Youtube Analysis

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [4]:
```
Youtube_data=pd.read_csv("F:\most_subscribed_youtube_channels.csv")
```

In [5]:
```
Youtube_data
```

Out[5]:

	rank	Youtuber	subscribers	video views	video count	category	started
0	1	T-Series	222,000,000	198,459,090,822	17,317	Music	2006
1	2	YouTube Movies	154,000,000	0	0	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	140,000,000	135,481,339,848	786	Education	2006
3	4	SET India	139,000,000	125,764,252,686	91,271	Shows	2006
4	5	Music	116,000,000	0	0	NaN	2013
...	...	...	...	...	...	...	...
995	996	JP Plays	10,900,000	4,609,300,218	3,528	Gaming	2014
996	997	TrapMusicHDTV	10,900,000	4,070,521,973	690	Music	2013
997	998	Games EduUu	10,900,000	3,093,784,767	1,006	Gaming	2011
998	999	Hueva	10,900,000	3,040,301,750	831	Gaming	2012
999	1000	Dobre Brothers	10,900,000	2,808,411,693	590	People & Blogs	2017

1000 rows × 7 columns

In [6]:
```
Youtube_data.head(5) # top 5 Data
```

Out[6]:

	rank	Youtuber	subscribers	video views	video count	category	started
0	1	T-Series	222,000,000	198,459,090,822	17,317	Music	2006
1	2	YouTube Movies	154,000,000	0	0	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	140,000,000	135,481,339,848	786	Education	2006
3	4	SET India	139,000,000	125,764,252,686	91,271	Shows	2006
4	5	Music	116,000,000	0	0	NaN	2013

In [7]:
```
Youtube_data.tail(5) # Bottom 5 data
```

Out[7]:

	rank	Youtuber	subscribers	video views	video count	category	started
995	996	JP Plays	10,900,000	4,609,300,218	3,528	Gaming	2014
996	997	TrapMusicHDTV	10,900,000	4,070,521,973	690	Music	2013
997	998	Games EduUu	10,900,000	3,093,784,767	1,006	Gaming	2011
998	999	Hueva	10,900,000	3,040,301,750	831	Gaming	2012
999	1000	Dobre Brothers	10,900,000	2,808,411,693	590	People & Blogs	2017

In [8]:
```
Youtube_data.shape # to check the shape of the data(rows,column)
```

Out[8]:
```
(1000, 7)
```

In [9]:
```
Youtube_data.columns # to check all the columns
```

Out[9]:
```
Index(['rank', 'Youtuber', 'subscribers', 'video views', 'video count',
       'category', 'started'],
      dtype='object')
```

In [10]:
```
Youtube_data.duplicated().sum() # to check duplicate value sum
```

Out[10]:
```
0
```

In [11]:
```
Youtube_data.isnull().sum() # to check the null value in the data
```

Out[11]:
```
rank            0
Youtuber        0
subscribers     0
video views     0
video count     0
category       27
started         0
dtype: int64
```

Category column have 27 null values present

In [12]:
```
Youtube_data.info() # to check the info of the data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   rank         1000 non-null   int64 
 1   Youtuber     1000 non-null   object
 2   subscribers  1000 non-null   object
 3   video views  1000 non-null   object
 4   video count  1000 non-null   object
 5   category     973 non-null    object
 6   started      1000 non-null   int64 
dtypes: int64(2), object(5)
memory usage: 54.8+ KB
```

In [13]:
```
Youtube_data.describe() # to check the Aggregate value of the data
```

Out[13]:

	rank	started
count	1000.000000	1000.000000
mean	500.500000	2012.376000
std	288.819436	3.998076
min	1.000000	1970.000000
25%	250.750000	2010.000000
50%	500.500000	2013.000000
75%	750.250000	2015.000000
max	1000.000000	2021.000000

In [14]:
```
Youtube_data.nunique() # to check the unique values in data
```

Out[14]:
```
rank           1000
Youtuber        999
subscribers     286
video views     991
video count     856
category         18
started          18
dtype: int64
```

In [15]:
```
Youtube_data.isnull().sum()
```

Out[15]:
```
rank            0
Youtuber        0
subscribers     0
video views     0
video count     0
category       27
started         0
dtype: int64
```

As in category there is so many null values so now we Replace the Null values with Unknown

In [16]:
```
Youtube_data['category'].fillna('Unknown',inplace=True)
```

In [17]:
```
Youtube_data['category'].unique() # these are unique value present in the category column
```

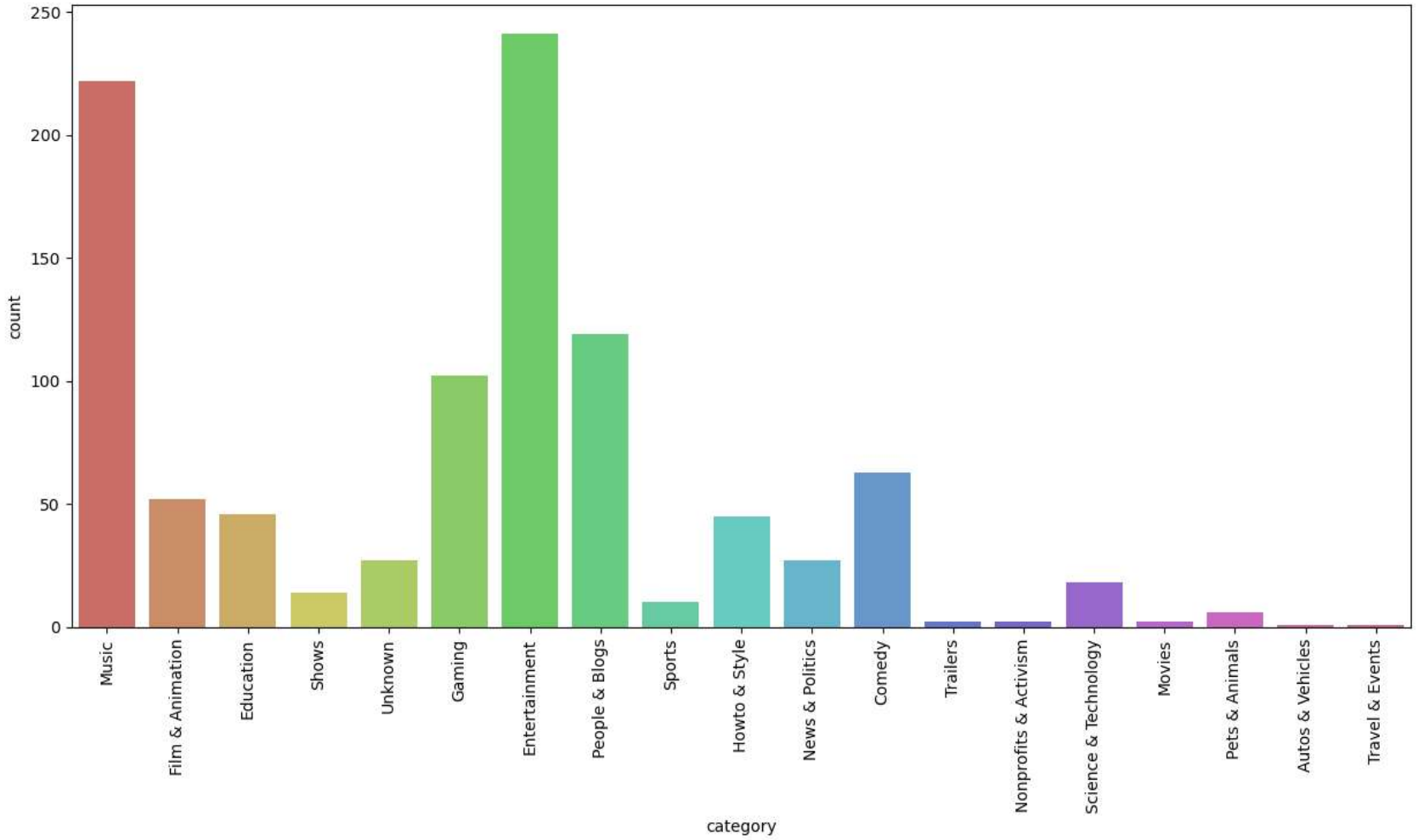
Out[17]:
```
array(['Music', 'Film & Animation', 'Education', 'Shows', 'Unknown',
       'Gaming', 'Entertainment', 'People & Blogs', 'Sports',
       'Howto & Style', 'News & Politics', 'Comedy', 'Trailers',
       'Nonprofits & Activism', 'Science & Technology', 'Movies',
       'Pets & Animals', 'Autos & Vehicles', 'Travel & Events'],
      dtype=object)
```

In [18]:
```
Youtube_data['category'].value_counts() # to count the unique value in the category column
```

Out[18]:
```
Entertainment           241
Music                   222
People & Blogs          119
Gaming                  102
Comedy                   63
Film & Animation         52
Education                46
Howto & Style            45
Unknown                  27
News & Politics          27
Science & Technology     18
Shows                    14
Sports                   10
Pets & Animals            6
Trailers                  2
Nonprofits & Activism     2
Movies                    2
Autos & Vehicles          1
Travel & Events           1
Name: category, dtype: int64
```

In [19]:
```
plt.figure(figsize=[15,7],)
print('countplot for category')
sns.countplot(Youtube_data['category'],data=Youtube_data,palette='hls')
plt.xticks(rotation=90)
plt.show()
```

```
countplot for category

C:\Users\em\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



From the above visualization we can say that maximum people watch Entertainment Channel followed by music channel

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

Out[20]:

	rank	Youtuber	subscribers	video views	video count	category	started
0	1	T-Series	222,000,000	198,459,090,822	17,317	Music	2006
1	2	YouTube Movies	154,000,000	0	0	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	140,000,000	135,481,339,848	786	Education	2006
3	4	SET India	139,000,000	125,764,252,686	91,271	Shows	2006
4	5	Music	116,000,000	0	0	Unknown	2013

In the video views and video count,subscribers columns the numbers are seperated by commas so we have to remove that

In [21]:
```
Youtube_data['video views']=Youtube_data['video views'].str.replace(',','')
```

In [22]:
```
Youtube_data['video count']=Youtube_data['video count'].str.replace(',','')
```

In [23]:
```
Youtube_data['subscribers']=Youtube_data['subscribers'].str.replace(',','')
```

In [24]:
```
Youtube_data.head()
```

Out[24]:

	rank	Youtuber	subscribers	video views	video count	category	started
0	1	T-Series	222000000	198459090822	17317	Music	2006
1	2	YouTube Movies	154000000	0	0	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	140000000	135481339848	786	Education	2006
3	4	SET India	139000000	125764252686	91271	Shows	2006
4	5	Music	116000000	0	0	Unknown	2013

It is reomved

In [25]:
```
Youtube_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   rank         1000 non-null   int64 
 1   Youtuber     1000 non-null   object
 2   subscribers  1000 non-null   object
 3   video views  1000 non-null   object
 4   video count  1000 non-null   object
 5   category     1000 non-null   object
 6   started      1000 non-null   int64 
dtypes: int64(2), object(5)
memory usage: 54.8+ KB
```

from the above result we can see that subscribers ,video views,video count are in string data type so we have to change the data type

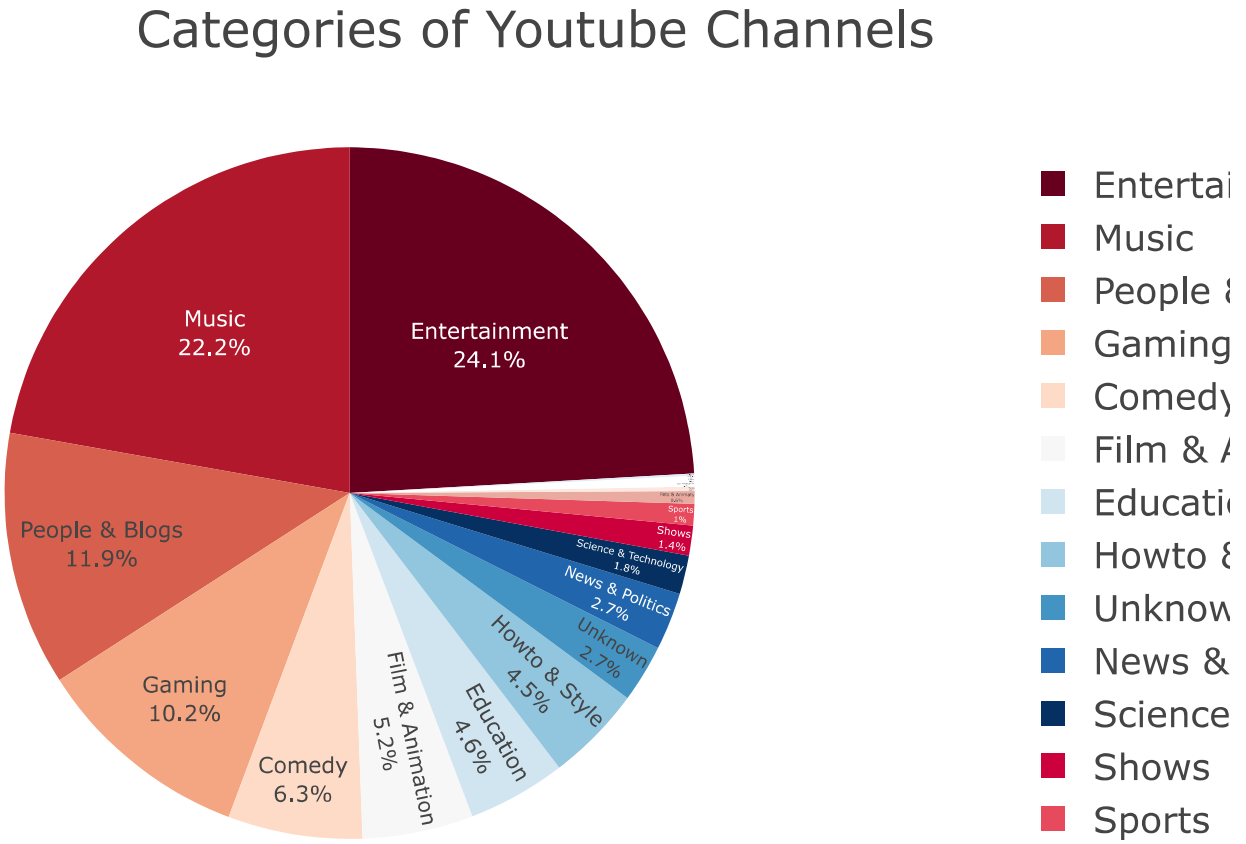
In [26]:
```
Youtube_data['subscribers']=Youtube_data['subscribers'].astype('int64')

Youtube_data['video views']=Youtube_data['video views'].astype('int64')

Youtube_data['video count']=Youtube_data['video count'].astype('int64')
```

In [27]:
```
Youtube_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   rank         1000 non-null   int64 
 1   Youtuber     1000 non-null   object
 2   subscribers  1000 non-null   int64 
 3   video views  1000 non-null   int64 
 4   video count  1000 non-null   int64 
 5   category     1000 non-null   object
 6   started      1000 non-null   int64 
dtypes: int64(5), object(2)
memory usage: 54.8+ KB
```

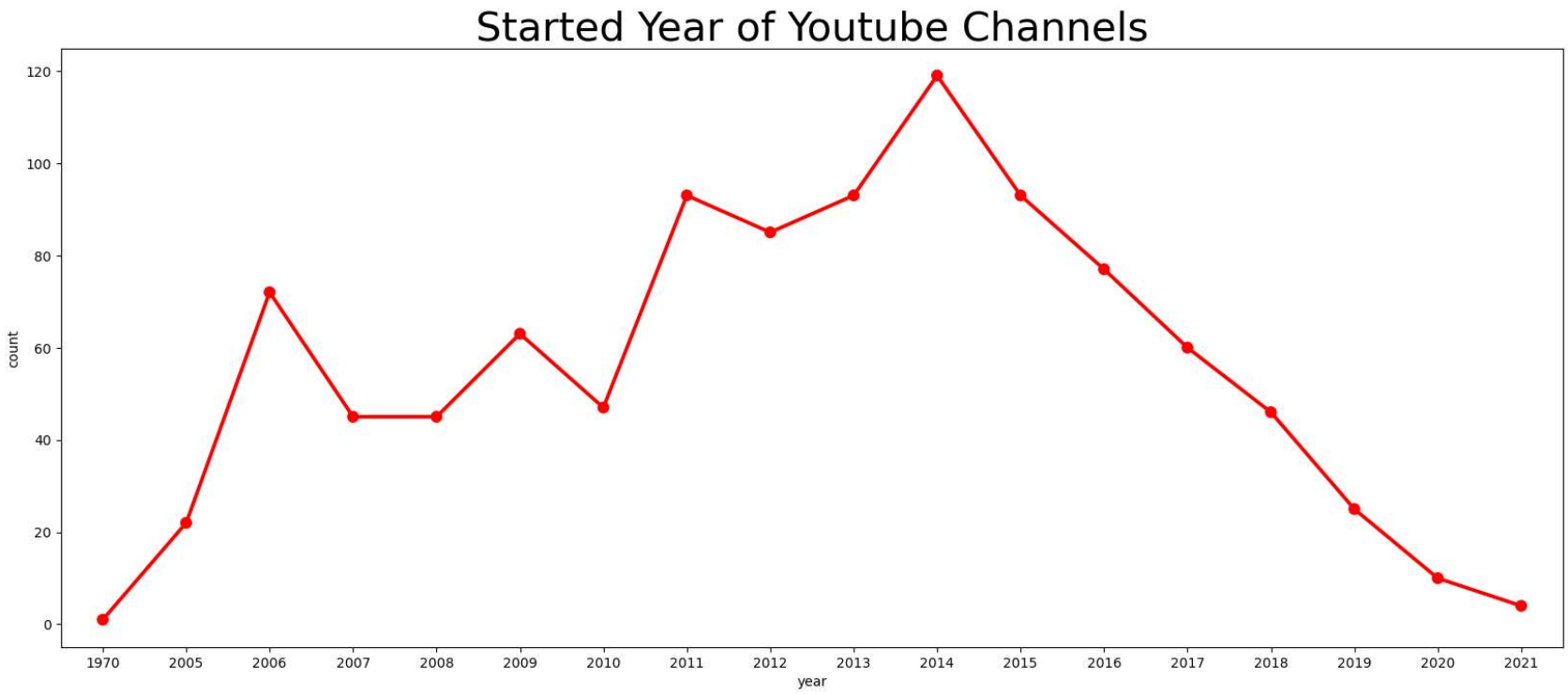
In [28]:
```
import plotly.express as px
```

In [30]:
```
categories=Youtube_data['category'].value_counts()
fig=px.pie(values=categories.values,
           names=categories.index,
           color_discrete_sequence=px.colors.sequential.RdBu,
           title='Categories of Youtube Channels',template='presentation')
fig.update_traces(textposition='inside',
                  textfont_size=11,
                  textinfo='percent+label')
fig.show()
```



In [31]:
```
year=Youtube_data['started'].value_counts()
plt.figure(figsize=(20,8))
sns.pointplot(x=year.index,y=year.values,color='red')
plt.xlabel('year')
plt.ylabel('count')
plt.title('Started Year of Youtube Channels',size=30,color='black')
```

Out[31]:
```
Text(0.5, 1.0, 'Started Year of Youtube Channels')
```

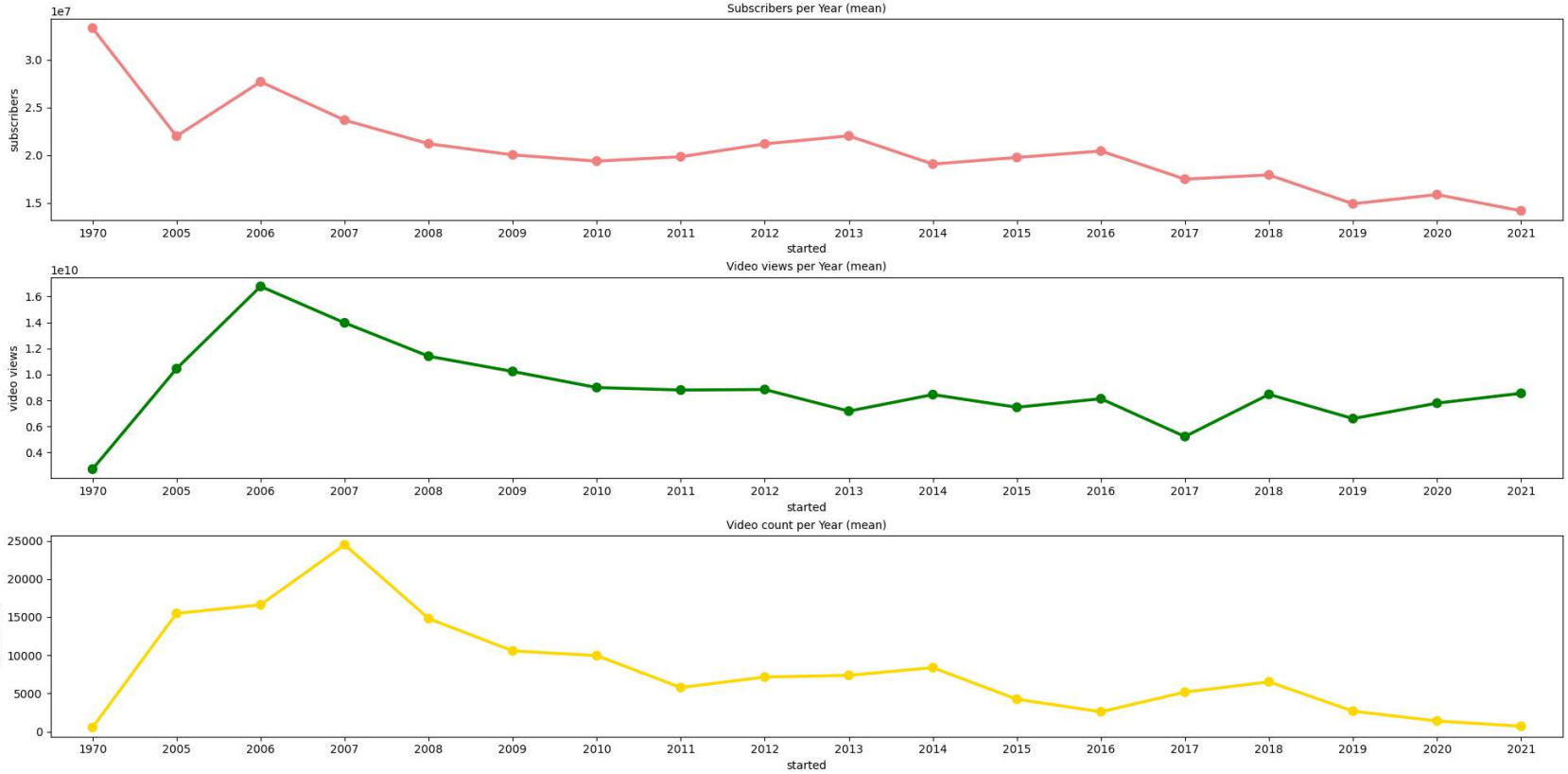


In [33]:
```
year_mean=Youtube_data.groupby('started').mean().reset_index()
year_mean
```

Out[33]:

	started	rank	subscribers	video views	video count
0	1970	100.000000	3.330000e+07	2.725287e+09	540.000000
1	2005	423.590909	2.197273e+07	1.044777e+10	15480.409091
2	2006	426.625000	2.767361e+07	1.676924e+10	16612.625000
3	2007	466.866667	2.365111e+07	1.396931e+10	24476.800000
4	2008	452.533333	2.118222e+07	1.140225e+10	14807.333333
5	2009	468.460317	2.001111e+07	1.023113e+10	10564.380952
6	2010	532.127660	1.935957e+07	8.997569e+09	9957.319149
7	2011	485.204301	1.981720e+07	8.804918e+09	5772.118280
8	2012	487.752941	2.116588e+07	8.844339e+09	7142.811765
9	2013	463.483871	2.200108e+07	7.183893e+09	7368.139785
10	2014	532.226891	1.904790e+07	8.453754e+09	8370.806723
11	2015	542.978495	1.974086e+07	7.481183e+09	4237.698925
12	2016	508.883117	2.041558e+07	8.136676e+09	2594.441558
13	2017	536.533333	1.746833e+07	5.230756e+09	5168.783333
14	2018	557.195652	1.791304e+07	8.470628e+09	6515.543478
15	2019	616.480000	1.488400e+07	6.609422e+09	2673.560000
16	2020	573.800000	1.585000e+07	7.795733e+09	1385.000000
17	2021	698.000000	1.415000e+07	8.552476e+09	696.750000

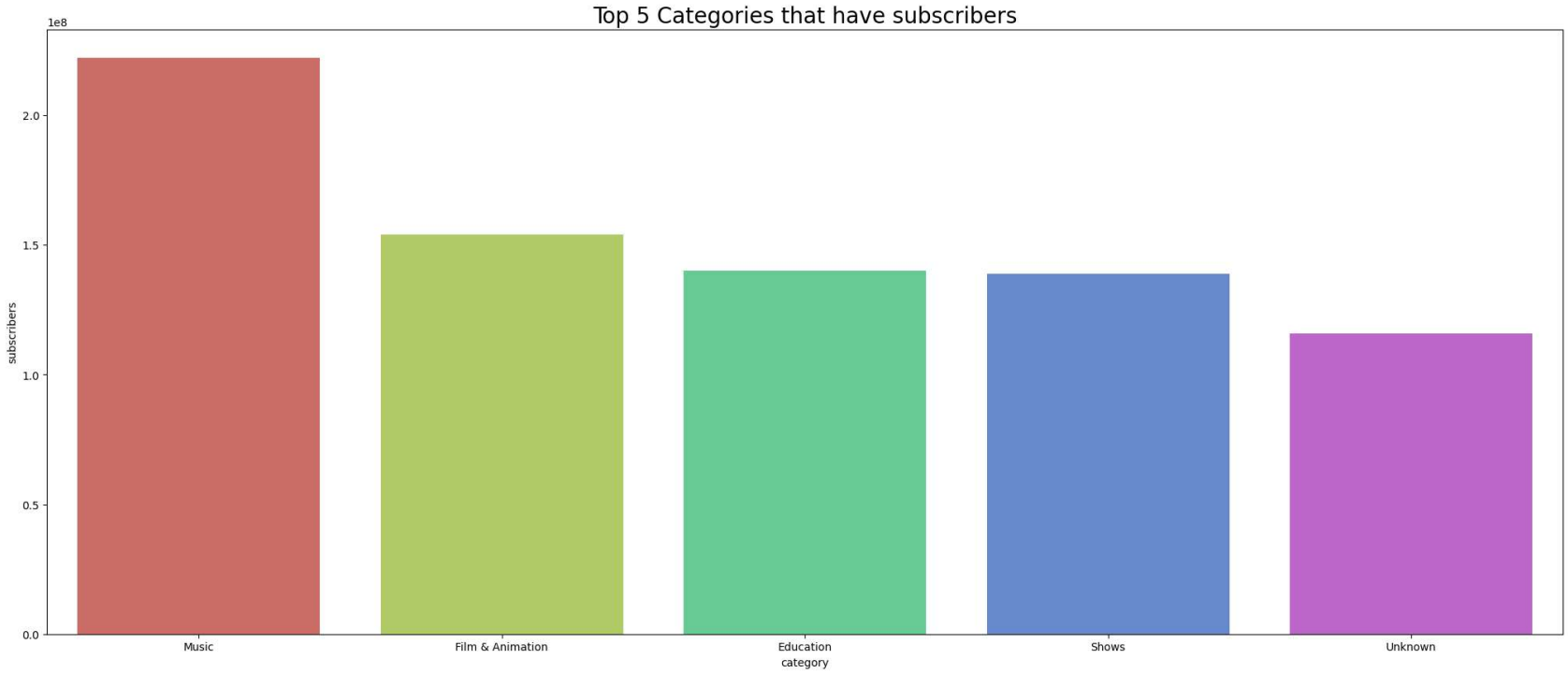
In [35]:
```
def pltplot(data, xcol, ycol, color, ax, title):
    sns.pointplot(data=data, x=xcol, y=ycol, color=color, ax=ax).set_title(title, size=10)
fig, ((ax1),(ax2),(ax3))=plt.subplots(ncols=1, nrows=3)
fig.set_size_inches(20,10)
fig.tight_layout(pad=3.0)
pltplot(year_mean,'started','subscribers','lightcoral', ax1,'Subscribers per Year (mean)')
pltplot(year_mean,'started','video views','green', ax2,'Video views per Year (mean)')
pltplot(year_mean,'started','video count','gold', ax3,'Video count per Year (mean)')
```



Top 5 Channel have maximum numbers of subscribers

In [37]:
```
subscribers=Youtube_data.sort_values('subscribers',ascending=False)
plt.figure(figsize=(25,10))
subscribers=subscribers[:5]
sns.barplot(x='category',y='subscribers',data=subscribers,palette='hls')
plt.title('Top 5 Categories that have subscribers',size=20)
```

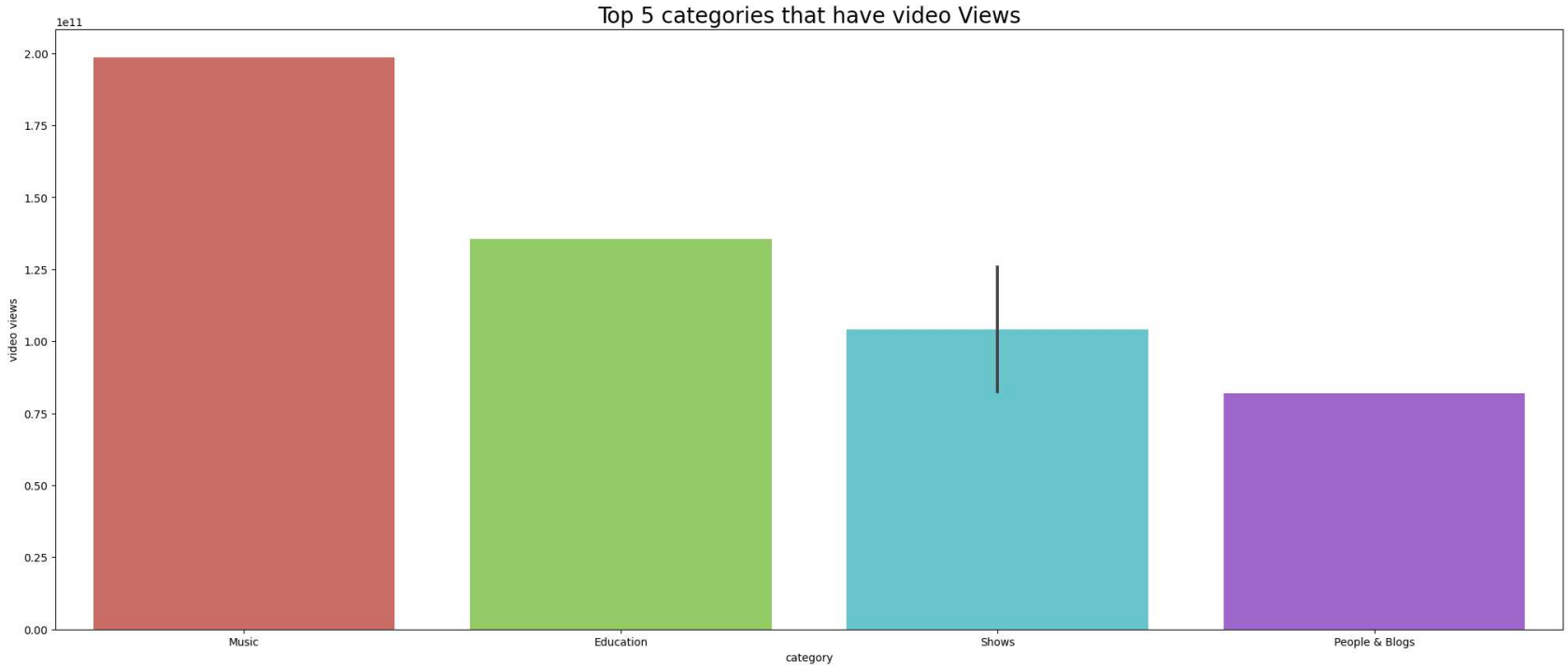
Out[37]:
```
Text(0.5, 1.0, 'Top 5 Categories that have subscribers')
```



Top 5 Channel have maximum numbers of video views

In [41]:
```
Videoviews=Youtube_data.sort_values('video views',ascending=False)
plt.figure(figsize=(25,10))
Videoviews=Videoviews[:5]
sns.barplot(x='category',y='video views',data=Videoviews,palette='hls')
plt.title('Top 5 categories that have video Views',size=20)
```

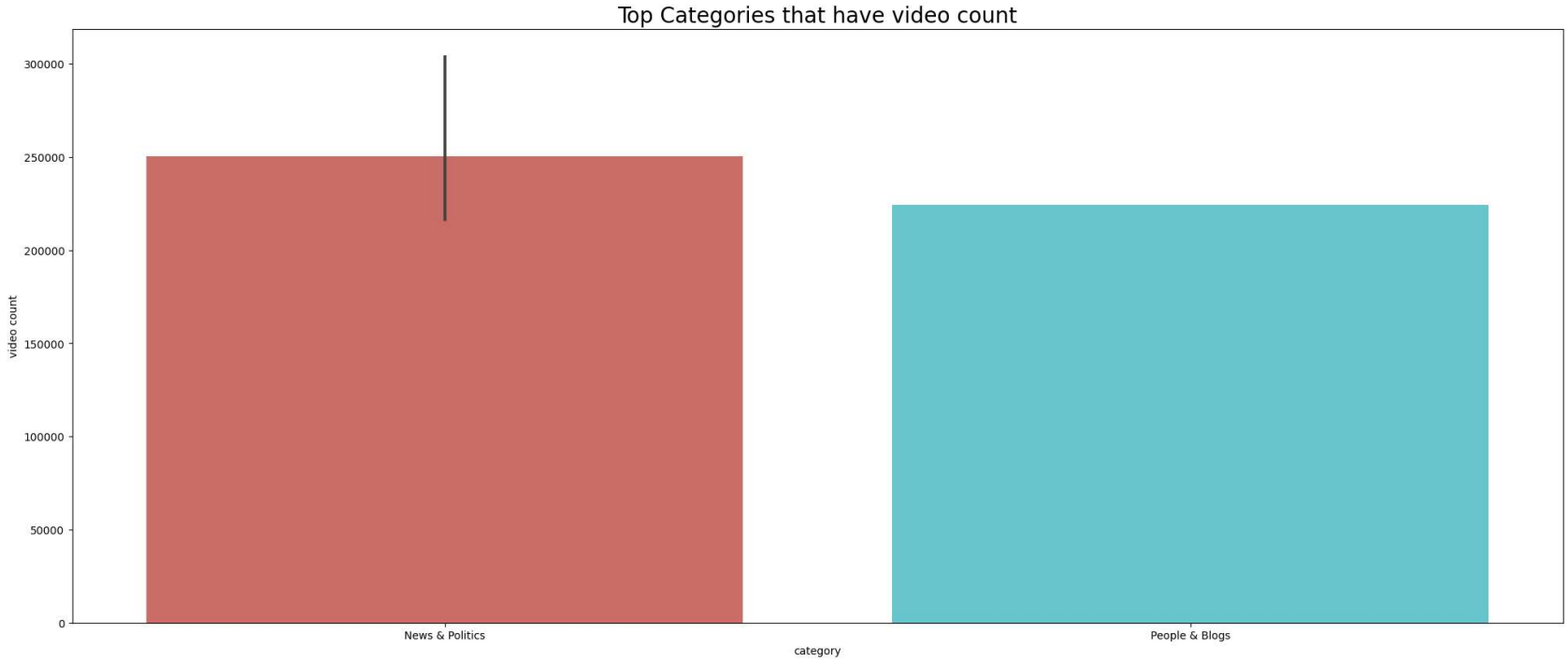
Out[41]:
```
Text(0.5, 1.0, 'Top 5 categories that have video Views')
```



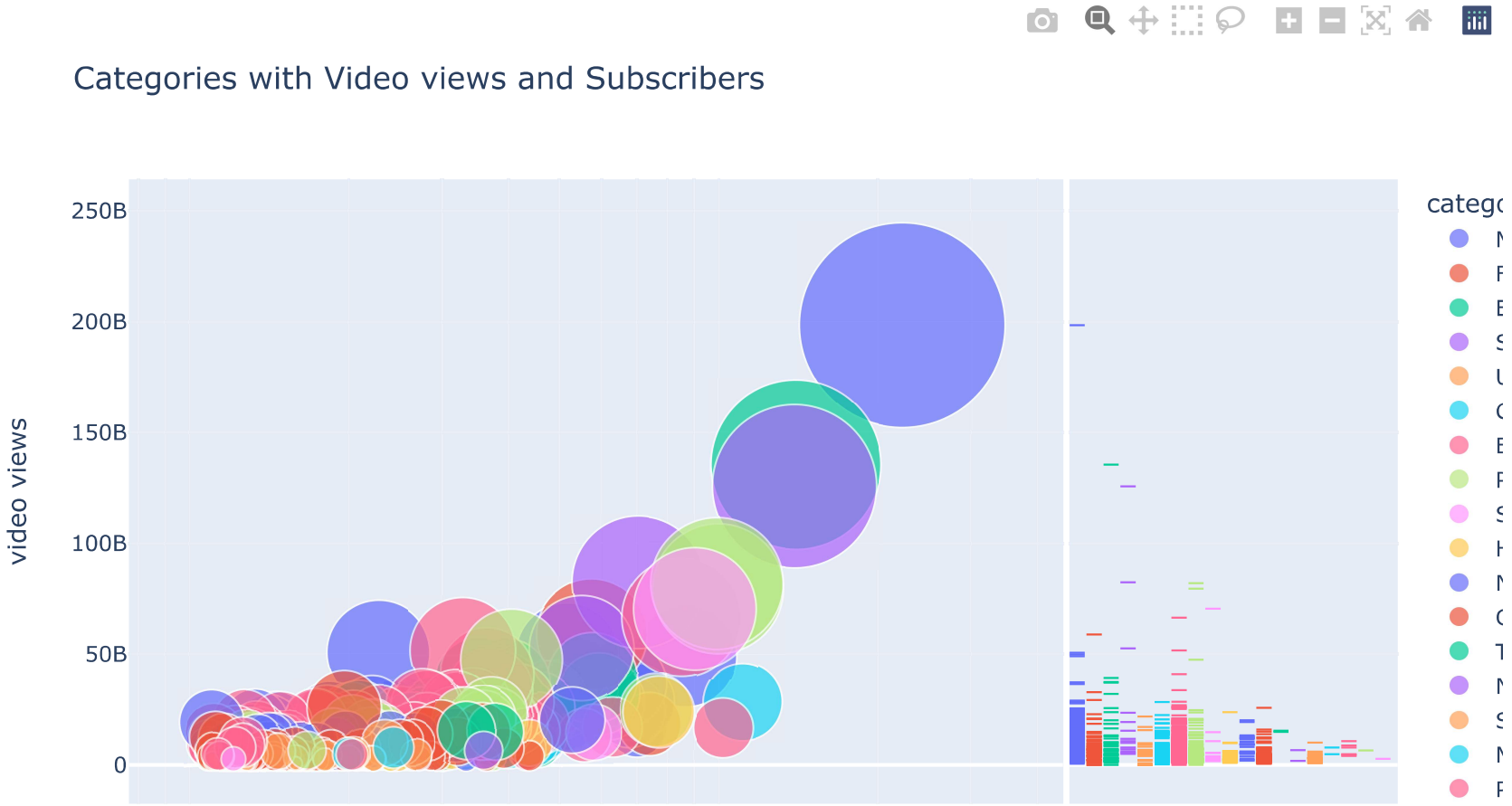
Top 5 categories that have video count

In [44]:
```
videocount=Youtube_data.sort_values('video count',ascending=False)
plt.figure(figsize=(25,10))
videocount=videocount[:5]
sns.barplot(x='category',y='video count',data=videocount,palette='hls')
plt.title('Top Categories that have video count',size=20)
```

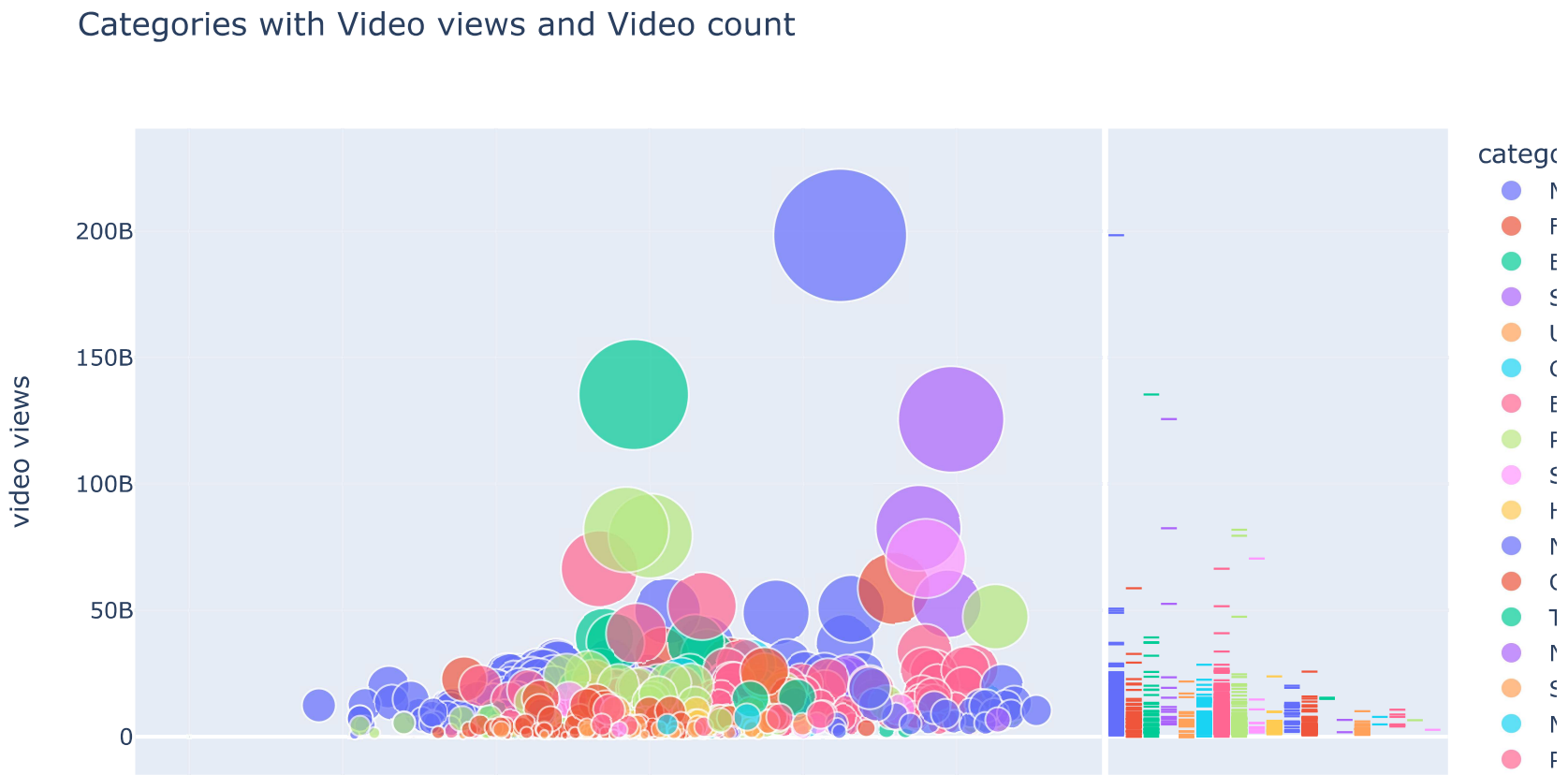
Out[44]:
```
Text(0.5, 1.0, 'Top Categories that have video count')
```



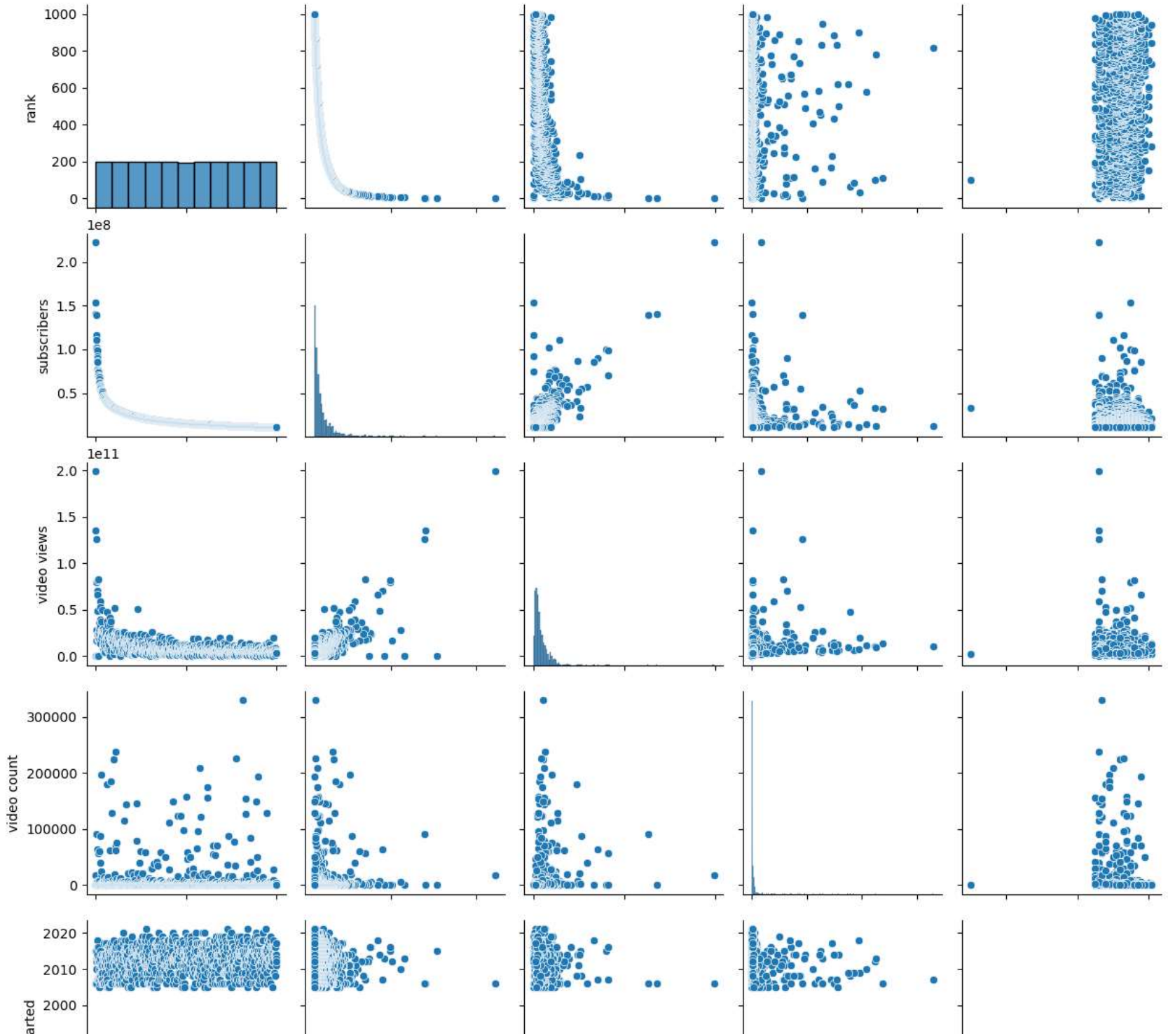
In [53]:
```
def plot(data, xcol, ycol, size, color,title):
    px.scatter(data, x=xcol,y=ycol,
               size=size, color=color,
               log_x=True, size_max=50).set_title(title,fontsize=20)
    axs.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
title1 = 'categories with video views and subscribers'
title2 = 'categories with video views and video counts'

fig = px.scatter(Youtube_data, x="subscribers", y="video views",
size="video views", color="category",
log_x=True, size_max=80,
title="Categories with Video views and Subscribers",
marginal_y='rug')
fig.show()
```

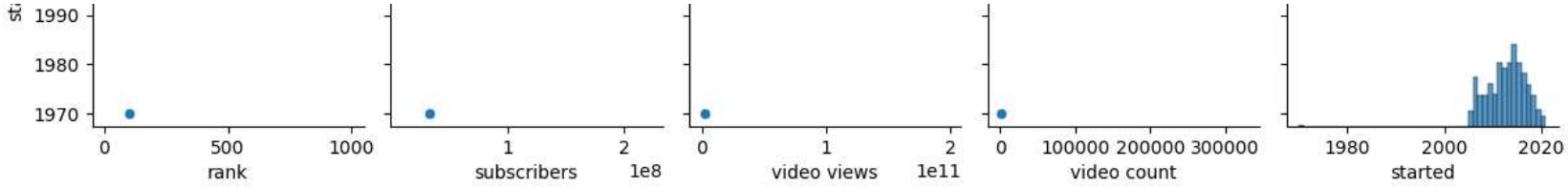

Categories with Video views and Subscribers
video views
250B
200B
150B
100B
50B
0
catego

In [49]:
```
fig = px.scatter(Youtube_data, x="video count", y="video views",
size="video views", color="category",
log_x=True, size_max=50,
title="Categories with Video views and Video count",
marginal_y='rug')
fig.show()
```

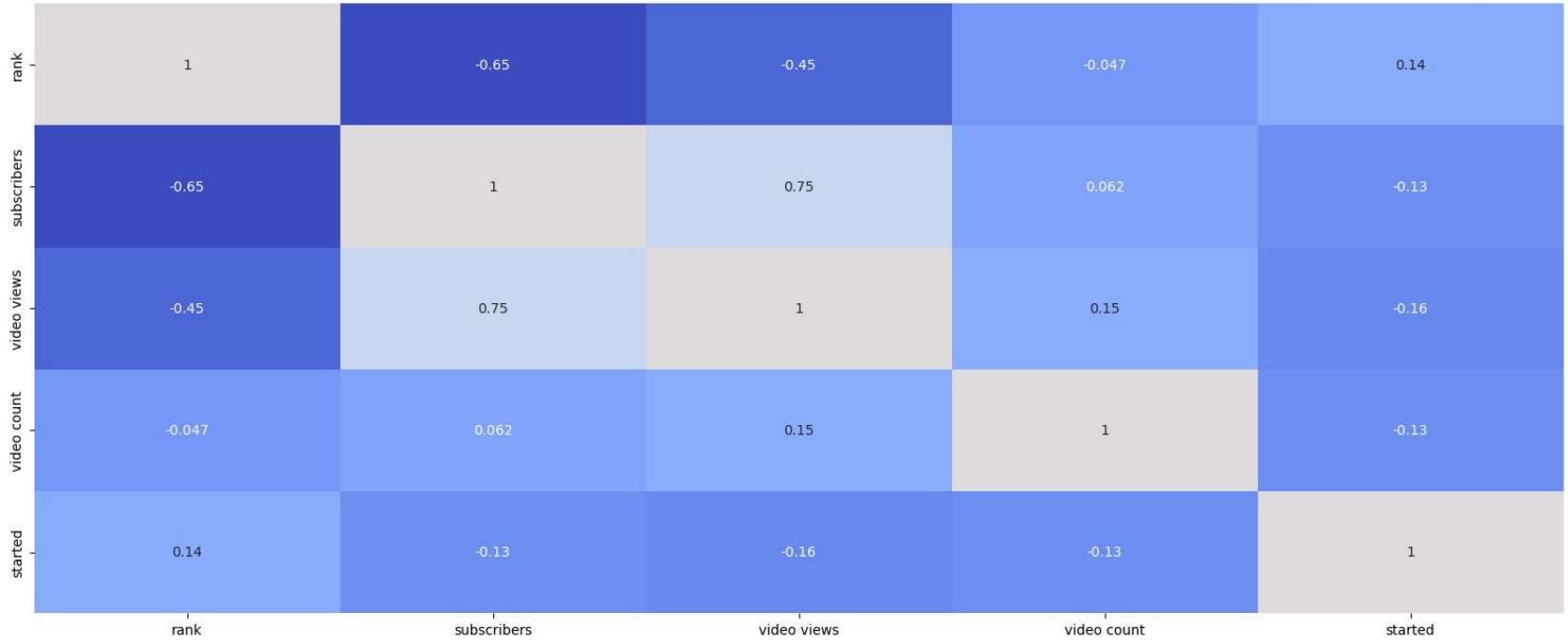


In [50]:
```
sns.pairplot(Youtube_data)
plt.show();
```


rank
1e8
subscribers
1e11
video views
video count
arted



In [51]:
```
plt.figure(figsize=(20,8))
sns.heatmap(Youtube_data.corr(), annot=True, center=True, cmap = 'coolwarm'
, cbar = False);
plt.show()
```



In [ ]: