In [1]:
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df= pd.read_csv(r"C:\Users\Dell\Desktop\youtubers.csv")
df
```

Out[2]:

	Rank	Username	Categories	Suscribers	Country	Visits	Likes	Commen
0	1	tseries	Música y baile	249500000	India	86200.0	2700	
1	2	MrBeast	Videojuegos, Humor	183500000	Estados Unidos	117400000.0	5300000	185
2	3	CoComelon	Educación	165500000	Unknown	7000000.0	24700	
3	4	SETIndia	NaN	162600000	India	15600.0	166	
4	5	KidsDianaShow	Animación, Juguetes	113500000	Unknown	3900000.0	12400	
...	...	...	...	...	...	...	...	
995	996	hamzymukbang	NaN	11700000	Estados Unidos	397400.0	14000	1
996	997	Adaahqueen	NaN	11700000	India	1100000.0	92500	1
997	998	LittleAngelIndonesia	Música y baile	11700000	Unknown	211400.0	745	
998	999	PenMultiplex	NaN	11700000	India	14000.0	81	
999	1000	OneindiaHindi	Noticias y Política	11700000	India	2200.0	31	

1000 rows × 9 columns

In [3]:
```
df.head()
```

Out[3]:

	Rank	Username	Categories	Suscribers	Country	Visits	Likes	Comments	
0	1	tseries	Música y baile	249500000	India	86200.0	2700	78	
1	2	MrBeast	Videojuegos, Humor	183500000	Estados Unidos	117400000.0	5300000	18500	htt
2	3	CoComelon	Educación	165500000	Unknown	7000000.0	24700	0	h
3	4	SETIndia	NaN	162600000	India	15600.0	166	9	htt
4	5	KidsDianaShow	Animación, Juguetes	113500000	Unknown	3900000.0	12400	0	

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

Out[4]:
```
(1000, 9)
```

In [5]:
```
len(df)
```

Out[5]:
```
1000
```

In [6]:
```
len(df.columns)
```

Out[6]:
```
9
```

In [7]:
```
df[['Rank','Username','Categories','Suscribers','Country','Visits','Likes','Comment
```

Out[7]:

	Rank	Username	Categories	Suscribers	Country	Visits	Likes	Commen
0	1	tseries	Música y baile	249500000	India	86200.0	2700	
1	2	MrBeast	Videojuegos, Humor	183500000	Estados Unidos	117400000.0	5300000	185
2	3	CoComelon	Educación	165500000	Unknown	7000000.0	24700	
3	4	SETIndia	NaN	162600000	India	15600.0	166	
4	5	KidsDianaShow	Animación, Juguetes	113500000	Unknown	3900000.0	12400	
...	...	...	...	...	...	...	...	
995	996	hamzymukbang	NaN	11700000	Estados Unidos	397400.0	14000	1
996	997	Adaahqueen	NaN	11700000	India	1100000.0	92500	1
997	998	LittleAngelIndonesia	Música y baile	11700000	Unknown	211400.0	745	
998	999	PenMultiplex	NaN	11700000	India	14000.0	81	
999	1000	OneindiaHindi	Noticias y Política	11700000	India	2200.0	31	

1000 rows × 9 columns

In [8]:
```
df['Username'] = df['Username'].str.replace(r'\W','')
df['Categories'] = df['Categories'].str.replace(r'\W','')
df['Country'] = df['Country'].str.replace(r'\W','')
df['Links'] = df['Links'].str.replace(r'\W','')
```

In [9]:
```
df[['Username','Categories','Country','Links']]
```

Out[9]:

	Username	Categories	Country	
0	tseries	Músicaybaile	India	httpyoutubecomchannelUCqFj5jknLsU
1	MrBeast	VideojuegosHumor	EstadosUnidos	httpyoutubecomchannelUCX6OQ3DkcsbYNE
2	CoComelon	Educación	Unknown	httpyoutubecomchannelUCbCmjCuTUZos(
3	SETIndia	NaN	India	httpyoutubecomchannelUCpEhnqL0y41EpW
4	KidsDianaShow	AnimaciónJuguetes	Unknown	httpyoutubecomchannelUCk8GzjMOrta
...	...	...	...	
995	hamzymukbang	NaN	EstadosUnidos	httpyoutubecomchannelUCPKNKldggio
996	Adaahqueen	NaN	India	httpyoutubecomchannelUCk3fFpqI5kDM
997	LittleAngelIndonesia	Músicaybaile	Unknown	httpyoutubecomchannelUCdrHrQf0o0TO
998	PenMultiplex	NaN	India	httpyoutubecomchannelUCObyBrdrtQ20BU
999	OneindiaHindi	NoticiasyPolítica	India	httpyoutubecomchannelUCOjgc1p2hJ4GZi(

1000 rows × 4 columns

In [10]:
```
#for charactical value
df['Username'] = df['Username'].fillna(df['Username'].mode()[0])
df['Categories'] = df['Categories'].fillna(df['Categories'].mode()[0])
df['Country'] = df['Country'].fillna(df['Country'].mode()[0])
df['Links'] = df['Links'].fillna(df['Links'].mode()[0])
#for Numerical Value
df['Rank'] = df['Rank'].fillna(np.mean(pd.to_numeric(df['Rank'])))
df['Suscribers'] = df['Suscribers'].fillna(np.mean(pd.to_numeric(df['Suscribers'])))
df['Visits'] = df['Visits'].fillna(np.mean(pd.to_numeric(df['Visits'])))
df['Likes'] = df['Likes'].fillna(np.mean(pd.to_numeric(df['Likes'])))
df['Comments'] = df['Comments'].fillna(np.mean(pd.to_numeric(df['Comments'])))
```

In [11]:
```
df[['Rank','Username','Categories','Suscribers','Country','Visits','Likes','Comment
```

Out[11]:

	Rank	Username	Categories	Suscribers	Country	Visits	Likes
0	1	tseries	Músicaybaile	249500000	India	86200.0	2700
1	2	MrBeast	VideojuegosHumor	183500000	EstadosUnidos	117400000.0	5300000
2	3	CoComelon	Educación	165500000	Unknown	7000000.0	24700
3	4	SETIndia	Músicaybaile	162600000	India	15600.0	166
4	5	KidsDianaShow	AnimaciónJuguetes	113500000	Unknown	3900000.0	12400
...	...	...	...	...	...	...	...
995	996	hamzymukbang	Músicaybaile	11700000	EstadosUnidos	397400.0	14000
996	997	Adaahqueen	Músicaybaile	11700000	India	1100000.0	92500
997	998	LittleAngelIndonesia	Músicaybaile	11700000	Unknown	211400.0	745
998	999	PenMultiplex	Músicaybaile	11700000	India	14000.0	81
999	1000	OneindiaHindi	NoticiasyPolítica	11700000	India	2200.0	31

1000 rows × 9 columns

In [12]:
```
df['Rank'] = df['Rank'].astype(int)
df['Suscribers'] = df['Suscribers'].astype(int)
df['Visits'] = df['Visits'].astype(int)
df['Likes'] = df['Likes'].astype(int)
df['Comments'] = df['Comments'].astype(int)
```

In [13]:
```
df[['Rank','Suscribers','Visits','Likes','Comments']]
```

Out[13]:

	Rank	Suscribers	Visits	Likes	Comments
0	1	249500000	86200	2700	78
1	2	183500000	117400000	5300000	18500
2	3	165500000	7000000	24700	0
3	4	162600000	15600	166	9
4	5	113500000	3900000	12400	0
...	...	...	...	...	...
995	996	11700000	397400	14000	124
996	997	11700000	1100000	92500	164
997	998	11700000	211400	745	0
998	999	11700000	14000	81	1
999	1000	11700000	2200	31	1

1000 rows × 5 columns

In [14]:
```
# Our New Data Frame looking like this
df[['Rank','Username','Categories','Suscribers','Country','Visits','Likes','Comment
```

Out[14]:

	Rank	Username	Categories	Suscribers	Country	Visits	Likes
0	1	tseries	Músicaybaile	249500000	India	86200	2700
1	2	MrBeast	VideojuegosHumor	183500000	EstadosUnidos	117400000	5300000
2	3	CoComelon	Educación	165500000	Unknown	7000000	24700
3	4	SETIndia	Músicaybaile	162600000	India	15600	166
4	5	KidsDianaShow	AnimaciónJuguetes	113500000	Unknown	3900000	12400
...	...	...	...	...	...	...	...
995	996	hamzymukbang	Músicaybaile	11700000	EstadosUnidos	397400	14000
996	997	Adaahqueen	Músicaybaile	11700000	India	1100000	92500
997	998	LittleAngelIndonesia	Músicaybaile	11700000	Unknown	211400	745
998	999	PenMultiplex	Músicaybaile	11700000	India	14000	81
999	1000	OneindiaHindi	NoticiasyPolítica	11700000	India	2200	31

1000 rows × 9 columns

In [15]:
```
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   int32 
 1   Username    1000 non-null   object
 2   Categories  1000 non-null   object
 3   Suscribers  1000 non-null   int32 
 4   Country     1000 non-null   object
 5   Visits      1000 non-null   int32 
 6   Likes       1000 non-null   int32 
 7   Comments    1000 non-null   int32 
 8   Links       1000 non-null   object
dtypes: int32(5), object(4)
memory usage: 50.9+ KB
```

In [16]:
```
#check for null values or missing values
pd.isnull(df).sum()
```

Out[16]:
```
Rank          0
Username      0
Categories    0
Suscribers    0
Country       0
Visits        0
Likes         0
Comments      0
Links         0
dtype: int64
```

In [17]:
```
#drop null values
df.dropna(inplace=True)
```

In [18]:
```
#duplicate values
df.duplicated().sum()
```

Out[18]:
```
0
```

In [19]:
```
df.shape
```

Out[19]:
```
(1000, 9)
```

In [20]:
```
df.head()
```

Out[20]:

	Rank	Username	Categories	Suscribers	Country	Visits	Likes	Comm
0	1	tseries	Músicaybaile	249500000	India	86200	2700	
1	2	MrBeast	VideojuegosHumor	183500000	EstadosUnidos	117400000	5300000	18
2	3	CoComelon	Educación	165500000	Unknown	7000000	24700	
3	4	SETIndia	Músicaybaile	162600000	India	15600	166	
4	5	KidsDianaShow	AnimaciónJuguetes	113500000	Unknown	3900000	12400	

In [21]:
```
df.tail()
```

Out[21]:

	Rank	Username	Categories	Suscribers	Country	Visits	Likes	Comme
995	996	hamzymukbang	Músicaybaile	11700000	EstadosUnidos	397400	14000	
996	997	Adaahqueen	Músicaybaile	11700000	India	1100000	92500	
997	998	LittleAngelIndonesia	Músicaybaile	11700000	Unknown	211400	745	
998	999	PenMultiplex	Músicaybaile	11700000	India	14000	81	
999	1000	OneindiaHindi	NoticiasyPolítica	11700000	India	2200	31	

In [22]:
```
#describe() method returns description of the datain the DataFrame(i.e. count,mean,
df.describe()
```

Out[22]:

	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 [23]:
```
# use describe() for specific columns
df[['Suscribers', 'Visits', 'Likes', 'Comments']].describe()
```

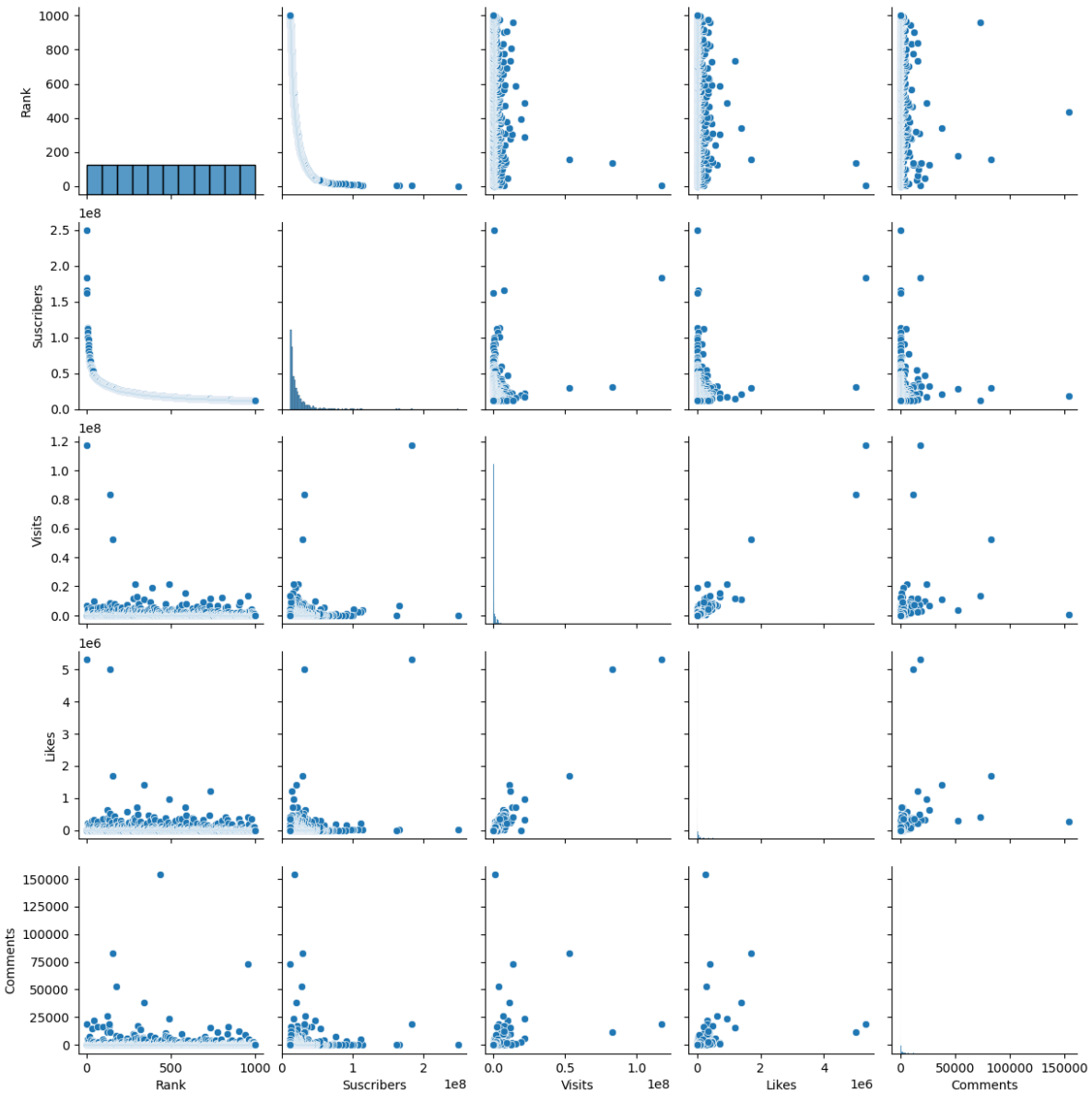
Out[23]:

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

Exploratory Data Analysis(EDA)

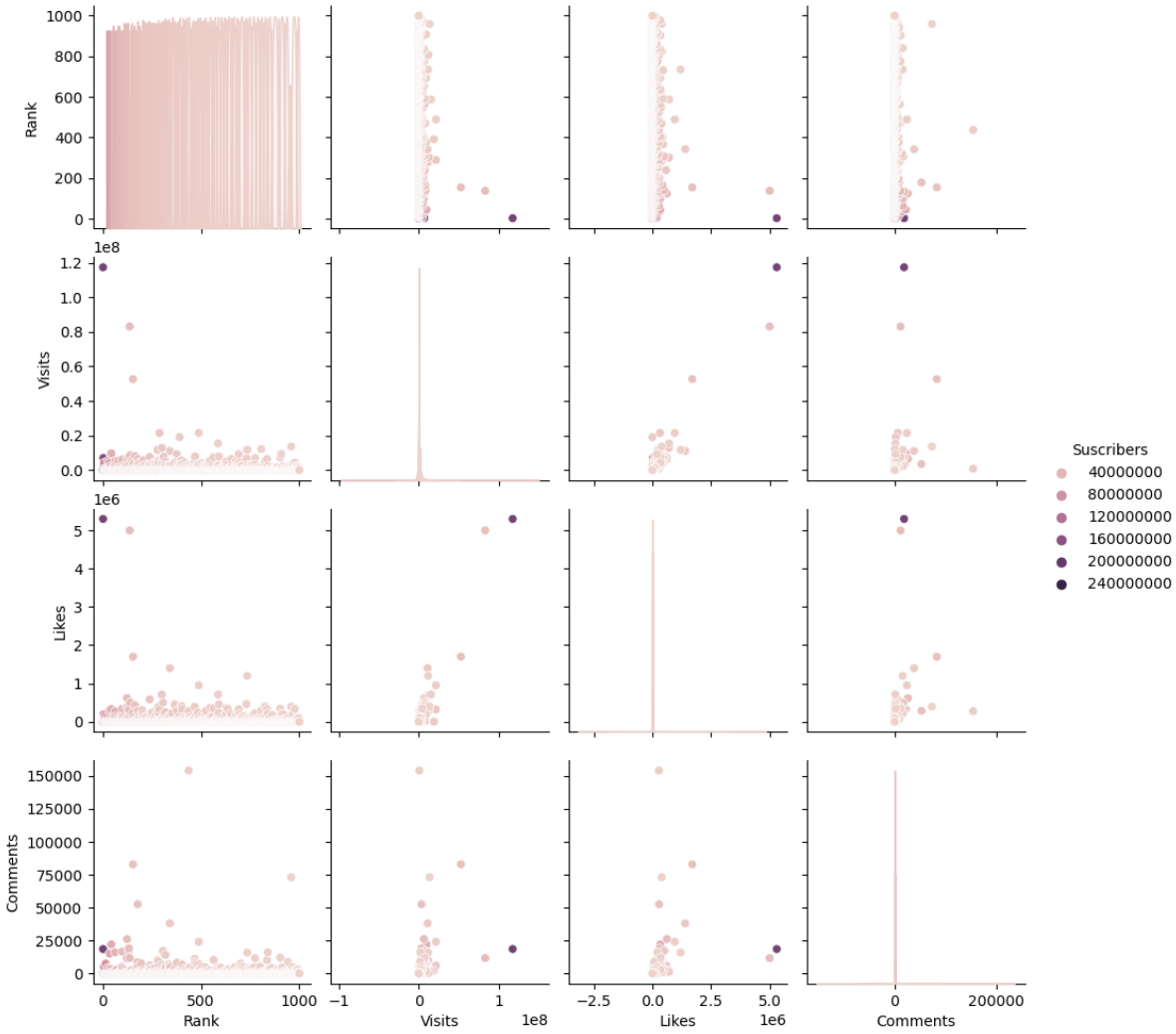
In [24]:
```
# To check Relationship
sns.pairplot(data=df, kind='scatter')
```

Out[24]:
```
<seaborn.axisgrid.PairGrid at 0x2553830b910>
```



In [25]:
```
sns.pairplot(data=df,hue='Suscribers')
```

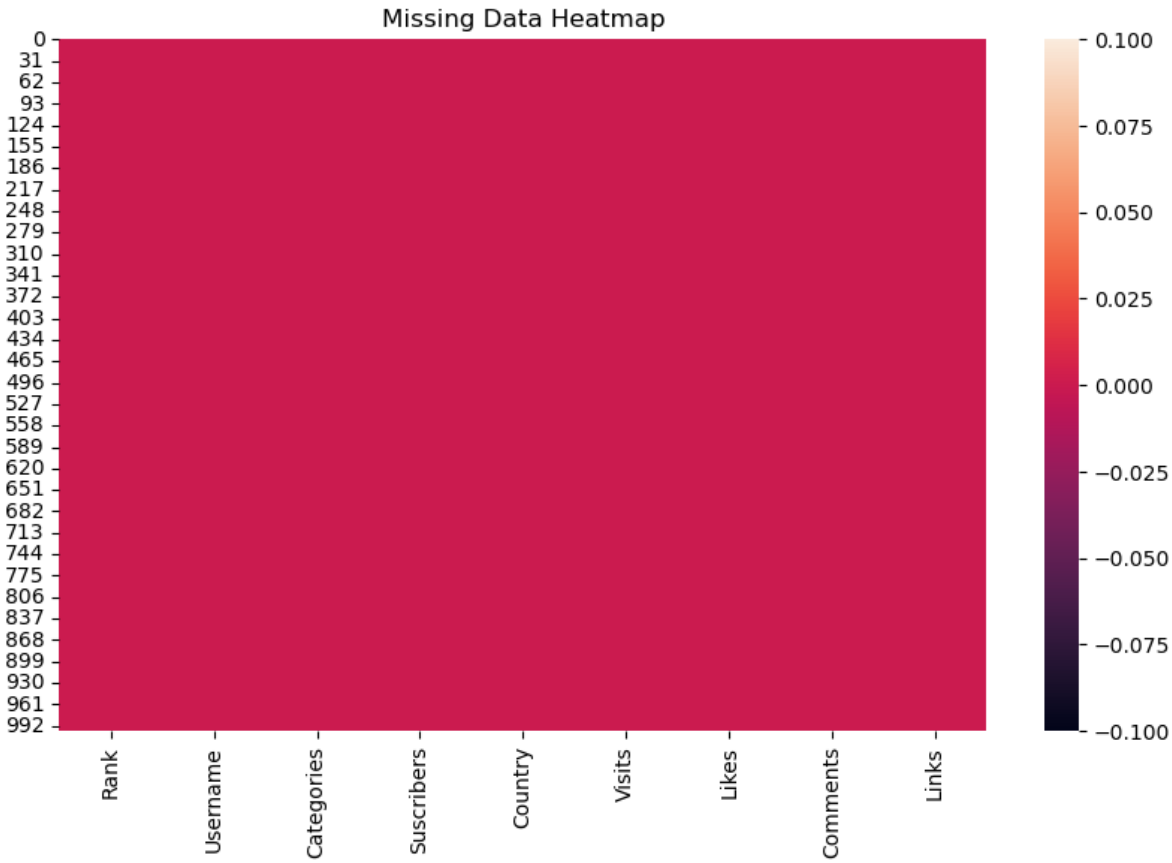
Out[25]: <seaborn.axisgrid.PairGrid at 0x2553c807490>



1. Data Exploration:

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

In [26]:
```
# Visualize missing data using a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df.isnull())
plt.title("Missing Data Heatmap")
plt.show()
```

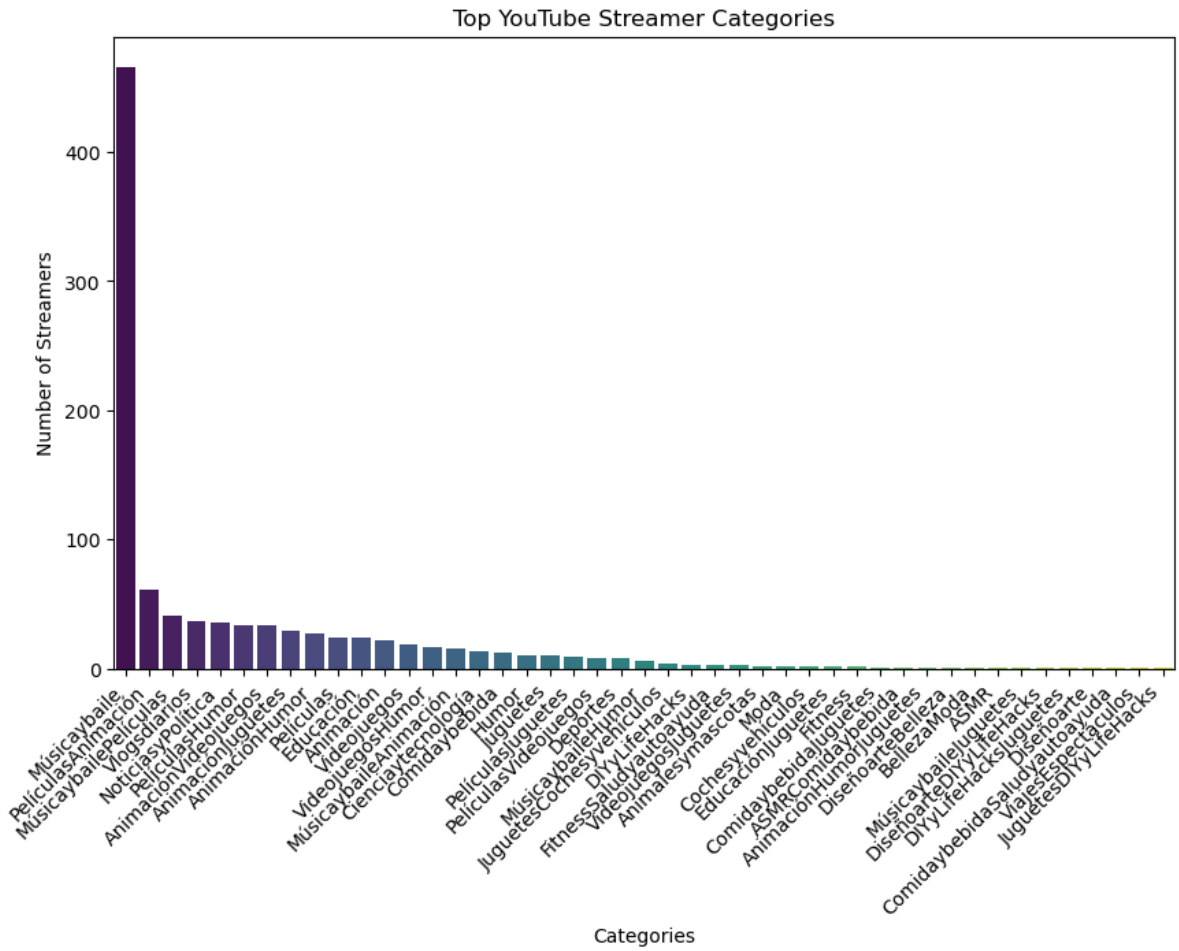


2. Trend Analysis:

- 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 [27]:
```
def analyze_category_trends(data):
    # Identify trends among the top YouTube streamers' categories
    category_counts = data['Categories'].value_counts()

    # Plot the most popular categories
    plt.figure(figsize=(10, 6))
    sns.barplot(x=category_counts.index, y=category_counts.values, palette='viridis
    plt.title('Top YouTube Streamer Categories')
    plt.xlabel('Categories')
    plt.ylabel('Number of Streamers')
    plt.xticks(rotation=45, ha='right')
    plt.show()

def analyze_correlation(data):
    # Is there a correlation between subscribers and likes or comments?
    correlation_visits_subscribers = data['Suscribers'].corr(data['Visits'])
    correlation_likes_subscribers = data['Suscribers'].corr(data['Likes'])
    correlation_comments_subscribers = data['Suscribers'].corr(data['Comments'])

    print(f'Correlation between Subscribers and Visits: {correlation_visits_subscri
    print(f'Correlation between Subscribers and Likes: {correlation_likes_subscribe
    print(f'Correlation between Subscribers and Comments: {correlation_comments_sub

if __name__ == "__main__":
```

```
    analyze_category_trends(df)
    analyze_correlation(df)
```

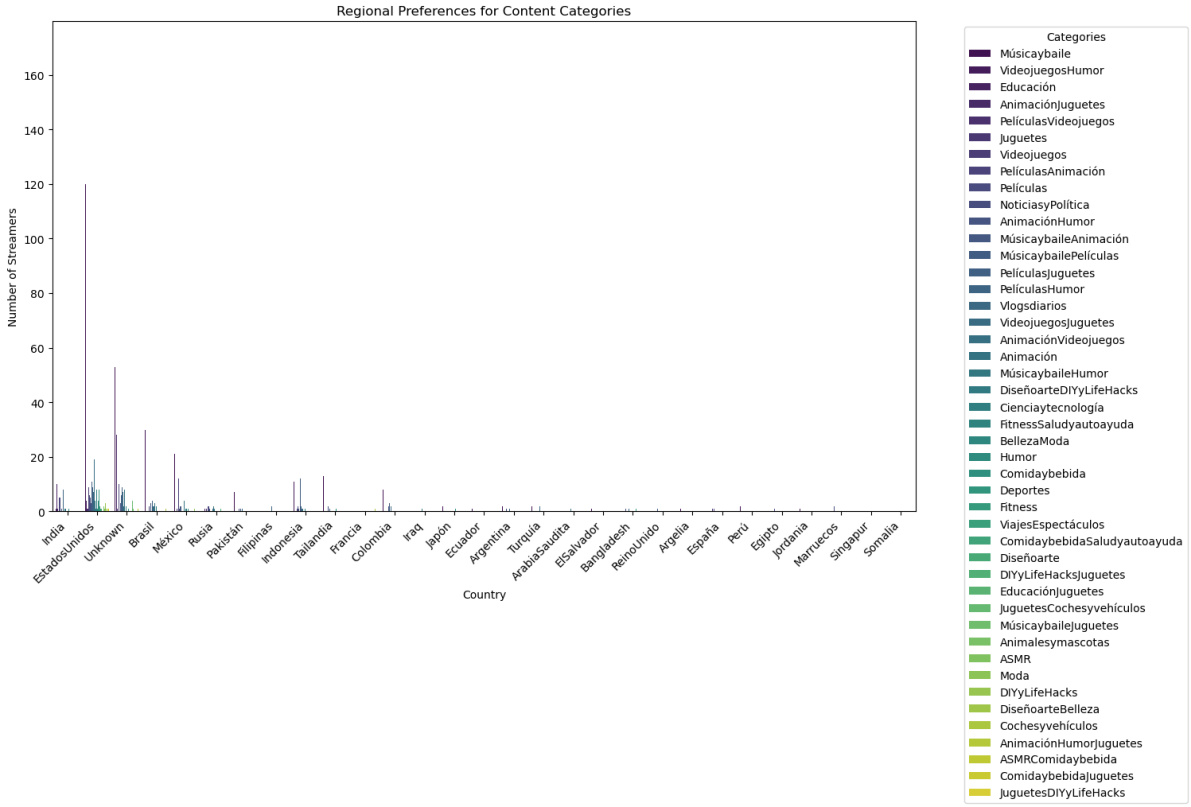



```
Correlation between Subscribers and Visits: 0.25
Correlation between Subscribers and Likes: 0.21
Correlation between Subscribers and Comments: 0.04
```

3. Audience Study:

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

In [28]:
```
def analyze_regional_preferences(data):
    # Analyze regional preferences for specific content categories
    plt.figure(figsize=(14, 8))
    sns.countplot(x='Country', hue='Categories', data=df, palette='viridis')
    plt.title('Regional Preferences for Content Categories')
    plt.xlabel('Country')
    plt.ylabel('Number of Streamers')
    plt.xticks(rotation=45, ha='right')
    plt.legend(title='Categories', bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.show()

if __name__ == "__main__":
    analyze_regional_preferences(df)
```

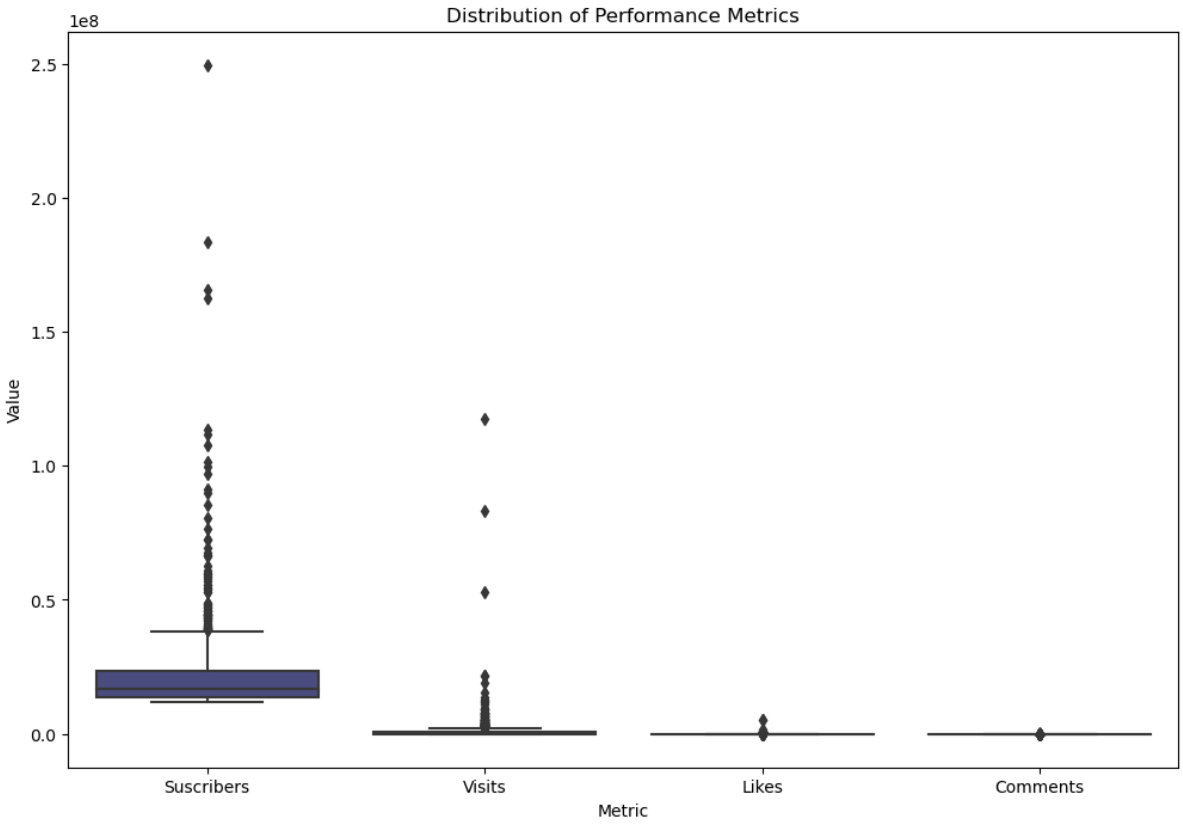


4. Performance Metrics:

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

In [29]:
```
def calculate_and_visualize_metrics(data):
    # Calculate average metrics
    avg_suscribers = data['Suscribers'].mean()
    avg_visits = data['Visits'].mean()
    avg_likes = data['Likes'].mean()
    avg_comments = data['Comments'].mean()

    print(f'Average Suscribers: {avg_suscribers:0.2f}')
    print(f'Average Visits: {avg_visits:0.2f}')
    print(f'Average Likes: {avg_likes:0.2f}')
    print(f'Average Comments: {avg_comments:0.2f}')

    # Visualize metrics
    plt.figure(figsize=(12, 8))
    metrics_data = data[['Suscribers', 'Visits', 'Likes', 'Comments']]
    sns.boxplot(x="variable", y="value", data=pd.melt(metrics_data), palette='viric
    plt.title('Distribution of Performance Metrics')
    plt.xlabel('Metric')
    plt.ylabel('Value')
    plt.show()

if __name__ == "__main__":
    calculate_and_visualize_metrics(df)
```

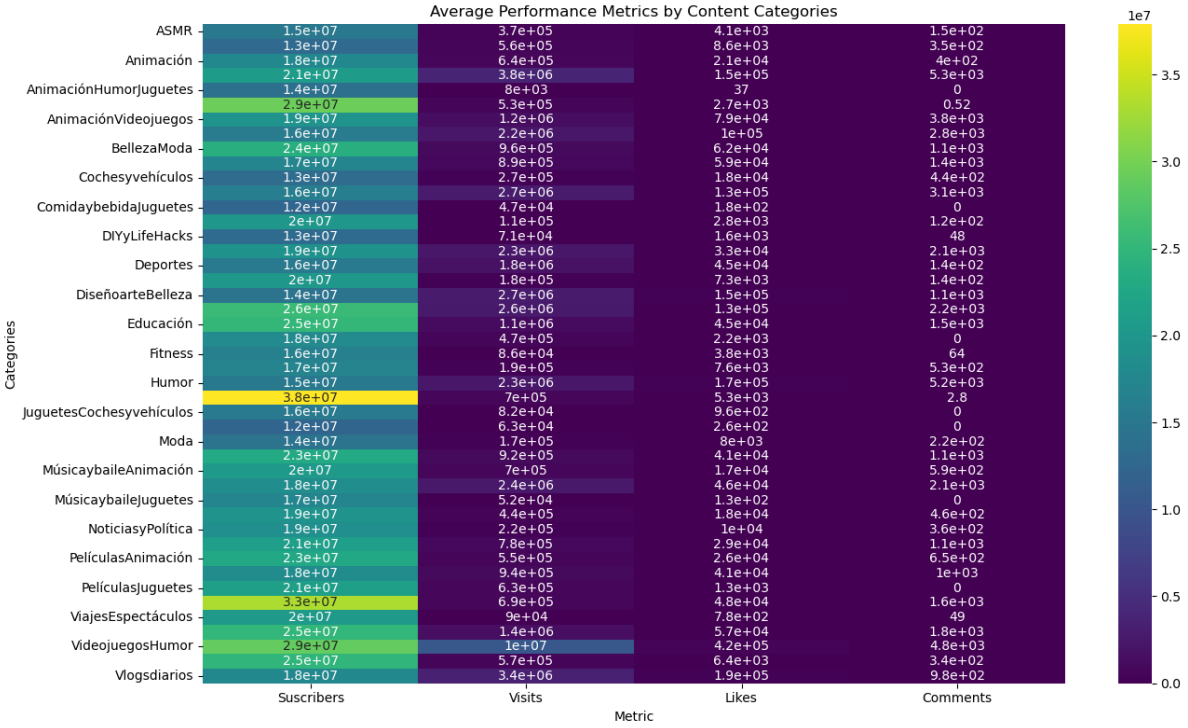
```
Average Suscribers: 21894400.00
Average Visits: 1209446.31
Average Likes: 53632.59
Average Comments: 1288.77
```



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 [30]:
```
def identify_high_performing_categories(data):
    # Identify categories with the highest average performance metrics
    avg_metrics_by_category = data.groupby('Categories').mean()

    # Visualize the highest performing categories
    plt.figure(figsize=(14, 8))
    sns.heatmap(avg_metrics_by_category[['Suscribers', 'Visits', 'Likes', 'Comments
    plt.title('Average Performance Metrics by Content Categories')
    plt.xlabel('Metric')
    plt.ylabel('Categories')
    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    identify_high_performing_categories(df)
```

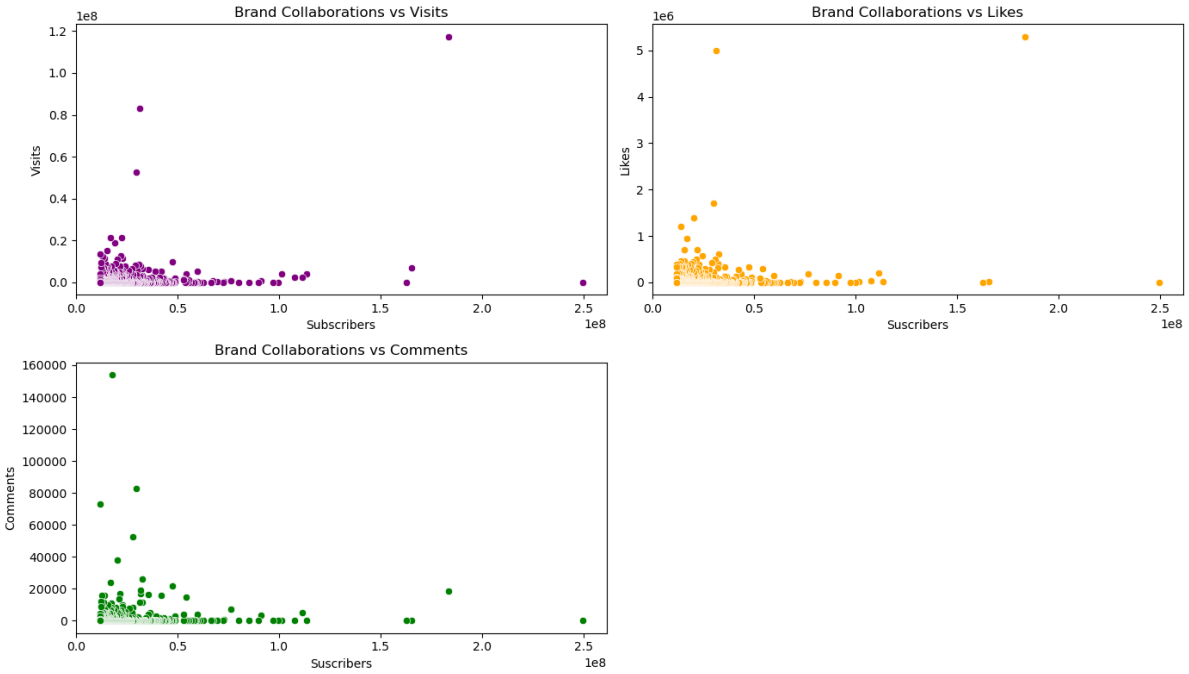


6. Brands and Collaborations:

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

In [31]:

```
def analyze_brand_collaborations(data):
    # Analyze whether streamers with high performance metrics receive more brand co
    plt.figure(figsize=(14, 8))
    # Scatter plot for Brand Collaborations vs Visits
    plt.subplot(2, 2, 1)
    sns.scatterplot(x='Suscribers', y='Visits', data=df, color='purple')
    plt.title('Brand Collaborations vs Visits')
    plt.xlabel('Subscribers')
    plt.ylabel('Visits')

    # Scatter plot for Brand Collaborations vs Likes
    plt.subplot(2, 2, 2)
    sns.scatterplot(x='Suscribers', y='Likes', data=df, color='orange')
    plt.title('Brand Collaborations vs Likes')
    plt.xlabel('Suscribers')
    plt.ylabel('Likes')

    # Scatter plot for Brand Collaborations vs Comments
    plt.subplot(2, 2, 3)
    sns.scatterplot(x='Suscribers', y='Comments', data=df, color='green')
    plt.title('Brand Collaborations vs Comments')
    plt.xlabel('Suscribers')
    plt.ylabel('Comments')

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    analyze_brand_collaborations(df)
```



In [ ]:
```
def analyze_marketing_campaigns(data):
    # Analyze whether streamers with high performance metrics receive more marketir
    plt.figure(figsize=(14, 8))

    # Scatter plot for Suscribers vs Visits
    plt.subplot(2, 2, 1)
    sns.scatterplot(x='Suscribers', y='Visits', data=df, color='blue')
    plt.title('Marketing Campaigns vs Visits')
    plt.xlabel('Subscribers')
    plt.ylabel('Visits')

    # Scatter plot for Visits vs Likes
    plt.subplot(2, 2, 2)
    sns.scatterplot(x='Suscribers', y='Likes', data=df, color='green')
    plt.title('Marketing Campaigns vs Likes')
    plt.xlabel('Subscribers')
    plt.ylabel('Likes')

    # Scatter plot for MarketingCampaigns vs Comments
    plt.subplot(2, 2, 3)
    sns.scatterplot(x='Suscribers', y='Comments', data=df, color='orange')
    plt.title('Marketing Campaigns vs Comments')
    plt.xlabel('Subscribers')
    plt.ylabel('Comments')

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    analyze_marketing_campaigns(df)
```

7. Benchmarking:

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

In [ ]:
```
def benchmark_top_performers(data):
    # Calculate average values for each metric
    avg_suscribers = data['Suscribers'].mean()
    avg_visits = data['Visits'].mean()
    avg_likes = data['Likes'].mean()
    avg_comments = data['Comments'].mean()

    # Identify streamers with above-average performance
    top_performers = data[
        (data['Suscribers'] > avg_suscribers) &
        (data['Visits'] > avg_visits) &
        (data['Likes'] > avg_likes) &
        (data['Comments'] > avg_comments)
    ]

    return top_performers

if __name__ == "__main__":
    # Call the function with your DataFrame 'df'
    top_performers = benchmark_top_performers(df)

    # Display the top-performing content creators
    print("Top-Performing Content Creators:")
    print(top_performers[['Username', 'Suscribers', 'Visits', 'Likes', 'Comments']]
```

8. Content Recommendations:

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

In [ ]:
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics.pairwise import cosine_similarity
```

In [ ]:
```
def normalize_metrics(data):
    # Normalize performance metrics using Min-Max scaling
    scaler = MinMaxScaler()
    metrics_scaled = scaler.fit_transform(data[['Suscribers', 'Visits', 'Likes', 'C
    data[['Suscribers', 'Visits', 'Likes', 'Comments']] = metrics_scaled
    return data

def content_recommendations(username, data):
    # Normalize metrics
    normalized_data = normalize_metrics(data)

    # Check if the user exists in the dataset
    if username not in data['Username'].values:
        print(f"User {username} not found in the dataset.")
        return None

    # Extract the categories subscribed by the user
    user_categories = data.loc[data['Username'] == username, 'Categories'].values[0

    # Create a user profile based on the average metrics of the user's subscribed c
    user_profile = normalized_data[normalized_data['Categories'].str.split(',').app

    # Calculate cosine similarity between user profile and streamers' metrics
    similarity_scores = cosine_similarity([user_profile[['Suscribers', 'Visits', 'L

    # Add similarity scores to the DataFrame
    data['SimilarityScore'] = similarity_scores[0]
```

```
    # Recommend top N streamers with the highest similarity scores
    top_recommendations = data.sort_values(by='SimilarityScore', ascending=False).h

    return top_recommendations[['Username', 'Categories', 'Suscribers', 'Visits', '

if __name__ == "__main__":
    # Assume 'user1' is the username for which we want to provide recommendations
    user_recommendations = content_recommendations('tseries', df)

    # Display the content recommendations for the user
    print("Content Recommendations for tseries:")
    print(user_recommendations)
```

In [ ]: