DATA ANALYSIS ON TOP YOUTUBERS

Importing Dependencies

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

Importing the Dataset

```
youtube_data = pd.read_csv('/content/youtubers_df.csv', encoding = 'ISO-8859-1')
```

```
youtube_data.shape
```

```
(1000, 9)
```

```
youtube_data.head()
```

	Rank	Username	Categories	Suscribers	Country	Visits	Likes	Comments	
0	1	tseries	MÃºsica y baile	249500000.0	India	86200.0	2700.0	78.0	http://youtube.com/channel/UCq-Fj5jknLsUf-M
1	2	MrBeast	Videojuegos, Humor	183500000.0	Estados Unidos	117400000.0	5300000.0	18500.0	http://youtube.com/channel/UCX6OQ3DkcsbYNE6
2	3	CoComelon	EducaciÃ³n	165500000.0	Unknown	7000000.0	24700.0	0.0	http://youtube.com/channel/UCbCmjCuTUZos6I
3	4	SETIndia	NaN	162600000.0	India	15600.0	166.0	9.0	http://youtube.com/channel/UCpEhnqL0y41EpW2

DATA EXPLORATION AND CLEANING

```
youtube_data.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
```

```
youtube_data.columns
```

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

```
youtube_data = youtube_data.rename(columns = {'Suscribers': 'Subscribers'})
```

```
youtube_data.columns
```

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

```
youtube_data.isnull().sum()
```

```
Rank             0
Username         0
```

```
Categories     306
Subscribers      0
Country          0
Visits           0
Likes            0
Comments         0
Links            0
dtype: int64
```

```
youtube_data.isnull().sum()
```

```
Rank             0
Username         0
Categories     306
Subscribers      0
Country          0
Visits           0
Likes            0
Comments         0
Links            0
dtype: int64
```

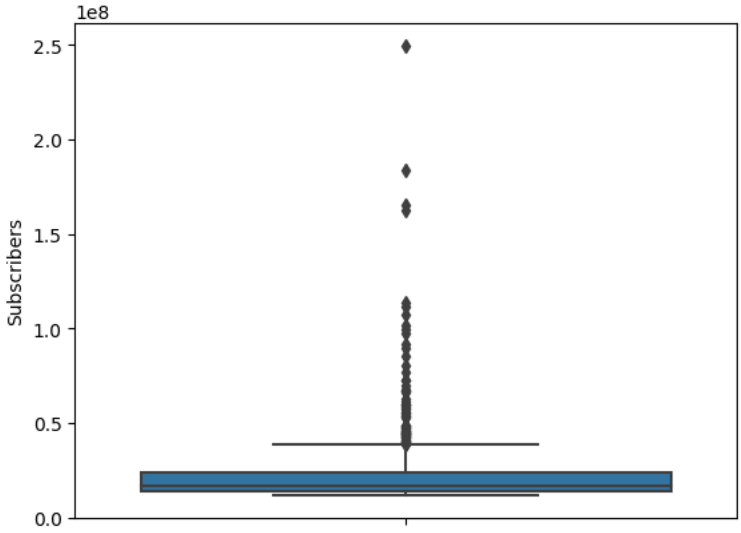
```
youtube_data['Categories'].fillna('Unknown', inplace = True)
```

```
youtube_data.isnull().sum()
```

```
Rank           0
Username       0
Categories     0
Subscribers    0
Country        0
Visits         0
Likes          0
Comments       0
Links          0
dtype: int64
```

CHECKING THE OUTLIERS

```
sb.boxplot(y = youtube_data['Subscribers'])
plt.show()
```



REMOVING THE OUTLIERS

Removing the outliers for column

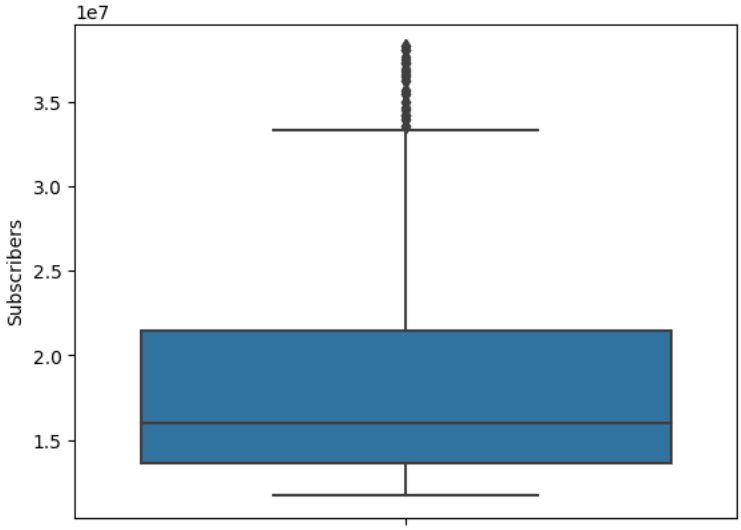
```
column1 = 'Subscribers'
# Calculating the interquartile range (IQR)
q1 = youtube_data[column1].quantile(0.25)
q3 = youtube_data[column1].quantile(0.75)
IQR = q3 - q1
IQR
```

```
9900000.0
```

```
# Defining the lower bound and upper bound for the outliers
lower_bound = q1 - 1.5 * IQR
upper_bound = q3 + 1.5 * IQR
```

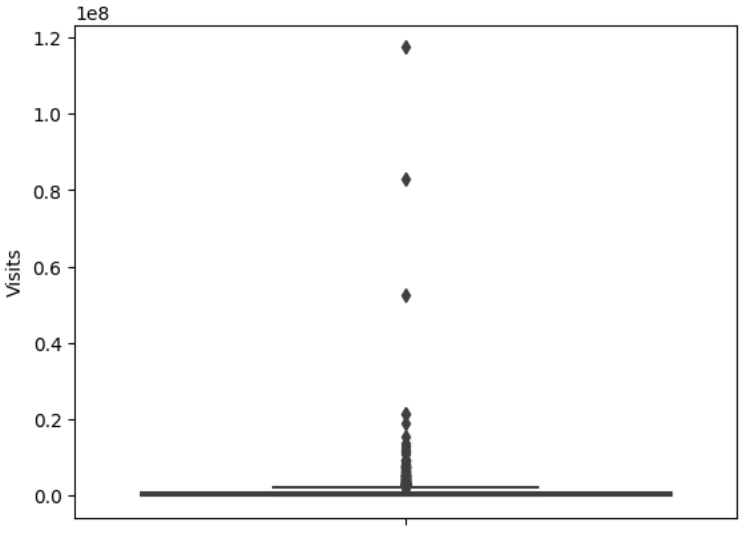
```
# Filtering the DataFrame to remove outliers
youtube_data_outliers = youtube_data[(youtube_data[column1] >= lower_bound) & (youtube_data[column1] <= upper_bound)]
```

```
# Boxplot without outliers
sb.boxplot(y = youtube_data_outliers[column1])
plt.show()
```




```
sb.boxplot(y = youtube_data['Visits'])
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Removing the outliers for Visits

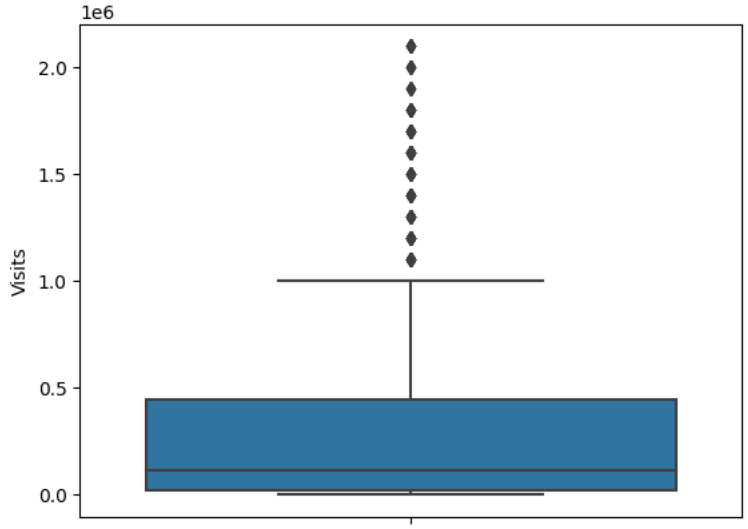
```
column2 = "Visits"
q1 = youtube_data[column2].quantile(0.25)
q3 = youtube_data[column2].quantile(0.75)
IQR_visits = q3 - q1
IQR_visits
```

```
833500.0
```

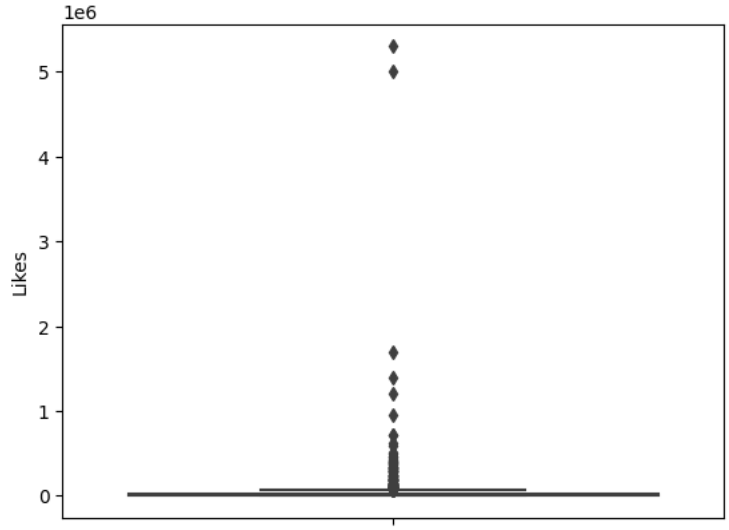
```
lower_bound = q1 - 1.5 * IQR_visits
upper_bound = q3 + 1.5 * IQR_visits
```

```
youtube_data_outliers = youtube_data[(youtube_data[column2] >= lower_bound) & (youtube_data[column2] <= upper_bound)]
```

```
sb.boxplot(y = youtube_data_outliers[column2])
plt.show()
```




```
sb.boxplot(y = youtube_data['Likes'])
plt.show()
```



Removing the outliers for Likes

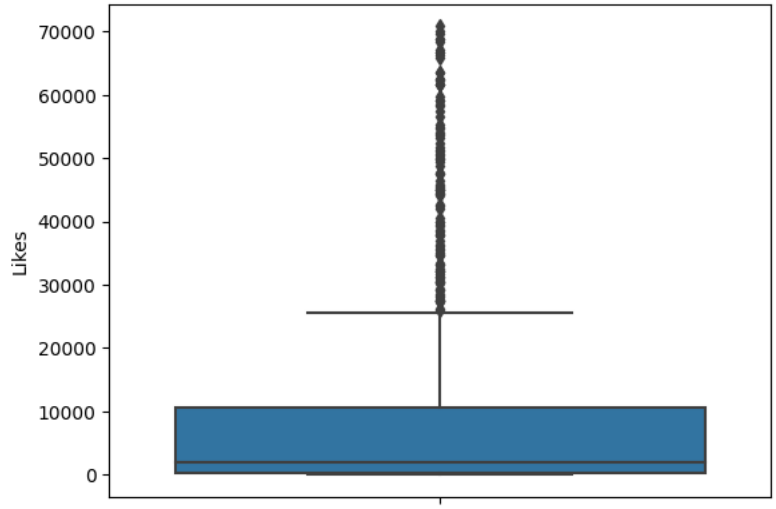
```
column3 = 'Likes'
q1 = youtube_data[column3].quantile(0.25)
q3 = youtube_data[column3].quantile(0.75)
IQR_likes = q3 - q1
IQR_likes
```

```
28178.25
```

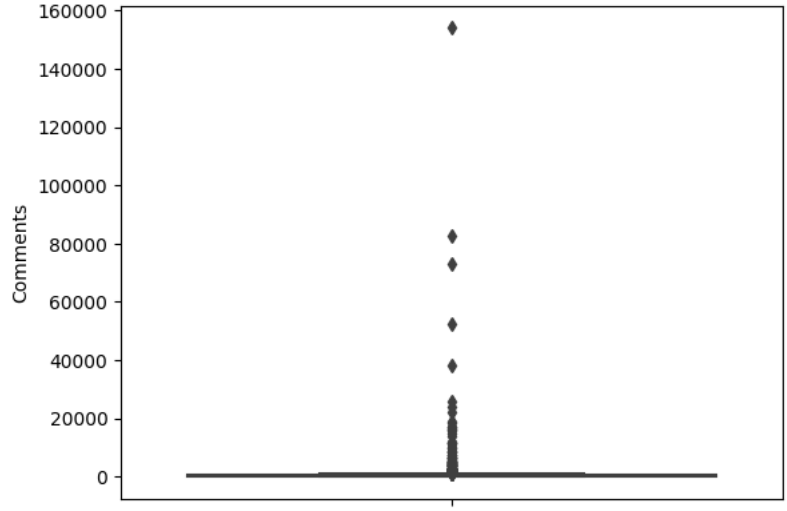
```
lower_bound = q1 - 1.5 * IQR_likes
upper_bound = q3 + 1.5 * IQR_likes
```

```
youtube_data_outliers = youtube_data[(youtube_data[column3] >= lower_bound) & (youtube_data[column3] <= upper_bound)]
```

```
sb.boxplot(y = youtube_data_outliers[column3])
plt.show()
```




```
sb.boxplot(y = youtube_data['Comments'])
plt.show()
```



Removing the outliers for Comments

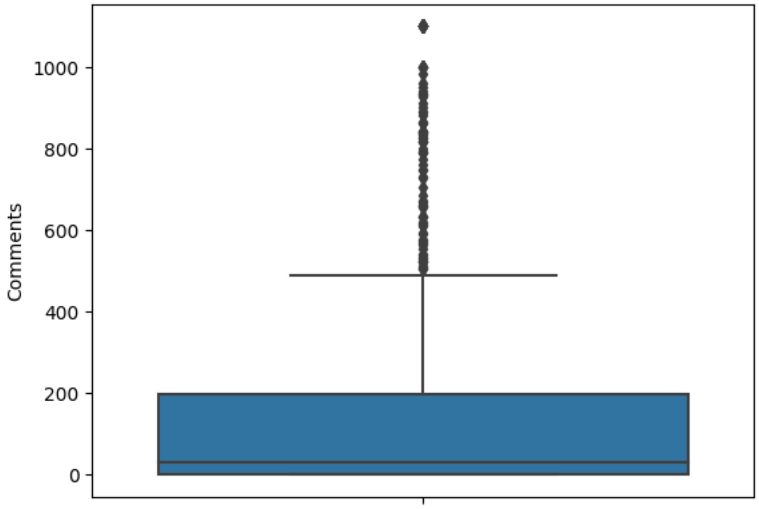
```
column4 = 'Comments'
q1 = youtube_data[column4].quantile(0.25)
q3 = youtube_data[column4].quantile(0.75)
IQR_comments = q3 - q1
IQR_comments
```

```
470.0
```

```
lower_bound = q1 - 1.5 * IQR_comments
upper_bound = q3 + 1.5 * IQR_comments
```

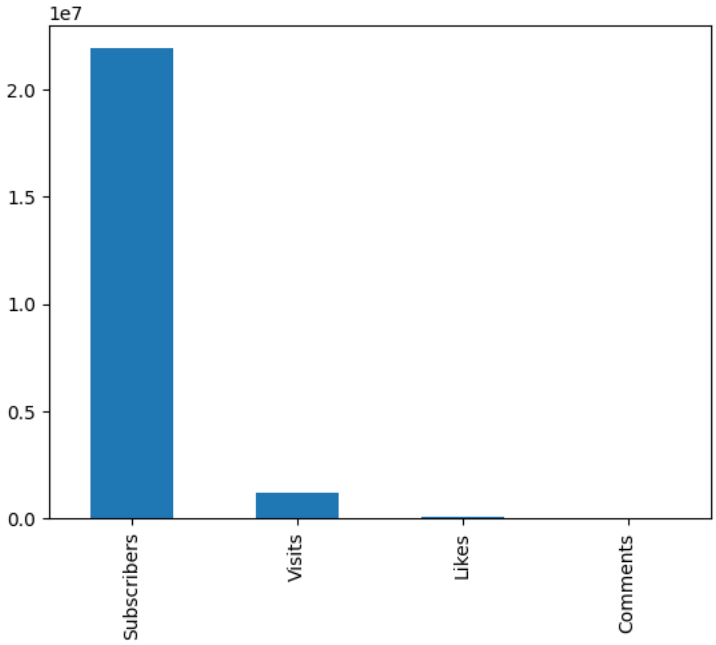
```
youtube_data_outliers = youtube_data[(youtube_data[column4] >= lower_bound) & (youtube_data[column4] <= upper_bound)]
```

```
sb.boxplot(y = youtube_data_outliers[column4])
plt.show()
```



PERFORMANCE METRICS

```
average_metrics = youtube_data[['Subscribers', 'Visits', 'Likes', 'Comments']].mean()
average_metrics.plot(kind = 'bar')
plt.show()
```



CONTENT CATEGORIES

```
category_distribution = youtube_data['Categories'].value_counts()
print(category_distribution)
```

```
Unknown                              306
MÃºsica y baile                      160
PelÃculas, AnimaciÃ³n                61
MÃºsica y baile, PelÃculas           41
Vlogs diarios                         37
Noticias y PolÃtica                 36
PelÃculas, Humor                    34
AnimaciÃ³n, Videojuegos               34
AnimaciÃ³n, Juguetes                  29
AnimaciÃ³n, Humor                     27
PelÃculas                           24
EducaciÃ³n                            24
AnimaciÃ³n                            22
Videojuegos                           19
Videojuegos, Humor                    17
MÃºsica y baile, AnimaciÃ³n            16
Ciencia y tecnologÃa                14
Comida y bebida                       12
```

```
Humor                                10
Juguetes                             10
PelÃculas, Juguetes                  9
PelÃculas, Videojuegos               8
Deportes                              8
MÃºsica y baile, Humor                6
Juguetes, Coches y vehÃculos         4
DIY y Life Hacks                      3
Fitness, Salud y autoayuda            3
Videojuegos, Juguetes                 3
Animales y mascotas                   2
Moda                                  2
Coches y vehÃculos                   2
EducaciÃ³n, Juguetes                  2
Fitness                               2
Comida y bebida, Juguetes             1
ASMR, Comida y bebida                 1
AnimaciÃ³n, Humor, Juguetes           1
DiseÃ±o/arte, Belleza                 1
Belleza, Moda                         1
ASMR                                  1
MÃºsica y baile, Juguetes             1
DiseÃ±o/arte, DIY y Life Hacks        1
DIY y Life Hacks, Juguetes            1
DiseÃ±o/arte                          1
Comida y bebida, Salud y autoayuda    1
Viajes, EspectÃ¡culos                 1
Juguetes, DIY y Life Hacks            1
Name: Categories, dtype: int64
```

PERFORMING KMEANS CLUSTERING

To ensure that the values have a mean 0 and standard deviation 1

```
scaler = StandardScaler()
scaled_metric = scaler.fit_transform(youtube_data[['Subscribers', 'Visits', 'Likes', 'Comments']])
kmeans = KMeans(n_clusters = 2)
youtube_data['cluster'] = kmeans.fit_predict(scaled_metric)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
  warnings.warn(
```

Top performing content creator

```
performer = youtube_data[youtube_data['cluster'] == 1]
performer
```

	Rank	Username	Categories	Subscribers	Country	Visits	Likes	Comment
1	2	MrBeast	Videojuegos, Humor	183500000.0	Estados Unidos	117400000.0	5300000.0	18500

DATAFRAME SHAPE

Before removing outliers

```
youtube_data.shape
```

```
(1000, 10)
```

After removing outliers

```
youtube_data_outliers.shape
```

```
(849, 9)
```

CONCLUSION

- The analysis provided valuable insights into the top YouTube streamers.
- Missing values were handled, outliers were removed, and K-Means clustering was applied to identify potential high-performing creators.

- The overall goal was to refine the dataset and improve the accuracy of subsequent analyses.

In summary, the analysis helps in understanding the characteristics of top YouTubers, identifying outliers, and clustering them based on performance metrics. The results can be further utilized for strategic decision-making in the context of content creation and audience engagement.