```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np
```

```
import pandas as pd

df= pd.read_csv('/content/drive/MyDrive/netflix.csv')
display(df.head())
```

	show_id	type	title	director	cast	country	date_added	release_
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	

Double-click (or enter) to edit

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64 
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

```
df.nunique()
```

	0
show_id	8807
type	2
title	8807
director	4528
cast	7692
country	748
date_added	1767
release_year	74
rating	17
duration	220
listed_in	514
description	8775

dtype: int64

```
df.shape
```

```
(8807, 12)
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

```
#Checking for null values in each column
df.isna().sum()
```

	0
show_id	0
type	0
title	0
director	2634
cast	825
country	831
date_added	10
release_year	0
rating	4
duration	3
listed_in	0
description	0

dtype: int64

Task

Un-nest columns in the DataFrame loaded from "https://drive.google.com/drive/u/0/my-drive/Netflix.csv" and "https://drive.google.com/drive/u/0/my-drive/content/drive/MyDrive/Netflix.csv" that have cells with multiple comma-separated values by creating multiple rows.

Identify columns with comma-separated values

Subtask:

Examine the data to identify columns where cells contain multiple values separated by commas.

Reasoning: Inspect the first few rows and check the info of the dataframe to identify columns that might contain comma-separated values. Then sample some rows from the suspected columns to confirm.

```
display(df.head())
df.info()
display(df['cast'].sample(5))
display(df['listed_in'].sample(5))
display(df['country'].sample(5))
```

	show_id	type	title	director	cast	country	date_added	release_
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	
					Mayur More,			

4	s5	TV Show	Kota Factory	NaN	Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64 
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

	cast
8585	Mohamed Ragab, Lamitta Frangieh, Mohsen Mansou...
3040	Ahmed Sylla, Alban Ivanov, Ornella Fleury, Nat...
7554	NaN
2133	Anjelica Bette Fellini, Maddie Phillips, Kadee...
5092	Kouki Uchiyama, Ayumu Murase, Megumi Han, Ami ...

dtype: object

	listed_in
7589	Kids' TV, TV Comedies
2645	Anime Series, Crime TV Shows, International TV...
6535	Documentaries, International Movies
3384	Horror Movies, International Movies
4509	Dramas, International Movies, Sports Movies

dtype: object

	country
2321	United States

1124 United States

Un-nesting a column

For the chosen column, split the comma-separated strings and duplicate the corresponding rows for each value.

Reasoning: Split the 'listed_in' column and explode the DataFrame to un-nest the values, then display the head of the resulting DataFrame.

```
#Firstly, we will identify the columns which have comma separated values and cr

col_containing_commas = []

for col in df.columns:
  if df[col].dtype=='object':
    if df[col].str.contains(',').any():
      col_containing_commas.append(col)

col_containing_commas
```

```
['title',
 'director',
 'cast',
 'country',
 'date_added',
 'listed_in',
 'description']
```

```
df.columns
```

```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country',
'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

```
#To start with, performing string split for listed_in column
df['listed_in'] = df['listed_in'].str.split(',')
```

```
#Performing string split for other columns

df['cast'] = df['cast'].str.split(',')
df['country'] = df['country'].str.split(',')
df['director'] = df['director'].str.split(',')
```

```
#Unnesting for the other columns using explode
df= df.explode('director').explode('country').explode('cast').explode('listed_i
```

```
df.shape
```

```
(202065, 12)
```

CHECKING THE NULL VALUES

```
df.isnull().sum()
```

	0
show_id	0
type	0
title	0
director	50643
cast	2149
country	11897
date_added	158
release_year	0
rating	67
duration	3
listed_in	0
description	0

dtype: int64

Rating column has incorrect data, last 3 rows are simialr to Duration column values. Lets check the specific cells where it is null value.

```
df[df['duration'].isnull()]
```

	show_id	type	title	director	cast	country	date_added	release_ye
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	20
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	September 16, 2016	20
5813	s5814	Movie	Louis C.K.: Live at	Louis C.K.	Louis	United	August 15,	20

Corresponding null values of Duration are dispalyed in Rating column, hence we shall replace the Duration column with corresponding Rating column values.

```
# Identify rows where duration is null and rating contains duration info
mask = df['duration'].isnull() & df['rating'].astype(str).str.contains('min|Sea

# Replace null duration with rating for these rows
df.loc[mask, 'duration'] = df.loc[mask, 'rating']
```

```
df['duration'].isnull().any()
```

```
np.False_
```

Replace duration related info values in Ratings column with NR(Not Rated).

```
# Identify rows where duration is null and rating contains duration info
mask1 = df['rating'].astype(str).str.contains('min|Season', na=False)


# Replace null duration with rating for these rows
df.loc[mask1, 'rating'] = 'NR'

#*************************************************************************
#ANOTHER APPROACH USING NP.WHERE()
#import numpy as np

# Create the condition for replacement
#condition = df['rating'].astype(str).str.contains('min|Season', na=False)

# Use np.where to replace values in the 'rating' column
#df['rating'] = np.where(condition, 'NR', df['rating'])


# Calculate the mode of 'date_added' for each 'release_year' and fill nulls
df['date_added'] = df['date_added'].fillna(
    df.groupby('release_year')['date_added'].transform(lambda x: x.mode()[0] if
)
```

```
from datetime import datetime
from dateutil.parser import parse
arr=[]
for i in df['date_added'].values:
  dt1=parse(i)
  arr.append(dt1.strftime('%Y-%m-%d'))
df['Modified_Added_date'] =arr
df['Modified_Added_date']=pd.to_datetime(df['Modified_Added_date'])
df['month_added']=df['Modified_Added_date'].dt.month
# Corrected: Use isocalendar().week instead of .week
df['week_Added']=df['Modified_Added_date'].dt.isocalendar().week
df['year']=df['Modified_Added_date'].dt.year
df.head()
```

	show_id	type	title	directors	actors	country	date_added	release_y
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	Unknown Actor	United States	September 25, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Ama Qamata	South Africa	September 24, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Ama Qamata	South Africa	September 24, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Ama Qamata	South Africa	September 24, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Khosi Ngema	South Africa	September 24, 2021	2

```
# Rename the 'old_column_name' to 'new_column_name'
df.rename(columns={'cast': 'actors','listed_in': 'genre','director':'directors'

# Display the updated column names
print(df.columns)
```

```
Index(['show_id', 'type', 'title', 'directors', 'actors', 'country',
       'date_added', 'release_year', 'rating', 'duration', 'genre',
       'description', 'Modified_Added_date', 'month_added', 'year'],
      dtype='object')
```

```
#replacing nan values of director and actor by Unknown Actor and Director

df['actors'].fillna('Unknown Actor',inplace=True)
df['directors'].fillna('Unknown Director',inplace=True)
df.head()
```

```
/tmp/ipython-input-24-3806944103.py:3: FutureWarning: A value is trying to
The behavior will change in pandas 3.0. This inplace method will never work

For example, when doing 'df[col].method(value, inplace=True)', try using 'd


  df['actors'].fillna('Unknown Actor',inplace=True)
/tmp/ipython-input-24-3806944103.py:4: FutureWarning: A value is trying to
The behavior will change in pandas 3.0. This inplace method will never work

For example, when doing 'df[col].method(value, inplace=True)', try using 'd


  df['directors'].fillna('Unknown Director',inplace=True)
```

	show_id	type	title	directors	actors	country	date_added	release_y
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	Unknown Actor	United States	September 25, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Ama Qamata	South Africa	September 24, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Ama Qamata	South Africa	September 24, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Ama Qamata	South Africa	September 24, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Khosi Ngema	South Africa	September 24, 2021	2

```
# Attempt to fill null 'country' values based on the mode of 'country' for each
df['country'].fillna(
    df.groupby('directors')['country'].transform(lambda x: x.mode()[0] if not x
    inplace=True
)

# For any remaining nulls, attempt to fill based on the mode of 'country' for e
df['country'].fillna(
    df.groupby('actors')['country'].transform(lambda x: x.mode()[0] if not x.mo
    inplace=True
)

# Fill any remaining null values with 'unknown'
df['country'].fillna('unknown', inplace=True)

# Check for null values to confirm
print("Null values after imputation:")
print(df.isnull().sum()['country'])
```

```
/tmp/ipython-input-25-3339311038.py:2: FutureWarning: A value is trying to
The behavior will change in pandas 3.0. This inplace method will never work

For example, when doing 'df[col].method(value, inplace=True)', try using 'd


  df['country'].fillna(
Null values after imputation:
0
```

```
df['type'].value_counts()
```

type	count
Movie	145917
TV Show	56148

dtype: int64

```
df.duplicated().value_counts()
```

	count
False	202058
True	7

dtype: int64

```
df.isnull().sum()
```

	0
show_id	0
type	0
title	0
directors	0
actors	0
country	0
date_added	0
release_year	0
rating	67
duration	0
genre	0
description	0
Modified_Added_date	0
month_added	0
year	0

dtype: int64

```
df[df['rating'].isnull()]
```

	show_id	type	title	directors	actors	country	date_added	re
			13TH: A Conversation					

5989	s5990	Movie	Conversation with Oprah Winfrey & Ava ...	Unknown Director	Oprah Winfrey	United States	January 26, 2017
5989	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava ...	Unknown Director	Ava DuVernay	United States	January 26, 2017
6827	s6828	TV Show	Gargantia on the Verdurous Planet	Unknown Director	Kaito Ishikawa	Japan	December 1, 2016
6827	s6828	TV Show	Gargantia on the Verdurous Planet	Unknown Director	Kaito Ishikawa	Japan	December 1, 2016
6827	s6828	TV Show	Gargantia on the Verdurous Planet	Unknown Director	Hisako Kanemoto	Japan	December 1, 2016
...	...	...	...	...	...	...	...
7537	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Francesco Migliore	Italy	March 1, 2017
7537	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Albrecht Weimer	Italy	March 1, 2017
7537	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Giulia Dichiaro	Italy	March 1, 2017
7537	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Alessandra Oriti Niosi	Italy	March 1, 2017
7537	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Andreas Segeritz	Italy	March 1, 2017

67 rows × 15 columns

After cleaning up and preprocessing the dataset, no null values are present in the dataset.

```
#Check the various duration values of movies/shows
df['duration'].value_counts()
```

duration	count
1 Season	35035
2 Seasons	9559
3 Seasons	5084
94 min	4343
106 min	4040
...	...
3 min	4
5 min	3
8 min	2
9 min	2
11 min	2

220 rows × 1 columns

dtype: int64

```
#Data frame for movies and tv shows

movie_df = df[df['type'] == 'Movie']
tv_df = df[df['type'] == 'TV Show']
```

```
print("Movie Duration Stats:")
print(movie_df['duration'].describe())
```

```
Movie Duration Stats:
count     145917
unique       205
top       94 min
freq        4343
Name: duration, dtype: object
```

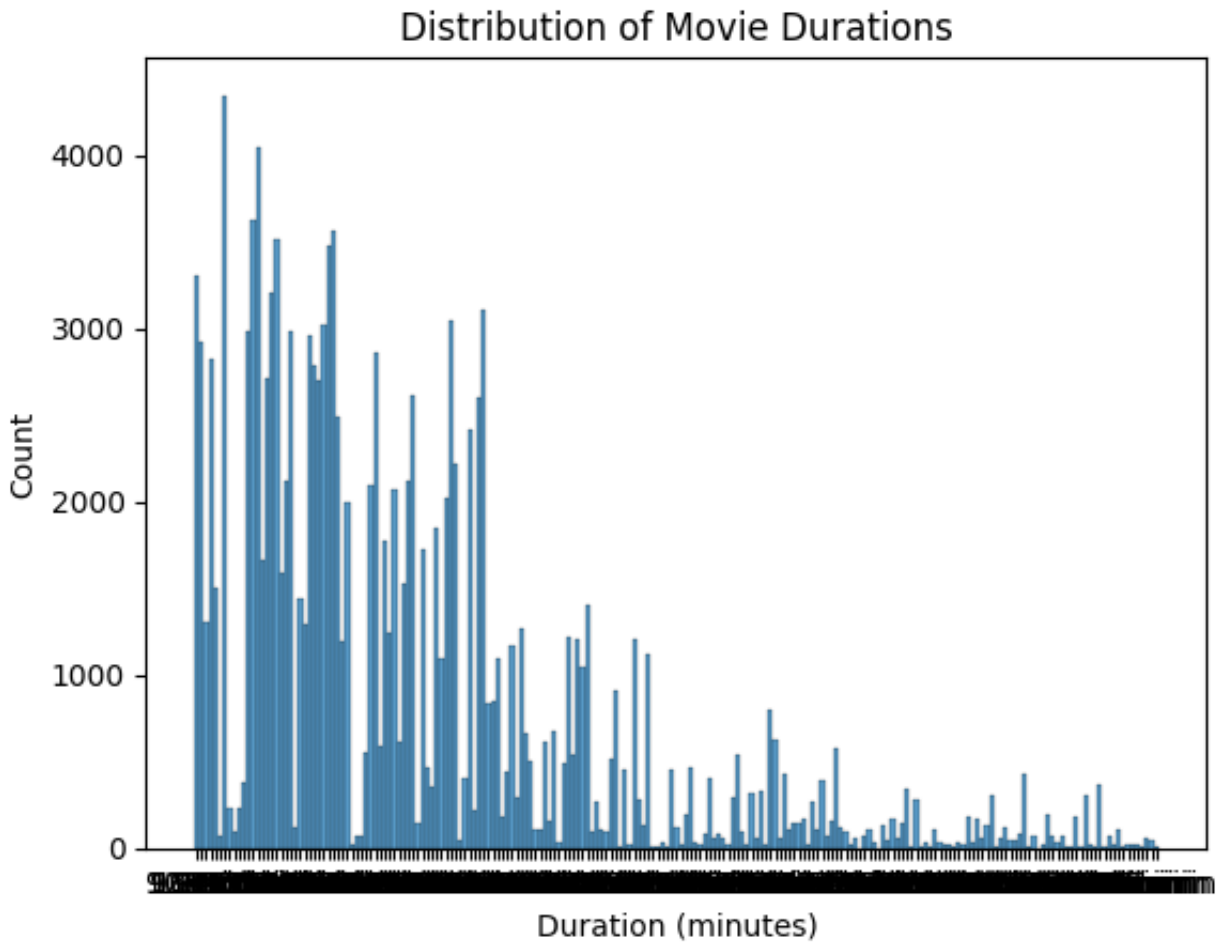
```
print("TV Show Seasons Stats:")
print(tv_df['duration'].value_counts().sort_index())
```

```
TV Show Seasons Stats:
duration
1 Season      35035
10 Seasons      220
11 Seasons       30
12 Seasons      111
13 Seasons      132
15 Seasons       96
17 Seasons       30
2 Seasons      9559
3 Seasons      5084
4 Seasons      2134
5 Seasons      1698
6 Seasons       633
7 Seasons       843
8 Seasons       286
9 Seasons       257
Name: count, dtype: int64
```

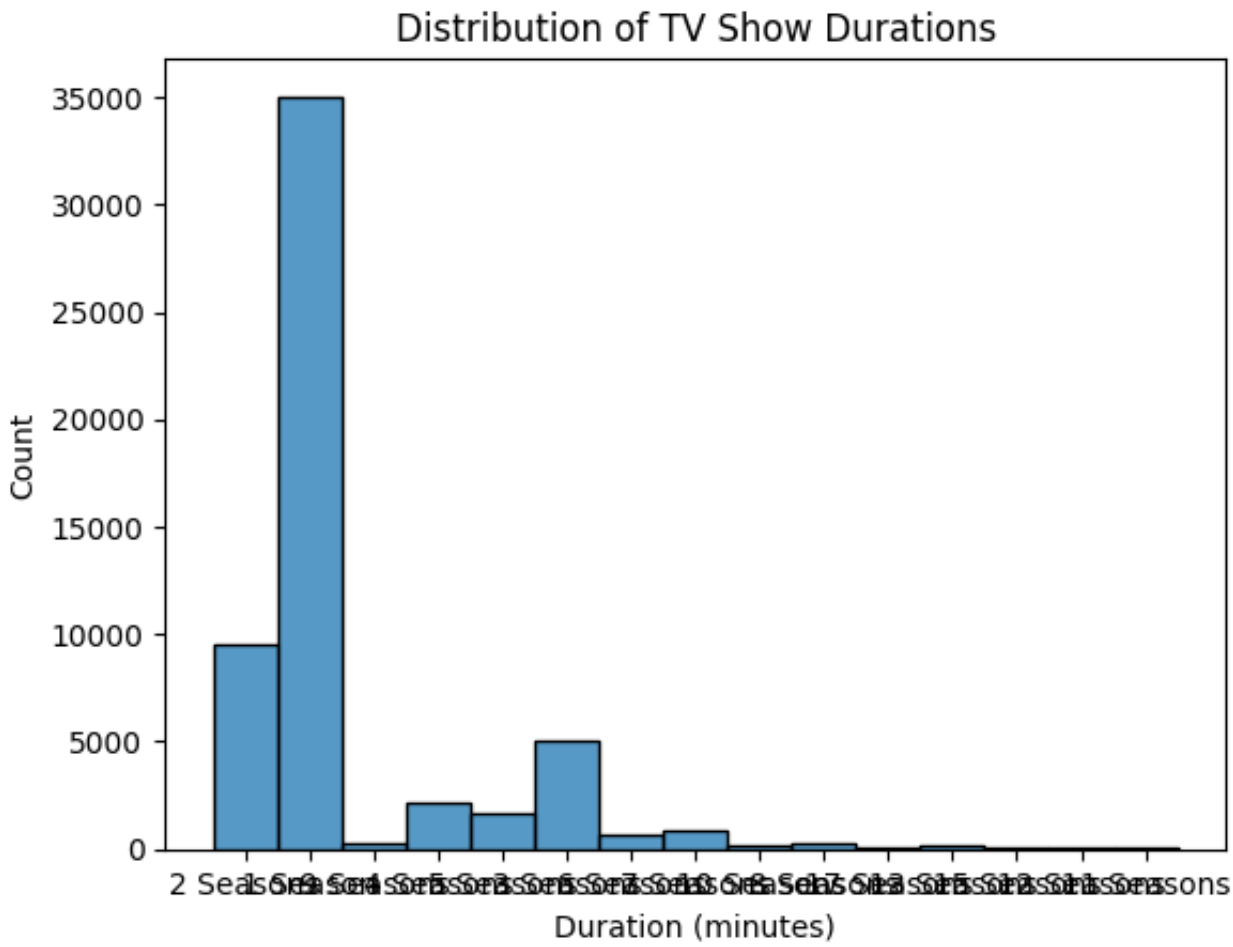
```
# Get the percentage of movies and TV shows
percentage_by_type = df['type'].value_counts(normalize=True) * 100

# Display the percentages
print(percentage_by_type)
```

```
type
Movie      72.212902
TV Show    27.787098
Name: proportion, dtype: float64
```

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.histplot(movie_df['duration'], bins=50)
plt.title('Distribution of Movie Durations')
plt.xlabel('Duration (minutes)')
plt.ylabel('Count')
plt.show()
```



```
sns.histplot(tv_df['duration'], bins=20)
plt.title('Distribution of TV Show Durations')
plt.xlabel('Duration (minutes)')
plt.ylabel('Count')
plt.show()
```




```
#Lets check the occurrences of ratings
df['rating'].value_counts()
```

Show hidden output

By checking the occurrences of ratings, most shows are rated as TV-MA, suitable for Adults.

```
df['genre'].value_counts().head()
```

genre	count
International Movies	27141
Dramas	19657
Comedies	13894
Action & Adventure	12216
Dramas	10149

dtype: int64

International Movies , Dramas , Comedies , Action & Adventure , Dramas - **Top 5 Genres.** that viewers are viewing.

```
df['country'].value_counts().head()
```

country	count
United States	55675
India	23202
United Kingdom	9802
United States	9521
Japan	7680

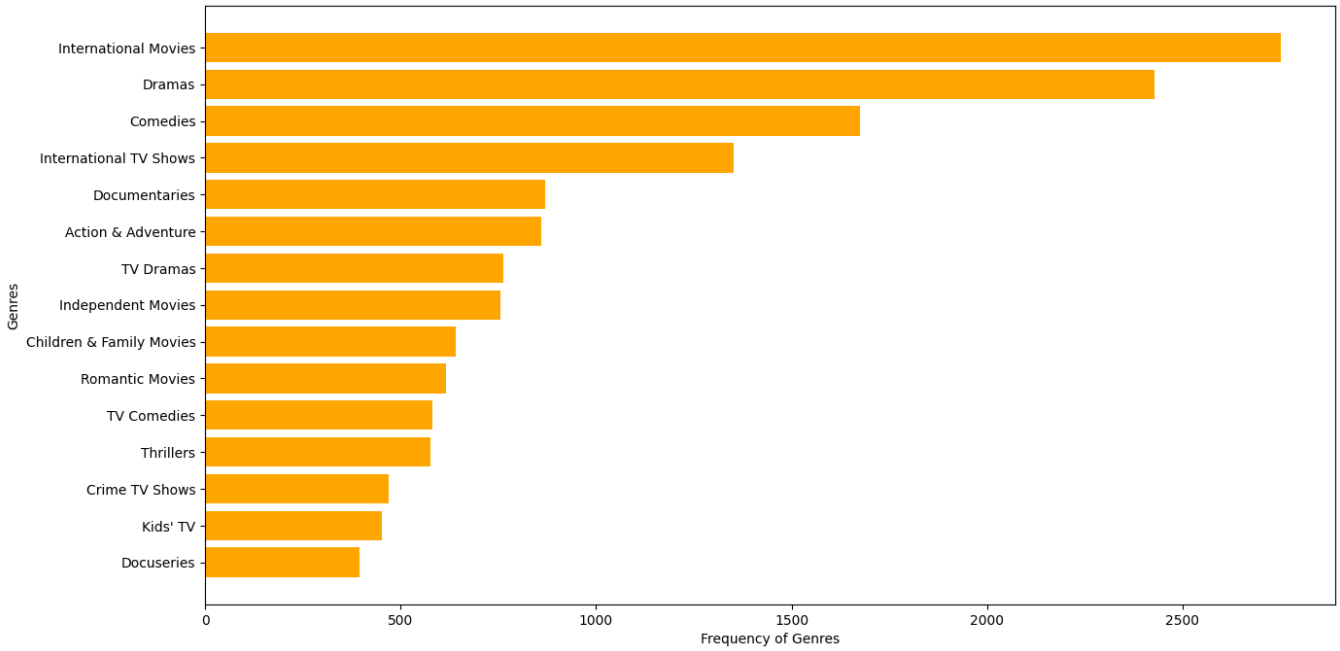
dtype: int64

United States, India are the TOP 2 countries who are producing movies/shows.

```
# Clean up leading/trailing spaces in un-nested columns
for col in ['directors', 'actors', 'country', 'genre']:
    # Ensure the column is of string type before applying .str methods
    df[col] = df[col].astype(str).str.strip()

# Display the head to see the effect of cleaning
display(df.head())
```

	show_id	type	title	directors	actors	country	date_added	release_y
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	Unknown Actor	United States	September 25, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Ama Qamata	South Africa	September 24, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Ama Qamata	South Africa	September 24, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Ama Qamata	South Africa	September 24, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Khosi Ngema	South Africa	September 24, 2021	2

```
#Check for the top genres in which there movies/shows
df_genre=df.groupby(['genre']).agg({"title":"nunique"}).reset_index().sort_valu

plt.figure(figsize=(15,8))
plt.barh(df_genre[::-1]['genre'], df_genre[::-1]['title'],color=['orange'])
plt.xlabel('Frequency of Genres')
plt.ylabel('Genres')
plt.show()
```



Genre with most tv or shows is **INTERNATIONAL MOVIES**.

SECOND MOST VIEWED GENRE IS DRAMAS.

```
df['duration_copy'] = df['duration'].copy()
```

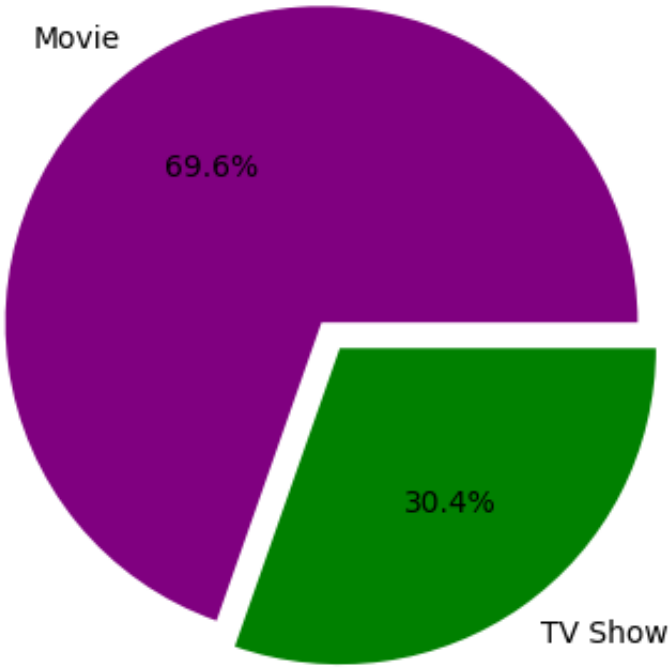
```
# Separate movie durations and convert to numeric
movie_durations_numeric = df[df['type'] == 'Movie']['duration'].str.replace(' m

# Define bins and labels for movie durations
bins1 = [-1, 1, 50, 80, 100, 120, 150, 200, 315]
labels1 = ['<1', '1-50', '50-80', '80-100', '100-120', '120-150', '150-200', '2

# Apply pd.cut to the numerical movie durations and create a new column for the
# We'll create a new column in the original dataframe and fill only for movie r
df['duration_category'] = pd.NA # Initialize the new column with pandas NA
df.loc[df['type'] == 'Movie', 'duration_category'] = pd.cut(movie_durations_num

# Display the head of the DataFrame, including the new column
display(df.head())
```

	show_id	type	title	directors	actors	country	date_added	release_y
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	Unknown Actor	United States	September 25, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Ama Qamata	South Africa	September 24, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Ama Qamata	South Africa	September 24, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Ama Qamata	South Africa	September 24, 2021	2
1	s2	TV Show	Blood & Water	Unknown Director	Khosi Ngema	South Africa	September 24, 2021	2

```
#number of distinct titles on the basis of type
df.groupby(['type']).agg({"title":"nunique"})
```

type	title
Movie	6131
TV Show	2676

```
df_type=df.groupby(['type']).agg({"title":"nunique"}).reset_index()
plt.pie(df_type['title'],explode=(0.05,0.05), labels=df_type['type'],colors=['
plt.show()
```

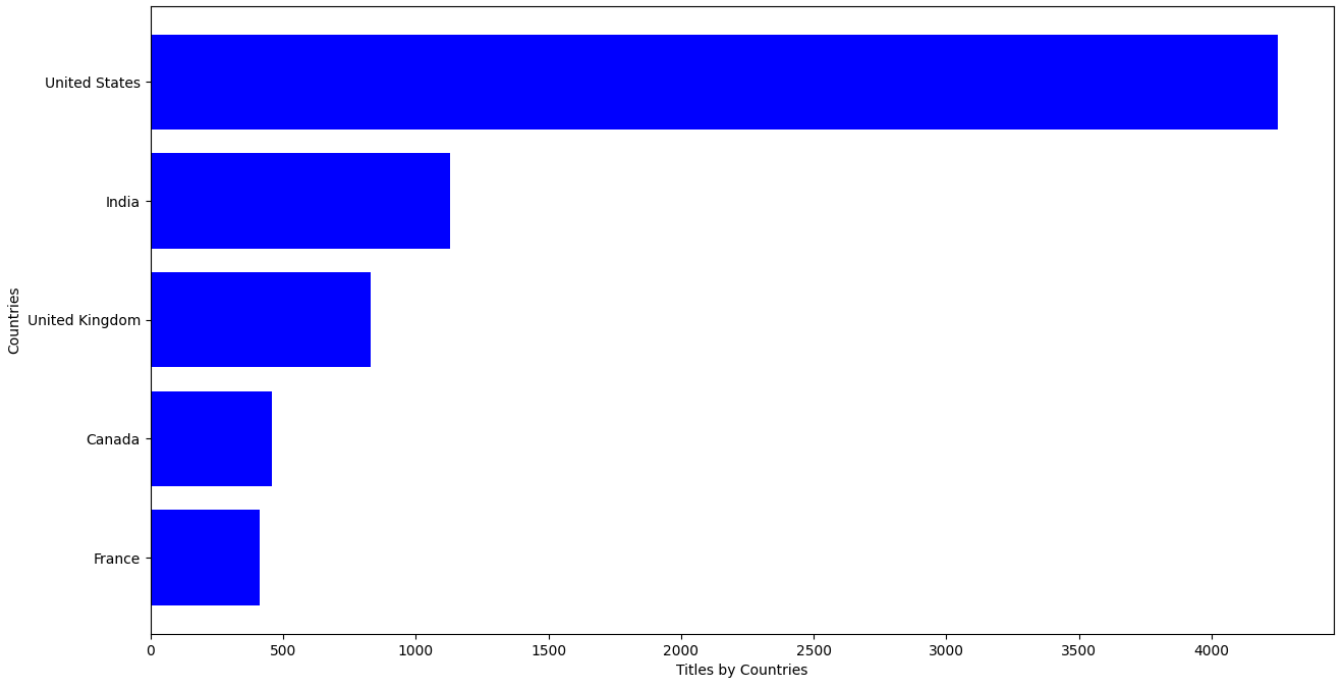


```
#number of distinct titles on the basis of country
df.groupby(['country']).agg({"title":"nunique"})
```

country	title
	8
Afghanistan	1
Albania	1
Algeria	3
Angola	1
...	...
Venezuela	4
Vietnam	7
West Germany	5
Zimbabwe	3
unknown	200

124 rows × 1 columns

```
df_country=df.groupby(['country']).agg({"title":"nunique"}).reset_index().sort_
plt.figure(figsize=(15,8))
plt.barh(df_country[::-1]['country'], df_country[::-1]['title'],color=['blue'])
plt.xlabel('Titles by Countries')
plt.ylabel('Countries')
plt.show()
```



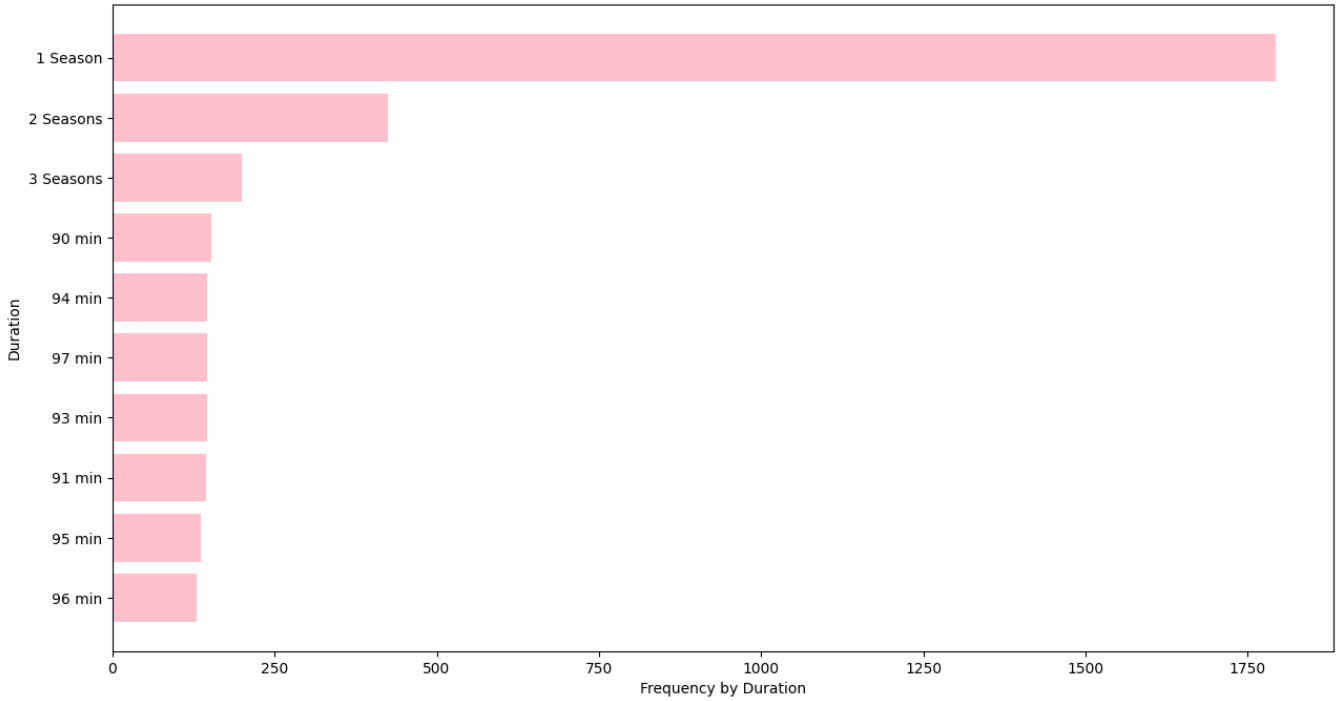
US tops the list of most no of titles(movies/shows) produced in a country.

```
#number of distinct titles on the basis of rating
df.groupby(['rating']).agg({"title":"nunique"}).sort_values(by='title', ascendi
```

rating	title
TV-MA	3207
TV-14	2160
TV-PG	863
R	799
PG-13	490
TV-Y7	334
TV-Y	307
PG	287
TV-G	220
NR	83
G	41
TV-Y7-FV	6
NC-17	3
UR	3

Most of the movies/shows are rated as TV-MA, for adults.

```
df_duration=df.groupby(['duration']).agg({"title":"nunique"}).reset_index().sor
plt.figure(figsize=(15,8))
plt.barh(df_duration[::-1]['duration'], df_duration[::-1]['title'],color=['pink
plt.xlabel('Frequency by Duration')
plt.ylabel('Duration')
plt.show()
```



By this result,

Shows with only 1 season are most viewed.

Movies with duration 90 mins,94 mins are most viewed.

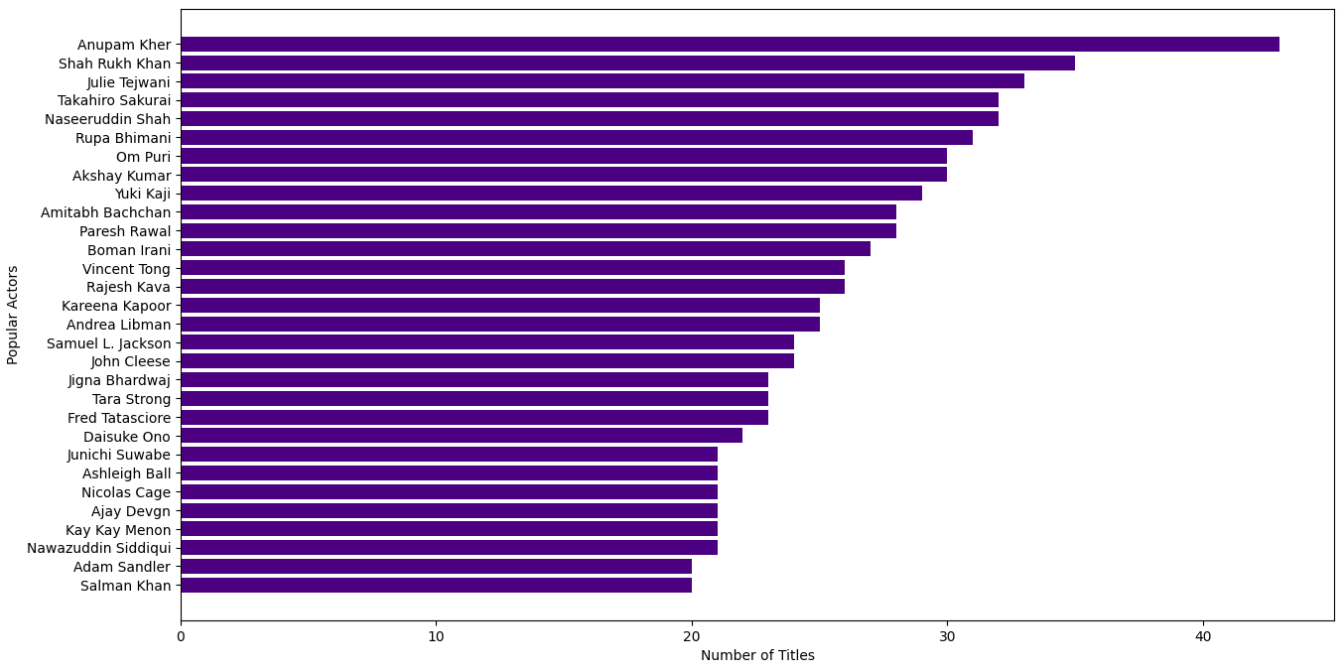
```
#number of distinct titles on the basis of Actors
df.groupby(['actors']).agg({"title":"nunique"})
```

actors	title
"Riley" Lakdhar Dridi	1
'Najite Dede	2
2 Chainz	1
2Mex	1
4Minute	1
...	...
Şevket Çoruh	1
Şinasi Yurtsever	3
Şükran Ovalı	1
Şükrü Özyıldız	2
Ṣọpẹ́ Dìrísù	1

36440 rows × 1 columns

```
# Filter out 'unknown actor' entries before grouping and counting
df_known_actors = df[df['actors'] != 'Unknown Actor'].copy()

# Group by actors, count unique titles, sort and get top 30 (since we removed '
df_actors=df_known_actors.groupby(['actors']).agg({"title":"nunique"}).reset_ir
```

```
plt.figure(figsize=(15,8))
plt.barh(df_actors[::-1]['actors'], df_actors[::-1]['title'],color=['indigo'])
plt.xlabel('Number of Titles') # Changed label to be more accurate
plt.ylabel('Popular Actors')
plt.show()
```



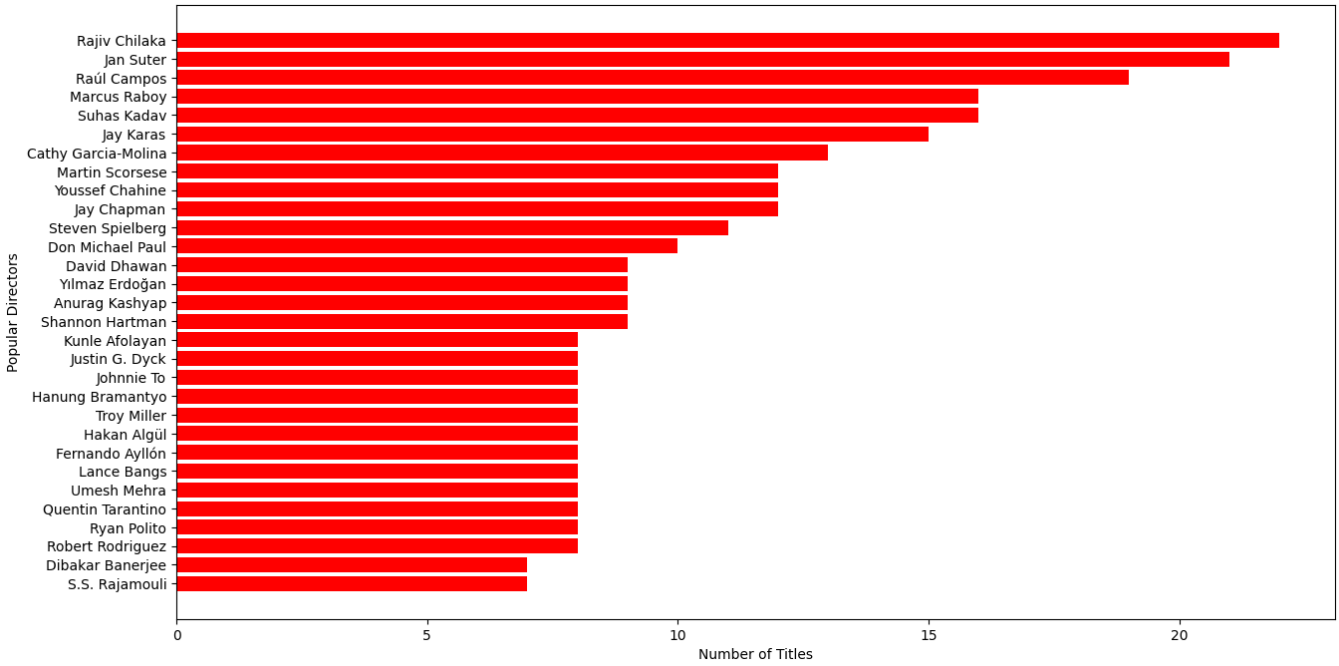
Anupam Kher has acted in most no of movies, next is Shah Ruk Khan.

```
#number of distinct titles on the basis of Directors
df.groupby(['directors']).agg({'title':'nunique'})
```

directors	title
A. L. Vijay	2
A. Raajdheep	1
A. Salaam	1
A.R. Murugadoss	2
Aadish Keluskar	1
...	...
Éric Warin	1
Ísold Uggadóttir	1
Óskar Thór Axelsson	1
Ömer Faruk Sorak	3
Şenol Sönmez	2

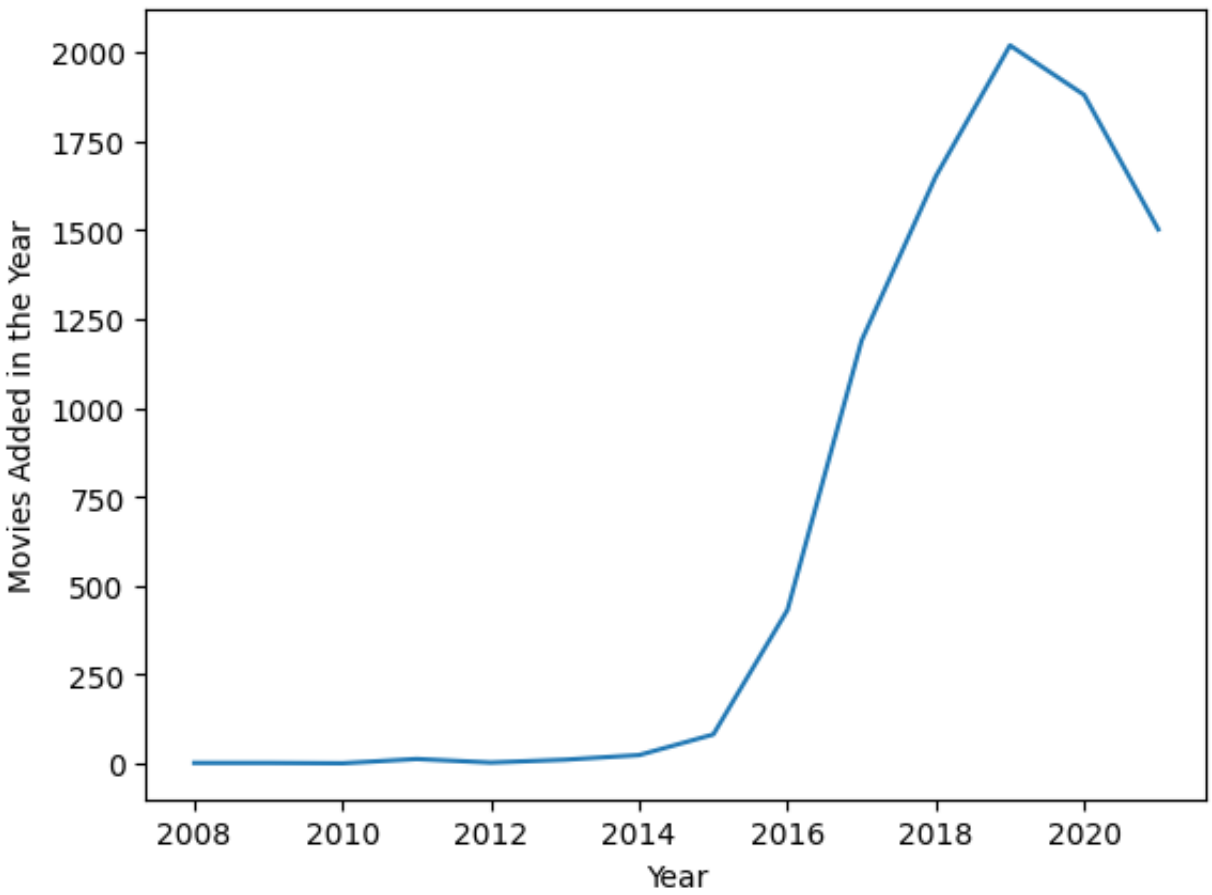
4994 rows × 1 columns

```
df_directors = df.groupby(['directors']).agg({'title':'nunique'}).reset_index()

# Filter out the 'Unknown Director' from the df_directors DataFrame
df_directors = df_directors[df_directors['directors']!= 'Unknown Director']

# Sort and get top 30 directors (since we filtered out 'Unknown Director')
df_directors = df_directors.sort_values(by=['title'],ascending=False)[:30]


plt.figure(figsize=(15,8))
plt.barh(df_directors[::-1]['directors'],df_directors[::-1]['title'],color=['re
plt.xlabel('Number of Titles') # Changed label to be more accurate
plt.ylabel('Popular Directors')
plt.show()
```



Rajiv Chilaka is the top director who had dorector most no of titles.

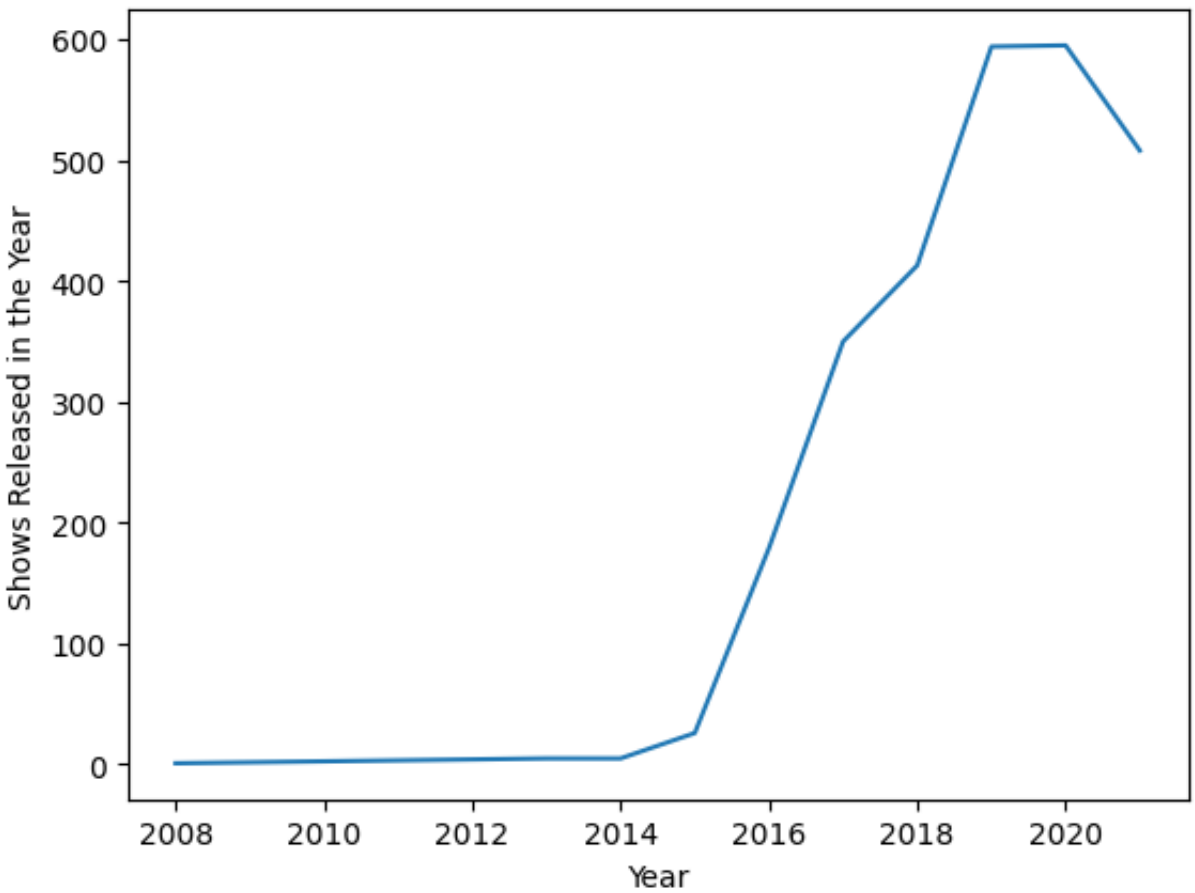
```
#Trend of Movies added to Netflix over the years
df_year=df.groupby(['year']).agg({"title":"nunique"}).reset_index()
sns.lineplot(data=df_year, x='year', y='title')
plt.ylabel("Movies Added in the Year")
plt.xlabel("Year")
plt.show()
```



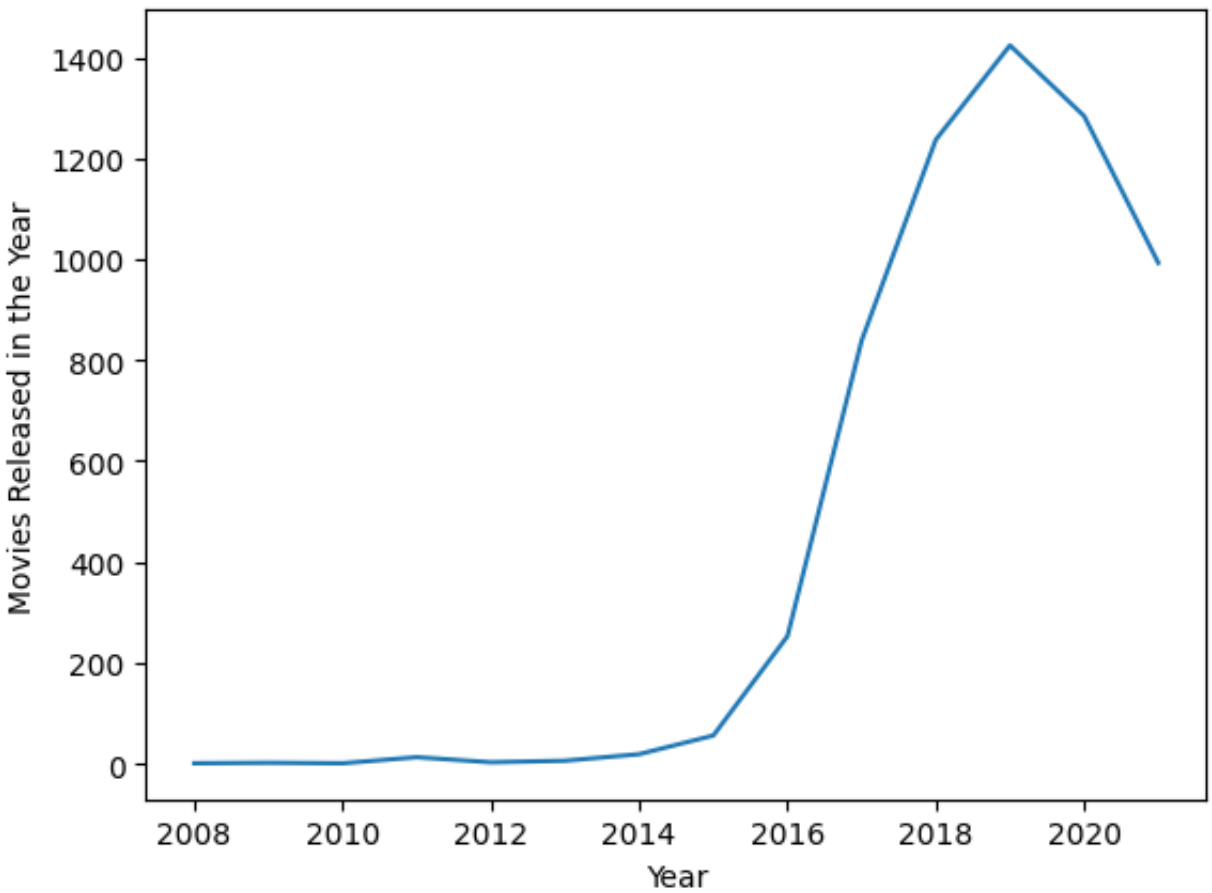
Most of the movies were added from 2015 to 2019.

There was s drop in addition of movies from 2019.

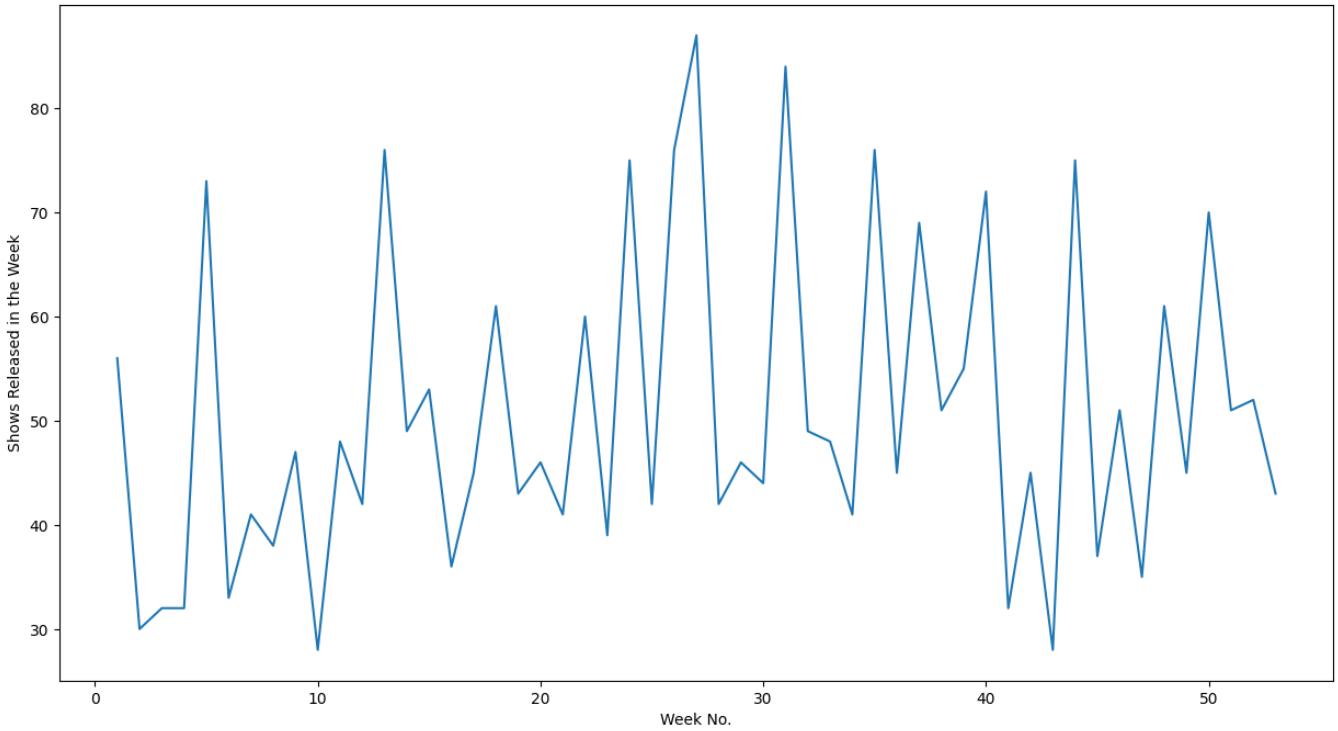
```
#Trend of Shows added to Netflix over the years
df_tv_year=tv_df.groupby(['year']).agg({"title":"nunique"}).reset_index()
sns.lineplot(data=df_tv_year, x='year', y='title')
plt.ylabel("Shows Released in the Year")
plt.xlabel("Year")
plt.show()
```



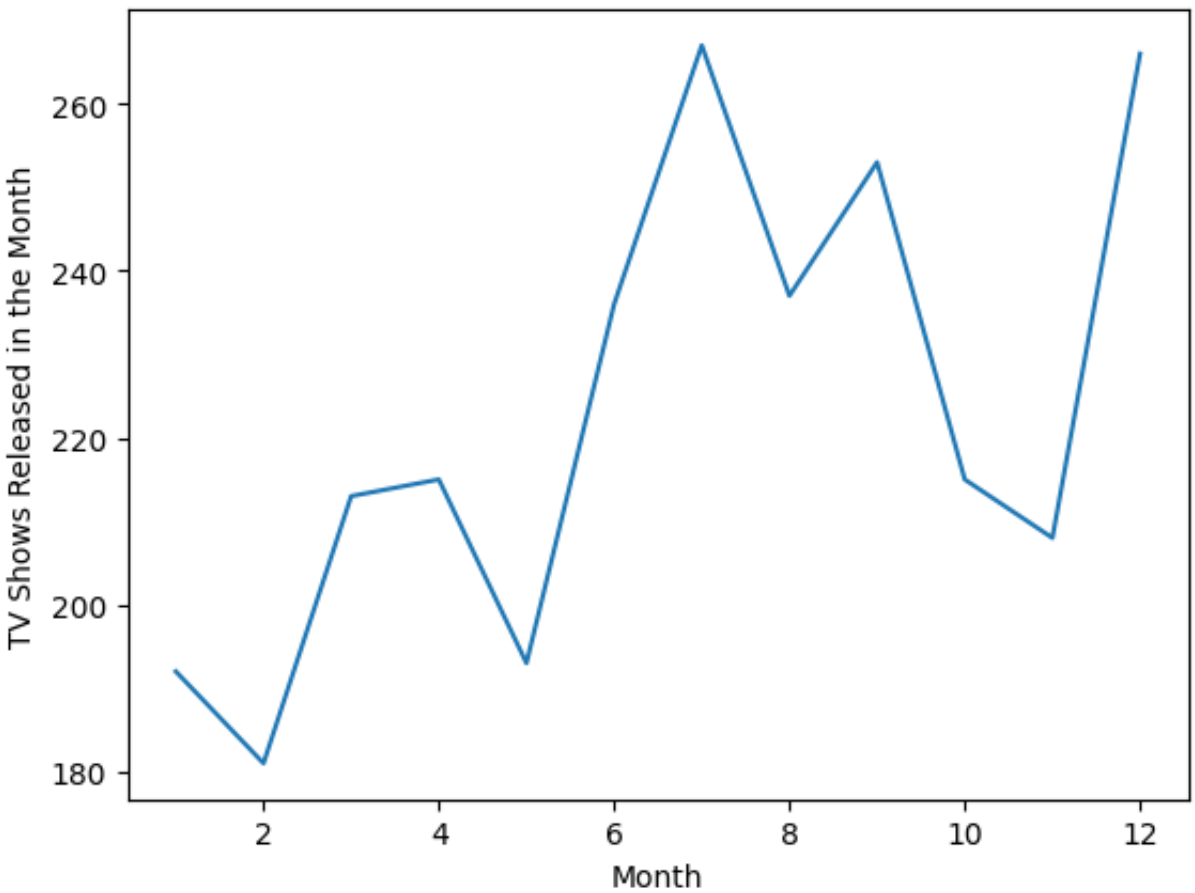
```
#Trend of Movies released in Netflix over the years
df_movie_year=movie_df.groupby(['year']).agg({"title":"nunique"}).reset_index()
sns.lineplot(data=df_movie_year, x='year', y='title')
plt.ylabel("Movies Released in the Year")
plt.xlabel("Year")
plt.show()
```



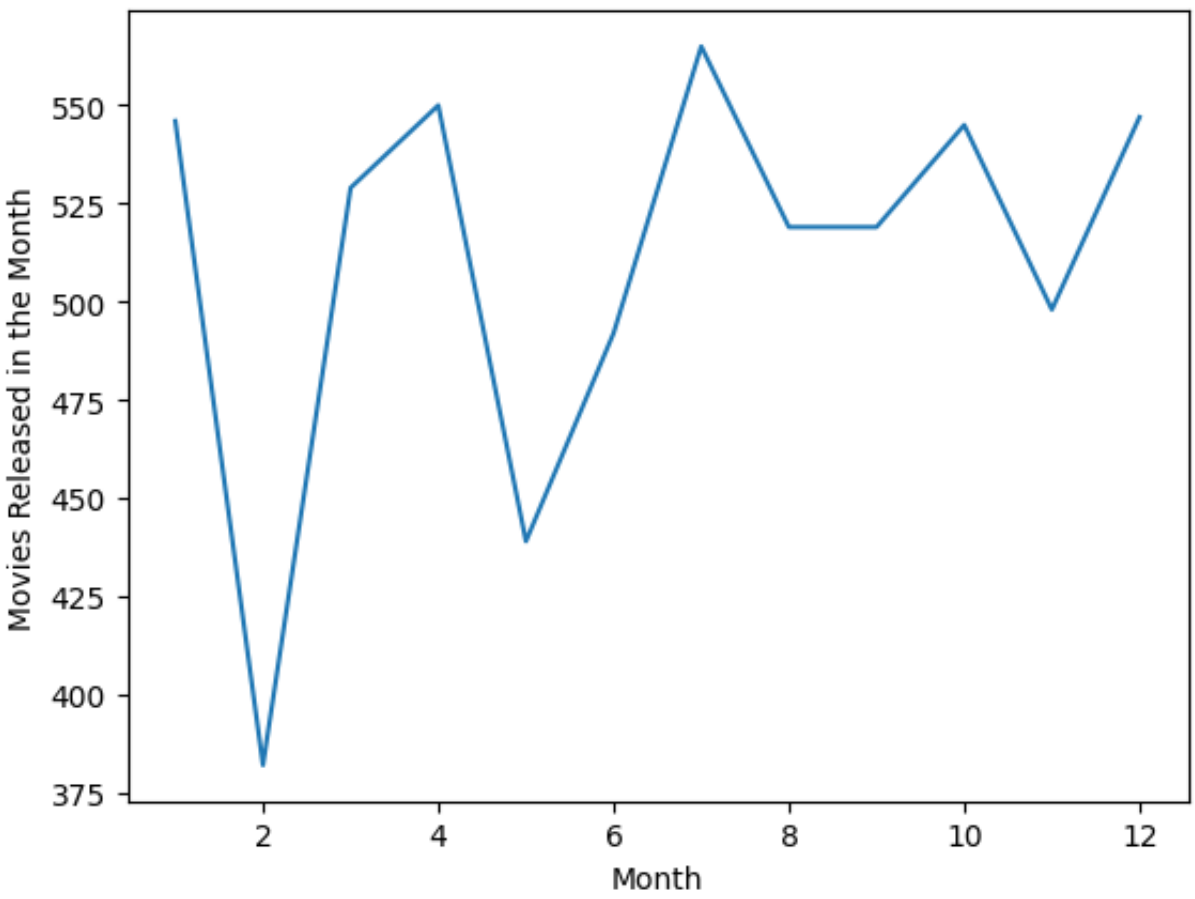
```
#Trend of shows released in a week in Netflix
tv_df = df[df['type'] == 'TV Show'].copy()

df_week=tv_df.groupby(['week_Added']).agg({"title":"nunique"}).reset_index()
plt.figure(figsize=(15,8))
sns.lineplot(data=df_week, x='week_Added', y='title')
plt.ylabel("Shows Released in the Week")
plt.xlabel("Week No.")
plt.show()
```



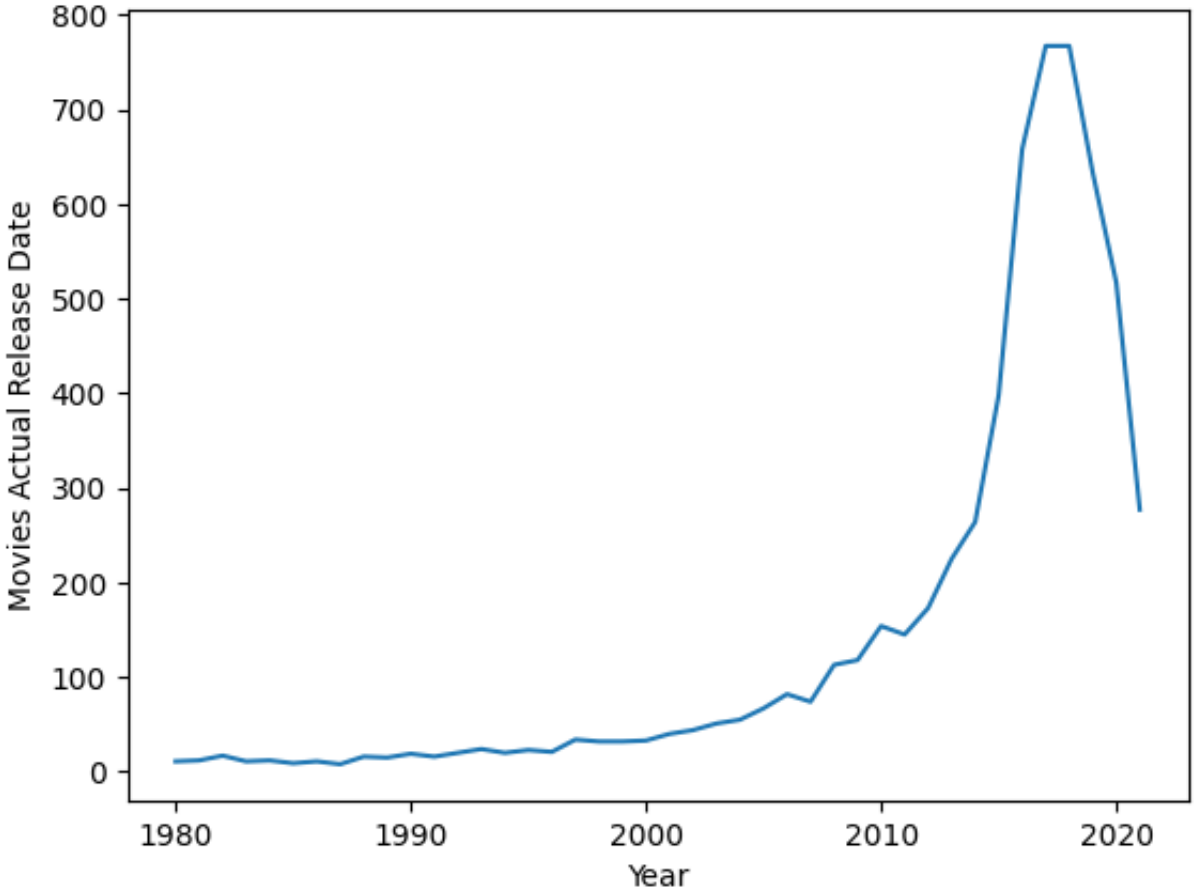
```
#Trend of shows released in a month in Netflix
df_tv_month=tv_df.groupby(['month_added']).agg({"title":"nunique"}).reset_index
sns.lineplot(data=df_tv_month, x='month_added', y='title')
plt.ylabel("TV Shows Released in the Month")
plt.xlabel("Month")
plt.show()
```



```
#Trend of movies released in a month in Netflix

df_movie_month=movie_df.groupby(['month_added']).agg({"title":"nunique"}).reset
sns.lineplot(data=df_movie_month, x='month_added', y='title')
plt.ylabel("Movies Released in the Month")
plt.xlabel("Month")
plt.show()
```



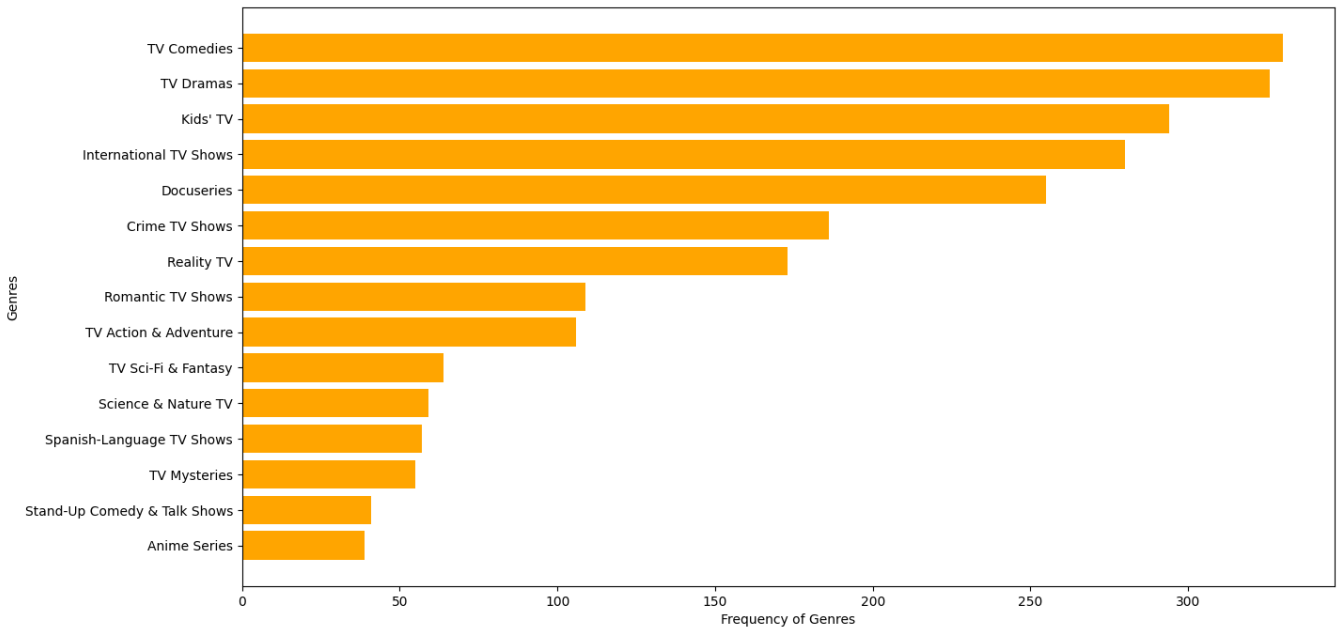
```
df_release_year=movie_df[movie_df['release_year']>=1980].groupby(['release_year
sns.lineplot(data=df_release_year, x='release_year', y='title')
plt.ylabel("Movies Actual Release Date")
plt.xlabel("Year")
plt.show()
```




```
#below countries will be analyzed for both shows and movies
shows_and_movies=['United States','India','United Kingdom']
#below countries will be only analyzed on basis of shows
only_shows=['Japan','South Korea']
```

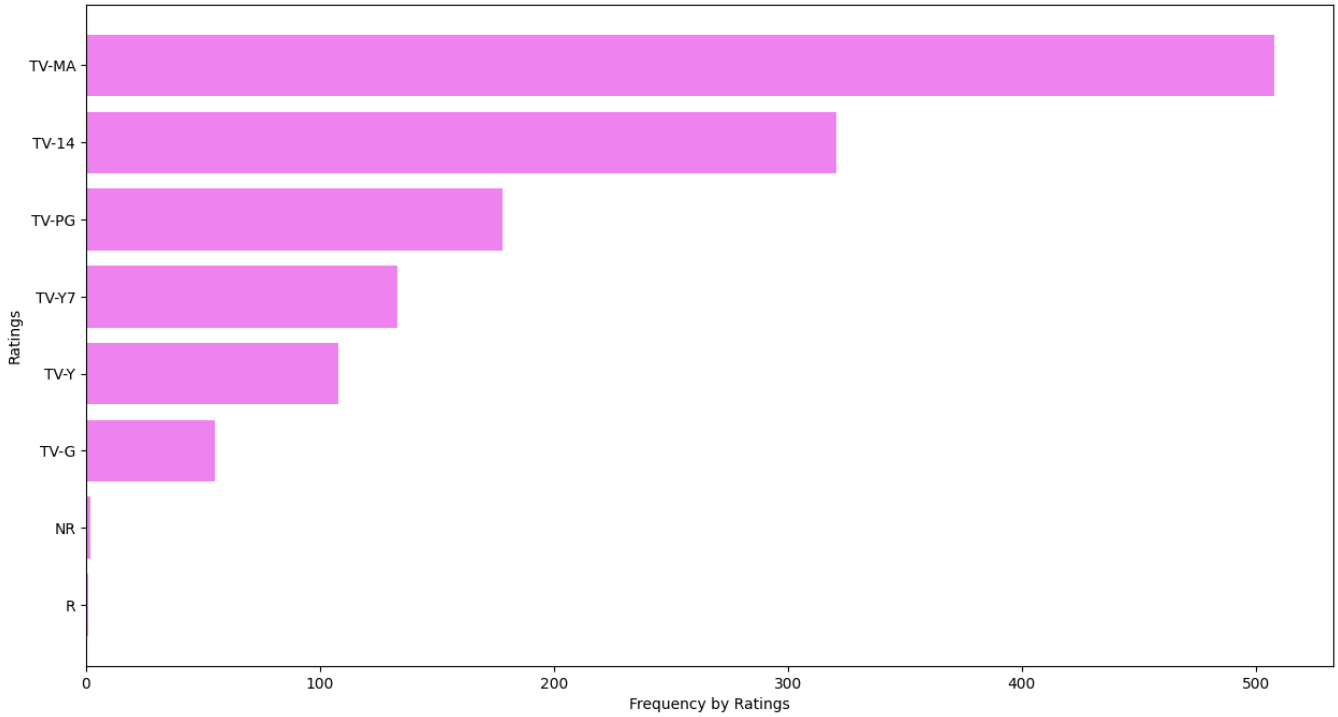
```
#Analyzing USA for both shows and movies
df_usa_shows=df[df['country']=='United States'][df[df['country']=='United State
df_usa_movies=df[df['country']=='United States'][df[df['country']=='United Stat
```

```
df_genre=df_usa_shows.groupby(['genre']).agg({"title":"nunique"}).reset_index()
plt.figure(figsize=(15,8))
plt.barh(df_genre[::-1]['genre'], df_genre[::-1]['title'],color=['orange'])
plt.xlabel('Frequency of Genres')
plt.ylabel('Genres')
plt.show()
```



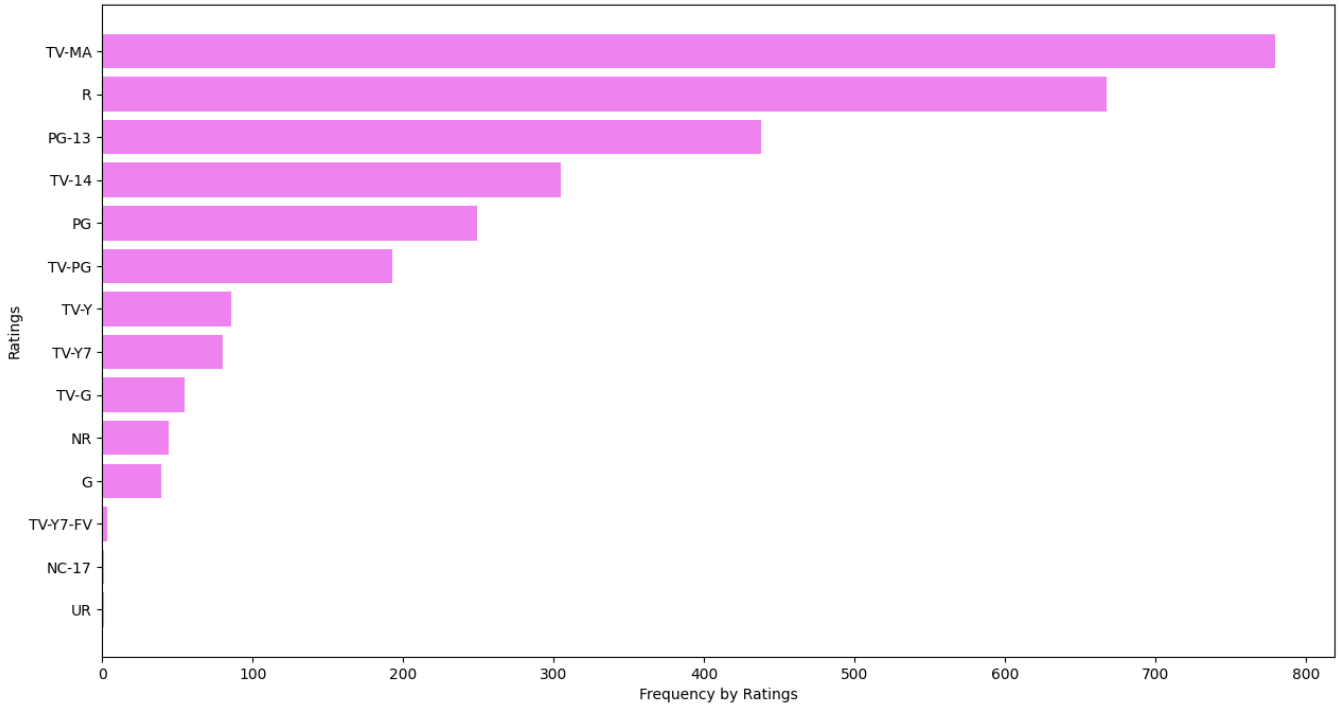
In US TV Shows, TV Comedies and TV Dramas genres are the most added.

```
df_rating=df_usa_shows.groupby(['rating']).agg({"title":"nunique"}).reset_index
plt.figure(figsize=(15,8))
plt.barh(df_rating[::-1]['rating'], df_rating[::-1]['title'],color=['violet'])
plt.xlabel('Frequency by Ratings')
plt.ylabel('Ratings')
plt.show()
```



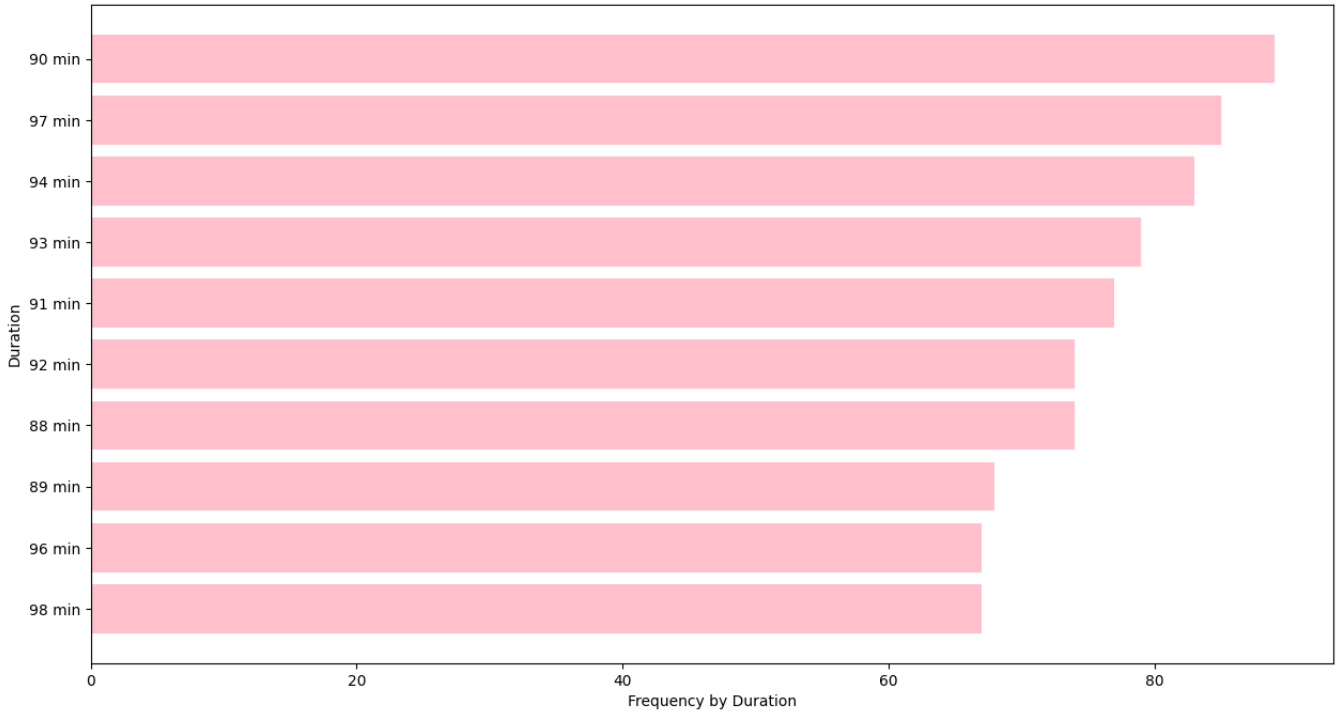
TV shows are the most rated as 'TV-MA' in US TV Shows, close to 500 shows.

```
df_rating=df_usa_movies.groupby(['rating']).agg({"title":"nunique"}).reset_inde
plt.figure(figsize=(15,8))
plt.barh(df_rating[::-1]['rating'], df_rating[::-1]['title'],color=['violet'])
plt.xlabel('Frequency by Ratings')
plt.ylabel('Ratings')
plt.show()
```



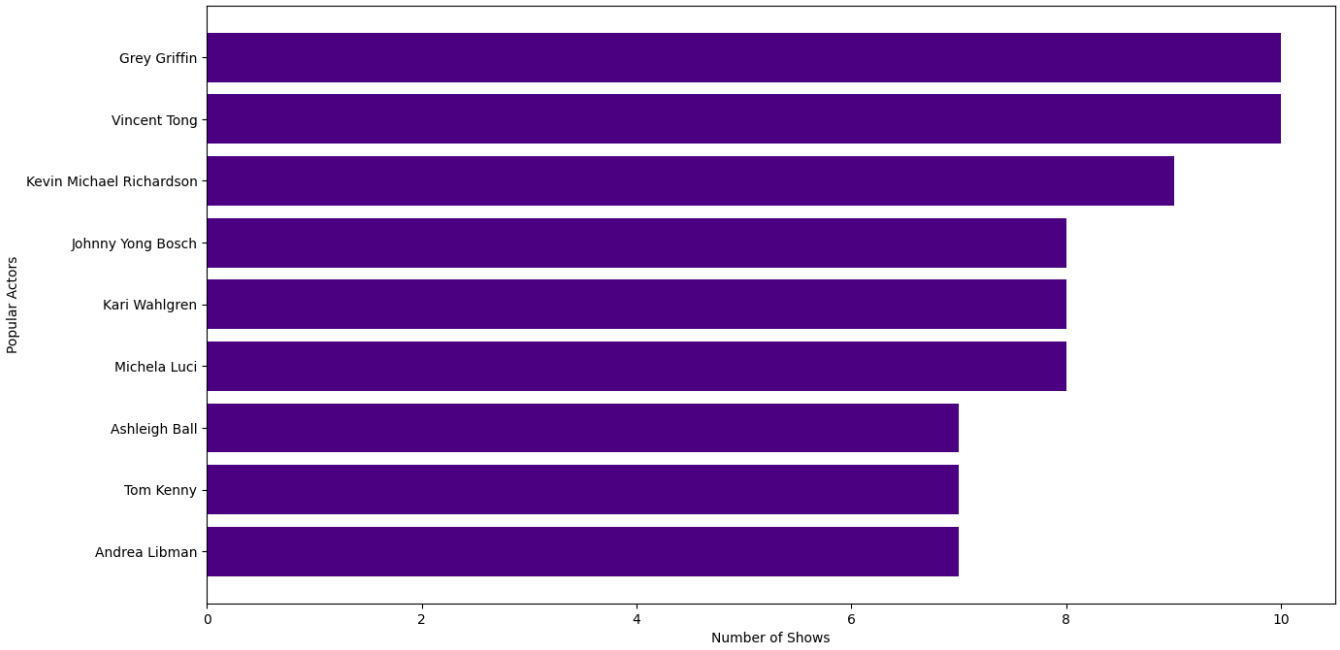
In USA, most TV shows are rated as TV-MA.

```
df_duration=df_usa_movies.groupby(['duration']).agg({"title":"nunique"}).reset_
plt.figure(figsize=(15,8))
plt.barh(df_duration[::-1]['duration'], df_duration[::-1]['title'],color=['pink
plt.xlabel('Frequency by Duration')
plt.ylabel('Duration')
plt.show()
```



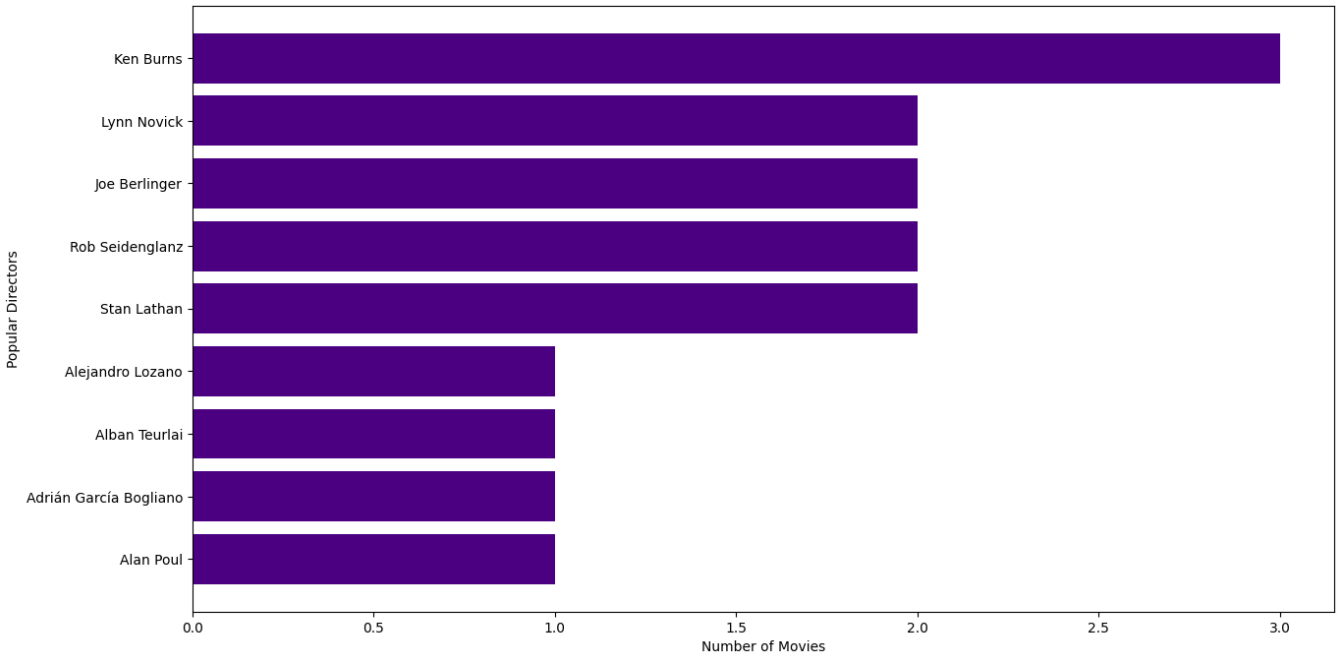
USA population prefer to watch movies whose duration is around ** 90 mins**

```
df_actors=df_usa_shows.groupby(['actors']).agg({"title":"nunique"}).reset_index
df_actors=df_actors[df_actors['actors']!='Unknown Actor']
plt.figure(figsize=(15,8))
plt.barh(df_actors[::-1]['actors'], df_actors[::-1]['title'],color=['indigo'])
plt.xlabel('Number of Shows')
plt.ylabel('Popular Actors')
plt.show()
```



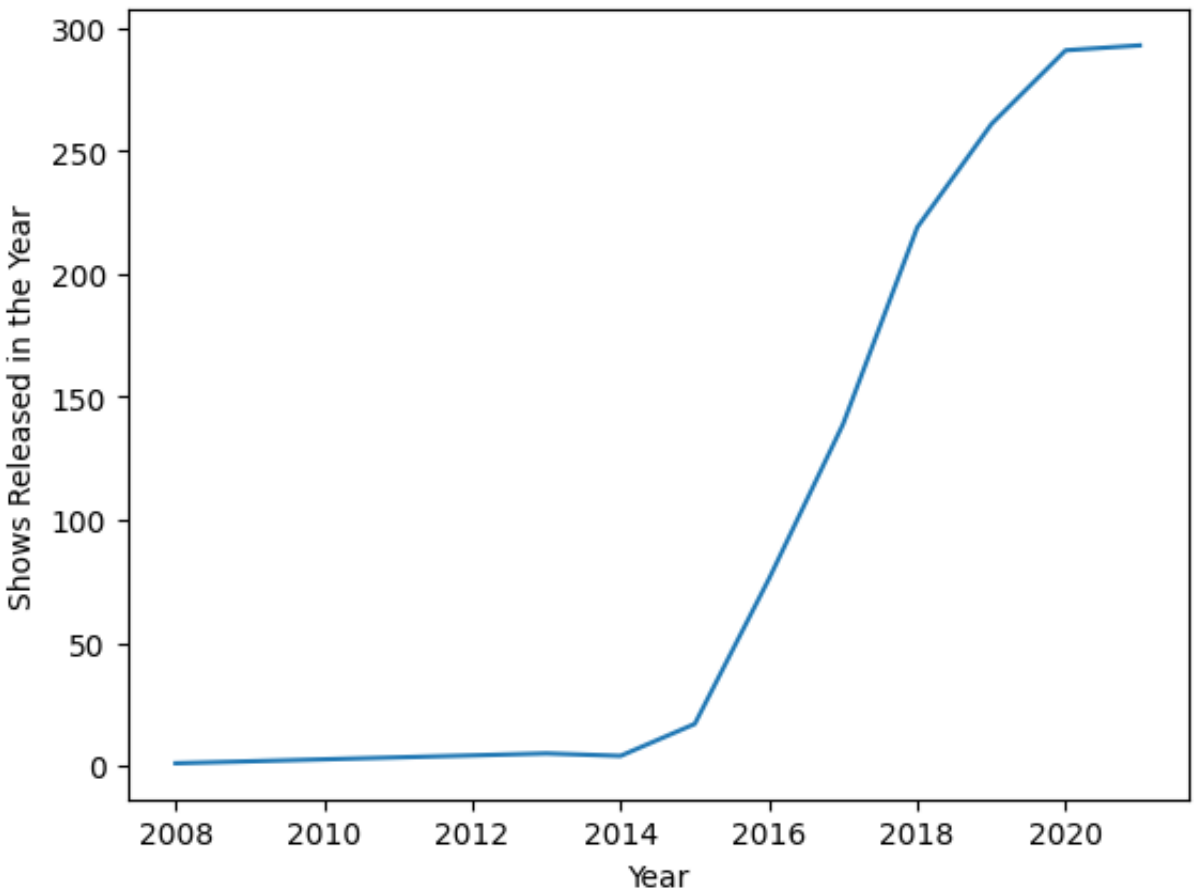
Grey Griffin has acted in most no of US shows.

```
df_directors=df_usa_shows.groupby(['directors']).agg({"title":"nunique"}).reset
df_directors=df_directors[df_directors['directors']!='Unknown Director']
plt.figure(figsize=(15,8))
plt.barh(df_directors[::-1]['directors'], df_directors[::-1]['title'],color=['i
plt.xlabel('Number of Movies')
plt.ylabel('Popular Directors')
plt.show()
```

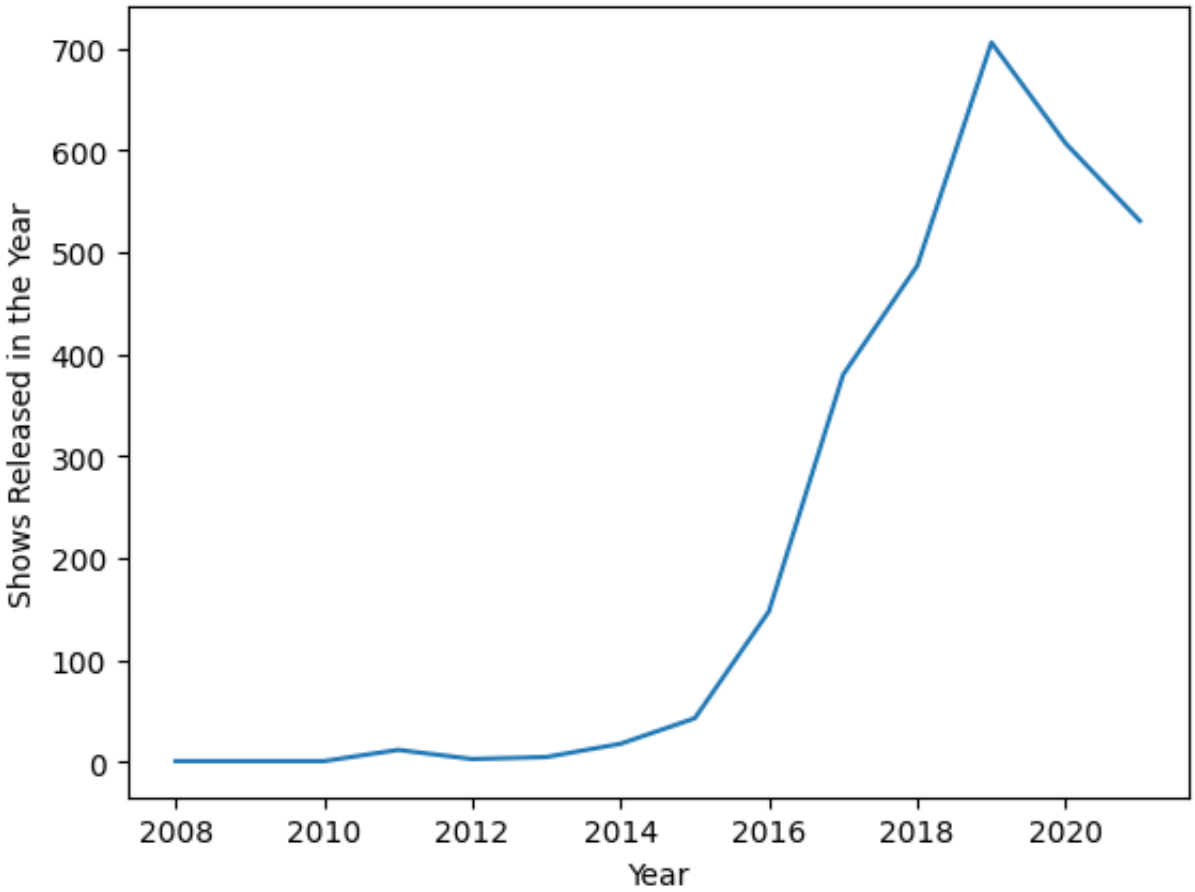


Ken Burns has directed most of the shows in USA.

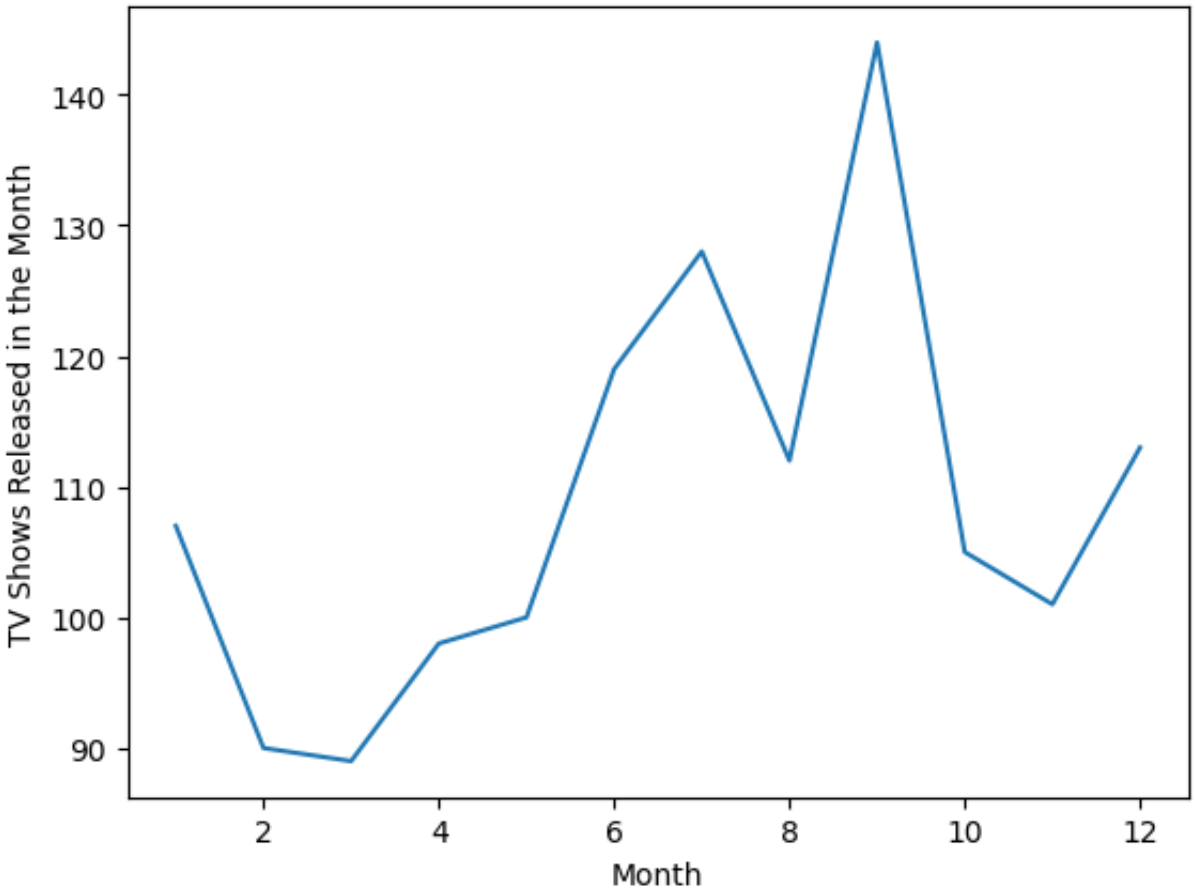
```
#Shows released in yearly in USA
df_year=df_usa_shows.groupby(['year']).agg({"title":"nunique"}).reset_index()
sns.lineplot(data=df_year, x='year', y='title')
plt.ylabel("Shows Released in the Year")
plt.xlabel("Year")
plt.show()
```



```
df_year=df_usa_movies.groupby(['year']).agg({"title":"nunique"}).reset_index()
sns.lineplot(data=df_year, x='year', y='title')
plt.ylabel("Shows Released in the Year")
plt.xlabel("Year")
plt.show()
```



```
df_month=df_usa_shows.groupby(['month_added']).agg({"title":"nunique"}).reset_i
sns.lineplot(data=df_month, x='month_added', y='title')
plt.ylabel("TV Shows Released in the Month")
plt.xlabel("Month")
plt.show()
```



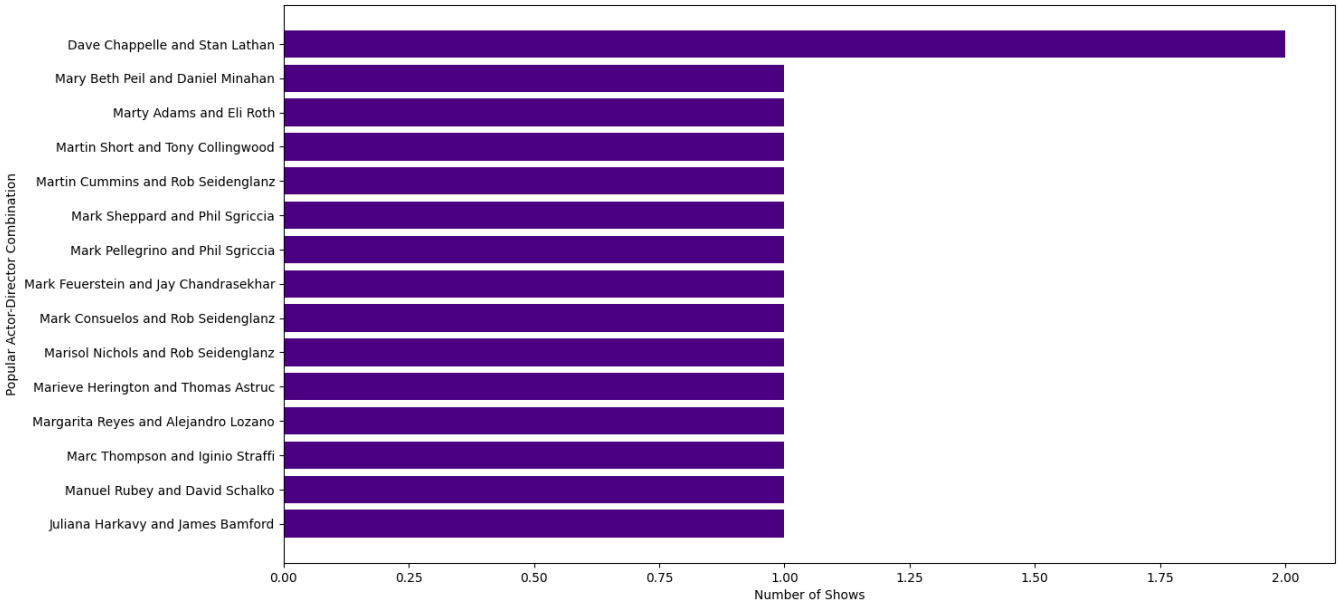
```
#Analysing a combination of actors and directors
df_usa_movies['Actor_Director_Combination'] = df_usa_movies.actors.str.cat(df_u
df_usa_movies_subset=df_usa_movies[df_usa_movies['actors']!='Unknown Actor']
df_usa_movies_subset=df_usa_movies_subset[df_usa_movies_subset['directors']!='U
df_usa_movies_subset.head()
```

	show_id	type	title	directors	actors	country	date_added	release_
6	s7	Movie	My Little Pony: A New Generation	Robert Cullen	Vanessa Hudgens	United States	September 24, 2021	
6	s7	Movie	My Little Pony: A New Generation	Robert Cullen	Kimiko Glenn	United States	September 24, 2021	
6	s7	Movie	My Little Pony: A New Generation	Robert Cullen	James Marsden	United States	September 24, 2021	
6	s7	Movie	My Little Pony: A New Generation	Robert Cullen	Liza Koshy	United States	September 24, 2021	
6	s7	Movie	My Little Pony: A New Generation	Robert Cullen	Ken Jeong	United States	September 24, 2021	

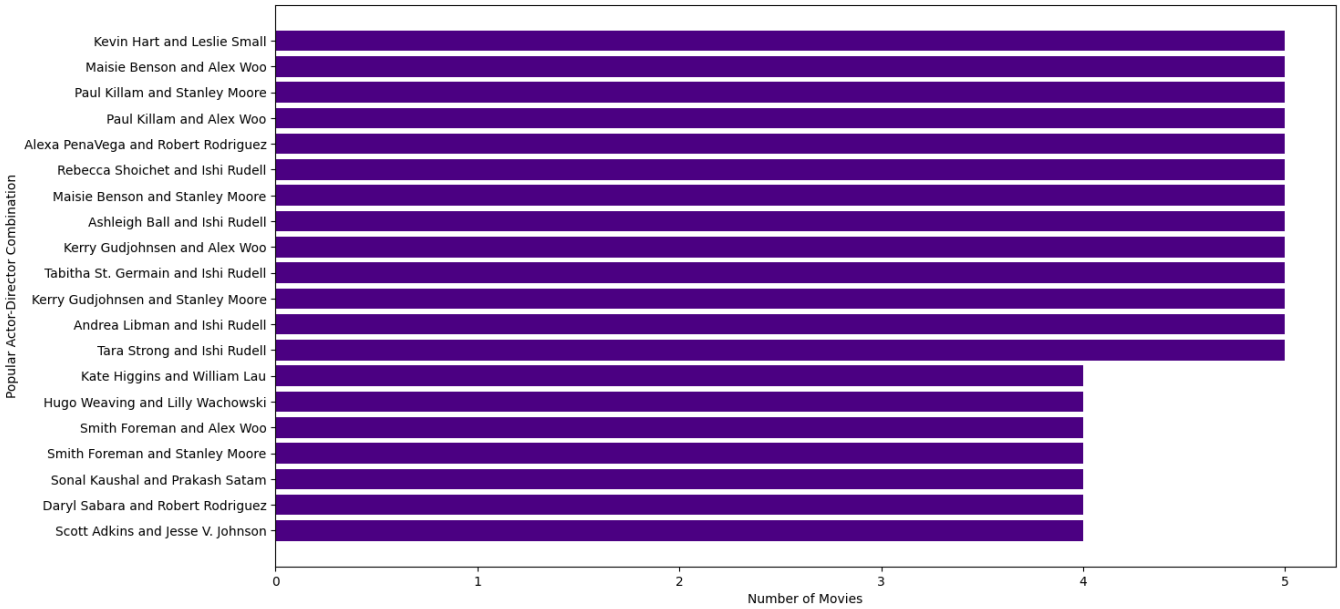
```
df_usa_shows['Actor_Director_Combination'] = df_usa_shows.actors.str.cat(df_usa
df_usa_shows_subset=df_usa_shows[df_usa_shows['actors']!='Unknown Actor']
df_usa_shows_subset=df_usa_shows_subset[df_usa_shows_subset['directors']!='Unkr
df_usa_shows_subset.head()
```

	show_id	type	title	directors	actors	country	date_added	release_yea
5	s6	TV Show	Midnight Mass	Mike Flanagan	Kate Siegel	United States	September 24, 2021	202
5	s6	TV Show	Midnight Mass	Mike Flanagan	Kate Siegel	United States	September 24, 2021	202
5	s6	TV Show	Midnight Mass	Mike Flanagan	Kate Siegel	United States	September 24, 2021	202
5	s6	TV Show	Midnight Mass	Mike Flanagan	Zach Gilford	United States	September 24, 2021	202
5	s6	TV Show	Midnight Mass	Mike Flanagan	Zach Gilford	United States	September 24, 2021	202

```
df_actors_directors=df_usa_shows_subset.groupby(['Actor_Director_Combination'])
plt.figure(figsize=(15,8))
plt.barh(df_actors_directors[::-1]['Actor_Director_Combination'], df_actors_dir
plt.xlabel('Number of Shows')
plt.ylabel('Popular Actor-Director Combination')
plt.show()
```



```
df_actors_directors=df_usa_movies_subset.groupby(['Actor_Director_Combination']
plt.figure(figsize=(15,8))
plt.barh(df_actors_directors[::-1]['Actor_Director_Combination'], df_actors_dir
plt.xlabel('Number of Movies')
plt.ylabel('Popular Actor-Director Combination')
plt.show()
```

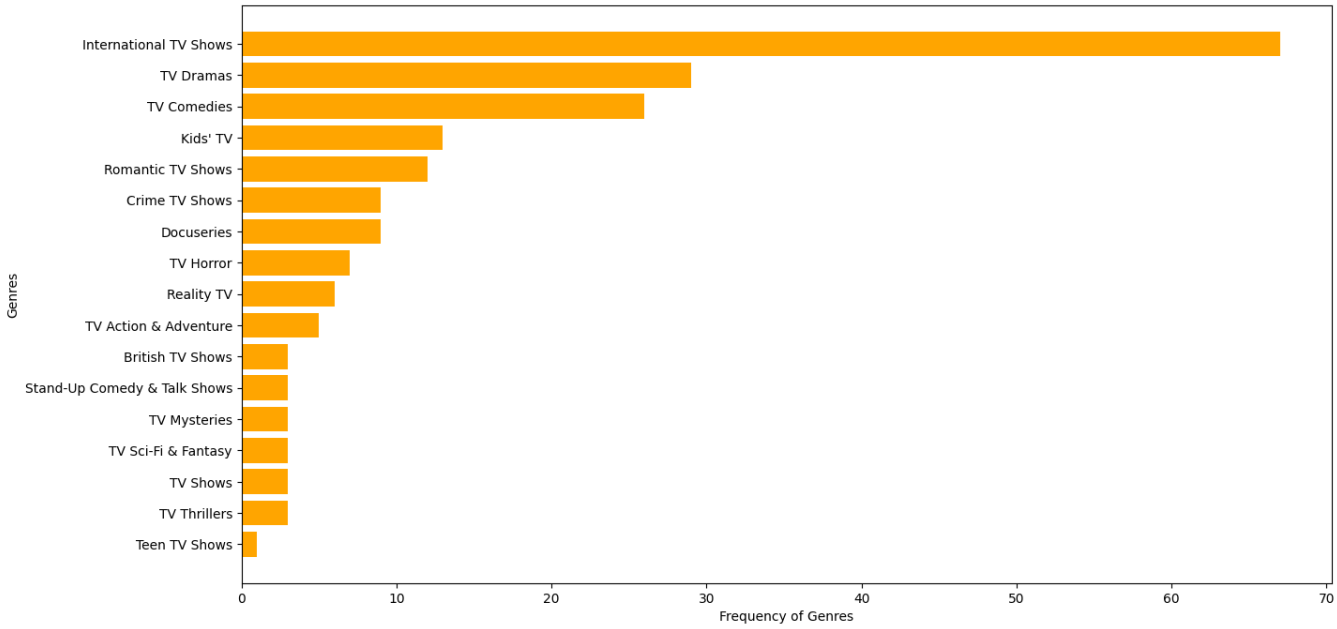


```
df_actors_directors[::-1]['Actor_Director_Combination'].values
```

```
array(['Scott Adkins and Jesse V. Johnson',
       'Daryl Sabara and Robert Rodriguez',
       'Sonal Kaushal and Prakash Satam',
       'Smith Foreman and Stanley Moore', 'Smith Foreman and Alex Woo',
       'Hugo Weaving and Lilly Wachowski', 'Kate Higgins and William Lau',
       'Tara Strong and Ishi Rudell', 'Andrea Libman and Ishi Rudell',
       'Kerry Gudjohnsen and Stanley Moore',
       'Tabitha St. Germain and Ishi Rudell',
       'Kerry Gudjohnsen and Alex Woo', 'Ashleigh Ball and Ishi Rudell',
       'Maisie Benson and Stanley Moore',
       'Rebecca Shoichet and Ishi Rudell',
       'Alexa PenaVega and Robert Rodriguez', 'Paul Killam and Alex Woo',
       'Paul Killam and Stanley Moore', 'Maisie Benson and Alex Woo',
       'Kevin Hart and Leslie Small'], dtype=object)
```

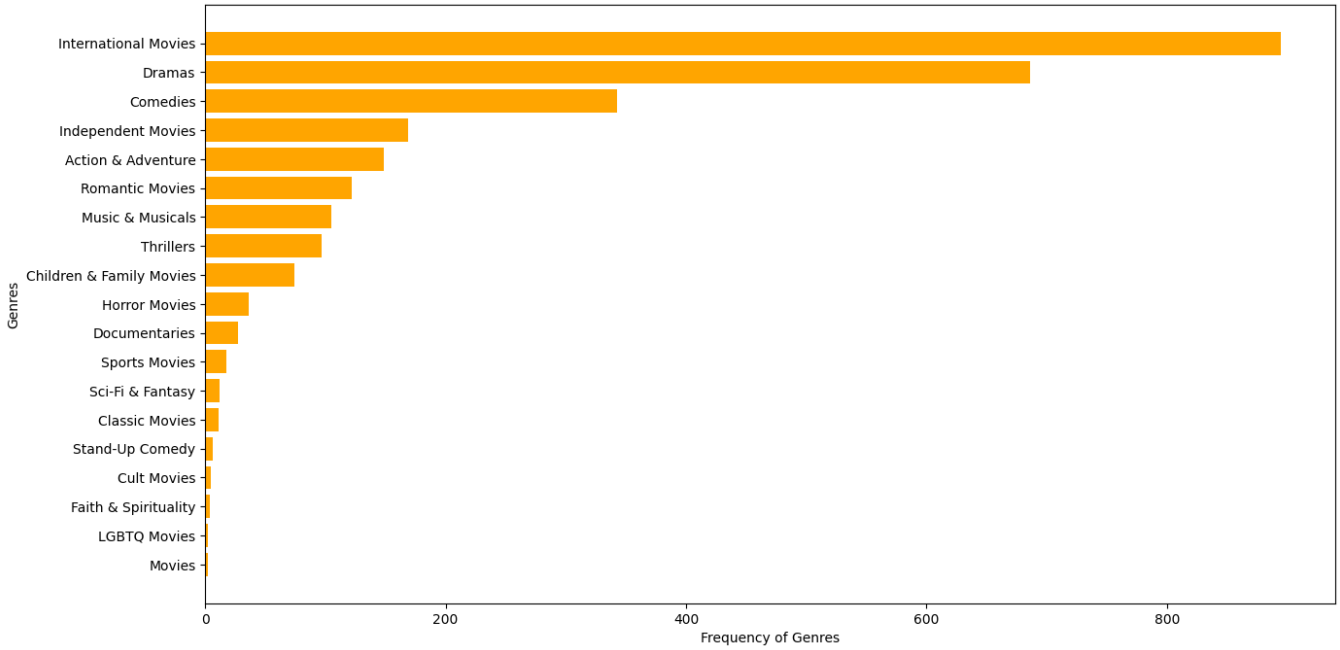
```
#Analyzing India for both shows and movies
df_india_shows = df[df['country']=='India'] [df[df['country']=='India']['type']
df_india_movies = df[df['country']=='India'] [df[df['country']=='India']['type'
```

```
#Top genre tv shows in india
df_india_shows_genre = df_india_shows.groupby(['genre']).agg({'title':'nunique'

plt.figure(figsize=(15,8))
plt.barh(df_india_shows_genre[::-1]['genre'], df_india_shows_genre[::-1]['title
plt.xlabel('Frequency of Genres')
plt.ylabel('Genres')
plt.show()
```




```
#Top genre movies in india
#Top genre tv shows in india
df_india_movie_genre = df_india_movies.groupby(['genre']).agg({'title':'nunique
```

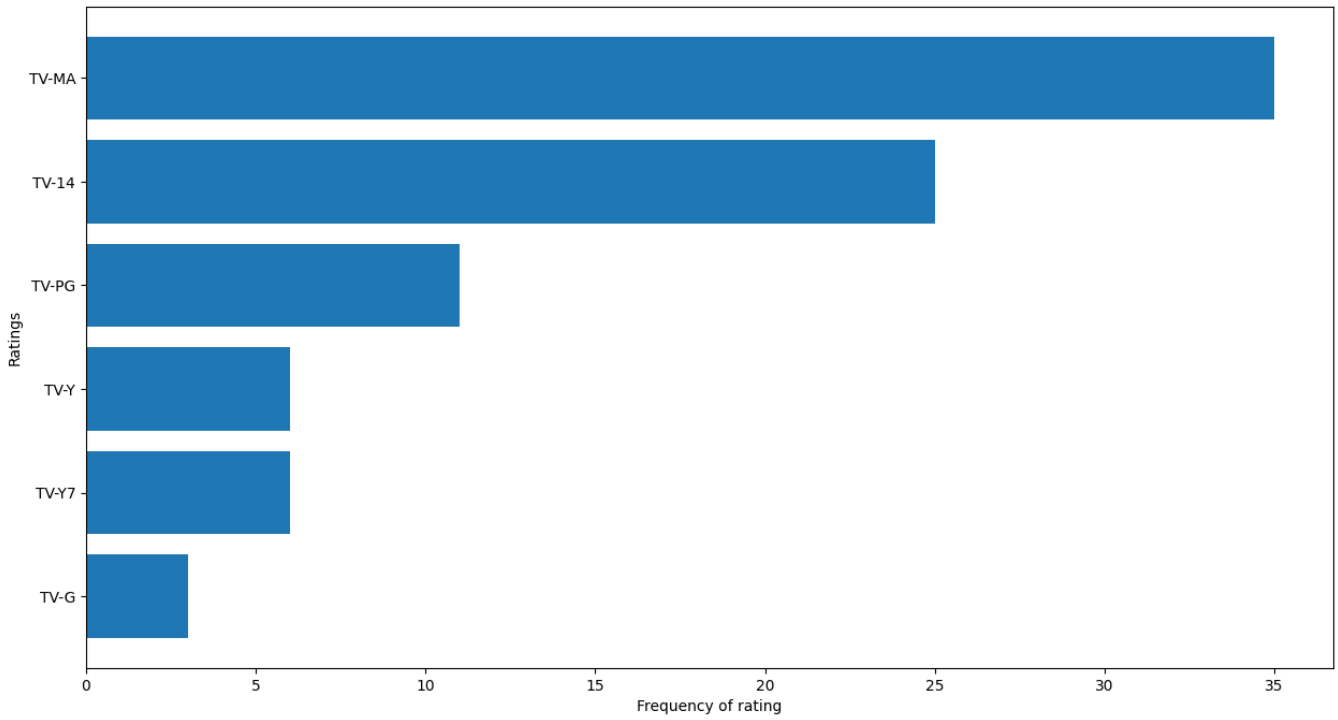
```
plt.figure(figsize=(15,8))
plt.barh(df_india_movie_genre[::-1]['genre'], df_india_movie_genre[::-1]['title
plt.xlabel('Frequency of Genres')
plt.ylabel('Genres')
plt.show()
```




```
df_india_shows_rating = df_india_shows.groupby(['rating']).agg({'title':'nuniqu
```

```
plt.figure(figsize=(15,8))
plt.barh( df_india_shows_rating[::-1]['rating'], df_india_shows_rating[::-1]['t
plt.xlabel('Frequency of rating')
plt.ylabel('Ratings')
```

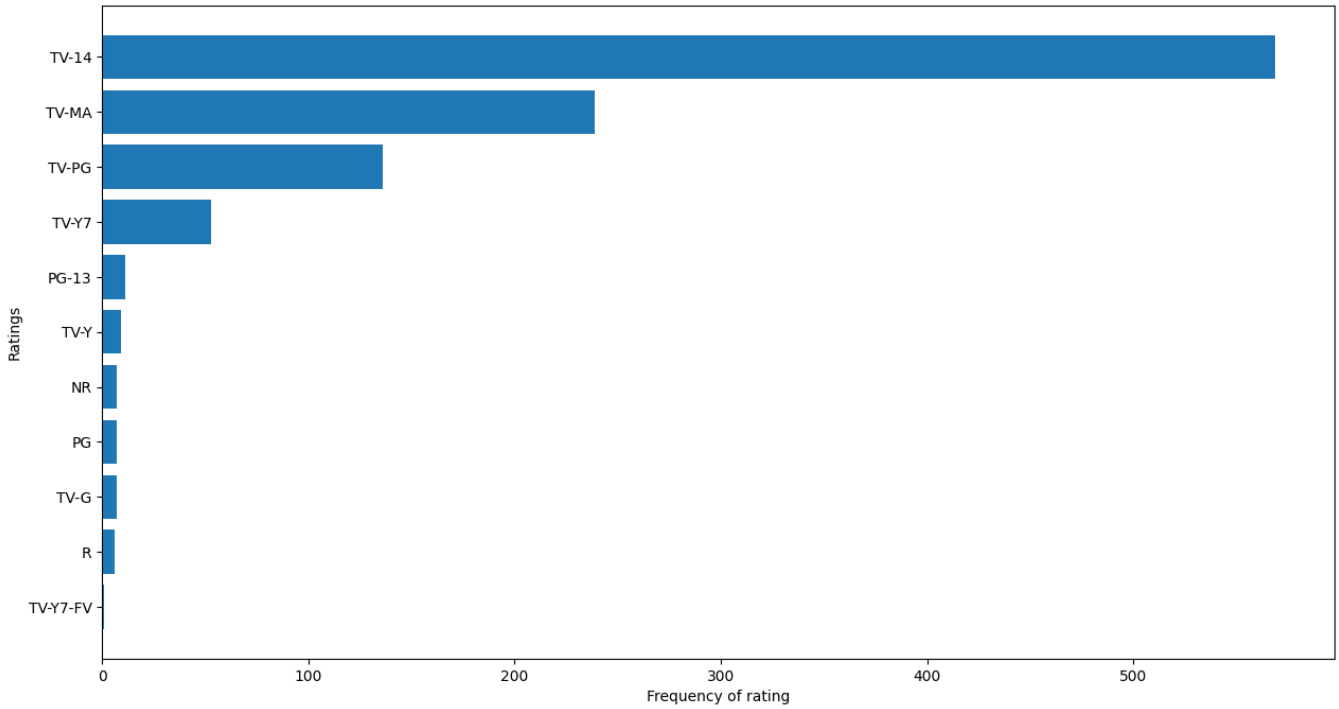
```
Text(0, 0.5, 'Ratings')
```




```
df_india_movies_rating = df_india_movies.groupby(['rating']).agg({'title':'nuni
```

```
plt.figure(figsize=(15,8))
plt.barh( df_india_movies_rating[::-1]['rating'], df_india_movies_rating[::-1]|
plt.xlabel('Frequency of rating')
plt.ylabel('Ratings')
```

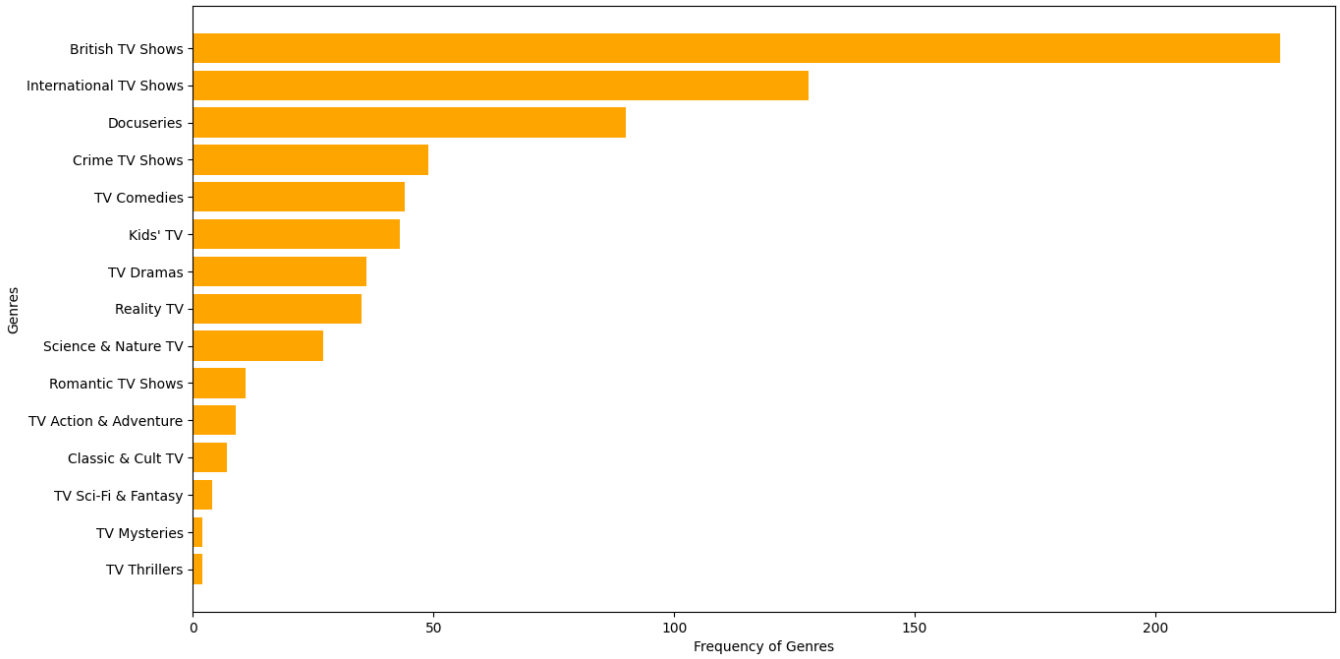
```
Text(0, 0.5, 'Ratings')
```



Lets analyze movies and shows based on genre, rating, duration, actors, directors in **UK.**

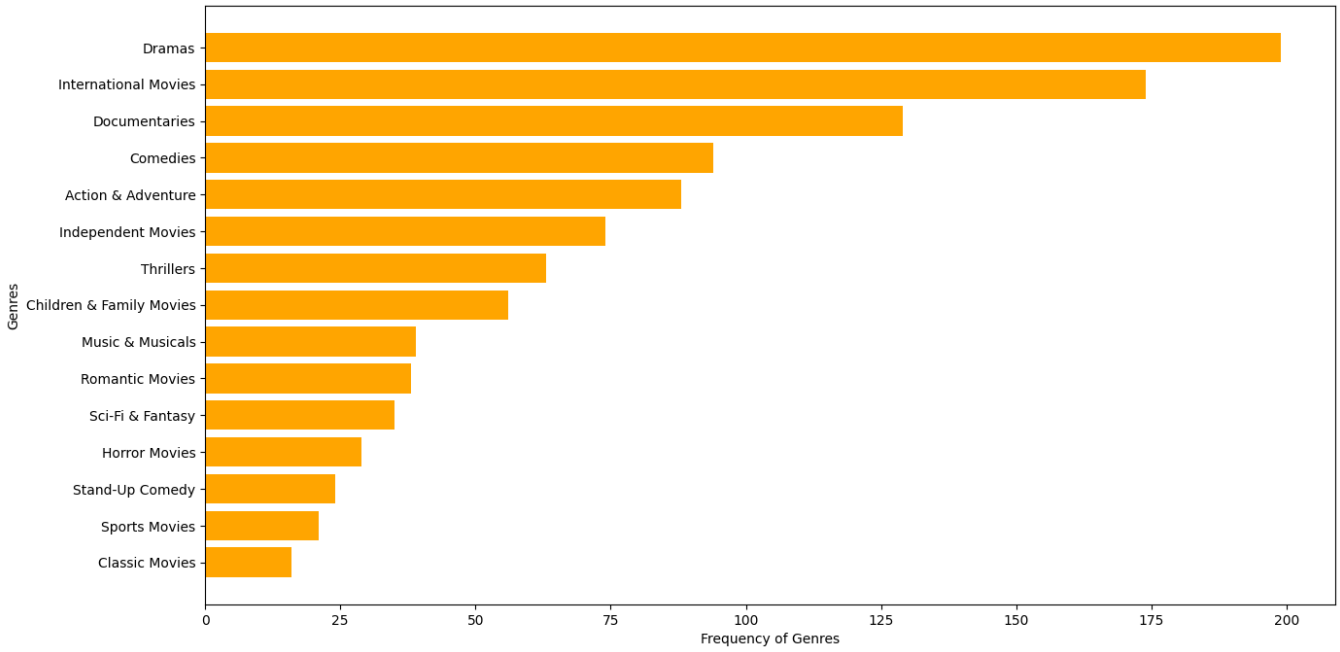
```
#Analyzing India for both shows and movies
df_uk_shows=df[df['country']=='United Kingdom'][df[df['country']=='United Kingc
df_uk_movies=df[df['country']=='United Kingdom'][df[df['country']=='United King
```

```
#UK population preference based on genre
df_genre=df_uk_shows.groupby(['genre']).agg({"title":"nunique"}).reset_index().
plt.figure(figsize=(15,8))
plt.barh(df_genre[::-1]['genre'], df_genre[::-1]['title'],color=['orange'])
plt.xlabel('Frequency of Genres')
plt.ylabel('Genres')
plt.show()
```



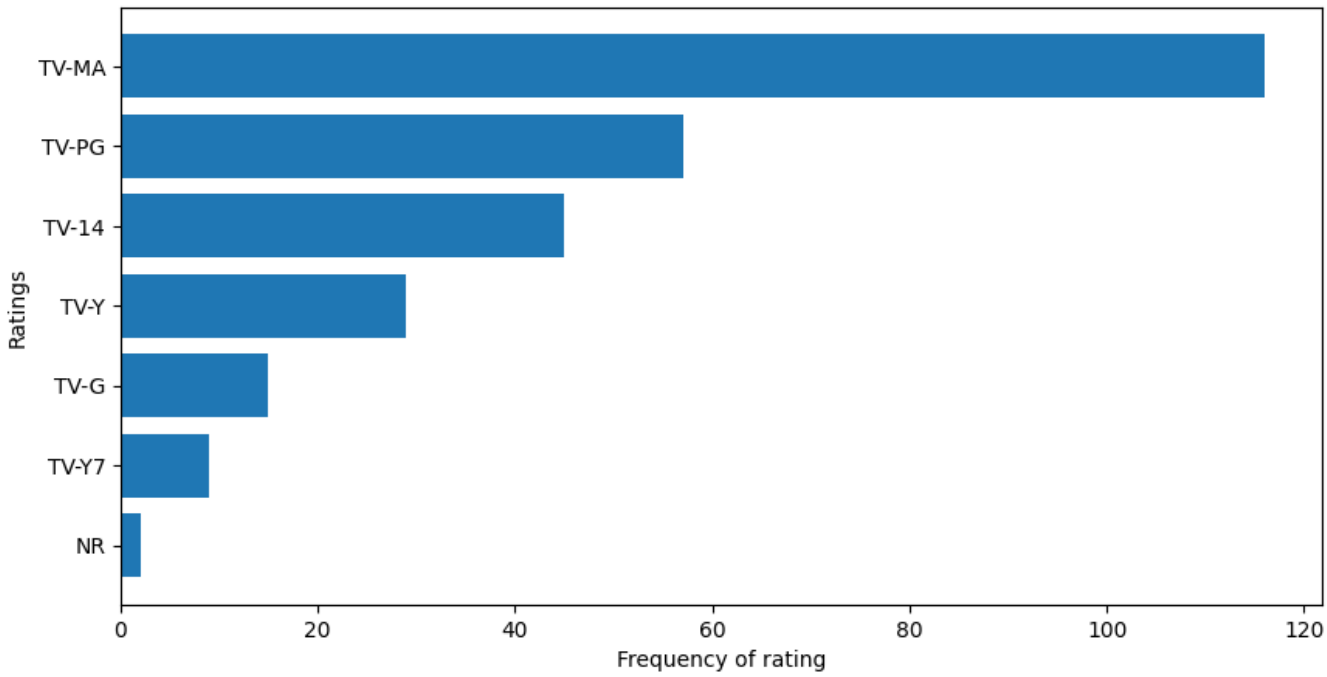
Clearly, UK population prefers British TV Shows over International TV Shows.

```
df_genre=df_uk_movies.groupby(['genre']).agg({"title":"nunique"}).reset_index()
plt.figure(figsize=(15,8))
plt.barh(df_genre[::-1]['genre'], df_genre[::-1]['title'],color=['orange'])
plt.xlabel('Frequency of Genres')
plt.ylabel('Genres')
plt.show()
```



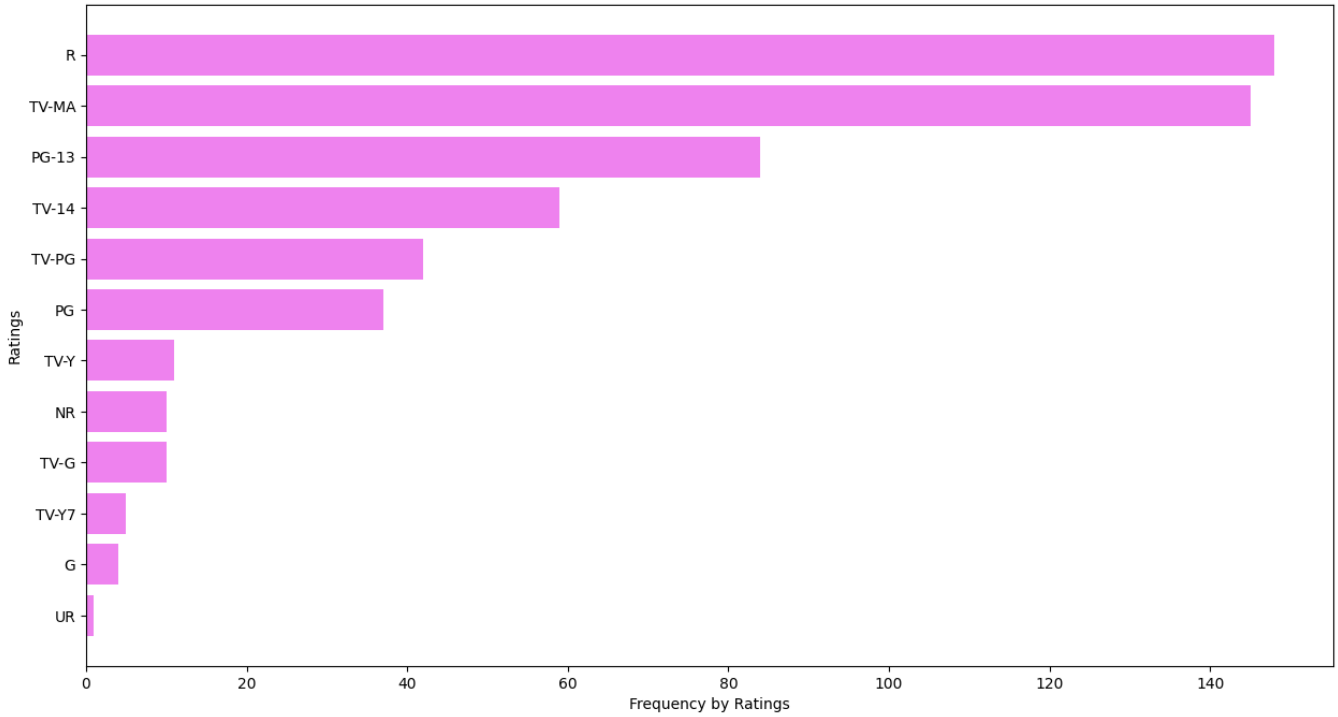
UK population is more into Drama based movies.

```
#UK's shows based on rating

df_rating = df_uk_shows.groupby(['rating']).agg({'title':'nunique'}).reset_inde
plt.figure(figsize=(10,5))
plt.barh( df_rating[::-1]['rating'], df_rating[::-1]['title'])
plt.xlabel('Frequency of rating')
plt.ylabel('Ratings')
plt.show()
```



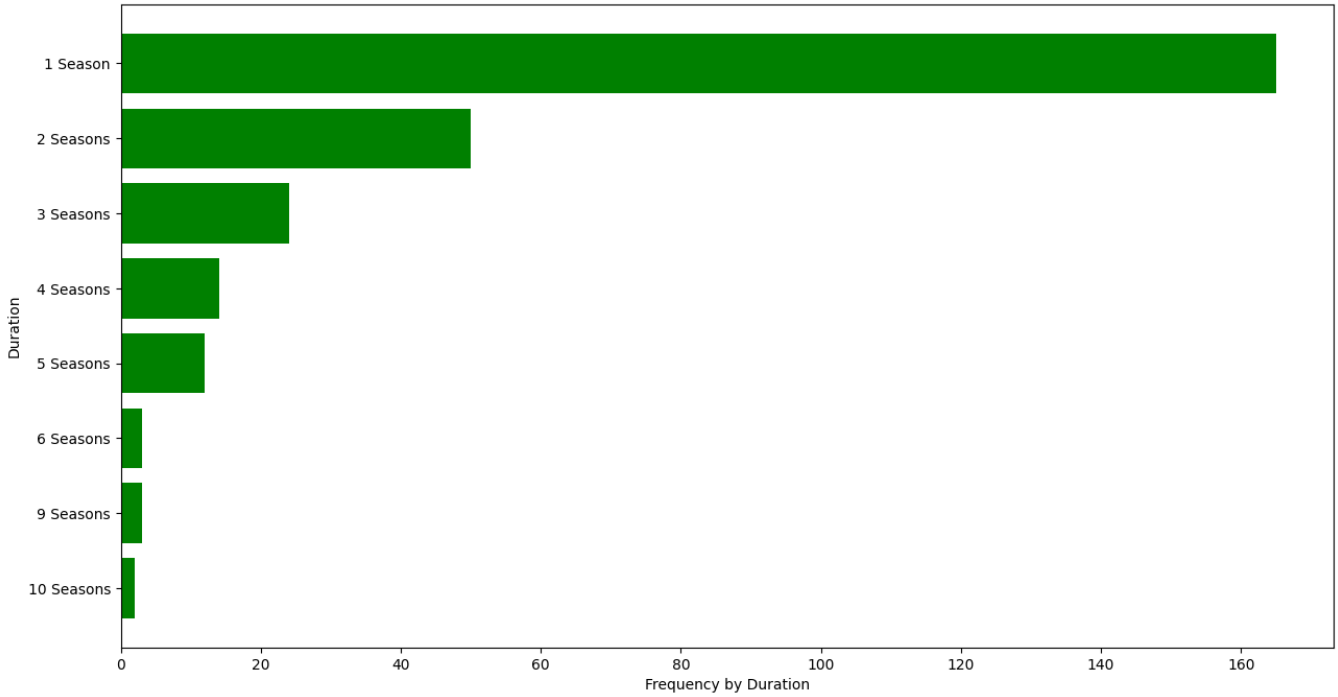
Shows in UK are mostly rated as TV-MA, it is suitable for Adults.

```
df_rating=df_uk_movies.groupby(['rating']).agg({"title":"nunique"}).reset_index
plt.figure(figsize=(15,8))
plt.barh(df_rating[::-1]['rating'], df_rating[::-1]['title'],color=['violet'])
plt.xlabel('Frequency by Ratings')
plt.ylabel('Ratings')
plt.show()
```



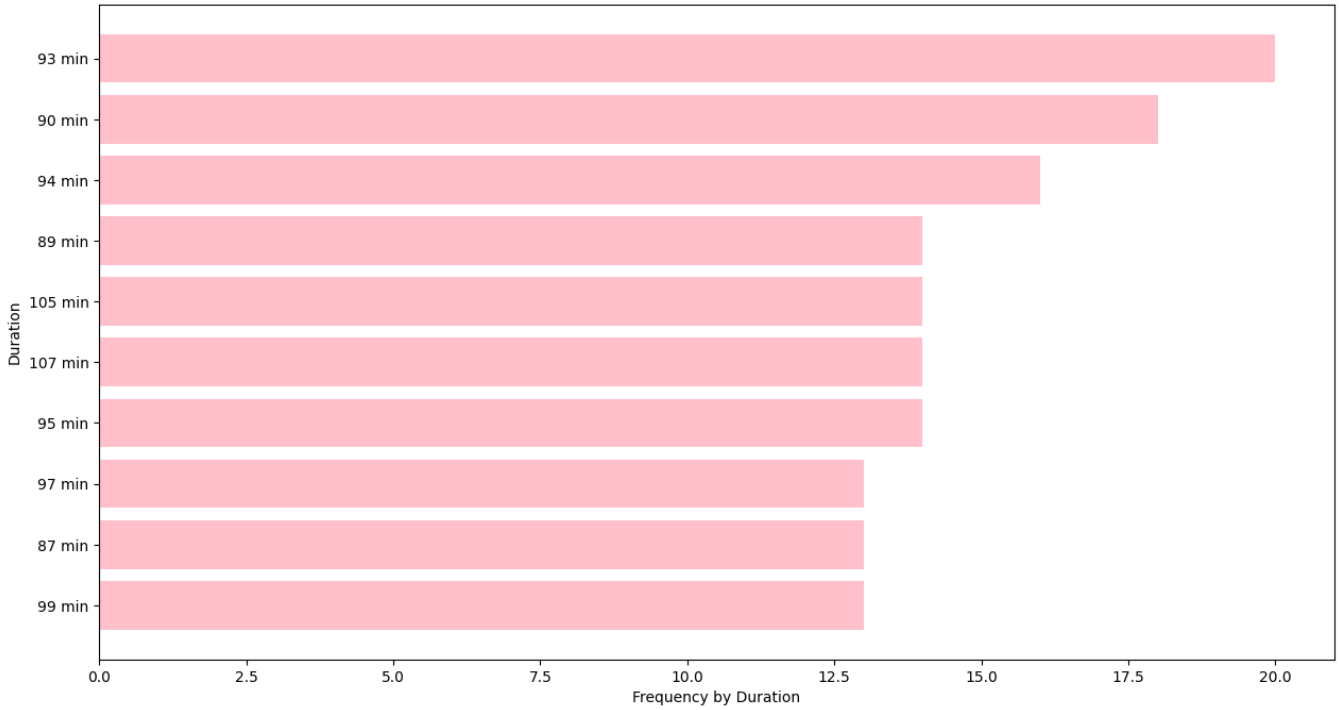
Movies in UK are mostly rated as 'R' that means, requiring accompanying parents or adult guardians for anyone under the age of 17.

```
df_duration=df_uk_shows.groupby(['duration']).agg({"title":"nunique"}).reset_ir
plt.figure(figsize=(15,8))
plt.barh(df_duration[::-1]['duration'], df_duration[::-1]['title'],color=['gree
plt.xlabel('Frequency by Duration')
plt.ylabel('Duration')
plt.show()
```



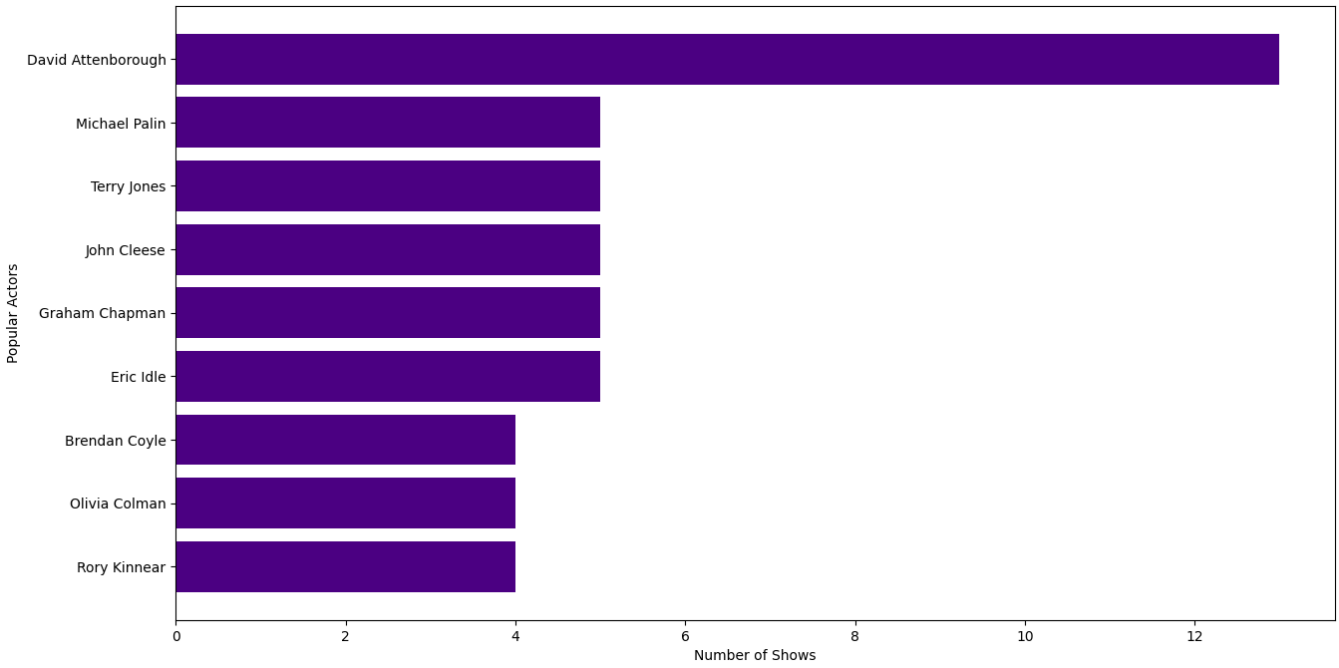
Most of the UK's popualtion can watch shows of 1 season.

```
df_duration=df_uk_movies.groupby(['duration']).agg({"title":"nunique"}).reset_i
plt.figure(figsize=(15,8))
plt.barh(df_duration[::-1]['duration'], df_duration[::-1]['title'],color=['pink
plt.xlabel('Frequency by Duration')
plt.ylabel('Duration')
plt.show()
```



Most of the UK's popualtion can watch movies of length upto 93 mins.

```
df_actors=df_uk_shows.groupby(['actors']).agg({"title":"nunique"}).reset_index(
df_actors=df_actors[df_actors['actors']!='Unknown Actor']
plt.figure(figsize=(15,8))
plt.barh(df_actors[::-1]['actors'], df_actors[::-1]['title'],color=['indigo'])
plt.xlabel('Number of Shows')
plt.ylabel('Popular Actors')
plt.show()
```

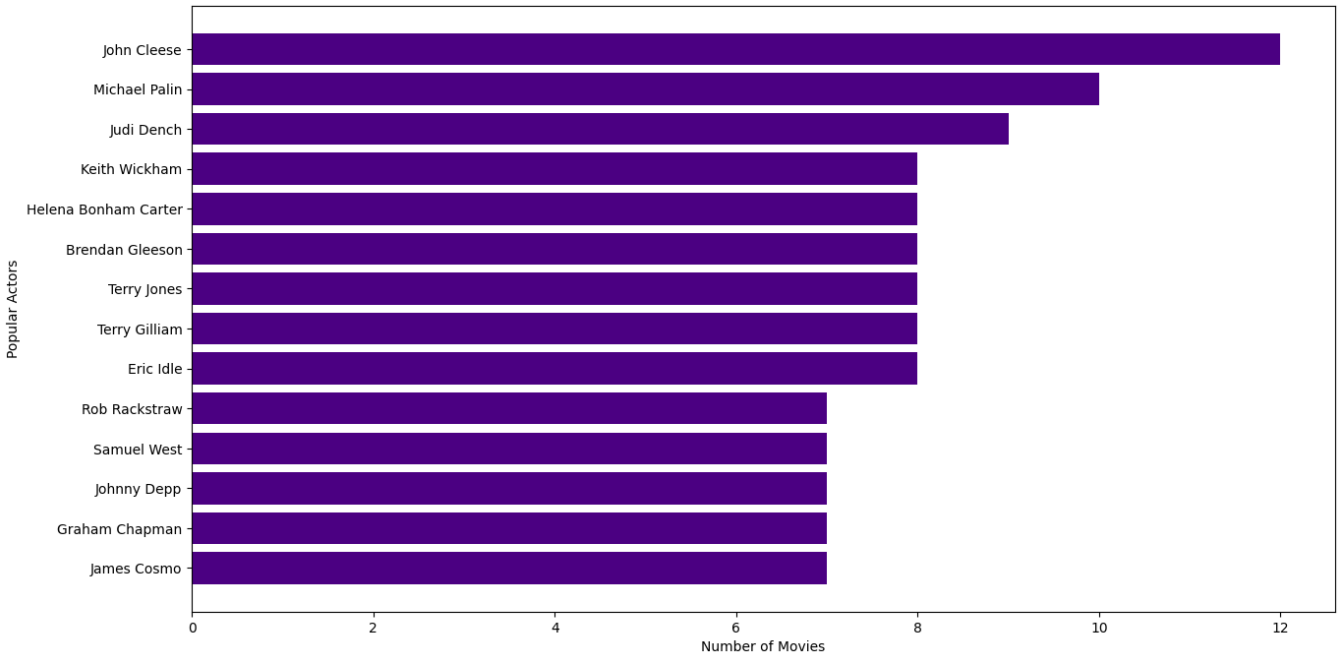


David Attenborough, one of the naturalists and most renowned and veteran has most no of Shows to his name.

```
df_actors['actors'].values
```

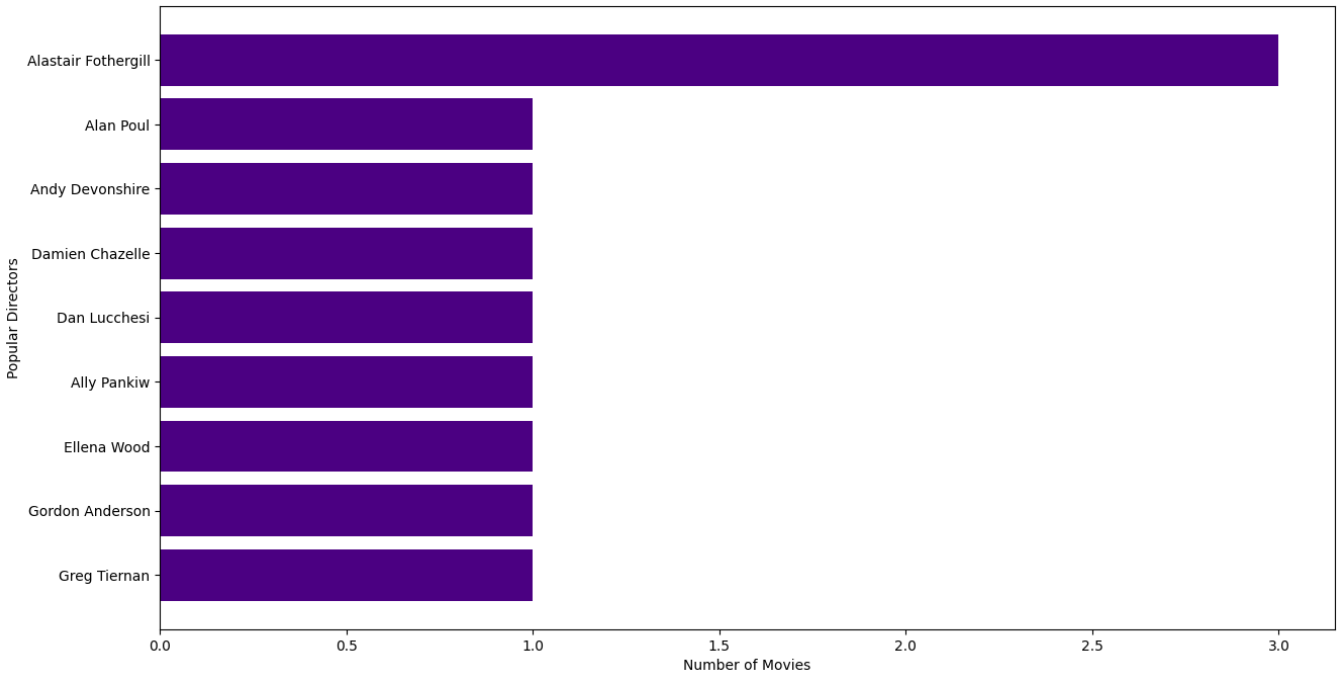
```
array(['David Attenborough', 'Michael Palin', 'Terry Jones',
       'John Cleese', 'Graham Chapman', 'Eric Idle', 'Brendan Coyle',
       'Olivia Colman', 'Rory Kinnear'], dtype=object)
```

```
df_actors=df_uk_movies.groupby(['actors']).agg({"title":"nunique"}).reset_index
df_actors=df_actors[df_actors['actors']!='Unknown Actor']
plt.figure(figsize=(15,8))
plt.barh(df_actors[::-1]['actors'], df_actors[::-1]['title'],color=['indigo'])
plt.xlabel('Number of Movies')
plt.ylabel('Popular Actors')
plt.show()
```



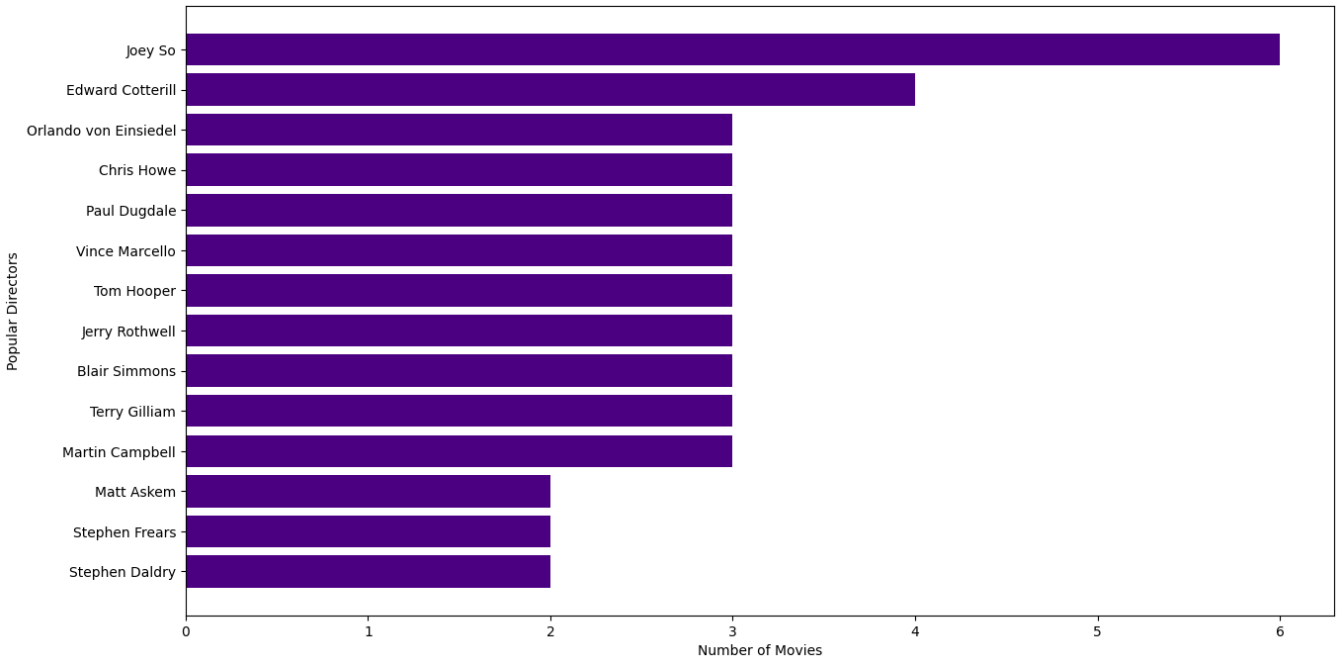
JOHN CLEESE has most no of movies released in UK.

```
df_directors=df_uk_shows.groupby(['directors']).agg({"title":"nunique"}).reset_
df_directors=df_directors[df_directors['directors']!='Unknown Director']
plt.figure(figsize=(15,8))
plt.barh(df_directors[::-1]['directors'], df_directors[::-1]['title'],color=['i
plt.xlabel('Number of Movies')
plt.ylabel('Popular Directors')
plt.show()
```



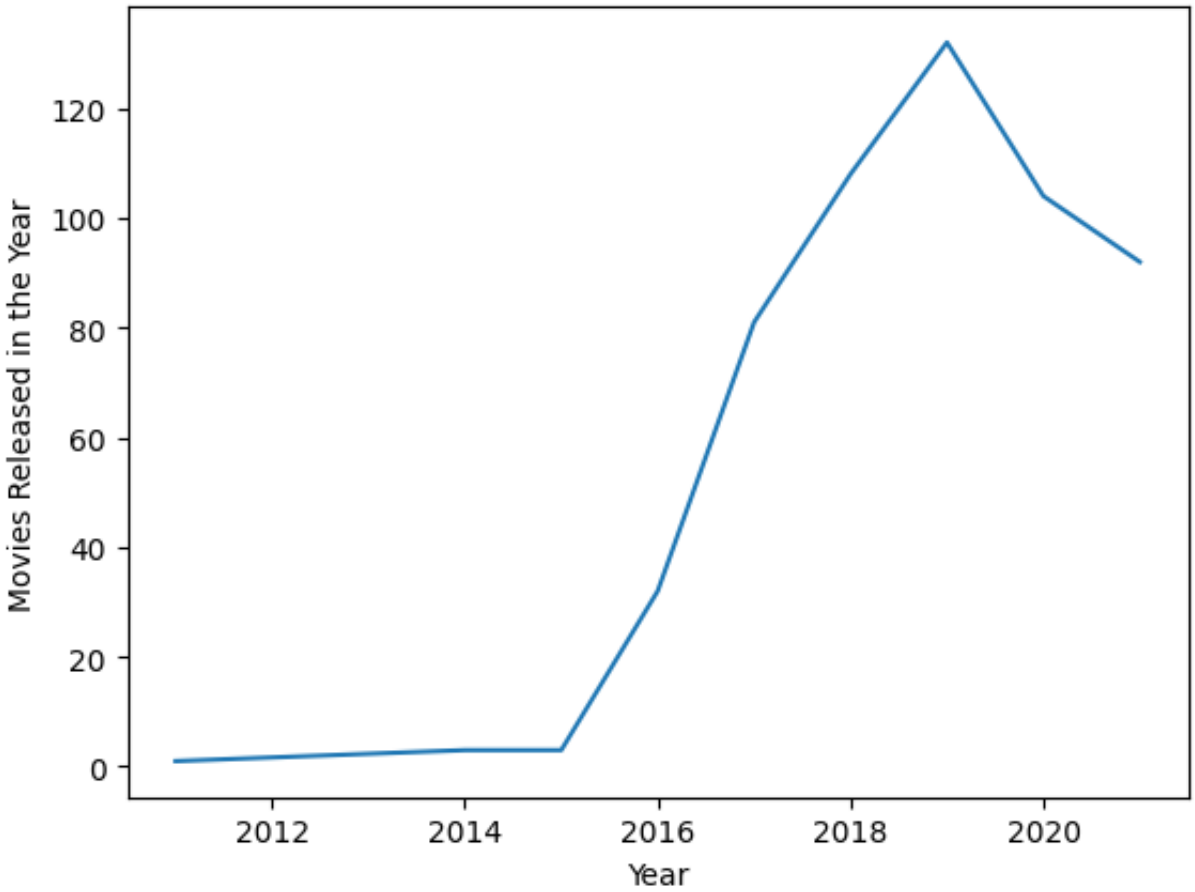
Alastair Fothergill has directed most no of TV Shows in UK.

```
df_directors=df_uk_movies.groupby(['directors']).agg({"title":"nunique"}).reset
df_directors=df_directors[df_directors['directors']!='Unknown Director']
plt.figure(figsize=(15,8))
plt.barh(df_directors[::-1]['directors'], df_directors[::-1]['title'],color=['i
plt.xlabel('Number of Movies')
plt.ylabel('Popular Directors')
plt.show()
```



Joey So has directed most no of Movies in UK.

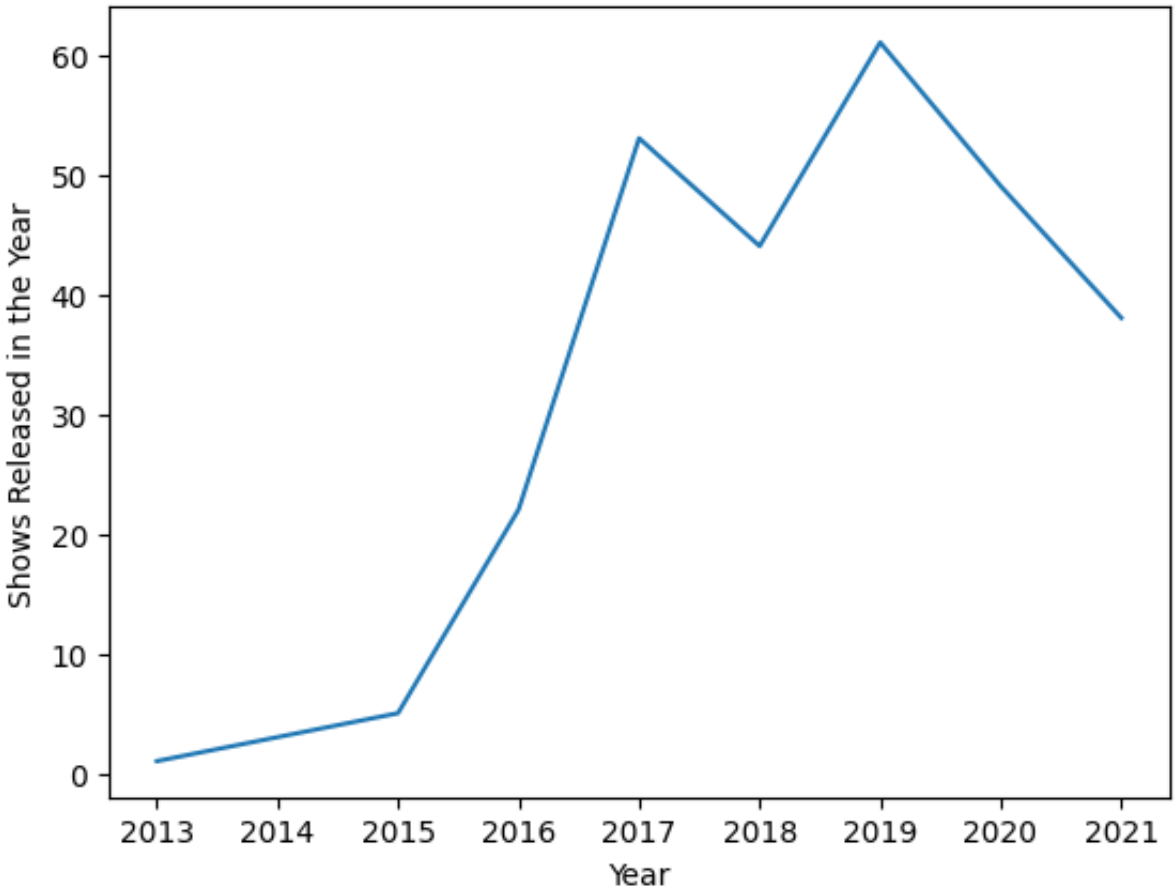
```
df_year=df_uk_movies.groupby(['year']).agg({"title":"nunique"}).reset_index()
sns.lineplot(data=df_year, x='year', y='title')
plt.ylabel("Movies Released in the Year")
plt.xlabel("Year")
plt.show()
```



Netflix has started to increase in releasing no of movies in UK form 2014 till 2018.

from 2018 it has dropped in releasing the mvoies may be due to Covid situation.

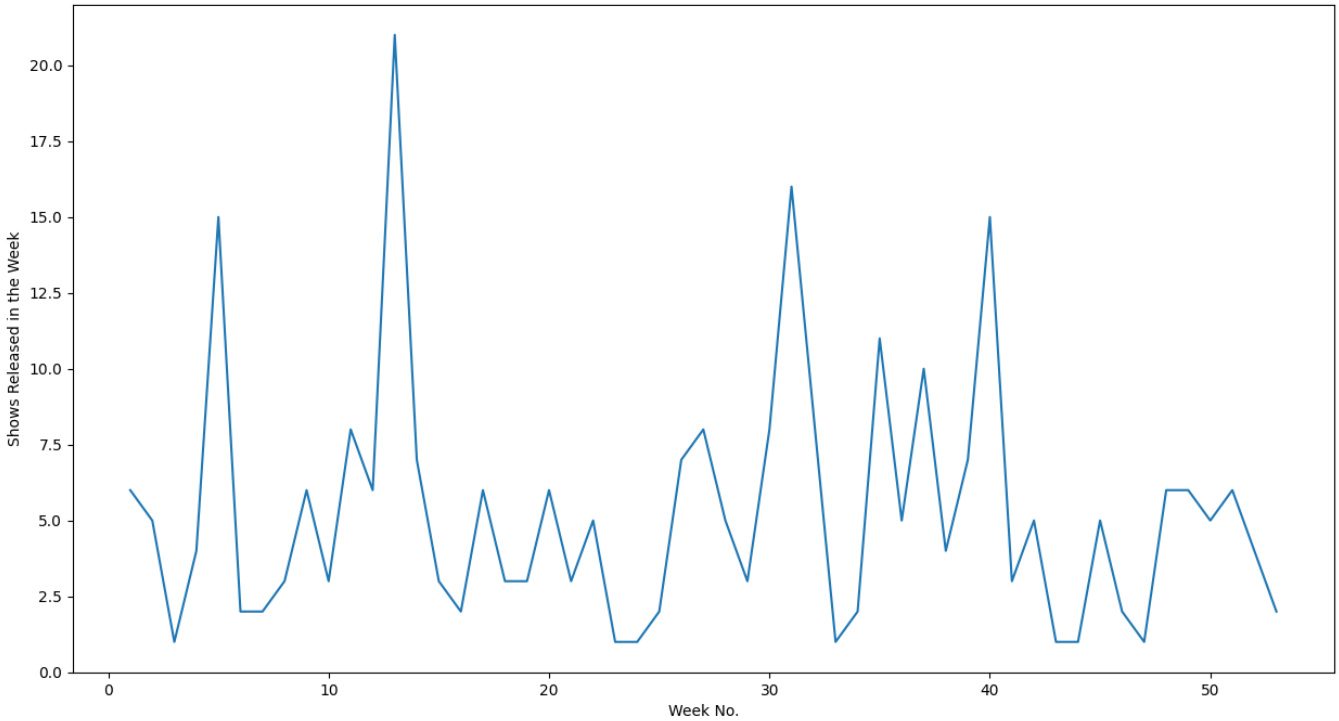
```
df_year=df_uk_shows.groupby(['year']).agg({"title":"nunique"}).reset_index()
sns.lineplot(data=df_year, x='year', y='title')
plt.ylabel("Shows Released in the Year")
plt.xlabel("Year")
plt.show()
```



Shows released has seen increase from 2014 to 2017. Saw a dip in mid of 2016 till 2018.

Again, no of shows released increased from 2018 for an year upto 2019, then again saw a dip from 2019.

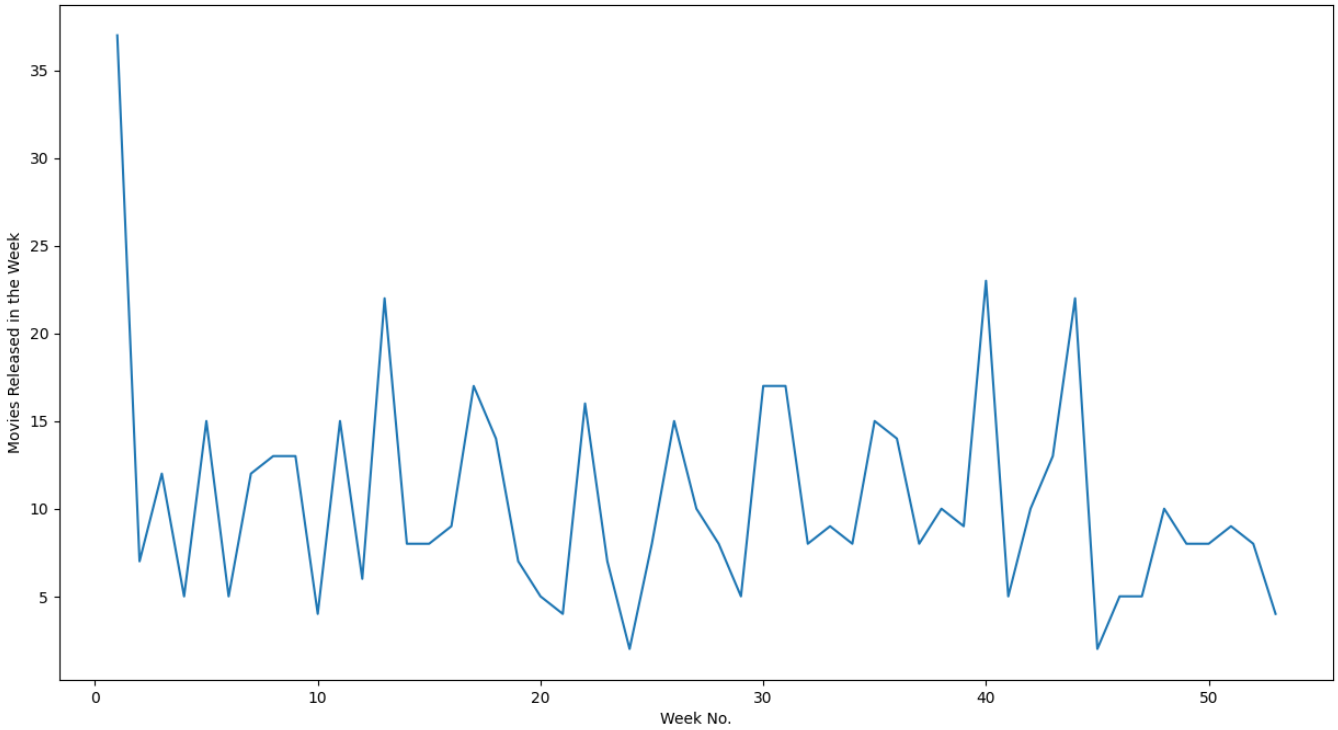
```
df_week=df_uk_shows.groupby(['week_Added']).agg({"title":"nunique"}).reset_inde
plt.figure(figsize=(15,8))
sns.lineplot(data=df_week, x='week_Added', y='title')
plt.ylabel("Shows Released in the Week")
plt.xlabel("Week No.")
plt.show()
```



On a rough basis, Netflix has released shows every 2 weeks and good no of shows were released.

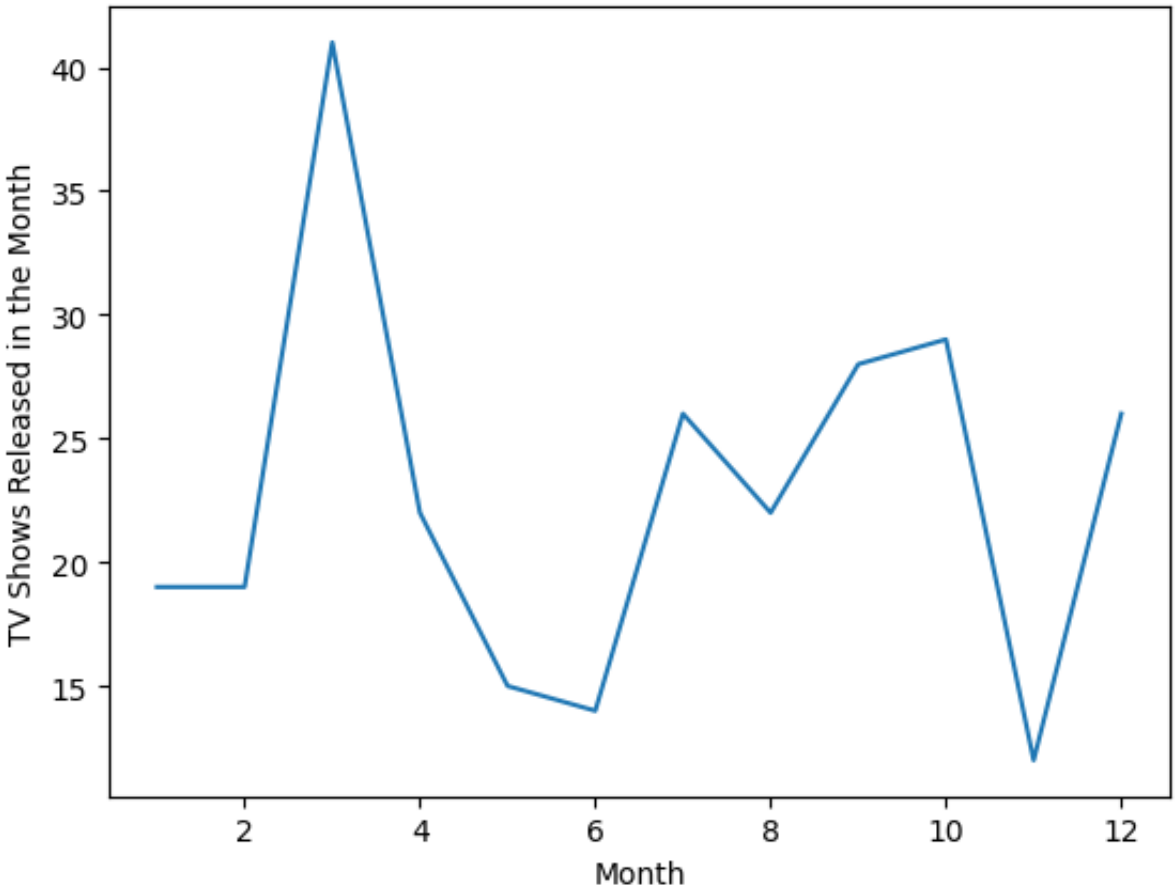
In week 11-12, >20 shows were released, it took some time for people to consume that released content and then in 18th week and 20th week it released.

```
df_week=df_uk_movies.groupby(['week_Added']).agg({"title":"nunique"}).reset_inc
plt.figure(figsize=(15,8))
sns.lineplot(data=df_week, x='week_Added', y='title')
plt.ylabel("Movies Released in the Week")
plt.xlabel("Week No.")
plt.show()
```



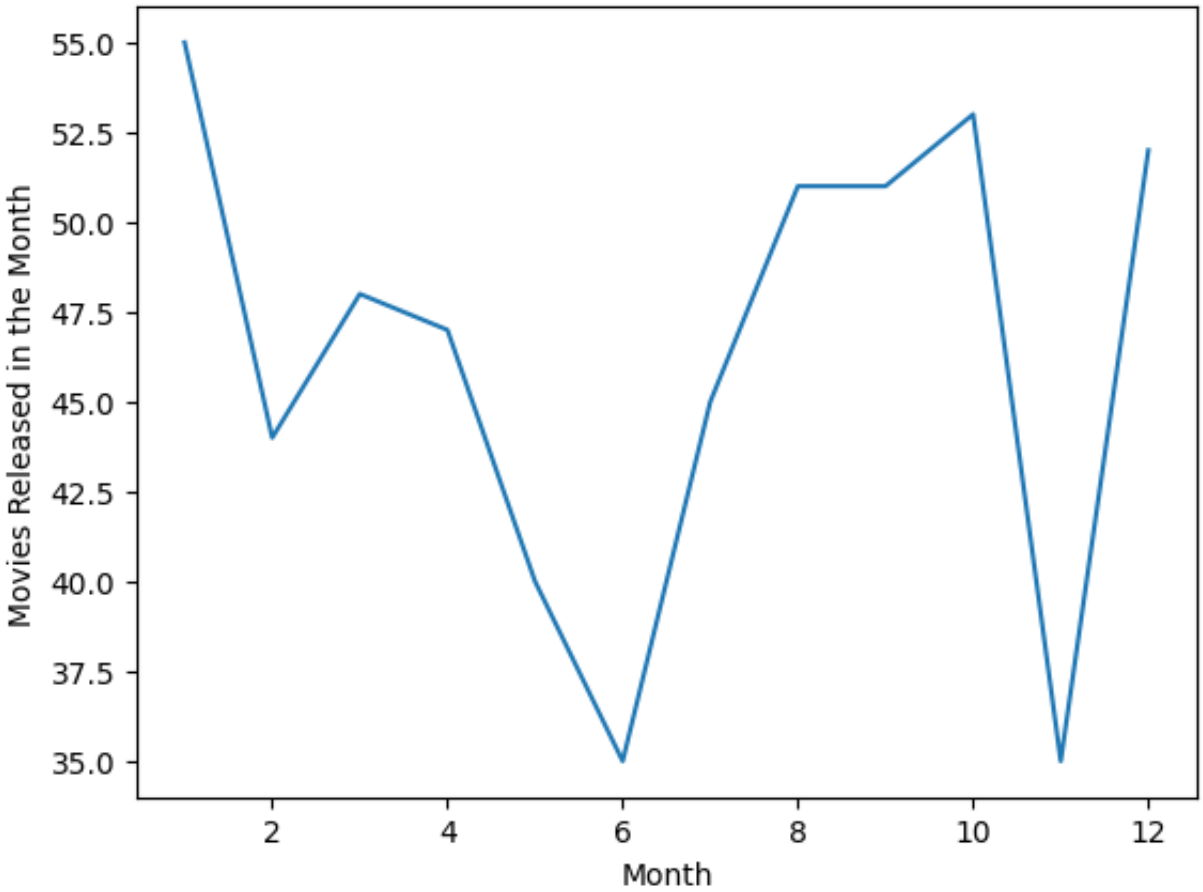
Good cycle of movie release and no release are seen in weekly basis. Every alternate week movies were released.

```
df_month=df_uk_shows.groupby(['month_added']).agg({"title":"nunique"}).reset_ir
sns.lineplot(data=df_month, x='month_added', y='title')
plt.ylabel("TV Shows Released in the Month")
plt.xlabel("Month")
plt.show()
```

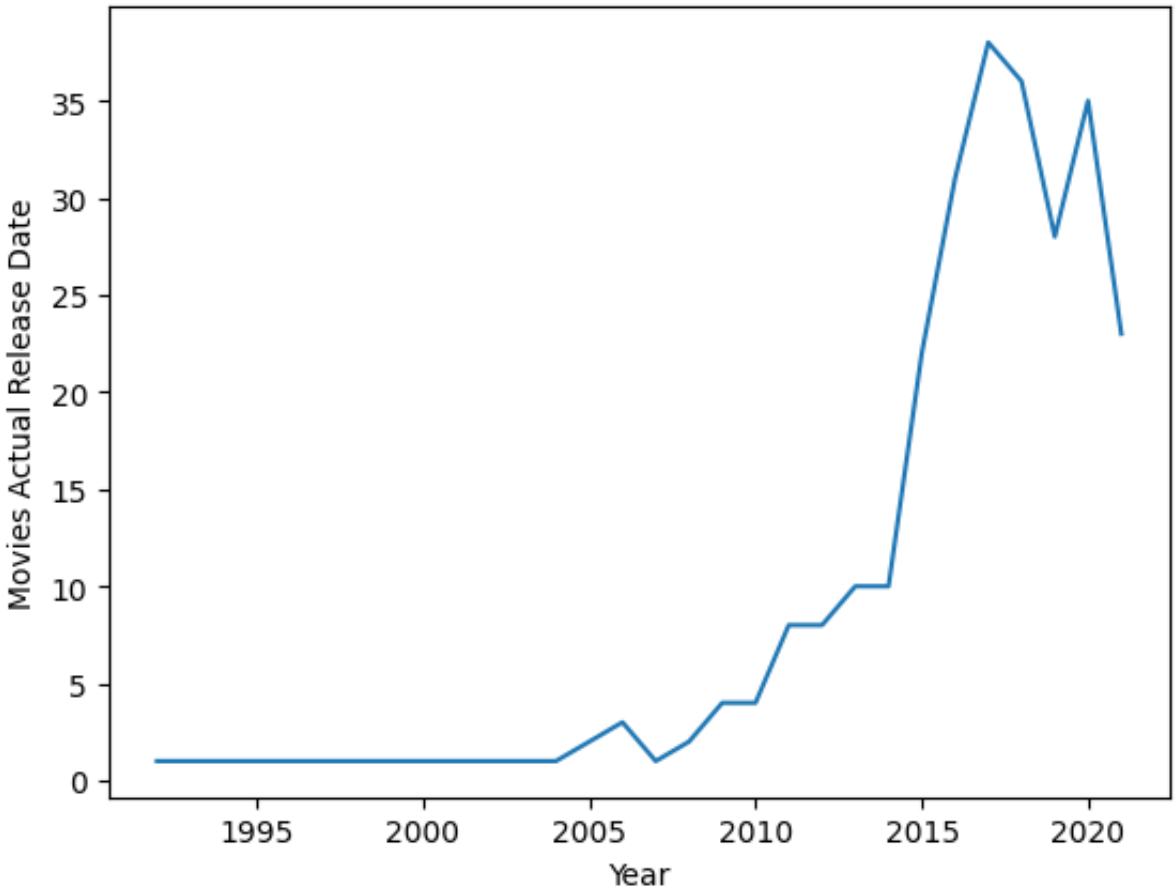


TV shows released is maximum in March of the years.

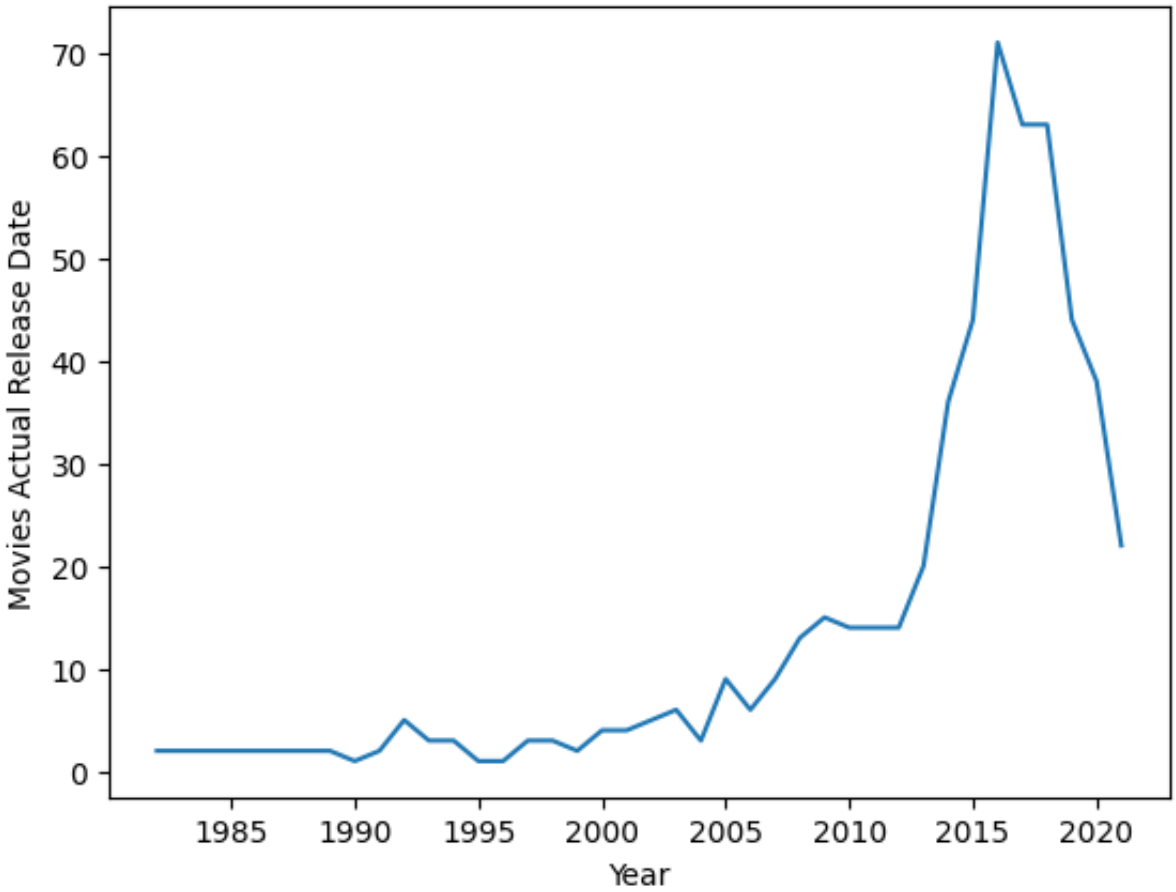
```
df_month=df_uk_movies.groupby(['month_added']).agg({"title":"nunique"}).reset_i
sns.lineplot(data=df_month, x='month_added', y='title')
plt.ylabel("Movies Released in the Month")
plt.xlabel("Month")
plt.show()
```



```
df_release_year=df_uk_shows[df_uk_shows['release_year']>=1980].groupby(['releas
sns.lineplot(data=df_release_year, x='release_year', y='title')
plt.ylabel("Movies Actual Release Date")
plt.xlabel("Year")
plt.show()
```



```
df_release_year=df_uk_movies[df_uk_movies['release_year']>=1980].groupby(['rele
sns.lineplot(data=df_release_year, x='release_year', y='title')
plt.ylabel("Movies Actual Release Date")
plt.xlabel("Year")
plt.show()
```



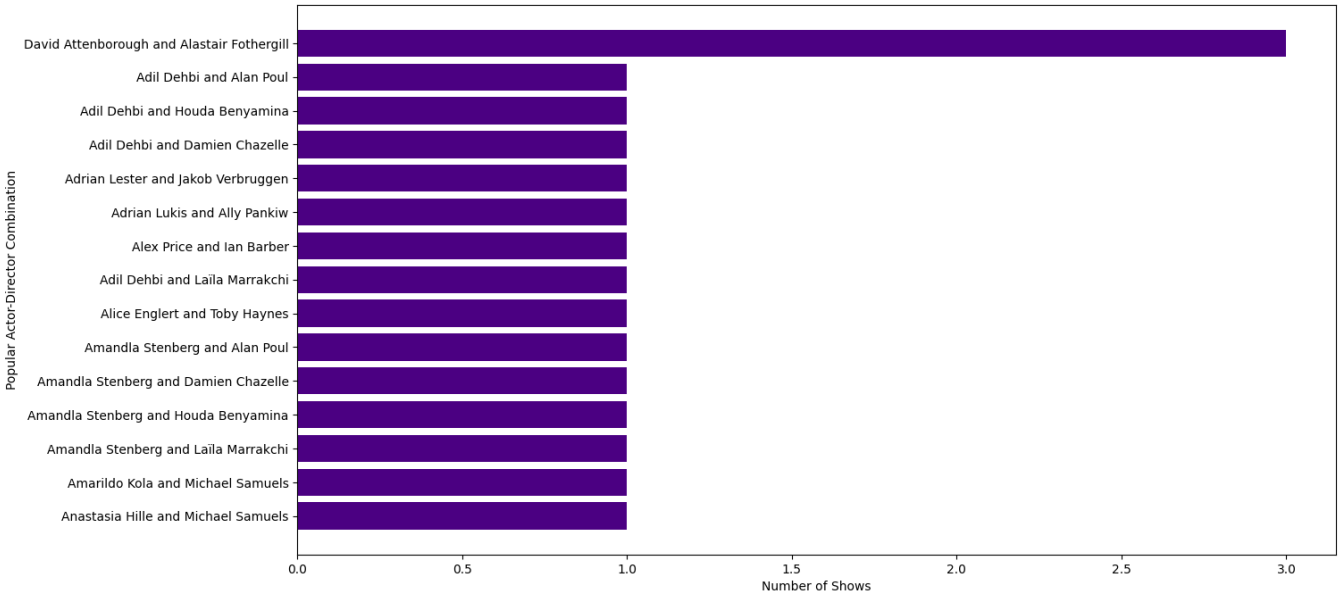
```
#Analysing a combination of actors and directors
df_uk_movies['Actor_Director_Combination'] = df_uk_movies.actors.str.cat(df_uk_
df_uk_movies_subset=df_uk_movies[df_uk_movies['actors']!='Unknown Actor']
df_uk_movies_subset=df_uk_movies_subset[df_uk_movies_subset['directors']!='Unkr
df_uk_movies_subset.head()
```

	show_id	type	title	directors	actors	country	date_added	release_
7	s8	Movie	Sankofa	Haile Gerima	Kofi Ghanaba	United Kingdom	September 24, 2021	
7	s8	Movie	Sankofa	Haile Gerima	Kofi Ghanaba	United Kingdom	September 24, 2021	
7	s8	Movie	Sankofa	Haile Gerima	Kofi Ghanaba	United Kingdom	September 24, 2021	
7	s8	Movie	Sankofa	Haile Gerima	Oyafunmike Ogunlano	United Kingdom	September 24, 2021	
7	s8	Movie	Sankofa	Haile Gerima	Oyafunmike Ogunlano	United Kingdom	September 24, 2021	

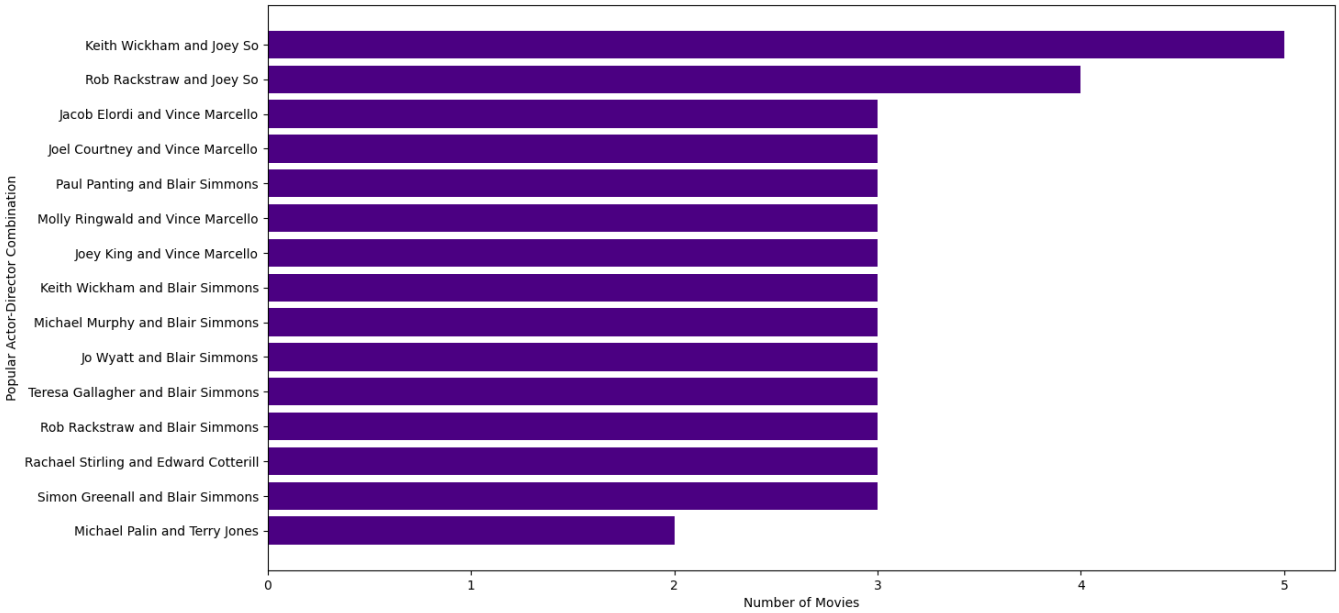
```
df_uk_shows['Actor_Director_Combination'] = df_uk_shows.actors.str.cat(df_uk_sh
df_uk_shows_subset=df_uk_shows[df_uk_shows['actors']!='Unknown Actor']
df_uk_shows_subset=df_uk_shows_subset[df_uk_shows_subset['directors']!='Unknowr
df_uk_shows_subset.head()
```

	show_id	type	title	directors	actors	country	date_added	release_yea
8	s9	TV Show	The Great British Baking Show	Andy Devonshire	Mel Giedroyc	United Kingdom	September 24, 2021	202
8	s9	TV Show	The Great British Baking Show	Andy Devonshire	Mel Giedroyc	United Kingdom	September 24, 2021	202
8	s9	TV Show	The Great British Baking Show	Andy Devonshire	Sue Perkins	United Kingdom	September 24, 2021	202
8	s9	TV Show	The Great British Baking Show	Andy Devonshire	Sue Perkins	United Kingdom	September 24, 2021	202
8	s9	TV Show	The Great British Baking Show	Andy Devonshire	Mary Berry	United Kingdom	September 24, 2021	202

```
df_actors_directors=df_uk_shows_subset.groupby(['Actor_Director_Combination']).
plt.figure(figsize=(15,8))
plt.barh(df_actors_directors[::-1]['Actor_Director_Combination'], df_actors_dir
plt.xlabel('Number of Shows')
plt.ylabel('Popular Actor-Director Combination')
plt.show()
```



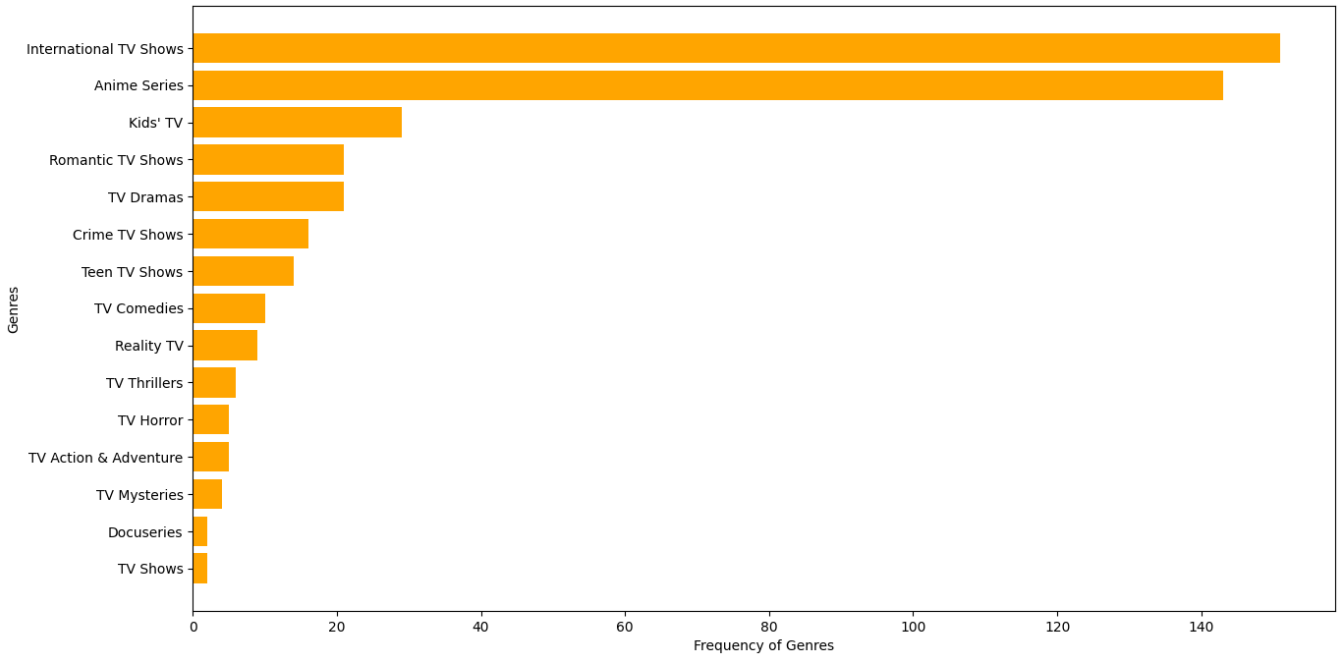
```
df_actors_directors=df_uk_movies_subset.groupby(['Actor_Director_Combination'])
plt.figure(figsize=(15,8))
plt.barh(df_actors_directors[::-1]['Actor_Director_Combination'], df_actors_dir
plt.xlabel('Number of Movies')
plt.ylabel('Popular Actor-Director Combination')
plt.show()
```



Analyzing Japan for both shows and movies

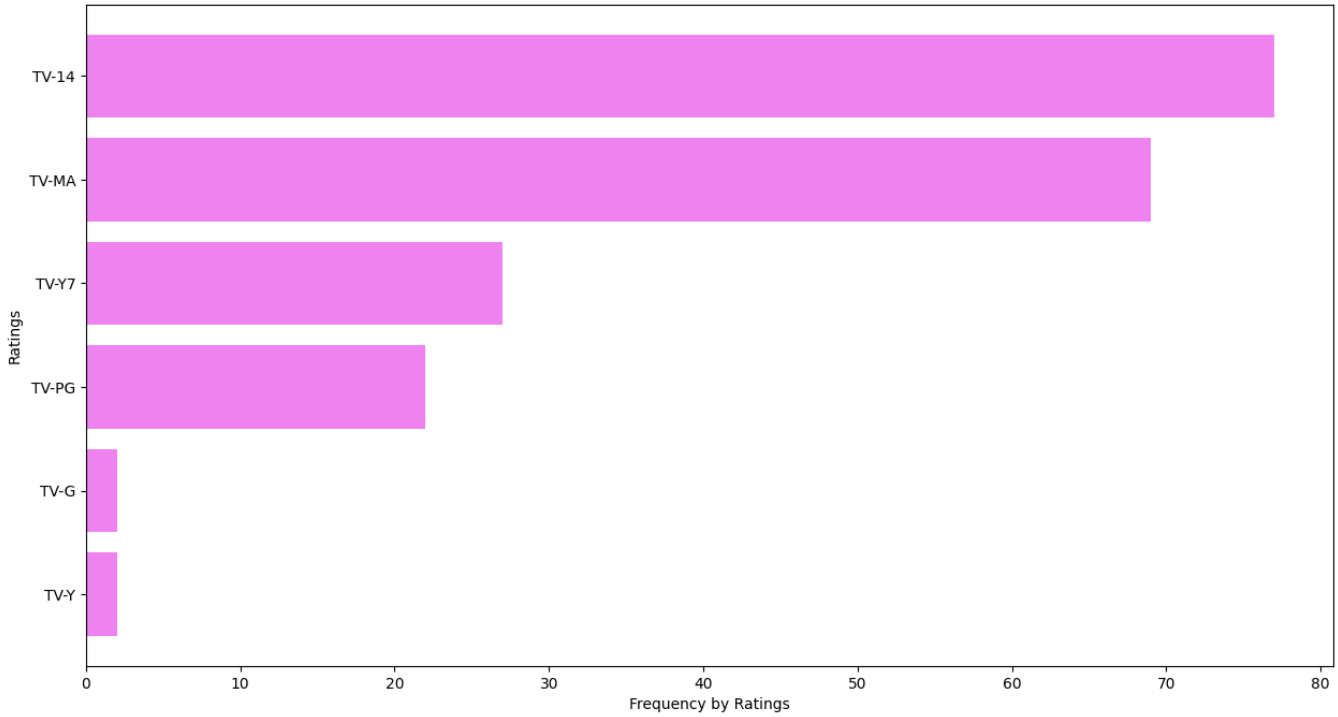
```
df_japan_shows=df[df['country']=='Japan'][df[df['country']=='Japan']['type']=='
```

```
df_genre=df_japan_shows.groupby(['genre']).agg({"title":"nunique"}).reset_index
plt.figure(figsize=(15,8))
plt.barh(df_genre[::-1]['genre'], df_genre[::-1]['title'],color=['orange'])
plt.xlabel('Frequency of Genres')
plt.ylabel('Genres')
plt.show()
```



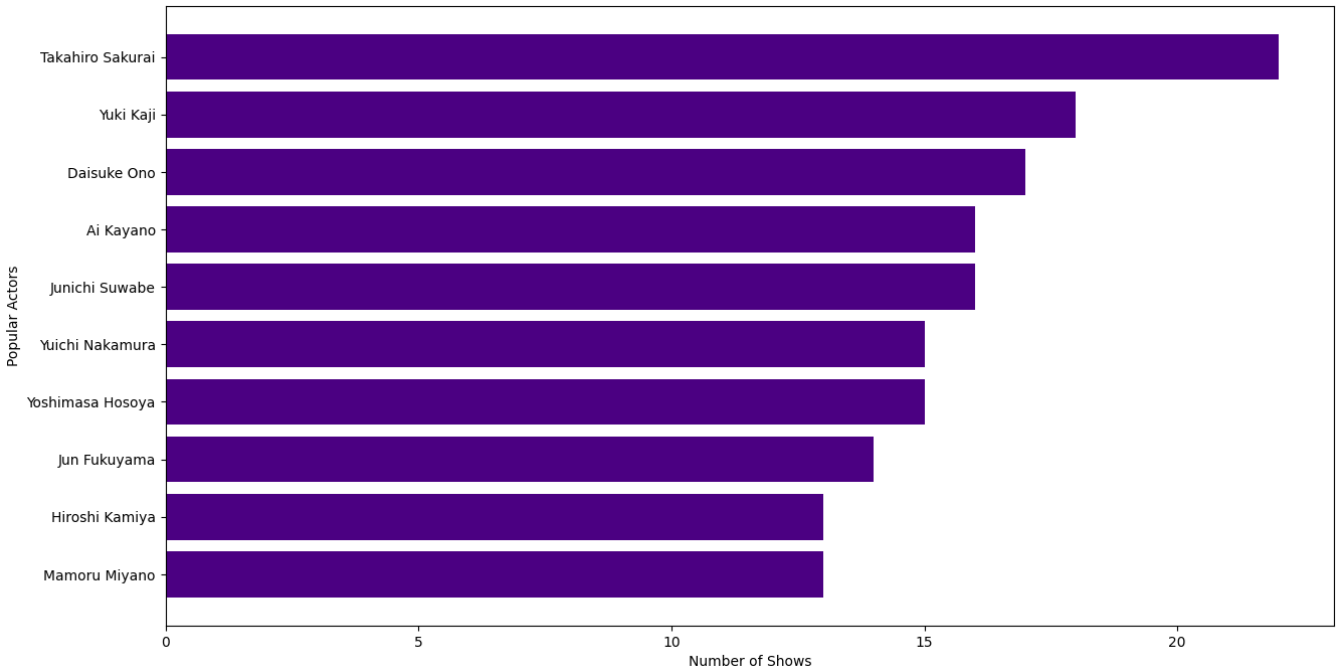
International TV Shows and Anime Genres are popular in TV Shows in Japan

```
df_rating=df_japan_shows.groupby(['rating']).agg({"title":"nunique"}).reset_inc
plt.figure(figsize=(15,8))
plt.barh(df_rating[::-1]['rating'], df_rating[::-1]['title'],color=['violet'])
plt.xlabel('Frequency by Ratings')
plt.ylabel('Ratings')
plt.show()
```



So it seems plaussible to conclude that the popular ratings across Netflix includes TV-14 Mature Audiences in TV Shows

```
df_actors=df_japan_shows.groupby(['actors']).agg({"title":"nunique"}).reset_inc
df_actors=df_actors[df_actors['actors']!='Unknown Actor']
plt.figure(figsize=(15,8))
plt.barh(df_actors[::-1]['actors'], df_actors[::-1]['title'],color=['indigo'])
plt.xlabel('Number of Shows')
plt.ylabel('Popular Actors')
plt.show()
```



Popular Actors in TV Shows in Japan are:-

'Takahiro Sakurai'

, 'Yuki Kaji'

, 'Daisuke Ono'

, 'Junichi Suwabe'

, 'Ai Kayano'

, 'Yuichi Nakamura'

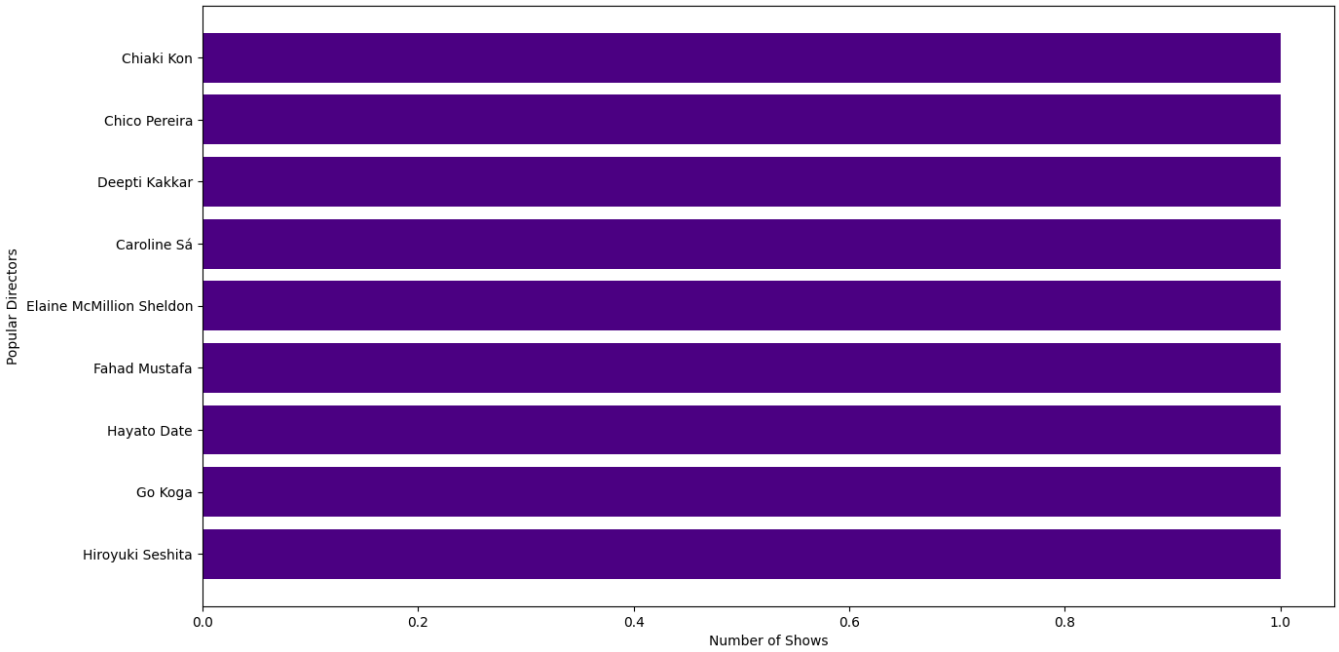
, 'Yoshimasa Hosoya'

, 'Jun Fukuyama'

, 'Hiroshi Kamiya'

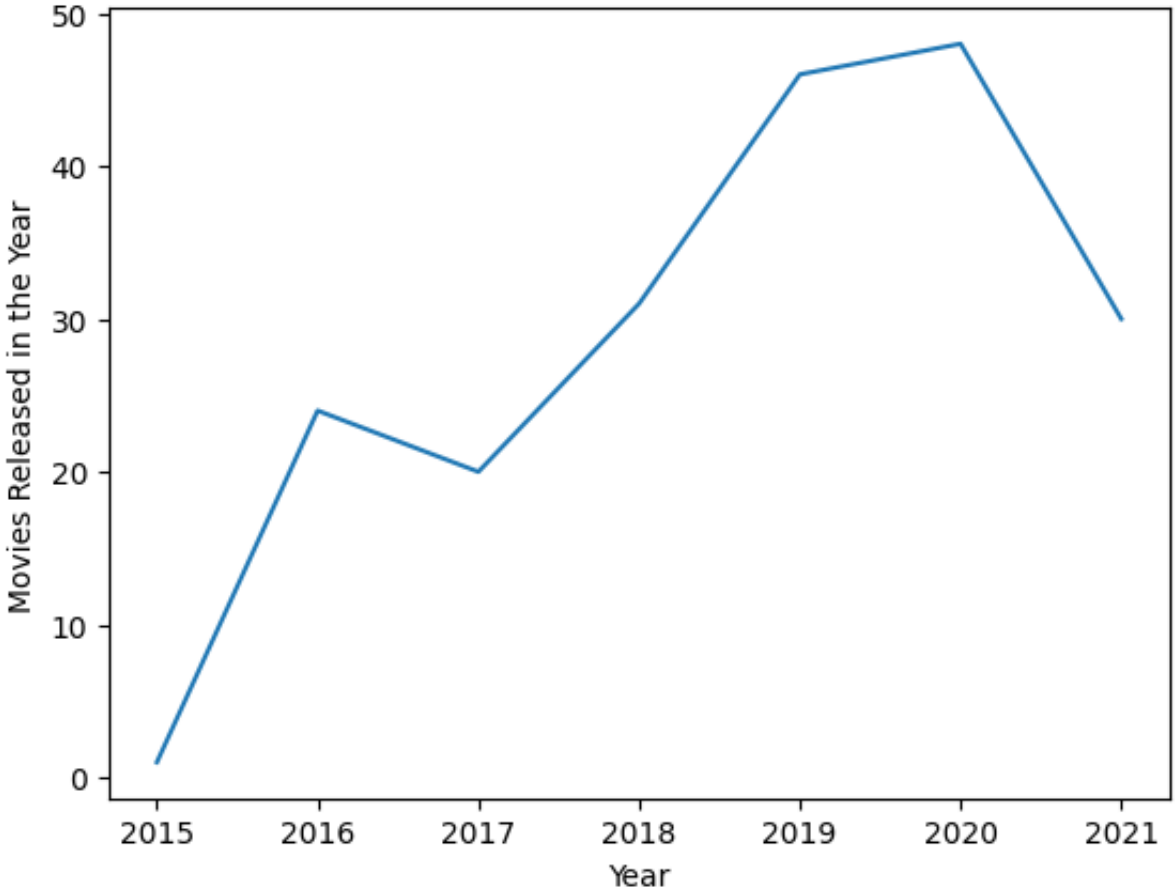
, 'Kana Hanazawa'

```
df_directors=df_japan_shows.groupby(['directors']).agg({"title":"nunique"}).res
df_directors=df_directors[df_directors['directors']!='Unknown Director']
plt.figure(figsize=(15,8))
plt.barh(df_directors[::-1]['directors'], df_directors[::-1]['title'],color=['i
plt.xlabel('Number of Shows')
plt.ylabel('Popular Directors')
plt.show()
```



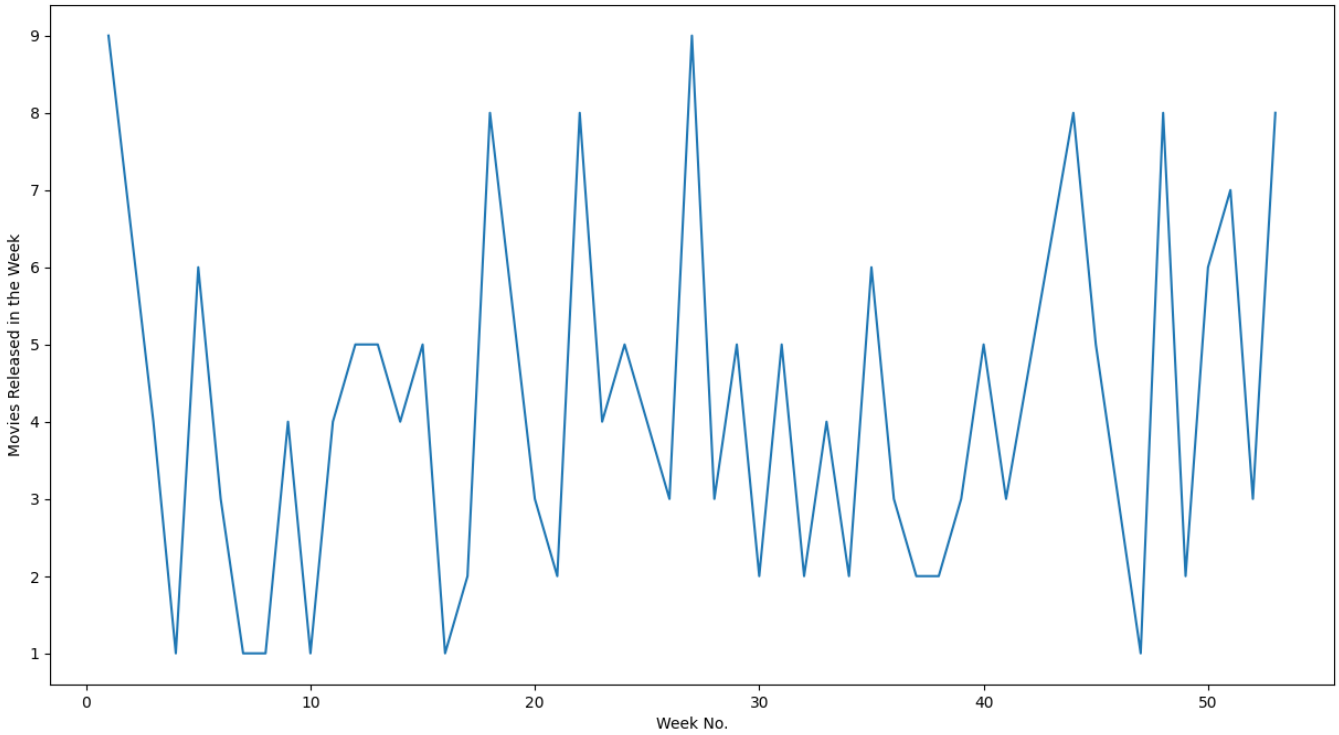
All Directors are one time directors only

```
df_year=df_japan_shows.groupby(['year']).agg({"title":"nunique"}).reset_index()
sns.lineplot(data=df_year, x='year', y='title')
plt.ylabel("Movies Released in the Year")
plt.xlabel("Year")
plt.show()
```

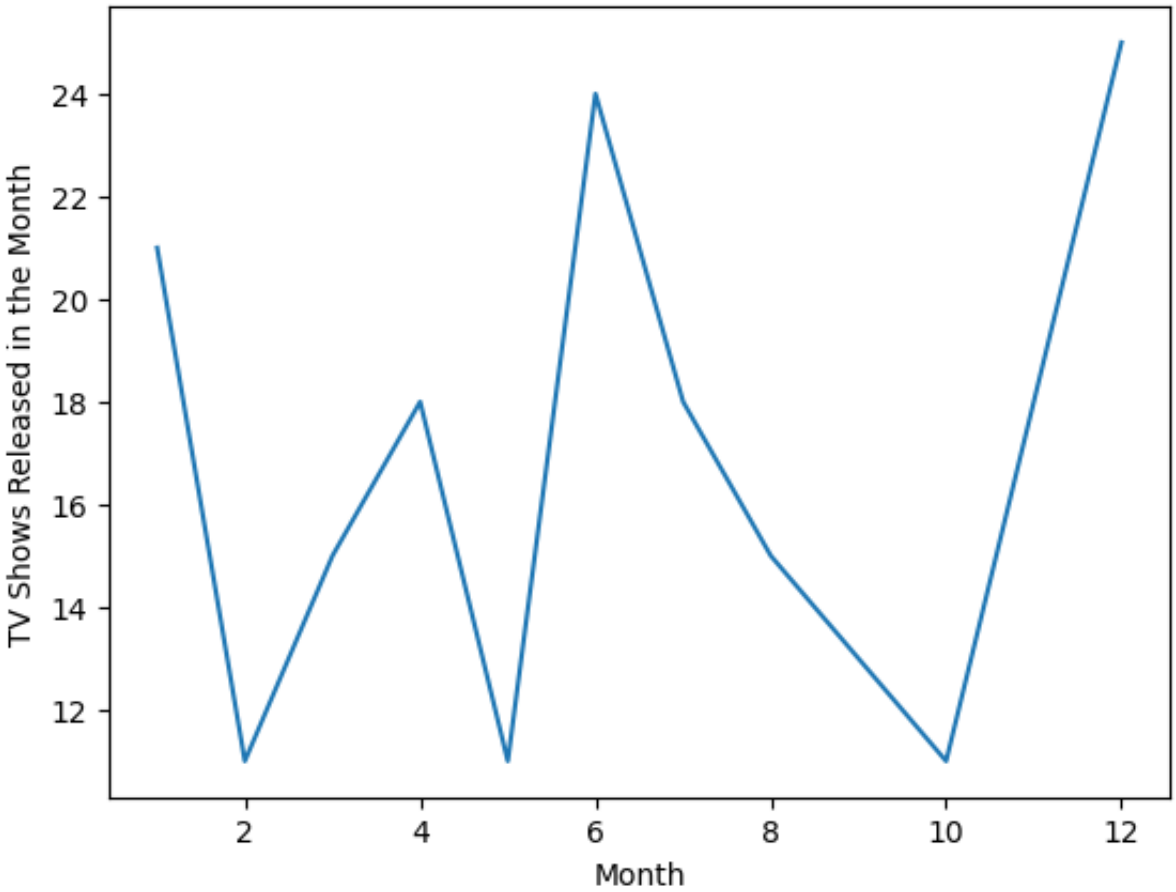


In Japan, TV Shows have diminished in 2017 from 2016 and then increased till 2020 after which it has reduced in 2021.

```
df_week=df_japan_shows.groupby(['week_Added']).agg({"title":"nunique"}).reset_i
plt.figure(figsize=(15,8))
sns.lineplot(data=df_week, x='week_Added', y='title')
plt.ylabel("Movies Released in the Week")
plt.xlabel("Week No.")
plt.show()
```

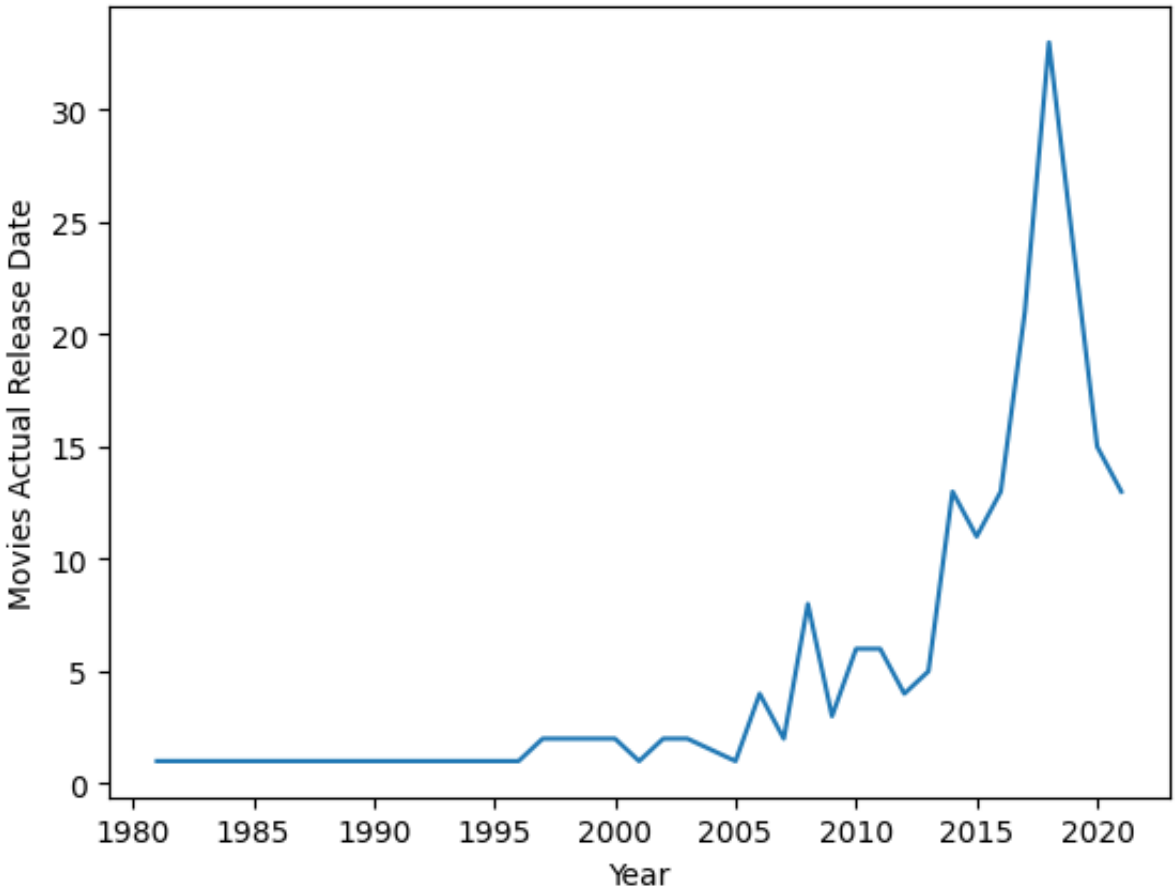


```
df_month=df_japan_shows.groupby(['month_added']).agg({"title":"nunique"}).reset
sns.lineplot(data=df_month, x='month_added', y='title')
plt.ylabel("TV Shows Released in the Month")
plt.xlabel("Month")
plt.show()
```



TV Shows are added in Netflix by significant numbers in April and January in Japan

```
df_release_year=df_japan_shows[df_japan_shows['release_year']>=1980].groupby(['
sns.lineplot(data=df_release_year, x='release_year', y='title')
plt.ylabel("Movies Actual Release Date")
plt.xlabel("Year")
plt.show()
```

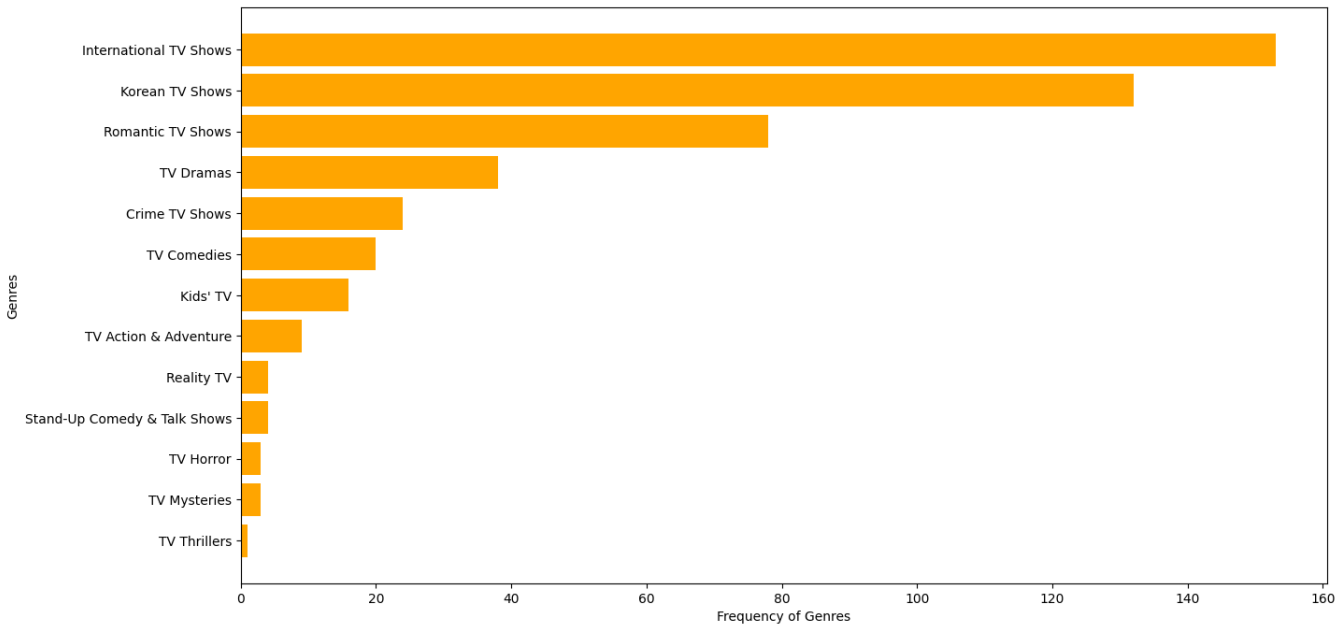


Reduction in TV Shows after 2019 in Japan

Analyzing South Korea for both shows and movies

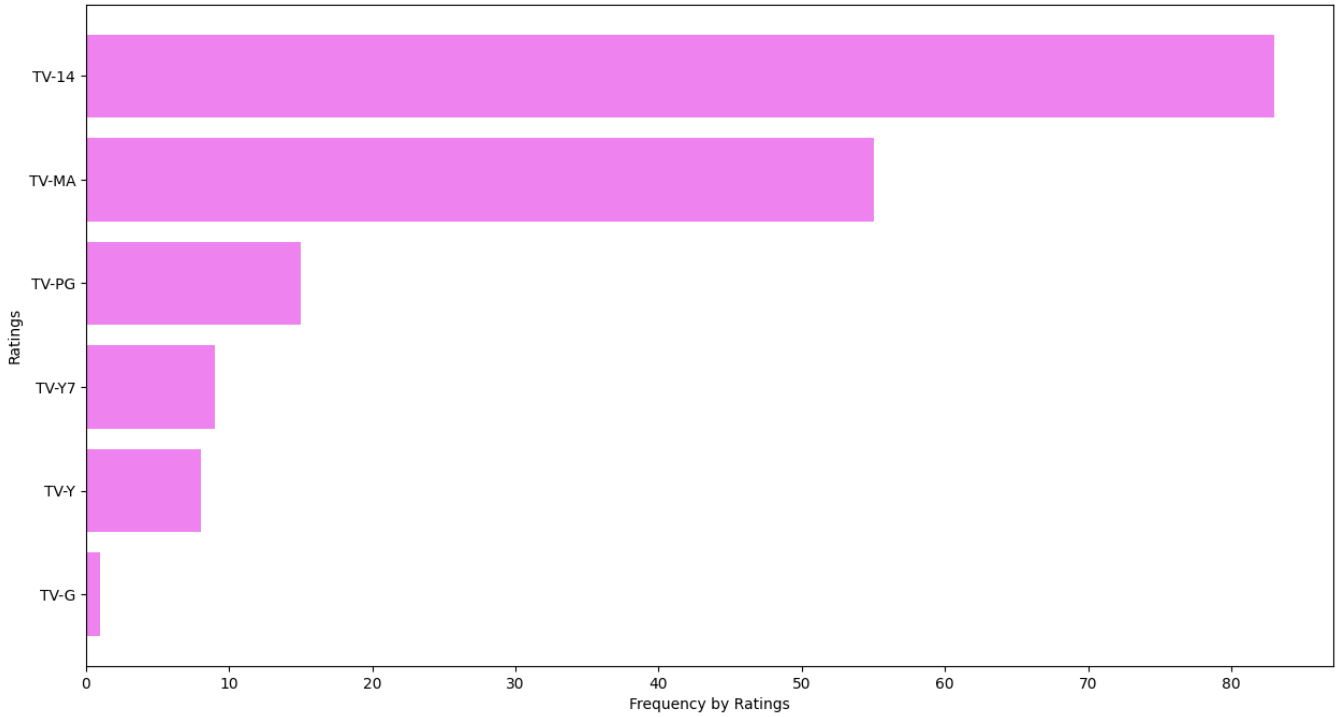
```
df_sk_shows=df[df['country']=='South Korea'][df[df['country']=='South Korea']['
```

```
df_genre=df_sk_shows.groupby(['genre']).agg({"title":"nunique"}).reset_index().
plt.figure(figsize=(15,8))
plt.barh(df_genre[::-1]['genre'], df_genre[::-1]['title'],color=['orange'])
plt.xlabel('Frequency of Genres')
plt.ylabel('Genres')
plt.show()
```



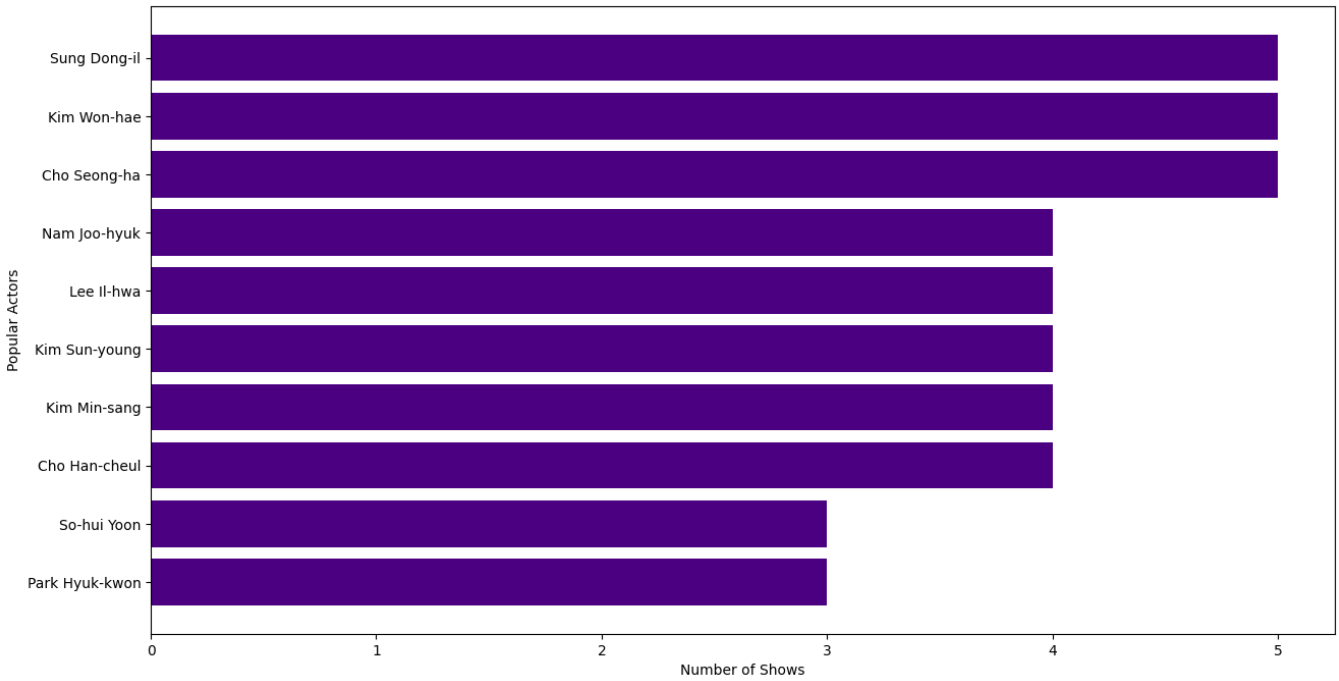
International TV Shows,Romantic TV Shows,Drama,Crime and Comedy Genres are popular in TV Shows in S.Korea.

```
df_rating=df_sk_shows.groupby(['rating']).agg({"title":"nunique"}).reset_index(
plt.figure(figsize=(15,8))
plt.barh(df_rating[::-1]['rating'], df_rating[::-1]['title'],color=['violet'])
plt.xlabel('Frequency by Ratings')
plt.ylabel('Ratings')
plt.show()
```



So it seems plaussible to conclude that the popular ratings across Netflix includes TV-14 and Mature Audiences in TV Shows

```
df_actors=df_sk_shows.groupby(['actors']).agg({"title":"nunique"}).reset_index(
df_actors=df_actors[df_actors['actors']!='Unknown Actor']
plt.figure(figsize=(15,8))
plt.barh(df_actors[::-1]['actors'], df_actors[::-1]['title'],color=['indigo'])
plt.xlabel('Number of Shows')
plt.ylabel('Popular Actors')
plt.show()
```



Popular Actors in TV Shows in South Korea are:-

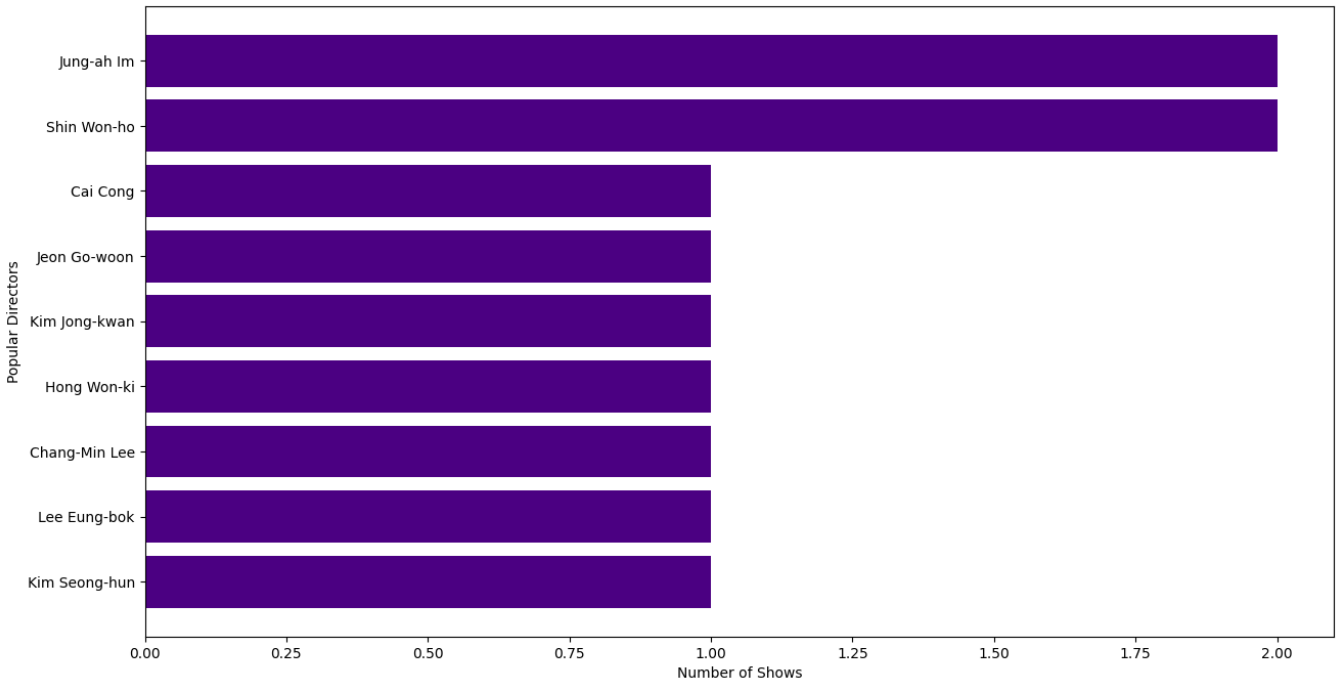
'Sung Dong-il' ,

'Kim Won-hae' ,

'Cho Seong-ha' ,

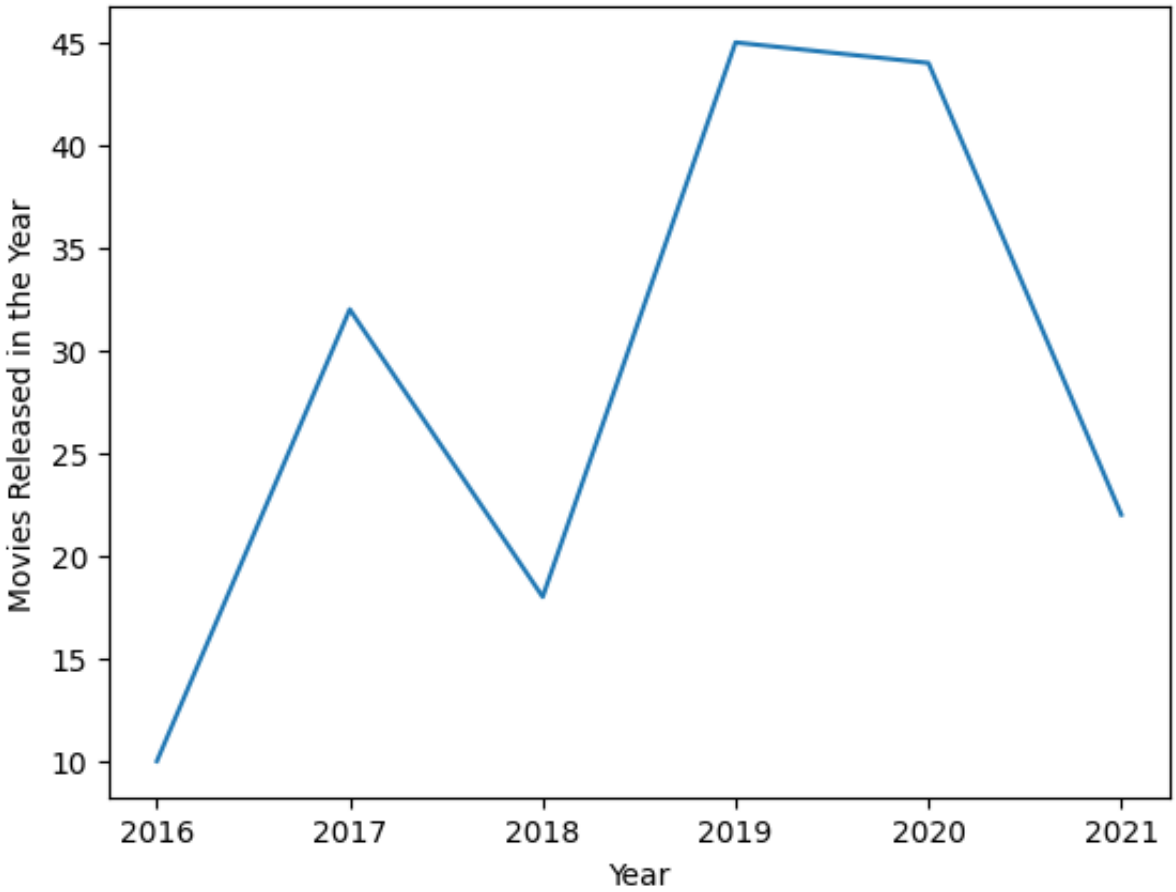
'Nam Joo-hyuk'

```
df_directors=df_sk_shows.groupby(['directors']).agg({"title":"nunique"}).reset_
df_directors=df_directors[df_directors['directors']!='Unknown Director']
plt.figure(figsize=(15,8))
plt.barh(df_directors[::-1]['directors'], df_directors[::-1]['title'],color=['j
plt.xlabel('Number of Shows')
plt.ylabel('Popular Directors')
plt.show()
```



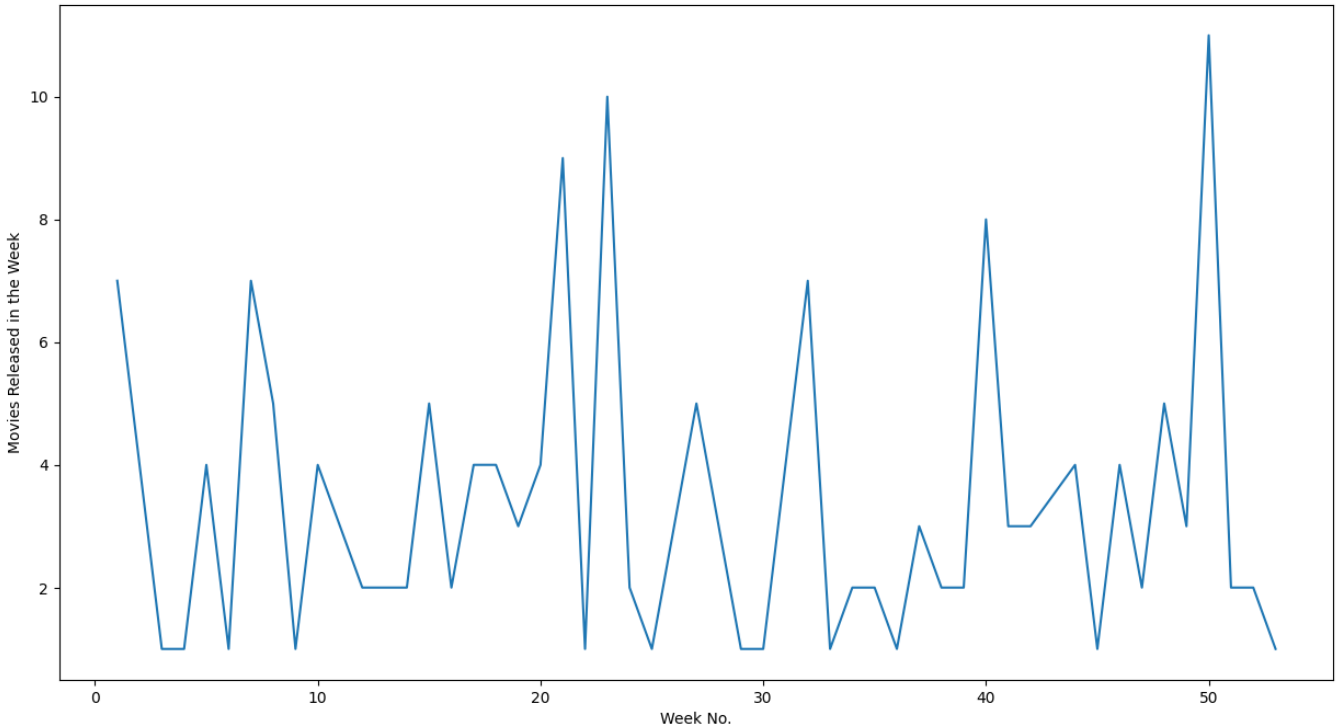
Two directors have directed 2 shows and rest all Directors are one time directors only

```
df_year=df_sk_shows.groupby(['year']).agg({"title":"nunique"}).reset_index()
sns.lineplot(data=df_year, x='year', y='title')
plt.ylabel("Movies Released in the Year")
plt.xlabel("Year")
plt.show()
```

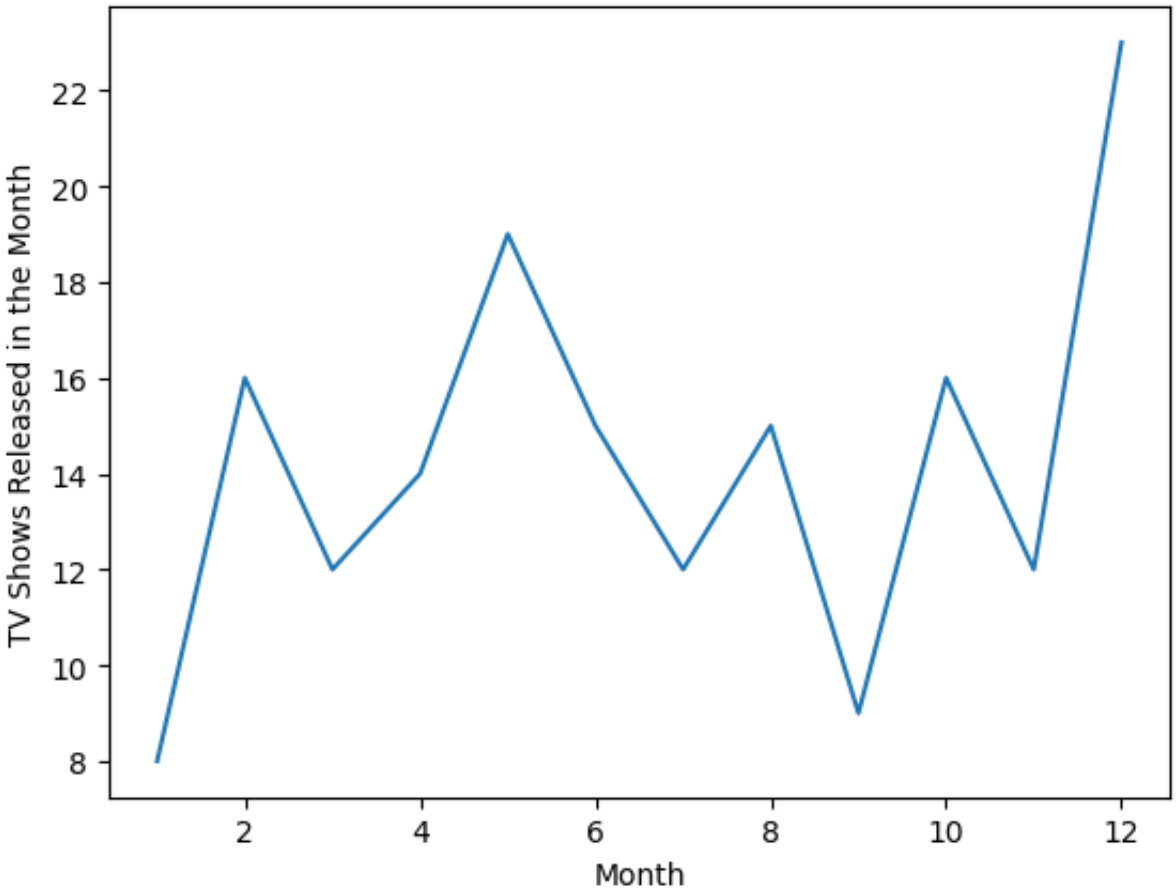


In South Korea, number of TV Shows reduced in 2018 from 2017, then increased till 2019 but have been on a heavy downfall since then

```
df_week=df_sk_shows.groupby(['week_Added']).agg({"title":"nunique"}).reset_inde
plt.figure(figsize=(15,8))
sns.lineplot(data=df_week, x='week_Added', y='title')
plt.ylabel("Movies Released in the Week")
plt.xlabel("Week No.")
plt.show()
```

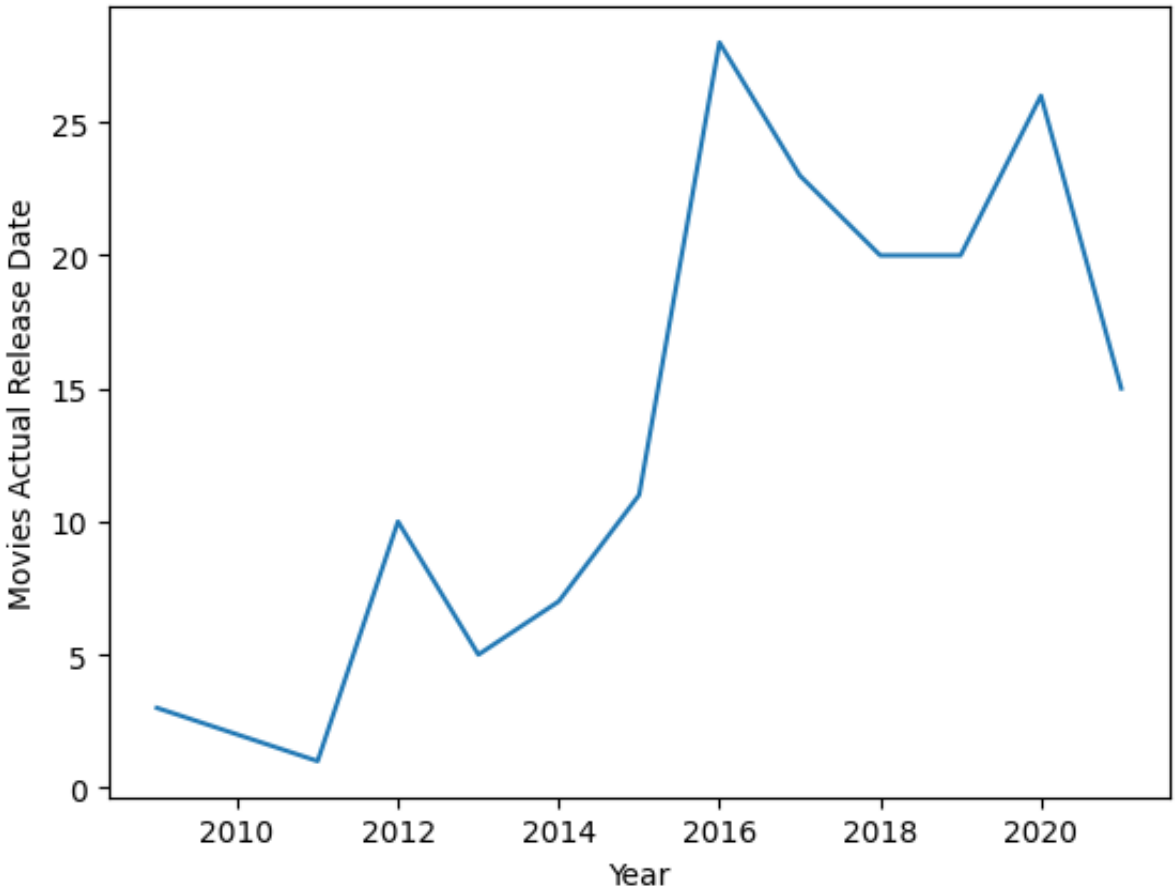


```
df_month=df_sk_shows.groupby(['month_added']).agg({"title":"nunique"}).reset_ir
sns.lineplot(data=df_month, x='month_added', y='title')
plt.ylabel("TV Shows Released in the Month")
plt.xlabel("Month")
plt.show()
```



TV Shows are added in Netflix by significant numbers in May and January in South Korea

```
df_release_year=df_sk_shows[df_sk_shows['release_year']>=1980].groupby(['releas
sns.lineplot(data=df_release_year, x='release_year', y='title')
plt.ylabel("Movies Actual Release Date")
plt.xlabel("Year")
plt.show()
```



The number of TV Shows in S.Korea reached peak in 2016. It then reached a second peak in 2019. It has reduced in 2021 from 2020.

Recommendations

1) The most popular Genres across the countries and in both TV Shows and Movies are Drama, Comedy and International TV Shows/Movies, so content aligning to that is recommended.

2) Add TV Shows in July/August and Movies in last week of the year/first month of the next year.

3) For USA audience 80-120 mins is the recommended length for movies and Kids TV Shows are also popular along with the genres in first point, hence recommended.

4) For UK audience, recommended length for movies is same as that of USA (80-120 mins).

5) The target audience in USA and India is recommended to be 14+ and above ratings while for UK, its recommended to be completely Mature/R content .

6) Add movies for Indian Audience, it has been declining since 2018.

7) Anime Genre for Japan and Romantic Genre in TV Shows for South Korean audiences is recommended.

8) While creating content, take into consideration the popular actors/directors for that country. Also take into account the director-actor combination which is highly recommended.