In [1]:
```
# importing numpy library(Numerical Python)
import numpy as np
# importing pandas library(Panel Data)
import pandas as pd
```

In [2]:
```
#Read File
#latin-1 useful when working with text data which contain special characters
df = pd.read_csv("Global YouTube Statistics.csv", encoding='latin-1')
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]
```

In [3]:
```
df.shape
```

Out[3]:
```
(995, 28)
```

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

Out[4]:
```
27860
```

In [5]:
```
# Dimenison
df.ndim
```

Out[5]:
```
2
```

In [6]:
```
df.columns
```

Out[6]:
```
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')
```

In [7]:
```
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
```

In [8]:
```
df.describe()
```

Out[8]:

	rank	subscribers	video views	uploads	video_views_rank	country_rank	channel_type_rank	video_views_for_the_last_3
count	995.00000	9.950000e+02	9.950000e+02	995.000000	9.940000e+02	879.000000	962.000000	9.3900
mean	498.00000	2.298241e+07	1.103954e+10	9187.125628	5.542489e+05	386.053470	745.719335	1.7561
std	287.37606	1.752611e+07	1.411084e+10	34151.352254	1.362782e+06	1232.244746	1944.386561	4.1637
min	1.00000	1.230000e+07	0.000000e+00	0.000000	1.000000e+00	1.000000	1.000000	1.0000
25%	249.50000	1.450000e+07	4.288145e+09	194.500000	3.230000e+02	11.000000	27.000000	2.0137
50%	498.00000	1.770000e+07	7.760820e+09	729.000000	9.155000e+02	51.000000	65.500000	6.4085
75%	746.50000	2.460000e+07	1.355470e+10	2667.500000	3.584500e+03	123.000000	139.750000	1.6882
max	995.00000	2.450000e+08	2.280000e+11	301308.000000	4.057944e+06	7741.000000	7741.000000	6.5890

8 rows × 21 columns

In [9]:
```
#Create a copy of df
df = df.copy()
```

In [10]:
```
df.columns = df.columns.str.title()
print(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')
```

In [11]:
```
# Delete column 'B'
del df['Rank']
print(df.columns)
```

```
Index(['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')
```

In [12]:
```
print(df.columns)

# Verify if 'Video Views' column exists
if 'Video Views' in df.columns:
    percentile_75th = df['Video Views'].quantile(0.75)
    # Rest of your code that uses 'Video Views' column
else:
    print("The 'Video Views' column does not exist in the DataFrame.")
```

```
Index(['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')
```

In [13]:
```
#Checking duplicate rows
duplicates = df.duplicated()

# Filter the df using the boolean series to examine the duplicates
duplicate_rows = df[duplicates]

# No duplicate rows exist in the df
print(duplicate_rows)
```

```
Empty DataFrame
Columns: [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 27 columns]
```

Missing Value

In [14]:
```
count_missing_value = df.isnull().sum()
print(count_missing_value)
```

```
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
```

In [15]:
```
count_missing_value[count_missing_value>0]
```

Out[15]:
```
Category                                 46
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
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
```

In [16]:
```
#Selecting object datatype
categorical_columns = df.select_dtypes(include = ['object']).columns

# Replace object datatype column missing values with 'Unknown'
df[categorical_columns] = df[categorical_columns].fillna('Unknown')
```

In [17]:
```
#Selecting numeric datatype
numerical_columns = df.select_dtypes(include = ['float', 'int64']).columns

# Replace numerical datatype column missing values with 'Unknown'
df[numerical_columns] = df[numerical_columns].fillna(0)
```

In [18]:
```
#Recalculating number of missing values
count_missing_value = df.isnull().sum()

count_missing_value[count_missing_value > 0]
```

Out[18]:
```
Series([], dtype: int64)
```

In [19]:
```
#Convert float to integers
df = df.astype({'Video_Views_For_The_Last_30_Days':'int64','Video_Views_Rank':'int64','Population':'int64' ,
                'Subscribers_For_Last_30_Days' :'int64','Created_Year':'int64', 'Country_Rank' :'int64',
                'Channel_Type_Rank'  :'int64' ,'Urban_Population'  :'int64','Video Views' :'int64'  })
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 995 entries, 0 to 994
Data columns (total 27 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Youtuber                                995 non-null    object
 1   Subscribers                             995 non-null    int64
 2   Video Views                             995 non-null    int64
 3   Category                                995 non-null    object
 4   Title                                   995 non-null    object
 5   Uploads                                 995 non-null    int64
 6   Country                                 995 non-null    object
 7   Abbreviation                            995 non-null    object
 8   Channel_Type                            995 non-null    object
 9   Video_Views_Rank                        995 non-null    int64
 10  Country_Rank                            995 non-null    int64
 11  Channel_Type_Rank                       995 non-null    int64
 12  Video_Views_For_The_Last_30_Days        995 non-null    int64
 13  Lowest_Monthly_Earnings                 995 non-null    float64
 14  Highest_Monthly_Earnings                995 non-null    float64
 15  Lowest_Yearly_Earnings                  995 non-null    float64
 16  Highest_Yearly_Earnings                 995 non-null    float64
 17  Subscribers_For_Last_30_Days            995 non-null    int64
 18  Created_Year                            995 non-null    int64
 19  Created_Month                           995 non-null    object
 20  Created_Date                            995 non-null    float64
 21  Gross Tertiary Education Enrollment (%) 995 non-null    float64
 22  Population                              995 non-null    int64
 23  Unemployment Rate                       995 non-null    float64
 24  Urban_Population                        995 non-null    int64
 25  Latitude                                995 non-null    float64
 26  Longitude                               995 non-null    float64
dtypes: float64(9), int64(11), object(7)
memory usage: 210.0+ KB
```

In [20]:
```
#Removing non-alphanumeric characters
non_alphanumeric_char = r'[^a-zA-Z0-9\s.,!?&\'-]'
```

In [21]:
```
df['Youtuber'] = df['Youtuber'].str.replace(non_alphanumeric_char, '')
df['Title'] = df['Title'].str.replace(non_alphanumeric_char, '')
```

```
C:\Users\NANDHINI\AppData\Local\Temp\ipykernel_5188\2660877321.py:1: FutureWarning: The default value of regex
will change from True to False in a future version.
  df['Youtuber'] = df['Youtuber'].str.replace(non_alphanumeric_char, '')
C:\Users\NANDHINI\AppData\Local\Temp\ipykernel_5188\2660877321.py:2: FutureWarning: The default value of regex
will change from True to False in a future version.
  df['Title'] = df['Title'].str.replace(non_alphanumeric_char, '')
```

In [22]:
```
#Removig whitespace
df['Youtuber'] = df['Youtuber'].str.strip()
df['Title'] = df['Title'].str.strip()
```

In [23]:
```
#Filter view to remove and replace
filter_youtuber_rows = df['Youtuber'].str.contains(non_alphanumeric_char, regex = True)
print(filter_youtuber_rows)
# regex
filter_title_rows = df['Title'].str.contains(non_alphanumeric_char, regex = True)
```

```
0      False
1      False
2      False
3      False
4      False
       ...
990    False
991    False
992    False
993    False
994    False
Name: Youtuber, Length: 995, dtype: bool
```

In [24]:
```
#String to remove whitespace
filter_youtuber_rows_results = filter_youtuber_rows[filter_youtuber_rows == True]
print(filter_youtuber_rows_results)
```

```
Series([], Name: Youtuber, dtype: bool)
```

In [25]:
```
filter_title_rows_results = filter_title_rows[filter_title_rows == True]
print(filter_title_rows_results)
```

```
Series([], Name: Title, dtype: bool)
```

In [26]:
```
df.sort_values(by = 'Subscribers', ascending = False)
df
```

Out[26]:

	Youtuber	Subscribers	Video Views	Category	Title	Uploads	Country	Abbreviation	Channel_Type	Video_Views_Ra
0	T-Series	245000000	228000000000	Music	T-Series	20082	India	IN	Music	
1	YouTube Movies	170000000	0	Film & Animation	youtubemovies	1	United States	US	Games	40551
2	MrBeast	166000000	28368841870	Entertainment	MrBeast	741	United States	US	Entertainment	
3	Cocomelon - Nursery Rhymes	162000000	164000000000	Education	Cocomelon - Nursery Rhymes	966	United States	US	Education	
4	SET India	159000000	148000000000	Shows	SET India	116536	India	IN	Entertainment	
...	...	...	...	...	...	...	...	...	...	
990	Natan por A	12300000	9029609749	Sports	Natan por A	1200	Brazil	BR	Entertainment	5
991	Free Fire India Official	12300000	1674409945	People & Blogs	Free Fire India Official	1500	India	IN	Games	61
992	Panda	12300000	2214684303	Unknown	HybridPanda	2452	United Kingdom	GB	Games	129(
993	RobTopGames	12300000	374123483	Gaming	RobTopGames	39	Sweden	SE	Games	351
994	Make Joke Of	12300000	2129773714	Comedy	Make Joke Of	62	India	IN	Comedy	45

995 rows × 27 columns

In [27]:
```
df = df.reset_index(drop = True)
df
```

Out[27]:

	Youtuber	Subscribers	Video Views	Category	Title	Uploads	Country	Abbreviation	Channel_Type	Video_Views_Ra
0	T-Series	245000000	228000000000	Music	T-Series	20082	India	IN	Music	
1	YouTube Movies	170000000	0	Film & Animation	youtubemovies	1	United States	US	Games	40551
2	MrBeast	166000000	28368841870	Entertainment	MrBeast	741	United States	US	Entertainment	
3	Cocomelon - Nursery Rhymes	162000000	164000000000	Education	Cocomelon - Nursery Rhymes	966	United States	US	Education	
4	SET India	159000000	148000000000	Shows	SET India	116536	India	IN	Entertainment	
...	...	...	...	...	...	...	...	...	...	
990	Natan por A	12300000	9029609749	Sports	Natan por A	1200	Brazil	BR	Entertainment	5
991	Free Fire India Official	12300000	1674409945	People & Blogs	Free Fire India Official	1500	India	IN	Games	61
992	Panda	12300000	2214684303	Unknown	HybridPanda	2452	United Kingdom	GB	Games	129(
993	RobTopGames	12300000	374123483	Gaming	RobTopGames	39	Sweden	SE	Games	351
994	Make Joke Of	12300000	2129773714	Comedy	Make Joke Of	62	India	IN	Comedy	45

995 rows × 27 columns

In [28]:
```
numerical_summary = df.describe().round().astype(int)
print(numerical_summary)
```

```
       Subscribers  Video Views  Uploads  Video_Views_Rank  Country_Rank  \
count          995          995      995               995           995
mean       22982412  -2147483648     9187            553692           341
std        17526105  -2147483648    34151           1362210          1165
min        12300000            0        0                 0             0
25%        14500000  -2147483648      194               318             5
50%        17700000  -2147483648      729               913            34
75%        24600000  -2147483648     2668              3579           114
max       245000000  -2147483648   301308           4057944          7741

       Channel_Type_Rank  Video_Views_For_The_Last_30_Days  \
count                995                               995
mean                 721                         165726691
std                 1917                         406500963
min                    0                                 0
25%                   24                          13517000
50%                   62                          56358000
75%                  137                         158565500
max                 7741                       -2147483648

       Lowest_Monthly_Earnings  Highest_Monthly_Earnings  \
count                      995                       995
mean                     36886                    589808
std                      71859                   1148622
min                          0                         0
25%                       2700                     43500
50%                      13300                    212700
75%                      37900                    606800
max                     850900                  13600000

       Lowest_Yearly_Earnings  Highest_Yearly_Earnings  \
count                     995                      995
mean                   442257                  7081814
std                    861216                 13797038
min                         0                        0
25%                     32650                   521750
50%                    159500                  2600000
75%                    455100                  7300000
max                  10200000                163400000

       Subscribers_For_Last_30_Days  Created_Year  Created_Date  \
count                           995           995           995
mean                         230848          2003            16
std                          526109           142             9
min                               0             0             0
25%                               0          2009             8
50%                          100000          2013            16
75%                          200000          2016            23
max                         8000000          2022            31

       Gross Tertiary Education Enrollment (%)  Population  Unemployment Rate  \
count                                      995         995                995
mean                                        56   377183615                  8
std                                         32   464716939                  6
min                                          0           0                  0
25%                                         28    50339443                  4
50%                                         60   270203917                  5
75%                                         88   328239523                 15
max                                        113  1397715000                 15

       Urban_Population  Latitude  Longitude
count               995       995        995
mean          196497954        23        -12
std           162539432        21         79
min                   0       -38       -172
25%            40827302         5        -96
50%           183241641        24         -3
75%           270663028        37         79
max           842933962        62        138
```

```
C:\Users\NANDHINI\anaconda3\Lib\site-packages\pandas\core\dtypes\astype.py:189: RuntimeWarning: invalid value e
ncountered in cast
  return values.astype(dtype, copy=copy)
```

In [29]:
```
object_summary = df[categorical_columns].describe(include = ['object'])
print(object_summary)
```

```
       Youtuber       Category Title        Country Abbreviation  \
count       995            995   995            995          995
unique      978             19   975             50           50
top              Entertainment        United States           US
freq         16            241    16            313          313

        Channel_Type Created_Month
count            995           995
unique            15            13
top     Entertainment           Jan
freq             304           101
```

In [30]:
```
df.to_csv('Global_youtube_statistics_2023_cleaned.csv',index = False, encoding = 'latin-1')
```

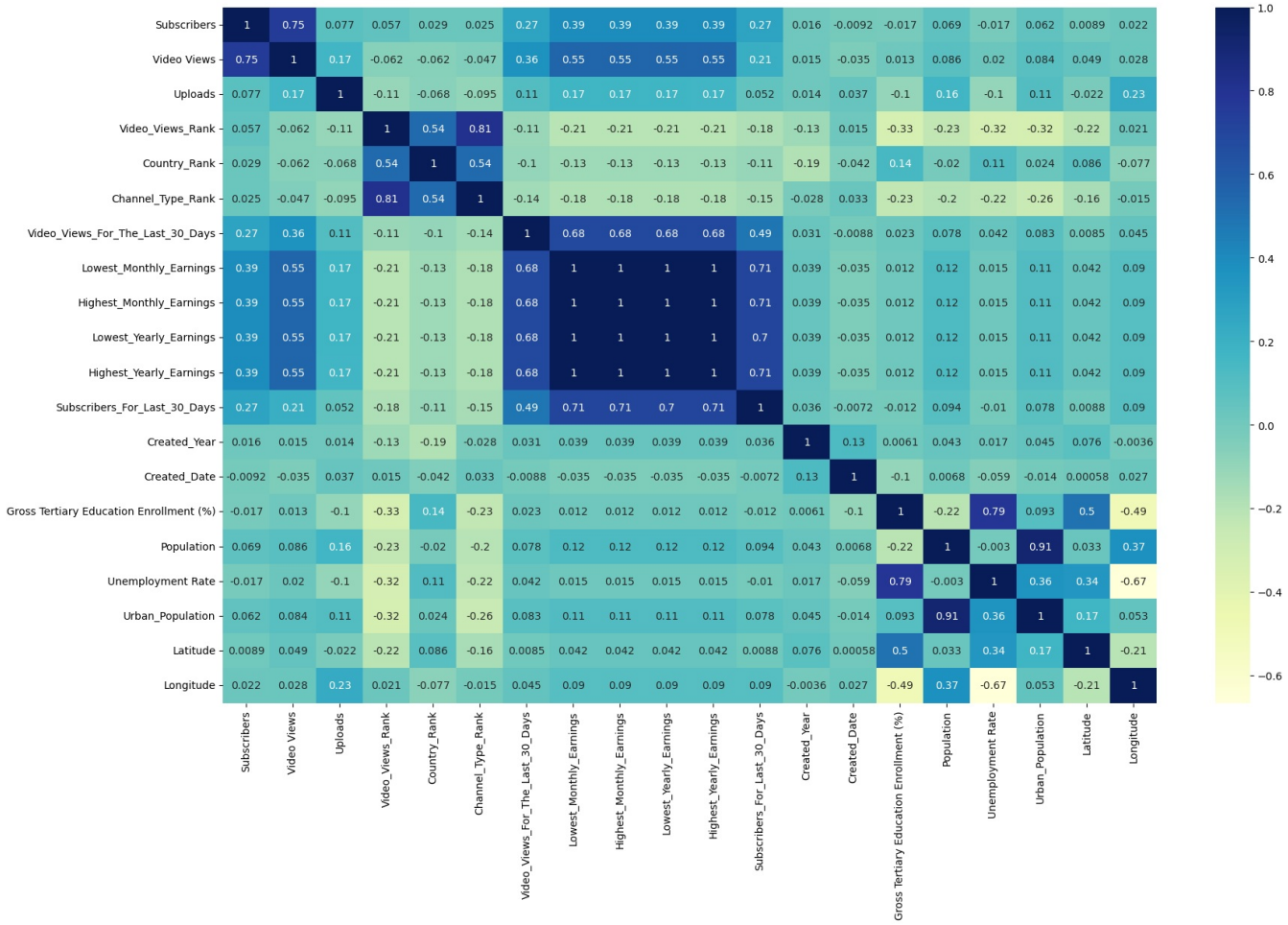
In [31]:
```
#Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python
import matplotlib.pyplot as plt
#Seaborn is a library for making statistical graphics in Python
import seaborn as sns
import plotly.express as px
```

In [32]:
```
most_subscribed_channel = df[df['Subscribers'] == df['Subscribers'].max()]['Youtuber']

print(f"The channel with the most subscribers is:")
print(most_subscribed_channel)
```

```
The channel with the most subscribers is:
0    T-Series
Name: Youtuber, dtype: object
```

In [33]:
```
plt.rcParams['figure.figsize'] = (20, 12)

sns.heatmap(df.corr(), annot=True, cmap='YlGnBu')

# Display the correlation matrix heatmap
plt.show()
```

```
C:\Users\NANDHINI\AppData\Local\Temp\ipykernel_5188\3771483966.py:3: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(), annot=True, cmap='YlGnBu')
```



In [ ]:
```
```

In [34]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 995 entries, 0 to 994
Data columns (total 27 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Youtuber                                995 non-null    object
 1   Subscribers                             995 non-null    int64
 2   Video Views                             995 non-null    int64
 3   Category                                995 non-null    object
 4   Title                                   995 non-null    object
 5   Uploads                                 995 non-null    int64
 6   Country                                 995 non-null    object
 7   Abbreviation                            995 non-null    object
 8   Channel_Type                            995 non-null    object
 9   Video_Views_Rank                        995 non-null    int64
 10  Country_Rank                            995 non-null    int64
 11  Channel_Type_Rank                       995 non-null    int64
 12  Video_Views_For_The_Last_30_Days        995 non-null    int64
 13  Lowest_Monthly_Earnings                 995 non-null    float64
 14  Highest_Monthly_Earnings                995 non-null    float64
 15  Lowest_Yearly_Earnings                  995 non-null    float64
 16  Highest_Yearly_Earnings                 995 non-null    float64
 17  Subscribers_For_Last_30_Days            995 non-null    int64
 18  Created_Year                            995 non-null    int64
 19  Created_Month                           995 non-null    object
 20  Created_Date                            995 non-null    float64
 21  Gross Tertiary Education Enrollment (%) 995 non-null    float64
 22  Population                              995 non-null    int64
 23  Unemployment Rate                       995 non-null    float64
 24  Urban_Population                        995 non-null    int64
 25  Latitude                                995 non-null    float64
 26  Longitude                               995 non-null    float64
dtypes: float64(9), int64(11), object(7)
memory usage: 210.0+ KB
```

In [35]:

```python
max_subscribers = df['Subscribers'].max()
max_views = df['Video Views'].max()
# Converting into trillion
def format_and_check_trillion(number):
    if number == 1e12:
        formatted_number = f'{number / 1e12:,.2f} trillion'
    else:
        formatted_number = f'{number:,.0f}'
    return formatted_number

number = max_views
formatted_number = format_and_check_trillion(number)


max_subscribers_channel = df[df['Subscribers'] == max_subscribers]['Youtuber'].values[0]
print(f"channel '{max_subscribers_channel}' with most subscribers '{max_subscribers}' and views are '{formatted
```

```
channel 'T-Series' with most subscribers '245000000' and views are '228,000,000,000'
```

In [36]:

```python
# Group by 'Country' and 'Channel Type' and count occurrences
channel_counts = df.groupby(['Country', 'Category']).size().reset_index(name='Count')

# Find the top channel type in each country
top_channel_by_country = channel_counts.groupby('Country').apply(lambda x: x.loc[x['Count'].idxmax()])

print(top_channel_by_country)
```

Country	Country	Category	Count
Afghanistan	Afghanistan	Music	1
Andorra	Andorra	Howto & Style	1
Argentina	Argentina	Music	4
Australia	Australia	Gaming	3
Bangladesh	Bangladesh	News & Politics	1
Barbados	Barbados	Entertainment	1
Brazil	Brazil	Music	16
Canada	Canada	Music	5
Chile	Chile	Gaming	3
China	China	Howto & Style	1
Colombia	Colombia	Music	7
Cuba	Cuba	Gaming	1
Ecuador	Ecuador	Entertainment	1
Egypt	Egypt	Howto & Style	1
El Salvador	El Salvador	Gaming	1
Finland	Finland	Gaming	1
France	France	Music	2
Germany	Germany	Entertainment	2
India	India	Entertainment	45
Indonesia	Indonesia	Entertainment	12
Iraq	Iraq	Comedy	1
Italy	Italy	Entertainment	2
Japan	Japan	Entertainment	3
Jordan	Jordan	Entertainment	2
Kuwait	Kuwait	Gaming	1
Latvia	Latvia	Comedy	1
Malaysia	Malaysia	Film & Animation	1
Mexico	Mexico	Entertainment	8
Morocco	Morocco	Music	1
Netherlands	Netherlands	Gaming	1
Pakistan	Pakistan	Entertainment	3
Peru	Peru	Entertainment	1
Philippines	Philippines	Entertainment	3
Russia	Russia	People & Blogs	8
Samoa	Samoa	Music	1
Saudi Arabia	Saudi Arabia	Entertainment	3
Singapore	Singapore	Entertainment	1
South Korea	South Korea	Music	7
Spain	Spain	Gaming	8
Sweden	Sweden	Gaming	1
Switzerland	Switzerland	Sports	1
Thailand	Thailand	Entertainment	9
Turkey	Turkey	Entertainment	2
Ukraine	Ukraine	Entertainment	5
United Arab Emirates	United Arab Emirates	People & Blogs	2
United Kingdom	United Kingdom	Music	14
United States	United States	Music	73
Unknown	Unknown	Entertainment	28
Venezuela	Venezuela	People & Blogs	1
Vietnam	Vietnam	Comedy	1

In [37]:
```
max_uploads_channel = df[df['Uploads'] == df['Uploads'].max()]

# Get the name and count of the channel with the maximum uploads
Youtuber_name = max_uploads_channel['Youtuber'].values[0]
upload_count = max_uploads_channel['Uploads'].values[0]

print(f"The channel with the most uploads is '{Youtuber_name}' with '{upload_count}' uploads.")
```

```
The channel with the most uploads is 'ABP NEWS' with '301308' uploads.
```

In [38]:
```
max_category = df['Category'].value_counts().idxmax()

print(f"Most Category : {max_category}")
count_most_common = df['Category'].value_counts().max()

print(f"The most common channel type is '{max_category}' with a count of {count_most_common}.")
```

```
Most Category : Entertainment
The most common channel type is 'Entertainment' with a count of 241.
```

In [39]:
```
#Top 10 subscribers
df_top =df.nlargest(10, 'Subscribers')
print(df_top)
```

```
                    Youtuber  Subscribers   Video Views          Category  \
0                   T-Series    245000000  228000000000             Music
1              YouTube Movies    170000000             0  Film & Animation
2                     MrBeast    166000000   28368841870     Entertainment
3  Cocomelon - Nursery Rhymes    162000000  164000000000         Education
4                   SET India    159000000  148000000000             Shows
5                       Music    119000000             0           Unknown
6             Kids Diana Show    112000000   93247040539    People & Blogs
7                   PewDiePie    111000000   29058044447            Gaming
8                Like Nastya    106000000   90479060027    People & Blogs
9              Vlad and Niki     98900000   77180169894     Entertainment

                       Title  Uploads        Country Abbreviation  \
0                   T-Series    20082          India           IN
1              youtubemovies        1  United States           US
2                     MrBeast      741  United States           US
3  Cocomelon - Nursery Rhymes      966  United States           US
4                   SET India   116536          India           IN
5                       Music        0        Unknown      Unknown
6             Kids Diana Show     1111  United States           US
7                   PewDiePie     4716          Japan           JP
8            Like Nastya Vlog      493         Russia           RU
9              Vlad and Niki      574  United States           US

    Channel_Type  Video_Views_Rank  ...  Subscribers_For_Last_30_Days  \
0          Music                 1  ...                       2000000
1          Games           4055159  ...                             0
2  Entertainment                48  ...                       8000000
3      Education                 2  ...                       1000000
4  Entertainment                 3  ...                       1000000
5          Music           4057944  ...                             0
6  Entertainment                 5  ...                             0
7  Entertainment                44  ...                             0
8         People               630  ...                        100000
9  Entertainment                 8  ...                        600000

   Created_Year  Created_Month  Created_Date  \
0          2006            Mar          13.0
1          2006            Mar           5.0
2          2012            Feb          20.0
3          2006            Sep           1.0
4          2006            Sep          20.0
5          2013            Sep          24.0
6          2015            May          12.0
7          2010            Apr          29.0
8          2016            Jan          14.0
9          2018            Apr          23.0

   Gross Tertiary Education Enrollment (%)  Population  Unemployment Rate  \
0                                     28.1  1366417754               5.36
1                                     88.2   328239523              14.70
2                                     88.2   328239523              14.70
3                                     88.2   328239523              14.70
4                                     28.1  1366417754               5.36
5                                      0.0           0               0.00
6                                     88.2   328239523              14.70
7                                     63.2   126226568               2.29
8                                     81.9   144373535               4.59
9                                     88.2   328239523              14.70

   Urban_Population   Latitude   Longitude
0         471031528  20.593684   78.962880
1         270663028  37.090240  -95.712891
2         270663028  37.090240  -95.712891
3         270663028  37.090240  -95.712891
4         471031528  20.593684   78.962880
5                 0   0.000000    0.000000
6         270663028  37.090240  -95.712891
7         115782416  36.204824  138.252924
8         107683889  61.524010  105.318756
9         270663028  37.090240  -95.712891

[10 rows x 27 columns]
```

In [40]:
```
print("Top 10 YouTube Channels by No. Subscribers")


for index, row in df_top.iterrows():
    print(f"{row['Title']} - {row['Subscribers']} ")
```

```
Top 10 YouTube Channels by No. Subscribers
T-Series - 245000000
youtubemovies - 170000000
MrBeast - 166000000
Cocomelon - Nursery Rhymes - 162000000
SET India - 159000000
Music - 119000000
Kids Diana Show - 112000000
PewDiePie - 111000000
Like Nastya Vlog - 106000000
Vlad and Niki - 98900000
```

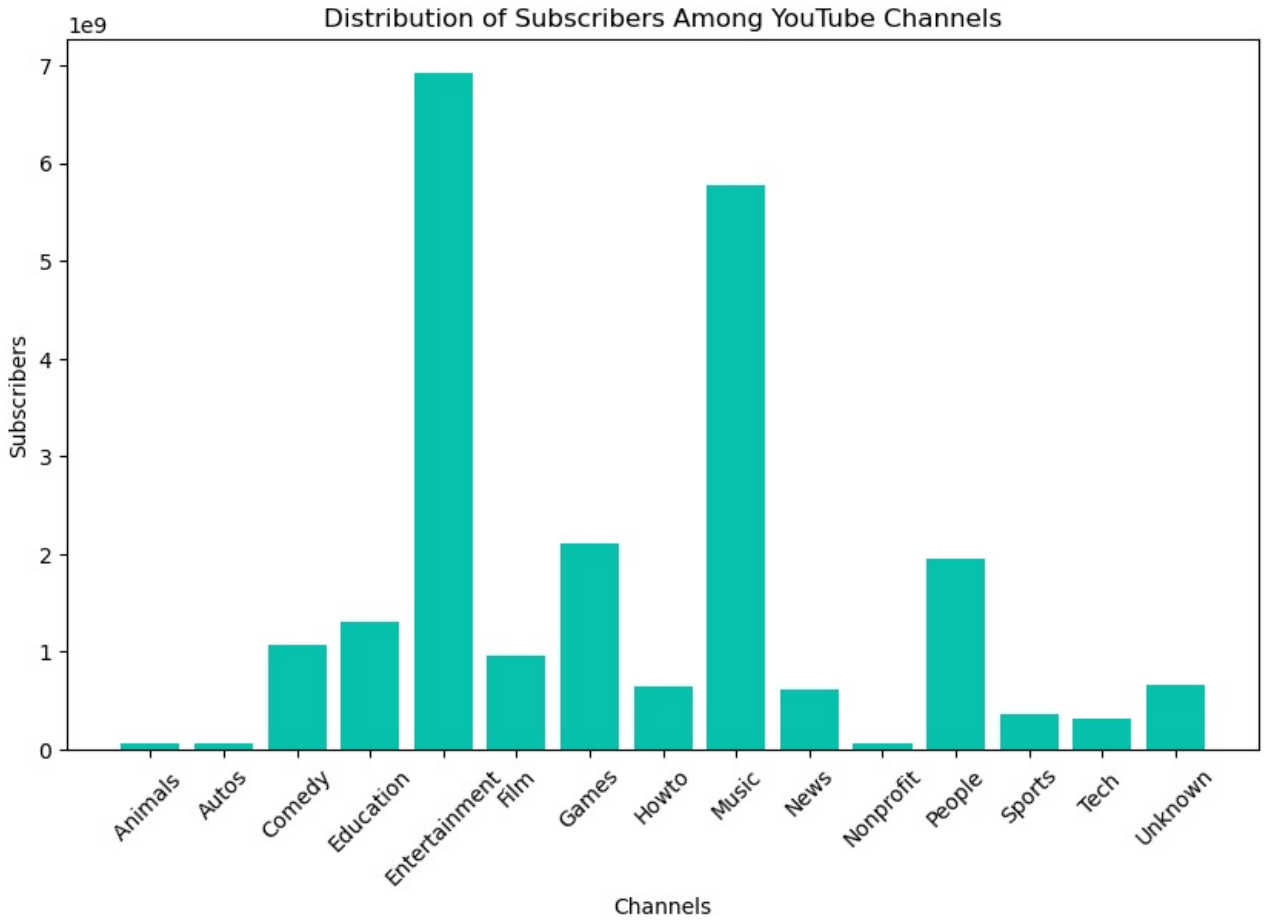
In [41]:
```python
aggregated_data = df.groupby('Channel_Type')['Subscribers'].sum().reset_index()
aggregated_data = aggregated_data.sort_values(by='Subscribers',ascending = False)
print(aggregated_data)
```

```
     Channel_Type  Subscribers
4   Entertainment   6922500000
8           Music   5771500000
6           Games   2111600000
11         People   1953300000
3       Education   1300200000
2          Comedy   1063700000
5            Film    964400000
14        Unknown    658700000
7           Howto    649100000
9            News    611700000
12         Sports    361500000
13           Tech    320800000
1           Autos     64400000
0         Animals     58600000
10      Nonprofit     55500000
```

In [42]:
```python
selected_columns = ['Youtuber', 'Subscribers']

# Filter df for top 10 most subscribed YouTube channels
top_10_channels = df.loc[0:9, selected_columns]

# Sort values so that highest output is descending, this is for the horizontal bar chart
top_10_channels_desc = top_10_channels.sort_values(by = 'Subscribers', ascending = True)

# Validate filtered df output
print(top_10_channels_desc)
```

```
                    Youtuber  Subscribers
9               Vlad and Niki     98900000
8                 Like Nastya    106000000
7                   PewDiePie    111000000
6             Kids Diana Show    112000000
5                       Music    119000000
4                   SET India    159000000
3  Cocomelon - Nursery Rhymes    162000000
2                     MrBeast    166000000
1              YouTube Movies    170000000
0                    T-Series    245000000
```

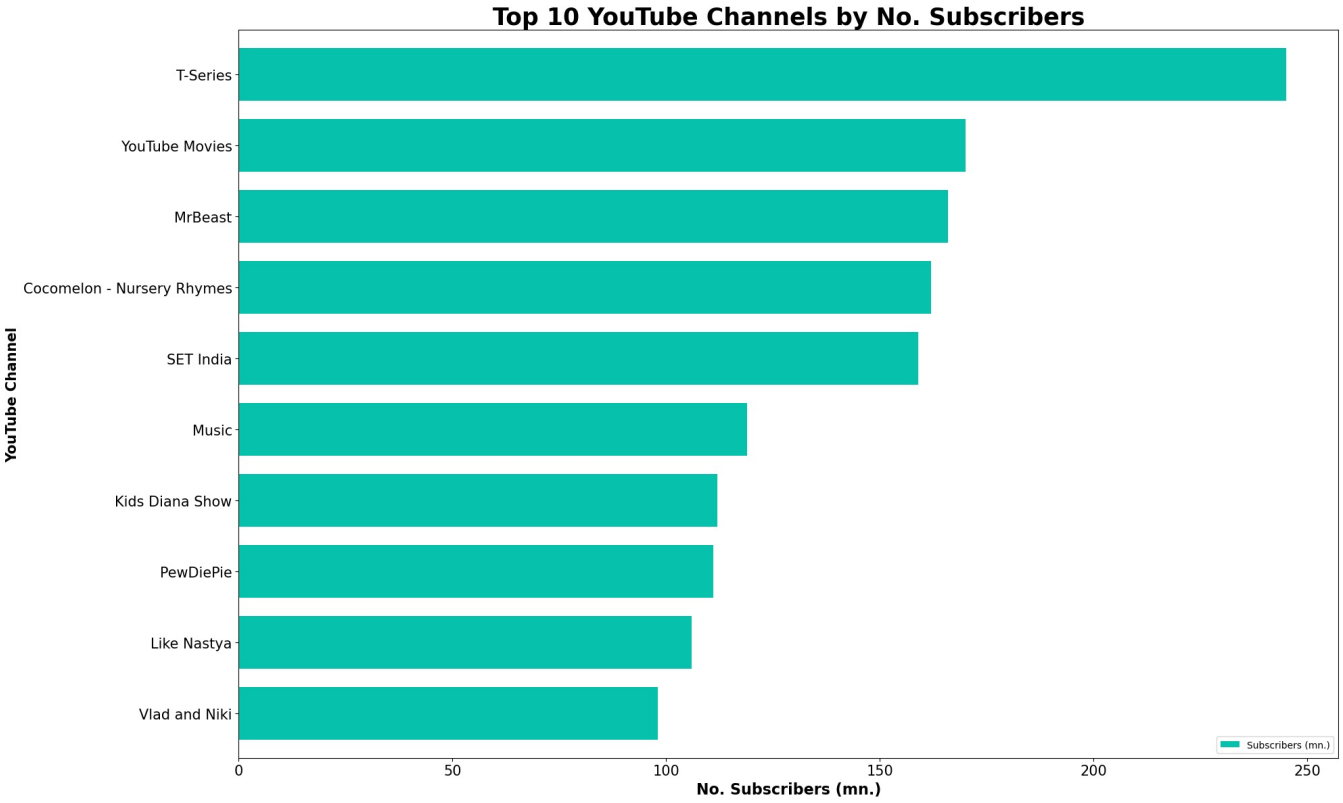
In [43]:
```python
# Sample data (replace this with your actual data)
channels = ['Animals ', 'Autos', 'Comedy', 'Education','Entertainment','Film','Games','Howto','Music','News','N
            'People','Sports','Tech ','Unknown']
subscribers = [58600000, 64400000, 1063700000, 1300200000,6922500000,964400000,2111600000,649100000,5771500000,
               1953300000,361500000,320800000,658700000]

# Create a bar chart to visualize subscriber distribution
plt.figure(figsize=(10, 6))
plt.bar(channels, subscribers, color='#06c2ac')
plt.xlabel('Channels')
plt.ylabel('Subscribers')
plt.title('Distribution of Subscribers Among YouTube Channels')
plt.xticks(rotation=45)
plt.show()
```



In [44]:
```
top_10_channels['Subscribers (mn.)'] = (top_10_channels['Subscribers'] / 1000000).astype(int)
top_10_channels_desc['Subscribers (mn.)'] = (top_10_channels_desc['Subscribers'] / 1000000).astype(int)

# Select columns to show in output
top_10_channels_desc = top_10_channels_desc[['Youtuber', 'Subscribers (mn.)']]

# Validate sorted output
print(top_10_channels_desc)
```

```
                   Youtuber  Subscribers (mn.)
9              Vlad and Niki                 98
8                Like Nastya                106
7                  PewDiePie                111
6            Kids Diana Show                112
5                      Music                119
4                  SET India                159
3  Cocomelon - Nursery Rhymes                162
2                    MrBeast                166
1             YouTube Movies                170
0                    T-Series                245
```

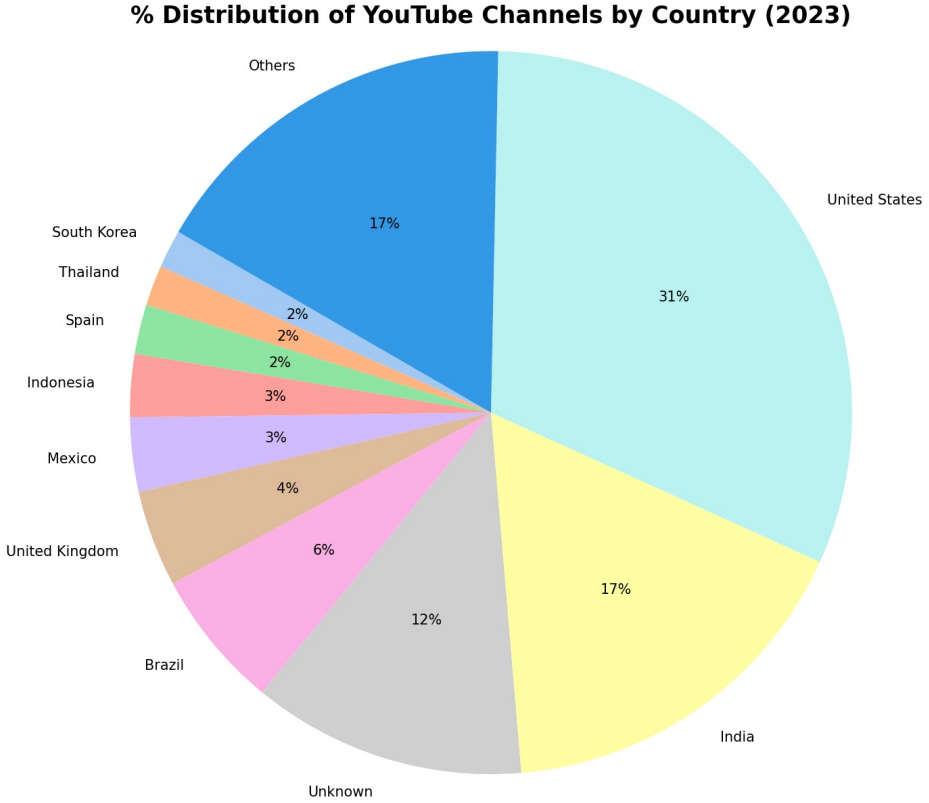
In [45]:
```
top_10_channels_desc.plot.barh(x = 'Youtuber', y = 'Subscribers (mn.)', stacked = True, color = '#06c2ac', widt

# Adjust chart formatting
plt.title('Top 10 YouTube Channels by No. Subscribers', fontsize = 25, weight = 'bold')
plt.xlabel('No. Subscribers (mn.)', fontsize = 16,  weight = 'bold')
plt.ylabel('YouTube Channel', fontsize = 15, weight = 'bold')
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
plt.tight_layout()
plt.show()
```



In [46]:
```
category_counts_asc = df['Category'].value_counts(ascending = True)
top_10_categories_asc = category_counts_asc.tail(10)
```

In [47]:
```
country_counts_asc = df['Country'].value_counts(ascending = True)

# Select the top 10 countries based on the number of YouTube channels
top_10_countries = country_counts_asc.tail(10)

# Data for the top 10 categories and the sum of the remaining categories
data = top_10_countries.tolist()
data += [country_counts_asc[:-10].sum()]

# Labels for the top 10 categories and 'Others' for the remaining categories
labels = top_10_countries.index.tolist() + ['Others']

# Set colour palette for the top 10 categories & a seperate colour for 'Others'
colors = sns.color_palette("pastel", 10)

# Gray color for 'Others'
colors += [(0.2, 0.6, 0.9)]
```

In [48]:
```
wedges, texts, autotexts = plt.pie(data, labels = labels, autopct = '%1.0f%%', colors = colors, startangle = 15

# equal aspect ratio ensures that pie is drawn as a circle
plt.axis('equal')

# increase the font size of the pie chart labels and percentage values
for text in texts:
    text.set_fontsize(15)
for autotext in autotexts:
    autotext.set_fontsize(15)

# Adjust chart formatting
plt.title('% Distribution of YouTube Channels by Country (2023)', fontsize = 25, weight = 'bold')
plt.tight_layout()
```



In [49]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 995 entries, 0 to 994
Data columns (total 27 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Youtuber                                995 non-null    object
 1   Subscribers                             995 non-null    int64
 2   Video Views                             995 non-null    int64
 3   Category                                995 non-null    object
 4   Title                                   995 non-null    object
 5   Uploads                                 995 non-null    int64
 6   Country                                 995 non-null    object
 7   Abbreviation                            995 non-null    object
 8   Channel_Type                            995 non-null    object
 9   Video_Views_Rank                        995 non-null    int64
 10  Country_Rank                            995 non-null    int64
 11  Channel_Type_Rank                       995 non-null    int64
 12  Video_Views_For_The_Last_30_Days        995 non-null    int64
 13  Lowest_Monthly_Earnings                 995 non-null    float64
 14  Highest_Monthly_Earnings                995 non-null    float64
 15  Lowest_Yearly_Earnings                  995 non-null    float64
 16  Highest_Yearly_Earnings                 995 non-null    float64
 17  Subscribers_For_Last_30_Days            995 non-null    int64
 18  Created_Year                            995 non-null    int64
 19  Created_Month                           995 non-null    object
 20  Created_Date                            995 non-null    float64
 21  Gross Tertiary Education Enrollment (%) 995 non-null    float64
 22  Population                              995 non-null    int64
 23  Unemployment Rate                       995 non-null    float64
 24  Urban_Population                        995 non-null    int64
 25  Latitude                                995 non-null    float64
 26  Longitude                               995 non-null    float64
dtypes: float64(9), int64(11), object(7)
memory usage: 210.0+ KB
```

In [50]:
```
!jt -t chesterish
```

In [51]:
```
df_mns = df

# Create new columns to support visualisation
df_mns['Subscribers (mn.)'] = (df_mns['Subscribers'] / 1000000).astype(int)
df_mns['Video Views (bn.)'] = (df_mns['Video Views'] / 1000000000).astype(int)
df_mns['Uploads (k.)'] = (df_mns['Uploads'] / 1000)

# Create new subset for scatter plot
df_sub_pop = df_mns[['Subscribers (mn.)', 'Video Views (bn.)']]
df_sub_pop
```

Out[51]:

	Subscribers (mn.)	Video Views (bn.)
0	245	228
1	170	0
2	166	28
3	162	164
4	159	148
...	...	...
990	12	9
991	12	1
992	12	2
993	12	0
994	12	2

995 rows × 2 columns

In [53]:
```
df.info()
```

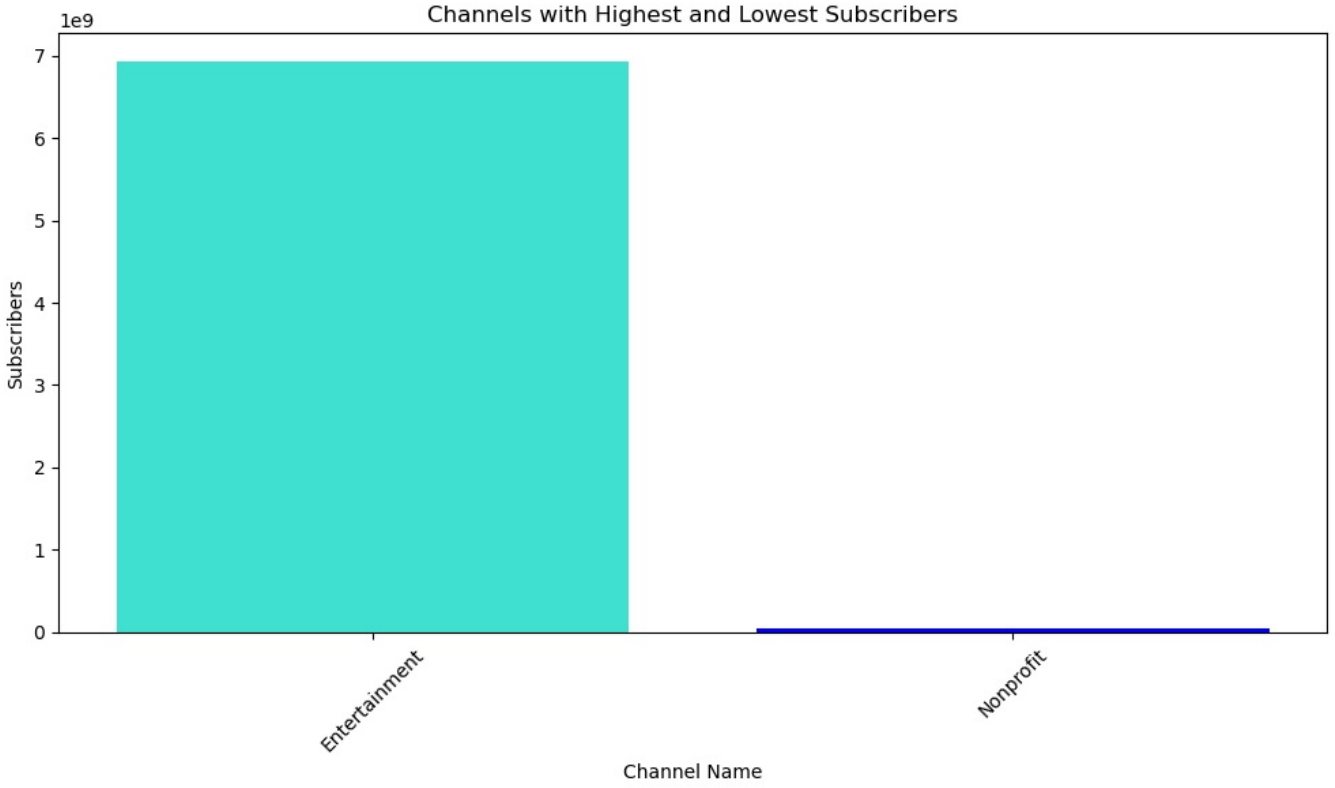
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 995 entries, 0 to 994
Data columns (total 30 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   Youtuber                                995 non-null    object 
 1   Subscribers                             995 non-null    int64  
 2   Video Views                             995 non-null    int64  
 3   Category                                995 non-null    object 
 4   Title                                   995 non-null    object 
 5   Uploads                                 995 non-null    int64  
 6   Country                                 995 non-null    object 
 7   Abbreviation                            995 non-null    object 
 8   Channel_Type                            995 non-null    object 
 9   Video_Views_Rank                        995 non-null    int64  
 10  Country_Rank                            995 non-null    int64  
 11  Channel_Type_Rank                       995 non-null    int64  
 12  Video_Views_For_The_Last_30_Days        995 non-null    int64  
 13  Lowest_Monthly_Earnings                 995 non-null    float64
 14  Highest_Monthly_Earnings                995 non-null    float64
 15  Lowest_Yearly_Earnings                  995 non-null    float64
 16  Highest_Yearly_Earnings                 995 non-null    float64
 17  Subscribers_For_Last_30_Days            995 non-null    int64  
 18  Created_Year                            995 non-null    int64  
 19  Created_Month                           995 non-null    object 
 20  Created_Date                            995 non-null    float64
 21  Gross Tertiary Education Enrollment (%) 995 non-null    float64
 22  Population                              995 non-null    int64  
 23  Unemployment Rate                       995 non-null    float64
 24  Urban_Population                        995 non-null    int64  
 25  Latitude                                995 non-null    float64
 26  Longitude                               995 non-null    float64
 27  Subscribers (mn.)                       995 non-null    int32  
 28  Video Views (bn.)                       995 non-null    int32  
 29  Uploads (k.)                            995 non-null    float64
dtypes: float64(10), int32(2), int64(11), object(7)
memory usage: 225.6+ KB
```

In [54]:
```
distinct_country_count = df['Country'].nunique()
print(f"There are {distinct_country_count} distinct countries in the dataset.")
```

```
There are 50 distinct countries in the dataset.
```

In [55]:
```
data = {'Channel Name': ['Animals ', 'Autos', 'Comedy', 'Education','Entertainment','Film','Games',
                         'Howto','Music','News','Nonprofit','People','Sports','Tech ','Unknown'],
        'Subscribers': [58600000, 64400000, 1063700000, 1300200000,6922500000,964400000,2111600000,
                        649100000,5771500000,611700000,55500000,1953300000,361500000,320800000,658700000]}
df = pd.DataFrame(data)

# Find the channel with the highest and lowest number of subscribers
max_subscribers = df[df['Subscribers'] == df['Subscribers'].max()]
min_subscribers = df[df['Subscribers'] == df['Subscribers'].min()]

combined_data = pd.concat([max_subscribers, min_subscribers])

# Create a bar chart
plt.figure(figsize=(10, 6))
plt.bar(combined_data['Channel Name'], combined_data['Subscribers'], color=['turquoise', 'blue'])
plt.xlabel('Channel Name')
plt.ylabel('Subscribers')
plt.title('Channels with Highest and Lowest Subscribers')
plt.xticks(rotation=45)
plt.tight_layout()
```

```
# Display the chart
plt.show()
```



In [56]:
```
selected_columns = ['Subscribers']

# Filter df for top 10 most subscribed YouTube channels
top_10_subscribers = df.loc[0:10, selected_columns]

# Sort values so that highest output is descending, this is for the horizontal bar chart
top_10_subscriber_desc = top_10_channels.sort_values(by = 'Subscribers', ascending = False)

# Validate filtered df output
print(top_10_channels_desc)
```

```
                    Youtuber  Subscribers (mn.)
9               Vlad and Niki                 98
8                 Like Nastya                106
7                   PewDiePie                111
6              Kids Diana Show                112
5                        Music                119
4                    SET India                159
3  Cocomelon - Nursery Rhymes                162
2                     MrBeast                166
1               YouTube Movies                170
0                     T-Series                245
```

In [57]:
```
!pip install geopy
```

```
Requirement already satisfied: geopy in c:\users\nandhini\anaconda3\lib\site-packages (2.4.0)
Requirement already satisfied: geographiclib<3,>=1.52 in c:\users\nandhini\anaconda3\lib\site-packages (from ge
opy) (2.0)
```

In [58]:
```
data = pd.DataFrame({
    'Country': ['India', 'United States', 'United States', 'United States', 'India', 'Unknown',
                'United States', 'Japan', 'Russia', 'United States', 'India'],
    'Population': [1366417754, 328239523, 328239523, 328239523, 1366417754, 0,
                   328239523, 126226568, 144373535, 328239523, 1366417754],
    'Subscribers': [245000000, 170000000, 166000000, 162000000, 159000000, 119000000,
                    112000000, 111000000, 106000000, 98900000, 96700000]
})

# Display data as a table
print(f"Highest Subscribers with Population")
print(data)
```

```
Highest Subscribers with Population
          Country  Population  Subscribers
0           India  1366417754    245000000
1   United States   328239523    170000000
2   United States   328239523    166000000
3   United States   328239523    162000000
4           India  1366417754    159000000
5         Unknown           0    119000000
6   United States   328239523    112000000
7           Japan   126226568    111000000
8          Russia   144373535    106000000
9   United States   328239523     98900000
10          India  1366417754     96700000
```

In [59]:

```python
import folium

# Create a map centered at a specific location
m = folium.Map(location=[20, 0], zoom_start=2)

# Define the locations and their coordinates with country names and population count
locations = {
    'India': (20.5937, 78.9629, 'India', 1366417754),
    'United States': (37.7749, -122.4194, 'United States', 328239523),
    'Japan': (36.2048, 138.2529, 'Japan', 126226568),
    'Russia': (61.5240, 105.3188, 'Russia', 144373535)
}

 add markers to the map
ems():
lation = coords

de),
Population: {population}",
en')
```

-> Trust Notebook