YouTube_Analytics_2023

September 21, 2023

[ ]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')
from matplotlib.ticker import FuncFormatter
from matplotlib import ticker
```

[ ]:
```python
data = pd.read_csv("DATA/Global_YouTube_Statistics.csv", encoding="ISO-8859-1")

data.head()
```

[ ]:
```
   rank                    Youtuber  subscribers   video views  \
0     1                    T-Series    245000000  2.280000e+11
1     2              YouTube Movies    170000000  0.000000e+00
2     3                     MrBeast    166000000  2.836884e+10
3     4  Cocomelon - Nursery Rhymes    162000000  1.640000e+11
4     5                   SET India    159000000  1.480000e+11

           category                       Title  uploads        Country  \
0             Music                    T-Series    20082          India
1  Film & Animation               youtubemovies        1  United States
2     Entertainment                     MrBeast      741  United States
3         Education  Cocomelon - Nursery Rhymes      966  United States
4             Shows                   SET India   116536          India

  Abbreviation   channel_type  …  subscribers_for_last_30_days  \
0           IN          Music  …                     2000000.0
1           US          Games  …                           NaN
2           US  Entertainment  …                     8000000.0
3           US      Education  …                     1000000.0
4           IN  Entertainment  …                     1000000.0

   created_year  created_month  created_date  \
0        2006.0            Mar          13.0
```

```
1       2006.0            Mar           5.0
2       2012.0            Feb          20.0
3       2006.0            Sep           1.0
4       2006.0            Sep          20.0

   Gross tertiary education enrollment (%)    Population  Unemployment rate  \
0                                     28.1  1.366418e+09               5.36
1                                     88.2  3.282395e+08              14.70
2                                     88.2  3.282395e+08              14.70
3                                     88.2  3.282395e+08              14.70
4                                     28.1  1.366418e+09               5.36

   Urban_population   Latitude  Longitude
0       471031528.0  20.593684  78.962880
1       270663028.0  37.090240 -95.712891
2       270663028.0  37.090240 -95.712891
3       270663028.0  37.090240 -95.712891
4       471031528.0  20.593684  78.962880

[5 rows x 28 columns]
```

[ ]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 995 entries, 0 to 994
Data columns (total 28 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   rank                                     995 non-null    int64
 1   Youtuber                                 995 non-null    object
 2   subscribers                              995 non-null    int64
 3   video views                              995 non-null    float64
 4   category                                 949 non-null    object
 5   Title                                    995 non-null    object
 6   uploads                                  995 non-null    int64
 7   Country                                  873 non-null    object
 8   Abbreviation                             873 non-null    object
 9   channel_type                             965 non-null    object
 10  video_views_rank                         994 non-null    float64
 11  country_rank                             879 non-null    float64
 12  channel_type_rank                        962 non-null    float64
 13  video_views_for_the_last_30_days         939 non-null    float64
 14  lowest_monthly_earnings                  995 non-null    float64
 15  highest_monthly_earnings                 995 non-null    float64
 16  lowest_yearly_earnings                   995 non-null    float64
 17  highest_yearly_earnings                  995 non-null    float64
 18  subscribers_for_last_30_days             658 non-null    float64
 19  created_year                             990 non-null    float64
```

```
 20  created_month                            990 non-null    object
 21  created_date                             990 non-null    float64
 22  Gross tertiary education enrollment (%)  872 non-null    float64
 23  Population                               872 non-null    float64
 24  Unemployment rate                        872 non-null    float64
 25  Urban_population                         872 non-null    float64
 26  Latitude                                 872 non-null    float64
 27  Longitude                                872 non-null    float64
dtypes: float64(18), int64(3), object(7)
memory usage: 217.8+ KB
```

[ ]:
```
data.columns
```

[ ]:
```
Index(['rank', 'Youtuber', 'subscribers', 'video views', 'category', 'Title',
       'uploads', 'Country', 'Abbreviation', 'channel_type',
       'video_views_rank', 'country_rank', 'channel_type_rank',
       'video_views_for_the_last_30_days', 'lowest_monthly_earnings',
       'highest_monthly_earnings', 'lowest_yearly_earnings',
       'highest_yearly_earnings', 'subscribers_for_last_30_days',
       'created_year', 'created_month', 'created_date',
       'Gross tertiary education enrollment (%)', 'Population',
       'Unemployment rate', 'Urban_population', 'Latitude', 'Longitude'],
      dtype='object')
```

[ ]:
```
data.tail()
```

[ ]:
```
     rank                  Youtuber  subscribers   video views  \
990   991            Natan por Aï¿½     12300000  9.029610e+09
991   992  Free Fire India Official     12300000  1.674410e+09
992   993                     Panda     12300000  2.214684e+09
993   994               RobTopGames     12300000  3.741235e+08
994   995             Make Joke Of     12300000  2.129774e+09

           category                     Title  uploads         Country  \
990          Sports            Natan por Aï¿½     1200          Brazil
991  People & Blogs  Free Fire India Official     1500           India
992             NaN               HybridPanda     2452  United Kingdom
993          Gaming               RobTopGames       39          Sweden
994          Comedy             Make Joke Of       62           India

    Abbreviation   channel_type  …  subscribers_for_last_30_days  \
990           BR  Entertainment  …                      700000.0
991           IN          Games  …                      300000.0
992           GB          Games  …                        1000.0
993           SE          Games  …                      100000.0
994           IN         Comedy  …                      100000.0

     created_year  created_month  created_date  \
```

```
990        2017.0           Feb          12.0
991        2018.0           Sep          14.0
992        2006.0           Sep          11.0
993        2012.0           May           9.0
994        2017.0           Aug           1.0

     Gross tertiary education enrollment (%)    Population  Unemployment rate  \
990                                     51.3  2.125594e+08              12.08
991                                     28.1  1.366418e+09               5.36
992                                     60.0  6.683440e+07               3.85
993                                     67.0  1.028545e+07               6.48
994                                     28.1  1.366418e+09               5.36

     Urban_population   Latitude  Longitude
990       183241641.0 -14.235004 -51.925280
991       471031528.0  20.593684  78.962880
992        55908316.0  55.378051  -3.435973
993         9021165.0  60.128161  18.643501
994       471031528.0  20.593684  78.962880

[5 rows x 28 columns]
```

[ ]:
```
data.describe()
```

[ ]:
```
            rank   subscribers   video views         uploads  video_views_rank  \
count  995.00000  9.950000e+02  9.950000e+02      995.000000      9.940000e+02
mean   498.00000  2.298241e+07  1.103954e+10     9187.125628      5.542489e+05
std    287.37606  1.752611e+07  1.411084e+10    34151.352254      1.362782e+06
min      1.00000  1.230000e+07  0.000000e+00        0.000000      1.000000e+00
25%    249.50000  1.450000e+07  4.288145e+09      194.500000      3.230000e+02
50%    498.00000  1.770000e+07  7.760820e+09      729.000000      9.155000e+02
75%    746.50000  2.460000e+07  1.355470e+10     2667.500000      3.584500e+03
max    995.00000  2.450000e+08  2.280000e+11   301308.000000      4.057944e+06

       country_rank  channel_type_rank  video_views_for_the_last_30_days  \
count    879.000000         962.000000                      9.390000e+02
mean     386.053470         745.719335                      1.756103e+08
std     1232.244746        1944.386561                      4.163782e+08
min        1.000000           1.000000                      1.000000e+00
25%       11.000000          27.000000                      2.013750e+07
50%       51.000000          65.500000                      6.408500e+07
75%      123.000000         139.750000                      1.688265e+08
max     7741.000000        7741.000000                      6.589000e+09

       lowest_monthly_earnings  highest_monthly_earnings  …  \
count               995.000000              9.950000e+02  …
mean              36886.148281              5.898078e+05  …
```

```
std           71858.724092       1.148622e+06  …
min               0.000000       0.000000e+00  …
25%            2700.000000       4.350000e+04  …
50%           13300.000000       2.127000e+05  …
75%           37900.000000       6.068000e+05  …
max          850900.000000       1.360000e+07  …

       highest_yearly_earnings  subscribers_for_last_30_days  created_year  \
count             9.950000e+02                  6.580000e+02    990.000000
mean              7.081814e+06                  3.490791e+05   2012.630303
std               1.379704e+07                  6.143554e+05      4.512503
min               0.000000e+00                  1.000000e+00   1970.000000
25%               5.217500e+05                  1.000000e+05   2009.000000
50%               2.600000e+06                  2.000000e+05   2013.000000
75%               7.300000e+06                  4.000000e+05   2016.000000
max               1.634000e+08                  8.000000e+06   2022.000000

       created_date  Gross tertiary education enrollment (%)    Population  \
count    990.000000                               872.000000  8.720000e+02
mean      15.746465                                63.627752  4.303873e+08
std        8.777520                                26.106893  4.727947e+08
min        1.000000                                 7.600000  2.025060e+05
25%        8.000000                                36.300000  8.335541e+07
50%       16.000000                                68.000000  3.282395e+08
75%       23.000000                                88.200000  3.282395e+08
max       31.000000                               113.100000  1.397715e+09

       Unemployment rate  Urban_population    Latitude   Longitude
count         872.000000      8.720000e+02  872.000000  872.000000
mean            9.279278      2.242150e+08   26.632783  -14.128146
std             4.888354      1.546874e+08   20.560533   84.760809
min             0.750000      3.558800e+04  -38.416097 -172.104629
25%             5.270000      5.590832e+07   20.593684  -95.712891
50%             9.365000      2.706630e+08   37.090240  -51.925280
75%            14.700000      2.706630e+08   37.090240   78.962880
max            14.720000      8.429340e+08   61.924110  138.252924

[8 rows x 21 columns]
```

[ ]:
```
data.shape
```

[ ]:
```
(995, 28)
```

[ ]:
```
data.isnull().sum()
```

[ ]:
```
rank                                        0
Youtuber                                    0
```

```
subscribers                                 0
video views                                 0
category                                   46
Title                                       0
uploads                                     0
Country                                   122
Abbreviation                              122
channel_type                               30
video_views_rank                            1
country_rank                              116
channel_type_rank                          33
video_views_for_the_last_30_days           56
lowest_monthly_earnings                     0
highest_monthly_earnings                    0
lowest_yearly_earnings                      0
highest_yearly_earnings                     0
subscribers_for_last_30_days              337
created_year                                5
created_month                               5
created_date                                5
Gross tertiary education enrollment (%)   123
Population                                123
Unemployment rate                         123
Urban_population                          123
Latitude                                  123
Longitude                                 123
dtype: int64
```

[ ]:
```
data['category'].unique()
```

[ ]:
```
array(['Music', 'Film & Animation', 'Entertainment', 'Education', 'Shows',
       nan, 'People & Blogs', 'Gaming', 'Sports', 'Howto & Style',
       'News & Politics', 'Comedy', 'Trailers', 'Nonprofits & Activism',
       'Science & Technology', 'Movies', 'Pets & Animals',
       'Autos & Vehicles', 'Travel & Events'], dtype=object)
```

[ ]:
```
data['category'].nunique()
```

[ ]:
```
18
```

[ ]:
```
data.duplicated().sum()
```

[ ]:
```
0
```

[ ]:
```
data = data.dropna()
```

[ ]:
```
data[data['category'].isnull()].head(5)
```

[ ]: Empty DataFrame
Columns: [rank, Youtuber, subscribers, video views, category, Title, uploads, Country, Abbreviation, channel_type, video_views_rank, country_rank, channel_type_rank, video_views_for_the_last_30_days, lowest_monthly_earnings, highest_monthly_earnings, lowest_yearly_earnings, highest_yearly_earnings, subscribers_for_last_30_days, created_year, created_month, created_date, Gross tertiary education enrollment (%), Population, Unemployment rate, Urban_population, Latitude, Longitude]
Index: []

[0 rows x 28 columns]

[ ]:
```python
dropped_features = [
    'Abbreviation', 'created_month', 'created_date', 'Latitude', 'Longitude'
]
data = data.drop(data[dropped_features], axis=1)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 554 entries, 0 to 994
Data columns (total 23 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   rank                                     554 non-null    int64
 1   Youtuber                                 554 non-null    object
 2   subscribers                              554 non-null    int64
 3   video views                              554 non-null    float64
 4   category                                 554 non-null    object
 5   Title                                    554 non-null    object
 6   uploads                                  554 non-null    int64
 7   Country                                  554 non-null    object
 8   channel_type                             554 non-null    object
 9   video_views_rank                         554 non-null    float64
 10  country_rank                             554 non-null    float64
 11  channel_type_rank                        554 non-null    float64
 12  video_views_for_the_last_30_days         554 non-null    float64
 13  lowest_monthly_earnings                  554 non-null    float64
 14  highest_monthly_earnings                 554 non-null    float64
 15  lowest_yearly_earnings                   554 non-null    float64
 16  highest_yearly_earnings                  554 non-null    float64
 17  subscribers_for_last_30_days             554 non-null    float64
 18  created_year                             554 non-null    float64
 19  Gross tertiary education enrollment (%)  554 non-null    float64
 20  Population                               554 non-null    float64
 21  Unemployment rate                        554 non-null    float64
 22  Urban_population                         554 non-null    float64
dtypes: float64(15), int64(3), object(5)
memory usage: 103.9+ KB
```

[ ]:
```
numerical_cols = ['subscribers', 'video views', 'uploads',
                  'lowest_monthly_earnings', 'highest_monthly_earnings',
                  'lowest_yearly_earnings', 'highest_yearly_earnings',
                  'video_views_for_the_last_30_days',␣
 ↪'subscribers_for_last_30_days']

correlation_matrix = data[numerical_cols].corr()

correlation_matrix
```

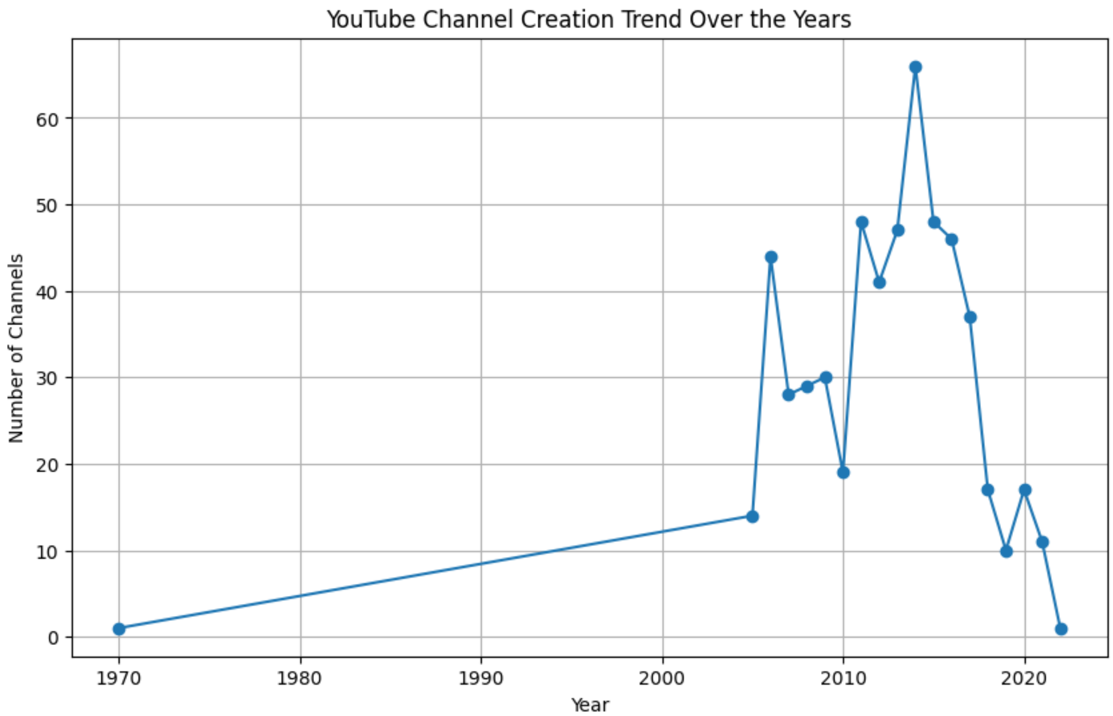
[ ]:
```
                                  subscribers  video views   uploads  \
subscribers                          1.000000     0.850776  0.074791
video views                          0.850776     1.000000  0.147477
uploads                              0.074791     0.147477  1.000000
lowest_monthly_earnings              0.534713     0.637991  0.155557
highest_monthly_earnings             0.534388     0.637376  0.156062
lowest_yearly_earnings               0.534883     0.638861  0.156639
highest_yearly_earnings              0.534721     0.638019  0.155540
video_views_for_the_last_30_days     0.321182     0.360970  0.079308
subscribers_for_last_30_days         0.361800     0.203693  0.009067

                                  lowest_monthly_earnings  \
subscribers                                      0.534713
video views                                      0.637991
uploads                                          0.155557
lowest_monthly_earnings                          1.000000
highest_monthly_earnings                         0.999937
lowest_yearly_earnings                           0.999914
highest_yearly_earnings                          0.999998
video_views_for_the_last_30_days                 0.585838
subscribers_for_last_30_days                     0.616795

                                  highest_monthly_earnings  \
subscribers                                       0.534388
video views                                       0.637376
uploads                                           0.156062
lowest_monthly_earnings                           0.999937
highest_monthly_earnings                          1.000000
lowest_yearly_earnings                            0.999832
highest_yearly_earnings                           0.999935
video_views_for_the_last_30_days                  0.585788
subscribers_for_last_30_days                      0.617150

                                  lowest_yearly_earnings  \
subscribers                                     0.534883
video views                                     0.638861
uploads                                         0.156639
```

```
lowest_monthly_earnings                        0.999914
highest_monthly_earnings                       0.999832
lowest_yearly_earnings                         1.000000
highest_yearly_earnings                        0.999912
video_views_for_the_last_30_days               0.585758
subscribers_for_last_30_days                   0.615874

                                  highest_yearly_earnings  \
subscribers                                      0.534721
video views                                      0.638019
uploads                                          0.155540
lowest_monthly_earnings                          0.999998
highest_monthly_earnings                         0.999935
lowest_yearly_earnings                           0.999912
highest_yearly_earnings                          1.000000
video_views_for_the_last_30_days                 0.585837
subscribers_for_last_30_days                     0.616816

                                  video_views_for_the_last_30_days  \
subscribers                                               0.321182
video views                                               0.360970
uploads                                                   0.079308
lowest_monthly_earnings                                   0.585838
highest_monthly_earnings                                  0.585788
lowest_yearly_earnings                                    0.585758
highest_yearly_earnings                                   0.585837
video_views_for_the_last_30_days                          1.000000
subscribers_for_last_30_days                              0.366320

                                  subscribers_for_last_30_days
subscribers                                           0.361800
video views                                           0.203693
uploads                                               0.009067
lowest_monthly_earnings                               0.616795
highest_monthly_earnings                              0.617150
lowest_yearly_earnings                                0.615874
highest_yearly_earnings                               0.616816
video_views_for_the_last_30_days                      0.366320
subscribers_for_last_30_days                          1.000000
```

This code calculates and visualizes the trend in YouTube channel creation over the years using the "created_year" column. Once you've completed this step, you can proceed to the next analysis. If you'd like to continue with the next step or have any questions, please feel free to ask.

[ ]:
```
yearly_trend = data['created_year'].value_counts().sort_index()
```

[ ]:
```python
# Visualize the YouTube channel creation trend over the years
plt.figure(figsize=(10, 6))
plt.plot(yearly_trend.index, yearly_trend.values, marker='o', linestyle='-')
plt.title('YouTube Channel Creation Trend Over the Years')
plt.xlabel('Year')
plt.ylabel('Number of Channels')
plt.grid(True)
plt.show()
```



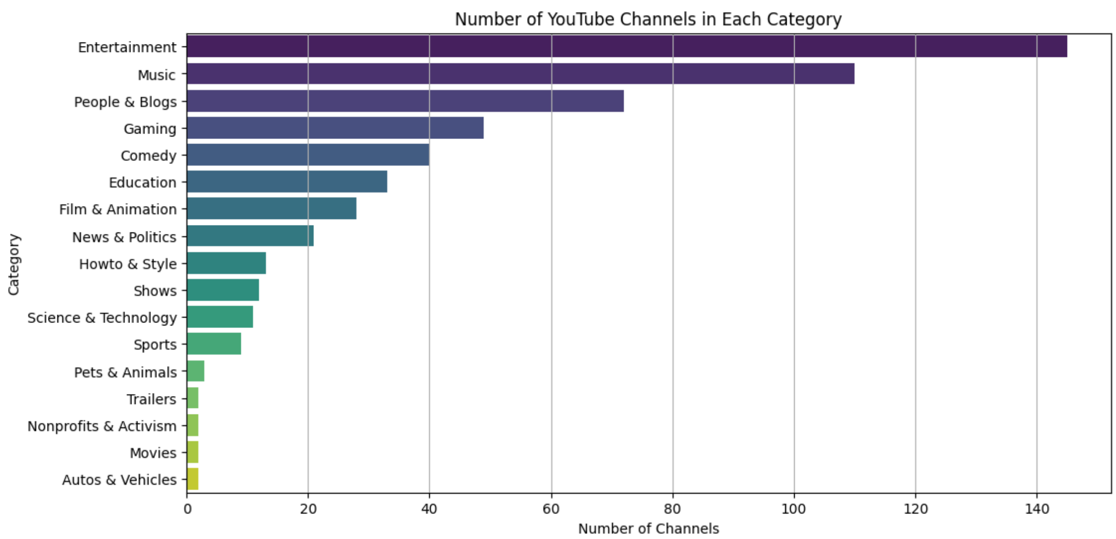
This code calculates and visualizes the number of YouTube channels in each category using a bar plot. It will help you understand which categories have the most channels. If you have any questions or would like to proceed with the next analysis step, please feel free to ask.

[ ]:
```python
# Calculate the number of channels in each category

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

[ ]:
```python
category_counts = category_counts.sort_values(ascending=False)
# Sort the categories by the number of channels in descending order
```

[ ]:
```python
# Visualize the number of channels in each category using a bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x=category_counts.values, y=category_counts.index,␣
↪palette="viridis")
```

```
plt.title('Number of YouTube Channels in Each Category')
plt.xlabel('Number of Channels')
plt.ylabel('Category')
plt.grid(axis='x')
plt.show()
```

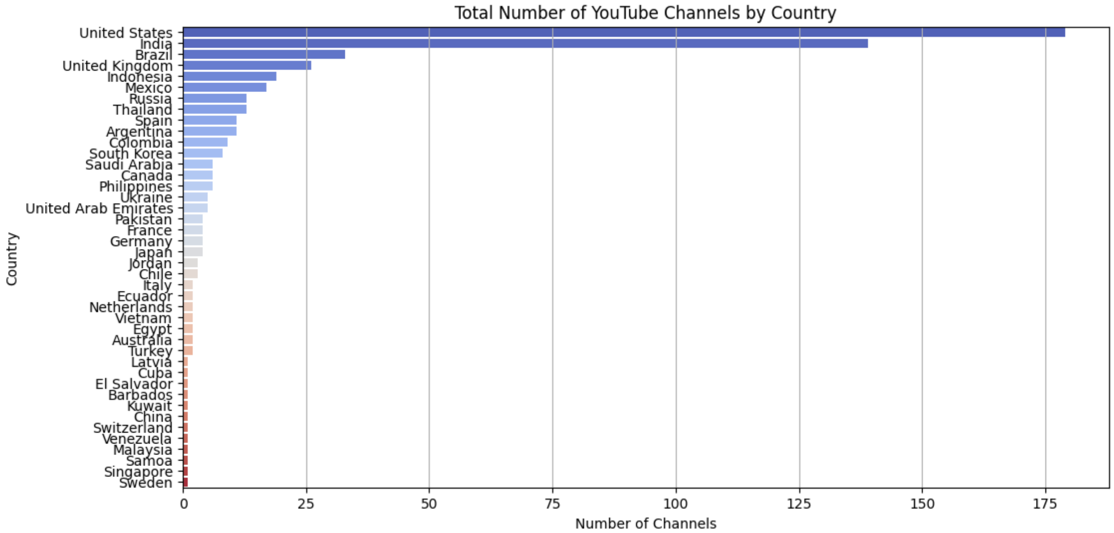


This code calculates and visualizes the total number of YouTube channels from each country using a bar plot. It will provide insights into which countries have the highest number of YouTube channels. If you have any questions or would like to proceed with the next analysis step, please feel free to ask.

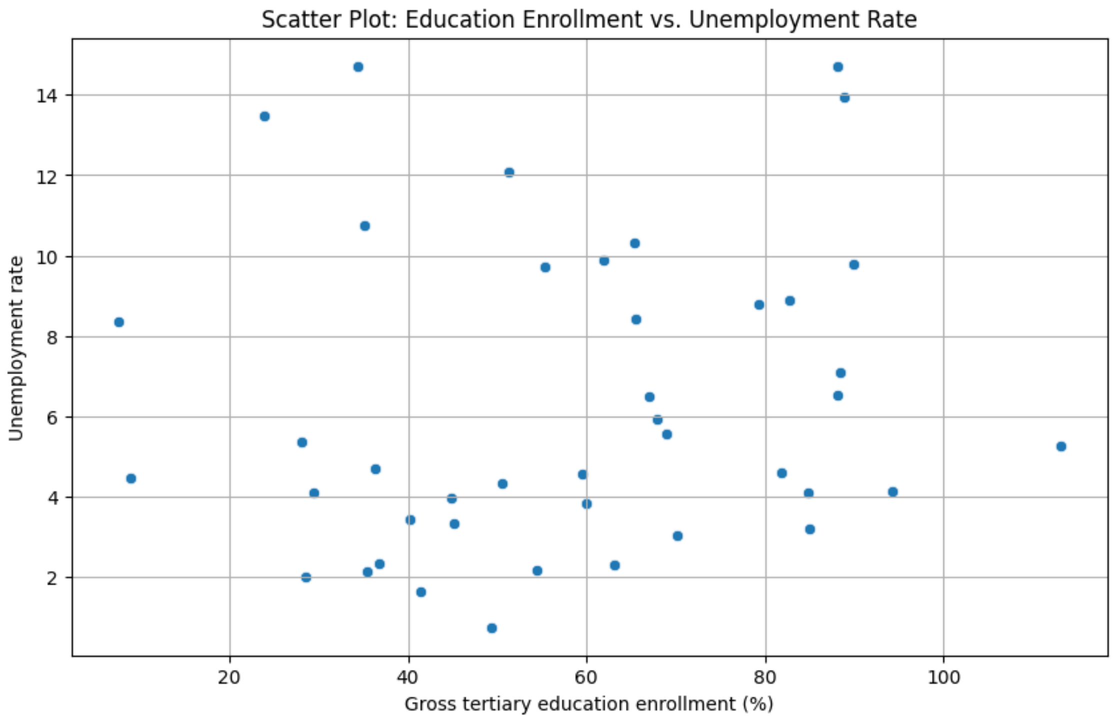
[ ]:
```
# Calculate the total number of channels from each country
country_counts = data['Country'].value_counts()
```

[ ]:
```
# Sort the countries by the total number of channels in descending order
country_counts = country_counts.sort_values(ascending=False)
```

[ ]:
```
# Visualize the total number of channels from each country using a bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x=country_counts.values, y=country_counts.index, palette="coolwarm")
plt.title('Total Number of YouTube Channels by Country')
plt.xlabel('Number of Channels')
plt.ylabel('Country')
plt.grid(axis='x')
plt.show()
```



[ ]:
```
# Scatter plot to analyze the relationship between "Gross tertiary education␣
↪enrollment (%)" and "Unemployment rate"
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Gross tertiary education enrollment (%)', y='Unemployment␣
↪rate', data=data)
plt.title('Scatter Plot: Education Enrollment vs. Unemployment Rate')
plt.xlabel('Gross tertiary education enrollment (%)')
plt.ylabel('Unemployment rate')
plt.grid(True)
plt.show()
```

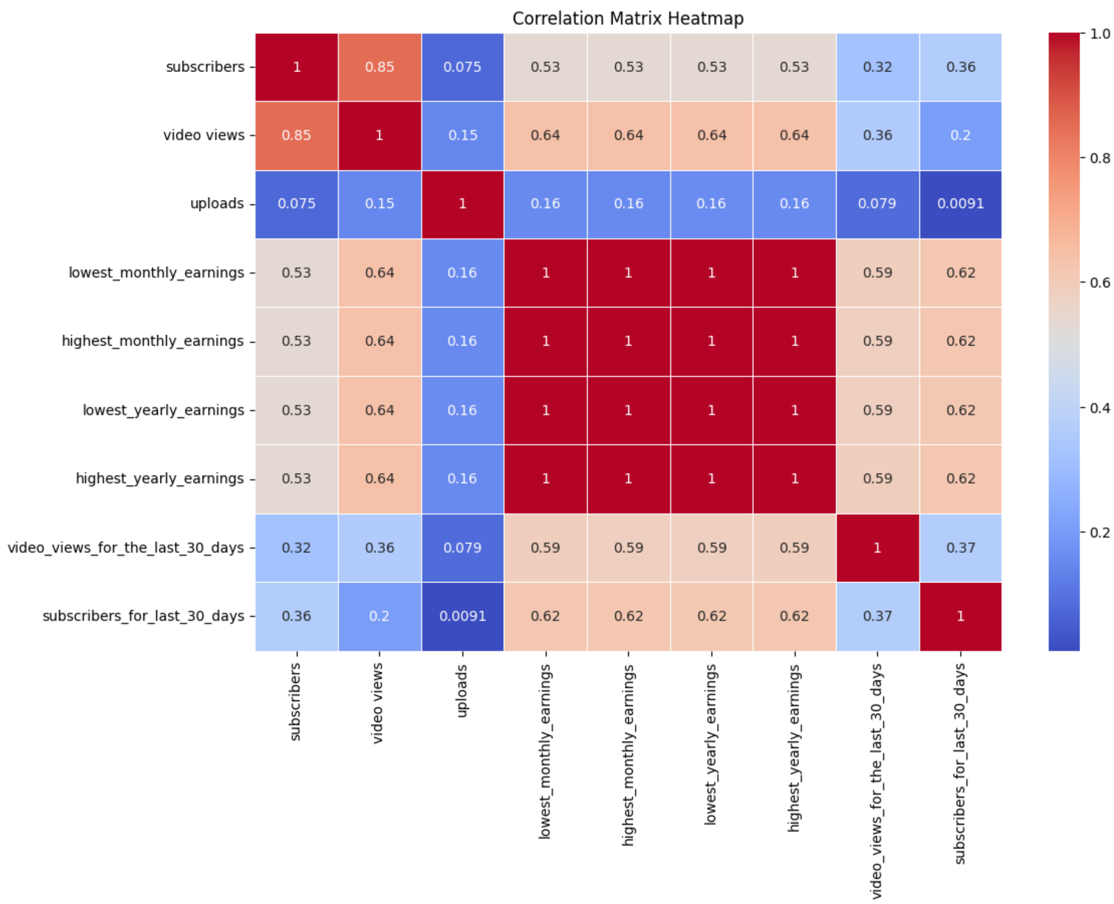


Analysis 2: Correlation Matrix Heatmap

```
[ ]: # Select numerical columns for correlation analysis
numerical_cols = ['subscribers', 'video views', 'uploads',
                  'lowest_monthly_earnings', 'highest_monthly_earnings',
                  'lowest_yearly_earnings', 'highest_yearly_earnings',
                  'video_views_for_the_last_30_days',␣
 ↪'subscribers_for_last_30_days']
```

```
[ ]: # Calculate the correlation matrix
correlation_matrix = data[numerical_cols].corr()
```

```
[ ]: # Create a heatmap to visualize the correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", linewidths=.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



Explanation: We create a heatmap to visualize the correlation matrix, with annotations to show the correlation coefficients. This heatmap provides insights into how strongly each numerical variable is related to others.

Regression Models:

Linear Regression: To predict the number of subscribers for YouTube channels.

[ ]:
```python
# Import necessary libraries

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

[ ]:
```python
# Select features and target variable
X = data[['video views', 'uploads']]
y = data['subscribers']
```

[ ]:
```python
# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[ ]:
```
# Create and train a Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

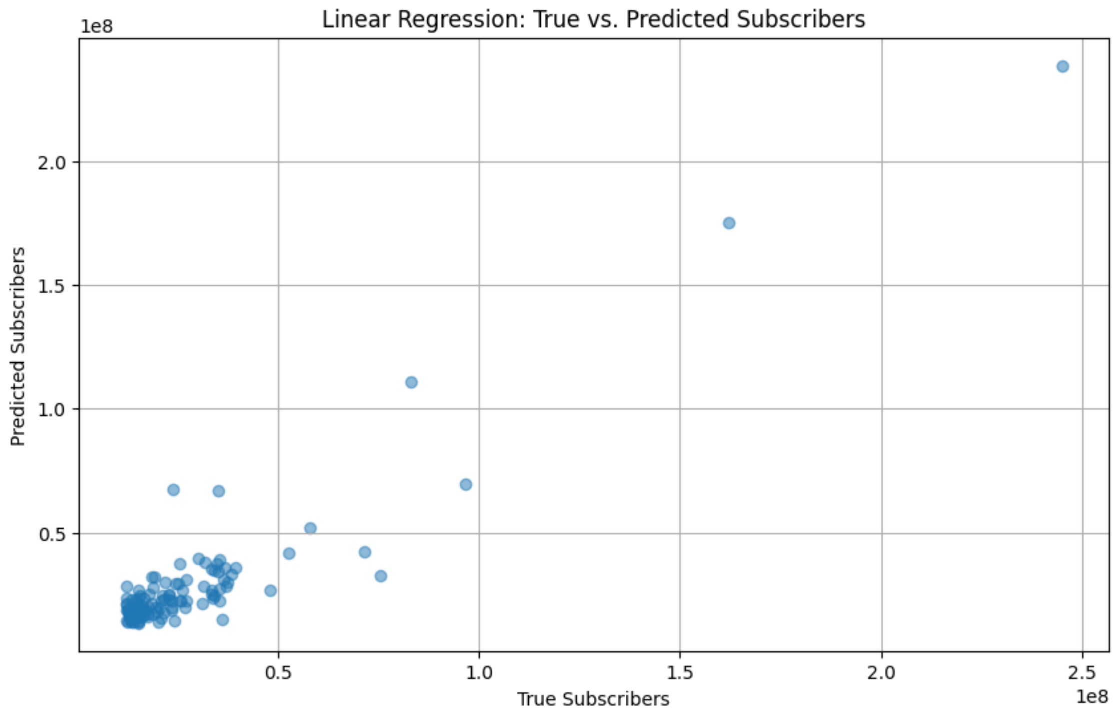
[ ]:
```
LinearRegression()
```

[ ]:
```
# Make predictions on the test set
y_pred = model.predict(X_test)
```

[ ]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 107379845764229.67
R-squared: 0.8670140324743885
```

[ ]:
```
# Visualize Linear Regression predictions
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.title('Linear Regression: True vs. Predicted Subscribers')
plt.xlabel('True Subscribers')
plt.ylabel('Predicted Subscribers')
plt.grid(True)
plt.show()
```



Non-Linear Regression (e.g., Ridge, Lasso, ElasticNet): To handle complex relationships in predicting subscriber counts

[ ]:
```
# Import necessary libraries
from sklearn.linear_model import Ridge, Lasso, ElasticNet
```

[ ]:
```
# Create and train Ridge Regression model
ridge_model = Ridge(alpha=1.0)
ridge_model.fit(X_train, y_train)
```

[ ]: Ridge()

[ ]:
```
# Create and train Lasso Regression model
lasso_model = Lasso(alpha=1.0)
lasso_model.fit(X_train, y_train)
```

[ ]: Lasso()

[ ]:
```
# Create and train ElasticNet Regression model
elasticnet_model = ElasticNet(alpha=1.0, l1_ratio=0.5)
elasticnet_model.fit(X_train, y_train)
```

[ ]: ElasticNet()

```
[ ]: # Make predictions using each model
ridge_predictions = ridge_model.predict(X_test)
lasso_predictions = lasso_model.predict(X_test)
elasticnet_predictions = elasticnet_model.predict(X_test)
```

```
[ ]: # Evaluate the models
ridge_mse = mean_squared_error(y_test, ridge_predictions)
lasso_mse = mean_squared_error(y_test, lasso_predictions)
elasticnet_mse = mean_squared_error(y_test, elasticnet_predictions)

print("Ridge Mean Squared Error:", ridge_mse)
print("Lasso Mean Squared Error:", lasso_mse)
print("ElasticNet Mean Squared Error:", elasticnet_mse)
```

```
Ridge Mean Squared Error: 107379845764229.9
Lasso Mean Squared Error: 107379845764235.2
ElasticNet Mean Squared Error: 107379845764289.69
```

Classification Models:

Decision Trees or Random Forests: To classify channels as successful or not.

```
[ ]: # Import necessary libraries
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load the dataset and preprocess if needed
# Select features and target variable
# Split the data into training and testing sets
```

```
[ ]: # Create and train a Decision Tree Classifier
decision_tree_model = DecisionTreeClassifier(random_state=42)
decision_tree_model.fit(X_train, y_train)
```

```
[ ]: DecisionTreeClassifier(random_state=42)
```

```
[ ]: # Create and train a Random Forest Classifier
random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest_model.fit(X_train, y_train)
```

```
[ ]: RandomForestClassifier(random_state=42)
```

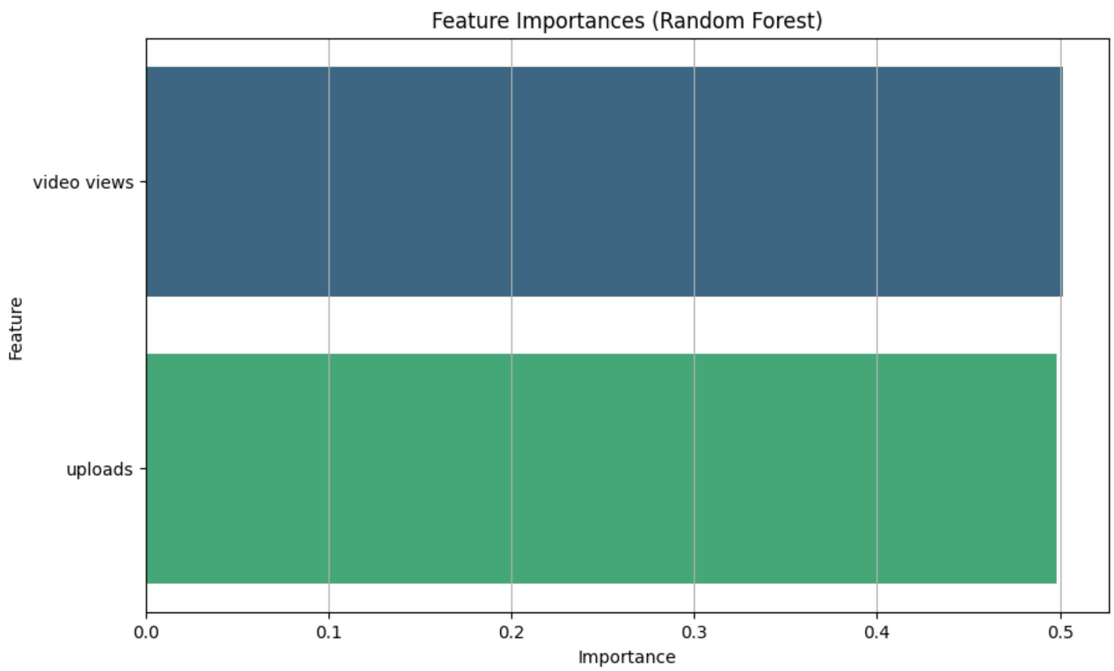
```
[ ]: # Make predictions on the test set
decision_tree_predictions = decision_tree_model.predict(X_test)
random_forest_predictions = random_forest_model.predict(X_test)
```

```
[ ]: # Evaluate the models
decision_tree_accuracy = accuracy_score(y_test, decision_tree_predictions)
```

```
random_forest_accuracy = accuracy_score(y_test, random_forest_predictions)

print("Decision Tree Accuracy:", decision_tree_accuracy)
print("Random Forest Accuracy:", random_forest_accuracy)
```

```
Decision Tree Accuracy: 0.009009009009009009
Random Forest Accuracy: 0.009009009009009009
```

[ ]:
```
# Visualize feature importances from Random Forest Classifier
importances = random_forest_model.feature_importances_
features = X.columns
plt.figure(figsize=(10, 6))
sns.barplot(x=importances, y=features, palette="viridis")
plt.title('Feature Importances (Random Forest)')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.grid(axis='x')
plt.show()
```



Logistic Regression: To predict the channel type (individual or brand).

[ ]:
```
# Import necessary libraries
from sklearn.linear_model import LogisticRegression

# Load the dataset and preprocess if needed
# Select features and target variable
```

```
# Split the data into training and testing sets
```

```
[ ]: # Create and train a Logistic Regression model
logistic_regression_model = LogisticRegression()
logistic_regression_model.fit(X_train, y_train)
```

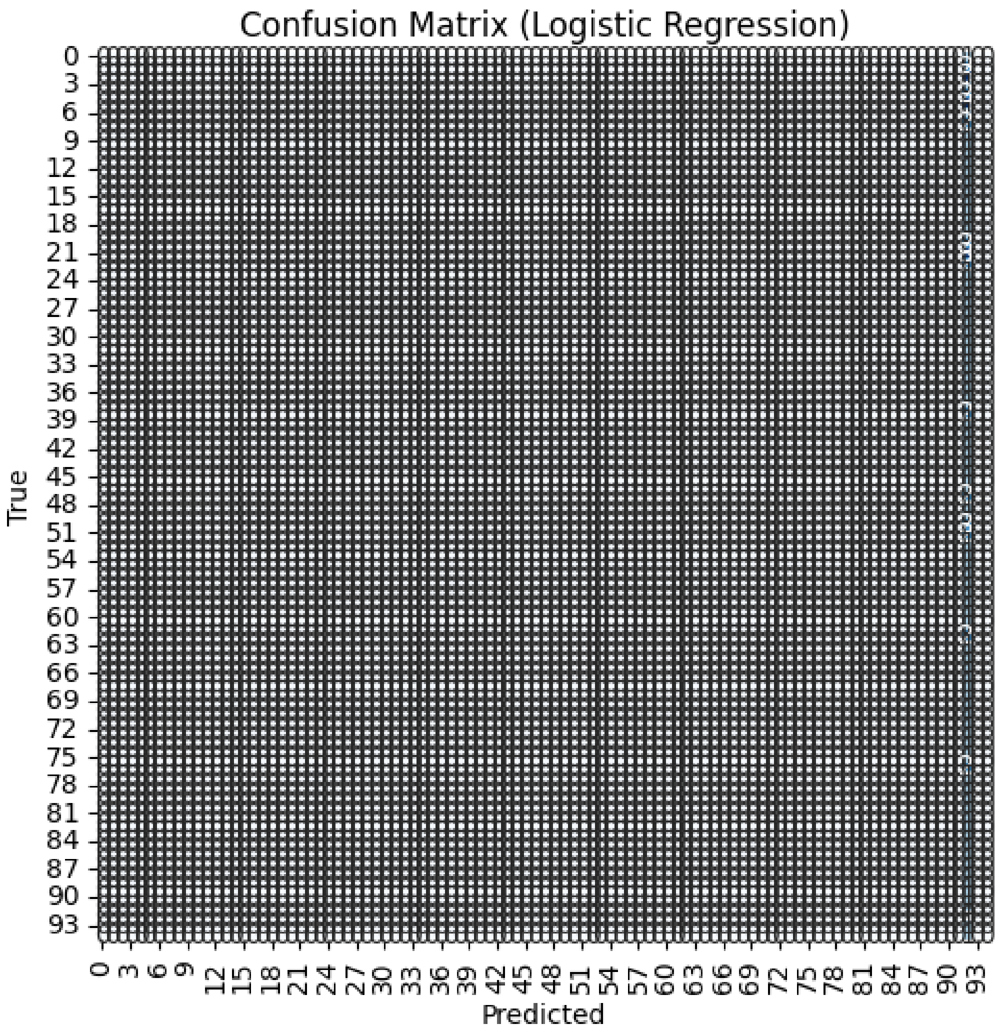
```
[ ]: LogisticRegression()
```

```
[ ]: # Make predictions on the test set
logistic_regression_predictions = logistic_regression_model.predict(X_test)
```

```
[ ]: # Evaluate the model
logistic_regression_accuracy = accuracy_score(y_test,␣
 ↪logistic_regression_predictions)

print("Logistic Regression Accuracy:", logistic_regression_accuracy)
```

```
Logistic Regression Accuracy: 0.0
```

```
[ ]: # Visualize the confusion matrix for Logistic Regression
from sklearn.metrics import confusion_matrix

conf_matrix = confusion_matrix(y_test, logistic_regression_predictions)
plt.figure(figsize=(6, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title('Confusion Matrix (Logistic Regression)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```



Sentiment Analysis on 'Title' Column:

```
[ ]: # Import the necessary libraries
import nltk
from nltk.corpus import stopwords
from collections import Counter

# Download the stop words list
nltk.download('stopwords')

# Get the titles as text data (assuming your column is named 'Title')
text_data = data['Title']

# Load the stop words list
stop_words = set(stopwords.words('english'))
```

```
# Clean and lowercase the text, remove punctuation
cleaned_text = text_data.str.lower().str.replace('[^\w\s]', '').str.split()

# Remove stop words from the cleaned text
filtered_text = [[word for word in doc if word not in stop_words] for doc in␣
↪cleaned_text]

# Calculate term frequencies
flat_list = [item for sublist in filtered_text for item in sublist]
term_frequencies = Counter(flat_list)

# Show the most common terms and their frequencies
most_common_terms = term_frequencies.most_common(10)
print(most_common_terms)
```

```
[('-', 26), ('kids', 19), ('tv', 16), ('music', 14), ('official', 12), ('&',
11), ('news', 11), ('rhymes', 9), ('songs', 9), ('ýýýýýýýý', 9)]
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\pnrde\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[ ]:
```
# Import the necessary libraries
from textblob import TextBlob

# Get the titles as a list (assuming your column is named 'Title')
titles = data['Title']

# Perform sentiment analysis for each title
sentiments = []
for title in titles:
    analysis = TextBlob(title)
    sentiment = analysis.sentiment.polarity  # Sentiment ranges from -1␣
↪(negative) to 1 (positive)
    sentiments.append(sentiment)

# Add the sentiment scores to your dataframe
data['Title_Sentiments'] = sentiments

# Examine the results
data.head()
```

[ ]:
```
   rank                   Youtuber  subscribers   video views  \
0     1                   T-Series    245000000  2.280000e+11
2     3                    MrBeast    166000000  2.836884e+10
3     4  Cocomelon - Nursery Rhymes    162000000  1.640000e+11
4     5                  SET India    159000000  1.480000e+11
```

```
8    9                Like Nastya    106000000  9.047906e+10

         category                        Title  uploads        Country  \
0           Music                     T-Series    20082          India
2   Entertainment                      MrBeast      741  United States
3       Education  Cocomelon - Nursery Rhymes      966  United States
4           Shows                    SET India   116536          India
8  People & Blogs             Like Nastya Vlog      493         Russia

    channel_type  video_views_rank  …  lowest_yearly_earnings  \
0          Music               1.0  …               6800000.0
2  Entertainment              48.0  …               4000000.0
3      Education               2.0  …               5900000.0
4  Entertainment               3.0  …               5500000.0
8         People             630.0  …                146800.0

   highest_yearly_earnings  subscribers_for_last_30_days  created_year  \
0              108400000.0                     2000000.0        2006.0
2               64700000.0                     8000000.0        2012.0
3               94800000.0                     1000000.0        2006.0
4               87500000.0                     1000000.0        2006.0
8                2300000.0                      100000.0        2016.0

   Gross tertiary education enrollment (%)    Population  Unemployment rate  \
0                                     28.1  1.366418e+09               5.36
2                                     88.2  3.282395e+08              14.70
3                                     88.2  3.282395e+08              14.70
4                                     28.1  1.366418e+09               5.36
8                                     81.9  1.443735e+08               4.59

   Urban_population  Title_Sentiment  Title_Sentiments
0       471031528.0              0.0               0.0
2       270663028.0              0.0               0.0
3       270663028.0              0.0               0.0
4       471031528.0              0.0               0.0
8       107683889.0              0.0               0.0

[5 rows x 25 columns]
```