HOTSTAR ANALYSIS

Import libraries

In [1]:
```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

In [2]:
```
os.chdir('D:\\Hack Challenge\\Project\\datasets')
```

In [3]:
```
hotstar = pd.read_csv('hotstar.csv')
```

In [4]:
```
hotstar.head(5)
```

Out[4]:

	imdb_id	title	plot	type	rated	year	released_at	added_at	runtime	genre	director	writer	actors	language	country	awards	metascore	imdb_rating	imdl
0	tt0147800	10 Things I Hate About You	A pretty, popular teenager can't go out on a d...	movie	PG-13	1999	31 Mar 1999	November 12, 2019	97 min	Comedy, Drama, Romance	Gil Junger	Karen McCullah, Kirsten Smith	Heath Ledger, Julia Stiles, Joseph Gordon-Levi...	English, French	USA	2 wins & 13 nominations.	70.0	7.3	
1	tt7019028	101 Dalmatian Street	This series follows the lives of Delilah and D...	series	NaN	2018–	25 Mar 2019	February 28, 2020	NaN	Animation, Comedy, Family	NaN	NaN	Josh Brener, Michaela Dietz, Bert Davis, Abiga...	English	UK, USA, Canada	NaN	NaN	6.2	
2	tt0115433	101 Dalmatians	An evil high-fashion designer plots to steal D...	movie	G	1996	27 Nov 1996	November 12, 2019	103 min	Adventure, Comedy, Crime, Family	Stephen Herek	Dodie Smith (novel), John Hughes (screenplay)	Glenn Close, Jeff Daniels, Joely Richardson, J...	English, Spanish	USA, UK	Nominated for 1 Golden Globe. Another 3 wins &...	49.0	5.7	
3	tt0324941	101 Dalmatians 2: Patch's London Adventure	Being one of 101 takes its toll on Patch, who ...	movie	G	2002	21 Jan 2003	November 12, 2019	74 min	Animation, Adventure, Comedy, Family, Musical	Jim Kammerud, Brian Smith	Jim Kammerud (story), Dan Root (story), Garret...	Barry Bostwick, Jason Alexander, Martin Short,...	English	USA	5 wins & 10 nominations.	NaN	5.8	
4	tt0211181	102 Dalmatians	Cruella DeVil gets out of prison and goes afte...	movie	G	2000	22 Nov 2000	November 12, 2019	100 min	Adventure, Comedy, Family	Kevin Lima	Dodie Smith (novel), Kristen Buckley (story), ...	Glenn Close, Gérard Depardieu, Ioan Gruffudd, ...	English	USA, UK	Nominated for 1 Oscar. Another 1 win & 7 nomin...	35.0	4.9	

In [5]:
```
hotstar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 992 entries, 0 to 991
Data columns (total 19 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   imdb_id      894 non-null    object
 1   title        894 non-null    object
 2   plot         866 non-null    object
 3   type         894 non-null    object
 4   rated        742 non-null    object
 5   year         894 non-null    object
 6   released_at  874 non-null    object
 7   added_at     992 non-null    object
 8   runtime      838 non-null    object
 9   genre        885 non-null    object
 10  director     689 non-null    object
 11  writer       743 non-null    object
 12  actors       870 non-null    object
 13  language     865 non-null    object
 14  country      869 non-null    object
 15  awards       556 non-null    object
 16  metascore    292 non-null    float64
 17  imdb_rating  879 non-null    float64
 18  imdb_votes   879 non-null    object
dtypes: float64(2), object(17)
memory usage: 147.4+ KB
```

In [6]:
```
hotstar.describe()
```

Out[6]:

	metascore	imdb_rating
count	292.000000	879.000000
mean	62.061644	6.656428
std	15.776455	1.020352
min	19.000000	1.500000
25%	51.000000	6.100000
50%	61.000000	6.700000
75%	73.000000	7.400000
max	99.000000	9.700000

Total movies, series and shows

In [7]:
```
movies = hotstar[hotstar['type'] == 'movie']
total_movies = movies['type'].count()
total_movies
```

Out[7]: 680

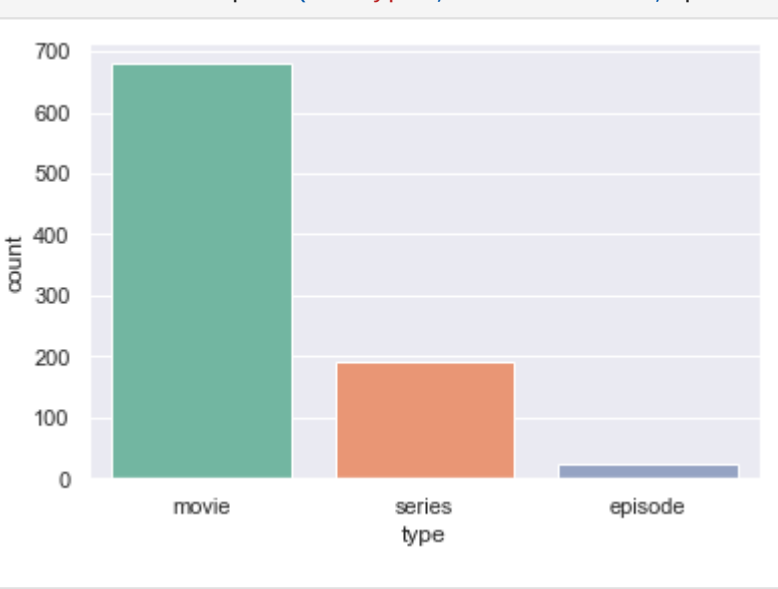
In [8]:
```
series = hotstar[hotstar['type'] == 'series']
total_series = series['type'].count()
total_series
```

Out[8]: 191

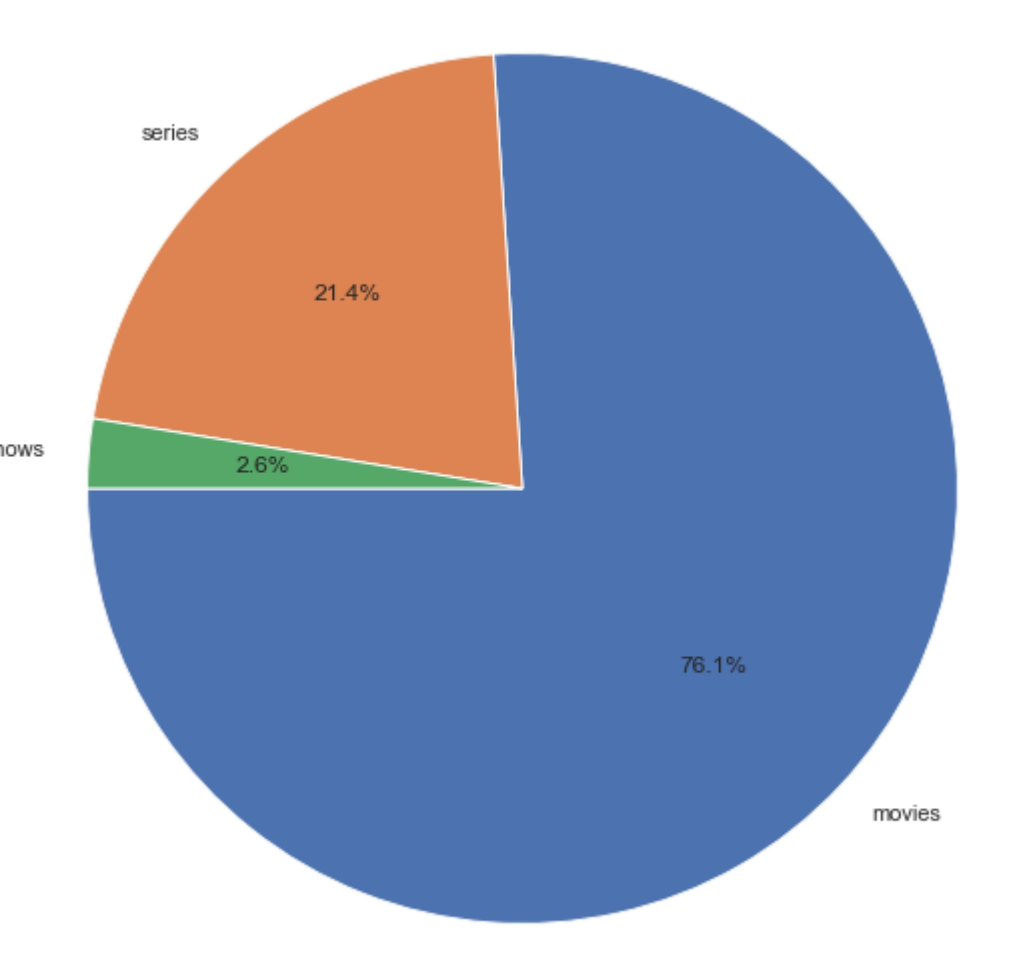
In [9]:
```
shows = hotstar[hotstar['type'] == 'episode']
total_shows = shows['type'].count()
total_shows
```

Out[9]: 23

In [10]:
```
sns.set(style='darkgrid')
ax = sns.countplot(x='type', data=hotstar, palette='Set2')
```

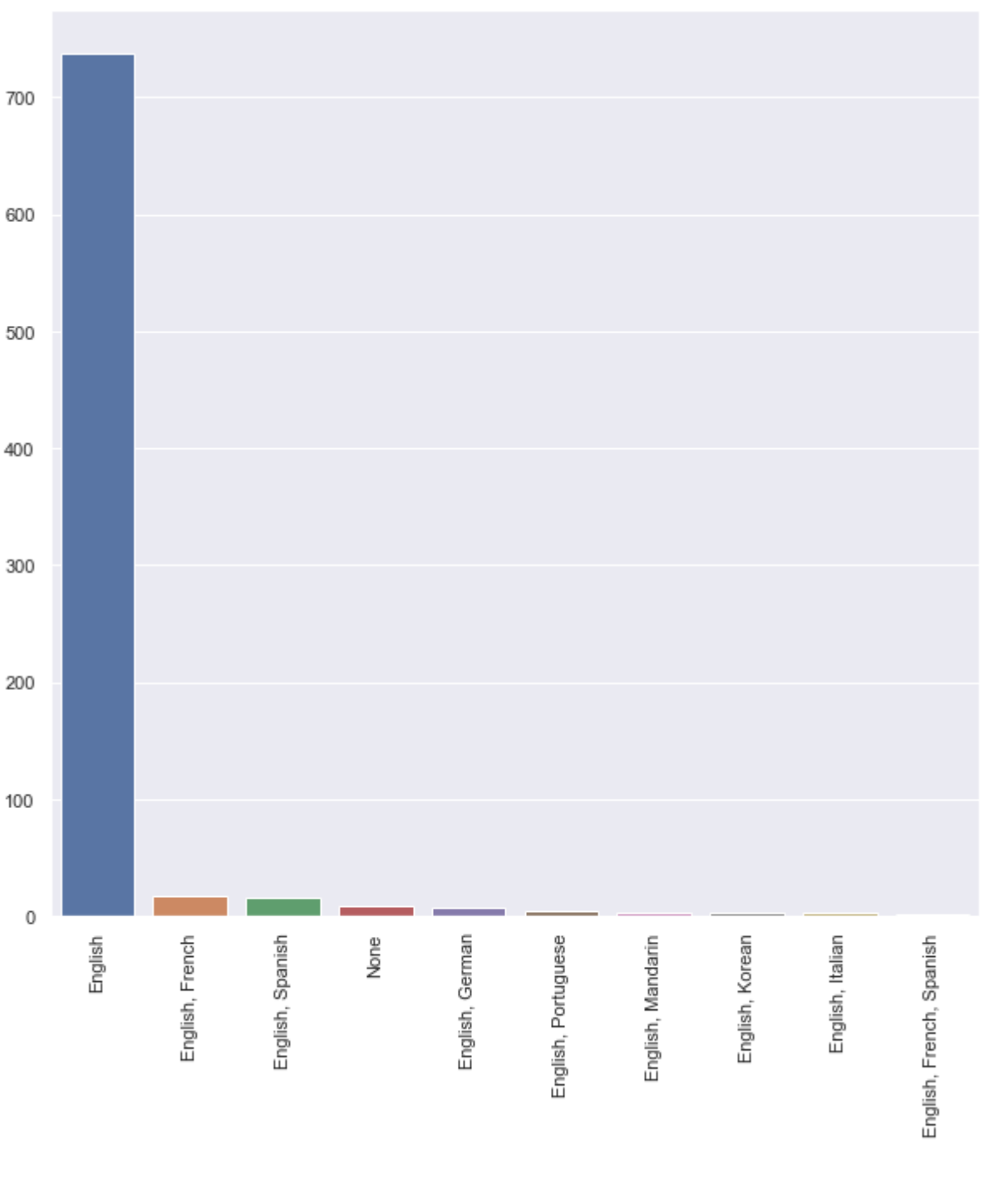


In [11]:
```
y = [total_movies, total_series, total_shows]
labels = ['movies', 'series', 'shows']
f, ax = plt.subplots(figsize=(10, 10))
ax.pie(y, labels=labels, autopct='%1.1f%%', startangle=180)
plt.show()
```



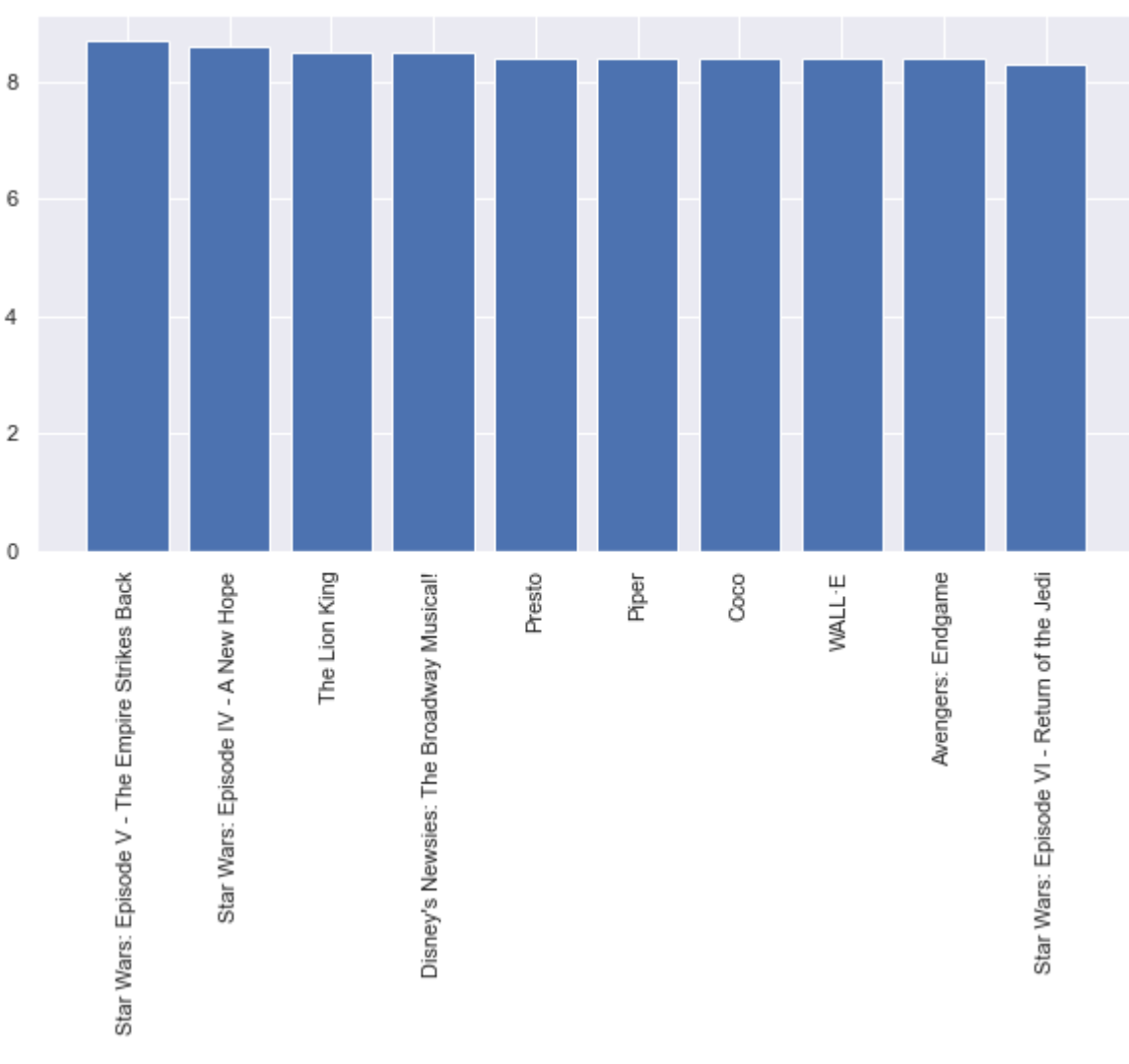
Top count Languages

In [12]:
```
Y = np.array(hotstar['language'].value_counts().values[:10])
labels = np.array(hotstar['language'].value_counts().index[:10])
plt.figure(figsize=(10, 10))
ax = sns.barplot(x=labels, y=Y)
ax.set_xticklabels(labels, rotation=90)
plt.show()
```



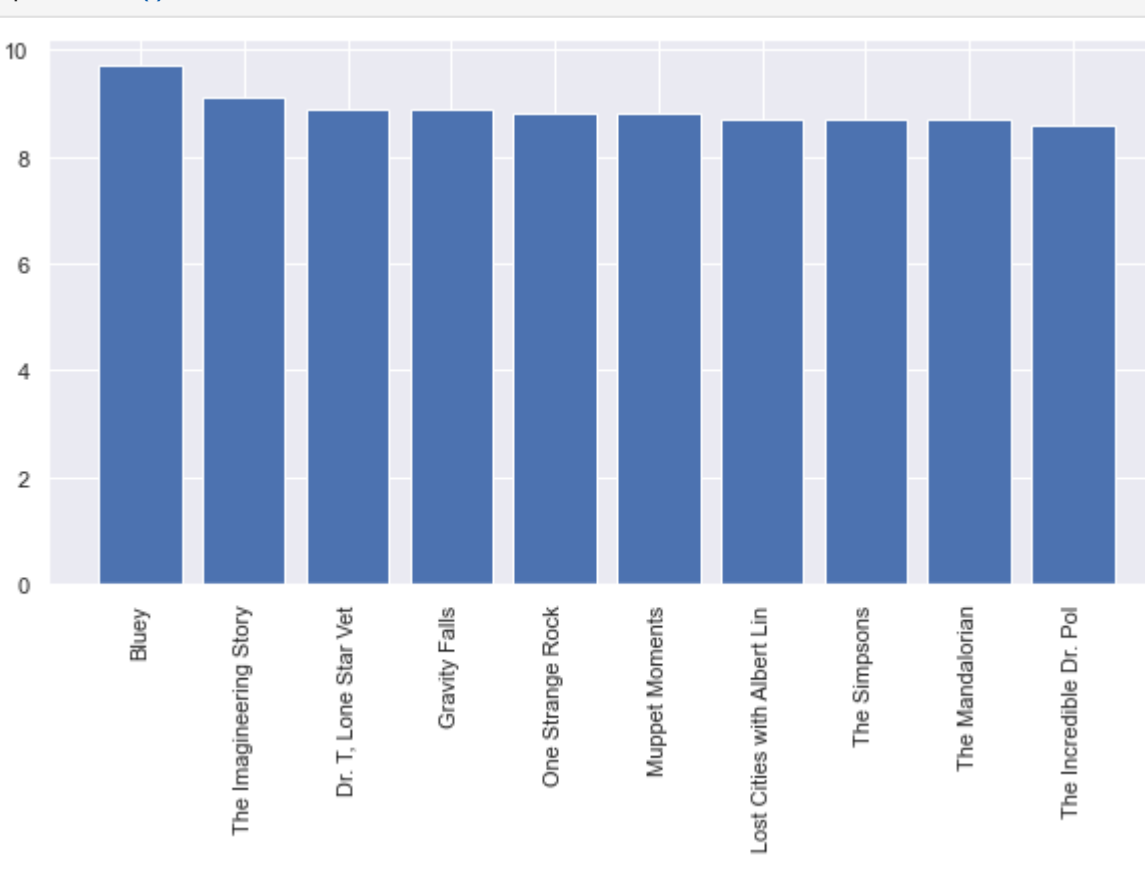
Top 10 movies

In [13]:
```
top_movies = movies.sort_values('imdb_rating',ascending=False).head(10)
plt.figure(figsize=(10, 5))
plt.bar(top_movies['title'], top_movies['imdb_rating'])
plt.xticks(rotation=90)
plt.show()
```



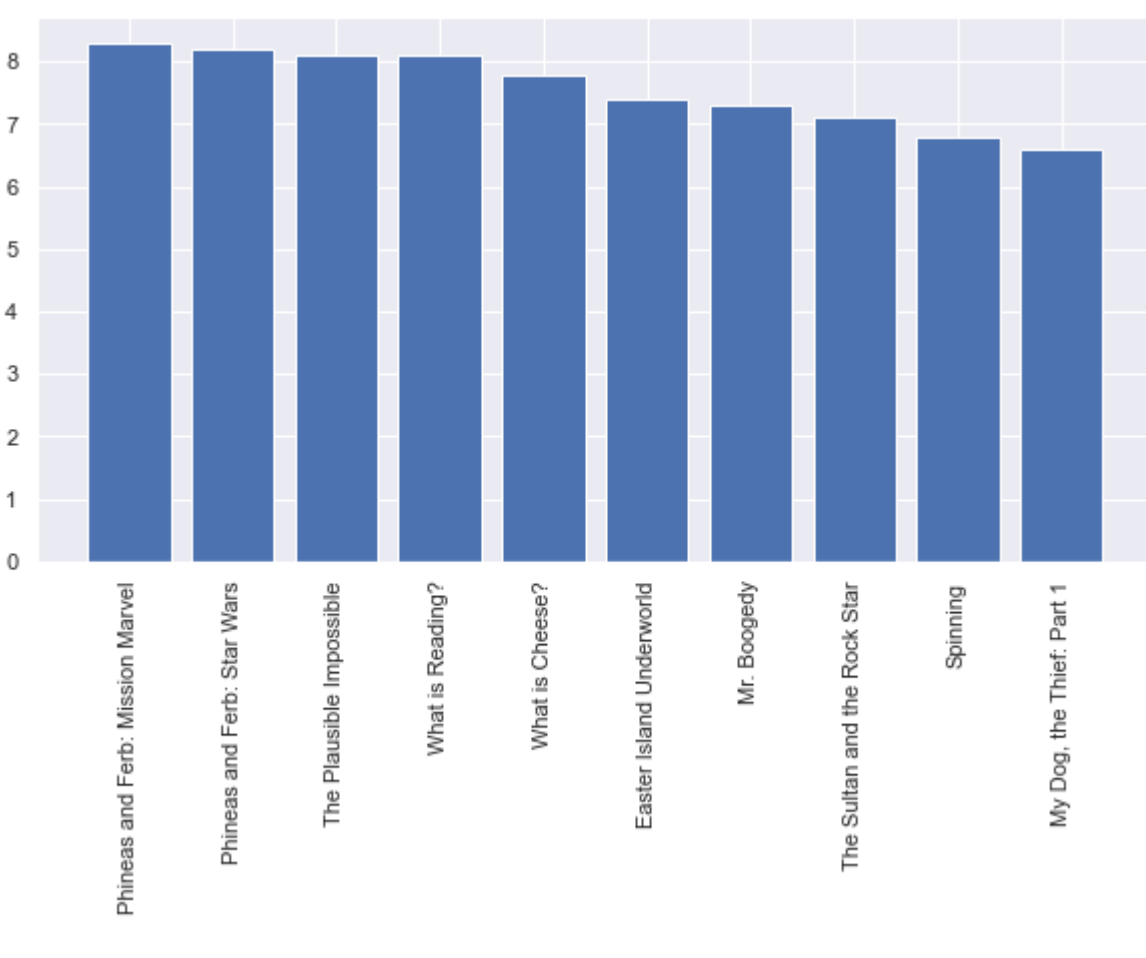
Top 10 series

In [14]:
```
top_series = series.sort_values('imdb_rating', ascending=False).head(10)
plt.figure(figsize=(10, 5))
plt.bar(top_series['title'], top_series['imdb_rating'])
plt.xticks(rotation=90)
plt.show()
```



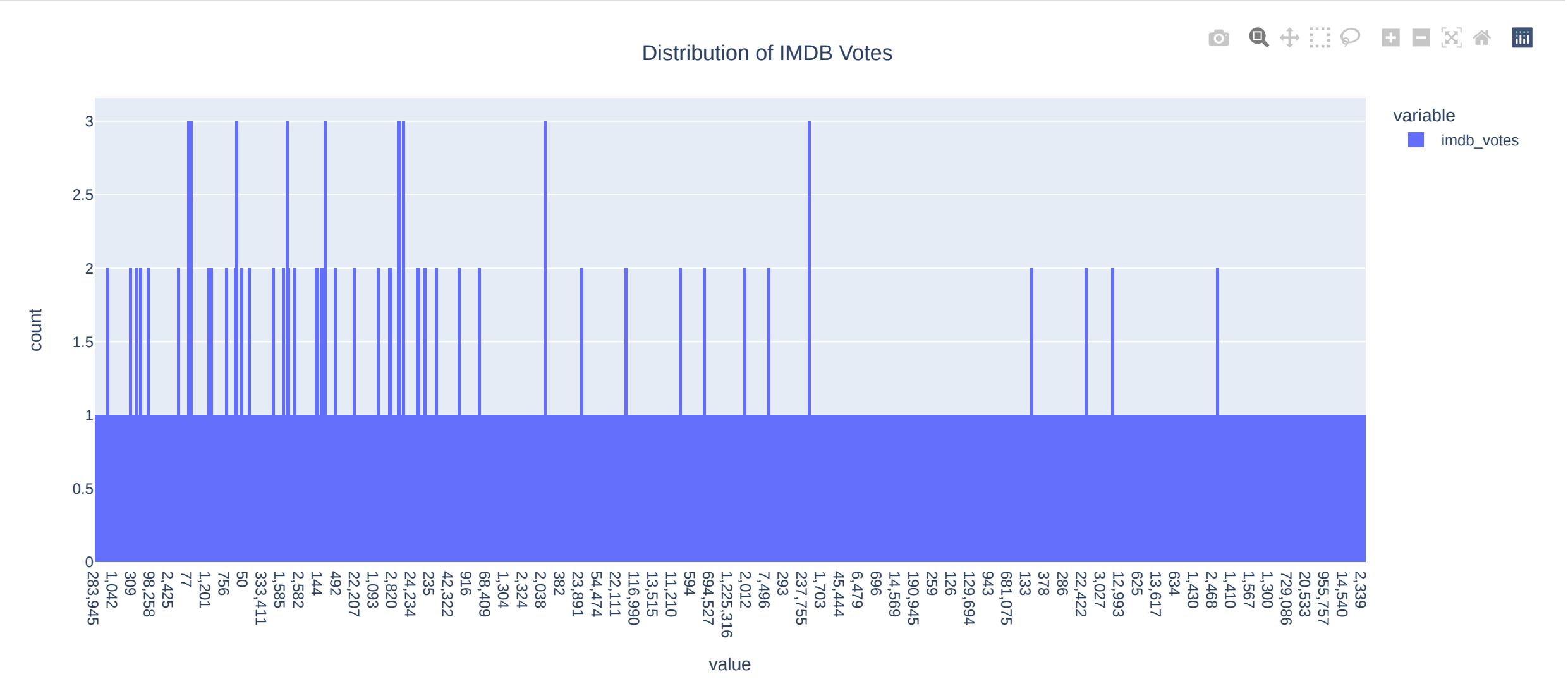
Top 10 Shows

In [15]:
```
top_shows = shows.sort_values('imdb_rating', ascending=False).head(10)
plt.figure(figsize=(10,5))
plt.bar(top_shows['title'], top_shows['imdb_rating'])
plt.xticks(rotation=90)
plt.show()
```



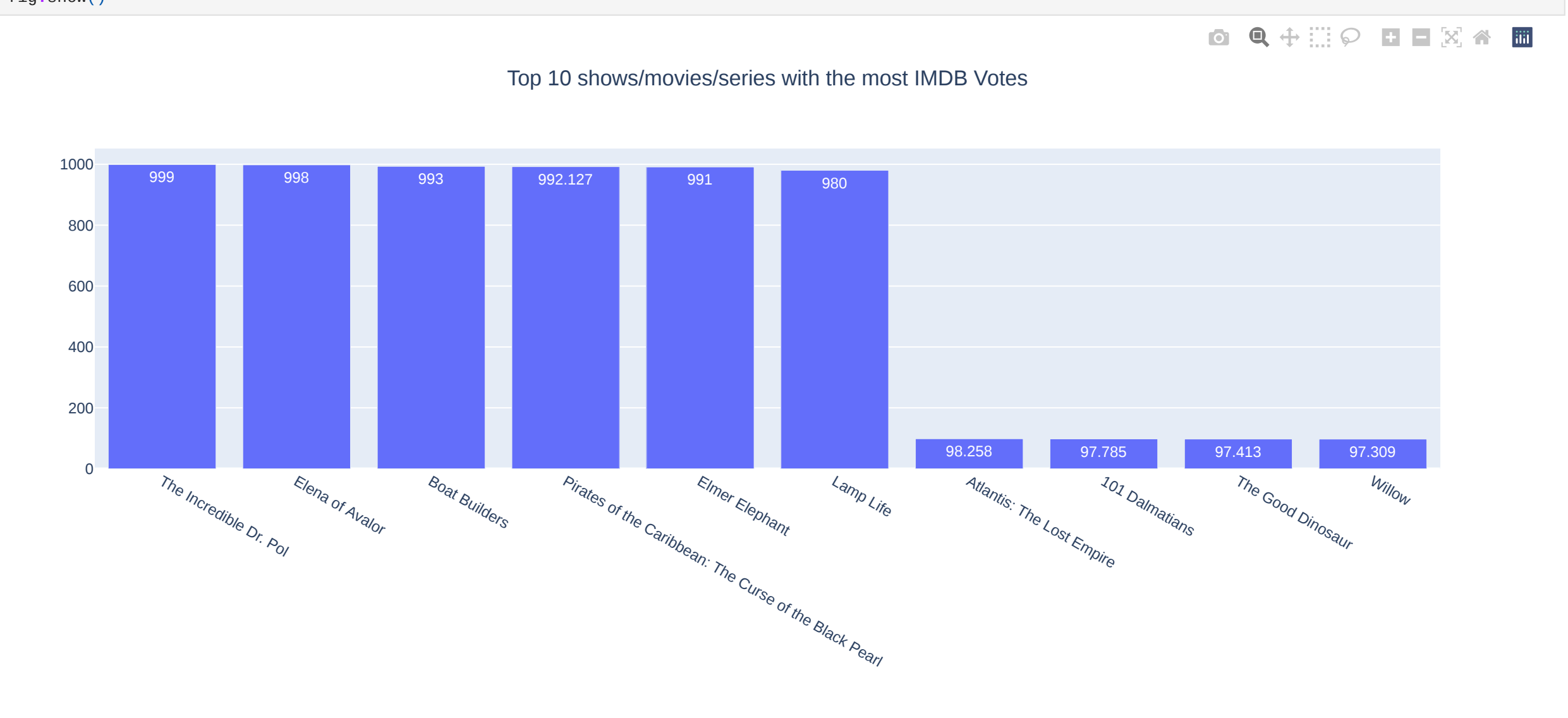
Distribution Of IMDB votes

In [16]:
```
fig = px.histogram(hotstar['imdb_votes'])
fig.update_layout(title='Distribution of IMDB Votes', title_x=0.5)
```



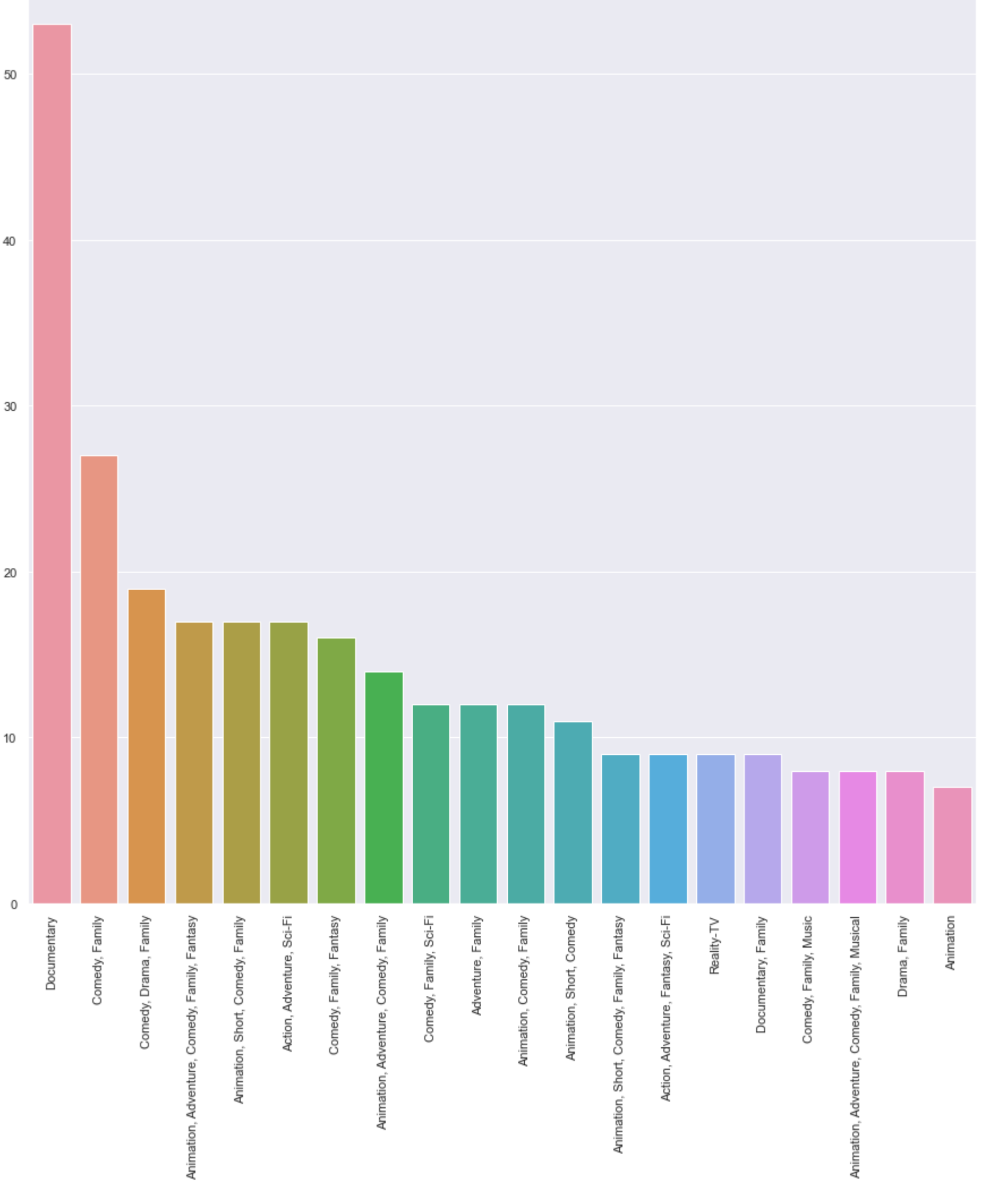
Top 10 movies, series and shows based on IMDB votes

In [17]:
```
import plotly.graph_objects as go
df_top10_v = hotstar.sort_values('imdb_votes',ascending=False).head(10)
df_top10_v['imdb_votes'] = df_top10_v['imdb_votes'].apply(lambda x: x.replace(',','.')).astype(float)

fig = go.Figure(data=[go.Bar(
            x=df_top10_v['title'], y=df_top10_v['imdb_votes'],
            text=df_top10_v['imdb_votes'],
            textposition='auto',
        )])

fig.update_layout(title='Top 10 shows/movies/series with the most IMDB Votes',title_x=0.5)
fig.show()
```



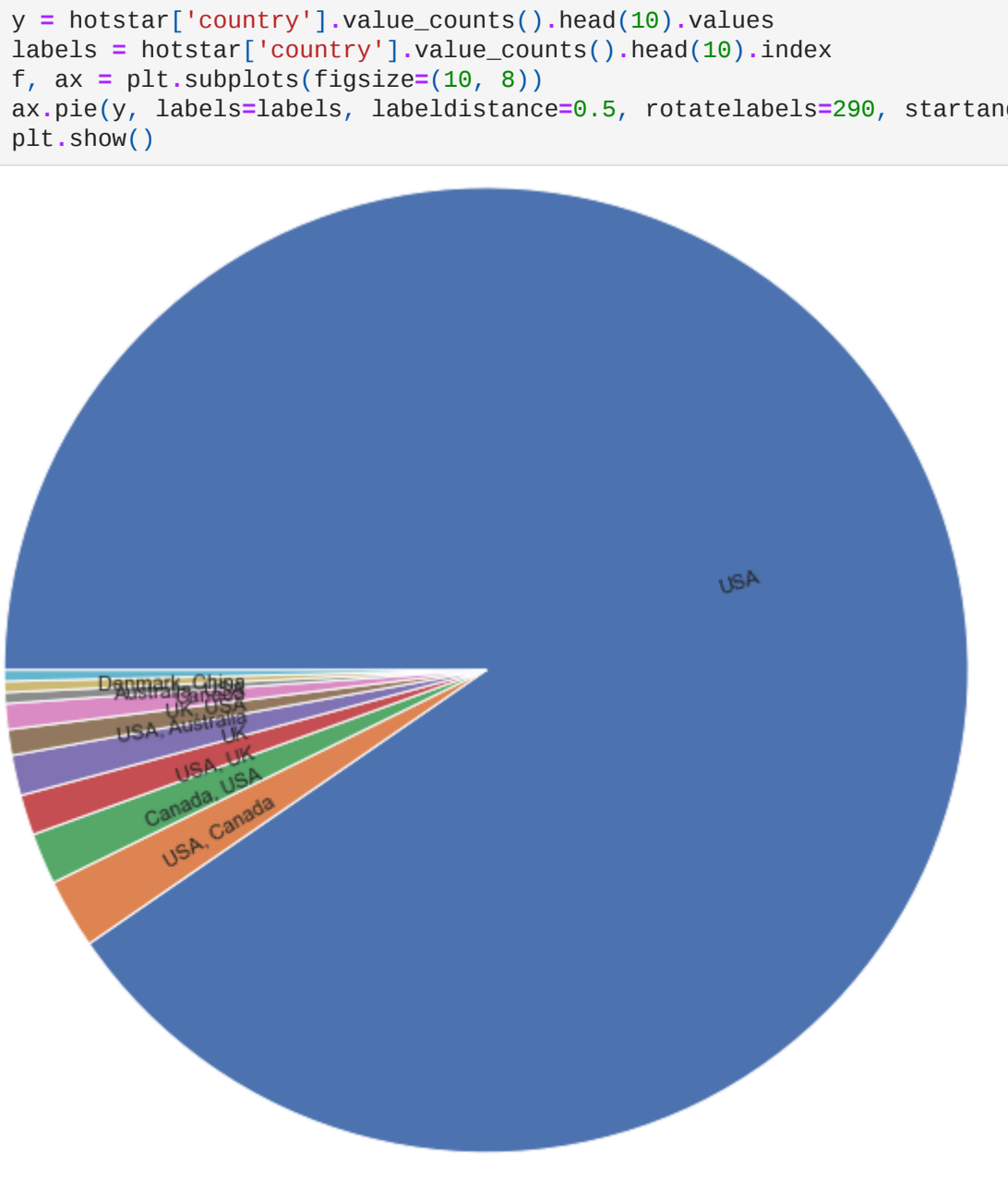
Top 20 Genre count

In [18]:
```
plt.figure(figsize=(15, 15))
ax=sns.barplot(x=hotstar['genre'].value_counts().head(20).index, y=hotstar['genre'].value_counts().head(20).values)
ax.set_xticklabels(hotstar['genre'].value_counts().head(20).index, rotation=90)
plt.show()
```



Country count

In [19]:
```
y = hotstar['country'].value_counts().head(10).values
labels = hotstar['country'].value_counts().head(10).index
f, ax = plt.subplots(figsize=(10, 8))
ax.pie(y, labels=labels, labeldistance=0.5, rotatelabels=290, startangle=180, radius=1.5, counterclock=False)
plt.show()
```



Content Based Recommendation system on movie plots

In [20]:
```
import warnings
from pandas.core.common import SettingWithCopyWarning
warnings.simplefilter(action="ignore", category=SettingWithCopyWarning)
```

In [21]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
```

In [22]:
```
hotstar['plot'] = hotstar['plot'].fillna('')
```

In [23]:
```
tfid = TfidfVectorizer()
plot_matrix = tfid.fit_transform(hotstar['plot'])
```

In [24]:
```
similarity_matrix = linear_kernel(plot_matrix, plot_matrix)
```

In [25]:
```
mapping = pd.Series(hotstar.index, index=hotstar.title)
```

In [26]:
```
def plot_content_recommender(m_name):
    m_index = mapping[m_name]
    similarity_score = list(enumerate(similarity_matrix[m_index]))
    similarity_score = sorted(similarity_score,key=lambda x:x[1],reverse=True)
    similarity_score = similarity_score[1:10]
    indices = [i[0] for i in similarity_score]
    return hotstar.title.iloc[indices]
```

In [27]:
```
plot_content_recommender('Coco')
```

Out[27]:
```
183                          Milo Murphy's Law
738                                  Tall Tale
268                              Fantasia 2000
45     Bambi and the Great Prince of the Forest
267                                   Fantasia
39                               Austin & Ally
327                           Great Migrations
273                               Finding Nemo
656                                 Robin Hood
Name: title, dtype: object
```

In [28]:
```
plot_content_recommender('Cars')
```

Out[28]:
```
100                Cars 3
781    The Christmas Star
99                 Cars 2
112     Time Travel Mater
109    Rescue Squad Mater
102           El Materdor
113           Tokyo Mater
818         The Lion King
830          The Love Bug
Name: title, dtype: object
```