Business Case: Netflix - Data Exploration & Visualisation :

In [1]:

```
import numpy as np
import pandas as pd
import datetime
```

In [2]:

```
df=pd.read_csv('netflix.csv')
```

In [3]:

```
df
```

Out[3]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	25-Sep-21	2020	PG-13	90 min	Documentaries
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	24-Sep-21	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	24-Sep-21	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	24-Sep-21	2021	TV-MA	1 Season	Docuseries, Reality TV
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	24-Sep-21	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...
...	...	...	...	...	...	...	...	...	...	...	...
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	20-Nov-19	2007	R	158 min	Cult Movies, Dramas, Thrillers
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	1-Jul-19	2018	TV-Y7	2 Seasons	Kids' TV, Korean TV Shows, TV Comedies
					Jesse						

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	1-Nov-19	2009	R	88 min	Comedies, Horror Movies
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	11-Jan-20	2006	PG	88 min	Children & Family Movies, Comedies
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	2-Mar-19	2015	TV-14	111 min	Dramas, International Movies, Music & Musicals

8807 rows × 12 columns

In [4]:

```
df.shape
```

Out[4]:

```
(8807, 12)
```

In [5]:

```
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
```

In [6]:

```
df.nunique()
```

Out[6]:

```
show_id         8807
type               2
title           8804
director        4528
cast            7692
country          748
date_added      1767
release_year      74
rating            17
duration         220
listed_in        514
description     8775
dtype: int64
```

In [7]:

```
#Total 8807 movies/TV shows data is provided in the dataset.
```

In [8]:

```
df.describe()
```

Out[8]:

	release_year
count	8807.000000
mean	2014.180198
std	8.819312
min	1925.000000
25%	2013.000000
50%	2017.000000
75%	2019.000000
max	2021.000000

In [9]:

```
df.describe(include = object)
```

Out[9]:

	show_id	type	title	director	cast	country	date_added	rating	duration	listed_in	description
count	8807	8807	8807	6173	7982	7976	8797	8803	8804	8807	8807
unique	8807	2	8804	4528	7692	748	1767	17	220	514	8775
top	s1	Movie	15-Aug	Rajiv Chilaka	David Attenborough	United States	1-Jan-20	TV-MA	1 Season	Dramas, International Movies	Paranormal activity at a lush, abandoned prope...
freq	1	6131	2	19	19	2818	109	3207	1793	362	4

In [10]:

```
df.isna().sum()
```

Out[10]:

```
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
```

In [11]:

```
#3 missing values are found in duration column , and it is also found that by mistake tho
se data got entered in rating column
```

In [12]:

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

Out[12]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	descrip
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	4-Apr-17	2017	74 min	NaN	Movies	Louis C.K. mu on religion, ete love,
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	16-Sep-16	2010	84 min	NaN	Movies	Emmy-winr comedy wi Louis C.K. bri
5813	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	15-Aug-16	2015	66 min	NaN	Movies	The comic puts tradem hilarious/thoug

In [13]:

```
ind = df[df['duration'].isna()].index
```

In [14]:

```
df.loc[ind] = df.loc[ind].fillna(method = 'ffill' , axis = 1)
```

In [15]:

```
# replaced the wrong entries done in the rating column
df.loc[ind ,'rating'] = 'Not Available'
```

In [16]:

```
df.loc[ind]
```

Out[16]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	desc
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	4-Apr-17	2017	Not Available	74 min	Movies	Louis C.K. on religion, lo
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	16-Sep-16	2010	Not Available	84 min	Movies	Emmy-v comedy Louis C.K.
5813	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	15-Aug-16	2015	Not Available	66 min	Movies	The comic p trac hilarious/tho

In [17]:

```
df[df.rating.isna()]
```

Out[17]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_
5989	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava	NaN	Oprah Winfrey, Ava DuVernay	NaN	26-Jan-17	2017	NaN	37 min	Movi

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_
6827	s6828	TV Show	Gargantia on the Verdurous Planet	NaN	Kaito Ishikawa, Hisako Kanemoto, Ai Kayano, Ka...	Japan	1-Dec-16	2013	NaN	1 Season	Anim Serie Internation TV Show
7312	s7313	TV Show	Little Lunch	NaN	Flynn Curry, Olivia Deeble, Madison Lu, Oisín ...	Australia	1-Feb-18	2015	NaN	1 Season	Kids' TV, T Comedi
7537	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Leone Frisa, Paolo Vaccarino, Francesco Miglio...	Italy	1-Mar-17	2015	NaN	115 min	Drama

In [18]:

```
indices = df[df.rating.isna()].index
indices
```

Out[18]:

```
Index([5989, 6827, 7312, 7537], dtype='int64')
```

In [19]:

```
df.loc[indices , 'rating'] = 'Not Available'
```

In [20]:

```
df.loc[indices]
```

Out[20]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	list
5989	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava ...	NaN	Oprah Winfrey, Ava DuVernay	NaN	26-Jan-17	2017	Not Available	37 min	M
6827	s6828	TV Show	Gargantia on the Verdurous Planet	NaN	Kaito Ishikawa, Hisako Kanemoto, Ai Kayano, Ka...	Japan	1-Dec-16	2013	Not Available	1 Season	A S Internat TV S
7312	s7313	TV Show	Little Lunch	NaN	Flynn Curry, Olivia Deeble, Madison Lu, Oisín ...	Australia	1-Feb-18	2015	Not Available	1 Season	Kids' T Com
7537	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Leone Frisa, Paolo Vaccarino, Francesco Miglio...	Italy	1-Mar-17	2015	Not Available	115 min	Dr

In [21]:

```
df.rating.unique()
```

Out[21]:

```
array(['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7', 'R',
       'TV-G', 'G', 'NC-17', 'Not Available', 'NR', 'TV-Y7-FV', 'UR'],
      dtype=object)
```

In [22]:

```
#In rating column , NR (Not rated) is same as UR (Unrated). lets change UR to NR
```

In [23]:

```
df.loc[df['rating'] == 'UR' , 'rating'] = 'NR'
df.rating.value_counts()
```

Out[23]:

```
rating
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
Not Available       7
TV-Y7-FV            6
NC-17               3
Name: count, dtype: int64
```

In [24]:

```
#dropped the null from date_added column
```

In [25]:

```
df.drop(df.loc[df['date_added'].isna()].index , axis = 0 , inplace = True)
```

In [26]:

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

Out[26]:

```
date_added
1-Jan-20     109
1-Nov-19      89
1-Mar-18      75
31-Dec-19     74
1-Oct-18      71
             ...
4-Dec-16       1
21-Nov-16      1
19-Nov-16      1
17-Nov-16      1
11-Jan-20      1
Name: count, Length: 1767, dtype: int64
```

In [27]:

```
df['date_added'] = pd.to_datetime(df['date_added'])
df['date_added']
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_11132\303252933.py:1: UserWarning: Could not i
nfer format, so each element will be parsed individually, falling back to `dateutil`. To
ensure parsing is consistent and as-expected, please specify a format.
```

```
df['date_added'] = pd.to_datetime(df['date_added'])
```

Out[27]:

```
0      2021-09-25
1      2021-09-24
2      2021-09-24
3      2021-09-24
4      2021-09-24
          ...
8802   2019-11-20
8803   2019-07-01
8804   2019-11-01
8805   2020-01-11
8806   2019-03-02
Name: date_added, Length: 8797, dtype: datetime64[ns]
```

In [28]:

```
df['year_added'] = df['date_added'].dt.year
```

In [29]:

```
df['month_added'] = df['date_added'].dt.month
```

In [30]:

```
df[['date_added' , 'year_added' , 'month_added']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8797 entries, 0 to 8806
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   date_added   8797 non-null   datetime64[ns]
 1   year_added   8797 non-null   int32
 2   month_added  8797 non-null   int32
dtypes: datetime64[ns](1), int32(2)
memory usage: 206.2 KB
```

In [31]:

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

Out[31]:

```
show_id            0
type               0
title              0
director        2624
cast             825
country          830
date_added         0
release_year       0
rating             0
duration           0
listed_in          0
description        0
year_added         0
month_added        0
dtype: int64
```

In [32]:

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

Out[32]:

```
array(['Movie', 'TV Show'], dtype=object)
```

In [33]:

```
movies  = df.loc[df['type'] == 'Movie']
tv_shows = df.loc[df['type'] == 'TV Show']
```

In [34]:

```
movies.duration.value_counts()
```

Out[34]:

```
duration
90 min     152
94 min     146
97 min     146
93 min     146
91 min     144
          ...
208 min      1
5 min        1
16 min       1
186 min      1
191 min      1
Name: count, Length: 205, dtype: int64
```

In [35]:

```
tv_shows.duration.value_counts()
```

Out[35]:

```
duration
1 Season      1793
2 Seasons      421
3 Seasons      198
4 Seasons       94
5 Seasons       64
6 Seasons       33
7 Seasons       23
8 Seasons       17
9 Seasons        9
10 Seasons       6
13 Seasons       2
15 Seasons       2
12 Seasons       2
17 Seasons       1
11 Seasons       1
Name: count, dtype: int64
```

In [36]:

```
#when was first movie added on netflix and when is the most recent movie added on netflix
as per data i.e. dataset duration
```

In [37]:

```
timeperiod = pd.Series((df['date_added'].min().strftime('%B %Y') , df['date_added'].max(
).strftime('%B %Y')))
timeperiod.index = ['first' , 'Most Recent']
timeperiod
```

Out[37]:

```
first            January 2008
Most Recent    September 2021
dtype: object
```

In [38]:

```
#The oldest and the most recent movie/TV show released on the Netflix in which year?
```

In [39]:

```
df.release_year.min() , df.release_year.max()
```

Out[39]:

(1925, 2021)

In [40]:

```
df.loc[(df.release_year == df.release_year.min()) | (df.release_year == df.release_year.max())].sort_values('release_year')
```

Out[40]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_i
4250	s4251	TV Show	Pioneers: First Women Filmmakers*	NaN	NaN	NaN	2018-12-30	1925	TV-14	1 Season	TV Show
966	s967	Movie	Get the Grift	Pedro Antonio	Marcus Majella, Samantha Schmütz, Caito Mainie...	Brazil	2021-04-28	2021	TV-MA	95 min	Comedies Internationa Movie
967	s968	TV Show	Headspace Guide to Sleep	NaN	Evelyn Lewis Prieto	NaN	2021-04-28	2021	TV-G	1 Season	Docuseries Science Nature T
968	s969	TV Show	Sexify	NaN	Aleksandra Skraba, Maria Sobocińska, Sandra Dr...	Poland	2021-04-28	2021	TV-MA	1 Season	International T Shows, T Comedies, T Drama
972	s973	TV Show	Fatma	NaN	Burcu Biricik, Uğur Yücel, Mehmet Yılmaz Ak, H...	Turkey	2021-04-27	2021	TV-MA	1 Season	International T Shows, T Dramas, T Thriller
...	...	...	...	...	...	...	...	...	...	...	.
466	s467	TV Show	My Unorthodox Life	NaN	NaN	NaN	2021-07-14	2021	TV-MA	1 Season	Reality T
467	s468	Movie	Private Network: Who Killed Manuel Buendía?	Manuel Alcalá	Daniel Giménez Cacho	NaN	2021-07-14	2021	TV-MA	100 min	Documentaries Internationa Movie
468	s469	Movie	The Guide to the Perfect Family	Ricardo Trogi	Louis Morissette, Émilie Bierre, Catherine Cha...	NaN	2021-07-14	2021	TV-MA	102 min	Comedies Dramas Internationa Movie
471	s472	Movie	Day of Destiny	Akay Mason, Abosi Ogba	Olumide Oworu, Denola Grey, Gbemi Akinlade, Ji...	NaN	2021-07-13	2021	TV-PG	110 min	Children Family Movies Dramas Internationa.
		TV	The Netflix		David Spade, London	United			TV-	1	Stand-U Comedy & Tal

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in
8437	s8438	Show	Afterparty	NaN	Hughes Fortune Feimster	States	2021-01-02	2021	MA	Season	Comedie

593 rows × 14 columns

In [41]:

```
#Which are different ratings available on Netflix in each type of content? Check the numb
er of content released in each type
```

In [42]:

```
df.groupby(['type' , 'rating'])['show_id'].count()
```

Out[42]:

```
type     rating
Movie    G                 41
         NC-17              3
         NR                78
         Not Available      5
         PG               287
         PG-13            490
         R                797
         TV-14           1427
         TV-G             126
         TV-MA           2062
         TV-PG            540
         TV-Y             131
         TV-Y7            139
         TV-Y7-FV           5
TV Show  NR                 4
         Not Available      2
         R                  2
         TV-14            730
         TV-G              94
         TV-MA           1143
         TV-PG            321
         TV-Y             175
         TV-Y7            194
         TV-Y7-FV           1
Name: show_id, dtype: int64
```

In [43]:

```
#Working on the columns having maximum null values and the columns having comma separated
multiple values for each record -Country column
```

In [44]:

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

Out[44]:

```
country
United States                          2812
India                                   972
United Kingdom                          418
Japan                                   244
South Korea                             199
                                       ...
Romania, Bulgaria, Hungary                1
Uruguay, Guatemala                        1
France, Senegal, Belgium                  1
Mexico, United States, Spain, Colombia    1
United Arab Emirates, Jordan              1
Name: count, Length: 748, dtype: int64
```

In [45]:

```
#We see that many movies are produced in more than 1 country. Hence, the country column h
as comma separated values of countries.
#This makes it difficult to analyse how many movies were produced in each country. We can
use explode function in pandas to split the country column into different rows.
#we are Creating a separate table for country , to avoid the duplicasy of records in our
origional table after exploding.
```

In [46]:

```
country_tb = df[['show_id' , 'type' , 'country']]
country_tb.dropna(inplace = True)
country_tb['country'] = country_tb['country'].apply(lambda x : x.split(','))
country_tb = country_tb.explode('country')
country_tb
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_11132\679330132.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  country_tb.dropna(inplace = True)
C:\Users\ASUS\AppData\Local\Temp\ipykernel_11132\679330132.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  country_tb['country'] = country_tb['country'].apply(lambda x : x.split(','))
```

Out[46]:

	show_id	type	country
0	s1	Movie	United States
1	s2	TV Show	South Africa
4	s5	TV Show	India
7	s8	Movie	United States
7	s8	Movie	Ghana
...	...	...	...
8801	s8802	Movie	Jordan
8802	s8803	Movie	United States
8804	s8805	Movie	United States
8805	s8806	Movie	United States
8806	s8807	Movie	India

10010 rows × 3 columns

In [47]:

```
# some duplicate values are found, which have unnecessary spaces. some empty strings foun
d
country_tb['country'] = country_tb['country'].str.strip()
```

In [48]:

```
country_tb.loc[country_tb['country'] == '']
```

Out[48]:

	show_id	type	country
193	s194	TV Show	
365	s366	Movie	

	show_id	type	country
1192	s1193	Movie	
2224	s2225	Movie	
4653	s4654	Movie	
5925	s5926	Movie	
7007	s7008	Movie	

In [49]:

```
country_tb = country_tb.loc[country_tb['country'] != '']
```

In [50]:

```
country_tb['country'].nunique()
```

Out[50]:

122

In [51]:

```
#Netflix has movies from the total 122 countries.Total movies and tv shows in each countr
y
```

In [52]:

```
x = country_tb.groupby(['country' , 'type'])['show_id'].count().reset_index()
x.pivot(index = ['country'] , columns = 'type' , values = 'show_id').sort_values('Movie'
,ascending = False)
```

Out[52]:

type	Movie	TV Show
country		
United States	2752.0	932.0
India	962.0	84.0
United Kingdom	534.0	271.0
Canada	319.0	126.0
France	303.0	90.0
...	...	...
Azerbaijan	NaN	1.0
Belarus	NaN	1.0
Cuba	NaN	1.0
Cyprus	NaN	1.0
Puerto Rico	NaN	1.0

122 rows × 2 columns

In [53]:

```
#Director column
```

In [54]:

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

Out[54]:

```
director
Rajiv Chilaka                     19
Raúl Campos, Jan Suter            18
Marcus Raboy                      16
Suhas Kadav                       16
```

```
Jay Karas                          14
                                   ..
Raymie Muzquiz, Stu Livingston      1
Joe Menendez                        1
Eric Bross                          1
Will Eisenberg                      1
Mozez Singh                         1
Name: count, Length: 4528, dtype: int64
```

In [55]:

```
#There are some movies which are directed by multiple directors. Hence multiple names of
directors are given in comma separated format. We will explode the director column as wel
l. It will create many duplicate records in originaltable hence we created separate table
for directors.
```

In [56]:

```
dir_tb = df[['show_id' , 'type' , 'director']]
dir_tb.dropna(inplace = True)
dir_tb['director'] = dir_tb['director'].apply(lambda x : x.split(','))
dir_tb
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_11132\2245723733.py:2: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  dir_tb.dropna(inplace = True)
C:\Users\ASUS\AppData\Local\Temp\ipykernel_11132\2245723733.py:3: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  dir_tb['director'] = dir_tb['director'].apply(lambda x : x.split(','))
```

Out[56]:

	show_id	type	director
0	**s1**	**Movie**	**[Kirsten Johnson]**
2	**s3**	**TV Show**	**[Julien Leclercq]**
5	**s6**	**TV Show**	**[Mike Flanagan]**
6	**s7**	**Movie**	**[Robert Cullen, José Luis Ucha]**
7	**s8**	**Movie**	**[Haile Gerima]**
...	**...**	**...**	**...**
8801	**s8802**	**Movie**	**[Majid Al Ansari]**
8802	**s8803**	**Movie**	**[David Fincher]**
8804	**s8805**	**Movie**	**[Ruben Fleischer]**
8805	**s8806**	**Movie**	**[Peter Hewitt]**
8806	**s8807**	**Movie**	**[Mozez Singh]**

6173 rows × 3 columns

In [57]:

```
dir_tb = dir_tb.explode('director')
```

In [58]:

```
dir_tb['director'] = dir_tb['director'].str.strip()
```

In [59]:

```
# checking if empty stirngs are there in director column
dir_tb.director.apply(lambda x : True if len(x) == 0 else False).value_counts()
```

Out[59]:

```
director
False    6978
Name: count, dtype: int64
```

In [60]:

```
dir_tb
```

Out[60]:

	show_id	type	director
0	s1	Movie	Kirsten Johnson
2	s3	TV Show	Julien Leclercq
5	s6	TV Show	Mike Flanagan
6	s7	Movie	Robert Cullen
6	s7	Movie	José Luis Ucha
...	...	...	...
8801	s8802	Movie	Majid Al Ansari
8802	s8803	Movie	David Fincher
8804	s8805	Movie	Ruben Fleischer
8805	s8806	Movie	Peter Hewitt
8806	s8807	Movie	Mozez Singh

6978 rows × 3 columns

In [61]:

```
dir_tb['director'].nunique()
```

Out[61]:

```
4993
```

In [62]:

```
#There are total 4993 unique directors in the dataset.Total movies and tv shows directed
by each director
```

In [63]:

```
x = dir_tb.groupby(['director' , 'type'])['show_id'].count().reset_index()
x.pivot(index= ['director'] , columns = 'type' , values = 'show_id').sort_values('Movie'
,ascending = False)
```

Out[63]:

type	Movie	TV Show
director		
Rajiv Chilaka	22.0	NaN
Jan Suter	21.0	NaN
Raúl Campos	19.0	NaN
Suhas Kadav	16.0	NaN
Marcus Raboy	15.0	1.0

type	Movie	TV Show
Vijay S. Bhanushali director	NaN	1.0
Wouter Bouvijn	NaN	1.0
YC Tom Lee	NaN	1.0
Yasuhiro Irie	NaN	1.0
Yim Pilsung	NaN	1.0

4993 rows × 2 columns

In [64]:

```
#'listed_in' column to understand more about genres
```

In [65]:

```
genre_tb = df[['show_id' , 'type', 'listed_in']]
```

In [66]:

```
genre_tb['listed_in'] = genre_tb['listed_in'].apply(lambda x : x.split(','))
genre_tb = genre_tb.explode('listed_in')
genre_tb['listed_in'] = genre_tb['listed_in'].str.strip()
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_11132\1430222457.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  genre_tb['listed_in'] = genre_tb['listed_in'].apply(lambda x : x.split(','))
```

In [67]:

```
genre_tb
```

Out[67]:

	show_id	type	listed_in
0	s1	Movie	Documentaries
1	s2	TV Show	International TV Shows
1	s2	TV Show	TV Dramas
1	s2	TV Show	TV Mysteries
2	s3	TV Show	Crime TV Shows
...	...	...	...
8805	s8806	Movie	Children & Family Movies
8805	s8806	Movie	Comedies
8806	s8807	Movie	Dramas
8806	s8807	Movie	International Movies
8806	s8807	Movie	Music & Musicals

19303 rows × 3 columns

In [68]:

```
genre_tb.listed_in.unique()
```

Out[68]:

```
array(['Documentaries', 'International TV Shows', 'TV Dramas',
```

```
'TV Mysteries', 'Crime TV Shows', 'TV Action & Adventure',
'Docuseries', 'Reality TV', 'Romantic TV Shows', 'TV Comedies',
'TV Horror', 'Children & Family Movies', 'Dramas',
'Independent Movies', 'International Movies', 'British TV Shows',
'Comedies', 'Spanish-Language TV Shows', 'Thrillers',
'Romantic Movies', 'Music & Musicals', 'Horror Movies',
'Sci-Fi & Fantasy', 'TV Thrillers', "Kids' TV",
'Action & Adventure', 'TV Sci-Fi & Fantasy', 'Classic Movies',
'Anime Features', 'Sports Movies', 'Anime Series',
'Korean TV Shows', 'Science & Nature TV', 'Teen TV Shows',
'Cult Movies', 'TV Shows', 'Faith & Spirituality', 'LGBTQ Movies',
'Stand-Up Comedy', 'Movies', 'Stand-Up Comedy & Talk Shows',
'Classic & Cult TV'], dtype=object)
```

In [69]:

```
genre_tb.listed_in.nunique()
```

Out[69]:

```
42
```

In [70]:

```
df.merge(genre_tb , on = 'show_id' ).groupby(['type_y'])['listed_in_y'].nunique()
```

Out[70]:

```
type_y
Movie      20
TV Show    22
Name: listed_in_y, dtype: int64
```

In [71]:

```
#Movies have 20 genres and TV shows have 22 genres.
```

In [72]:

```
# total movies/TV shows in each genre
x = genre_tb.groupby(['listed_in' , 'type'])['show_id'].count().reset_index()
x.pivot(index = 'listed_in' , columns = 'type' , values = 'show_id').sort_index()
```

Out[72]:

type	Movie	TV Show
listed_in		
Action & Adventure	859.0	NaN
Anime Features	71.0	NaN
Anime Series	NaN	175.0
British TV Shows	NaN	252.0
Children & Family Movies	641.0	NaN
Classic & Cult TV	NaN	26.0
Classic Movies	116.0	NaN
Comedies	1674.0	NaN
Crime TV Shows	NaN	469.0
Cult Movies	71.0	NaN
Documentaries	869.0	NaN
Docuseries	NaN	394.0
Dramas	2427.0	NaN
Faith & Spirituality	65.0	NaN
Horror Movies	357.0	NaN

type	Movie	TV Show
listed_in		
Independent Movies	756.0	NaN
International Movies	2752.0	NaN
International TV Shows	NaN	1350.0
Kids' TV	NaN	449.0
Korean TV Shows	NaN	151.0
LGBTQ Movies	102.0	NaN
Movies	57.0	NaN
Music & Musicals	375.0	NaN
Reality TV	NaN	255.0
Romantic Movies	616.0	NaN
Romantic TV Shows	NaN	370.0
Sci-Fi & Fantasy	243.0	NaN
Science & Nature TV	NaN	92.0
Spanish-Language TV Shows	NaN	173.0
Sports Movies	219.0	NaN
Stand-Up Comedy	343.0	NaN
Stand-Up Comedy & Talk Shows	NaN	56.0
TV Action & Adventure	NaN	167.0
TV Comedies	NaN	574.0
TV Dramas	NaN	762.0
TV Horror	NaN	75.0
TV Mysteries	NaN	98.0
TV Sci-Fi & Fantasy	NaN	83.0
TV Shows	NaN	16.0
TV Thrillers	NaN	57.0
Teen TV Shows	NaN	69.0
Thrillers	577.0	NaN

In [73]:

```
# Exploring cast column
```

In [74]:

```
cast_tb = df[['show_id' , 'type' ,'cast']]
cast_tb.dropna(inplace = True)
cast_tb['cast'] = cast_tb['cast'].apply(lambda x : x.split(','))
cast_tb = cast_tb.explode('cast')
cast_tb
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_11132\2268781787.py:2: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  cast_tb.dropna(inplace = True)
C:\Users\ASUS\AppData\Local\Temp\ipykernel_11132\2268781787.py:3: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  cast_tb['cast'] = cast_tb['cast'].apply(lambda x : x.split(','))
```

Out[74]:

	show_id	type	cast
1	s2	TV Show	Ama Qamata
1	s2	TV Show	Khosi Ngema
1	s2	TV Show	Gail Mabalane
1	s2	TV Show	Thabang Molaba
1	s2	TV Show	Dillon Windvogel
...	...	...	...
8806	s8807	Movie	Manish Chaudhary
8806	s8807	Movie	Meghna Malik
8806	s8807	Movie	Malkeet Rauni
8806	s8807	Movie	Anita Shabdish
8806	s8807	Movie	Chittaranjan Tripathy

64057 rows × 3 columns

In [75]:

```
cast_tb['cast'] = cast_tb['cast'].str.strip()
```

In [76]:

```
# checking empty strings
cast_tb[cast_tb['cast'] == '']
```

Out[76]:

	show_id	type	cast

In [77]:

```
# Total actors on the Netflix
cast_tb.cast.nunique()
```

Out[77]:

```
36403
```

In [78]:

```
# Total movies/TV shows by each actor
x = cast_tb.groupby(['cast' , 'type'])['show_id'].count().reset_index()
x.pivot(index = 'cast' , columns = 'type' , values = 'show_id').sort_values('TV Show' ,
ascending = False)
```

Out[78]:

type	Movie	TV Show
cast		
Takahiro Sakurai	7.0	25.0
Yuki Kaji	10.0	19.0
Junichi Suwabe	4.0	17.0
Daisuke Ono	5.0	17.0
Ai Kayano	2.0	17.0
...	...	...
Şerif Sezer	1.0	NaN
Şevket Çoruh	1.0	NaN

	type	Movie	TV Show
Şinasi Yurtsever	cast	3.0	NaN
Şükran Ovalı	cast	1.0	NaN
Şọpẹ́ Dìrísù	cast	1.0	NaN

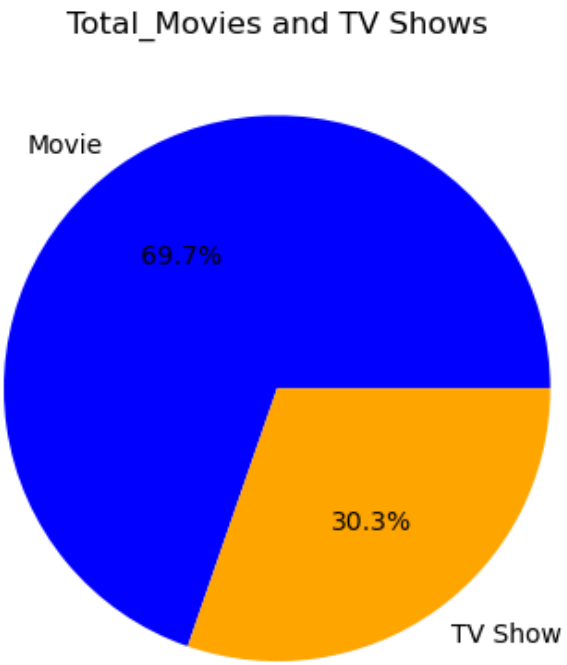
36403 rows × 2 columns

In [79]:

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

In [80]:

```
types = df.type.value_counts()
plt.pie(types,  labels=types.index, autopct='%1.1f%%' , colors = ['blue' , 'orange'])
plt.title('Total_Movies and TV Shows')
plt.show()
```



In [81]:

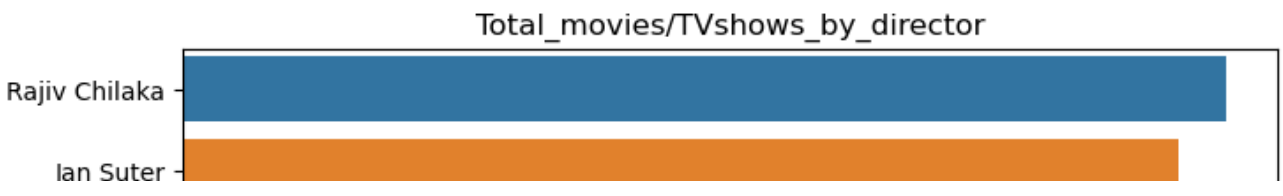
```
#It is observed that , around 70% content is Movies and around 30% content is TV shows.
```

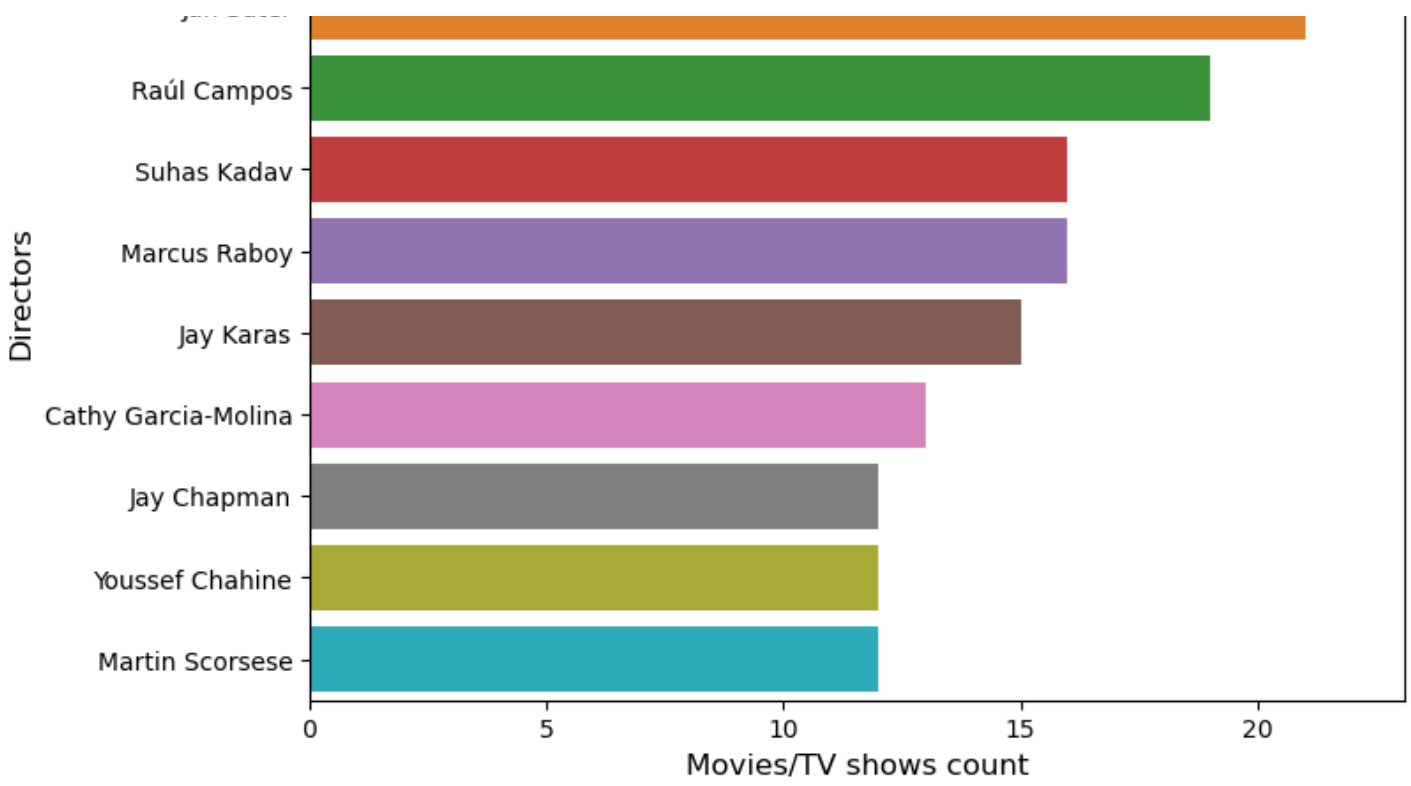
In [82]:

```
# total Movies directed by top 10 directors
top_10_dir = dir_tb.director.value_counts().head(10).index
df_new = dir_tb.loc[dir_tb['director'].isin(top_10_dir)]
```

In [83]:

```
plt.figure(figsize= (8 , 6))
sns.countplot(data=df_new, y='director', order=top_10_dir, orient='v')
plt.xlabel('total_movies/TV shows' , fontsize = 12)
plt.xlabel('Movies/TV shows count')
plt.ylabel('Directors' , fontsize = 12)
plt.title('Total_movies/TVshows_by_director')
plt.show()
```





In [84]:

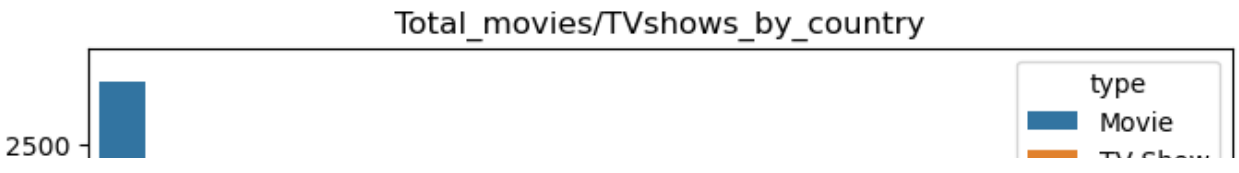
```
top_10_country = country_tb.country.value_counts().head(10).index
df_new = country_tb.loc[country_tb['country'].isin(top_10_country)]
x = df_new.groupby(['country' , 'type'])['show_id'].count().reset_index()
x.pivot(index = 'country' , columns = 'type' , values = 'show_id').sort_values('Movie',a
scending = False)
```

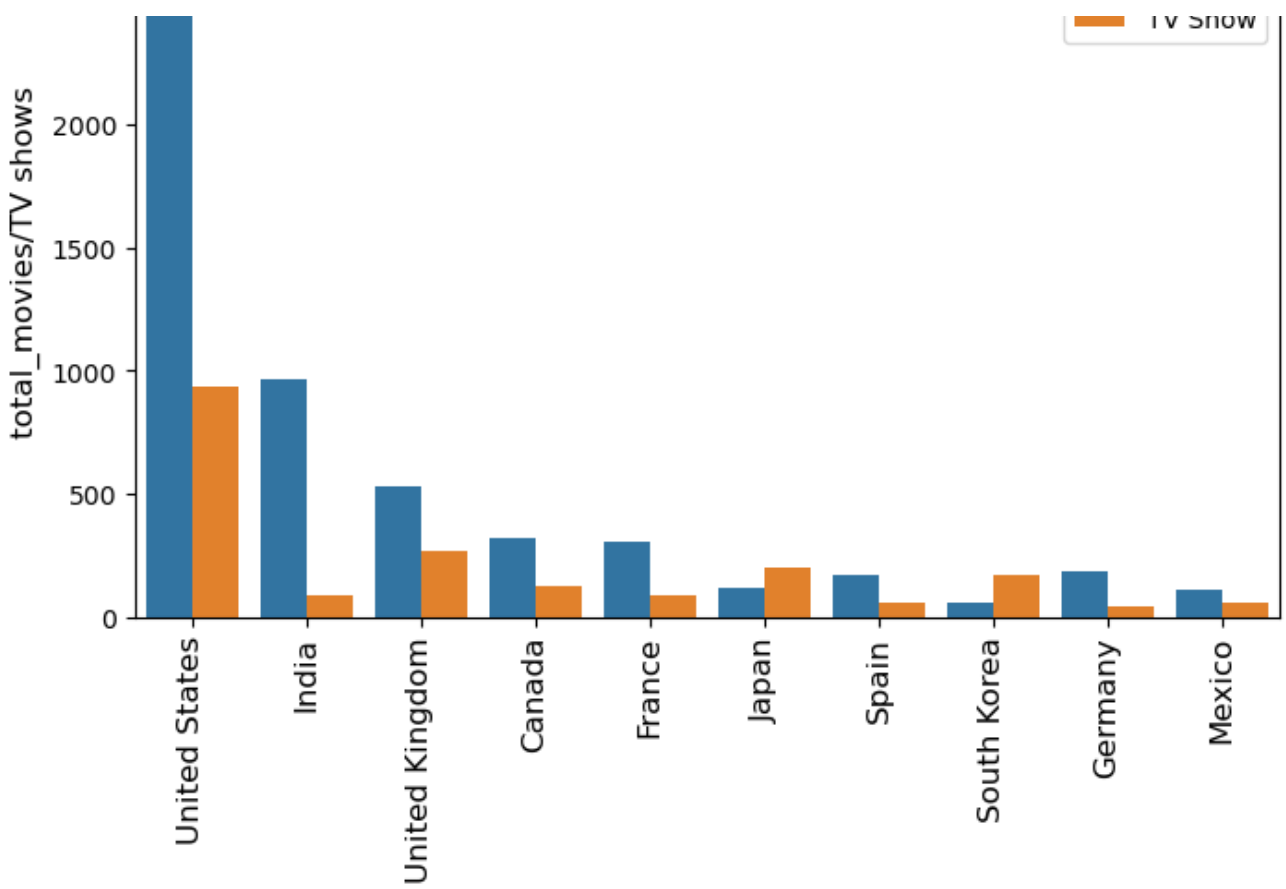
Out[84]:

type	Movie	TV Show
country		
United States	2752	932
India	962	84
United Kingdom	534	271
Canada	319	126
France	303	90
Germany	182	44
Spain	171	61
Japan	119	198
Mexico	111	58
South Korea	61	170

In [85]:

```
plt.figure(figsize= (8,5))
sns.countplot(data = df_new , x = 'country' , order = top_10_country , hue = 'type')
plt.xticks(rotation = 90 , fontsize = 12)
plt.ylabel('total_movies/TV shows' , fontsize = 12)
plt.xlabel('')
plt.title('Total_movies/TVshows_by_country')
plt.show()
```



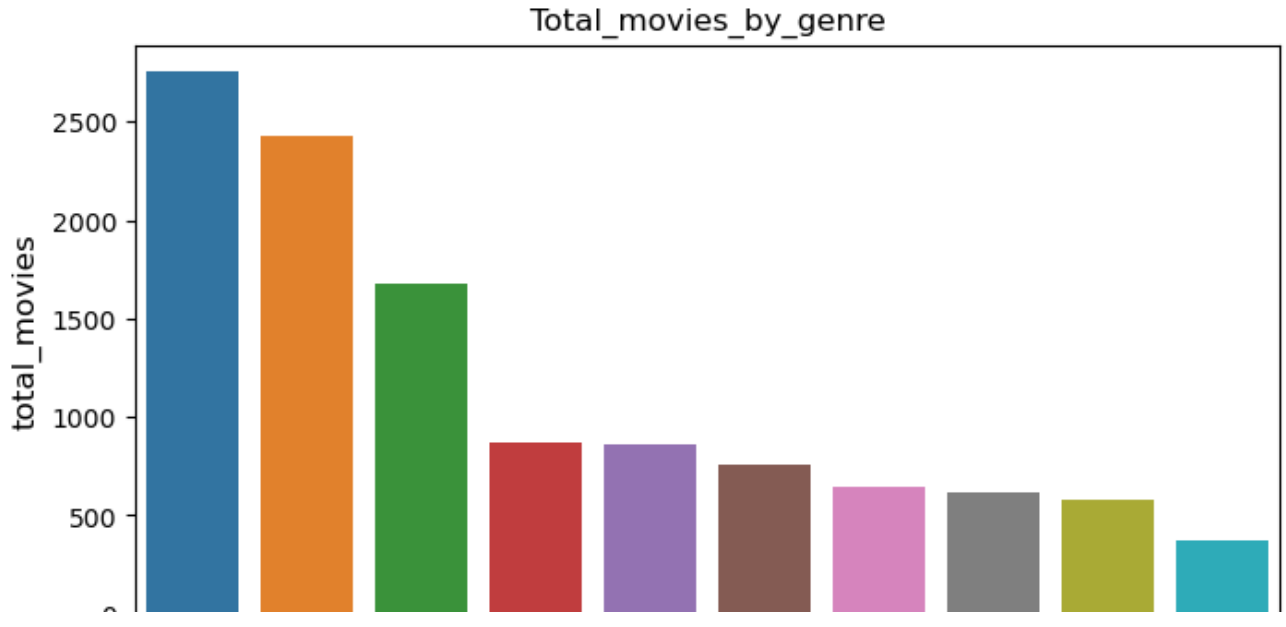


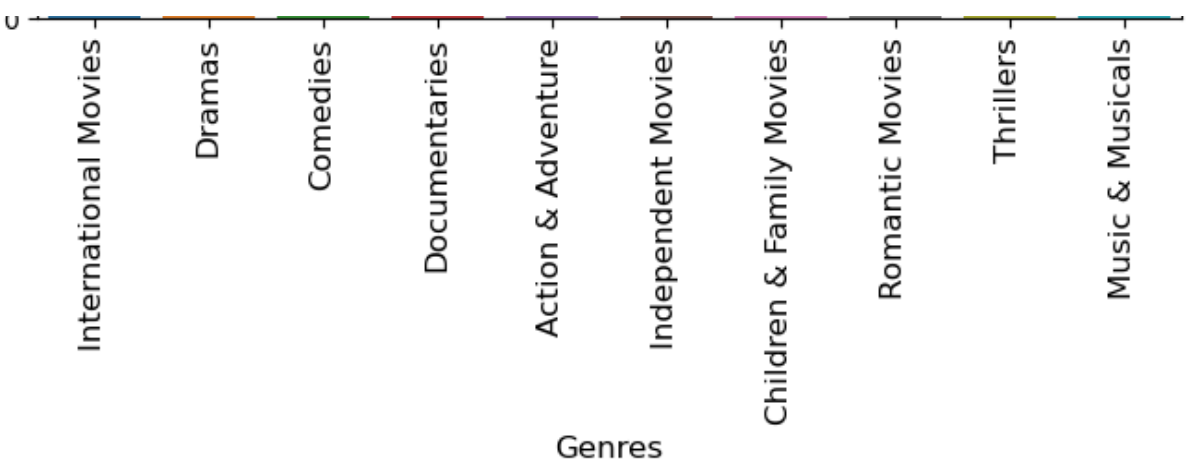
In [86]:

```
#United States is the HIGHEST contributor country on Netflix, followed by India and Unite
d Kingdom.
```

In [87]:

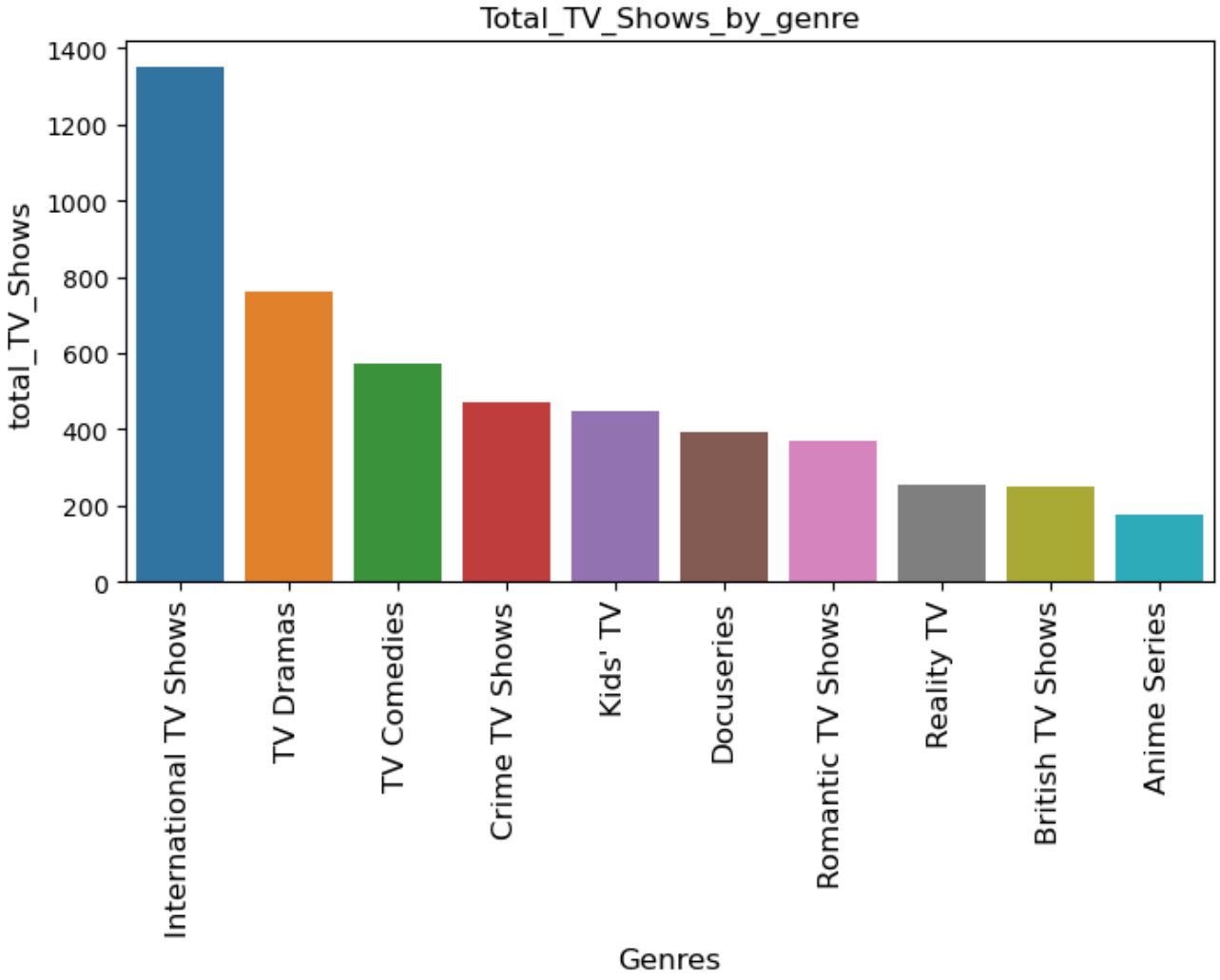
```
# Lets check the count for top 10 genres in Movies and TV_shows
top_10_movie_genres = genre_tb[genre_tb['type'] == 'Movie'].listed_in.value_counts().hea
d(10).index
df_movie = genre_tb.loc[genre_tb['listed_in'].isin(top_10_movie_genres)]
top_10_TV_genres = genre_tb[genre_tb['type'] == 'TV Show'].listed_in.value_counts().head
(10).index
df_tv = genre_tb.loc[genre_tb['listed_in'].isin(top_10_TV_genres)]
plt.figure(figsize= (8,4))
sns.countplot(data = df_movie , x = 'listed_in' , order = top_10_movie_genres)
plt.xticks(rotation = 90 , fontsize = 12)
plt.ylabel('total_movies' , fontsize = 12)
plt.xlabel('Genres' , fontsize = 12)
plt.title('Total_movies_by_genre')
plt.show()
```





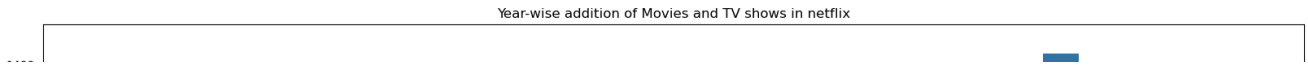
In [88]:

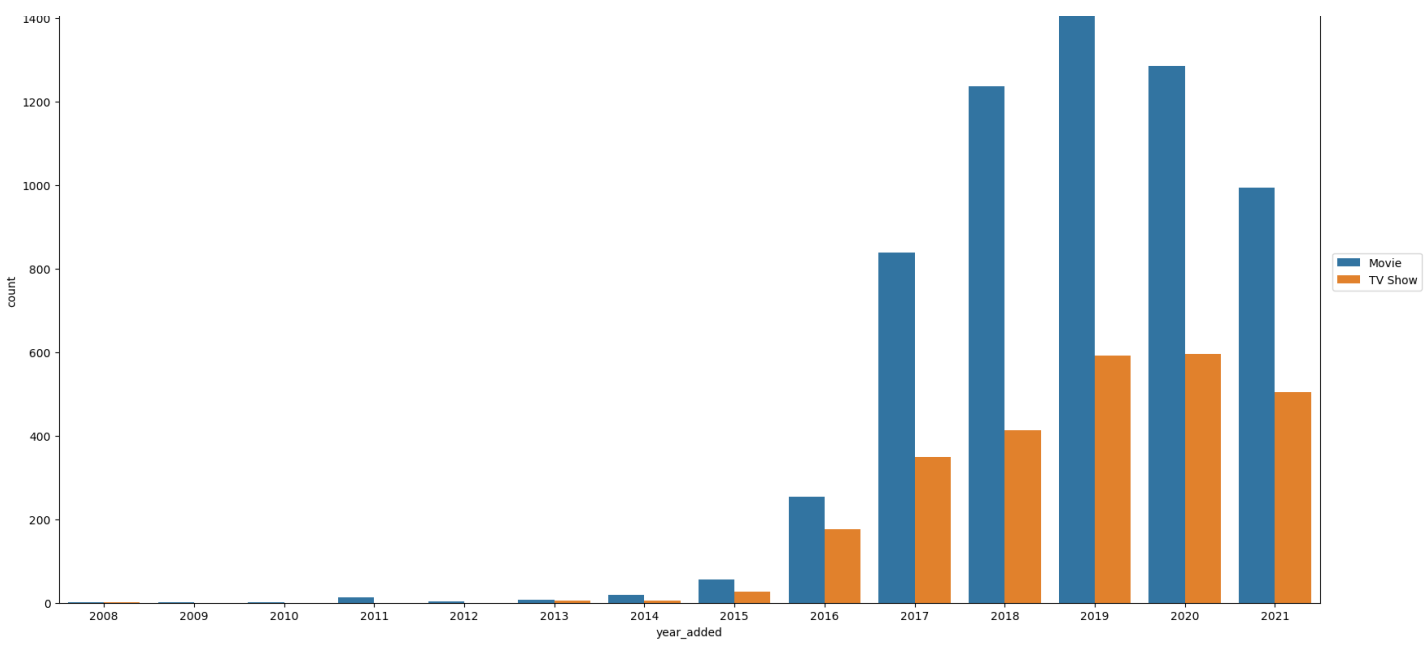
```
plt.figure(figsize= (8,4))
sns.countplot(data = df_tv , x = 'listed_in' , order = top_10_TV_genres)
plt.xticks(rotation = 90 , fontsize = 12)
plt.ylabel('total_TV_Shows' , fontsize = 12)
plt.xlabel('Genres' , fontsize = 12)
plt.title('Total_TV_Shows_by_genre')
plt.show()
```



In [89]:

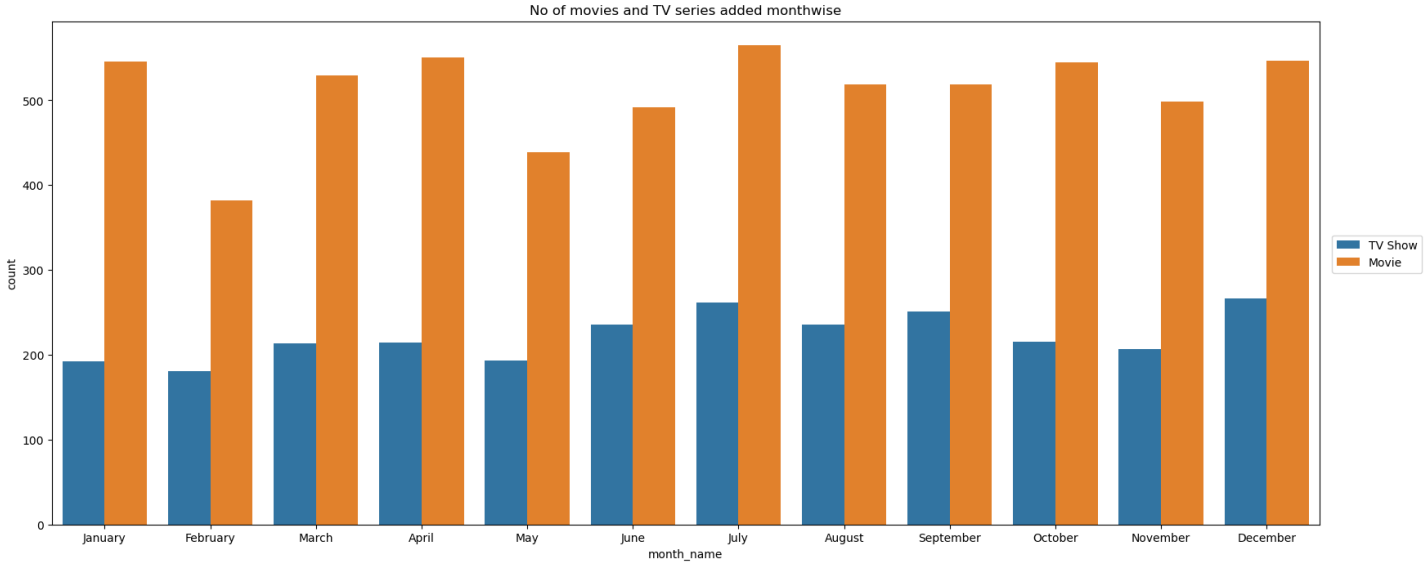
```
## Most shows added year on netflix between movies and TV shows
plt.figure(figsize=(20,10))
sns.countplot(x='year_added',data=df,hue='type')
plt.title('Year-wise addition of Movies and TV shows in netflix')
plt.legend(loc=(1.01,0.5))
plt.show()
```





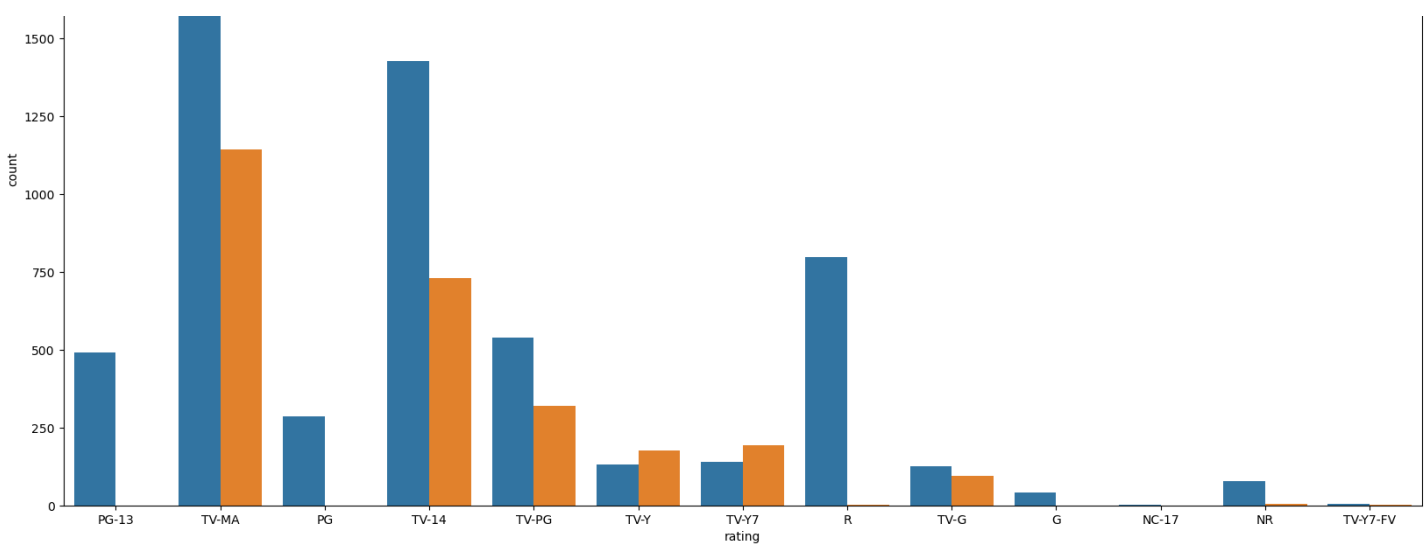
In [90]:

```
df_datetime = pd.DataFrame(df)
df_datetime['Year'] = df.date_added.dt.year
df_datetime['month'] = df.date_added.dt.month
df_datetime['day'] = df.date_added.dt.day_name()
df_datetime_month = df_datetime.sort_values(by ="month")
df_datetime_month['month_name'] = df.date_added.dt.month_name()
plt.figure(figsize=(20,8)) #defining fig size fot the graph image
sns.countplot(x = "month_name" , data = df_datetime_month , hue = "type")
plt.title("No of movies and TV series added monthwise") #title name of the plot
plt.legend(loc=(1.01,0.5))
plt.show()
```



In [91]:

```
# Rating wise barchart for netflix
rating_list = ['PG-13','TV-MA','PG','TV-14','TV-PG','TV-Y','TV-Y7','R','TV-G','G','NC-17'
,'NR','TV-Y7-FV','UR']
df_rating = df[(df.rating.isnull()==False) & (df.rating.apply(lambda x: x in rating_list
))]
plt.figure(figsize=(20,10))
sns.countplot(x='rating',data=df_rating,hue='type')
plt.show()
```

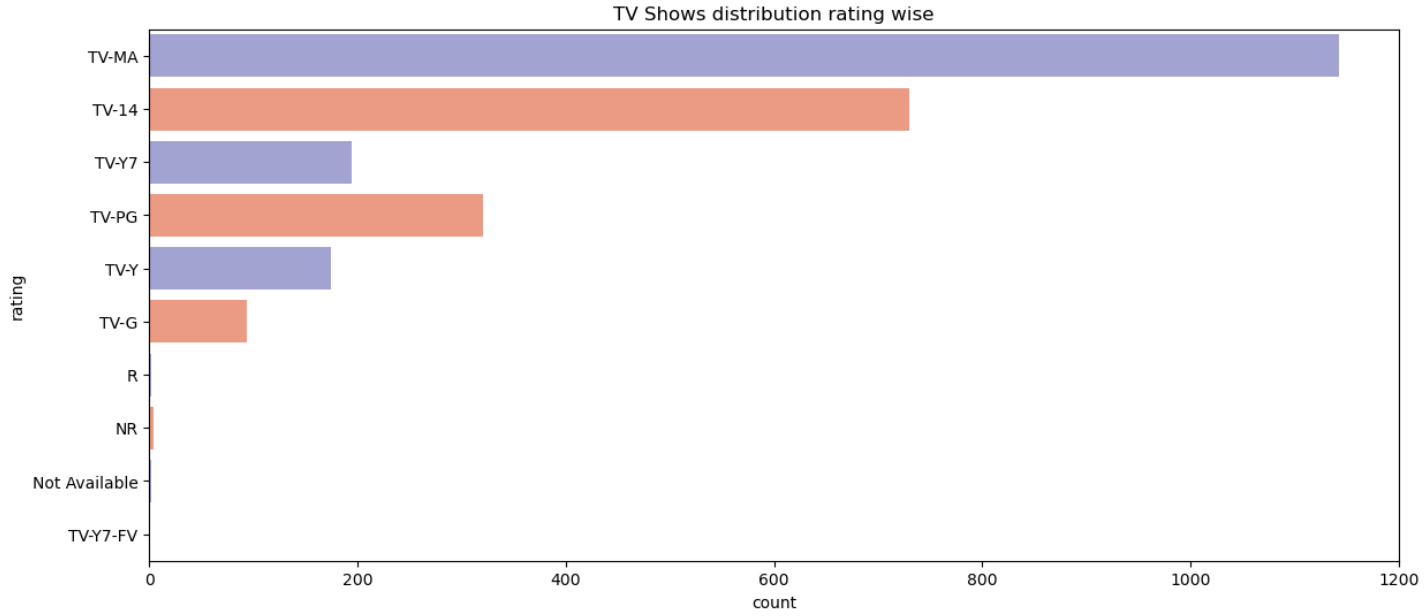


In [92]:

```
#Most movies are Rated TV-MA & TV-14
```

In [93]:

```
# countplot for Distributed TV Show ratings
plt.figure(figsize=(14,6))
tvshow_rating = df.loc[df["type"] == "TV Show" , ["type" , "rating"]]
sns.countplot( y="rating" , data =tvshow_rating,  palette=['#9b9bd9',"#fc9272"] )
plt.title("TV Shows distribution rating wise")
plt.show()
```

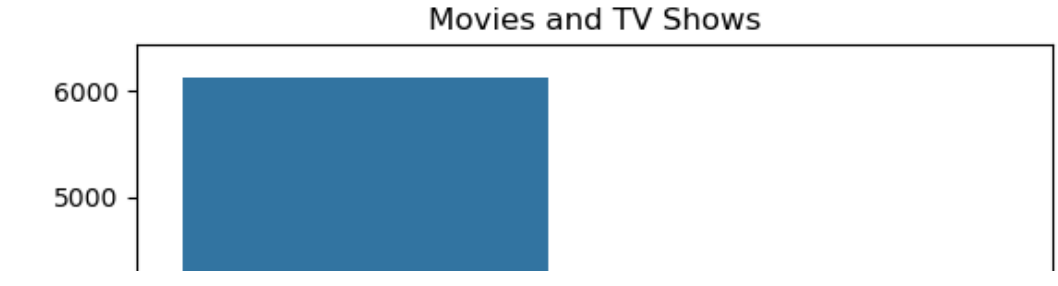


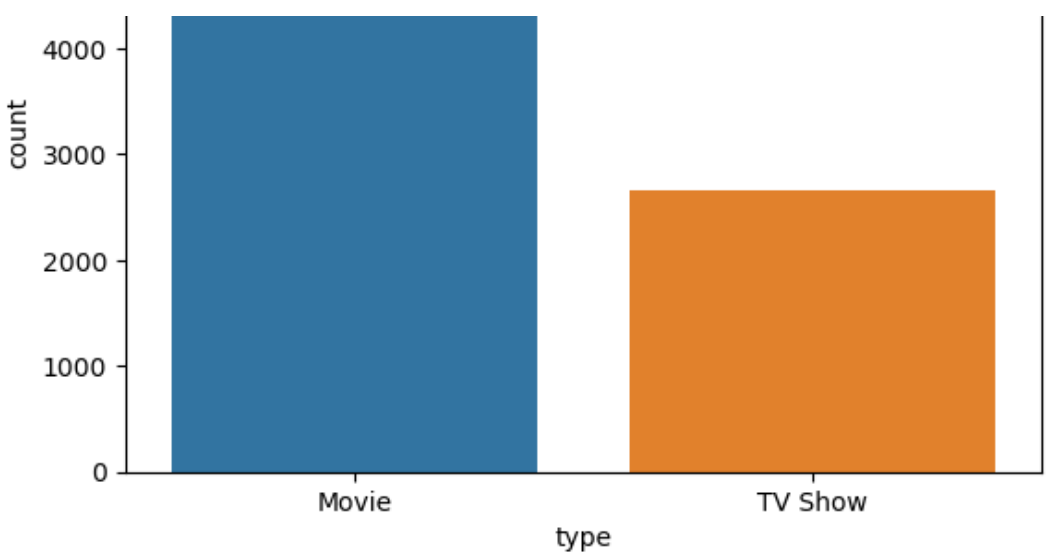
In [94]:

```
#Most TV shows are rated TV-MA
```

In [95]:

```
sns.countplot(x='type',data=df)
plt.title('Movies and TV Shows')
plt.show()
```



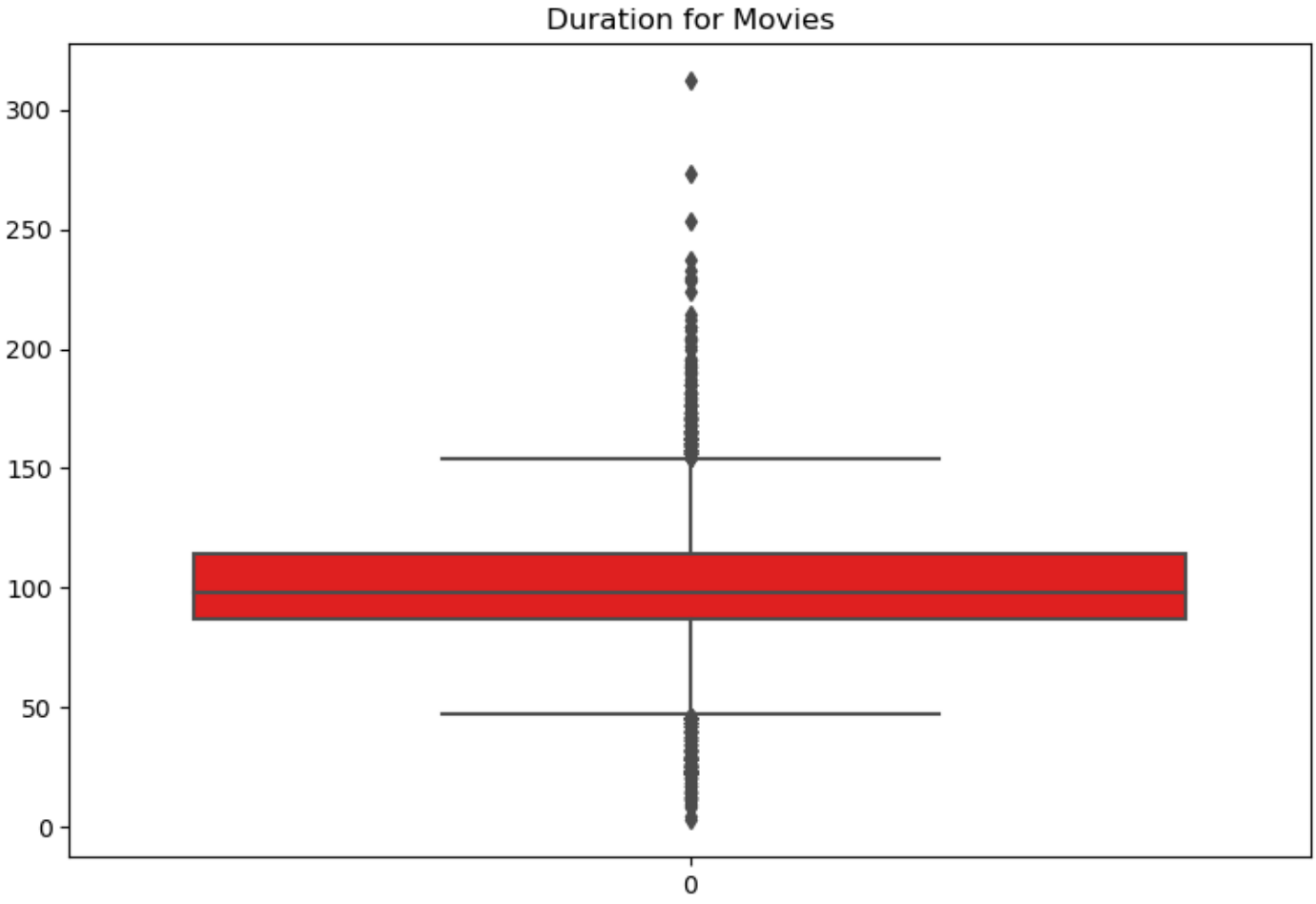


In [96]:

```
# Boxplot for duration of Movie
df = pd.read_csv('netflix.csv')
plt.figure(figsize=(20,6))
duration_df = df.loc[df.duration.str.contains("min")== True]['duration'].apply(lambda x:
x.split()[0]).astype('Int64')
plt.subplot(1,2,1)
plt.title('Duration for Movies')
sns.boxplot(duration_df , color = "red")
```

Out[96]:

```
<Axes: title={'center': 'Duration for Movies'}>
```

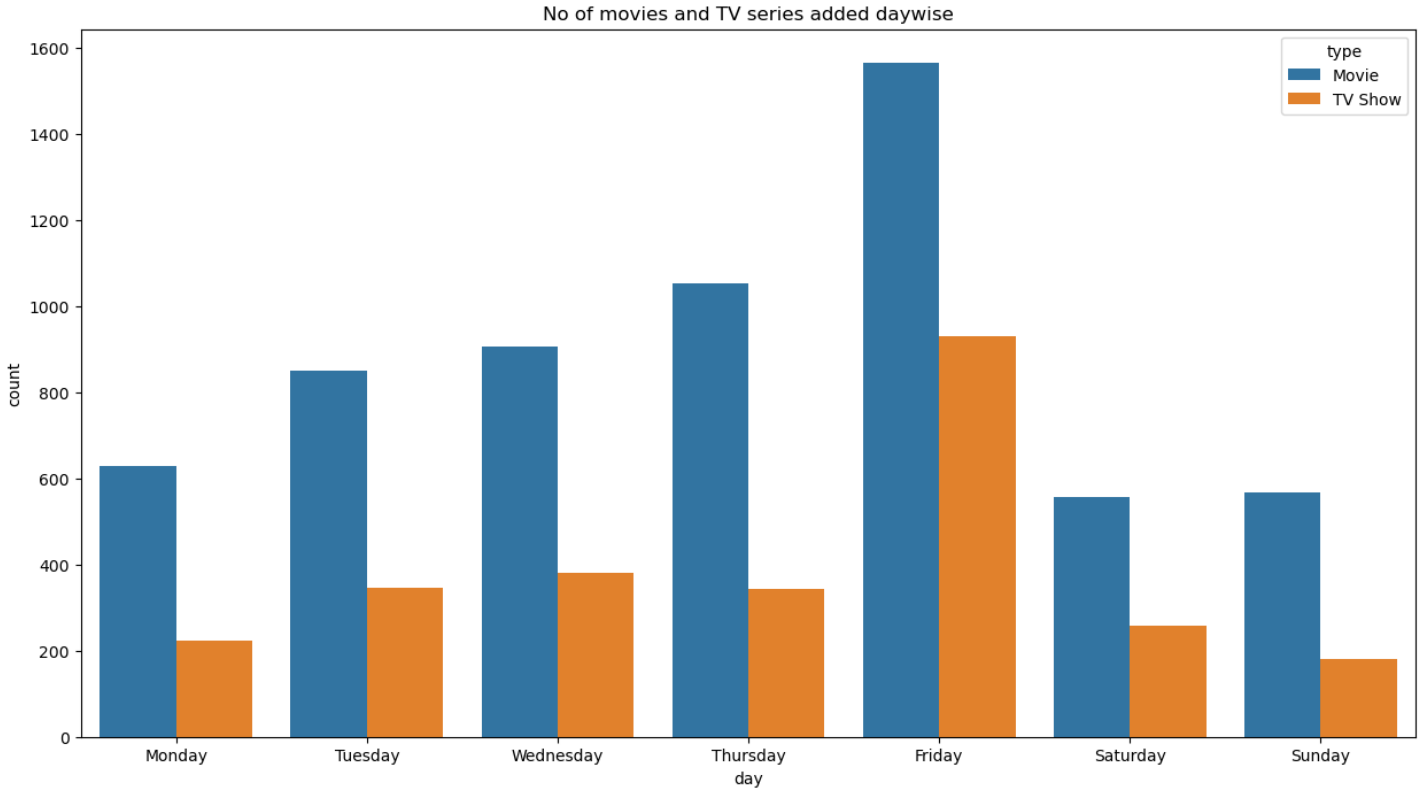


In [97]:

```
#Average duration of movies are around 100 min.
```

In [120]:

```
plt.figure(figsize=(15,8))
sns.countplot(x = "day" , data = df_datetime , hue = "type" ,  order=["Monday" , "Tuesday" , "Wednesday", "Thursday", "Friday", "Saturday" ,"Sunday"])
plt.title("No of movies and TV series added daywise")
plt.show()
```



In [121]:

```
#Most number of contents are uploaded during Friday and Highest number of contents uploaded is almost 2500.
```

In [110]:

```
top_20_country = country_tb.country.value_counts().head(20).index
top_20_country = country_tb.loc[country_tb['country'].isin(top_20_country)]
```

In [111]:

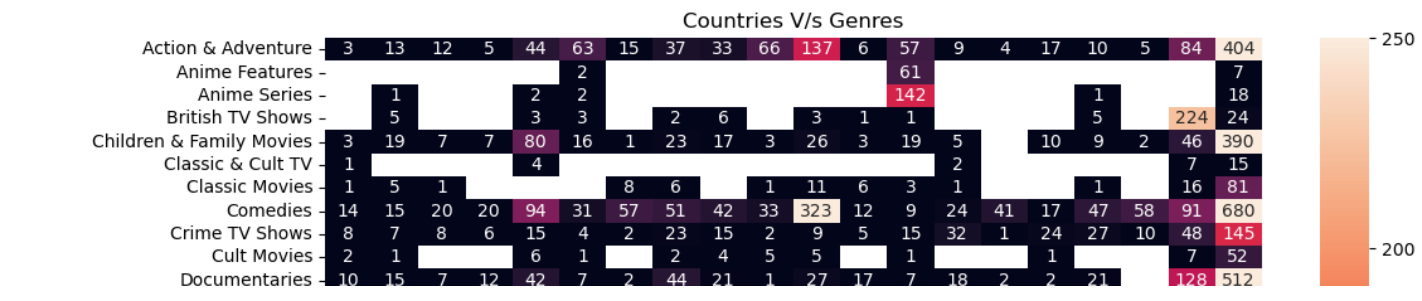
```
x = top_20_country.merge(genre_tb , on = 'show_id').drop_duplicates()
country_genre = x.groupby([ 'country' , 'listed_in'])['show_id'].count().sort_values(ascending = False).reset_index()
country_genre = country_genre.pivot(index = 'listed_in' , columns = 'country' , values = 'show_id')
```

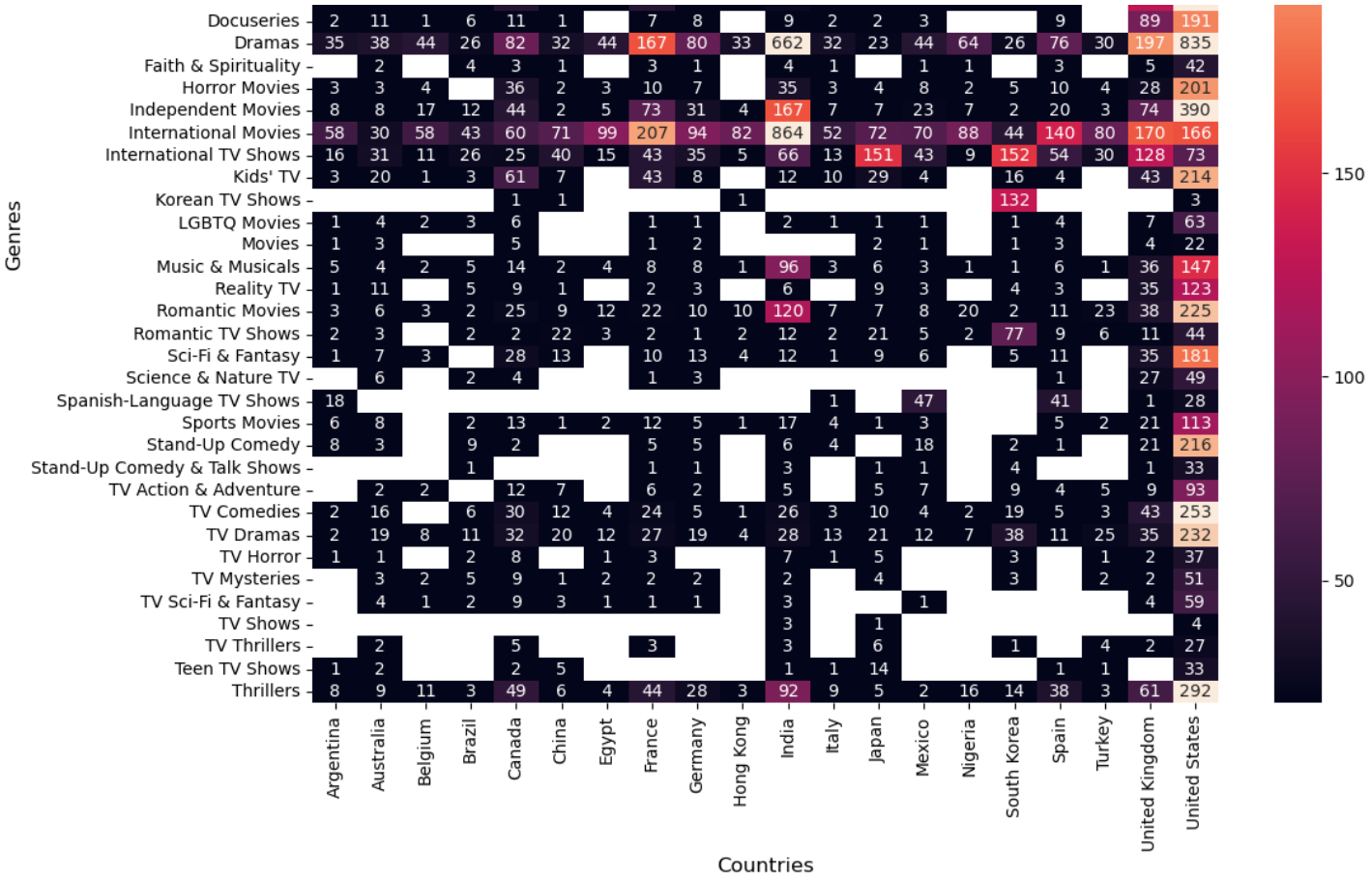
In [112]:

```
plt.figure(figsize = (12,10))
sns.heatmap(data = country_genre , annot = True , fmt=".0f" , vmin = 20 , vmax = 250,)
plt.xlabel('Countries' , fontsize = 12)
plt.ylabel('Genres' , fontsize = 12)
plt.title('Countries V/s Genres' , fontsize = 12)
```

Out[112]:

```
Text(0.5, 1.0, 'Countries V/s Genres')
```





In [ ]:

```
#Popular genres across countries: Action & Adventure, Children & Family Movies, Comedies, Dramas, International Movies & TV Shows, TV Dramas, Thrillers
#Country-specific genres: Korean TV shows (Korea), British TV Shows (UK), Anime features and Anime series (Japan), Spanish TV Shows (Argentina, Mexico and Spain)
#United States and UK have a good mix of almost all genres.
#Maximum International movies are produced in India.
```

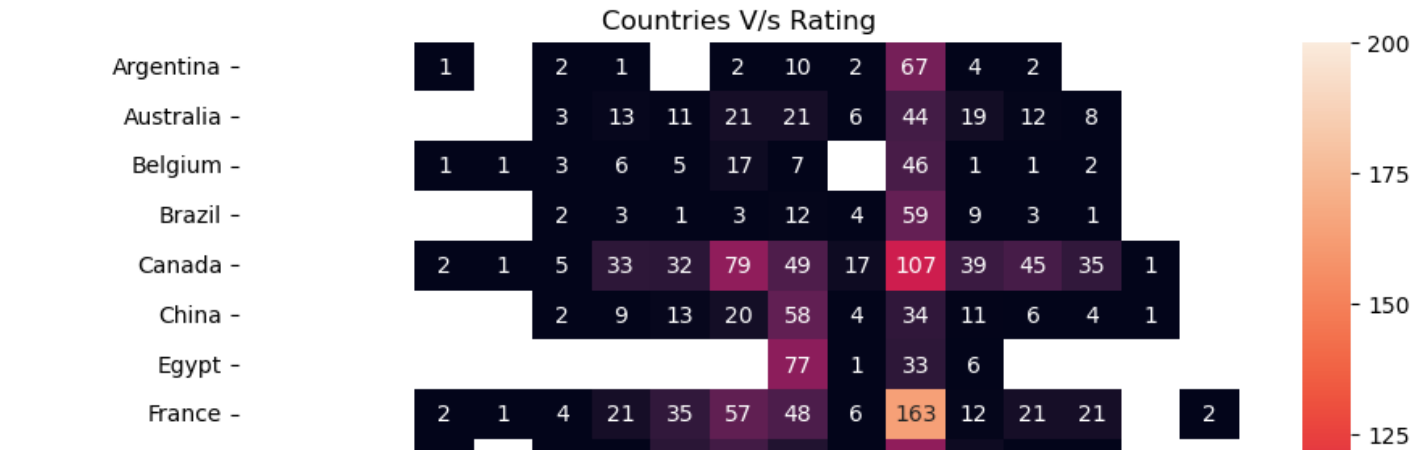
In [113]:

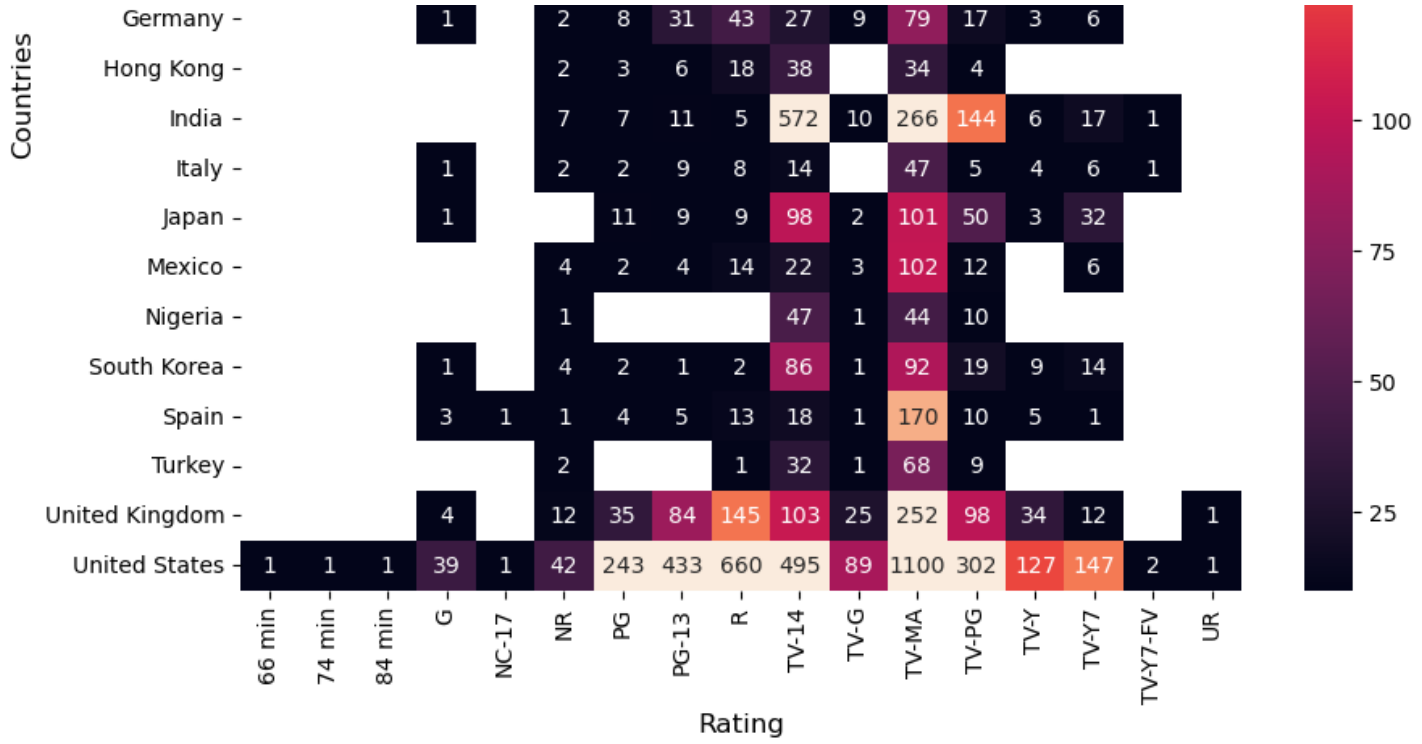
```
#Country-wise Rating of Content

x = top_20_country.merge(df , on = 'show_id').groupby(['country_x' , 'rating'])['show_id
'].count().reset_index()
country_rating = x.pivot(index = ['country_x'] , columns = 'rating' , values = 'show_id'
)
plt.figure(figsize = (10,8))
sns.heatmap(data = country_rating , annot = True , fmt=".0f"  , vmin = 10 , vmax=200)
plt.ylabel('Countries' , fontsize = 12)
plt.xlabel('Rating' , fontsize = 12)
plt.title('Countries V/s Rating' , fontsize = 12)
```

Out[113]:

```
Text(0.5, 1.0, 'Countries V/s Rating')
```





In [ ]:

```
#Overall, Netflix has an large amount of adult content across all countries (TV-MA & TV-14).
#India also has many titles rated TV-PG, other than TV-MA & TV-14.
#Only US, Canada, UK, France and Japan have content for young audiences (TV-Y & TV-Y7).
#There is scarce content for general audience (TV-G & G) across all countries except US.
```

In [ ]:

```
```

In [101]:

```
#top 5 directors
genre_list = [ 'Children & Family Movies', 'Comedies','Dramas', 'International Movies', 'Documentaries' ,
               'International TV Shows', 'Sci-Fi & Fantasy', 'Thrillers', 'Horror Movies'
]

x = dir_tb.merge(genre_tb , on = 'show_id').groupby([ 'listed_in' , 'director',])['show_id'].count().reset_index()

top_5_dir = x.loc[x['listed_in'] == 'Action & Adventure'].sort_values('show_id' , ascending = False).head()

for i in genre_list:
    new = x.loc[x['listed_in'] == i].sort_values('show_id' , ascending = False).head()
    top_5_dir = pd.concat([top_5_dir , new])

top_5_dir
```

Out[101]:

	listed_in	director	show_id
147	Action & Adventure	Don Michael Paul	9
550	Action & Adventure	S.S. Rajamouli	7
651	Action & Adventure	Toshiya Shinohara	7
215	Action & Adventure	Hidenori Inoue	7
606	Action & Adventure	Steven Spielberg	5
1215	Children & Family Movies	Rajiv Chilaka	22

	listed_in	director	show_id
1303	Children & Family Movies	Suhas Kadav	16
1211	Children & Family Movies	Prakash Satam	7
1241	Children & Family Movies	Robert Rodriguez	7
1288	Children & Family Movies	Steve Ball	6
1756	Comedies	David Dhawan	9
1905	Comedies	Hakan Algül	8
2686	Comedies	Suhas Kadav	8
2456	Comedies	Prakash Satam	7
1663	Comedies	Cathy Garcia-Molina	7
5935	Dramas	Youssef Chahine	12
4254	Dramas	Cathy Garcia-Molina	9
5099	Dramas	Martin Scorsese	9
4590	Dramas	Hanung Bramantyo	8
5544	Dramas	S.S. Rajamouli	7
7509	International Movies	Cathy Garcia-Molina	13
9330	International Movies	Youssef Chahine	10
9340	International Movies	Yılmaz Erdoğan	9
7620	International Movies	David Dhawan	8
8208	International Movies	Kunle Afolayan	8
3834	Documentaries	Vlad Yudin	6
3799	Documentaries	Thierry Donard	5
3217	Documentaries	Edward Cotterill	4
3262	Documentaries	Frank Capra	4
3075	Documentaries	Barry Avrich	4
9373	International TV Shows	Alastair Fothergill	3
9419	International TV Shows	Hsu Fu-chun	2
9436	International TV Shows	Jung-ah Im	2
9501	International TV Shows	Shin Won-ho	2
9478	International TV Shows	Pali Yahya	1
10752	Sci-Fi & Fantasy	Lilly Wachowski	4
10744	Sci-Fi & Fantasy	Lana Wachowski	4
10684	Sci-Fi & Fantasy	Guillermo del Toro	3
10790	Sci-Fi & Fantasy	Paul W.S. Anderson	3
10635	Sci-Fi & Fantasy	Barry Sonnenfeld	3
11974	Thrillers	Rathindran R Prasad	4
11698	Thrillers	David Fincher	4
11612	Thrillers	Anurag Kashyap	3
11636	Thrillers	Brad Anderson	3
11754	Thrillers	Gregory Hoblit	3
6280	Horror Movies	Rocky Soraya	6
6260	Horror Movies	Poj Arnon	5
6267	Horror Movies	Rathindran R Prasad	4
6191	Horror Movies	Leigh Janiak	3

	listed_in	director	show_id
6052	Horror Movies	Banjong Pisanthanakun	3

In [103]:

```
x = genre_tb.merge(country_tb , on = 'show_id').drop_duplicates()
x = x.groupby(['country' , 'listed_in'])['show_id'].count().reset_index()
x.loc[x['country'] == 'United States'].sort_values('show_id' , ascending = False).head(5
)

country_list = ['India'  , 'United Kingdom' , 'Canada' , 'France' , 'Japan']
top_5_genre = x.loc[x['country'].isin(['United States'])].sort_values('show_id' , ascend
ing = False).head(5)

for i in country_list:
    new = x.loc[x['country'] == i].sort_values('show_id' , ascending = False).head(5)
    top_5_genre = pd.concat( [top_5_genre , new] , ignore_index = True)
```

In [104]:

```
top_5_genre
```

Out[104]:

	country	listed_in	show_id
0	United States	Dramas	835
1	United States	Comedies	680
2	United States	Documentaries	512
3	United States	Action & Adventure	404
4	United States	Independent Movies	390
5	India	International Movies	864
6	India	Dramas	662
7	India	Comedies	323
8	India	Independent Movies	167
9	India	Action & Adventure	137
10	United Kingdom	British TV Shows	224
11	United Kingdom	Dramas	197
12	United Kingdom	International Movies	170
13	United Kingdom	International TV Shows	128
14	United Kingdom	Documentaries	128
15	Canada	Comedies	94
16	Canada	Dramas	82
17	Canada	Children & Family Movies	80
18	Canada	Kids' TV	61
19	Canada	International Movies	60
20	France	International Movies	207
21	France	Dramas	167
22	France	Independent Movies	73
23	France	Comedies	51
24	France	Thrillers	44
25	Japan	International TV Shows	151
26	Japan	Anime Series	142
27	Japan	International Movies	72

28	[illegible]	Anime [illegible]	[illegible]
29	Japan	Action & Adventure	57

In [105]:

```
x = cast_tb.merge(country_tb , on = 'show_id').drop_duplicates()
x = x.groupby(['country' , 'cast'])['show_id'].count().reset_index()
x.loc[x['country'].isin(['United States'])].sort_values('show_id' , ascending = False).h
ead(5)
```

Out[105]:

	country	cast	show_id
49405	United States	Tara Strong	22
48330	United States	Samuel L. Jackson	22
40463	United States	Fred Tatasciore	21
35733	United States	Adam Sandler	20
41672	United States	James Franco	19

In [106]:

```
country_list = ['India'  , 'United Kingdom' , 'Canada' , 'France' , 'Japan']
top_5_actors = x.loc[x['country'].isin(['United States'])].sort_values('show_id' , ascen
ding = False).head(5)
```

In [107]:

```
for i in country_list:
    new = x.loc[x['country'].isin([i])].sort_values('show_id' , ascending = False).head(
5)
    top_5_actors = pd.concat( [top_5_actors , new] , ignore_index = True)
```

In [108]:

```
# top 5 actors in top countries and their movies/tv shows count
top_5_actors
```

Out[108]:

	country	cast	show_id
0	United States	Tara Strong	22
1	United States	Samuel L. Jackson	22
2	United States	Fred Tatasciore	21
3	United States	Adam Sandler	20
4	United States	James Franco	19
5	India	Anupam Kher	40
6	India	Shah Rukh Khan	34
7	India	Naseeruddin Shah	31
8	India	Om Puri	29
9	India	Akshay Kumar	29
10	United Kingdom	David Attenborough	17
11	United Kingdom	John Cleese	16
12	United Kingdom	Michael Palin	14
13	United Kingdom	Terry Jones	12
14	United Kingdom	Eric Idle	12
15	Canada	John Paul Tremblay	14

	country	cast	show_id
16	Canada	Robb Wells	14
17	Canada	John Dunsworth	12
18	Canada	Vincent Tong	12
19	Canada	Ashleigh Ball	12
20	France	Wille Lindberg	5
21	France	Benoît Magimel	5
22	France	Gérard Depardieu	4
23	France	Blanche Gardin	4
24	France	Kristin Scott Thomas	4
25	Japan	Takahiro Sakurai	29
26	Japan	Yuki Kaji	28
27	Japan	Daisuke Ono	22
28	Japan	Junichi Suwabe	19
29	Japan	Ai Kayano	18

Insights based on Non-Graphical and Visual Analysis

- Around 70% content on Netflix is Movies and around 30% content is TV shows.
- The movies and TV shows uploading on the Netflix started from the year 2008, It had very lesser content till 2014.
- Year 2015 marks the drastic surge in the content getting uploaded on Netflix. It continues the uptrend since then and 2019 marks the highest number of movies and TV shows added on the Netflix. Year 2020 and 2021 has seen the drop in content added on Netflix, possibly because of Pandemic. But still , TV shows content have not dropped as drastic as movies.
- Since 2018, A drop in the movies is seen , but rise in TV shows is observed clearly. Being in continuous uptrend , TV shows surpassed the movies count in mid 2020. It shows the rise in popularity of tv shows in recent years.
- Netflix has movies from variety of directors. Around 4993 directors have their movies or tv shows on Netflix.
- Netflix has movies from total 122 countries, United States being the highset contributor with almost 37% of all the content.
- The release year for shows is concentrated in the range 2005-2021.
- 50 mins - 150 mins is the range of movie durations, excluding potential outliers.
- various ratings of content is avaialble on netfilx, for the various viewers categories like kids, adults , families. Highest -number of movies and TV shows are rated TV-MA (for mature audiences).
- Content in most of the ratings is available in lesser quanitity except in US. Ratings like TV-Y7 , TV-Y7 FV , PG ,TV-G , G , TV-Y , TV-PG are very less avaialble in all countries except US.
- International Movies and TV Shows , Dramas , and Comedies are the top 3 genres on Netflix for both Movies and TV shows.
- Mostly country specific popular genres are observed in each country. Only United States have a good mix of almost all genres. Eg. Korean TV shows (Korea), British TV Shows (UK), Anime features and Anime series (Japan) and so on.
- Indian Actors have been acted in maximum movies on netflix. Top 5 actors are in India based on quantity of movies.

Business Insights

- Netflix have majority of content which is released after the year 2000. It is observed that the content older than year 2000 is very scarce on Netflix. Senior Citizen could be the target audience for such content, which is almost missing currently.
- Maximum content (more than 80%) is
- TV-MA - Content intended for mature audiences aged 17 and above.
- TV-14 - Content suitable for viewers aged 14 and above.
- TV-PG - Parental guidance suggested (similar ratings - PG-13 , PG)
- R - Restricted Content, that may not be suitable for viewers under age 17.

- **These ratings' movies target Matured and Adult audience. Rest 20 % of the content is for kids aged below 13. It shows that Netflix is currently serving mostly Mature audiences or Children with parental guidance.**
- **Most popular genres on Netflix are International Movies and TV Shows , Dramas , Comedies, Action & Adventure, Children & Family Movies, Thrillers.**
- **Maximum content of Netflix which is around 75% , is coming from the top 10 countries. Rest of the world only contributes 25% of the content. More countries can be focussed in future to grow the business.**
- **drop in content is seen across all the countries and type of content in year 2020 and 2021, possibly because of Pandemic.**

Recommendations

- **Very limited genres are focussed in most of the countries except US. It seems the current available genres suits best for US and few countries but maximum countries need some more genres which are highly popular in the region. eg. Indian Mythological content is highly popular. We can create such more country specific genres and It might also be liked acorss the world just like Japanese Anime.**
- **Country specific insights - The content need to be targetting the demographic of any country. Netflix can produce higher number of content in the perticular rating as per demographic of the country. Eg.**
- **The country like India , which is highly populous , has maximum content available only in three rating TV-MA, TV-14 , TV-PG. It is unlikely to serve below 14 age and above 35 year age group .**

In [ ]: