Netflix Data Analysis and Visualization

About NETFLIX

Netflix is one of the most popular media and video streaming platforms. They have over 10000 movies or tv shows available on their platform, as of mid-2021, they have over 222M Subscribers globally. This tabular dataset consists of listings of all the movies and tv shows available on Netflix, along with details such as - cast, directors, ratings, release year, duration, etc.

Business Problem

Analyzing the data and generate insights that could help Netflix in deciding which type of shows/movies to produce and how they can grow the business in different countries

The provided dataset consists of a list of all the TV shows/movies available on Netflix:

- **Show_id**: Unique ID for every Movie / TV Show
- **Type**: Identifier - A Movie or TV Show
- **Title**: Title of the Movie / TV Show
- **Director**: Director of the Movie
- **Cast**: Actors involved in the movie/show
- **Country**: Country where the movie/show was produced
- **Date_added**: Date it was added on Netflix
- **Release_year**: Actual Release year of the movie/show
- **Rating**: TV Rating of the movie/show
- **Duration**: Total Duration - in minutes or number of seasons
- **Listed_in**: Genre
- **Description**: The summary description

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

In [2]:
```
original_data = pd.read_csv("Netflix_data.csv")
original_data.head()
```

Out[2]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	dur
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	9
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	Sea
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	Se
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	Se
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	Sea

In [3]:
```
data= original_data.copy(deep=True)
```

In [4]:
```
data.shape
```

Out[4]:
```
(8807, 12)
```

In [5]:
```
data.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]:
```
data.isnull().sum()
```

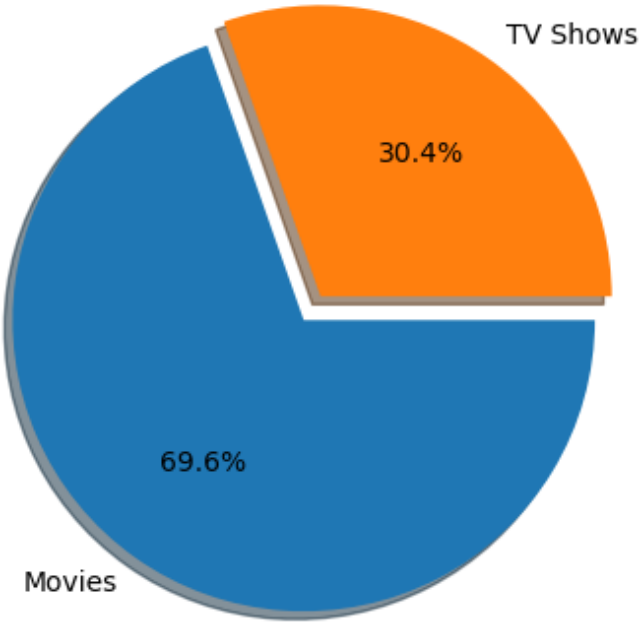
Out[6]:
```
show_id            0
type               0
title              0
director        2634
cast             825
country          831
date_added        10
release_year       0
rating             4
duration           3
listed_in          0
description        0
dtype: int64
```

In [7]:
```
data['type'].value_counts()
```

Out[7]:
```
type
Movie      6131
TV Show    2676
Name: count, dtype: int64
```

Let's see distribution among the Movies and TV Shows in the data

In [8]:
```
plt.pie(data['type'].value_counts(), labels = ['Movies', "TV Shows"],
        counterclock=False, shadow= True, explode = (0.1,0),autopct='%1.1f%%')
plt.show()
```



From the above chart It is Obvious that Netflix has more Movies than TV Shows

How has the number of movies released per year changed over the last 20-30 years?

In [9]:
```
movies_data = data[data['type']=='Movie']
```

In [10]:
```
movies_data.head()
```

Out[10]:

	show_id	type	title	director	cast	country	date_added	release_year	ratir
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	P(
6	s7	Movie	My Little Pony: A New Generation	Robert Cullen, José Luis Ucha	Vanessa Hudgens, Kimiko Glenn, James Marsden, ...	NaN	September 24, 2021	2021	P
7	s8	Movie	Sankofa	Haile Gerima	Kofi Ghanaba, Oyafunmike Ogunlano, Alexandra D...	United States, Ghana, Burkina Faso, United Kin...	September 24, 2021	1993	T M
9	s10	Movie	The Starling	Theodore Melfi	Melissa McCarthy, Chris O'Dowd, Kevin Kline, T...	United States	September 24, 2021	2021	P(
12	s13	Movie	Je Suis Karl	Christian Schwochow	Luna Wedler, Jannis Niewöhner, Milan Peschel, ...	Germany, Czech Republic	September 23, 2021	2021	T M

In [11]:
```
# Let's sort the movies data by year then we will bar plot

sorted_movies_by_year = movies_data.sort_values(by="release_year", ascending =
```

In [12]:
```
sorted_movies_by_year
```

Out[12]:

	show_id	type	title	director	cast	country	date_added	release_year	rat
570	s571	Movie	Dynasty Warriors	Chow Hin Yeung Roy	Wang Kai, Louis Koo, Han Geng, Tony Yang, Cari...	NaN	July 1, 2021	2021	TV
770	s771	Movie	Myriam Fares: The Journey	Myriam Fares	Myriam Fares	United Arab Emirates	June 3, 2021	2021	TV
766	s767	Movie	Alan Saldaña: Locked Up	Alex Díaz	Alan Saldaña	Mexico	June 3, 2021	2021	
765	s766	Movie	Xtreme	Daniel Benmayor	Teo García, Óscar Jaenada, Óscar Casas, Andrea...	Spain	June 4, 2021	2021	
764	s765	Movie	Trippin' with the Kandasamys	Jayan Moodley	Jailoshini Naidoo, Maeshni Naicker, Madhushan ...	South Africa	June 4, 2021	2021	TV
...	...	...	...	...	...	...	...	...	
8660	s8661	Movie	Undercover: How to Operate Behind Enemy Lines	John Ford	NaN	United States	March 31, 2017	1943	
8739	s8740	Movie	Why We Fight: The Battle of Russia	Frank Capra, Anatole Litvak	NaN	United States	March 31, 2017	1943	
8763	s8764	Movie	WWII: Report from the Aleutians	John Huston	NaN	United States	March 31, 2017	1943	
8205	s8206	Movie	The Battle of Midway	John Ford	Henry Fonda, Jane Darwell	United States	March 31, 2017	1942	TV
7790	s7791	Movie	Prelude to War	Frank Capra	NaN	United States	March 31, 2017	1942	TV

6131 rows × 12 columns

In [13]:
```
grouped_movies_by_year = sorted_movies_by_year.groupby(by= "release_year")['tit
grouped_movies_by_year
```

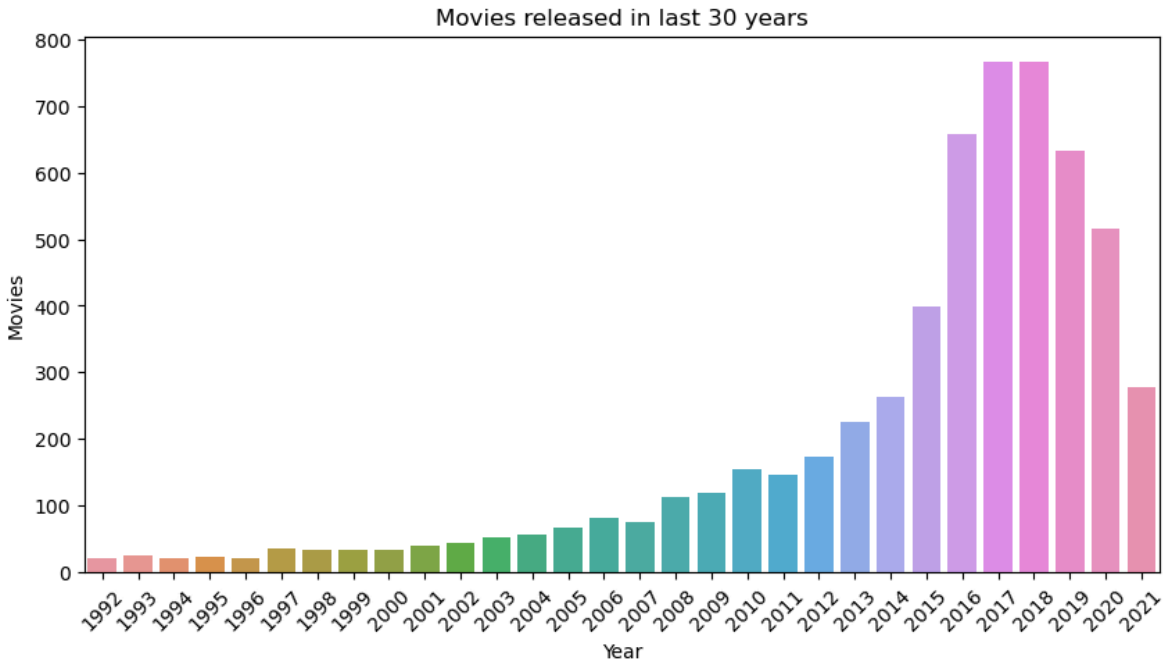
Out[13]:

	release_year	title
0	1942	2
1	1943	3
2	1944	3
3	1945	3
4	1946	1
...	...	...
68	2017	767
69	2018	767
70	2019	633
71	2020	517
72	2021	277

73 rows × 2 columns

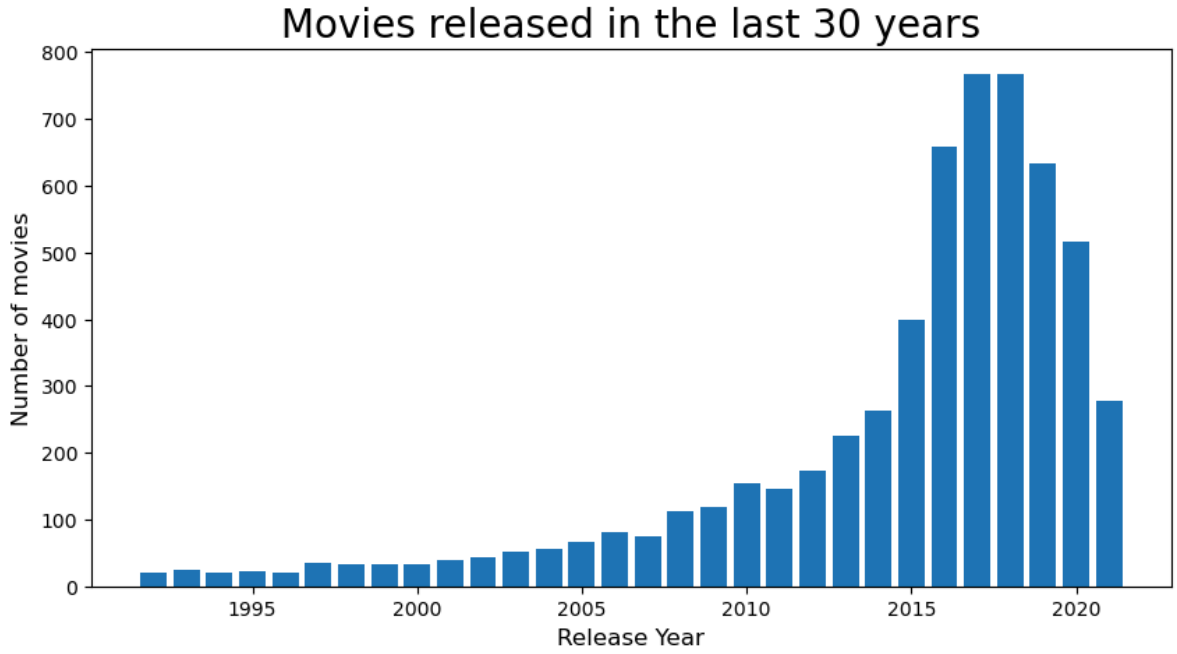
In [14]:
```
plt.figure(figsize= (10,5))
barplot = sns.barplot(grouped_movies_by_year.tail(30), x= "release_year", y= 't
barplot.set(xlabel= "Year", ylabel="Movies", title= 'Movies released in last 30
plt.xticks(rotation=45)

plt.show()
```



In [15]:
```
movies_last_30_year_released = grouped_movies_by_year.tail(30)
```

In [16]:
```
plt.figure(figsize= (10,5))
plt.bar(movies_last_30_year_released['release_year'],movies_last_30_year_releas
plt.xlabel('Release Year',fontsize=12)
plt.ylabel('Number of movies',fontsize=12)
plt.title('Movies released in the last 30 years',fontsize=20)
plt.show()
```



Content Growth: The number of movies released on Netflix over the last 30 years has shown significant growth, particularly in the last decade (2010-2020). The sharpest rise was seen between 2015 and 2020, coinciding with Netflix's global expansion and shift toward original content production

Actionable Item: Netflix should continue to expand its production in movies, which have seen a surge in demand over the last decade. By focusing on shorter, high-quality content in these genres, Netflix can cater to current viewer habits and attract more time-constrained users.

Convert the type of date_added to datetime for more usefull datetime analysis

In [17]:
```
data['date_added'] = pd.to_datetime(data['date_added'], format= "mixed")
```

In [18]:
```
data.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   datetime64[ns]
 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: datetime64[ns](1), int64(1), object(10)
memory usage: 825.8+ KB
```

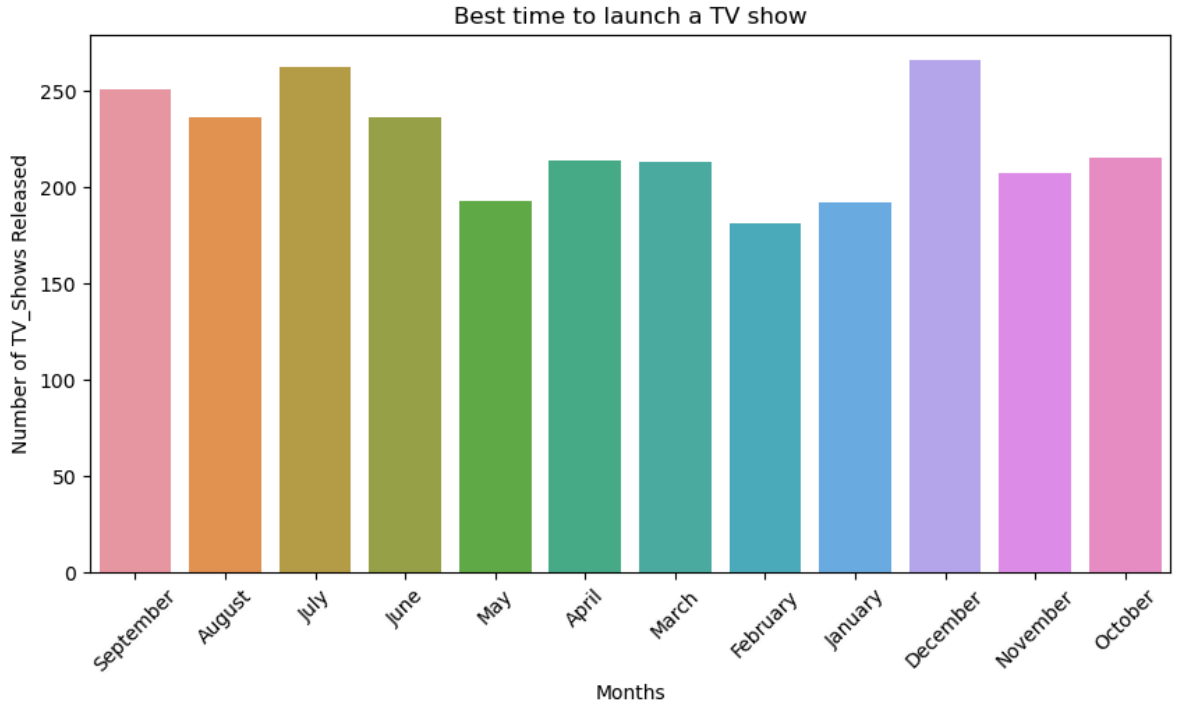
What is the best time to launch a TV show?

In [19]:
```
TVshows_data = data[data['type']=="TV Show"]
```

In [20]:
```
data.groupby(by=['date_added'])['date_added'].count()
```

Out[20]:
```
date_added
2008-01-01     1
2008-02-04     1
2009-05-05     1
2009-11-18     1
2010-11-01     1
              ..
2021-09-21     5
2021-09-22     9
2021-09-23     2
2021-09-24    10
2021-09-25     1
Name: date_added, Length: 1714, dtype: int64
```

In [21]:
```
plt.figure(figsize= (10,5))
sns.countplot(data=TVshows_data,x= TVshows_data['date_added'].dt.month_name())
plt.xticks(rotation= 45)
plt.title("Best time to launch a TV show")
plt.xlabel("Months")
plt.ylabel("Number of TV_Shows Released")
plt.show()
```



Insights: Seasonal Patterns in Viewership: From the above Plot Analysis, TV show releases over the past decade indicates that the best time to launch a TV show is during fall (July and September) and winter (December) which is holiday month. These periods consistently show higher viewership

Actionable Item: Netflix should aim to launch new TV shows in the June-September and especially in December for maximum engagement to capitalize on holiday viewing habits..

In [22]:
```
TVshows_data.isnull().sum()
```

Out[22]:
```
show_id          0
type             0
title            0
director      2446
cast           350
country        391
date_added      10
release_year     0
rating           2
duration         0
listed_in        0
description      0
dtype: int64
```

Comparison Comparison of TV shows vs. Movies.

Before that, Let's Deal with null values first

In [23]:
```
data.isnull().sum()
```

Out[23]:
```
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
```

lets drop the rows that has min null values i.e. <= 10

In [24]:
```
data.dropna(subset=['date_added', "rating", "duration"],inplace= True)
data.isnull().sum()
```

Out[24]:
```
show_id            0
type               0
title              0
director        2621
cast             825
country          829
date_added         0
release_year       0
rating             0
duration           0
listed_in          0
description        0
dtype: int64
```

There are few columns which has multiple names such as director column, cast column so in order find how many column let's go through all

In [25]:
```
data['director'].str.contains(',').count()
```

Out[25]:
```
6169
```

In [26]:
```
cols_to_check = data.columns
data.columns
```

Out[26]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_adde
d',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

In [27]:
```
for col in data.columns:
    if data[col].dtype =='object':
        multiple_names = data[col].str.contains(',').sum()
        print(f"'{col}' contains {multiple_names} rows that contains multiple n
    else:
        print(f"'{col}' is not object type\n")
```

```
'show_id' contains 0 rows that contains multiple names

'type' contains 0 rows that contains multiple names

'title' contains 138 rows that contains multiple names

'director' contains 614 rows that contains multiple names

'cast' contains 7089 rows that contains multiple names

'country' contains 1320 rows that contains multiple names

'date_added' is not object type

'release_year' is not object type

'rating' contains 0 rows that contains multiple names

'duration' contains 0 rows that contains multiple names

'listed_in' contains 6778 rows that contains multiple names

'description' contains 6433 rows that contains multiple names
```

As there are multiple cols that contains multiple values separated by comma, Let's focus more on cols that contains multiple directors, cast, countries, listed_in(genres) for better insights extraction.

In [28]:
```
data['director'] = data['director'].str.split(',')
data['cast'] = data['cast'].str.split(',')
data['country'] = data['country'].str.split(',')
data['listed_in'] = data['listed_in'].str.split(',')
```

In [29]:
```
data.head()
```

Out[29]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	du
0	s1	Movie	Dick Johnson Is Dead	[Kirsten Johnson]	NaN	[United States]	2021-09-25	2020	PG-13	9
1	s2	TV Show	Blood & Water	NaN	[Ama Qamata, Khosi Ngema, Gail Mabalane, Th...	[South Africa]	2021-09-24	2021	TV-MA	Se
2	s3	TV Show	Ganglands	[Julien Leclercq]	[Sami Bouajila, Tracy Gotoas, Samuel Jouy, ...	NaN	2021-09-24	2021	TV-MA	S
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	2021-09-24	2021	TV-MA	S
4	s5	TV Show	Kota Factory	NaN	[Mayur More, Jitendra Kumar, Ranjan Raj, Al...	[India]	2021-09-24	2021	TV-MA	Se

In [30]:
```
data_exploded = data.explode("director", ignore_index=True)
```

In [31]:
```
data_exploded = data_exploded.explode('cast', ignore_index= True)
```

In [32]:
```
data_exploded = data_exploded.explode('country', ignore_index= True)
```

In [33]:
```
data_exploded = data_exploded.explode('listed_in', ignore_index= True)
```

In [34]:
```
data_exploded.head()
```

Out[34]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	2020	PG-13	90 min
1	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	2021-09-24	2021	TV-MA	2 Seasons
2	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	2021-09-24	2021	TV-MA	2 Seasons
3	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	2021-09-24	2021	TV-MA	2 Seasons
4	s2	TV Show	Blood & Water	NaN	Khosi Ngema	South Africa	2021-09-24	2021	TV-MA	2 Seasons

In [35]:
```
data_exploded.shape
```

Out[35]:
```
(201837, 12)
```

Data Cleaning

Dealing with NaN values for the better analysis, Let's check how many Null values are present after doing the explode

In [36]:
```
data_exploded.isnull().sum()
```

Out[36]:
```
show_id            0
type               0
title              0
director       50425
cast            2149
country        11894
date_added         0
release_year       0
rating             0
duration           0
listed_in          0
description        0
dtype: int64
```

There could be possibilty where more than two columns such as director, cast, country are all missing or have null values at the same time. Those rows can be dropped since we don't have much use of it.

In [37]:
```
data_exploded[['director','cast','country','rating','duration','date_added']].l
    'director','cast','country','rating','duration','date_added']]).sum(axis=1)
```

Out[37]:

	director	cast	country	rating	duration	date_added
85	NaN	NaN	NaN	TV-MA	1 Season	2021-09-24
86	NaN	NaN	NaN	TV-MA	1 Season	2021-09-24
353	NaN	NaN	NaN	TV-MA	1 Season	2021-09-24
354	NaN	NaN	NaN	TV-MA	1 Season	2021-09-24
355	NaN	NaN	NaN	TV-MA	1 Season	2021-09-24
...	...	...	...	...	...	...
197166	NaN	NaN	NaN	TV-MA	1 Season	2017-12-27
197167	NaN	NaN	NaN	TV-MA	1 Season	2017-12-27
201778	NaN	NaN	NaN	TV-Y7	2 Seasons	2019-07-01
201779	NaN	NaN	NaN	TV-Y7	2 Seasons	2019-07-01
201780	NaN	NaN	NaN	TV-Y7	2 Seasons	2019-07-01

184 rows × 6 columns

In [38]:
```
rows_to_be_dropped = data_exploded[['director', 'cast', 'rating', 'duration', '
    pd.isnull(data_exploded[['director', 'cast', 'rating', 'duration', 'date_ad
data_exploded.drop(rows_to_be_dropped.index, inplace=True)
```

In [39]:
```
unique_directors = data_exploded.groupby(by=['director'])['director'].value_cou
```

In [40]:
```
unique_directors.sort_values('count', ascending=False)
```

Out[40]:

	director	count
3304	Martin Scorsese	419
5072	Youssef Chahine	409
1381	Cathy Garcia-Molina	356
4624	Steven Spielberg	355
3002	Lars von Trier	336
...	...	...
2389	James Moll	1
4153	Rob Zombie	1
4154	Robb Dipple	1
659	Todd Wider	1
877	Alex Stapleton	1

5118 rows × 2 columns

In [41]:
```
data_exploded['director']=data_exploded.groupby(['type'])['director'].transform
    lambda x: x.fillna(x.mode().iloc[0] if not x.mode().empty else np.nan))
```

The above code fills NaN values in the director column with most frequent/mode of combination of type.

Let's do it for cast by type and director and for the country too with director and cast respectively

In [42]:
```
data_exploded['cast']=data_exploded.groupby(['type','director'])['cast'].transf
    lambda x: x.fillna(x.mode().iloc[0] if not x.mode().empty else original_dat
```

In [43]:
```
data_exploded['country']=data_exploded.groupby(['cast','director'])['country'].
    lambda x: x.fillna(x.mode().iloc[0] if not x.mode().empty else original_dat

# mode of original_data['cast'] is David Attenborough

# mode of original_data['country'] is United States
```

In [44]:
```
data_exploded.isnull().sum()
```

Out[44]:
```
show_id         0
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64
```

In [45]:
```
data_exploded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201837 entries, 0 to 201836
Data columns (total 12 columns):
 #   Column        Non-Null Count   Dtype         
---  ------        --------------   -----         
 0   show_id       201837 non-null  object        
 1   type          201837 non-null  object        
 2   title         201837 non-null  object        
 3   director      201837 non-null  object        
 4   cast          201837 non-null  object        
 5   country       201837 non-null  object        
 6   date_added    201837 non-null  datetime64[ns]
 7   release_year  201837 non-null  int64         
 8   rating        201837 non-null  object        
 9   duration      201837 non-null  object        
 10  listed_in     201837 non-null  object        
 11  description   201837 non-null  object        
dtypes: datetime64[ns](1), int64(1), object(10)
memory usage: 18.5+ MB
```

Now that data has cleaned, let's gain more insights from the Cleaned Data

Analysis of actors/directors of different types of shows/movies.

In [46]:
```
#let's seperate data for movies and TV shows for category specific analysis
TV_show_data = data_exploded[data_exploded['type']=="TV Show"]
movies_data = data_exploded[data_exploded['type']=='Movie']
movies_data.head()
```

Out[46]:

	show_id	type	title	director	cast	country	date_added	release_year	rating
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	David Attenborough	United States	2021-09-25	2020	PG-13
159	s7	Movie	My Little Pony: A New Generation	Robert Cullen	Vanessa Hudgens	United States	2021-09-24	2021	PG
160	s7	Movie	My Little Pony: A New Generation	Robert Cullen	Kimiko Glenn	United States	2021-09-24	2021	PG
161	s7	Movie	My Little Pony: A New Generation	Robert Cullen	James Marsden	United States	2021-09-24	2021	PG
162	s7	Movie	My Little Pony: A New Generation	Robert Cullen	Sofia Carson	United States	2021-09-24	2021	PG

In [47]:
```
movies_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 145905 entries, 0 to 201836
Data columns (total 12 columns):
 #   Column        Non-Null Count   Dtype         
---  ------        --------------   -----         
 0   show_id       145905 non-null  object        
 1   type          145905 non-null  object        
 2   title         145905 non-null  object        
 3   director      145905 non-null  object        
 4   cast          145905 non-null  object        
 5   country       145905 non-null  object        
 6   date_added    145905 non-null  datetime64[ns]
 7   release_year  145905 non-null  int64         
 8   rating        145905 non-null  object        
 9   duration      145905 non-null  object        
 10  listed_in     145905 non-null  object        
 11  description   145905 non-null  object        
dtypes: datetime64[ns](1), int64(1), object(10)
memory usage: 14.5+ MB
```

In [48]:
```
TV_show_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 55932 entries, 1 to 201780
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   show_id       55932 non-null  object        
 1   type          55932 non-null  object        
 2   title         55932 non-null  object        
 3   director      55932 non-null  object        
 4   cast          55932 non-null  object        
 5   country       55932 non-null  object        
 6   date_added    55932 non-null  datetime64[ns]
 7   release_year  55932 non-null  int64         
 8   rating        55932 non-null  object        
 9   duration      55932 non-null  object        
 10  listed_in     55932 non-null  object        
 11  description   55932 non-null  object        
dtypes: datetime64[ns](1), int64(1), object(10)
memory usage: 5.5+ MB
```

I want duration to be in int values and remove the "min" string and get only number, let's do that..

In [49]:
```
def converting_into_min(mins):
    lis = mins.split(" ")
    return int(lis[0])
```

In [50]:
```
movies_data['duration'] = movies_data['duration'].apply(converting_into_min)
```

```
C:\Users\302sy\AppData\Local\Temp\ipykernel_784\3930214305.py:1: SettingWithCo
pyWarning:
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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.p
ydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy)
  movies_data['duration'] = movies_data['duration'].apply(converting_into_min)
```

In [51]:
```
movies_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 145905 entries, 0 to 201836
Data columns (total 12 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   show_id       145905 non-null  object
 1   type          145905 non-null  object
 2   title         145905 non-null  object
 3   director      145905 non-null  object
 4   cast          145905 non-null  object
 5   country       145905 non-null  object
 6   date_added    145905 non-null  datetime64[ns]
 7   release_year  145905 non-null  int64
 8   rating        145905 non-null  object
 9   duration      145905 non-null  int64
 10  listed_in     145905 non-null  object
 11  description   145905 non-null  object
dtypes: datetime64[ns](1), int64(2), object(9)
memory usage: 14.5+ MB
```

Let's check the top directors and actors in Movies vs. TV Shows.

In [52]:
```
movies_data.head()
```

Out[52]:

	show_id	type	title	director	cast	country	date_added	release_year	rating
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	David Attenborough	United States	2021-09-25	2020	PG-13
159	s7	Movie	My Little Pony: A New Generation	Robert Cullen	Vanessa Hudgens	United States	2021-09-24	2021	PG
160	s7	Movie	My Little Pony: A New Generation	Robert Cullen	Kimiko Glenn	United States	2021-09-24	2021	PG
161	s7	Movie	My Little Pony: A New Generation	Robert Cullen	James Marsden	United States	2021-09-24	2021	PG
162	s7	Movie	My Little Pony: A New Generation	Robert Cullen	Sofia Carson	United States	2021-09-24	2021	PG

In [53]:
```
movies_data['director'].value_counts().head(3)
```

Out[53]:
```
director
Martin Scorsese        1702
Youssef Chahine         409
Cathy Garcia-Molina     356
Name: count, dtype: int64
```

In [54]:
```
movies_data['cast'].value_counts().head(3)
```

Out[54]:
```
cast
David Attenborough    1093
 Alfred Molina         157
 Fortune Feimster      131
Name: count, dtype: int64
```

In [55]:
```
TV_show_data['director'].value_counts().head(3)
```

Out[55]:
```
director
Noam Murro         49331
Thomas Astruc        160
Damien Chazelle      104
Name: count, dtype: int64
```

In [56]:
```
TV_show_data['cast'].value_counts().head(3)
```

Out[56]:
```
cast
David Attenborough    896
 Takahiro Sakurai      54
 Yuki Kaji             41
Name: count, dtype: int64
```

From the above analysis we can say that, Top director is "Martin Scorsese" and "Noam Murro" with respect to Movies and TV Shows.

And The Top actor is "David Attenborough" in both the types.

We have the Genre and durations columns where we can do analysis of which genre has more duration, for this we can use box plot.

Let's Analyze how different content ratings (e.g., TV-MA, PG-13) are distributed across genres. This will also explore the relationship between ratings and the duration of movies and TV shows, providing insights into content suitability and audience targeting.

In [57]:
```
movies_data['genre'] = movies_data['listed_in'].str.strip()
TV_show_data['genre'] = TV_show_data['listed_in'].str.strip()
```

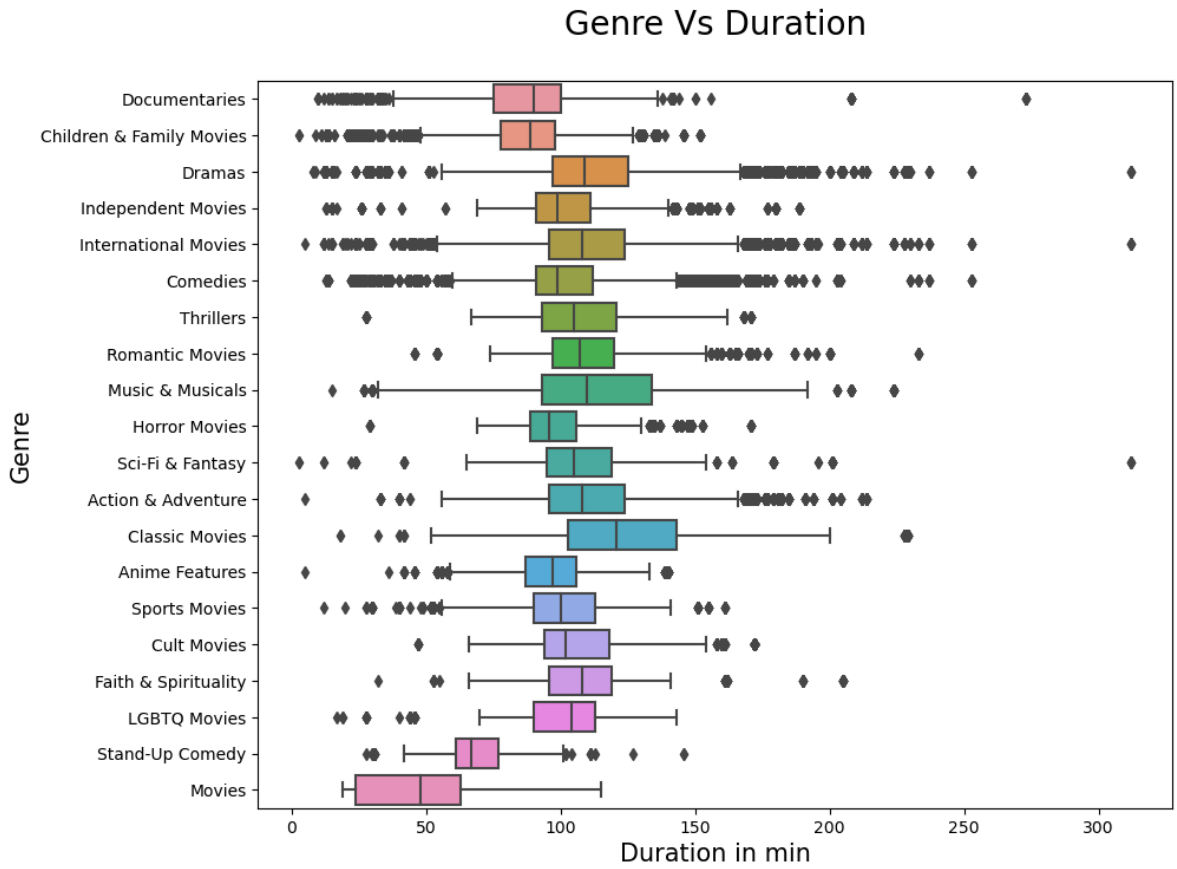
```
C:\Users\302sy\AppData\Local\Temp\ipykernel_784\2639548846.py:1: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.p
ydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy)
  movies_data['genre'] = movies_data['listed_in'].str.strip()
C:\Users\302sy\AppData\Local\Temp\ipykernel_784\2639548846.py:2: SettingWithCo
pyWarning:
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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.p
ydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy)
  TV_show_data['genre'] = TV_show_data['listed_in'].str.strip()
```

In [58]:
```
movies_by_genre = movies_data.groupby(by=['genre'])['title'].count()
```

In [59]:
```
# movies_data['duration']
plt.figure(figsize=(10,8))
sns.boxplot(data = movies_data,x = 'duration', y='genre')
plt.title("Genre Vs Duration", fontsize=20, y=1.05)
plt.xlabel("Duration in min", fontsize=15)
plt.ylabel("Genre", fontsize=15)
plt.show()
```



Movie Duration: The average movie duration across genres is around 90-120 minutes, but certain genres (e.g., Dramas and International Movies) have a tendency toward longer durations.

Only three Genres has more duration i.e. 300 min Dramas, International Movies, Sci-Fi & Fantasy, Which tells Customers have more attention span for these types of movies.

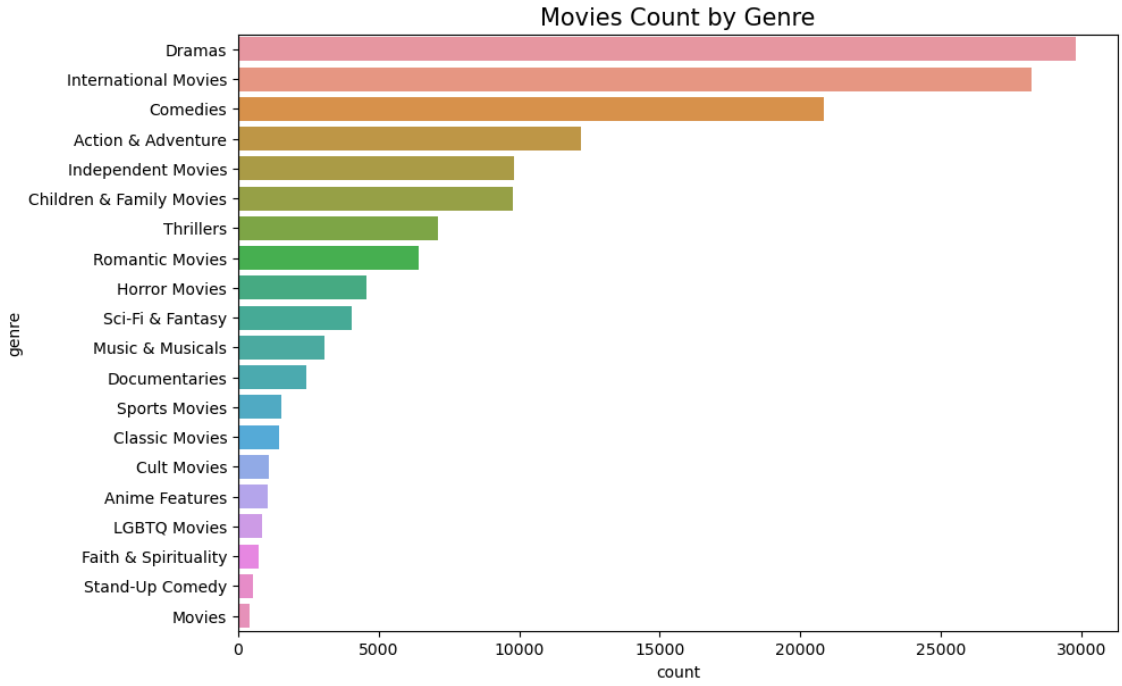
Let's Check the number of Movies and TV shows that are released Genre Wise

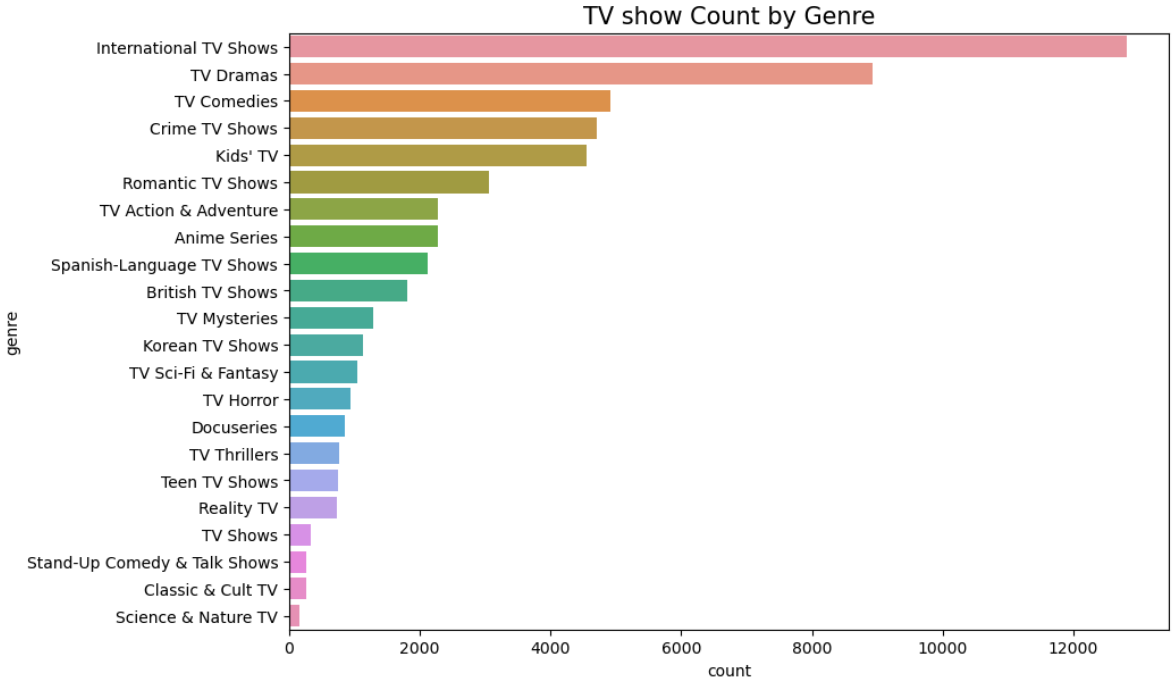
In [60]:
```
movies_list_genre_wise = movies_data['genre'].value_counts().reset_index()
TVShow_list_genre_wise = TV_show_data['genre'].value_counts().reset_index()
```

In [61]:
```
plt.figure(figsize=(10,15))
plt.subplot(2,1,1)
plt.title("Movies Count by Genre", fontsize = 15)
sns.barplot(data=movies_list_genre_wise,y='genre', x='count')


# plt.figure(figsize=(10,5))
plt.subplot(2,1,2)
plt.title("TV show Count by Genre", fontsize = 15)

sns.barplot(data=TVShow_list_genre_wise,y='genre', x='count')
plt.show()
```

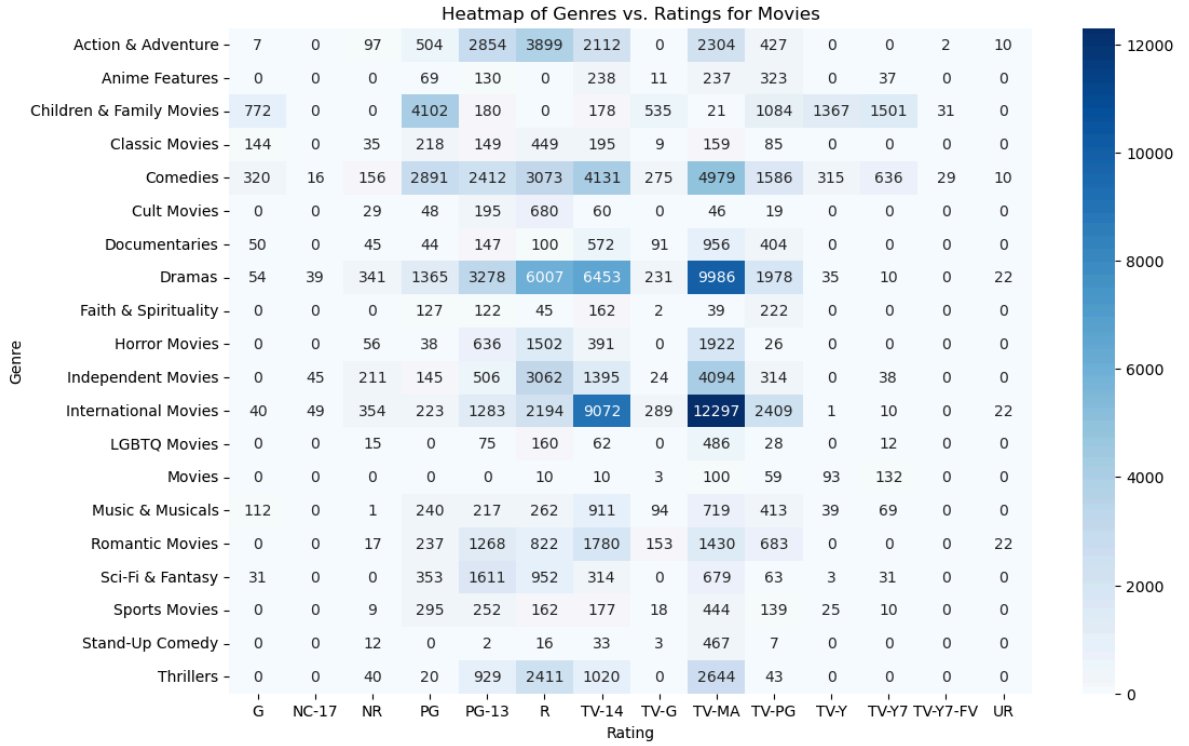




Type *Markdown* and LaTeX: α^2

In [62]:
```
genre_rating_crosstab = pd.crosstab(movies_data['genre'], movies_data['rating']
```

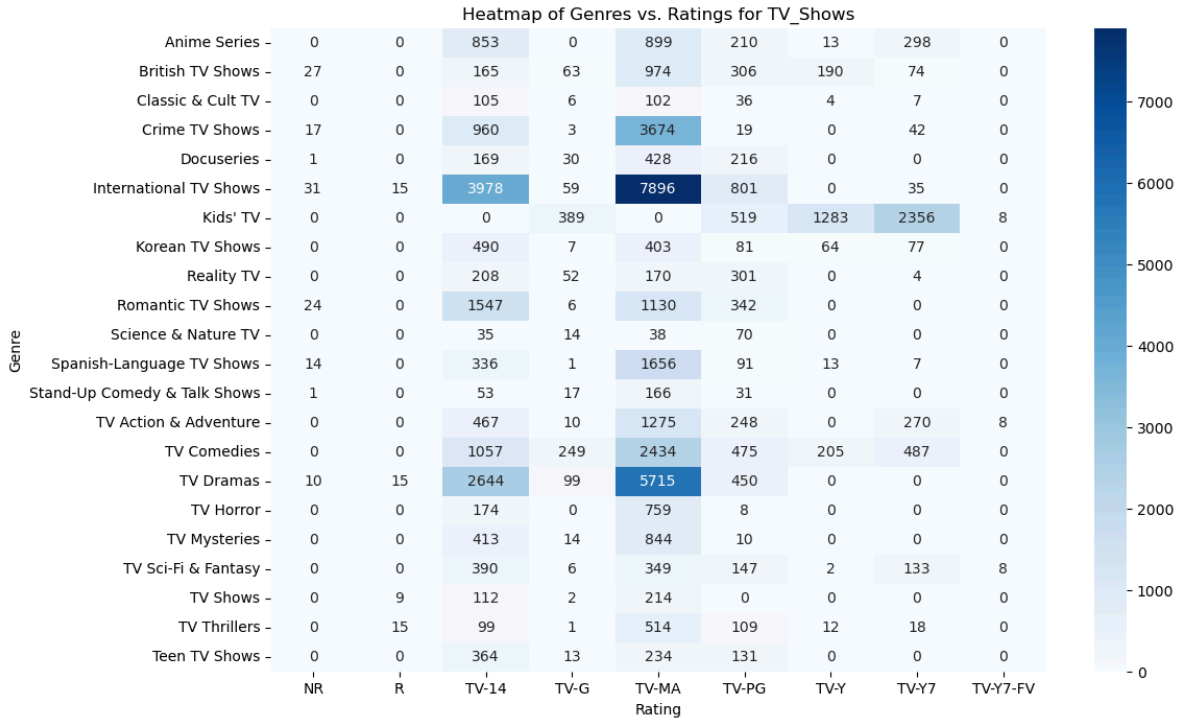
In [63]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(genre_rating_crosstab, cmap='Blues', annot=True, fmt='d')
plt.title('Heatmap of Genres vs. Ratings for Movies')
plt.xlabel('Rating')
plt.ylabel('Genre')
plt.show()
```



Insights from the HeatMap: The most number of movies are focused on TV-MA("mature Audience only") rated movies in which international movies and dramas are on top. Followed by TV-14 rated movies.

In [64]:
```
genre_rating_crosstab_tv_shows = pd.crosstab(TV_show_data['genre'], TV_show_dat
```

In [65]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(genre_rating_crosstab_tv_shows, cmap='Blues', annot=True, fmt='d')
plt.title('Heatmap of Genres vs. Ratings for TV_Shows')
plt.xlabel('Rating')
plt.ylabel('Genre')
plt.show()
```



Insights from the HeatMap: Similar to movies, TV_shows are also focused on TV-MA("mature Audience only") rated movies in which international TV_shows and TV dramas are on top. Followed by TV-14 rated movies.

Actionable Item: Based on the strong correlation between certain genres and content ratings, Netflix can better target its marketing strategies. For example, promoting TV-MA rated thrillers to mature audiences or focusing on family-friendly content for genres like Animation

In [66]:
```
data_exploded['country'] = data_exploded['country'].str.strip()
```

In [67]:
```
data_exploded['country'].value_counts().head(10)
```

Out[67]:
```
country
United States     69481
India             23264
United Kingdom    12958
Japan              8888
France             8280
Canada             8032
Spain              5351
South Korea        5148
Germany            4386
Mexico             3969
Name: count, dtype: int64
```

Here's the top 10 countries that produces movies in the Netflix, Being United States at the top followed by India, United Kingdown.

We should make long term Contract with the publishers in order to release most number of movies with the Netflix.

These insights and recommendations not only highlight key findings but also provide actionable steps that Netflix can take to optimize content offerings and better serve its audience