youtube data analysis 1

September 21, 2023

[ ]:
```
import pandas as pd
import numpy as np
```

[ ]:
```
import pandas as pd

# Use a raw string literal by adding 'r' before the string
file_path = r"C:\Users\siranjeevi\Dropbox\PC\Downloads\Global YouTube␣
 ↪Statistics.csv"

# Load the CSV file
df = pd.read_csv(file_path, encoding='latin')

# Display the DataFrame
df
```

[ ]:
```
     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
..    ...                          …          …           …
990   991                 Natan por Aï¿     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  \
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
..                …                            …      …              …
990           Sports                 Natan por Aï¿     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  \
0             IN          Music  …                     2000000.0
1             US          Games  …                           NaN
2             US  Entertainment  …                     8000000.0
3             US      Education  …                     1000000.0
4             IN  Entertainment  …                     1000000.0
..           …            …  …                          …
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  \
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
..            …              …           …
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  \
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
..                                       …          …                 …
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
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
..                …        …         …
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

[995 rows x 28 columns]
```

[ ]:
```
df.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
```

[ ]:
```
df.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]
```

[ ]:
```
print(df.duplicated().any())
print(df.shape)
```

```
False
(995, 28)
```

[ ]:
```
df.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')
```

[ ]:
```
for dtype in ['object', 'float', 'int']:
    print (f'Columns of {dtype} type:')
print(df.select_dtypes (include=[dtype]).columns.tolist())
print()
```

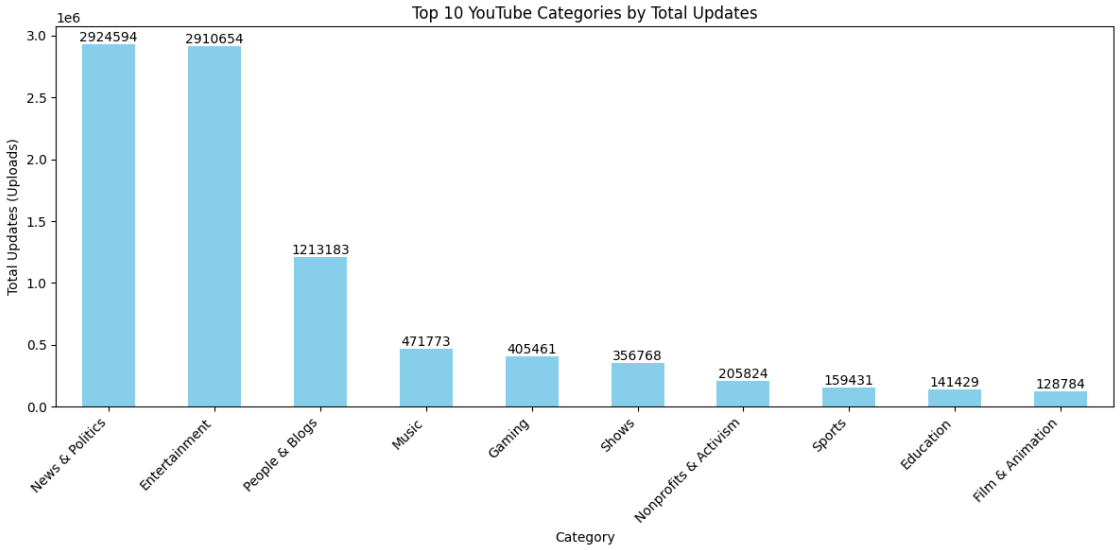
```
Columns of object type:
Columns of float type:
Columns of int type:
['rank', 'subscribers', 'uploads']
```

[ ]:
```
#data visualization
```

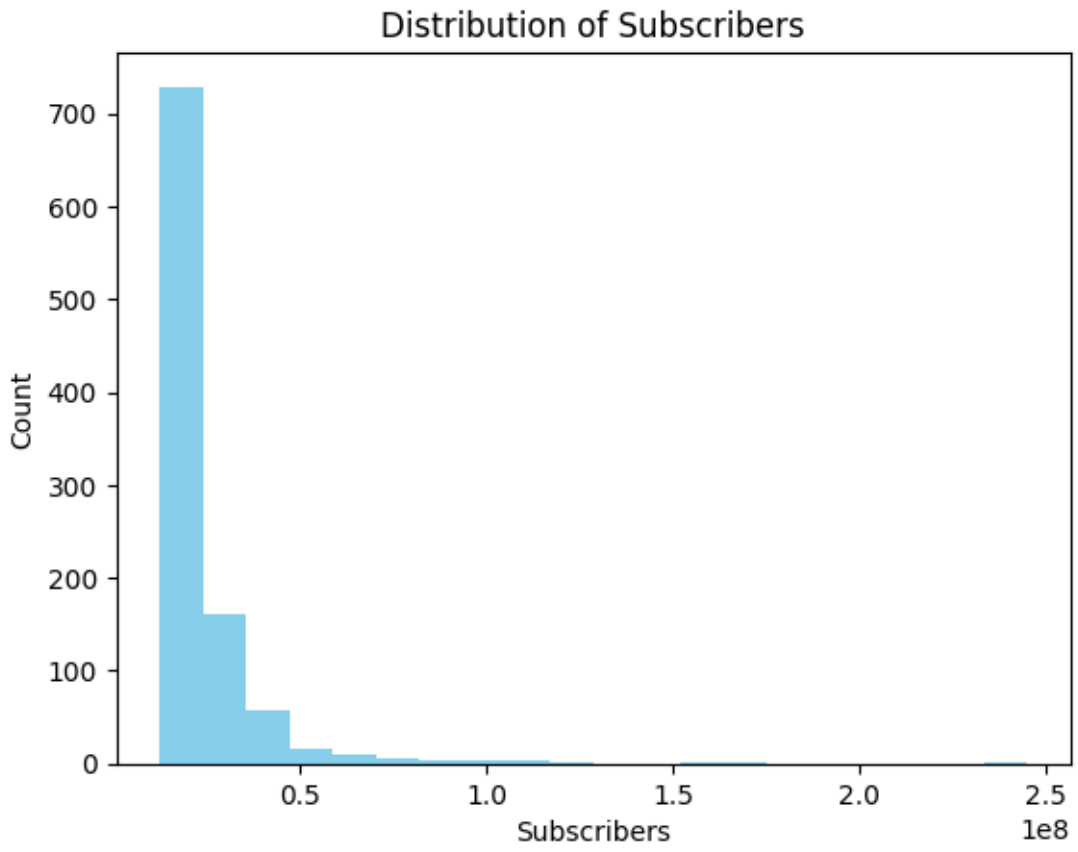
[ ]:
```
df['category'].unique
```

[ ]:
```
<bound method Series.unique of 0                 Music
1       Film & Animation
2          Entertainment
3              Education
4                  Shows
              …
990                Sports
991       People & Blogs
992        Entertainment
993               Gaming
994               Comedy
Name: category, Length: 995, dtype: object>
```

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Ensure 'uploads' column is numeric (it may already be numeric)
df['uploads'] = pd.to_numeric(df['uploads'], errors='coerce')

# Group by 'category' and sum the 'uploads' for each category, then select the␣
 ↪top 10
category_updates = df.groupby('category')['uploads'].sum().nlargest(10)

# Create a bar chart
plt.figure(figsize=(12, 6))
bars = category_updates.plot(kind='bar', color='skyblue')
plt.xlabel('Category')
plt.ylabel('Total Updates (Uploads)')
plt.title('Top 10 YouTube Categories by Total Updates')
plt.xticks(rotation=45, ha='right')

# Add labels for each bar
for idx, value in enumerate(category_updates):
    plt.text(idx, value, str(int(value)), ha='center', va='bottom', fontsize=10)

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



[ ]:
```
# Plot a histogram of subscribers
plt.hist(df['subscribers'], bins=20, color='skyblue')
plt.xlabel('Subscribers')
plt.ylabel('Count')
plt.title('Distribution of Subscribers')
plt.show()
```



[ ]:
```
import pandas as pd

# Load data from a CSV file into a Pandas DataFrame with 'latin1' encoding
df = pd.read_csv(r"C:\Users\siranjeevi\Dropbox\PC\Downloads\Global YouTube␣
 ↪Statistics.csv", encoding='latin1')

# Display the DataFrame
print(df)
```

```
     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
..    …                            …           …           …
990   991                 Natan por Aï¿     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  \
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
..                …                           …      …              …
990            Sports                Natan por Aï¿     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  \
0             IN          Music  …                     2000000.0
1             US          Games  …                           NaN
2             US  Entertainment  …                     8000000.0
3             US      Education  …                     1000000.0
4             IN  Entertainment  …                     1000000.0
..           …            …  …                           …
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  \
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
..            …              …           …
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  \
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
..                                       …           …                 …
```

```
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
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
..                 …         …         …
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

[995 rows x 28 columns]
```

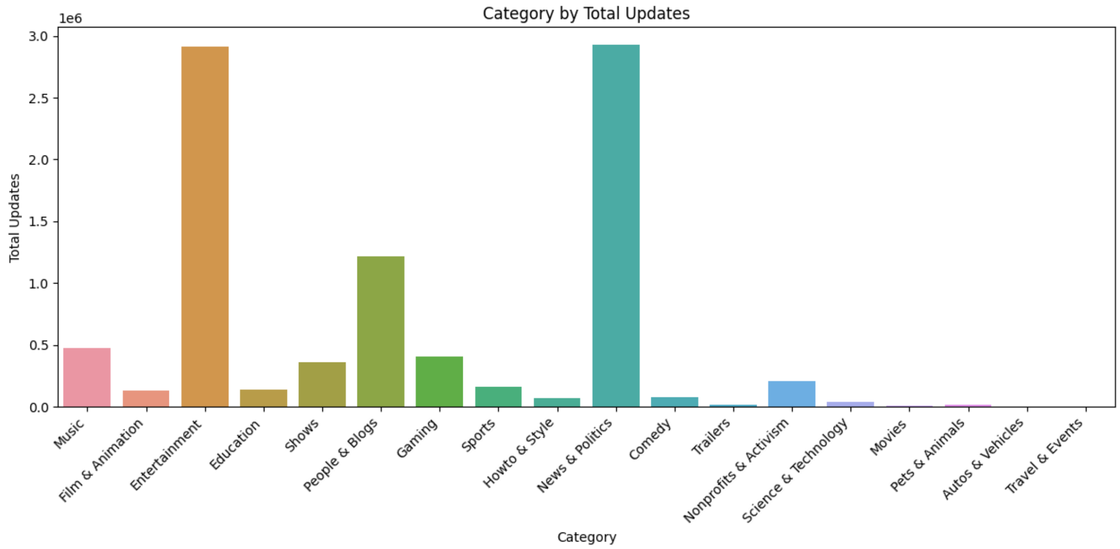
[ ]:
```python
# Handling missing values
for col in df.select_dtypes(include=['float64', 'int64']).columns:
    df[col].fillna(df[col].mean(), inplace=True)

for col in df.select_dtypes(include=['object']).columns:
    df[col].fillna(df[col].mode()[0], inplace=True)

# Checking for duplicated rows
print("Duplicated Rows:", df.duplicated().any())
```

```
Duplicated Rows: False
```

[ ]:
```python
# Section 1: Creating a bar chart for category by total updates
plt.figure(figsize=(12, 6))
sns.barplot(data=df, x='category', y='uploads', estimator=sum, ci=None)
plt.xticks(rotation=45, ha='right')
plt.title("Category by Total Updates")
plt.xlabel("Category")
plt.ylabel("Total Updates")
plt.tight_layout()
plt.show()
```

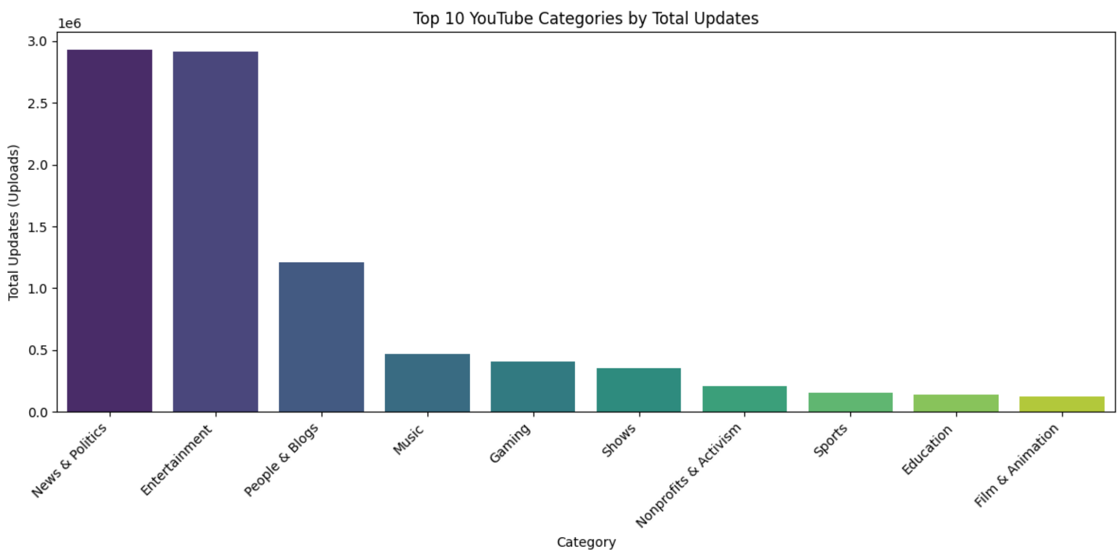
```
C:\Users\siranjeevi\AppData\Local\Temp\ipykernel_13160\437821085.py:3:
FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=df, x='category', y='uploads', estimator=sum, ci=None)
```

```
c:\Users\siranjeevi\AppData\Local\Programs\Python\Python311\Lib\site-
packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is
deprecated and will be removed in a future version. Use isinstance(dtype,
CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\siranjeevi\AppData\Local\Programs\Python\Python311\Lib\site-
packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is
deprecated and will be removed in a future version. Use isinstance(dtype,
CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\siranjeevi\AppData\Local\Programs\Python\Python311\Lib\site-
packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is
deprecated and will be removed in a future version. Use isinstance(dtype,
CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



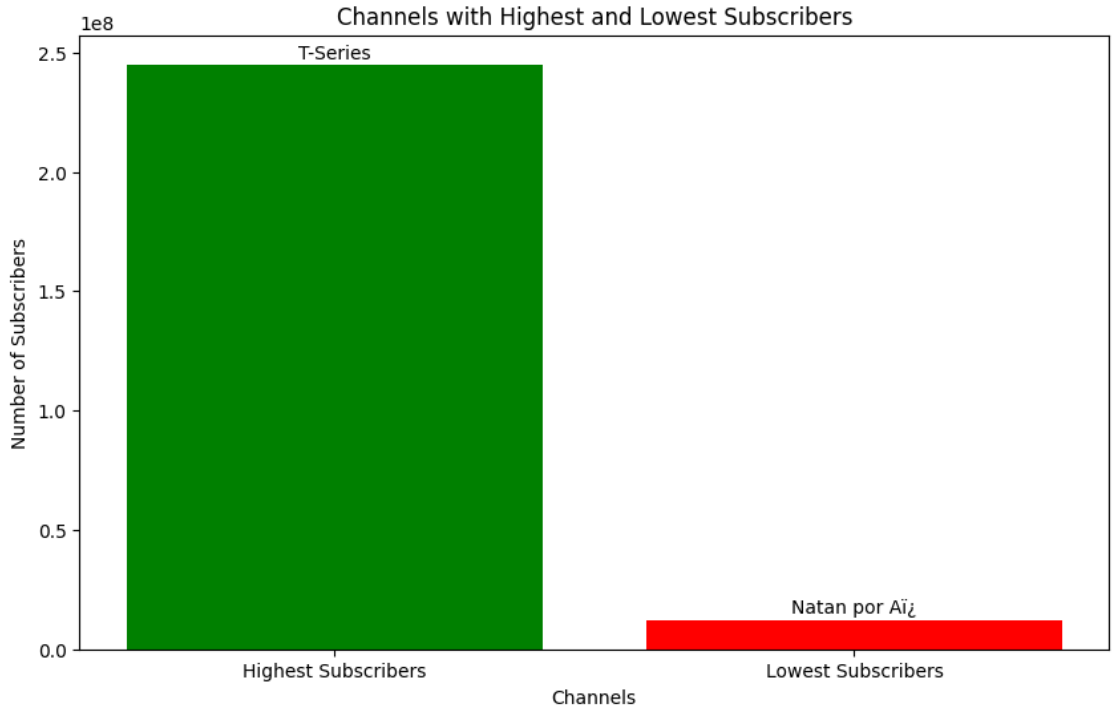
[ ]:
```
# Section 2: Creating a bar chart for the top 10 YouTube categories by total␣
↪updates
category_updates = df.groupby('category')['uploads'].sum().nlargest(10)
plt.figure(figsize=(12, 6))
sns.barplot(x=category_updates.index, y=category_updates.values,␣
↪palette='viridis')
plt.xticks(rotation=45, ha='right')
plt.title('Top 10 YouTube Categories by Total Updates')
plt.xlabel('Category')
plt.ylabel('Total Updates (Uploads)')
plt.tight_layout()
plt.show()
```

```
c:\Users\siranjeevi\AppData\Local\Programs\Python\Python311\Lib\site-
packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is
deprecated and will be removed in a future version. Use isinstance(dtype,
CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\siranjeevi\AppData\Local\Programs\Python\Python311\Lib\site-
packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is
deprecated and will be removed in a future version. Use isinstance(dtype,
CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\siranjeevi\AppData\Local\Programs\Python\Python311\Lib\site-
packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is
deprecated and will be removed in a future version. Use isinstance(dtype,
CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



[ ]:
```
# Section 3: Finding channels with the highest and lowest subscribers
highest_subscribers = df[df['subscribers'] == df['subscribers'].max()]
lowest_subscribers = df[df['subscribers'] == df['subscribers'].min()]
highest_subscribers_channel_name = highest_subscribers['Youtuber'].values[0]
lowest_subscribers_channel_name = lowest_subscribers['Youtuber'].values[0]

plt.figure(figsize=(10, 6))
plt.bar(['Highest Subscribers', 'Lowest Subscribers'],␣
 ↪[highest_subscribers['subscribers'].values[0],
                                                                    ␣
 ↪lowest_subscribers['subscribers'].values[0]],
        color=['green', 'red'])
```

```
plt.xlabel('Channels')
plt.ylabel('Number of Subscribers')
plt.title('Channels with Highest and Lowest Subscribers')
plt.text(0, highest_subscribers['subscribers'].values[0] + 1000000,␣
 ↪f'{highest_subscribers_channel_name}',
         ha='center', va='bottom', fontsize=10)
plt.text(1, lowest_subscribers['subscribers'].values[0] + 1000000,␣
 ↪f'{lowest_subscribers_channel_name}',
         ha='center', va='bottom', fontsize=10)
plt.show()
```



[ ]:
```
import pandas as pd
import plotly.express as px


# Load data from a CSV file into a Pandas DataFrame
df = pd.read_csv(r"C:\Users\siranjeevi\Dropbox\PC\Downloads\Global YouTube␣
 ↪Statistics.csv", encoding='latin1')

# Section 4: Creating a bar chart for channels with highest and lowest␣
 ↪subscribers using Plotly
channel_with_highest_subscribers = df[df['subscribers'] == df['subscribers'].
 ↪max()]
```

```
channel_with_lowest_subscribers = df[df['subscribers'] == df['subscribers'].
 ↪min()]
channels = pd.concat([channel_with_highest_subscribers,␣
 ↪channel_with_lowest_subscribers])
fig = px.bar(channels, x='Youtuber', y='subscribers', color='Youtuber',
             labels={'Youtuber': 'Channel', 'subscribers': 'Subscribers'},
             title='Channels with Highest and Lowest Subscribers',
             text='subscribers', height=400)
fig.update_xaxes(categoryorder='total ascending', title_text='')
fig.update_yaxes(title_text='Number of Subscribers')
fig.update_traces(texttemplate='%{text}', textposition='outside')
fig.show()
```

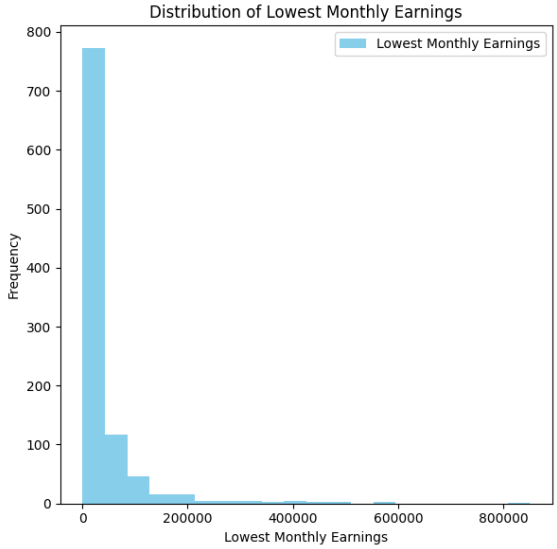
[ ]:
```
# Section 5: Creating a scatter plot for subscribers vs. video views
fig = px.scatter(df, x='subscribers', y='video views', title='Subscribers vs.␣
 ↪Video Views',
                 labels={'subscribers': 'Number of Subscribers', 'video views':␣
 ↪'Number of Video Views'})
fig.update_traces(marker=dict(size=8), selector=dict(mode='markers+lines'))
fig.update_layout(hovermode='closest')
fig.update_xaxes(showgrid=True)
fig.update_yaxes(showgrid=True)
fig.show()
```

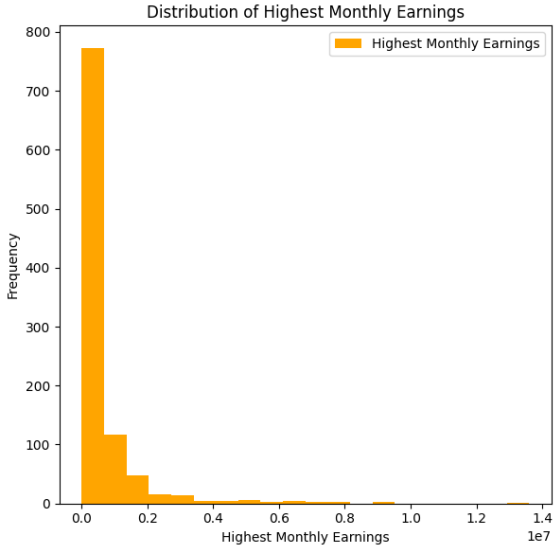
[ ]:
```
# Section 6: Calculating and printing the correlation coefficient between␣
 ↪subscribers and uploads
correlation_coefficient = df['subscribers'].corr(df['uploads'])
print(f'Correlation Coefficient: {correlation_coefficient:.2f}')
```

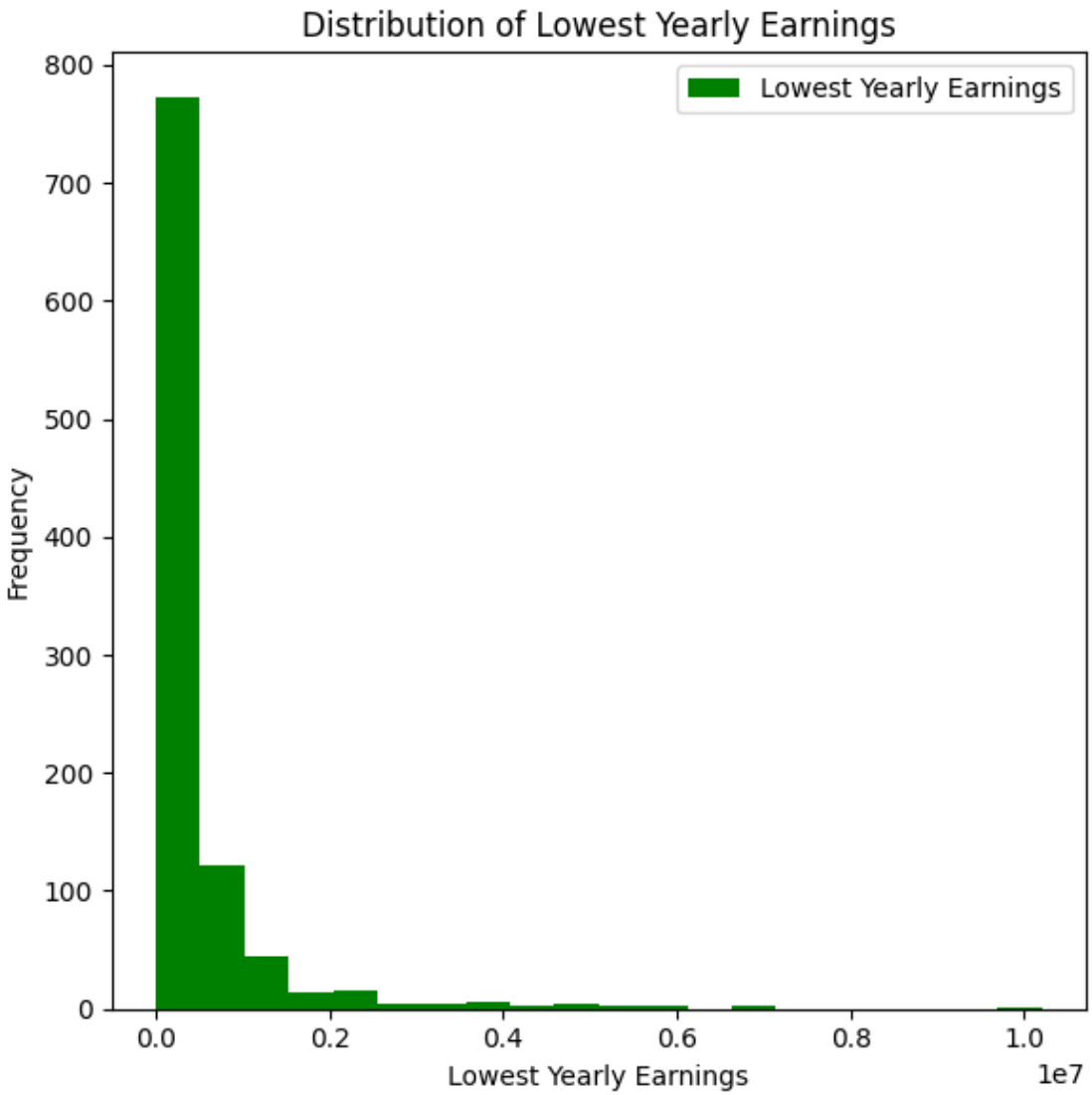
```
Correlation Coefficient: 0.08
```

[ ]:
```
import matplotlib.pyplot as plt

# Plot histograms for lowest and highest monthly earnings
plt.figure(figsize=(12, 6))

# Plot a histogram for lowest monthly earnings
plt.subplot(1, 2, 1)
plt.hist(df['lowest_monthly_earnings'], bins=20, color='skyblue', label='Lowest␣
 ↪Monthly Earnings')
plt.xlabel('Lowest Monthly Earnings')
plt.ylabel('Frequency')
plt.title('Distribution of Lowest Monthly Earnings')
plt.legend()

# Plot a histogram for highest monthly earnings
plt.subplot(1, 2, 2)
```

```
plt.hist(df['highest_monthly_earnings'], bins=20, color='orange',␣
  ↪label='Highest Monthly Earnings')
plt.xlabel('Highest Monthly Earnings')
plt.ylabel('Frequency')
plt.title('Distribution of Highest Monthly Earnings')
plt.legend()

plt.tight_layout()
plt.show()

# Plot a histogram for lowest yearly earnings
plt.figure(figsize=(6, 6))
plt.hist(df['lowest_yearly_earnings'], bins=20, color='green', label='Lowest␣
  ↪Yearly Earnings')
plt.xlabel('Lowest Yearly Earnings')
plt.ylabel('Frequency')
plt.title('Distribution of Lowest Yearly Earnings')
plt.legend()
plt.tight_layout()
plt.show()
```





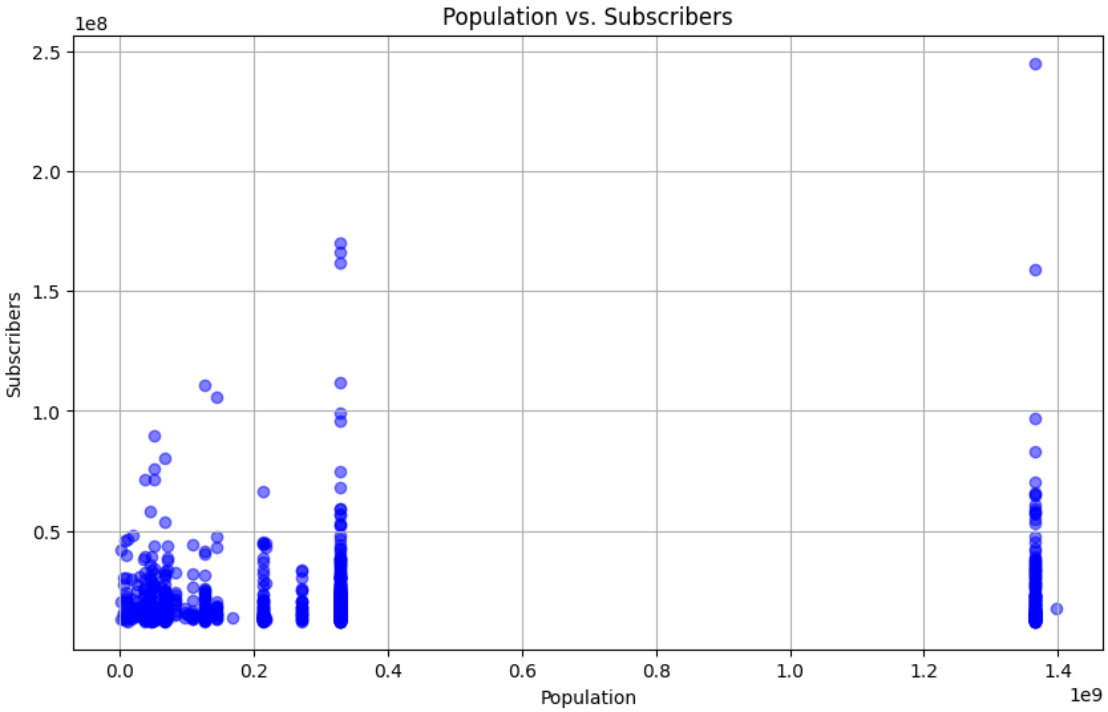


[ ]:
```
# Assuming you have a 'Country' column in your DataFrame
country_with_highest_channels = df['Country'].value_counts().idxmax()
highest_channel_count = df['Country'].value_counts().max()

print(f"The country with the highest number of YouTube channels is␣
  ↪{country_with_highest_channels} with {highest_channel_count} channels.")
```

```
The country with the highest number of YouTube channels is United States with
313 channels.
```

[ ]:
```
# Assuming you have 'Population' and 'subscribers' columns in your DataFrame
plt.figure(figsize=(10, 6))
```

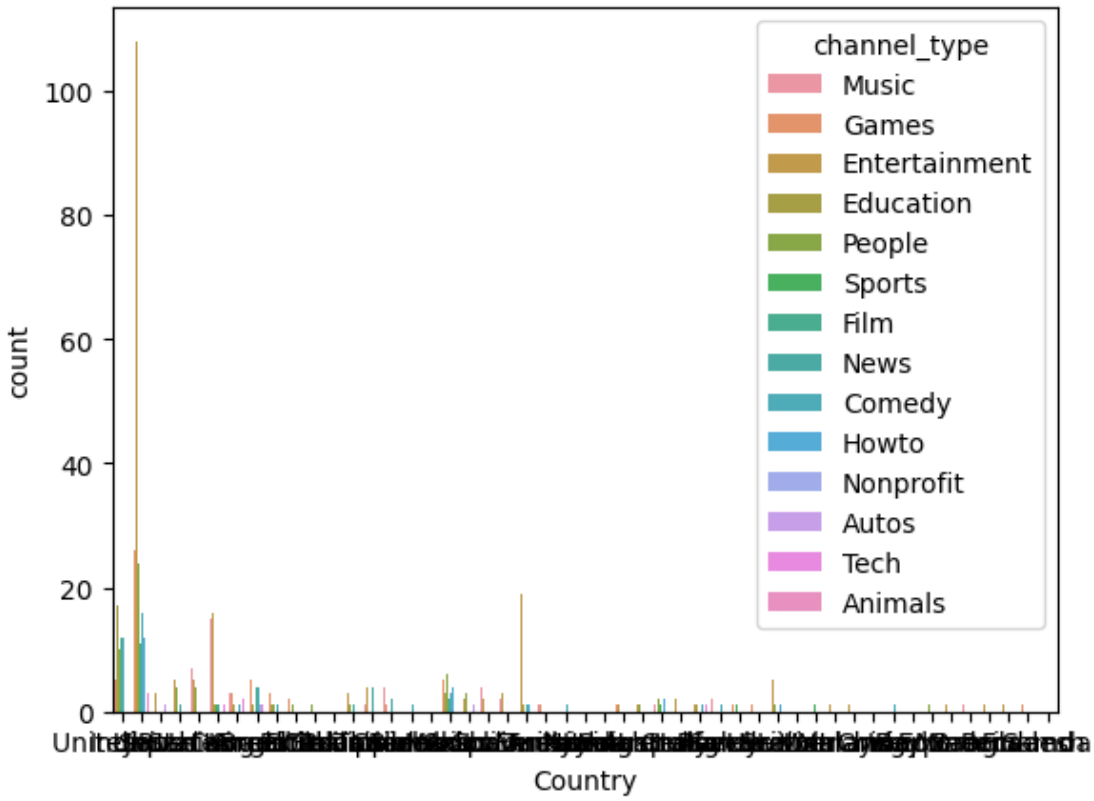
```
plt.scatter(df['Population'], df['subscribers'], alpha=0.5, color='blue')
plt.xlabel('Population')
plt.ylabel('Subscribers')
plt.title('Population vs. Subscribers')
plt.grid(True)
plt.show()
```



[ ]:
```
sns.countplot(data=df, x='Country', hue='channel_type')
import warnings

# Suppress FutureWarning messages
warnings.filterwarnings("ignore", category=FutureWarning)
```

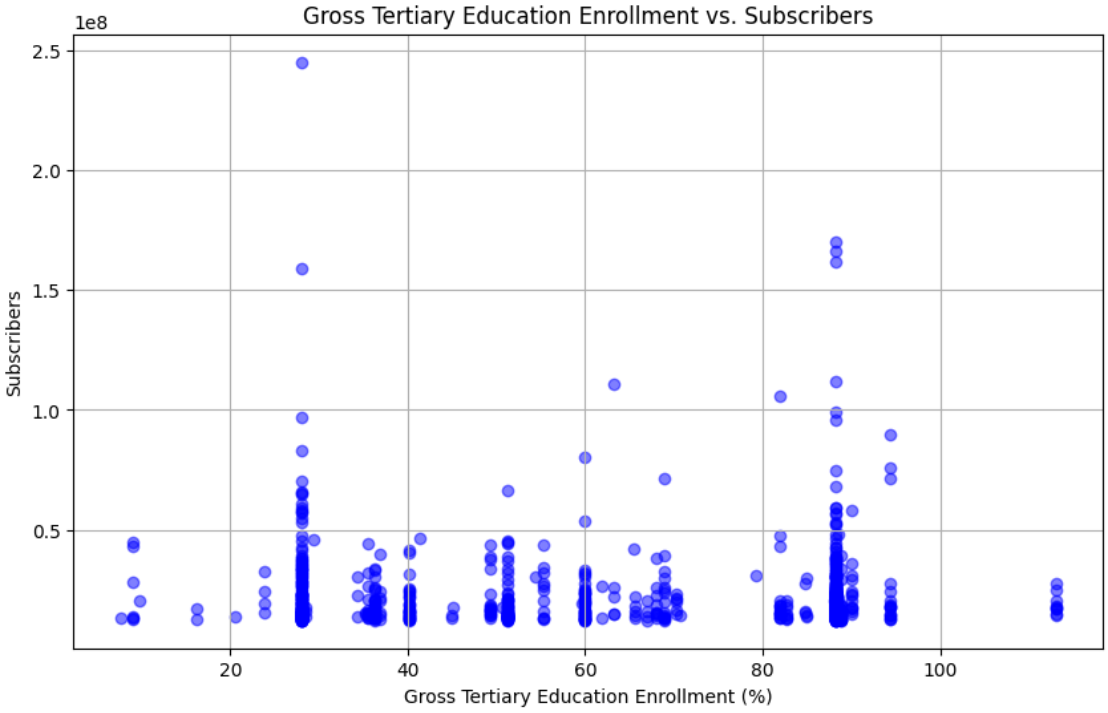
```
c:\Users\siranjeevi\AppData\Local\Programs\Python\Python311\Lib\site-
packages\seaborn\_oldcore.py:1498: FutureWarning:

is_categorical_dtype is deprecated and will be removed in a future version. Use
isinstance(dtype, CategoricalDtype) instead

c:\Users\siranjeevi\AppData\Local\Programs\Python\Python311\Lib\site-
packages\seaborn\_oldcore.py:1498: FutureWarning:

is_categorical_dtype is deprecated and will be removed in a future version. Use
isinstance(dtype, CategoricalDtype) instead
```

```
c:\Users\siranjeevi\AppData\Local\Programs\Python\Python311\Lib\site-
packages\seaborn\_oldcore.py:1498: FutureWarning:

is_categorical_dtype is deprecated and will be removed in a future version. Use
isinstance(dtype, CategoricalDtype) instead

c:\Users\siranjeevi\AppData\Local\Programs\Python\Python311\Lib\site-
packages\seaborn\_oldcore.py:1498: FutureWarning:

is_categorical_dtype is deprecated and will be removed in a future version. Use
isinstance(dtype, CategoricalDtype) instead
```




```
[ ]: # Assuming you have columns 'Gross tertiary education enrollment (%)' and␣
      ↪'subscribers' in your DataFrame
     correlation_coefficient = df['Gross tertiary education enrollment (%)'].
      ↪corr(df['subscribers'])

     # Create a scatter plot
     plt.figure(figsize=(10, 6))
```

```
plt.scatter(df['Gross tertiary education enrollment (%)'], df['subscribers'],␣
 ↪alpha=0.5, color='blue')
plt.xlabel('Gross Tertiary Education Enrollment (%)')
plt.ylabel('Subscribers')
plt.title('Gross Tertiary Education Enrollment vs. Subscribers')
plt.grid(True)
plt.show()

# Print the correlation coefficient
print(f'Correlation Coefficient between Gross Tertiary Education Enrollment and␣
 ↪Subscribers: {correlation_coefficient:.2f}')
```




```
Correlation Coefficient between Gross Tertiary Education Enrollment and
Subscribers: -0.01
```

[ ]:
```
# Assuming you have columns 'subscribers', 'video views', 'category', and␣
 ↪'Title' in your DataFrame

# Rank channels by subscribers within each category
df['Subscribers Rank'] = df.groupby('category')['subscribers'].
 ↪rank(ascending=False)

# Rank channels by video views within each category
```

```
df['Video Views Rank'] = df.groupby('category')['video views'].
  ↪rank(ascending=False)

# Rank channels by category
df['Category Rank'] = df['category'].rank()

# Rank channels by title (you can use other criteria or metrics)
df['Title Rank'] = df['Title'].rank()

# Print the DataFrame with ranking columns
print(df[['Youtuber', 'category', 'Title', 'subscribers', 'video views',␣
  ↪'Subscribers Rank', 'Video Views Rank', 'Category Rank', 'Title Rank']])
```

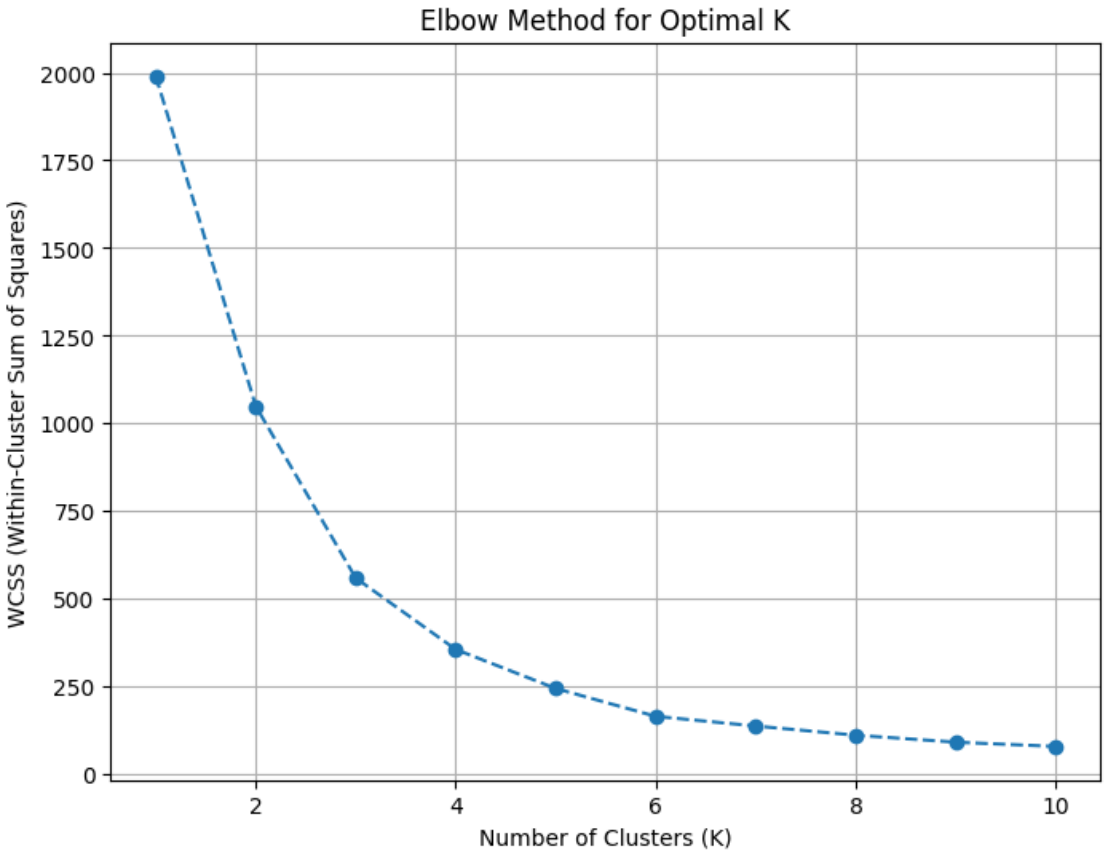
```
                    Youtuber          category                    Title  \
0                    T-Series             Music                  T-Series
1               YouTube Movies  Film & Animation              youtubemovies
2                     MrBeast     Entertainment                   MrBeast
3    Cocomelon - Nursery Rhymes         Education  Cocomelon - Nursery Rhymes
4                   SET India             Shows                 SET India
..                        …                 …                         …
990             Natan por Aï¿            Sports             Natan por Aï¿
991   Free Fire India Official    People & Blogs   Free Fire India Official
992                     Panda               NaN                 HybridPanda
993               RobTopGames            Gaming                 RobTopGames
994               Make Joke Of            Comedy                 Make Joke Of

     subscribers   video views  Subscribers Rank  Video Views Rank  \
0      245000000  2.280000e+11               1.0               1.0
1      170000000  0.000000e+00               1.0              46.0
2      166000000  2.836884e+10               1.0              12.0
3      162000000  1.640000e+11               1.0               1.0
4      159000000  1.480000e+11               1.0               1.0
..           …             …                 …                 …
990     12300000  9.029610e+09              11.0               5.0
991     12300000  1.674410e+09             132.0             124.0
992     12300000  2.214684e+09               NaN               NaN
993     12300000  3.741235e+08              94.0              93.0
994     12300000  2.129774e+09              69.0              62.0

     Category Rank  Title Rank
0            640.5       707.0
1            380.5       957.0
2            237.0       524.0
3             94.0       149.0
4            929.0       644.0
..             …           …
990          941.0       546.0
991          835.5       262.0
```

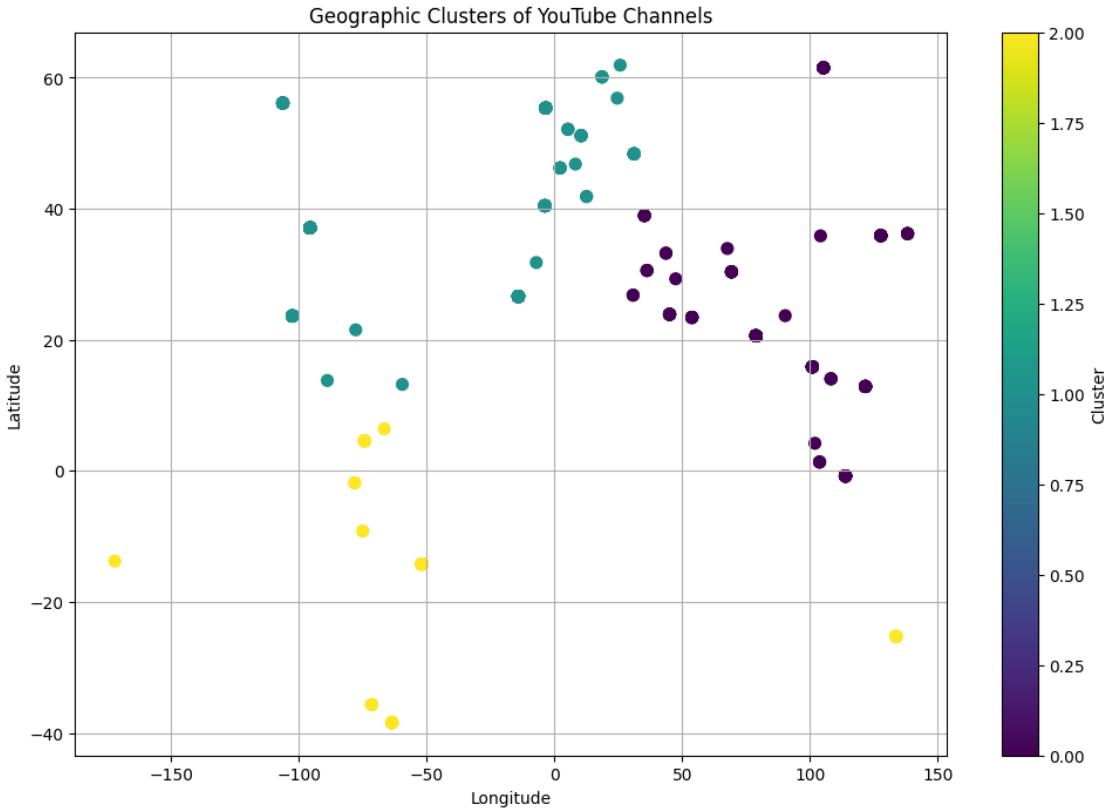
```
992            NaN       327.0
993          450.5       629.0
994           37.0       478.0

[995 rows x 9 columns]
```

[ ]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer  # Import the imputer

# Assuming you have columns 'Latitude' and 'Longitude' in your DataFrame
# Replace 'df' with your DataFrame name

# Select latitude and longitude columns
data = df[['Latitude', 'Longitude']]

# Create an imputer instance
imputer = SimpleImputer(strategy='mean')

# Fit the imputer to your data and transform it
data[['Latitude', 'Longitude']] = imputer.fit_transform(data[['Latitude',␣
 ↪'Longitude']])

# Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

# Determine the optimal number of clusters (K) using the Elbow Method
wcss = []  # Within-cluster sum of squares
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,␣
 ↪random_state=0)
    kmeans.fit(scaled_data)
    wcss.append(kmeans.inertia_)

# Plot the Elbow Method to choose the optimal K
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('WCSS (Within-Cluster Sum of Squares)')
plt.grid(True)
plt.show()
```

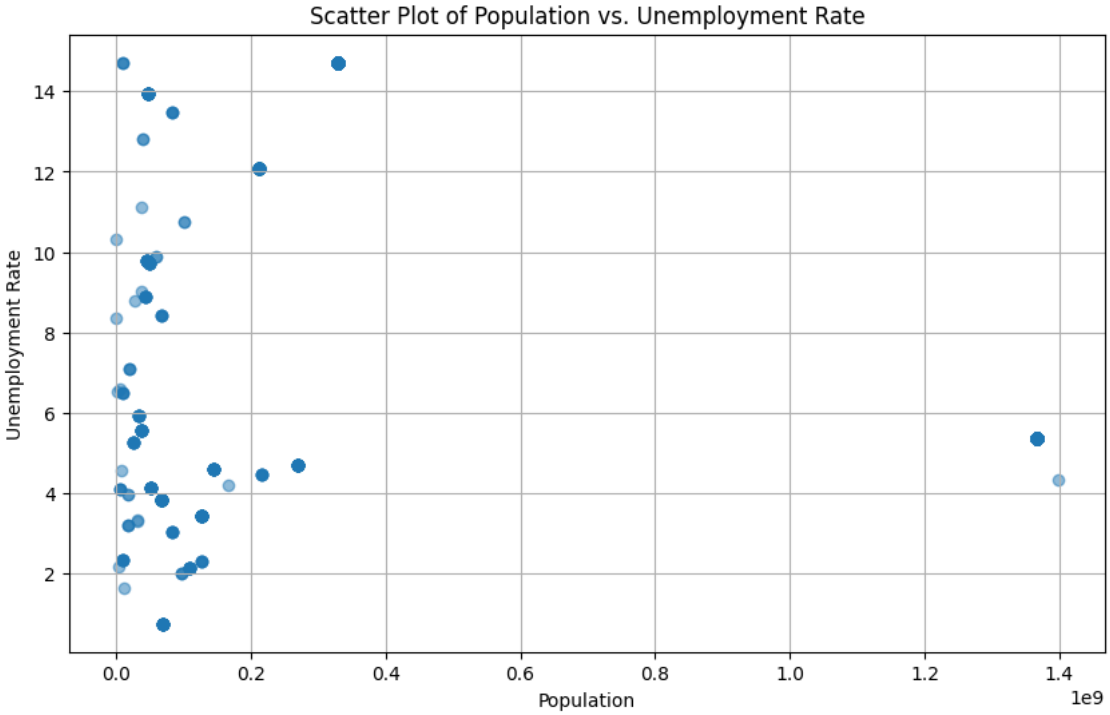
```
# Based on the Elbow Method, choose the optimal K (number of clusters)
optimal_k = 3  # Adjust this based on the Elbow Method plot

# Perform K-Means clustering with the optimal K
kmeans = KMeans(n_clusters=optimal_k, init='k-means++', max_iter=300,␣
 ↪n_init=10, random_state=0)
cluster_labels = kmeans.fit_predict(scaled_data)

# Add cluster labels to the DataFrame
df['Cluster'] = cluster_labels

# Plot the clusters on a map
plt.figure(figsize=(12, 8))
plt.scatter(df['Longitude'], df['Latitude'], c=df['Cluster'], cmap='viridis',␣
 ↪s=50)
plt.title('Geographic Clusters of YouTube Channels')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.colorbar(label='Cluster')
plt.grid(True)
plt.show()

# Analyze and interpret the clusters
# You can now analyze the clusters to identify regions where YouTube channels␣
 ↪are more popular.
```

```
C:\Users\siranjeevi\AppData\Local\Temp\ipykernel_15056\92689115.py:18:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```


Elbow Method for Optimal K
WCSS (Within-Cluster Sum of Squares)
Number of Clusters (K)
2000
1750
1500
1250
1000
750
500
250
0
2
4
6
8
10



[ ]:
```
import matplotlib.pyplot as plt

# Assuming you have columns 'Population' and 'Unemployment rate' in your␣
 ↪DataFrame
# Replace 'df' with your DataFrame name

# Scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(df['Population'], df['Unemployment rate'], alpha=0.5)
plt.title('Scatter Plot of Population vs. Unemployment Rate')
plt.xlabel('Population')
plt.ylabel('Unemployment Rate')
plt.grid(True)
plt.show()
```


Scatter Plot of Population vs. Unemployment Rate
Unemployment Rate
14
12
10
8
6
4
2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Population
1e9