[1]:
```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
# 1. Data Exploration
# Load the dataset
df=  pd.read_csv('youtubers_df.csv')
df
```

[2]:

	Rank	Username	Categories	Suscribers	Country	Visits	Likes	Comments	Links
0	1	tseries	Música y baile	249500000.0	India	86200.0	2700.0	78.0	http://youtube.com/channel/UCq-Fj5jknLsUf-MWSy...
1	2	MrBeast	Videojuegos, Humor	183500000.0	Estados Unidos	117400000.0	5300000.0	18500.0	http://youtube.com/channel/UCX6OQ3DkcsbYNE6H8u...
2	3	CoComelon	Educación	165500000.0	Unknown	7000000.0	24700.0	0.0	http://youtube.com/channel/UCbCmjCuTUZos6Inko4...
3	4	SETIndia	NaN	162600000.0	India	15600.0	166.0	9.0	http://youtube.com/channel/UCpEhnqL0y41EpW2TvW...
4	5	KidsDianaShow	Animación, Juguetes	113500000.0	Unknown	3900000.0	12400.0	0.0	http://youtube.com/channel/UCk8GzjMOrta8yxDcKf...
...	...	...	...	...	...	...	...	...	...
995	996	hamzymukbang	NaN	11700000.0	Estados Unidos	397400.0	14000.0	124.0	http://youtube.com/channel/UCPKNKldggioffXPkSm...
996	997	Adaahqueen	NaN	11700000.0	India	1100000.0	92500.0	164.0	http://youtube.com/channel/UCk3fFpqI5kDMf__mUP...
997	998	LittleAngelIndonesia	Música y baile	11700000.0	Unknown	211400.0	745.0	0.0	http://youtube.com/channel/UCdrHrQf0o0TO8YDntX...

[2]:
```
# 1. Data Exploration
# Load the dataset
df=  pd.read_csv('youtubers_df.csv')
df
```

[2]:

	Rank	Username	Categories	Suscribers	Country	Visits	Likes	Comments	Links
0	1	tseries	Música y baile	249500000.0	India	86200.0	2700.0	78.0	http://youtube.com/channel/UCq-Fj5jknLsUf-MWSy...
1	2	MrBeast	Videojuegos, Humor	183500000.0	Estados Unidos	117400000.0	5300000.0	18500.0	http://youtube.com/channel/UCX6OQ3DkcsbYNE6H8u...
2	3	CoComelon	Educación	165500000.0	Unknown	7000000.0	24700.0	0.0	http://youtube.com/channel/UCbCmjCuTUZos6Inko4...
3	4	SETIndia	NaN	162600000.0	India	15600.0	166.0	9.0	http://youtube.com/channel/UCpEhnqL0y41EpW2TvW...
4	5	KidsDianaShow	Animación, Juguetes	113500000.0	Unknown	3900000.0	12400.0	0.0	http://youtube.com/channel/UCk8GzjMOrta8yxDcKf...
...	...	...	...	...	...	...	...	...	...
995	996	hamzymukbang	NaN	11700000.0	Estados Unidos	397400.0	14000.0	124.0	http://youtube.com/channel/UCPKNKldggioffXPkSm...
996	997	Adaahqueen	NaN	11700000.0	India	1100000.0	92500.0	164.0	http://youtube.com/channel/UCk3fFpqI5kDMf__mUP...
997	998	LittleAngelIndonesia	Música y baile	11700000.0	Unknown	211400.0	745.0	0.0	http://youtube.com/channel/UCdrHrQf0o0TO8YDntX...
998	999	PenMultiplex	NaN	11700000.0	India	14000.0	81.0	1.0	http://youtube.com/channel/UCObyBrdrtQ20BU9PxH...
999	1000	OneindiaHindi	Noticias y Política	11700000.0	India	2200.0	31.0	1.0	http://youtube.com/channel/UCOjgc1p2hJ4GZi6pQQ...

1000 rows × 9 columns

[3]:
```
# Display the first few rows
```

[3]:
```
# Display the first few rows
df.head()
```

[3]:

	Rank	Username	Categories	Suscribers	Country	Visits	Likes	Comments	Links
0	1	tseries	Música y baile	249500000.0	India	86200.0	2700.0	78.0	http://youtube.com/channel/UCq-Fj5jknLsUf-MWSy...
1	2	MrBeast	Videojuegos, Humor	183500000.0	Estados Unidos	117400000.0	5300000.0	18500.0	http://youtube.com/channel/UCX6OQ3DkcsbYNE6H8u...
2	3	CoComelon	Educación	165500000.0	Unknown	7000000.0	24700.0	0.0	http://youtube.com/channel/UCbCmjCuTUZos6Inko4...
3	4	SETIndia	NaN	162600000.0	India	15600.0	166.0	9.0	http://youtube.com/channel/UCpEhnqL0y41EpW2TvW...
4	5	KidsDianaShow	Animación, Juguetes	113500000.0	Unknown	3900000.0	12400.0	0.0	http://youtube.com/channel/UCk8GzjMOrta8yxDcKf...

[4]:
```
# Display the last few rows
df.tail()
```

[4]:

	Rank	Username	Categories	Suscribers	Country	Visits	Likes	Comments	Links
995	996	hamzymukbang	NaN	11700000.0	Estados Unidos	397400.0	14000.0	124.0	http://youtube.com/channel/UCPKNKldggioffXPkSm...
996	997	Adaahqueen	NaN	11700000.0	India	1100000.0	92500.0	164.0	http://youtube.com/channel/UCk3fFpqI5kDMf__mUP...
997	998	LittleAngelIndonesia	Música y baile	11700000.0	Unknown	211400.0	745.0	0.0	http://youtube.com/channel/UCdrHrQf0o0TO8YDntX...
998	999	PenMultiplex	NaN	11700000.0	India	14000.0	81.0	1.0	http://youtube.com/channel/UCObyBrdrtQ20BU9PxH...
999	1000	OneindiaHindi	Noticias y Política	11700000.0	India	2200.0	31.0	1.0	http://youtube.com/channel/UCOjgc1p2hJ4GZi6pQQ...

[5]:
```
# Check the structure
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

[5]:
```
# Check the structure
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Rank        1000 non-null   int64
 1   Username    1000 non-null   object
 2   Categories  694 non-null    object
 3   Suscribers  1000 non-null   float64
 4   Country     1000 non-null   object
 5   Visits      1000 non-null   float64
 6   Likes       1000 non-null   float64
 7   Comments    1000 non-null   float64
 8   Links       1000 non-null   object
dtypes: float64(4), int64(1), object(4)
memory usage: 70.4+ KB
```

[6]:
```
# Get summary statistics
df.describe()
```

[6]:

	Rank	Suscribers	Visits	Likes	Comments
count	1000.000000	1.000000e+03	1.000000e+03	1.000000e+03	1000.000000
mean	500.500000	2.189440e+07	1.209446e+06	5.363259e+04	1288.768000
std	288.819436	1.682775e+07	5.229942e+06	2.580457e+05	6778.188308
min	1.000000	1.170000e+07	0.000000e+00	0.000000e+00	0.000000
25%	250.750000	1.380000e+07	3.197500e+04	4.717500e+02	2.000000
50%	500.500000	1.675000e+07	1.744500e+05	3.500000e+03	67.000000

[6]:
```
# Get summary statistics
df.describe()
```

[6]:

	Rank	Suscribers	Visits	Likes	Comments
count	1000.000000	1.000000e+03	1.000000e+03	1.000000e+03	1000.000000
mean	500.500000	2.189440e+07	1.209446e+06	5.363259e+04	1288.768000
std	288.819436	1.682775e+07	5.229942e+06	2.580457e+05	6778.188308
min	1.000000	1.170000e+07	0.000000e+00	0.000000e+00	0.000000
25%	250.750000	1.380000e+07	3.197500e+04	4.717500e+02	2.000000
50%	500.500000	1.675000e+07	1.744500e+05	3.500000e+03	67.000000
75%	750.250000	2.370000e+07	8.654750e+05	2.865000e+04	472.000000
max	1000.000000	2.495000e+08	1.174000e+08	5.300000e+06	154000.000000

[7]:
```
df.columns
```

[7]:
```
Index(['Rank', 'Username', 'Categories', 'Suscribers', 'Country', 'Visits',
       'Likes', 'Comments', 'Links'],
      dtype='object')
```

[8]:
```
# Check for missing values
df.isnull().sum()
```

[8]:
```
Rank            0
Username        0
Categories    306
Suscribers      0
```

[8]:
```
# Check for missing values
df.isnull().sum()
```

[8]:
```
Rank           0
Username       0
Categories    306
Suscribers     0
Country        0
Visits         0
Likes          0
Comments       0
Links          0
dtype: int64
```

[9]:
```
# Handle missing data and outliers
# Drop rows with missing values
df.dropna(inplace=True)
```

[10]:
```
# Check again for missing values
missing_values_after = df.isnull().sum()
print("Missing values after cleaning:")
print(missing_values_after)
```

```
Missing values after cleaning:
Rank          0
Username      0
Categories    0
Suscribers    0
Country       0
Visits        0
Likes         0
Comments      0
Links         0
```

[10]:
```
# Check again for missing values
missing_values_after = df.isnull().sum()
print("Missing values after cleaning:")
print(missing_values_after)
```

```
Missing values after cleaning:
Rank          0
Username      0
Categories    0
Suscribers    0
Country       0
Visits        0
Likes         0
Comments      0
Links         0
dtype: int64
```

[11]:
```
df.nunique()
```

[11]:
```
Rank          694
Username      689
Categories     45
Suscribers    234
Country        27
Visits        530
Likes         479
Comments      310
Links         689
dtype: int64
```

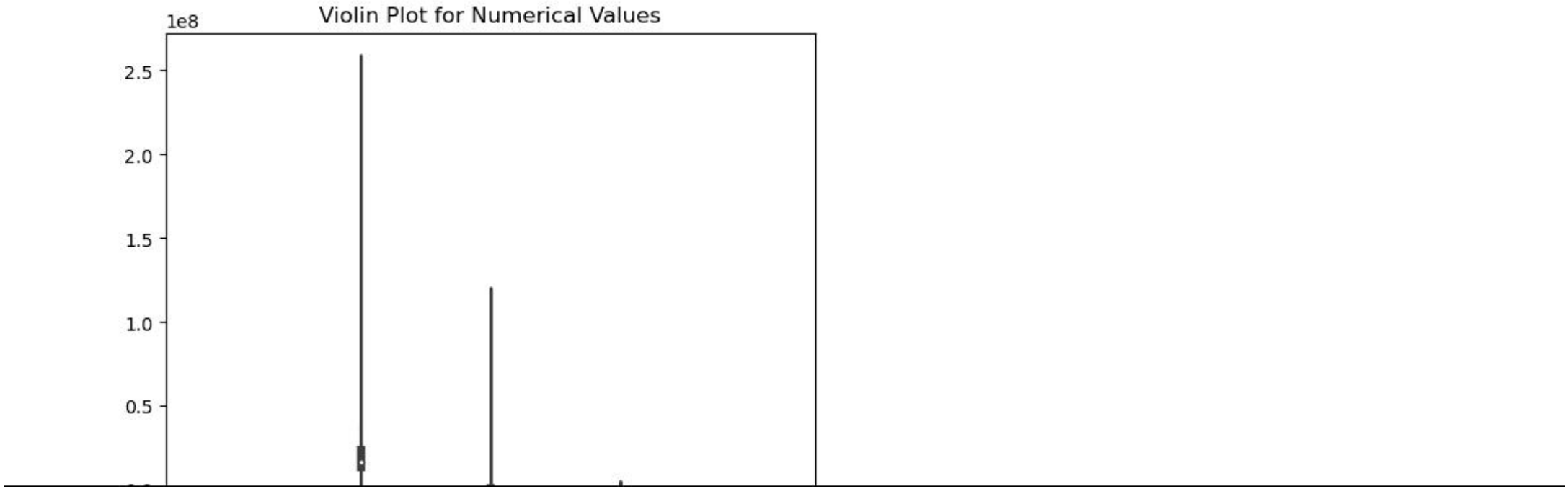
[13]:
```
df.shape
```

[13]:
```
(694, 9)
```

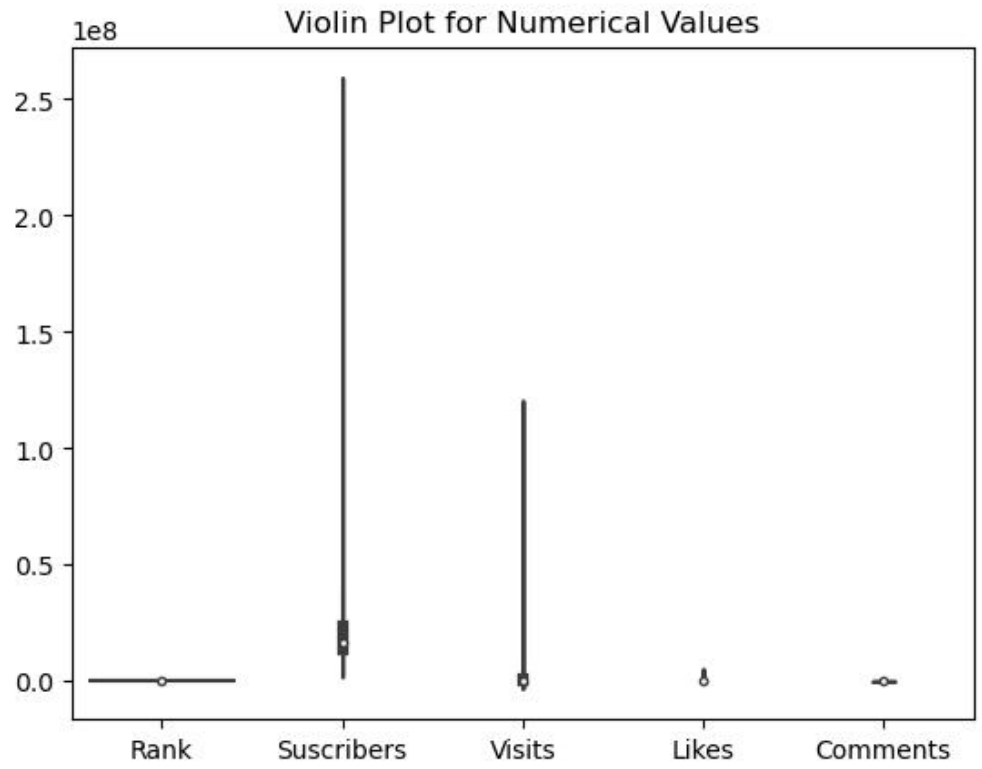
[14]:
```
df.index
```

[14]:
```
Index([  0,   1,   2,   4,   5,   6,   7,   8,   9,  10,
       ...
       983, 984, 985, 987, 988, 989, 990, 991, 997, 999],
      dtype='int64', length=694)
```

[15]:
```
# Check for outliers using violin plots
sns.violinplot(data=df)
plt.title("Violin Plot for Numerical Values")
plt.show()
```



[15]:
```
# Check for outliers using violin plots
sns.violinplot(data=df)
plt.title("Violin Plot for Numerical Values")
plt.show()
```

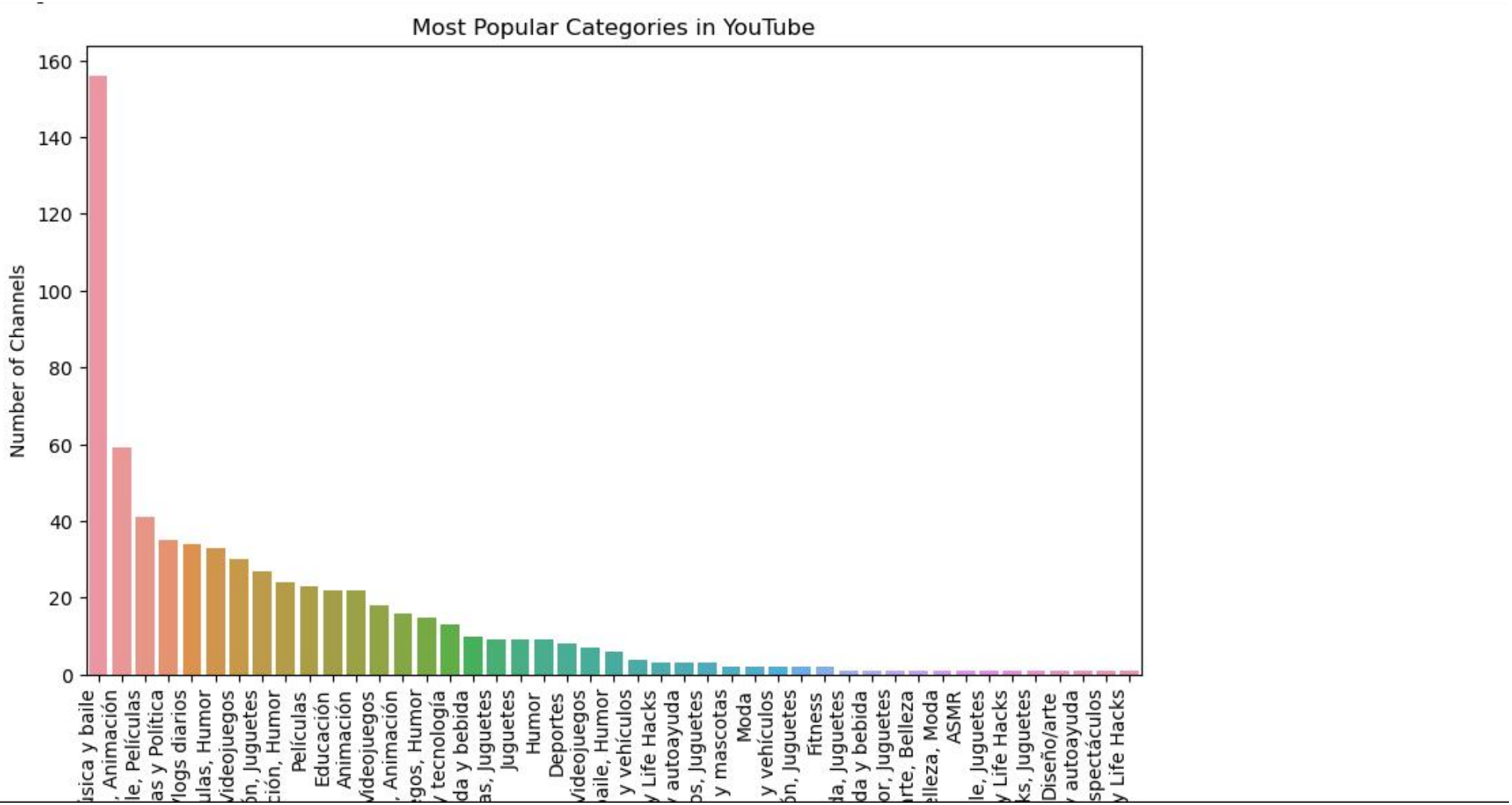


[16]:
```python
# Function to remove outliers using the z-score method
from scipy.stats import zscore
def remove_outliers_zscore(dataframe, columns, threshold=3):
    df_no_outliers = dataframe.copy()
    for column in columns:
        z_scores = zscore(df_no_outliers[column])
        df_no_outliers = df_no_outliers[(z_scores < threshold) & (z_scores > -threshold)]
    return df_no_outliers

# Choose columns to remove outliers from
columns_to_clean = ['Suscribers', 'Visits', 'Likes', 'Rank']

# Remove outliers using the z-score method
df = remove_outliers_zscore(df, columns_to_clean)
```

[17]:
```python
# 2. Trend Analysis
# 2.1 Identify Popular Categories
# Plot the most popular categories
plt.figure(figsize=(12, 8))
category_counts = df['Categories'].value_counts()
plt.figure(figsize=(10, 6))
sns.barplot(x=category_counts.index, y=category_counts.values)
plt.title('Most Popular Categories in YouTube')
plt.xlabel('Category')
plt.ylabel('Number of Channels')
plt.xticks(rotation=90, ha='right')
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```

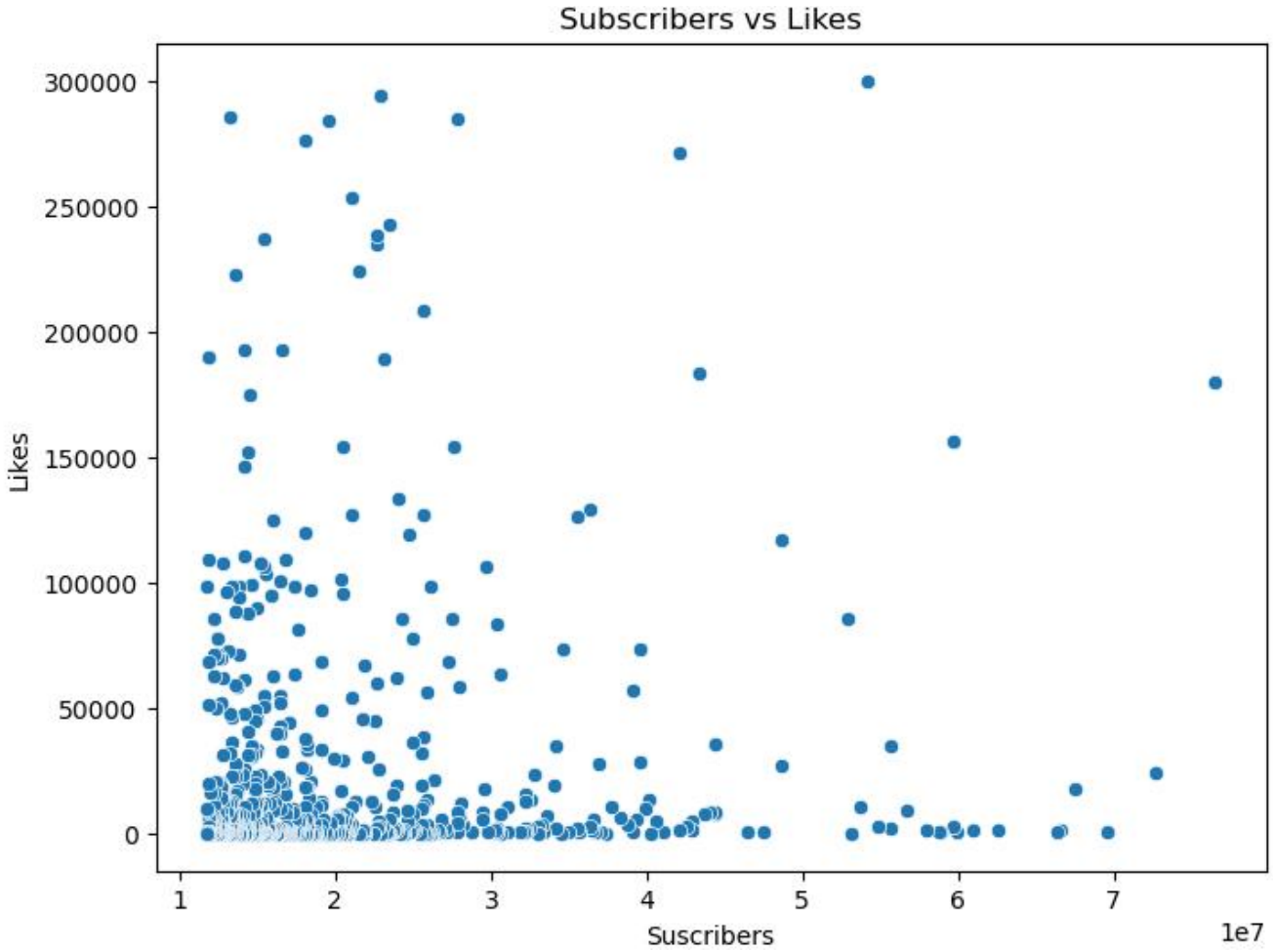

Most Popular Categ
160
140
120


Most Popular Categories in YouTube
Number of Channels
0
20
40
60
80
100
120
140
160
Música y baile
Animación
Películas
Videojuegos
Educación
Animación
Videojuegos
Humor
Películas
Educación
Animación
Videojuegos
Humor
Deportes
Videojuegos
Humor
Life Hacks
autoayuda
Juguetes
mascotas
Moda
vehículos
Juguetes
Fitness
Juguetes
Juguetes
Belleza
Moda
ASMR
Juguetes
Life Hacks
Juguetes
Diseño/arte
autoayuda
espectáculos
Life Hacks

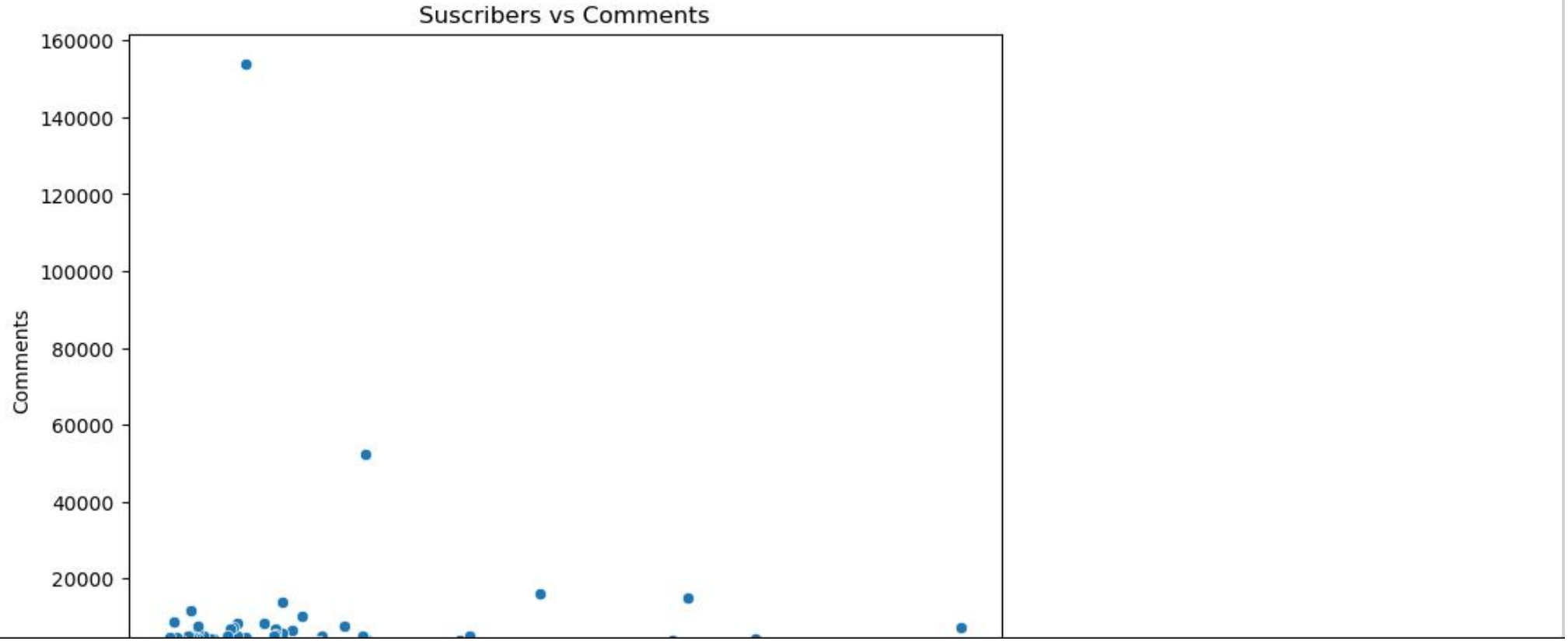
[18]:
```
#2.2 Correlation between Subscribers and Likes/Comments
# Calculate correlation
corr_likes = np.corrcoef(df['Suscribers'], df['Likes'])[0,1]
corr_comments = np.corrcoef(df['Suscribers'], df['Comments'])[0,1]

print(f'Correlation between Subscribers and Likes: {corr_likes}')
print(f'Correlation between Subscribers and Comments: {corr_comments}')
```

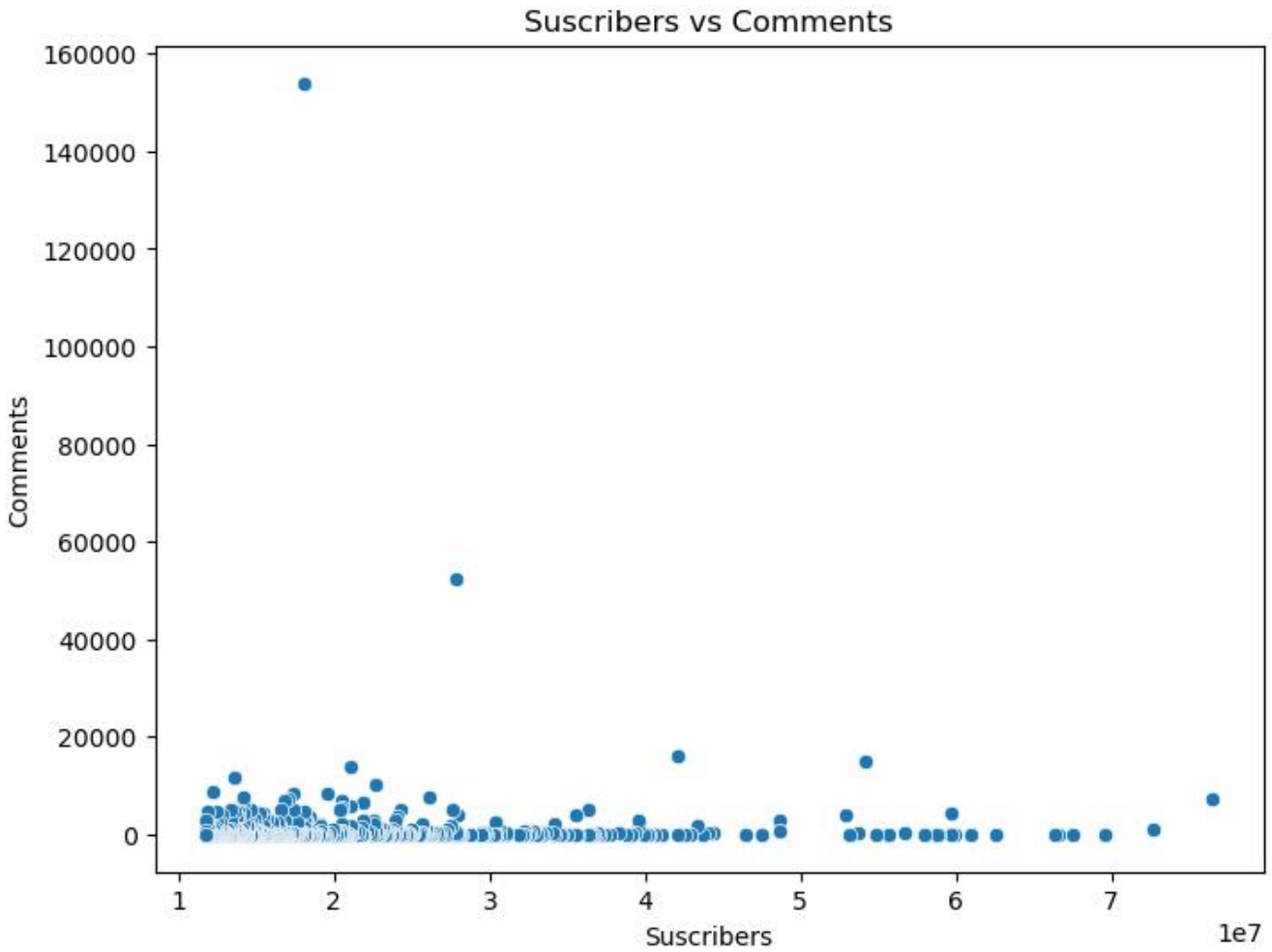
```
Correlation between Subscribers and Likes: 0.07958997301642443
Correlation between Subscribers and Comments: 0.019905291465977786
```

[19]:
```
# Correlation between Subscribers and Likes/Comments
# Plot the relationships
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Suscribers', y='Likes', data=df)
plt.title('Subscribers vs Likes')
plt.xlabel('Suscribers')
plt.ylabel('Likes')
plt.show()
```


Subscribers vs Likes
300000
250000
200000
150000
100000
50000
0
Likes
1
2
3
4
5
6
7
Suscribers
1e7

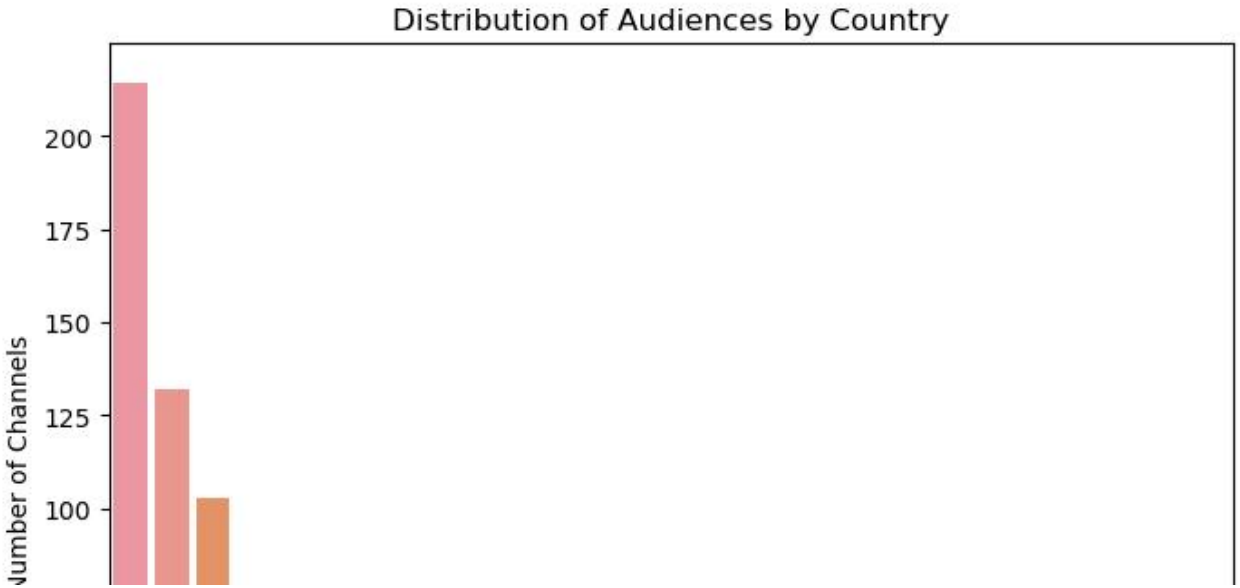
[20]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Suscribers', y='Comments', data=df)
plt.title('Suscribers vs Comments')
plt.xlabel('Suscribers')
plt.ylabel('Comments')
plt.show()
```

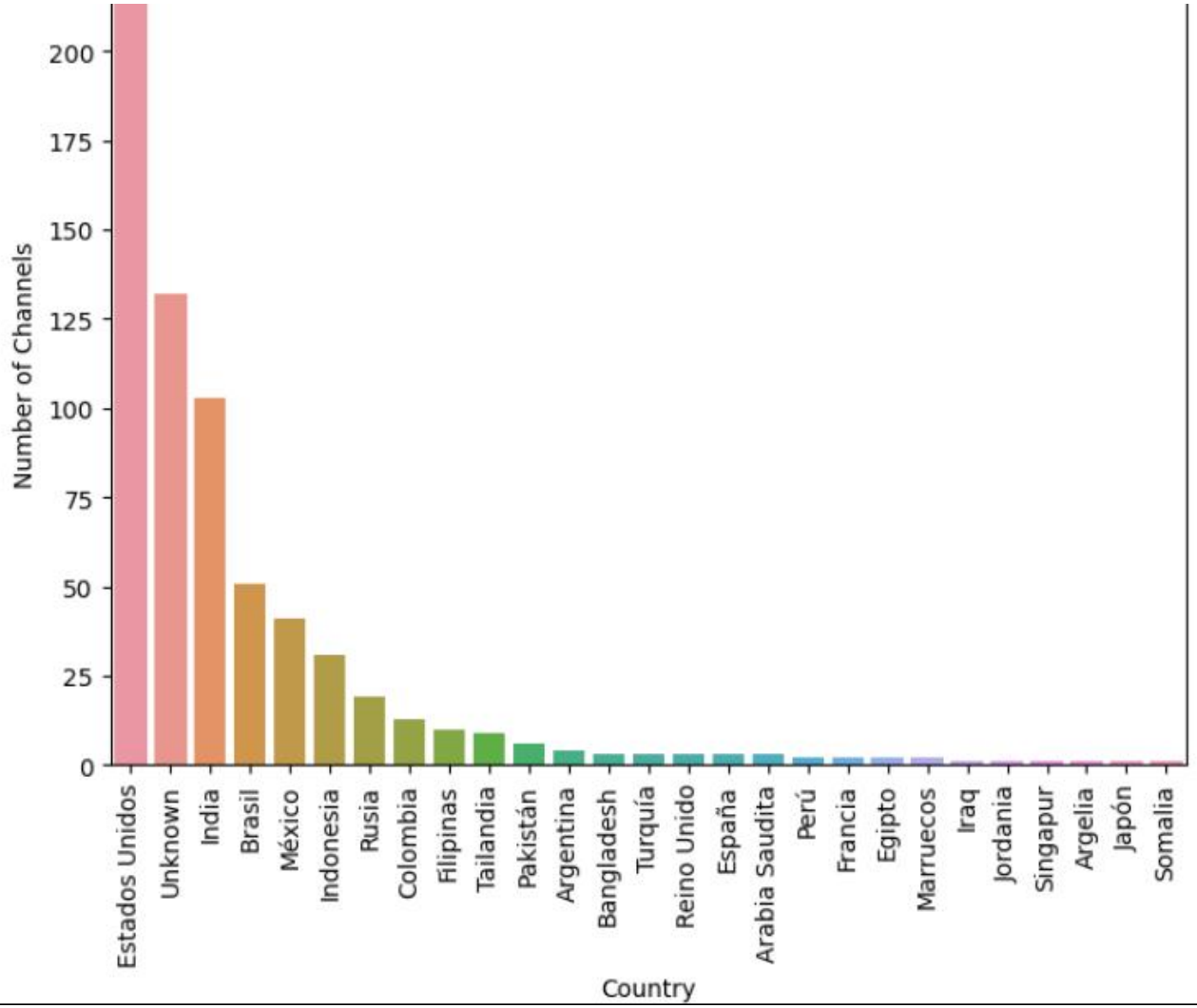



Suscribers vs Comments
160000
140000
120000
100000
80000
60000
40000
20000
0
Comments
1
2
3
4
5
6
7
Suscribers
1e7

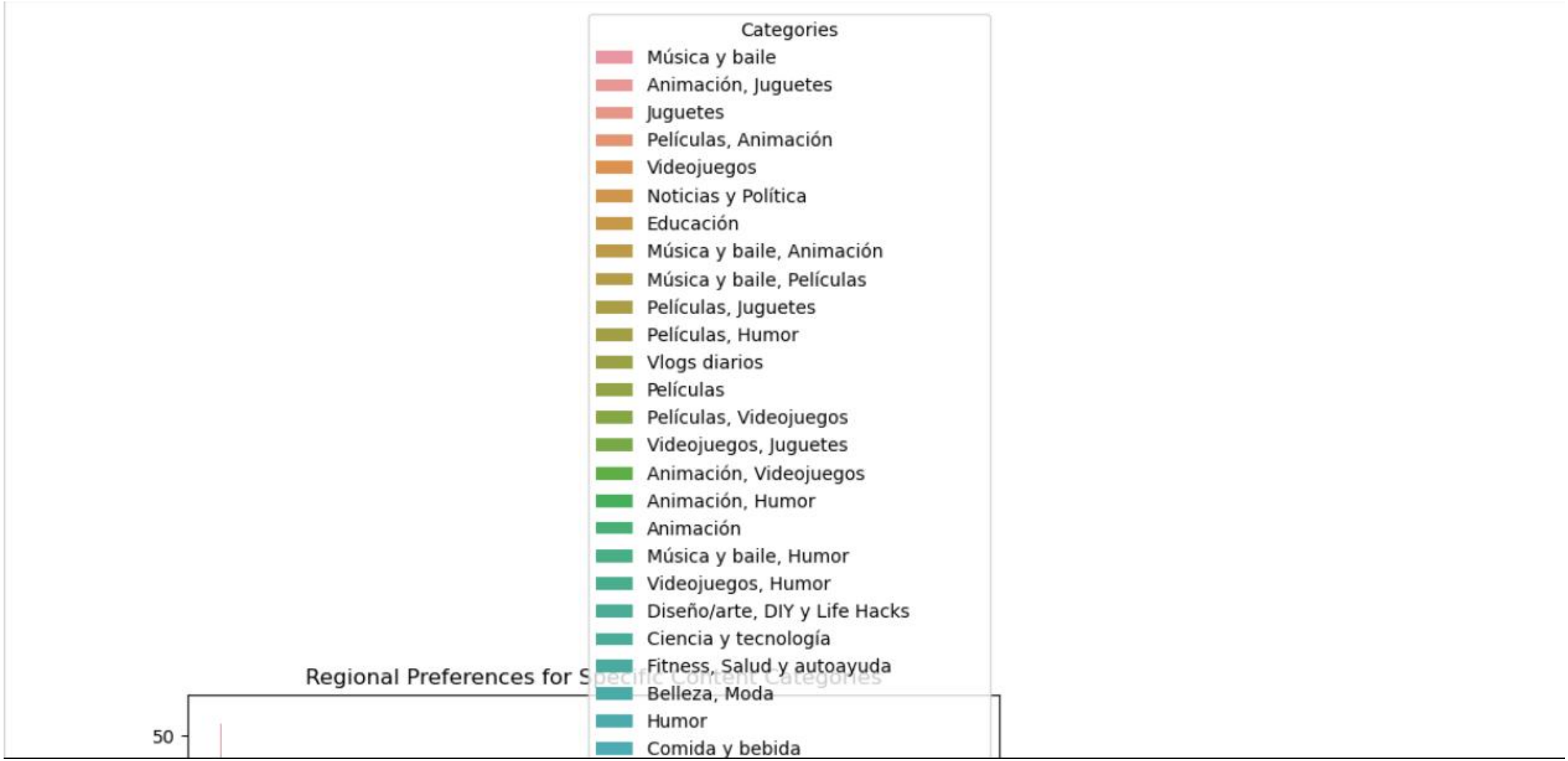
[21]:

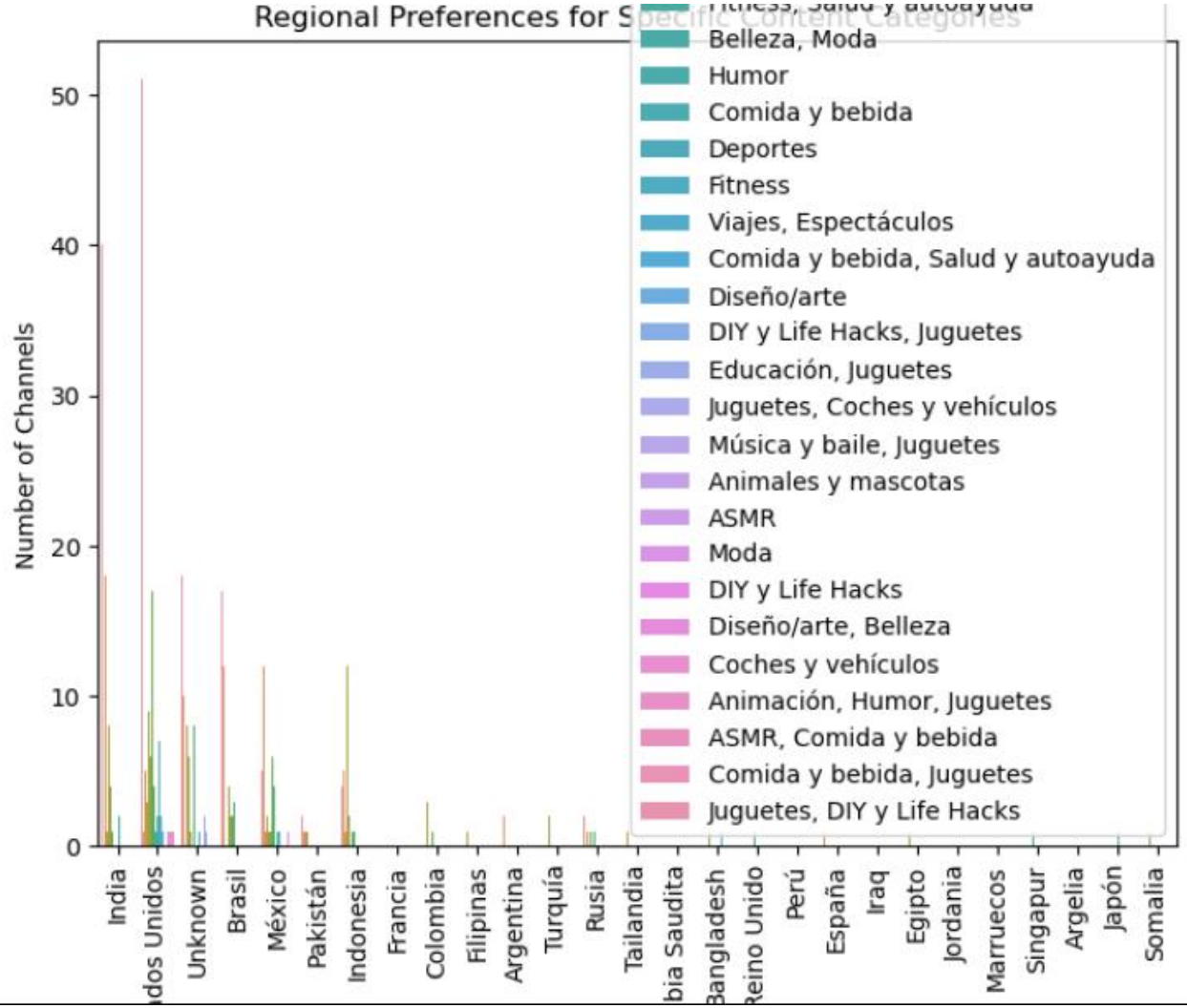
```
#Audience Study
#Distribution of Streamers' Audiences by Country
# Plot the distribution of audiences by country
plt.figure(figsize=(8, 6))
country_counts = df['Country'].value_counts()
sns.barplot(x=country_counts.index, y=country_counts.values)
plt.title('Distribution of Audiences by Country')
plt.xlabel('Country')
plt.ylabel('Number of Channels')
plt.xticks(rotation=90)
plt.show()
```

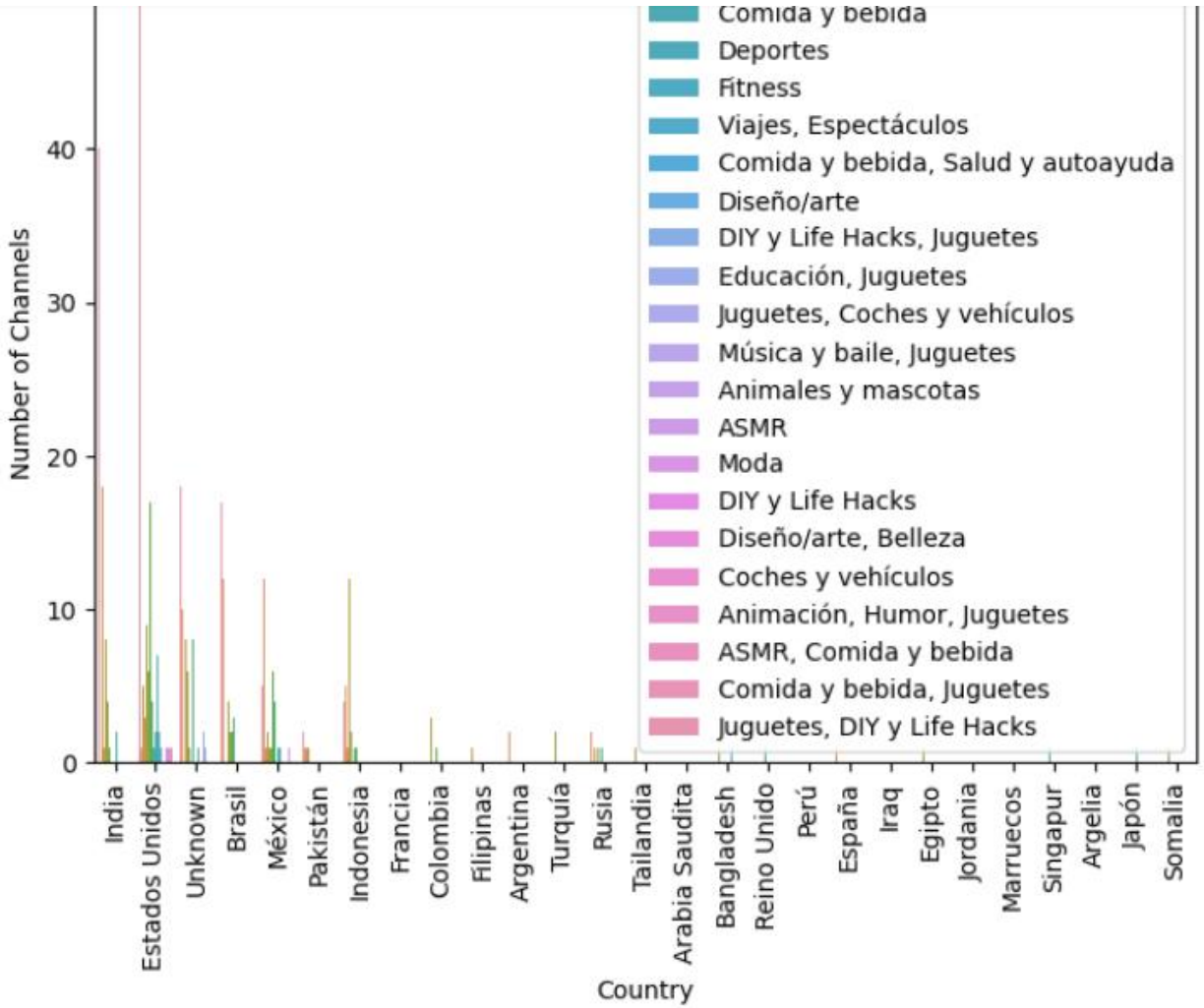



Number of Channels
0
25
50
75
100
125
150
175
200
Estados Unidos
Unknown
India
Brasil
México
Indonesia
Rusia
Colombia
Filipinas
Tailandia
Pakistán
Argentina
Bangladesh
Turquía
Reino Unido
España
Arabia Saudita
Perú
Francia
Egipto
Marruecos
Iraq
Jordania
Singapur
Argelia
Japón
Somalia
Country

[22]:
```
# Regional preferences for specific content categories
plt.figure(figsize=(8, 6))
sns.countplot(x='Country', hue='Categories', data=df)
plt.title('Regional Preferences for Specific Content Categories')
plt.xlabel('Country')
plt.ylabel('Number of Channels')
plt.xticks(rotation=90)
plt.show()
```


Categories
Música y baile
Animación, Juguetes
Juguetes
Películas, Animación
Videojuegos
Noticias y Política
Educación
Música y baile, Animación
Música y baile, Películas
Películas, Juguetes
Películas, Humor
Vlogs diarios
Películas
Películas, Videojuegos
Videojuegos, Juguetes
Animación, Videojuegos
Animación, Humor
Animación
Música y baile, Humor
Videojuegos, Humor
Diseño/arte, DIY y Life Hacks
Ciencia y tecnología
Fitness, Salud y autoayuda
Belleza, Moda
Humor
Comida y bebida
Regional Preferences for S
Specific Content Categories
50


Regional Preferences for S
Specific Content Categories
Number of Channels
0
10
20
30
40
50
India
dos Unidos
Unknown
Brasil
México
Pakistán
Indonesia
Francia
Colombia
Filipinas
Argentina
Turquía
Rusia
Tailandia
bia Saudita
Bangladesh
Reino Unido
Perú
España
Iraq
Egipto
Jordania
Marruecos
Singapur
Argelia
Japón
Somalia
Belleza, Moda
Humor
Comida y bebida
Deportes
Fitness
Viajes, Espectáculos
Comida y bebida, Salud y autoayuda
Diseño/arte
DIY y Life Hacks, Juguetes
Educación, Juguetes
Juguetes, Coches y vehículos
Música y baile, Juguetes
Animales y mascotas
ASMR
Moda
DIY y Life Hacks
Diseño/arte, Belleza
Coches y vehículos
Animación, Humor, Juguetes
ASMR, Comida y bebida
Comida y bebida, Juguetes
Juguetes, DIY y Life Hacks

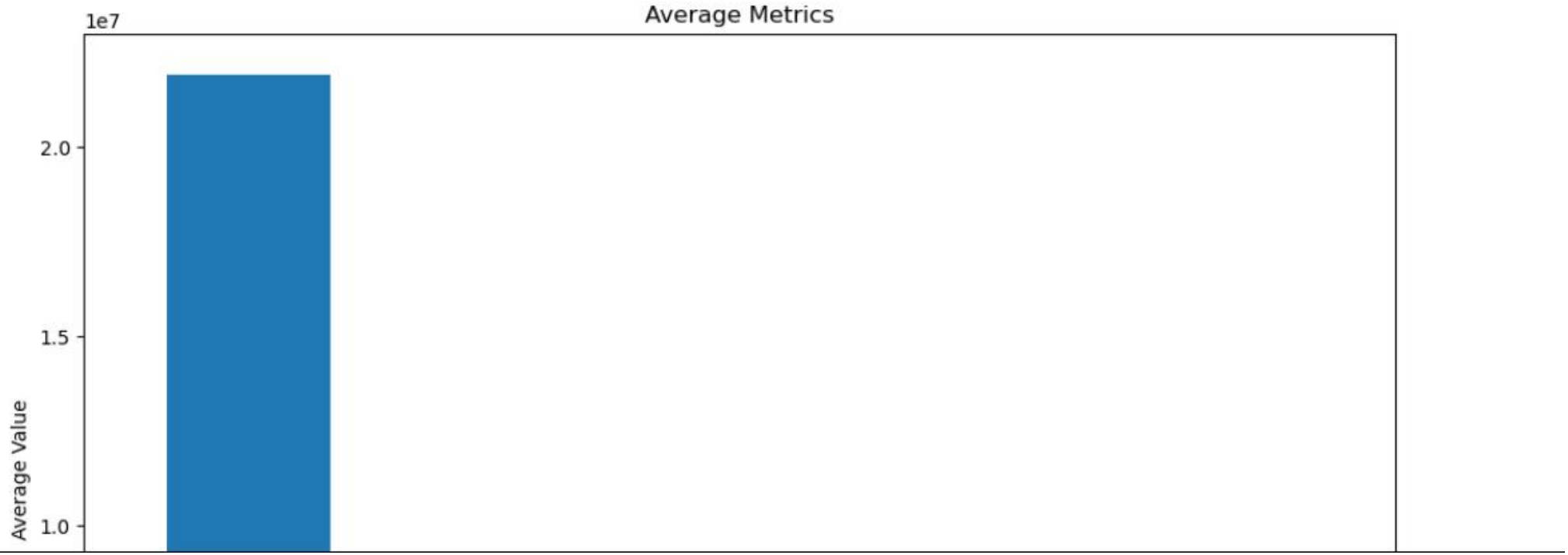

Comida y bebida
Deportes
Fitness
Viajes, Espectáculos
Comida y bebida, Salud y autoayuda
Diseño/arte
DIY y Life Hacks, Juguetes
Educación, Juguetes
Juguetes, Coches y vehículos
Música y baile, Juguetes
Animales y mascotas
ASMR
Moda
DIY y Life Hacks
Diseño/arte, Belleza
Coches y vehículos
Animación, Humor, Juguetes
ASMR, Comida y bebida
Comida y bebida, Juguetes
Juguetes, DIY y Life Hacks
Number of Channels
0
10
20
30
40
India
Estados Unidos
Unknown
Brasil
México
Pakistán
Indonesia
Francia
Colombia
Filipinas
Argentina
Turquía
Rusia
Tailandia
Arabia Saudita
Bangladesh
Reino Unido
Perú
España
Iraq
Egipto
Jordania
Marruecos
Singapur
Argelia
Japón
Somalia
Country

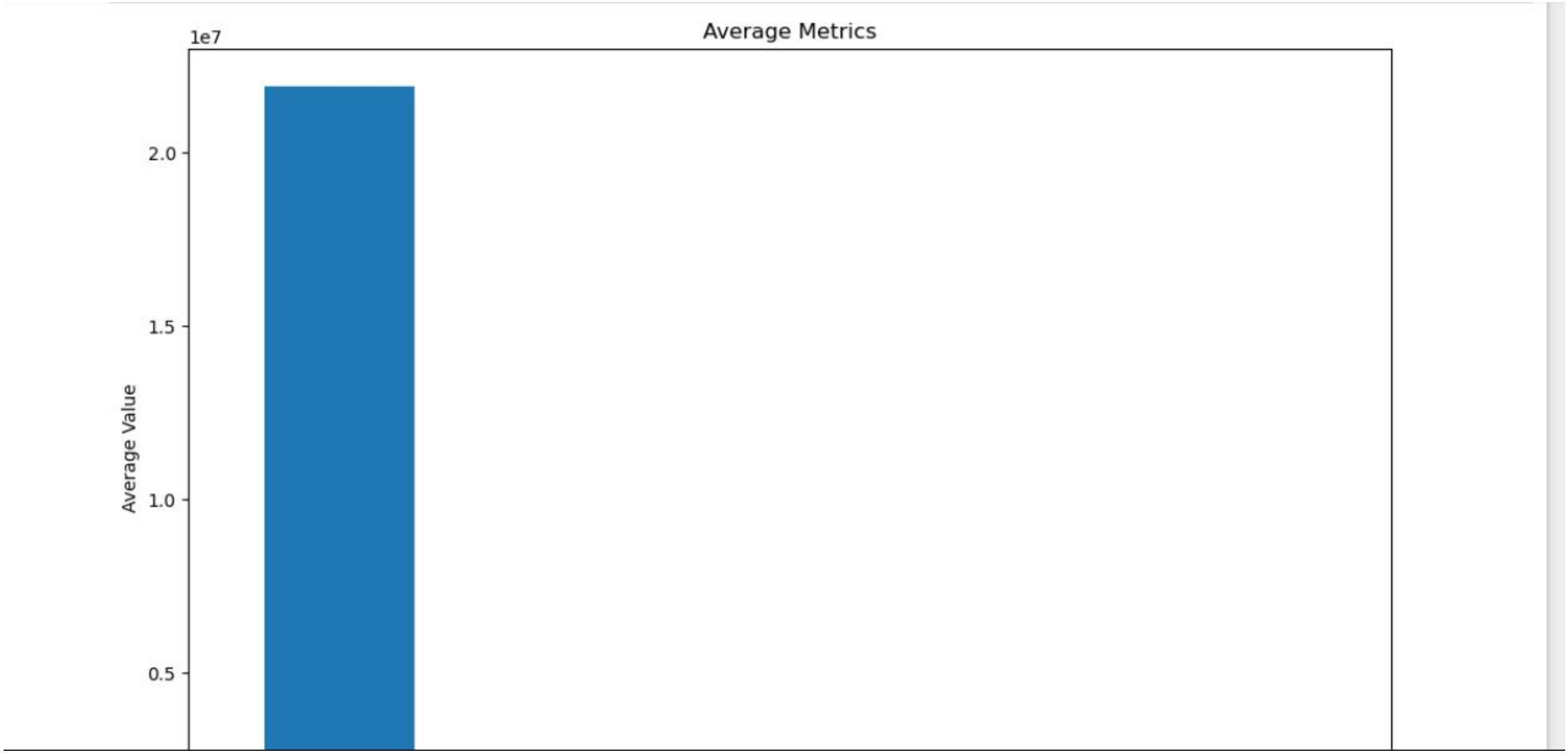
[25]:
```python
import pandas as pd

# Load the dataset
df=  pd.read_csv('youtubers_df.csv')

# Calculate average metrics
def calculate_metric(df):
    average_metrics = df[['Suscribers', 'Visits', 'Likes', 'Comments']].mean()
    return average_metrics

# Call the function and print the average metrics
average_metrics = calculate_metric(df)
print("Average metrics:")
print(average_metrics)
```

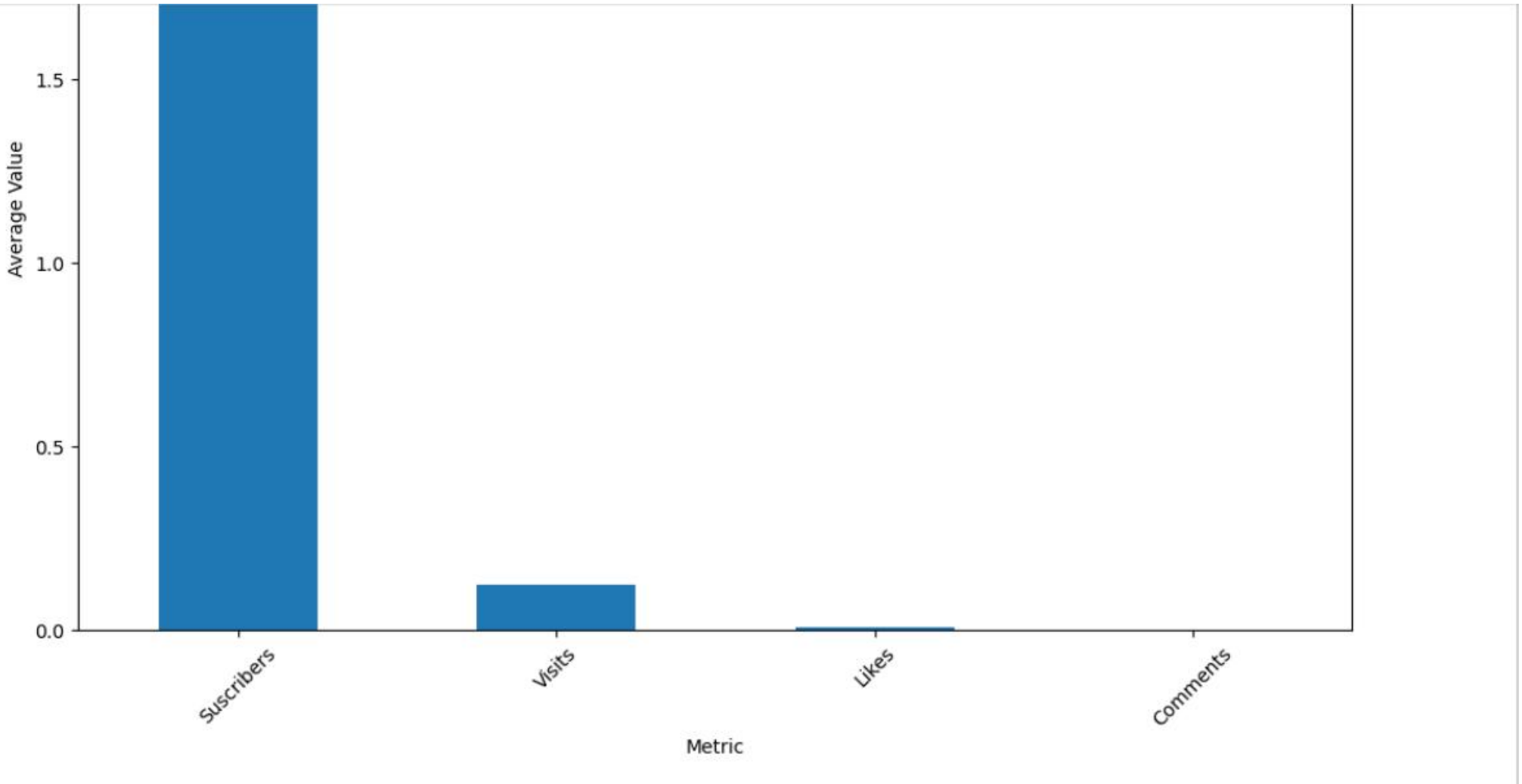
```
Average metrics:
Suscribers    2.189440e+07
Visits        1.209446e+06
Likes         5.363259e+04
Comments      1.288768e+03
dtype: float64
```

[26]:
```python
#Visualize Average Metrics
#Plot average metrics
plt.figure(figsize=(12, 8))
average_metrics.plot(kind='bar')
plt.title('Average Metrics')
plt.xlabel('Metric')
plt.ylabel('Average Value')
```

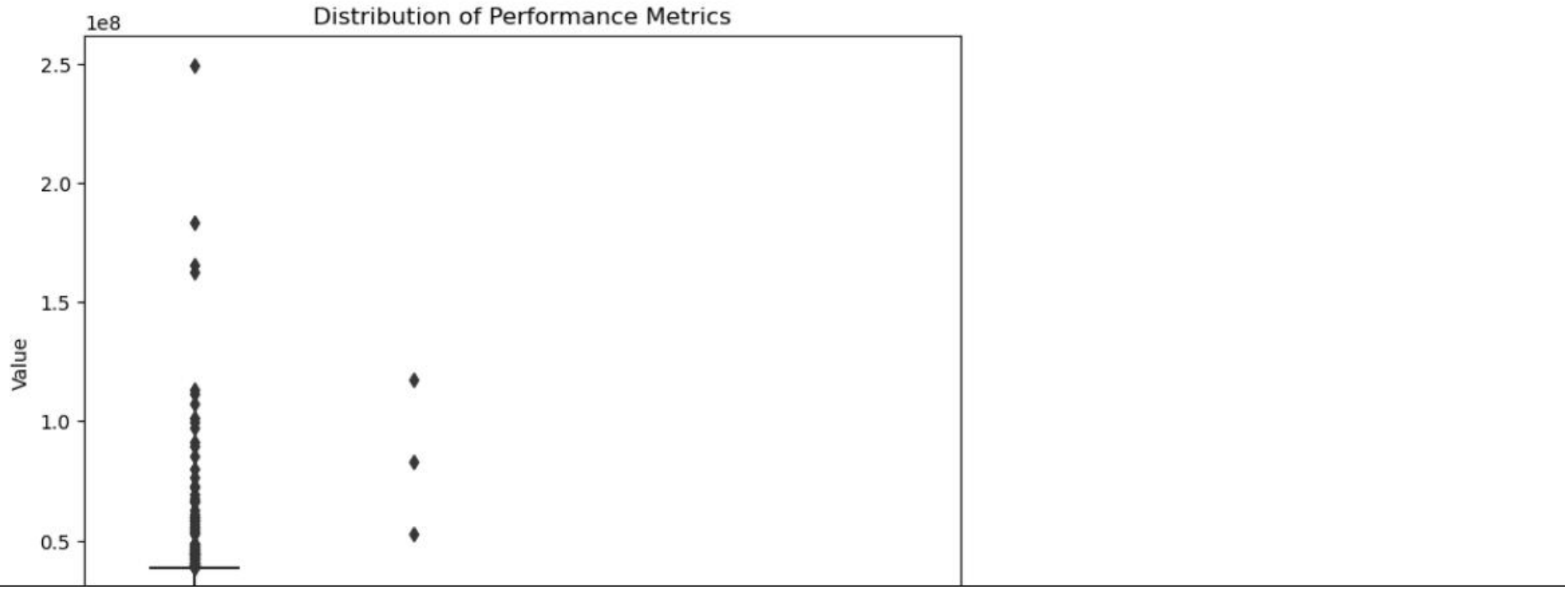
[26]:
```
#Visualize Average Metrics
#Plot average metrics
plt.figure(figsize=(12, 8))
average_metrics.plot(kind='bar')
plt.title('Average Metrics')
plt.xlabel('Metric')
plt.ylabel('Average Value')
plt.xticks(rotation=45)
plt.show()
```

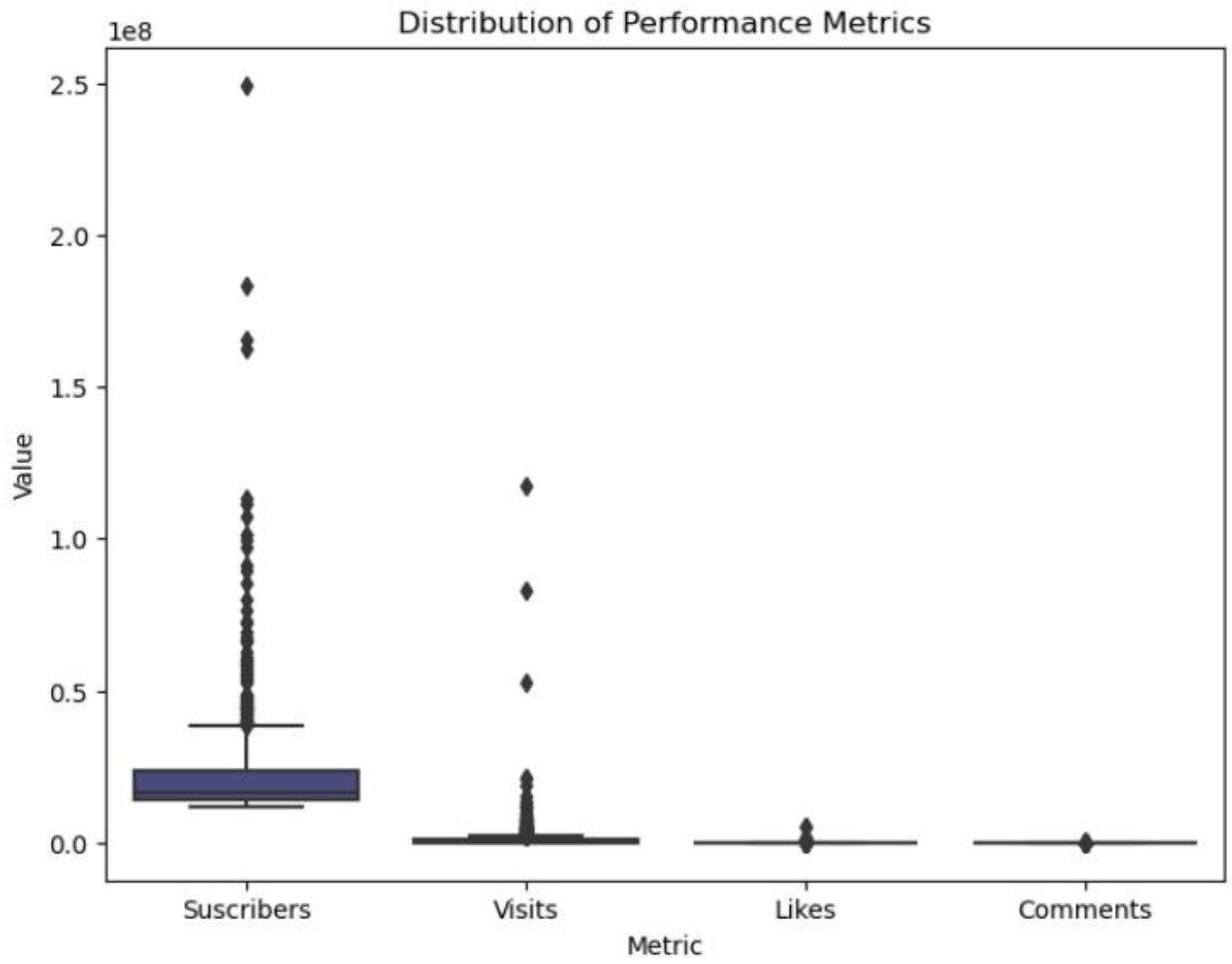



Average Metrics
1e7
2.0
1.5
1.0
0.5
Average Value


Average Value
1.5
1.0
0.5
0.0
Suscribers
Visits
Likes
Comments
Metric

[27]:
```
# 4.2 Identify Patterns or Anomalies
# Boxplot to identify patterns or anomalies
plt.figure(figsize=(8, 6))
sns.boxplot(data=df[['Suscribers', 'Visits', 'Likes', 'Comments']], palette="viridis")
plt.title('Distribution of Performance Metrics')
plt.xlabel('Metric')
plt.ylabel('Value')
plt.show()
```



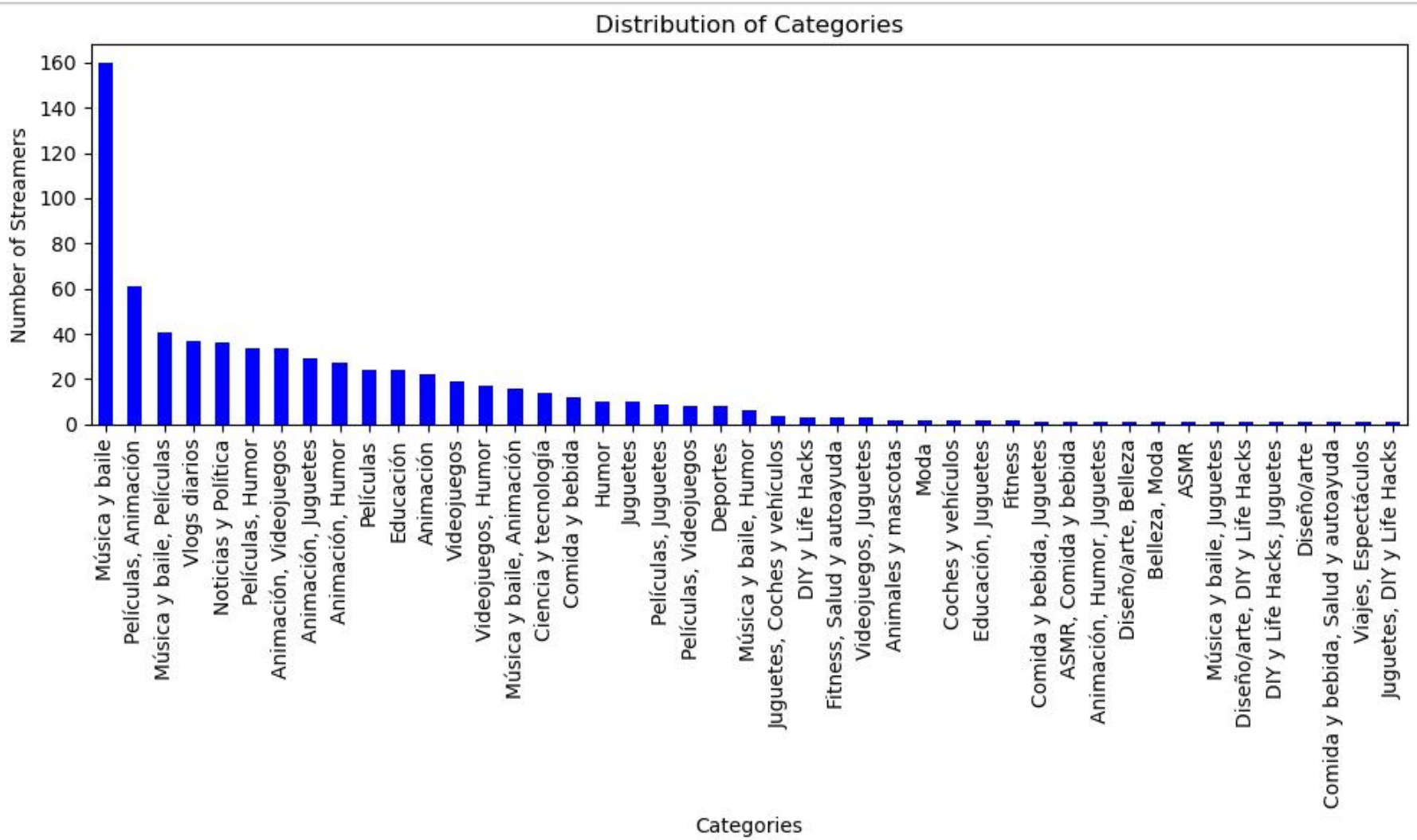

Distribution of Performance Metrics
1e8
2.5
2.0
1.5
1.0
0.5
0.0
Value
Suscribers
Visits
Likes
Comments
Metric

[28]:
```
# 5 Content Categories
#Eplore the distribution of  content categories
category_distribution = df['Categories'].value_counts()
print(category_distribution)
```

```
Categories
Música y baile                        160
Películas, Animación                   61
Música y baile, Películas              41
Vlogs diarios                          37
Noticias y Política                    36
Películas, Humor                       34
Animación, Videojuegos                 34
Animación, Juguetes                    29
Animación, Humor                       27
Películas                              24
Educación                              24
Animación                              22
Videojuegos                            19
Videojuegos, Humor                     17
Música y baile, Animación              16
Ciencia y tecnología                   14
Comida y bebida                        12
Humor                                  10
Juguetes                               10
Películas, Juguetes                     9
Películas, Videojuegos                  8
Deportes                                8
Música y baile, Humor                   6
Juguetes, Coches y vehículos            4
DIY y Life Hacks                        3
Fitness, Salud y autoayuda              3
Videojuegos, Juguetes                   3
Animales y mascotas                     2
```

```
Fitness, Salud y autoayuda              3
Videojuegos, Juguetes                   3
Animales y mascotas                     2
Moda                                    2
Coches y vehículos                      2
Educación, Juguetes                     2
Fitness                                 2
Comida y bebida, Juguetes               1
ASMR, Comida y bebida                   1
Animación, Humor, Juguetes              1
Diseño/arte, Belleza                    1
Belleza, Moda                           1
ASMR                                    1
Música y baile, Juguetes                1
Diseño/arte, DIY y Life Hacks           1
DIY y Life Hacks, Juguetes              1
Diseño/arte                             1
Comida y bebida, Salud y autoayuda      1
Viajes, Espectáculos                    1
Juguetes, DIY y Life Hacks              1
Name: count, dtype: int64
```

[29]:
```python
#Eplore the distribution of  content categories
plt.figure(figsize=(10,6))
category_distribution.plot(kind='bar', color='blue')

plt.title('Distribution of Categories')
plt.xlabel('Categories')
plt.ylabel('Number of Streamers')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```


Distribution of Categories
Number of Streamers
0
20
40
60
80
100
120
140
160
Música y baile
Películas, Animación
Música y baile, Películas
Vlogs diarios
Noticias y Política
Películas, Humor
Animación, Videojuegos
Animación, Juguetes
Animación, Humor
Películas
Educación
Animación
Videojuegos
Videojuegos, Humor
Música y baile, Animación
Ciencia y tecnología
Comida y bebida
Humor
Juguetes
Películas, Juguetes
Películas, Videojuegos
Deportes
Música y baile, Humor
Juguetes, Coches y vehículos
DIY y Life Hacks
Fitness, Salud y autoayuda
Videojuegos, Juguetes
Animales y mascotas
Moda
Coches y vehículos
Educación, Juguetes
Fitness
Comida y bebida, Juguetes
ASMR, Comida y bebida
Animación, Humor, Juguetes
Diseño/arte, Belleza
Belleza, Moda
ASMR
Música y baile, Juguetes
Diseño/arte, DIY y Life Hacks
DIY y Life Hacks, Juguetes
Diseño/arte
Comida y bebida, Salud y autoayuda
Viajes, Espectáculos
Juguetes, DIY y Life Hacks
Categories

[30]:
```
# Brands and Collaborations
# Anàlyze brand collaboration with high perfoming streamers
# Load the dataset
df=  pd.read_csv('youtubers_df.csv')
import pandas as pd

# Filtering high performers
high_performers = df[df['Suscribers'] > df['Suscribers'].mean()].copy()

# Calculating average metrics
average_metrics = high_performers[['Suscribers', 'Visits', 'Likes','Comments']].mean()

# Counting likes
high_performers['Like Count'] = high_performers['Links'].str.count(',') + 1

print(high_performers)
print(average_metrics)
```

```
     Rank          Username                Categories   Suscribers  \
0       1           tseries            Música y baile  249500000.0
1       2           MrBeast        Videojuegos, Humor  183500000.0
2       3         CoComelon                Educación  165500000.0
3       4          SETIndia                       NaN  162600000.0
4       5     KidsDianaShow       Animación, Juguetes  113500000.0
..    ...               ...                       ...          ...
298   299       williesalim          Películas, Humor   21900000.0
299   300     SMOL_official                       NaN   21900000.0
300   301      alfredolarin                       NaN   21900000.0
301   302  TlnovelasOficial  Música y baile, Animación   21900000.0
302   303        royaltyfam                     Humor   21900000.0

            Country       Visits      Likes  Comments  \
```

```
302   303        Royalty Ram                          Humor   21900000.0
```

	Country	Visits	Likes	Comments	\
0	India	86200.0	2700.0	78.0	
1	Estados Unidos	117400000.0	5300000.0	18500.0	
2	Unknown	7000000.0	24700.0	0.0	
3	India	15600.0	166.0	9.0	
4	Unknown	3900000.0	12400.0	0.0	
..	...	...	...	...	
298	Indonesia	396200.0	0.0	2800.0	
299	India	114500.0	2600.0	5.0	
300	El Salvador	12900000.0	707600.0	2100.0	
301	México	5500.0	152.0	5.0	
302	Estados Unidos	4700000.0	67000.0	6600.0	

	Links	Like Count
0	http://youtube.com/channel/UCq-Fj5jknLsUf-MWSy...	1
1	http://youtube.com/channel/UCX6OQ3DkcsbYNE6H8u...	1
2	http://youtube.com/channel/UCbCmjCuTUZos6Inko4...	1
3	http://youtube.com/channel/UCpEhnqL0y41EpW2TvW...	1
4	http://youtube.com/channel/UCk8GzjMOrta8yxDcKf...	1
..	...	...
298	http://youtube.com/channel/UCPCaXSwaos-QI03iZt...	1
299	http://youtube.com/channel/UCBBZ7No0AzEJ3qiatj...	1
300	http://youtube.com/channel/UCd5ApCORQsMOZZz5E9...	1
301	http://youtube.com/channel/UCKyU-wd-KY4PMOcOpP...	1
302	http://youtube.com/channel/UCja7QUMRG9AD8X2F_v...	1

```
[303 rows x 10 columns]
Suscribers    3.707030e+07
Visits        2.001183e+06
Likes         8.680301e+04
Comments      1.619927e+03
dtype: float64
```

[32]:
```
# Recommend streamers based on category and perfomance metrics
from sklearn.preprocessing import MinMaxScaler

# Fill missing categories with 'Unknown'
df['Categories'].fillna('Unknown', inplace=True)

# Normalize performance metrics
scaler = MinMaxScaler()
df[['Suscribers', 'Visits', 'Likes', 'Comments']] = scaler.fit_transform(
    df[['Suscribers', 'Visits', 'Likes', 'Comments']])

# Calculate engagement rate (likes + comments) / visits
df['EngagementRate'] = (df['Likes'] + df['Comments']) / df['Visits']

# Display the updated dataframe
df.head()
```

[32]:

	Rank	Username	Categories	Suscribers	Country	Visits	Likes	Comments	Links	EngagementRate
0	1	tseries	Música y baile	1.000000	India	0.000734	0.000509	0.000506	http://youtube.com/channel/UCq-Fj5jknLsUf-MWSy...	1.383641
1	2	MrBeast	Videojuegos, Humor	0.722456	Estados Unidos	1.000000	1.000000	0.120130	http://youtube.com/channel/UCX6OQ3DkcsbYNE6H8u...	1.120130
2	3	CoComelon	Educación	0.646762	Unknown	0.059625	0.004660	0.000000	http://youtube.com/channel/UCbCmjCuTUZos6Inko4...	0.078161
3	4	SETIndia	Unknown	0.634567	India	0.000133	0.000031	0.000058	http://youtube.com/channel/UCpEhnqL0y41EpW2TvW...	0.675519
4	5	KidsDianaShow	Animación, Juguetes	0.428091	Unknown	0.033220	0.002340	0.000000	http://youtube.com/channel/UCk8GzjMOrta8yxDcKf...	0.070429

[33]:

```python
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics.pairwise import cosine_similarity

# Load the dataset
df=  pd.read_csv('youtubers_df.csv')

# Fill missing categories with 'Unknown'
df['Categories'].fillna('Unknown', inplace=True)

# Normalize performance metrics
scaler = MinMaxScaler()
df[['Suscribers', 'Visits', 'Likes', 'Comments']] = scaler.fit_transform(
    df[['Suscribers', 'Visits', 'Likes', 'Comments']])

# Calculate engagement rate (likes + comments) / visits
df['EngagementRate'] = (df['Likes'] + df['Comments']) / df['Visits']

# Create a matrix with categories and engagement rate
feature_matrix = df[['Categories', 'EngagementRate']]

# Encode categories using one-hot encoding
feature_matrix = pd.get_dummies(feature_matrix, columns=['Categories'])

# Fill missing EngagementRate with the median value
feature_matrix['EngagementRate'].fillna(feature_matrix['EngagementRate'].median(), inplace=True)

# Compute cosine similarity between streamers
cosine_sim = cosine_similarity(feature_matrix, feature_matrix)

# Function to get recommendations
def get_recommendations(username, cosine_sim=cosine_sim):
```

```python
# Create a matrix with categories and engagement rate
feature_matrix = df[['Categories', 'EngagementRate']]

# Encode categories using one-hot encoding
feature_matrix = pd.get_dummies(feature_matrix, columns=['Categories'])

# Fill missing EngagementRate with the median value
feature_matrix['EngagementRate'].fillna(feature_matrix['EngagementRate'].median(), inplace=True)

# Compute cosine similarity between streamers
cosine_sim = cosine_similarity(feature_matrix, feature_matrix)

# Function to get recommendations
def get_recommendations(username, cosine_sim=cosine_sim):
    idx = df[df['Username'] == username].index[0]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:6]  # Get top 5 similar streamers
    streamer_indices = [i[0] for i in sim_scores]
    return df['Username'].iloc[streamer_indices]

# Example recommendation
recommended_streamers = get_recommendations('MrBeast')
print(recommended_streamers)
```

```
179      brentrivera
278      StokesTwins
588           Jesser
376          Sidemen
660    rebeccazamolo
Name: Username, dtype: object
```

```
    return df['Username'].iloc[streamer_indices]

# Example recommendation
recommended_streamers = get_recommendations('MrBeast')
print(recommended_streamers)
```

```
179      brentrivera
278      StokesTwins
588           Jesser
376          Sidemen
660    rebeccazamolo
Name: Username, dtype: object
```

Conclusion

In conclusion, our analysis of the YouTube streamers dataset revealed valuable insights, including popular content categories and audience preferences, as well as correlations between engagement metrics. We identified the top-performing streamers, providing benchmarks for success, and suggested a content recommendation system based on categories and performance metrics. These findings enhance the understanding of the YouTube streaming landscape, enabling stakeholders to make informed decisions and improve the user experience.

[ ]: