```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'imdb-100000-moviestvshows:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F3080471%2F5298240%2Fbundle%2Farchive.zip%3FX-Goog-Algorithm%3DGOOG4-RSA-SHA256%26X-Goog-Crede

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

```
Downloading imdb-100000-moviestvshows, 10291563 bytes compressed
[==================================================] 10291563 bytes downloaded
Downloaded and uncompressed: imdb-100000-moviestvshows
Data source import complete.
```

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/imdb-100000-moviestvshows/contentDataPrime.csv
/kaggle/input/imdb-100000-moviestvshows/contentDataGenre.csv
/kaggle/input/imdb-100000-moviestvshows/contentDataRegion.csv
```

We have already maked the mood classification model, now there comes the second part , where we have to made the mood based recommendation system, now this dataset have 10000 movies and Tv series , we dont need such a huge ampunt of movies, because e are making a movie recommendation system , where I will show top 10 movies according to refresh the mood of the user. We have to select the best movies/Tv_shows from the daatset , proper data analysis is required for that. Lets do it.

```
df1=pd.read_csv("/kaggle/input/imdb-100000-moviestvshows/contentDataGenre.csv")
df2=pd.read_csv("/kaggle/input/imdb-100000-moviestvshows/contentDataPrime.csv")
df3=pd.read_csv("/kaggle/input/imdb-100000-moviestvshows/contentDataRegion.csv")
```

```
df1.head(1)# genres dataset
```

	dataId	genre
0	102795	Drama

```
df_dup1 = df1.duplicated('dataId', keep = False)
df_dup1.value_counts()
```

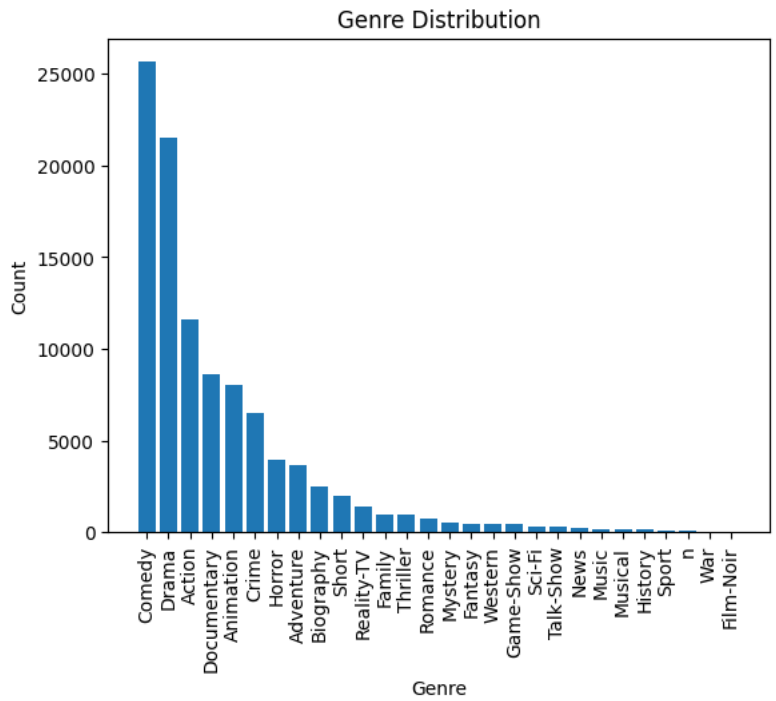
```
True     191655
False     27558
dtype: int64
```

```
# Delete duplicates from dataset 1
df1 = df1.drop_duplicates(subset='dataId')
```

```
(df1["genre"].value_counts()).shape # total no of genres
```

```
(28,)
```

```
import matplotlib.pyplot as plt
genre_counts = df1["genre"].value_counts()
plt.bar(genre_counts.index, genre_counts.values)
plt.xlabel("Genre")
plt.ylabel("Count")
plt.xticks(rotation=90)
plt.title("Genre Distribution")
plt.show()
```



Data Analysis: From this graph we can say that top 5 genres are Drama, Comedy, Romance, Action, Crime. But it includes both movies and Tv shows. So we should do individual plots for the Tv shows and movies which we will do in the df2 dataset.

```
df2.head(1)
```

	dataId	contentType	title	length	releaseYear	endYear	votes	rating	gross	certificate	description
0	102795	movie	Ratha Kanneer	154	1954	-1	349	8.5	-1	NaN	The story revolves around Mohanasundaram, a re...

```
df_dup1 = df2.duplicated('dataId', keep = False)
df_dup1.value_counts()
```

```
False    101602
True          4
dtype: int64
```

```
df2 = df2.drop_duplicates(subset ='dataId')
```

```
df2["gross"].value_counts()
```

```
-1           86955
 10000         660
 20000         434
 0             418
 30000         329
              ...
 36430000        1
 105490000       1
 75850000        1
 60090000        1
 144800000       1
Name: gross, Length: 4545, dtype: int64
```

gross –>

1. Basically depends on the how much users see the movie, it is not going to help us at any cost.
2. Also contains lots of missing values. I am going to drop it.

```
df2["endYear"].value_counts()
```

```
-1       91335
 2019      492
 2022      453
 2017      451
 2018      448
          ...
 1955       13
 2024       10
 1953        7
 1954        6
 1952        1
Name: endYear, Length: 74, dtype: int64
```

```
df2.drop(["endYear","gross"], axis=1, inplace=True)
```

```
df2.info()
```

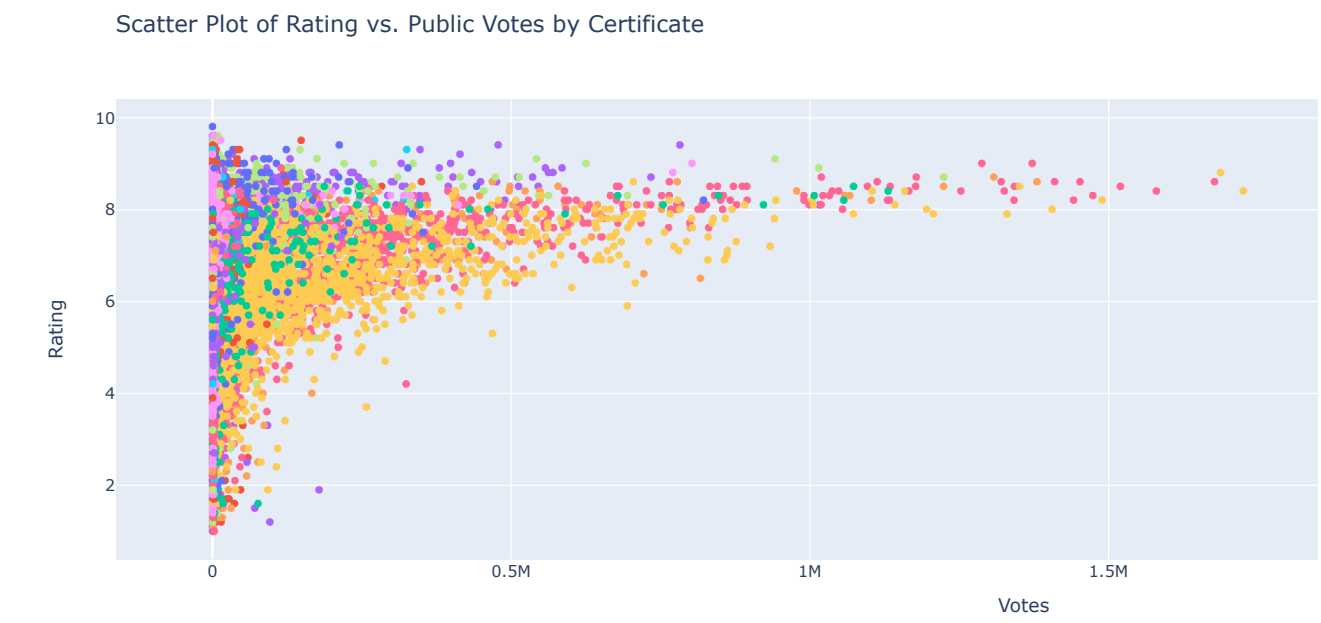
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101604 entries, 0 to 101604
Data columns (total 9 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   dataId       101604 non-null  int64  
 1   contentType  101604 non-null  object 
 2   title        101604 non-null  object 
 3   length       101604 non-null  object 
 4   releaseYear  101604 non-null  int64  
 5   votes        101604 non-null  int64  
 6   rating       101604 non-null  float64
 7   certificate  58356 non-null   object 
 8   description  101604 non-null  object 
dtypes: float64(1), int64(3), object(5)
memory usage: 7.8+ MB
```

I will drop the certificate column , as it contains many missing values and also in general user don't think about the certification of the movie before watching. Before droping this column , lets see the certificate column values.

```
df2['certificate'].value_counts()
```

```
Not Rated    12919
R            12262
TV-14         5211
TV-MA         4471
PG-13         4426
PG            3779
Approved      3450
TV-PG         3173
Unrated       1947
Passed        1939
TV-G          1789
G             1060
TV-Y7          783
TV-Y           587
GP             132
TV-Y7-FV       130
X               77
M               71
NC-17           47
M/PG            35
16+             18
TV-13           12
13+              7
F                5
AO               5
MA-17            4
E                3
E10+             3
12               3
Open             3
18+              2
18               1
T                1
EM               1
Name: certificate, dtype: int64
```

```
import plotly.express as px

# Assuming df is your DataFrame
fig = px.scatter(df2, x='votes', y='rating', color='certificate',
                 title='Scatter Plot of Rating vs. Public Votes by Certificate',
                 labels={'votes': 'Votes', 'rating': 'Rating', 'certificate': 'Certificate'},
                 )

# Show the interactive plot
fig.show()
```



You can see that the graph is skewed , because the movies which get higher rating with less no of votes are the best movies to consider , left and top most corner movies . you can see many not-rated movies comes under this category,also , you have to understand that other certified movies are also come in this category but as we required very less number of movies and also certificate have many missing values, also not rated movies can have higher ratings so we can drop this column.

```
df2.drop("certificate",axis=1,inplace=True)
```

```
df3.head(1)
```

	dataId	region
0	102795	India

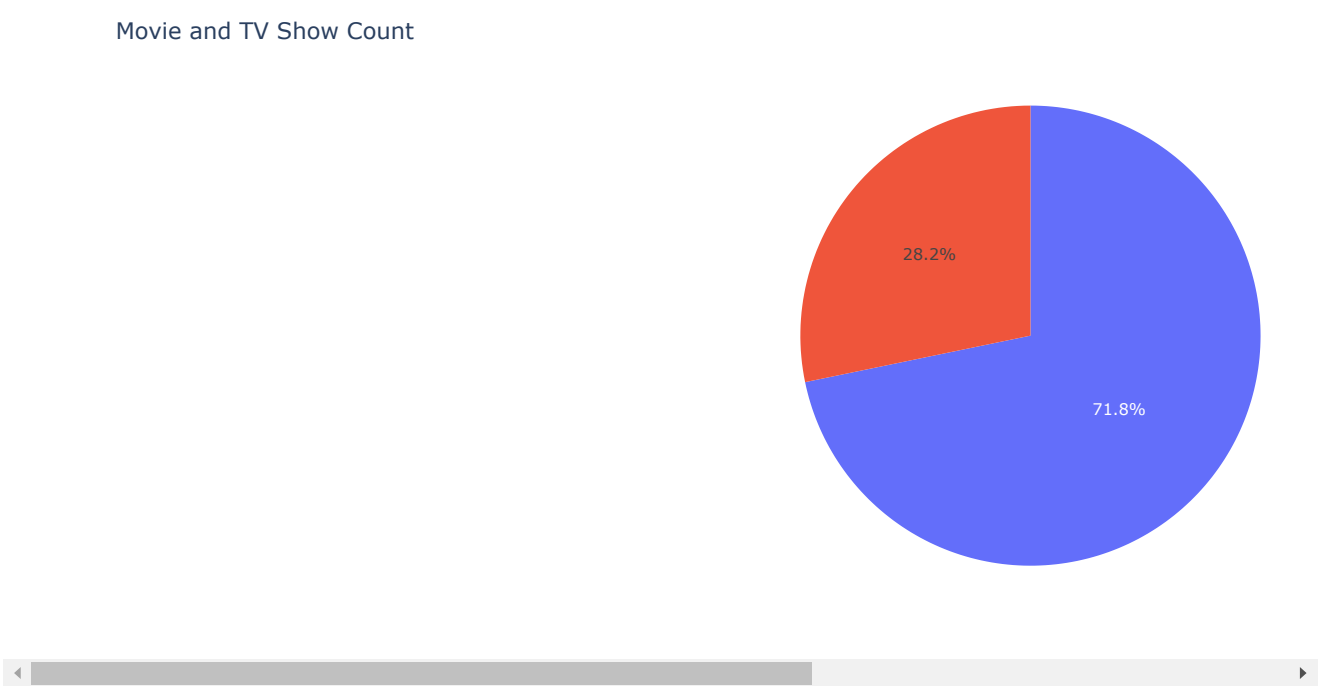
```
df_dup1 = df3.duplicated('dataId', keep = False)
df_dup1.value_counts()
df3 = df3.drop_duplicates(subset ='dataId')


merged_df = pd.merge(df3, df1,how='inner', on='dataId')
df = pd.merge(merged_df, df2, how = 'inner', on='dataId')


df.head(1)
```

	dataId	region	genre	contentType	title	length	releaseYear	votes	rating	description
0	102795	India	Drama	movie	Ratha Kanneer	154	1954	349	8.5	The story revolves around Mohanasundaram, a re...

```
df.info()
```

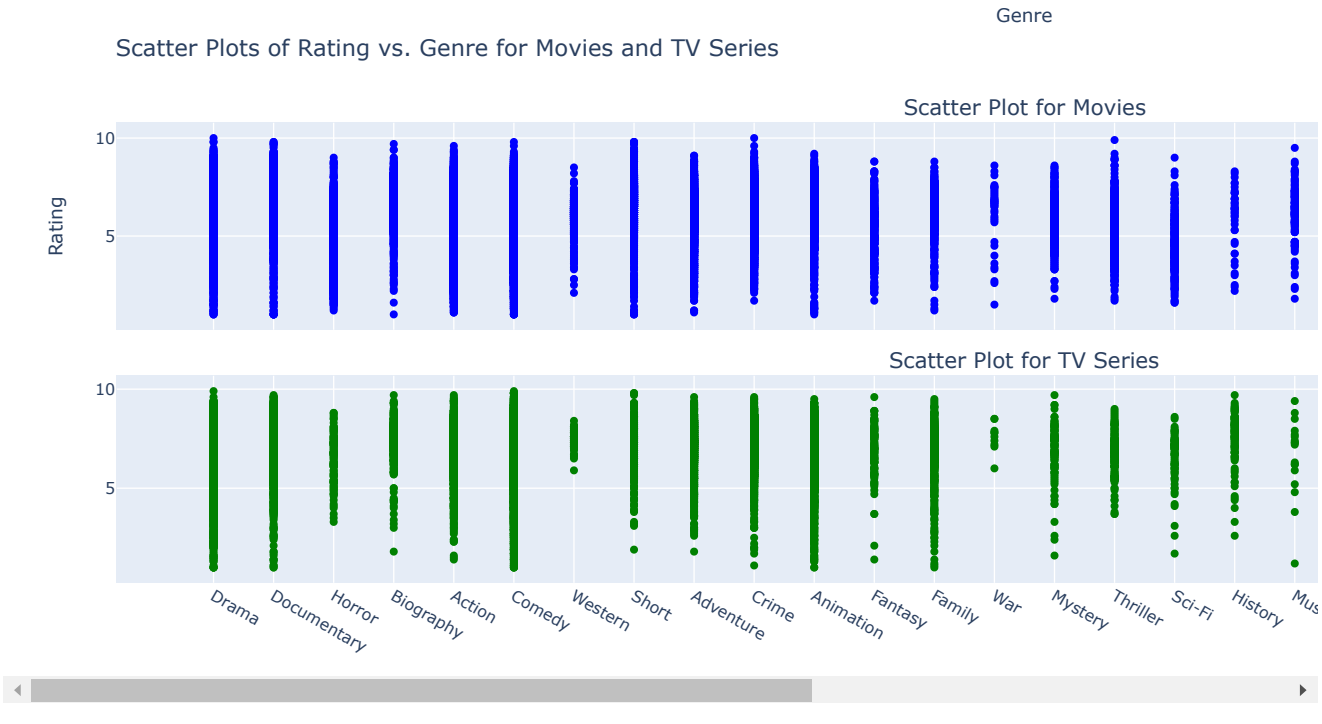
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101604 entries, 0 to 101603
Data columns (total 10 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   dataId       101604 non-null  int64  
 1   region       101604 non-null  object 
 2   genre        101604 non-null  object 
 3   contentType  101604 non-null  object 
 4   title        101604 non-null  object 
 5   length       101604 non-null  object 
 6   releaseYear  101604 non-null  int64  
 7   votes        101604 non-null  int64  
 8   rating       101604 non-null  float64
 9   description  101604 non-null  object 
dtypes: float64(1), int64(3), object(6)
memory usage: 8.5+ MB
```

```
content_type_counts = df['contentType'].value_counts()

fig = px.pie(values=content_type_counts.values, names=content_type_counts.index, title='Movie and TV Show Count')
fig.show()
```



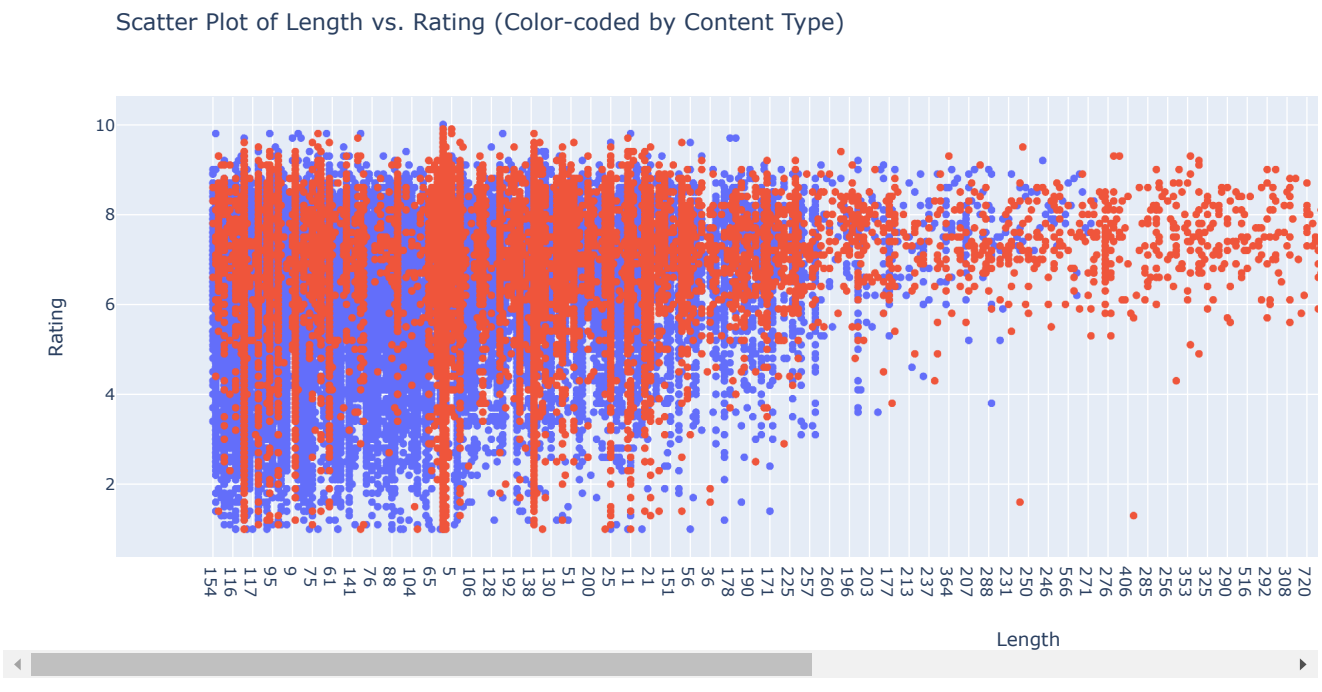
The dataset is movies biased but we will carefully observe it . We have to found that user prefer to see TV shows/ Movies.

```
import plotly.subplots as sp
import plotly.graph_objects as go

# Assuming df is your DataFrame
movies_df = df[df['contentType'] == 'movie']
tv_series_df = df[df['contentType'] == 'tvSeries']

# Create subplots
fig = sp.make_subplots(rows=2, cols=1, subplot_titles=['Scatter Plot for Movies', 'Scatter Plot for TV Series'],
                       shared_xaxes=True, vertical_spacing=0.1)

# Scatter plot for movies
scatter_movies = go.Scatter(x=movies_df['genre'], y=movies_df['rating'], mode='markers',
                            marker=dict(color='blue'), name='Movies')
fig.add_trace(scatter_movies, row=1, col=1)

# Scatter plot for TV series
scatter_tv_series = go.Scatter(x=tv_series_df['genre'], y=tv_series_df['rating'], mode='markers',
                               marker=dict(color='green'), name='TV Series')
fig.add_trace(scatter_tv_series, row=2, col=1)

# Update layout
fig.update_layout(title_text='Scatter Plots of Rating vs. Genre for Movies and TV Series',
                  xaxis_title='Genre', yaxis_title='Rating')

# Show the interactive plot
fig.show()
```



drama, comedy, documentary, animation, action, biography, sport, horror, adventure, thriller, crime these are the genres where the ratings are greate than 8 and which includes both tvseries and movies.

```
# Assuming df is your DataFrame
fig = px.scatter(df, x='length', y='rating', color='contentType',
                 title='Scatter Plot of Length vs. Rating (Color-coded by Content Type)',
                 labels={'length': 'Length', 'rating': 'Rating', 'contentType': 'Content Type'})

# Show the interactive plot
fig.show()
```



most of the values are within 570 who are higher rating shows , so I am going to drop those rows which are greater than 600. Definitely audience wants to see the conclusion of theshows in shorter period of time. I am also use only those movies who has rating above 8.0

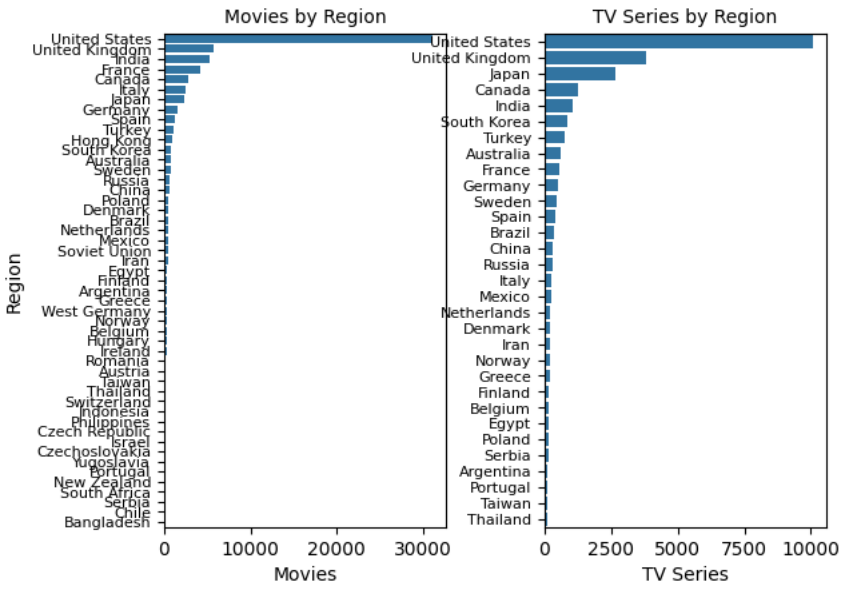
```
df['length'] = pd.to_numeric(df['length'], errors='coerce')
df_filtered = df[(df['length'] <= 600) & (df['rating'] >= 8)]
```

```
df_filtered
```

	dataId	region	genre	contentType	title	length	releaseYear	votes	rating	description
0	102795	India	Drama	movie	Ratha Kanneer	154.0	1954	349	8.5	The story revolves around Mohanasundaram, a re...
25	102820	Yugoslavia	Documentary	movie	Svet Koji Nestaje	109.0	1987	349	9.5	It is a love story. When the twelve-breasted b...
59	102854	United Kingdom	Animation	movie	Internet Story	10.0	2010	349	8.1	A fast-paced and thought-provoking film told t...
71	102866	United States	Short	movie	And Then	17.0	2021	349	9.5	Mana, a Japanese-American woman who arrives in...
74	102869	United Kingdom	Documentary	movie	Demetri Martin: If I	50.0	-1	349	8.0	Demetri Martin presents his existential dread ...
...	...	...	...	...	...	...	...	...	...	...
101594	491	United	Drama	tvSeries	The Morning	60.0	2019	106546	8.2	An inside look at the lives of the people who

```
import seaborn as sns
region_df =df.groupby(['region','contentType'], as_index = False)['dataId'].count().sort_values('dataId', ascending = False)
region_df = region_df[(region_df['dataId'] >= 100) & (region_df['region'] != 'n')]

fig , (ax1, ax2) = plt.subplots(1,2)

# Left graph
sns.barplot(x = 'dataId', y = 'region', data = region_df[region_df['contentType']=='movie'], ax = ax1)
ax1.set_xlabel('Movies')
ax1.set_ylabel('Region')
ax1.set_title('Movies by Region', fontsize = 10)
ax1.tick_params(axis='y', labelsize=8)

# Right graph
sns.barplot(x = 'dataId', y = 'region', data = region_df[region_df['contentType']=='tvSeries'], ax = ax2)
```

```
ax2.set_xlabel('TV Series')
ax2.set_ylabel('')
ax2.set_title('TV Series by Region', fontsize = 10)
ax2.tick_params(axis='y', labelsize=8)

plt.subplots_adjust(wspace=0.35);
```



Here definitely say that UsA, Uk, Japan, Canada, India movies are most rated , you can say that population is one of the factors but UK has 20 times less population than India , though it has high ratings. So I am going to extract top 5 regional movies.

```
regions_of_interest = ["India", "United States", "United Kingdom", "Japan", "Canada"]
df_filtered1 = df_filtered[df_filtered["region"].isin(regions_of_interest)]
```

```
df_filtered1
```

	dataId	region	genre	contentType	title	length	releaseYear	votes	rating	description
0	102795	India	Drama	movie	Ratha Kanneer	154.0	1954	349	8.5	The story revolves around Mohanasundaram, a re...
59	102854	United Kingdom	Animation	movie	Internet Story	10.0	2010	349	8.1	A fast-paced and thought-provoking film told t...
71	102866	United States	Short	movie	And Then	17.0	2021	349	9.5	Mana, a Japanese-American woman who arrives in...
74	102869	United Kingdom	Documentary	movie	Demetri Martin: If I	50.0	-1	349	8.0	Demetri Martin presents his existential dread ...
102	102897	Japan	Animation	movie	The Day I Bought a Star	16.0	2006	348	8.2	A young boy is tired of the city and escapes i...
...	...	...	...	...	...	...	...	...	...	...
101594	491	United	Drama	tvSeries	The Morning	60.0	2019	106546	8.2	An inside look at the lives of the people who

drama, comedy, documentary, animation, action, biography, sport, horror, adventure, thriller, crime

This is our final dataset ,now We already have 6 moods sad, angry , joy, happy, brave and lazy It is impossible to make a choice of movies for a specific mood based user, so I am going to use a following mapping :

1. drama, comedy
2. drama, documentary, sport, biography, animation
3. drama, horror, action
4. comedy, thriller, sport
5. sport, drama, comedy, animation, adventure, action
6. thriller, crime, sport,horror

Lets map the dataset,

```
# Create separate DataFrames based on genres
df0 = df_filtered1[df_filtered1['genre'].str.contains('Drama') | df_filtered1['genre'].str.contains('Comedy')]
df1 = df_filtered1[df_filtered1['genre'].str.contains('Drama') | df_filtered1['genre'].str.contains('Documentary') | df_filtered1['genre'].str.contains('Sport') | df_filtered1['genre'].str.contains('B
df2 = df_filtered1[df_filtered1['genre'].str.contains('Drama') | df_filtered1['genre'].str.contains('Horror') | df_filtered1['genre'].str.contains('Action')]
df3 = df_filtered1[df_filtered1['genre'].str.contains('Comedy') | df_filtered1['genre'].str.contains('Thriller') | df_filtered1['genre'].str.contains('Sport')]
df4 = df_filtered1[df_filtered1['genre'].str.contains('Sport') | df_filtered1['genre'].str.contains('Drama') | df_filtered1['genre'].str.contains('Comedy') | df_filtered1['genre'].str.contains('Anima
df5 = df_filtered1[df_filtered1['genre'].str.contains('Thriller') | df_filtered1['genre'].str.contains('Crime') | df_filtered1['genre'].str.contains('Sport') | df_filtered1['genre'].str.contains('Hor
```

```
print(df0.shape)
print(df1.shape)
print(df2.shape)
print(df3.shape)
print(df4.shape)
print(df5.shape)
```

```
(1936, 10)
(3153, 10)
(1198, 10)
(1165, 10)
(3102, 10)
(352, 10)
```

```
df0.to_csv('df0.csv', index=False)
df1.to_csv('df1.csv', index=False)
df2.to_csv('df2.csv', index=False)
```