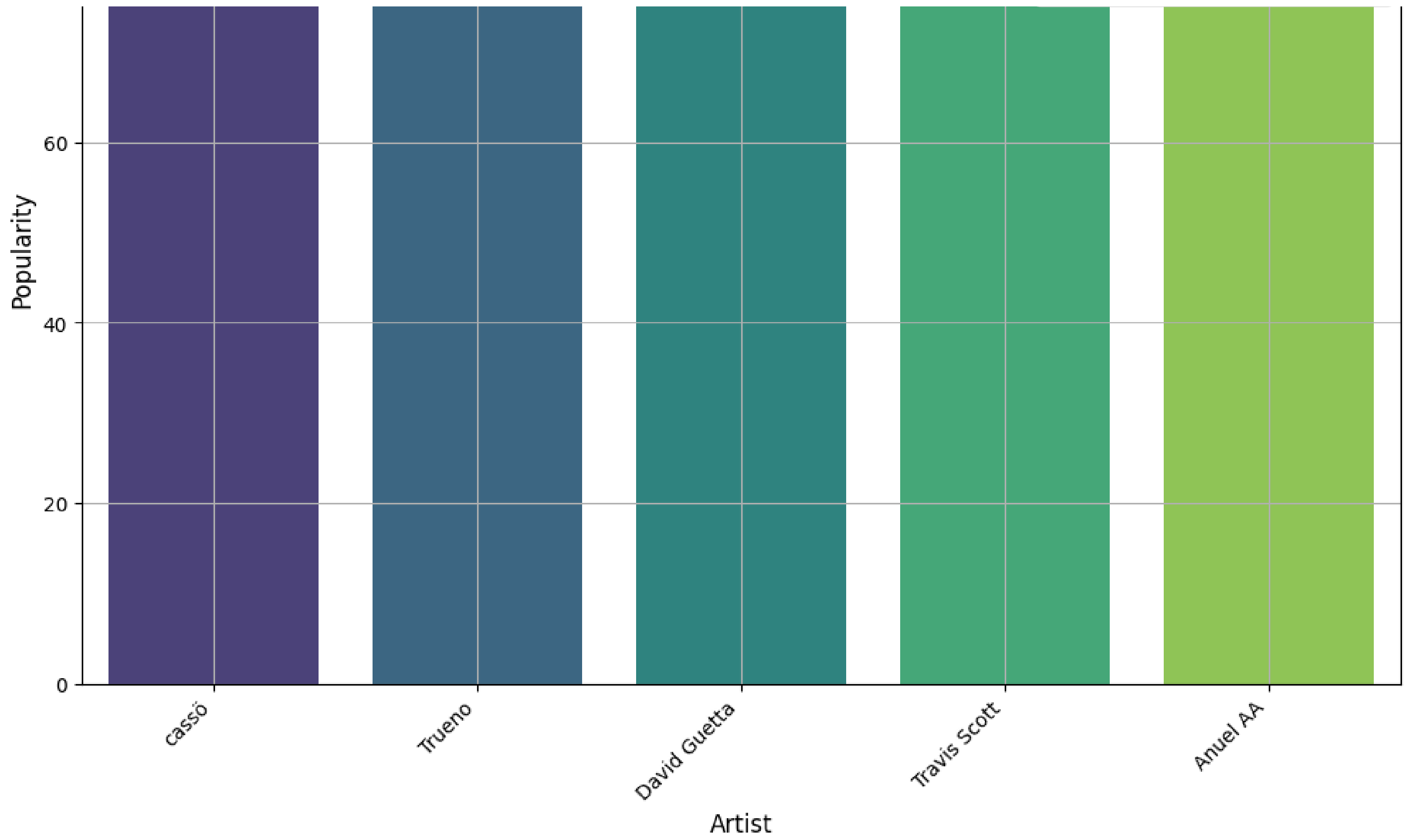
```
Top 5 most popular artists based on average popularity:
           Artist  Popularity
111         cassö   92.000000
102        Trueno   89.000000
24   David Guetta   87.000000
101  Travis Scott   85.666667
6         Anuel AA   85.000000
```

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

In [148]:

```
artist_popularity = data.groupby('Artist')['Popularity'].mean().reset_index()

top_5_artists = artist_popularity.sort_values(by='Popularity', ascending=False).head(5)

most_popular_tracks = []

for artist in top_5_artists['Artist']:
    artist_data = data[data['Artist'] == artist]

    most_popular_track = artist_data.loc[artist_data['Popularity'].idxmax()]
    most_popular_tracks.append({
        'Artist': artist,
        'Track Name': most_popular_track['Track Name'],
        'Popularity': most_popular_track['Popularity']
    })

most_popular_tracks_df = pd.DataFrame(most_popular_tracks)

plt.figure(figsize=(12, 8))
sns.barplot(data=most_popular_tracks_df, x='Artist', y='Popularity', hue='Track Name', p
alette='viridis')

plt.title('Most Popular Tracks for Top 5 Artists', fontsize=16)
plt.xlabel('Artist', fontsize=12)
plt.ylabel('Popularity', fontsize=12)
plt.xticks(rotation=45, ha='right')  # Rotate the x-axis labels for better readability
plt.legend(title='Track Name', loc='upper right', bbox_to_anchor=(1, 1))
plt.grid(True)
plt.show()
```





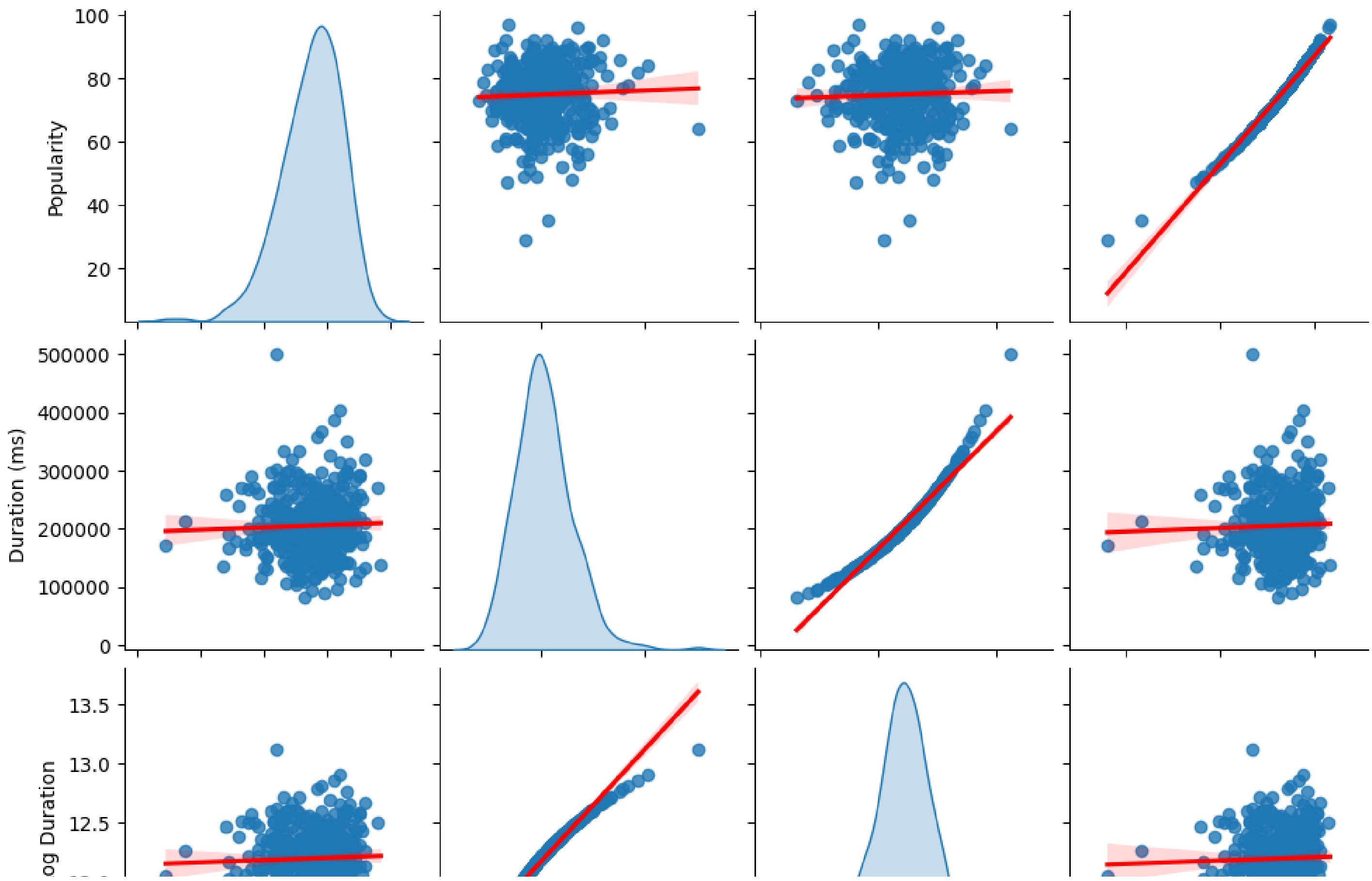
8.Visualize relationships between multiple numerical variables simultaneously using a pair plot.

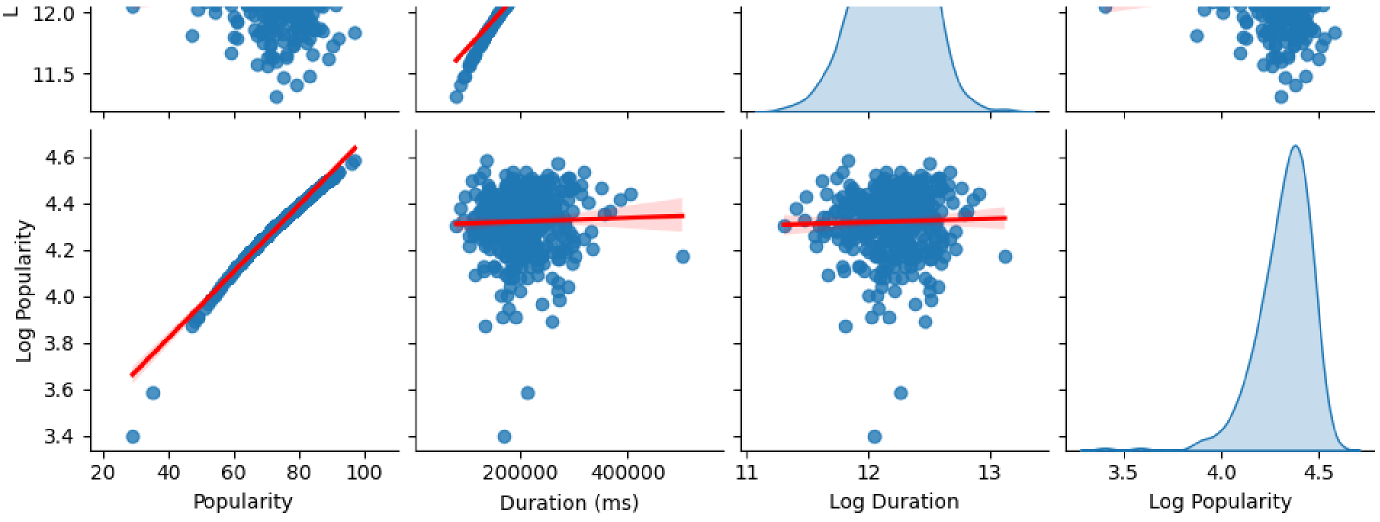
In [153]:

```
sns.pairplot(data[['Popularity', 'Duration (ms)', 'Log Duration', 'Log Popularity']], dia
g_kind='kde', kind='reg', plot_kws={'line_kws':{'color':'red'}})
```

Out[153]:

```
<seaborn.axisgrid.PairGrid at 0x7a9e2a0f6080>
```

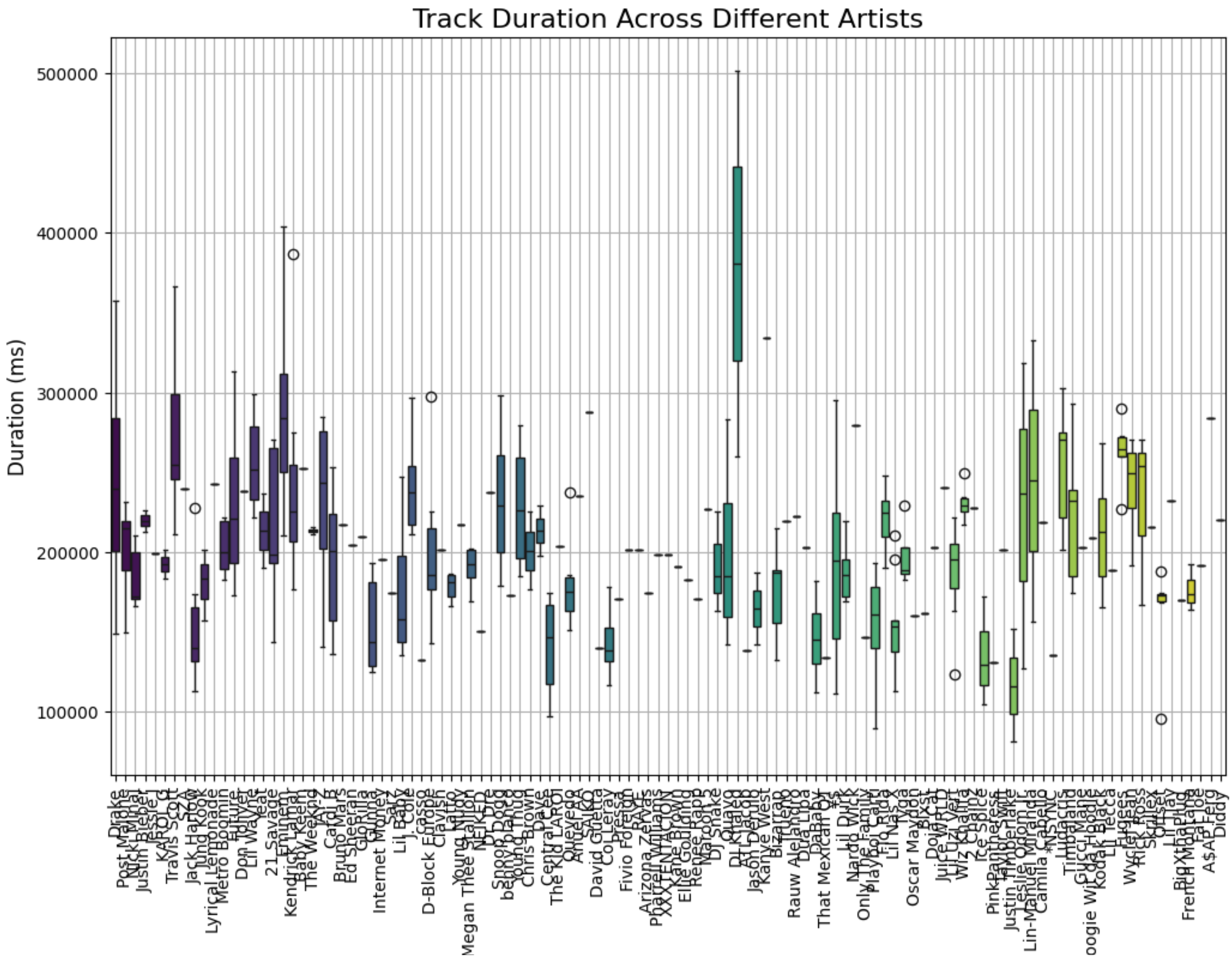




9.Does the duration of tracks vary significantly across different artists? Explore this visually using a box plot or violin plot.

In [154]:

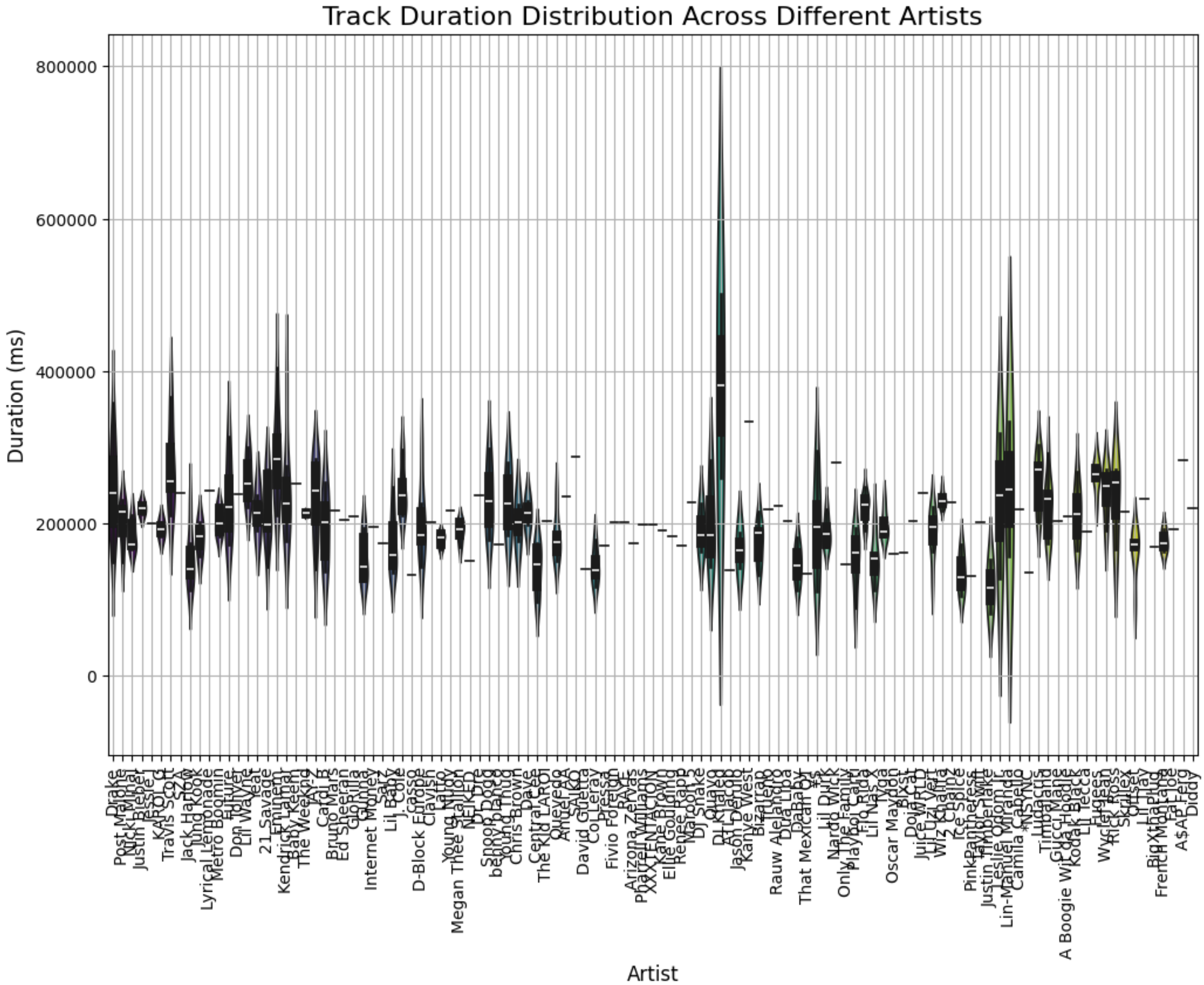
```
plt.figure(figsize=(12, 8))
sns.boxplot(data=data, x='Artist', y='Duration (ms)', palette='viridis')

# Customize the plot
plt.title('Track Duration Across Different Artists', fontsize=16)
plt.xlabel('Artist', fontsize=12)
plt.ylabel('Duration (ms)', fontsize=12)
plt.xticks(rotation=90)  # Rotate the x-axis labels for better readability
plt.grid(True)
plt.show()
```



Artist

In [155]:

```
plt.figure(figsize=(12, 8))
sns.violinplot(data=data, x='Artist', y='Duration (ms)', palette='viridis')

# Customize the plot
plt.title('Track Duration Distribution Across Different Artists', fontsize=16)
plt.xlabel('Artist', fontsize=12)
plt.ylabel('Duration (ms)', fontsize=12)
plt.xticks(rotation=90)  # Rotate the x-axis labels for better readability
plt.grid(True)
plt.show()
```



Conclusion:

1. using box-plot is easily understandable that the duration of track is varying across different artist.
2. althought for most of the artist its in between 100000 ms and 220000 ms but for some its even more than 300000 ms (possibility of outliers).
3. There are some overlap at starting part but at ending part, significant difference can be seen.

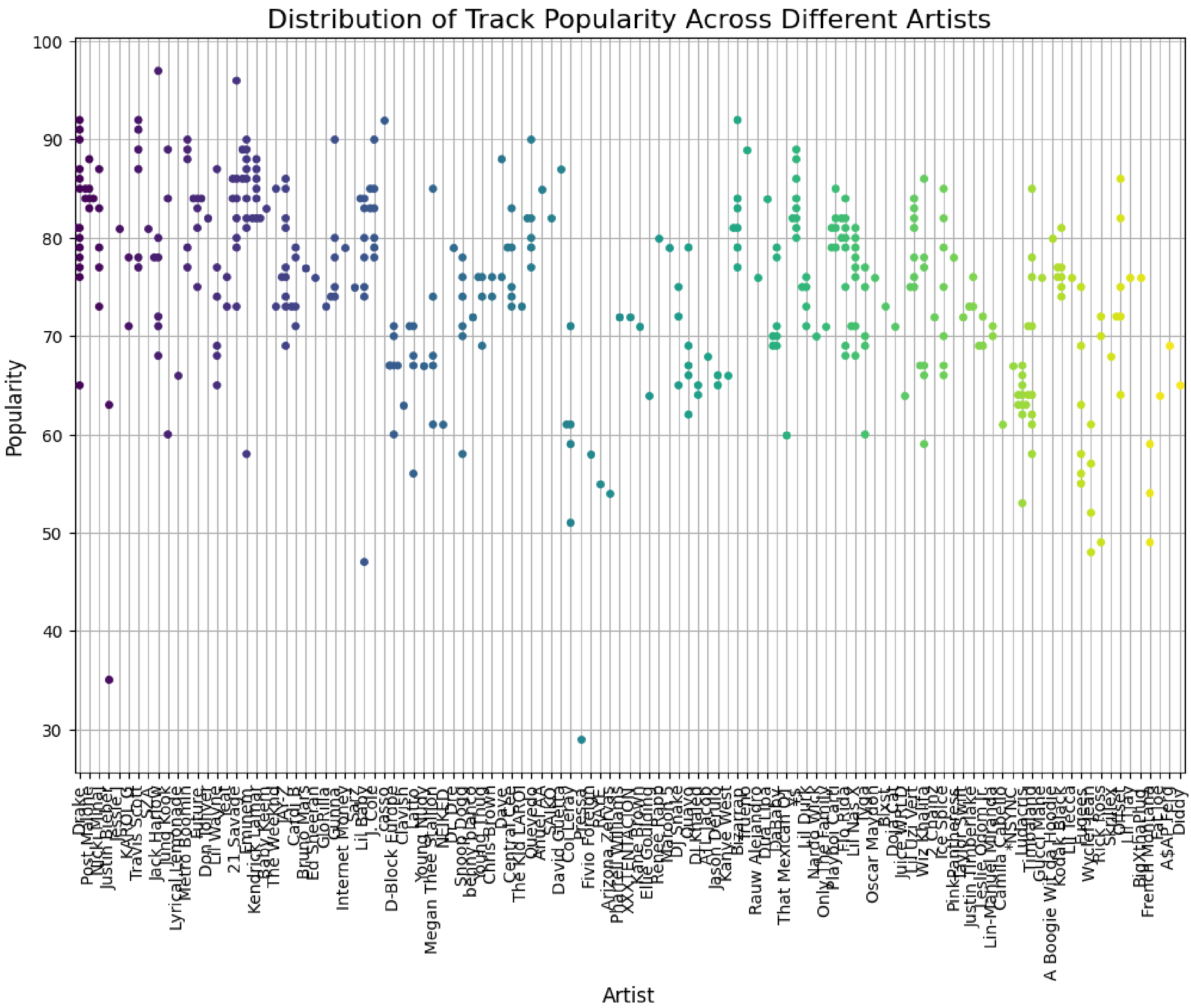
YES.

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

plot

In [157]:

```
plt.figure(figsize=(12, 8))
sns.swarmplot(data=data, x='Artist', y='Popularity', palette='viridis')

# Customize the plot
plt.title('Distribution of Track Popularity Across Different Artists', fontsize=16)
plt.xlabel('Artist', fontsize=12)
plt.ylabel('Popularity', fontsize=12)
plt.xticks(rotation=90)  # Rotate the x-axis labels for better readability
plt.grid(True)
plt.show()
```



Conclusion:

1. Most of the artist are having popularity in between 50 and 90.
2. A few, top artist are having popularity more than 90.

In [ ]: