In [2]:
```
import pandas as pd

path=r'C:\Users\DELL\Documents\Inter Career Data Analysis\youtubers_df.csv'

youtubers_df = pd.read_csv(path)

cat_column=youtubers_df.select_dtypes(include='object').columns
num_column=youtubers_df.select_dtypes(exclude='object').columns
```

In [3]:
```
youtubers_df
```

Out[3]:

	Rank	Username	Categories	Suscribers	Country	Visits	L
0	1	tseries	Música y baile	249500000.0	India	86200.0	27
1	2	MrBeast	Videojuegos, Humor	183500000.0	Estados Unidos	117400000.0	53000
2	3	CoComelon	Educación	165500000.0	Unknown	7000000.0	247
3	4	SETIndia	NaN	162600000.0	India	15600.0	1
4	5	KidsDianaShow	Animación, Juguetes	113500000.0	Unknown	3900000.0	124
...	...	...	...	...	...	...	
995	996	hamzymukbang	NaN	11700000.0	Estados Unidos	397400.0	140
996	997	Adaahqueen	NaN	11700000.0	India	1100000.0	925
997	998	LittleAngelIndonesia	Música y baile	11700000.0	Unknown	211400.0	7
998	999	PenMultiplex	NaN	11700000.0	India	14000.0	
999	1000	OneindiaHindi	Noticias y Política	11700000.0	India	2200.0	

1000 rows × 9 columns

Start by exploring the dataset to understand its structure and identify key variables.- Check for missing data and outliers

1. Data Exploration

In [4]:
```
import pandas as pd

path=r'C:\Users\DELL\Documents\Inter Career Data Analysis\youtubers_df.csv'

df=pd.read_csv(path)

print(df.head()) # first 5 values
```

```
   Rank       Username           Categories   Suscribers         Country  \
0     1        tseries       Música y baile  249500000.0           India
1     2        MrBeast   Videojuegos, Humor  183500000.0  Estados Unidos
2     3      CoComelon            Educación  165500000.0         Unknown
3     4       SETIndia                  NaN  162600000.0           India
4     5   KidsDianaShow  Animación, Juguetes  113500000.0         Unknown

        Visits      Likes  Comments  \
0      86200.0     2700.0      78.0
1  117400000.0  5300000.0   18500.0
2    7000000.0    24700.0       0.0
3      15600.0      166.0       9.0
4    3900000.0    12400.0       0.0

                                              Links
0  http://youtube.com/channel/UCq-Fj5jknLsUf-MWSy...
1  http://youtube.com/channel/UCX6OQ3DkcsbYNE6H8u...
2  http://youtube.com/channel/UCbCmjCuTUZos6Inko4...
3  http://youtube.com/channel/UCpEhnqL0y41EpW2TvW...
4  http://youtube.com/channel/UCk8GzjMOrta8yxDcKf...
```

In [5]:
```
print(df.tail())
```

```
     Rank             Username           Categories  Suscribers  \
995   996          hamzymukbang                  NaN  11700000.0
996   997            Adaahqueen                  NaN  11700000.0
997   998  LittleAngelIndonesia       Música y baile  11700000.0
998   999          PenMultiplex                  NaN  11700000.0
999  1000         OneindiaHindi  Noticias y Política  11700000.0

            Country     Visits    Likes  Comments  \
995  Estados Unidos   397400.0  14000.0     124.0
996           India  1100000.0  92500.0     164.0
997         Unknown   211400.0    745.0       0.0
998           India    14000.0     81.0       1.0
999           India     2200.0     31.0       1.0

                                              Links
995  http://youtube.com/channel/UCPKNKldggioffXPkSm...
996  http://youtube.com/channel/UCk3fFpqI5kDMf__mUP...
997  http://youtube.com/channel/UCdrHrQf0o0TO8YDntX...
998  http://youtube.com/channel/UCObyBrdrtQ20BU9PxH...
999  http://youtube.com/channel/UCOjgc1p2hJ4GZi6pQQ...
```

In [6]:
```
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
```

direct method to describe

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

Out[7]:

	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

In [8]:
```
num_column
```

Out[8]:
```
Index(['Rank', 'Suscribers', 'Visits', 'Likes', 'Comments'], dtype='object')
```

In [9]:
```
import numpy as np
df1=pd.DataFrame()

for i in num_column:
    Count=len(df[i])
    Min=min(df[i])
    Max=max(df[i])
    Mean=round(df[i].mean(),2)
    Median=round(df[i].median(),2)
    Std=round(df[i].std(),2)
    p_25=round(np.percentile(df[i],25),2)
    p_50=round(np.percentile(df[i],50),2)
    p_75=round(np.percentile(df[i],75),2)
    values=[Count,Min,Max,Mean,Median,Std,p_25,p_50,p_75]
    index=['Count','Min','Max','Mean','Median','Std','25%','50%','75%']
    cols=[i]
```

```
    df2=pd.DataFrame(values,index=index,columns=cols)
    df1=pd.concat([df1,df2],axis=1)
df1
```

Out[9]:

	Rank	Suscribers	Visits	Likes	Comments
Count	1000.00	1.000000e+03	1.000000e+03	1000.00	1000.00
Min	1.00	1.170000e+07	0.000000e+00	0.00	0.00
Max	1000.00	2.495000e+08	1.174000e+08	5300000.00	154000.00
Mean	500.50	2.189440e+07	1.209446e+06	53632.59	1288.77
Median	500.50	1.675000e+07	1.744500e+05	3500.00	67.00
Std	288.82	1.682775e+07	5.229942e+06	258045.69	6778.19
25%	250.75	1.380000e+07	3.197500e+04	471.75	2.00
50%	500.50	1.675000e+07	1.744500e+05	3500.00	67.00
75%	750.25	2.370000e+07	8.654750e+05	28650.00	472.00

Columns

In [10]:
```
print(df.columns)
```

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

missing value

In [11]:
```
missing_data = df.isnull().sum()
print(missing_data[missing_data > 0])
```

```
Categories    306
dtype: int64
```

outliers_data

In [12]:
```
data=df[i]
```

In [13]:
```
missing_data=df[i]
q1=np.percentile(data,25)
q2=np.percentile(data,50)
q3=np.percentile(data,75)

IQR=q3-q1

lb=q1-1.5*IQR
ub=q3+1.5*IQR

con1=df[i]<lb
con2=df[i]>ub
con3=con1|con2

outliers_data=data[con3]
outliers_data
```

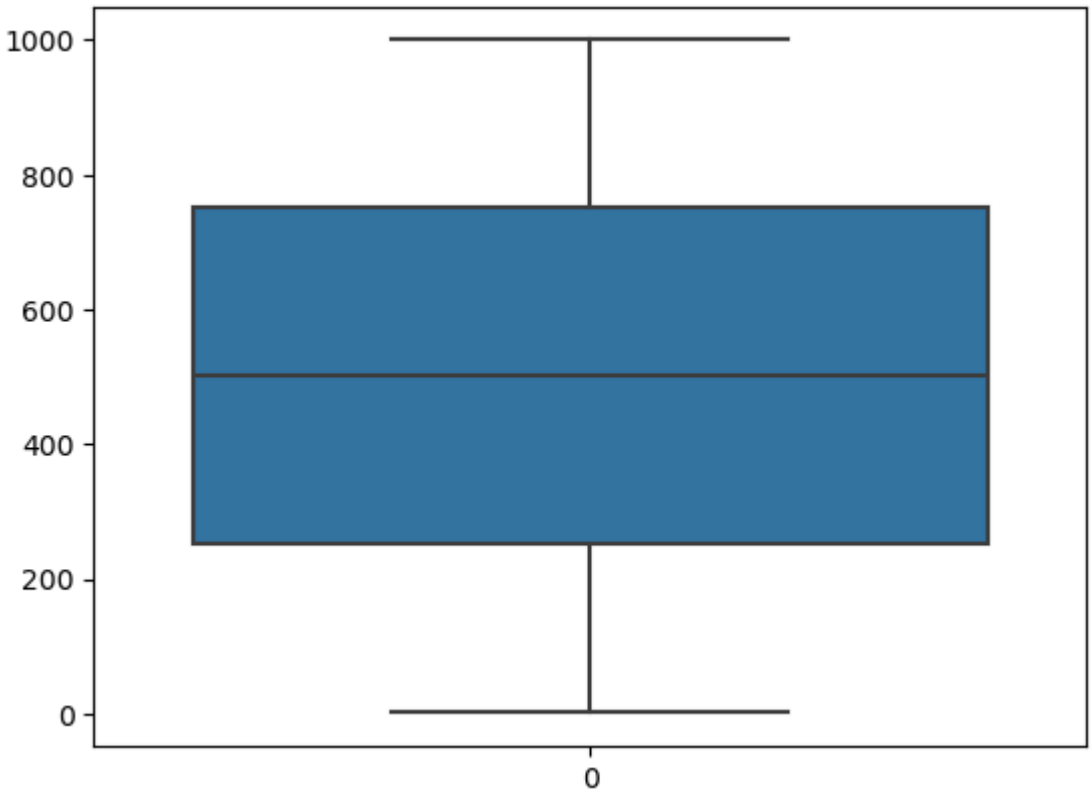
Out[13]:
```
1      18500.0
5       4900.0
10      3400.0
14      7400.0
26      4200.0
        ...
965     4800.0
976     2700.0
978     2300.0
983     2400.0
990     2900.0
Name: Comments, Length: 151, dtype: float64
```

boxplots Methods

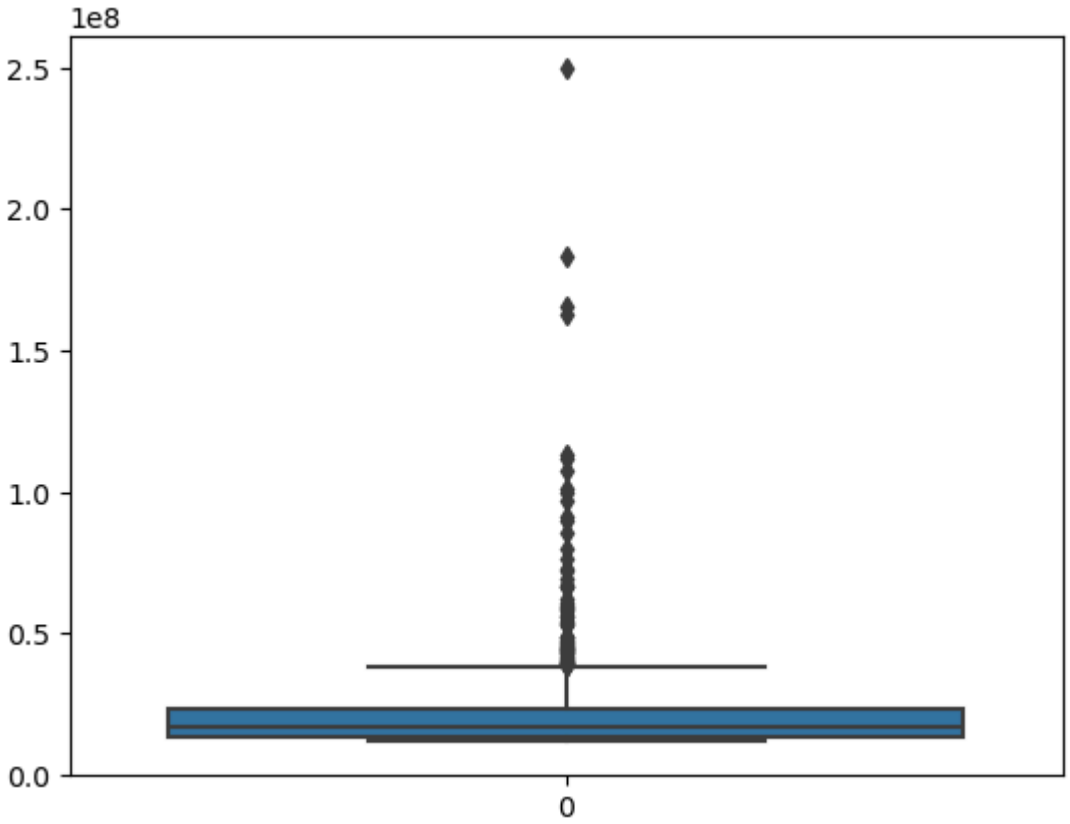
In [14]:
```
num_column
```

Out[14]:
```
Index(['Rank', 'Suscribers', 'Visits', 'Likes', 'Comments'], dtype='object')
```

In [15]:
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.boxplot(df['Rank'])
plt.show()
```



In [16]:
```
import seaborn as sns

sns.boxplot(df['Suscribers'])
plt.show()
```



Z-score method

In [17]:
```
from scipy import stats
import numpy as np

z_scores = np.abs(stats.zscore(df['Comments']))
outliers = np.where(z_scores > 3)  # Z-score > 3 indicates an outlier
print(df.iloc[outliers])
```

```
     Rank            Username              Categories  Suscribers  \
43     44           A4a4a4a4        Animación, Humor  47300000.0
123   124      MRINDIANHACKER                     NaN  32600000.0
153   154           DaFuqBoom        Animación, Humor  29800000.0
177   178             DanTDM  Animación, Videojuegos  27800000.0
341   342     triggeredinsaan                   Humor  20400000.0
436   437  BispoBrunoLeonardo          Música y baile  18000000.0
488   489   BeastPhilanthropy         Comida y bebida  16900000.0
958   959             dojacat          Música y baile  11900000.0

            Country      Visits      Likes  Comments  \
43            Rusia   9700000.0   330400.0   22000.0
123           India   6500000.0   617400.0   26000.0
153  Estados Unidos  52700000.0  1700000.0   82800.0
177  Estados Unidos   3500000.0   285000.0   52500.0
341           India  11100000.0  1400000.0   38000.0
436          Brasil    762100.0   276400.0  154000.0
488  Estados Unidos  21500000.0   952100.0   24000.0
958  Estados Unidos  13600000.0   395300.0   73000.0

                                                 Links
43   http://youtube.com/channel/UC2tsySbe9TNrI-xh2l...
123  http://youtube.com/channel/UCSiDGb0MnHFGjs4E2W...
153  http://youtube.com/channel/UCsSsgPaZ2GSmO6il8C...
177  http://youtube.com/channel/UCS5Oz6CHmeoF7vSad0...
341  http://youtube.com/channel/UCfLuT3JwLx8rvHjHfT...
436  http://youtube.com/channel/UCVNouUw3d3l5JYVCxh...
488  http://youtube.com/channel/UCAiLfjNXkNv24uhpzU...
958  http://youtube.com/channel/UCzpl23pGTHVYqvKsgY...
```

2. Trend Analysis:

- Identify trends among the top YouTube streamers. Which categories are the most popular? - Is there a correlation between the number of subscribers and the number of likes or comments?

In [20]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [21]:
```
# Clean the data
# Remove rows with missing values in important columns like 'Categories' and 'Su
youtubers_df_clean = youtubers_df.dropna(subset=['Categories', 'Suscribers', 'Vi

# Convert Subscribers, Visits, Likes, and Comments to numeric values (in case th
youtubers_df_clean['Suscribers'] = pd.to_numeric(youtubers_df_clean['Suscribers'
youtubers_df_clean['Visits'] = pd.to_numeric(youtubers_df_clean['Visits'], error
youtubers_df_clean['Likes'] = pd.to_numeric(youtubers_df_clean['Likes'], errors=
youtubers_df_clean['Comments'] = pd.to_numeric(youtubers_df_clean['Comments'], e

# Group by categories to get the average of key metrics
category_trends = youtubers_df_clean.groupby('Categories').agg({
    'Suscribers': 'mean',
    'Visits': 'mean',
    'Likes': 'mean',
    'Comments': 'mean'
}).sort_values(by='Suscribers', ascending=False)
```

```
# Calculate correlation between different metrics
correlations = youtubers_df_clean[['Suscribers', 'Visits', 'Likes', 'Comments']]

category_trends, correlations
```

Out[21]: (

\

Categories	Suscribers	Visits	Likes	Comments
Juguetes	3.788000e+07	7.005100e+05	5290.200000	2.800000
Películas, Videojuegos	3.325000e+07	6.940375e+05	48083.375000	1569.500000
Animación, Juguetes	2.937586e+07	5.254483e+05	2653.068966	0.517241
Videojuegos, Humor	2.876471e+07	1.023968e+07	420511.764706	4827.058824
Música y baile	2.683688e+07	3.743881e+05	17405.681250	1998.931250
Diseño/arte, DIY y Life Hacks	2.570000e+07	2.600000e+06	127300.000000	2200.000000
Educación	2.501250e+07	1.106042e+06	45060.750000	1537.250000
Videojuegos	2.498421e+07	1.387137e+06	57121.052632	1760.157895
Videojuegos, Juguetes	2.473333e+07	5.741667e+05	6400.000000	337.000000
Belleza, Moda	2.390000e+07	9.645000e+05	62300.000000	
Películas, Animación	2.269344e+07	5.513295e+05	25671.016393	
Películas, Juguetes	2.130000e+07	6.264667e+05	1332.888889	
Películas	2.114167e+07	7.758458e+05	28829.208333	
Animación, Humor	2.078519e+07	3.760126e+06	145768.333333	
Viajes, Espectáculos	2.040000e+07	8.950000e+04	782.000000	
Música y baile, Animación	2.040000e+07	6.957188e+05	17155.000000	
Comida y bebida, Salud y autoayuda	2.010000e+07	1.149000e+05	2800.000000	
Diseño/arte	2.010000e+07	1.785000e+05	7300.000000	
Música y baile, Películas	1.947561e+07	4.405902e+05	17966.414634	
Animación, Videojuegos	1.939412e+07	1.200059e+06	79294.029412	
DIY y Life Hacks, Juguetes	1.910000e+07	2.300000e+06	33200.000000	
Noticias y Política	1.878056e+07	2.187333e+05	10353.222222	
Música y baile, Humor	1.838333e+07	2.402933e+06	45783.333333	
Películas, Humor	1.829706e+07	9.387235e+05	40684.617647	
Educación, Juguetes	1.805000e+07	4.697500e+05	2185.000000	
Vlogs diarios	1.770000e+07	3.414338e+06	187244.945946	
Animación	1.764091e+07	6.367182e+05	21413.454545	
Música y baile, Juguetes	1.730000e+07	5.250000e+04	129.000000	
Ciencia y tecnología	1.726429e+07	8.871286e+05	59283.142857	
Fitness, Salud y autoayuda	1.710000e+07	1.946667e+05	7600.000000	
Fitness	1.635000e+07	8.620000e+04	3750.000000	
Comida y bebida	1.612500e+07	2.722450e+06	128664.750000	
Animales y mascotas	1.560000e+07	2.231450e+06	102750.000000	
Deportes	1.552500e+07	1.759525e+06	44949.000000	
Juguetes, Coches y vehículos	1.550000e+07	8.242500e+04	961.000000	
Humor	1.525000e+07	2.310400e+06	169990.000000	
ASMR	1.520000e+07	3.685000e+05	4100.000000	
Moda	1.440000e+07	1.726000e+05	8050.000000	
Diseño/arte, Belleza	1.440000e+07	2.700000e+06	152400.000000	
Animación, Humor, Juguetes	1.390000e+07	8.000000e+03	37.000000	
Coches y vehículos	1.320000e+07	2.664000e+05	18150.000000	
DIY y Life Hacks	1.306667e+07	7.146667e+04	1616.333333	
ASMR, Comida y bebida	1.300000e+07	5.575000e+05	8600.000000	
Comida y bebida, Juguetes	1.230000e+07	4.690000e+04	176.000000	
Juguetes, DIY y Life Hacks	1.220000e+07	6.260000e+04	256.000000	

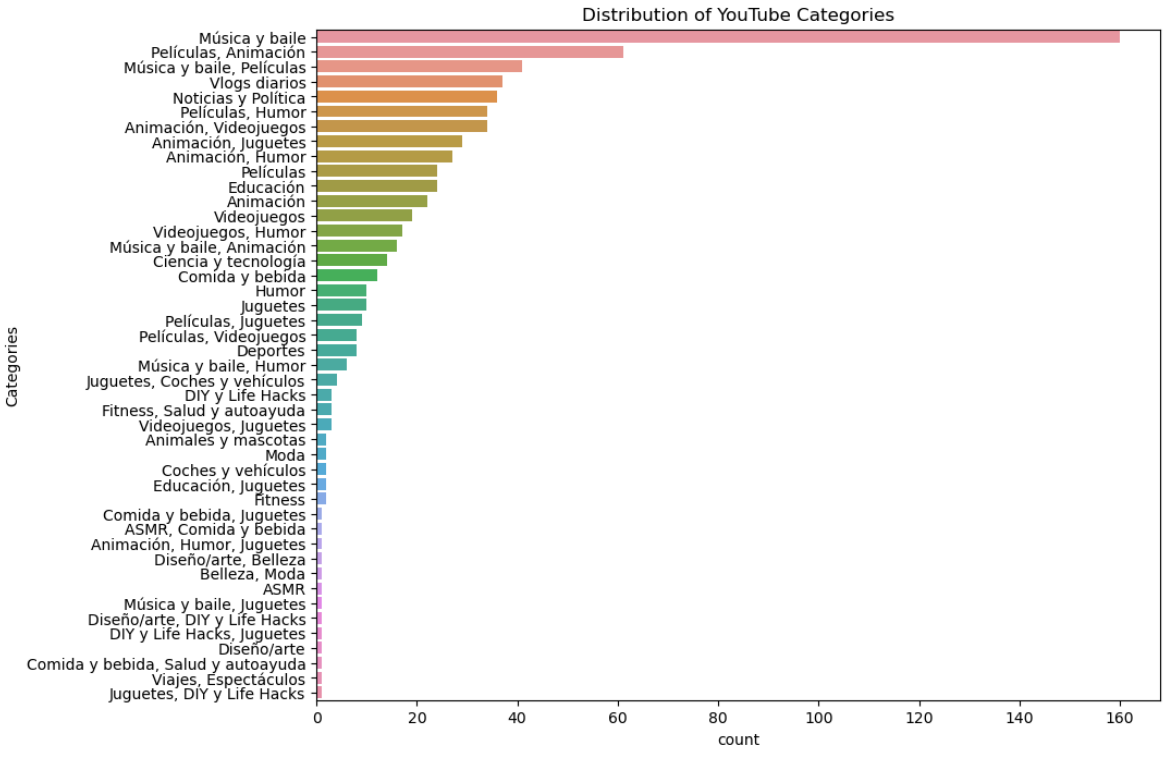
```
Belleza, Moda                          1100.000000
Películas, Animación                    645.655738
Películas, Juguetes                       0.000000
Películas                              1081.041667
Animación, Humor                       5344.962963
Viajes, Espectáculos                     49.000000
Música y baile, Animación               589.437500
Comida y bebida, Salud y autoayuda      117.000000
Diseño/arte                             140.000000
Música y baile, Películas               457.195122
Animación, Videojuegos                 3786.617647
DIY y Life Hacks, Juguetes             2100.000000
Noticias y Política                     358.916667
Música y baile, Humor                  2110.500000
Películas, Humor                       1008.794118
Educación, Juguetes                       0.000000
Vlogs diarios                           980.405405
Animación                               396.636364
Música y baile, Juguetes                  0.000000
Ciencia y tecnología                   1363.571429
Fitness, Salud y autoayuda              532.000000
Fitness                                  63.500000
Comida y bebida                        3053.416667
Animales y mascotas                    2806.000000
Deportes                                136.500000
Juguetes, Coches y vehículos              0.000000
Humor                                  5159.800000
ASMR                                    148.000000
Moda                                    218.500000
Diseño/arte, Belleza                   1100.000000
Animación, Humor, Juguetes                0.000000
Coches y vehículos                      439.500000
DIY y Life Hacks                         47.666667
ASMR, Comida y bebida                   349.000000
Comida y bebida, Juguetes                 0.000000
Juguetes, DIY y Life Hacks                0.000000  ,
             Suscribers    Visits     Likes  Comments
Suscribers     1.000000  0.286039  0.248389  0.037293
Visits         0.286039  1.000000  0.966643  0.319666
Likes          0.248389  0.966643  1.000000  0.311424
Comments       0.037293  0.319666  0.311424  1.000000)
```

In [28]:

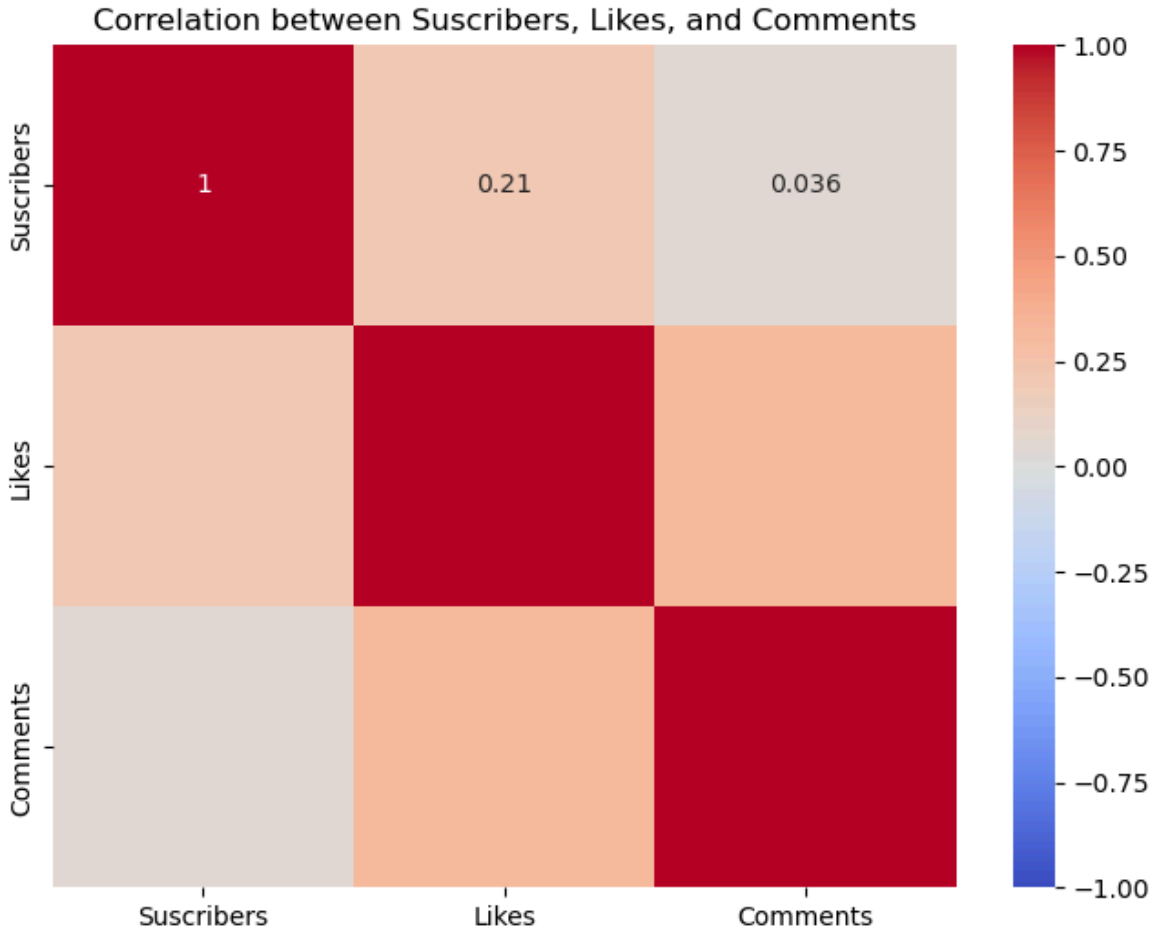
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import pearsonr

# Load the dataset
path=r'C:\Users\DELL\Documents\Inter Career Data Analysis\youtubers_df.csv'

data=pd.read_csv(path)


# Display basic info about the dataset
print(data.info())

# Plot the distribution of categories
plt.figure(figsize=(10,8))
sns.countplot(y='Categories', data=data, order=data['Categories'].value_counts()
plt.title('Distribution of YouTube Categories')
```

```
plt.show()

# Correlation between Subscribers, Likes, and Comments
correlation_matrix = data[['Suscribers', 'Likes', 'Comments']].corr()
print("Correlation Matrix:")
print(correlation_matrix)

# Visualize the correlation matrix
plt.figure(figsize=(8,6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation between Suscribers, Likes, and Comments')
plt.show()

# Compute the Pearson correlation for Subscribers and Likes
corr_subs_likes, _ = pearsonr(data['Suscribers'], data['Likes'])
print(f"Pearson Correlation (Subscribers vs Likes): {corr_subs_likes:.2f}")

# Compute the Pearson correlation for Subscribers and Comments
corr_subs_comments, _ = pearsonr(data['Suscribers'], data['Comments'])
print(f"Pearson Correlation (Subscribers vs Comments): {corr_subs_comments:.2f}"
```

```
<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
None
```



```
Correlation Matrix:
            Suscribers     Likes  Comments
Suscribers    1.000000  0.211639  0.036350
Likes         0.211639  1.000000  0.325911
Comments      0.036350  0.325911  1.000000
```



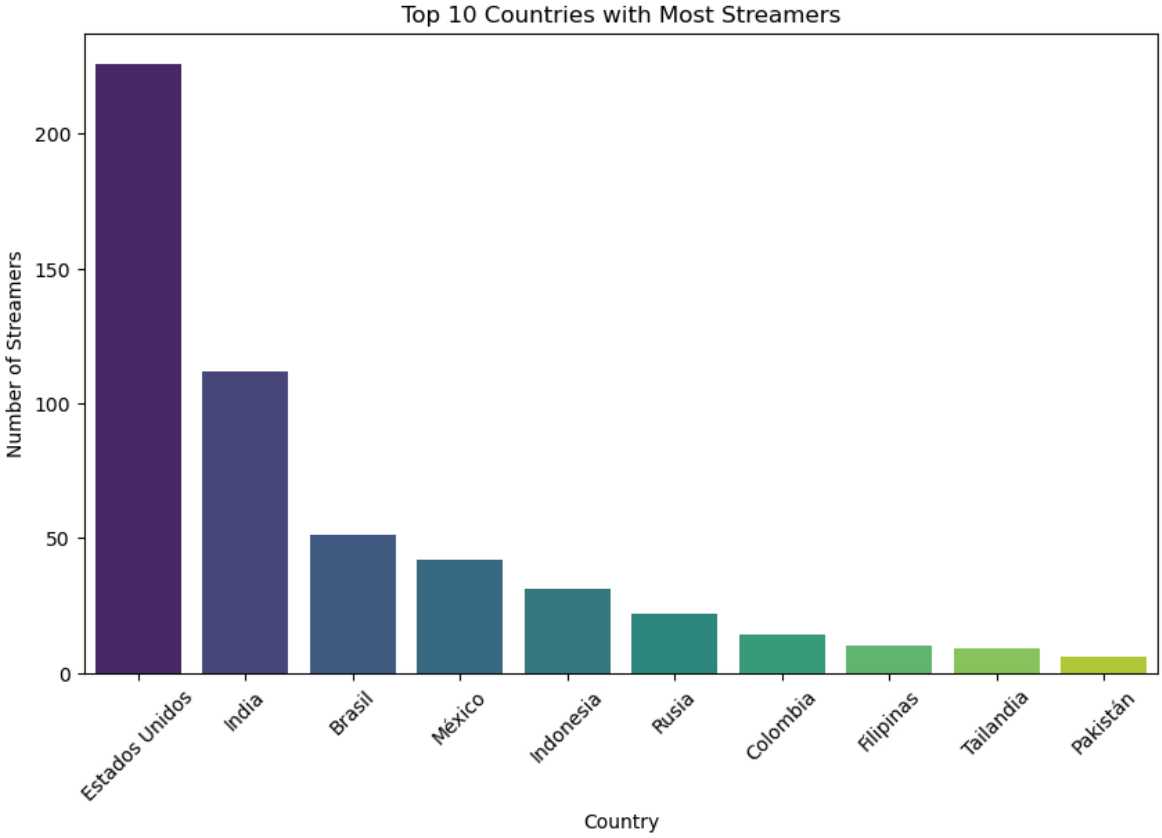

```
Pearson Correlation (Subscribers vs Likes): 0.21
Pearson Correlation (Subscribers vs Comments): 0.04
```

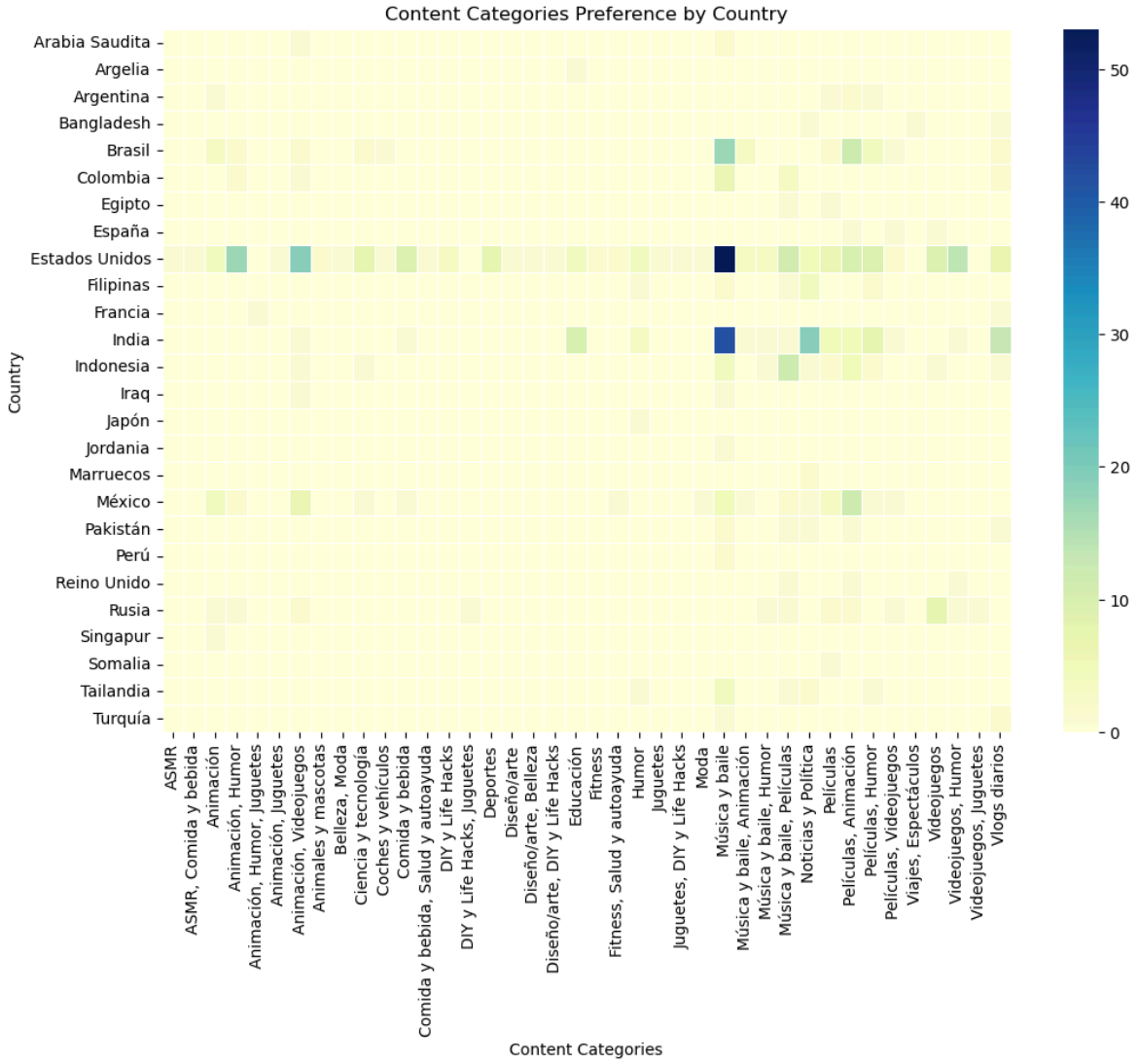
3. Audience Study:-

Analyze the distribution of streamers' audiences by country. Are there regional preferences for
specific content categories?

In [4]:
```python
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

path=r'C:\Users\DELL\Documents\Inter Career Data Analysis\youtubers_df.csv'

data = pd.read_csv(path)


# Clean up 'Country' and 'Categories' columns
data = data.dropna(subset=['Categories'])
data = data[data['Country'] != 'Unknown']

# Analyze audience distribution by country
country_dist = data['Country'].value_counts().head(10)
```

```python
# Plotting distribution of audiences by country
plt.figure(figsize=(10, 6))
sns.barplot(x=country_dist.index, y=country_dist.values, palette="viridis")
plt.title('Top 10 Countries with Most Streamers')
plt.xlabel('Country')
plt.ylabel('Number of Streamers')
plt.xticks(rotation=45)
plt.show()

# Analyze content preferences by country
category_country_dist = data.groupby(['Country', 'Categories']).size().unstack()

# Plotting heatmap of content categories by country
plt.figure(figsize=(12, 8))
sns.heatmap(category_country_dist, cmap='YlGnBu', linewidths=0.5)
plt.title('Content Categories Preference by Country')
plt.xlabel('Content Categories')
plt.ylabel('Country')
plt.xticks(rotation=90)
plt.show()
```



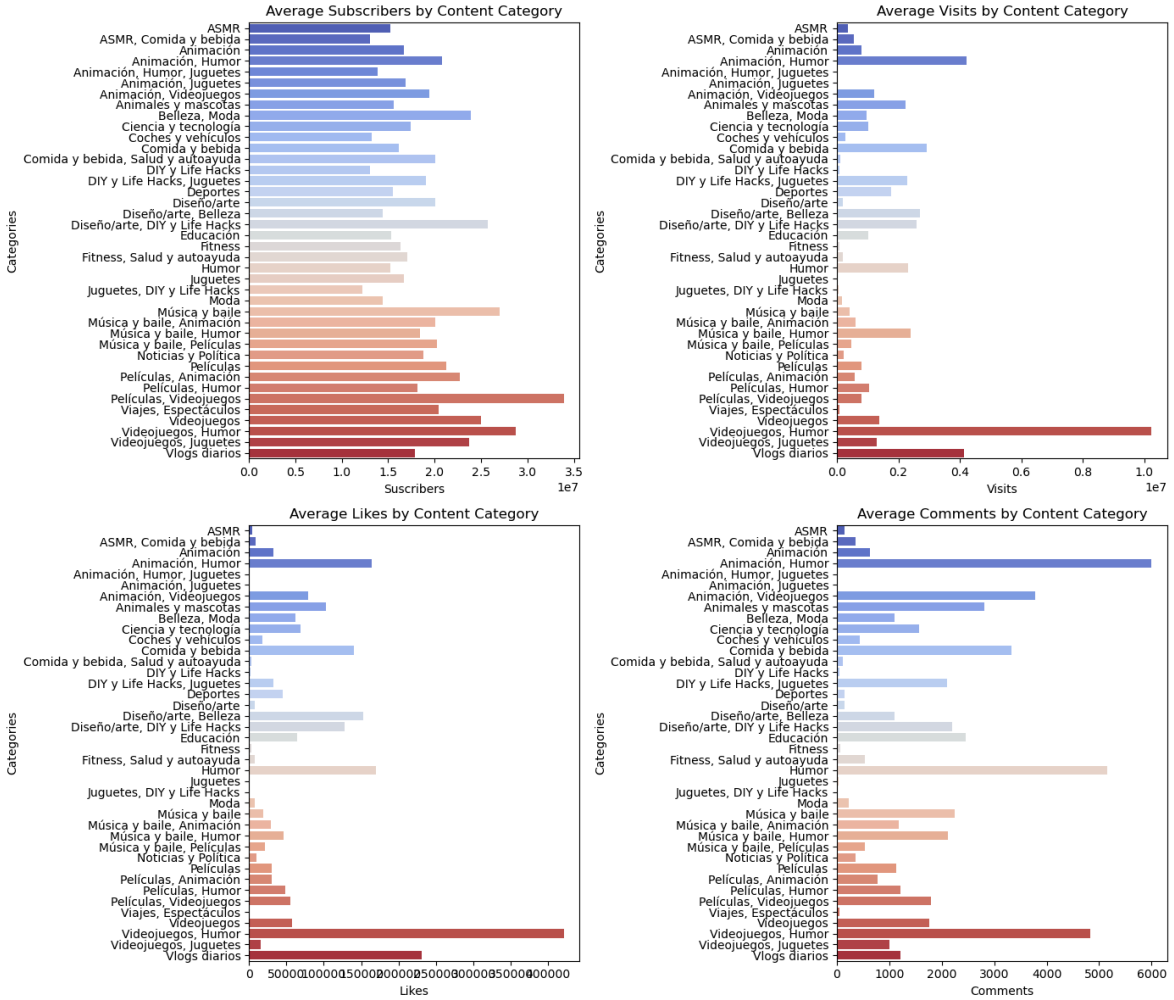


4. Performance Metrics:

- Calculate and visualize the average number of subscribers, visits, likes, and comments. - Are there patterns or anomalies in these met

In [5]:
```
# Group the data by content categories and calculate the average performance met
avg_metrics = data.groupby('Categories').agg({
    'Suscribers': 'mean',
    'Visits': 'mean',
    'Likes': 'mean',
    'Comments': 'mean'
}).reset_index()

# Plotting the average metrics for each content category
fig, axes = plt.subplots(2, 2, figsize=(14, 12))

# Average subscribers by content category
sns.barplot(x='Suscribers', y='Categories', data=avg_metrics, ax=axes[0, 0], pal
axes[0, 0].set_title('Average Subscribers by Content Category')

# Average visits by content category
sns.barplot(x='Visits', y='Categories', data=avg_metrics, ax=axes[0, 1], palette
axes[0, 1].set_title('Average Visits by Content Category')
```

```
# Average likes by content category
sns.barplot(x='Likes', y='Categories', data=avg_metrics, ax=axes[1, 0], palette=
axes[1, 0].set_title('Average Likes by Content Category')

# Average comments by content category
sns.barplot(x='Comments', y='Categories', data=avg_metrics, ax=axes[1, 1], palet
axes[1, 1].set_title('Average Comments by Content Category')

plt.tight_layout()
plt.show()

# Check for patterns or anomalies by looking at the summary statistics
avg_metrics.describe()
```



Out[5]:

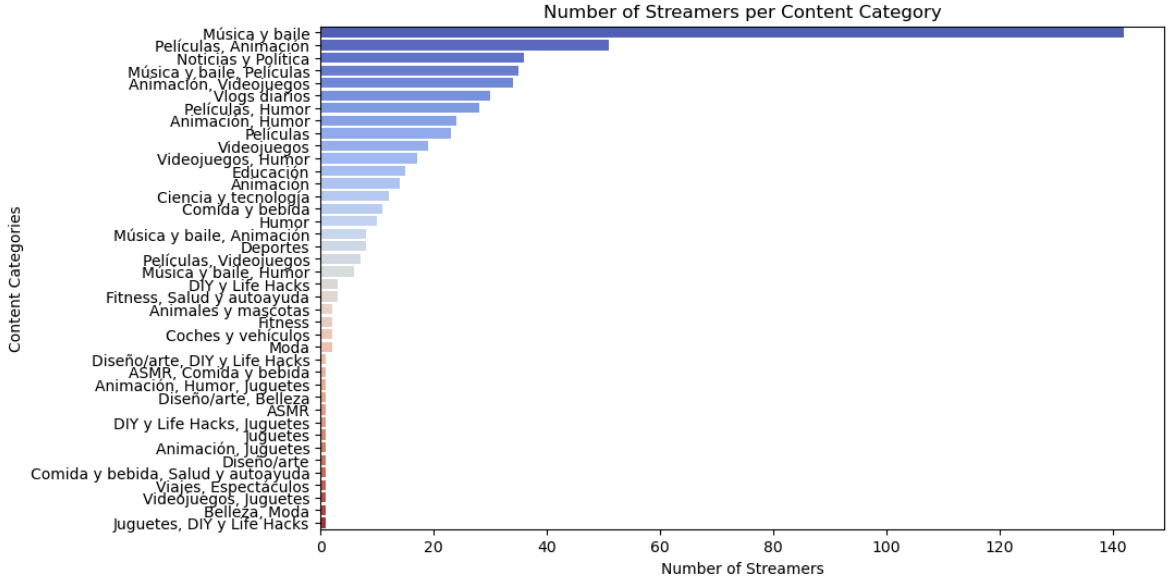
	Suscribers	Visits	Likes	Comments
count	4.000000e+01	4.000000e+01	40.000000	40.000000
mean	1.884249e+07	1.316394e+06	57959.133001	1365.858964
std	4.698334e+06	1.829049e+06	81544.355962	1513.582293
min	1.220000e+07	8.000000e+03	0.000000	0.000000
25%	1.548042e+07	1.906250e+05	7525.000000	146.000000
50%	1.800940e+07	7.890036e+05	30095.208440	1050.000000
75%	2.049479e+07	1.877506e+06	65467.287500	2102.625000
max	3.390000e+07	1.023968e+07	420511.764706	6001.666667

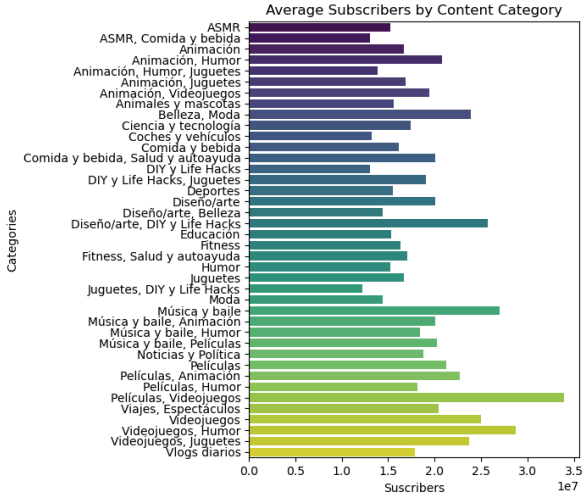
5. Content Categories:-

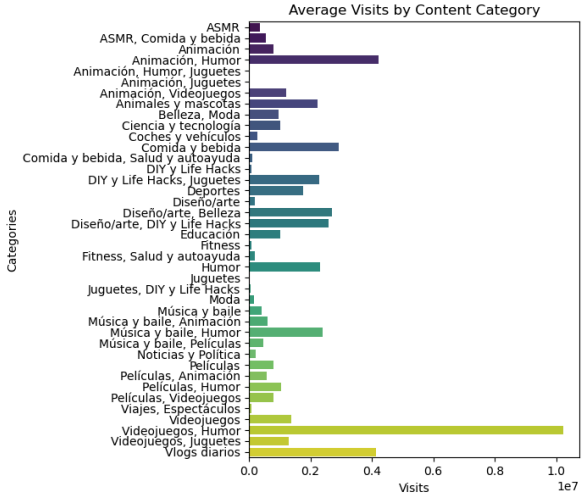
Explore the distribution of content categories. Which categories have the highest number of
streamers?- Are there specific categories with exceptional performance metrics?

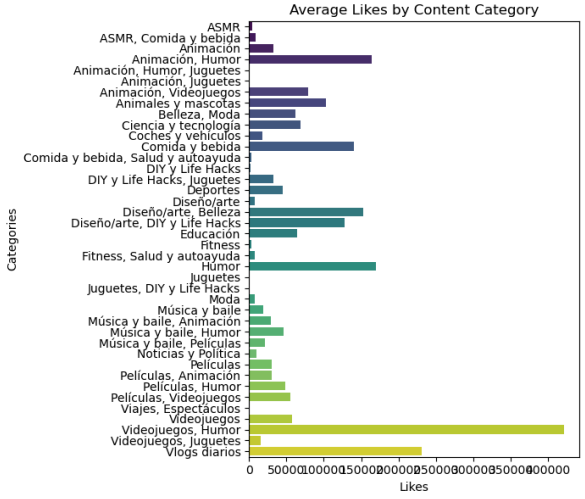
In [8]:
```
# Distribution of content categories - Number of streamers per category
category_dist = data['Categories'].value_counts()

# Plotting distribution of content categories
plt.figure(figsize=(10, 6))
sns.barplot(x=category_dist.values, y=category_dist.index, palette="coolwarm")
plt.title('Number of Streamers per Content Category')
plt.xlabel('Number of Streamers')
plt.ylabel('Content Categories')
plt.show()

# Performance metrics by content category
# Group by content categories and calculate average metrics
category_performance = data.groupby('Categories').agg({
    'Suscribers': 'mean',
    'Visits': 'mean',
    'Likes': 'mean',
    'Comments': 'mean'
}).reset_index()

# Plotting the performance metrics for each content category
fig, axes = plt.subplots(2, 2, figsize=(14, 12))

# Average subscribers by content category
sns.barplot(x='Suscribers', y='Categories', data=category_performance, ax=axes[0
axes[0, 0].set_title('Average Subscribers by Content Category')

# Average visits by content category
sns.barplot(x='Visits', y='Categories', data=category_performance, ax=axes[0, 1]
axes[0, 1].set_title('Average Visits by Content Category')

# Average likes by content category
sns.barplot(x='Likes', y='Categories', data=category_performance, ax=axes[1, 0],
```

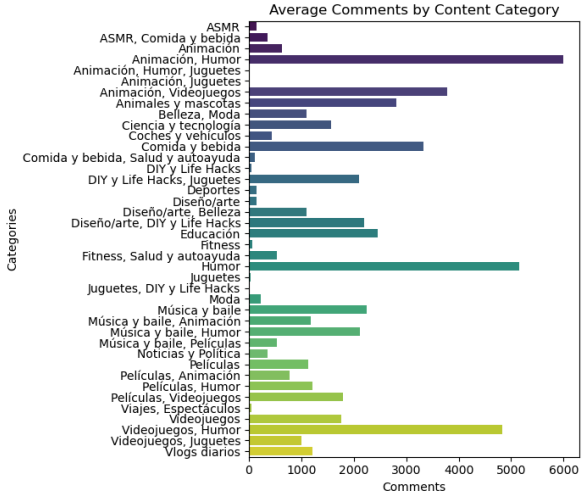
```
axes[1, 0].set_title('Average Likes by Content Category')

# Average comments by content category
sns.barplot(x='Comments', y='Categories', data=category_performance, ax=axes[1,
axes[1, 1].set_title('Average Comments by Content Category')

plt.tight_layout()
plt.show()

# Checking for any content categories with exceptional performance
category_performance.describe()
```











Out[8]:

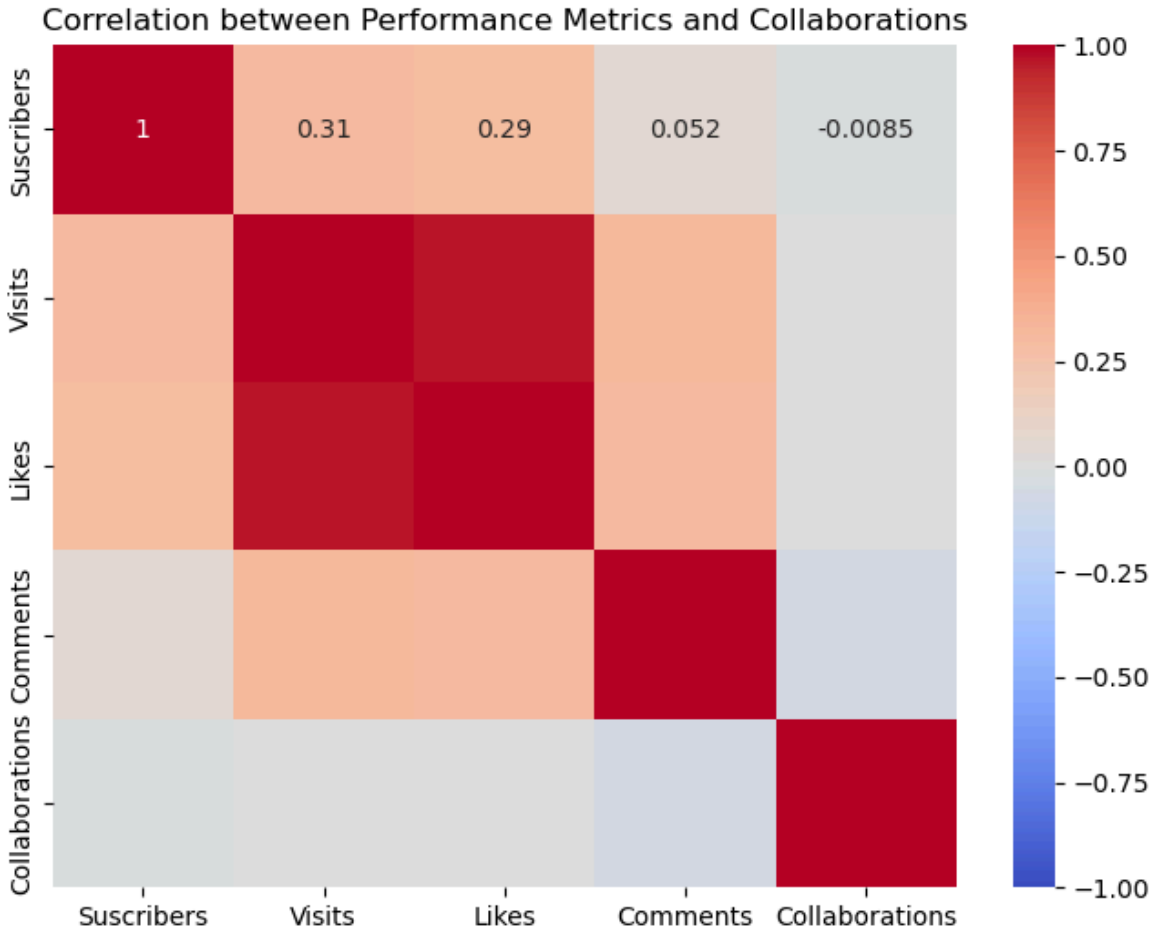
	Suscribers	Visits	Likes	Comments
count	4.000000e+01	4.000000e+01	40.000000	40.000000
mean	1.884249e+07	1.316394e+06	57959.133001	1365.858964
std	4.698334e+06	1.829049e+06	81544.355962	1513.582293
min	1.220000e+07	8.000000e+03	0.000000	0.000000
25%	1.548042e+07	1.906250e+05	7525.000000	146.000000
50%	1.800940e+07	7.890036e+05	30095.208440	1050.000000
75%	2.049479e+07	1.877506e+06	65467.287500	2102.625000
max	3.390000e+07	1.023968e+07	420511.764706	6001.666667

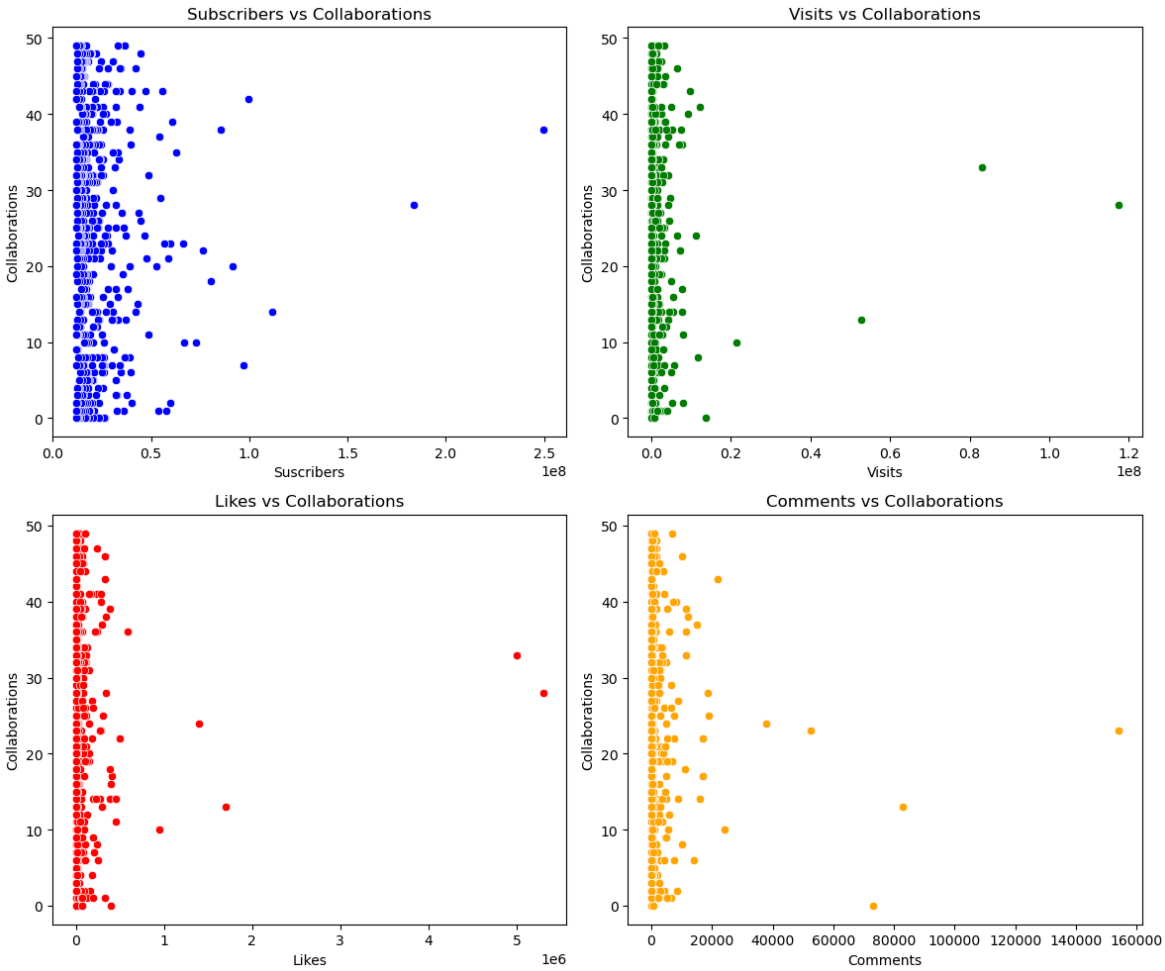
6.Brands and Collaborations:-

Analyze whether streamers with high performance metrics receive more brand collaborations and
marketing campaigns

In [14]:
```
# Assuming we have a 'Collaborations' column in the dataset
# Add a dummy 'Collaborations' column for the sake of analysis
import numpy as np
np.random.seed(42)  # For reproducibility

# Adding a synthetic 'Collaborations' column for the sake of this example
# In a real scenario, you'd replace this with actual data
data['Collaborations'] = np.random.randint(0, 50, size=len(data))

# Calculate correlations between performance metrics and collaborations
correlation_matrix = data[['Suscribers', 'Visits', 'Likes', 'Comments', 'Collabo

# Plot heatmap of the correlation matrix
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation between Performance Metrics and Collaborations')
plt.show()

# Scatter plots to visualize the relationship between each performance metric an
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# Subscribers vs Collaborations
sns.scatterplot(x='Suscribers', y='Collaborations', data=data, ax=axes[0, 0], co
axes[0, 0].set_title('Subscribers vs Collaborations')

# Visits vs Collaborations
sns.scatterplot(x='Visits', y='Collaborations', data=data, ax=axes[0, 1], color=
axes[0, 1].set_title('Visits vs Collaborations')

# Likes vs Collaborations
sns.scatterplot(x='Likes', y='Collaborations', data=data, ax=axes[1, 0], color='
axes[1, 0].set_title('Likes vs Collaborations')
```

```
# Comments vs Collaborations
sns.scatterplot(x='Comments', y='Collaborations', data=data, ax=axes[1, 1], colo
axes[1, 1].set_title('Comments vs Collaborations')

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





7. Benchmarking:-

Identify streamers with above-average performance in terms of subscribers, visits, likes, and comments.- Who are the top-performing content creators?

In [16]:
```
# Calculate average metrics for all streamers
average_metrics = data[['Suscribers', 'Visits', 'Likes', 'Comments']].mean()

# Filter streamers who have above-average performance
above_avg_streamers = data[
    (data['Suscribers'] > average_metrics['Suscribers']) &
    (data['Visits'] > average_metrics['Visits']) &
    (data['Likes'] > average_metrics['Likes']) &
    (data['Comments'] > average_metrics['Comments'])
]

# Sorting by subscribers for top performance
top_performers = above_avg_streamers.sort_values(by='Suscribers', ascending=Fals

# Display the top 10 performing streamers
top_10_performers = top_performers[['Username', 'Categories', 'Suscribers', 'Vis
print(top_10_performers)
```

	Username	Categories	Suscribers	Visits	\
1	MrBeast	Videojuegos, Humor	183500000.0	117400000.0	
5	PewDiePie	Películas, Videojuegos	111500000.0	2400000.0	
26	dudeperfect	Videojuegos	59700000.0	5300000.0	
34	TaylorSwift	Música y baile	54100000.0	4300000.0	
39	JuegaGerman	Películas, Animación	48600000.0	2000000.0	
43	A4a4a4a4	Animación, Humor	47300000.0	9700000.0	
62	KimberlyLoaiza	Música y baile	42100000.0	5300000.0	
96	TotalGaming093	Películas, Videojuegos	36300000.0	1500000.0	
100	markiplier	Animación, Videojuegos	35500000.0	2100000.0	
122	AboFlah	Animación, Videojuegos	32700000.0	3300000.0	

	Likes	Comments
1	5300000.0	18500.0
5	197300.0	4900.0
26	156500.0	4200.0
34	300400.0	15000.0
39	117100.0	3000.0
43	330400.0	22000.0
62	271300.0	16000.0
96	129400.0	4900.0
100	126500.0	3800.0
122	382000.0	11400.0

8. Content Recommendations:-

Propose a system for enhancing content recommendations to YouTube users based on streamers' categories and performance metrics.

In [18]:
```
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer

# Create a TF-IDF matrix based on the 'Categories' column
tfidf_vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf_vectorizer.fit_transform(data['Categories'].fillna(''))

# Calculate similarity between streamers based on content categories
cosine_similarities = cosine_similarity(tfidf_matrix)

# Recommendation function
def recommend_streamers(username, cosine_similarities, data, top_n=5):
    # Get the index of the streamer
    idx = data.index[data['Username'] == username].tolist()[0]

    # Get similarity scores for this streamer
    similarity_scores = list(enumerate(cosine_similarities[idx]))

    # Sort streamers based on similarity scores
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=Tr

    # Get the top N streamers
    top_similar_streamers = [i[0] for i in similarity_scores[1:top_n+1]]

    # Return the top similar streamers
    return data.iloc[top_similar_streamers][['Username', 'Categories', 'Suscribe

# Example: Recommend streamers similar to 'MrBeast'
```

```
recommendations = recommend_streamers('MrBeast', cosine_similarities, data, top_
print(recommendations)
```

	Username	Categories	Suscribers
179	brentrivera	Videojuegos, Humor	27600000.0
219	PrestonYT	Videojuegos, Humor	24900000.0
234	rug	Videojuegos, Humor	24300000.0
278	StokesTwins	Videojuegos, Humor	22700000.0
285	BenAzelart	Videojuegos, Humor	22500000.0

In [ ]: