In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px

%matplotlib inline
```

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

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

Dataset

Link:
https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940/original/(https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940/origina

The dataset provided to you 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

In [2]:

```
netflix_data = pd.read_csv(r'C:\Users\NABINA\Documents\netflix.csv')
netflix_data
```

Out[2]:

	show_id	type	title	director	cast	country	date_added	release_year	ra
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	
...	...	...	...	...	...	...	...	...	
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	2007	
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	2018	T
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	2009	
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	2006	

	show_id	type	title	director	cast	country	date_added	release_year	ra
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	2015	T

8807 rows × 12 columns

In [3]:

```
1 netflix_data.shape
```

Out[3]:

```
(8807, 12)
```

In [4]:

```
1 netflix_data.columns
```

Out[4]:

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

In [5]:

```
1 ##Statistical Analysis
2
3 netflix_data.describe(include=object)
```

Out[5]:

	show_id	type	title	director	cast	country	date_added	rating	duratio
count	8807	8807	8807	6173	7982	7976	8797	8803	880
unique	8807	2	8807	4528	7692	748	1767	17	22
top	s1	Movie	Dick Johnson Is Dead	Rajiv Chilaka	David Attenborough	United States	January 1, 2020	TV-MA	Seaso
freq	1	6131	1	19	19	2818	109	3207	179

In [6]:

```
1 netflix_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 [7]:

```
1 netflix_data.dtypes
```

Out[7]:

```
show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
rating          object
duration        object
listed_in       object
description     object
dtype: object
```

Performing the analysis on the copied dataframe

In [8]:

```
1 data=netflix_data.copy()
2 
```

OBSERVATIONS

In [9]:

```
1 data.shape
2
```

Out[9]:

```
(8807, 12)
```

In [10]:

```
1 data.columns
```

Out[10]:

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

In [11]:

```
1 data.head()
```

Out[11]:

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

First 5 rows of the data

In [12]:

```
1 data['type'].unique()
```

Out[12]:

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

In [13]:

```
1 #Changing 'UR' to 'NR' in Ratings
2
3 data.loc[data['rating']=='UR',"rating"]= 'NR'
```

Total Unique Value In Each Column

In [14]:

```
1 data.nunique()
```

Out[14]:

```
show_id         8807
type               2
title           8807
director        4528
cast            7692
country          748
date_added      1767
release_year      74
rating            16
duration         220
listed_in        514
description     8775
dtype: int64
```

Checking For Missing Values

In [15]:

```
m=data.isna().sum()
m
```

Out[15]:

```
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
```

The dataframe has 2634 missing values in director, 825 in cast, 831 in country, 10 in date_added, 4 in rating and 3 in duration

In [16]:

```
k=round((data.isna().sum()/data.shape[0])*100,2)
k
```

Out[16]:

```
show_id          0.00
type             0.00
title            0.00
director        29.91
cast             9.37
country          9.44
date_added       0.11
release_year     0.00
rating           0.05
duration         0.03
listed_in        0.00
description      0.00
dtype: float64
```

In [17]:

```
pd.concat([m,k],axis=1,keys=["Total","%"])
```

Out[17]:

	Total	%
show_id	0	0.00
type	0	0.00
title	0	0.00
director	2634	29.91
cast	825	9.37
country	831	9.44
date_added	10	0.11
release_year	0	0.00
rating	4	0.05
duration	3	0.03
listed_in	0	0.00
description	0	0.00

From the above data it is seen that almost 30% records in the director columns have null values . 9.37% of the cast column and 9.44% of country column have null values

In [18]:

```
data.dropna(subset=['rating','duration','date_added'], inplace=True)
data.reset_index()
data.shape
```

Out[18]:

```
(8790, 12)
```

The dataset has 8790 rows after dropping the missing values in columns rating, duration and date_added

Non-Graphical Analysis - Value Counts and Unique attributes

In [19]:

```
1 data.nunique()
```

Out[19]:

```
show_id         8790
type               2
title           8790
director        4526
cast            7678
country          748
date_added      1765
release_year      74
rating            13
duration         220
listed_in        513
description     8758
dtype: int64
```

Unique Values for all the features in the data dataframe , with show_id and title column containing no duplicated values

In [20]:

```
1 value_count_Types = data["type"].value_counts()
2 value_count_Types
```

Out[20]:

```
Movie      6126
TV Show    2664
Name: type, dtype: int64
```

Two type of shows in the dataset Movie and TV Show

In [21]:

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

Out[21]:

```
type     rating
Movie    G             41
         NC-17          3
         NR            78
         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
         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 [22]:

```
1 value_count_ratings=data["rating"].value_counts()
2 value_count_ratings
```

Out[22]:

```
TV-MA       3205
TV-14       2157
TV-PG        861
R            799
PG-13        490
TV-Y7        333
TV-Y         306
PG           287
TV-G         220
NR            82
G             41
TV-Y7-FV       6
NC-17          3
Name: rating, dtype: int64
```

In [31]:

```
a=data["rating"].unique()
a
```

Out[31]:

```
array(['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7', 'R',
       'TV-G', 'G', 'NC-17', 'NR', 'TV-Y7-FV', 'UR'], dtype=object)
```

In [23]:

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

Out[23]:

```
type     rating
Movie    G             41
         NC-17          3
         NR            78
         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
         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 [24]:

```
# Converting 'Rating' and 'Type' to categorical columns

data[['rating','type']] = data[['rating','type']].astype('category')
```

In [25]:

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

In [26]:

```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8790 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8790 non-null   object
 1   type          8790 non-null   category
 2   title         8790 non-null   object
 3   director      6169 non-null   object
 4   cast          7965 non-null   object
 5   country       7961 non-null   object
 6   date_added    8790 non-null   datetime64[ns]
 7   release_year  8790 non-null   int64
 8   rating        8790 non-null   category
 9   duration      8790 non-null   object
 10  listed_in     8790 non-null   object
 11  description   8790 non-null   object
dtypes: category(2), datetime64[ns](1), int64(1), object(8)
memory usage: 773.3+ KB
```

In [27]:

```
1 data.dtypes
```

Out[27]:

```
show_id                 object
type                  category
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                category
duration                object
listed_in               object
description             object
dtype: object
```

After making the changes, we have:

1 datetime column 1 int 2 category columns

In [28]:

```
movie_records = data.loc[data['type']=='Movie']
movie_records['duration'].value_counts()
```

Out[28]:

```
90 min     152
94 min     146
93 min     146
97 min     146
91 min     144
          ...
212 min      1
8 min        1
186 min      1
193 min      1
191 min      1
Name: duration, Length: 205, dtype: int64
```

In [29]:

```
TVShow_records = data.loc[data['type']=='TV Show']
TVShow_records['duration'].value_counts()
```

Out[29]:

```
1 Season      1791
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: duration, dtype: int64
```

Since movie and TV shows both have different format for duration, we can split duration into 2 columns - movie_duration & tv_seasons

In [30]:

```
#Adding separate column for duration of movies

data['movie_duration']=np.where(data['type']=='Movie', data['duration'].str[:-4], Nor
data['movie_duration'] = data['movie_duration'].astype('float')
```

In [31]:

```
#Adding separate column for duration of TV Shows

data['tv_seasons']=np.where(data['type']=='TV Show', data['duration'].str[:-7]
                            .apply(lambda x: x.strip()), None)
data['tv_seasons'] = data['tv_seasons'].astype('float')
```

In [32]:

```
# Extract year from date_added column

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

In [34]:

```
# Extract month from date_added column

data['month_added'] = data['date_added'].dt.month
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8790 entries, 0 to 8806
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   show_id         8790 non-null   object
 1   type            8790 non-null   category
 2   title           8790 non-null   object
 3   director        6169 non-null   object
 4   cast            7965 non-null   object
 5   country         7961 non-null   object
 6   date_added      8790 non-null   datetime64[ns]
 7   release_year    8790 non-null   int64
 8   rating          8790 non-null   category
 9   duration        8790 non-null   object
 10  listed_in       8790 non-null   object
 11  description     8790 non-null   object
 12  movie_duration  6126 non-null   float64
 13  tv_seasons      2664 non-null   float64
 14  year_added      8790 non-null   int64
 15  month_added     8790 non-null   int64
dtypes: category(2), datetime64[ns](1), float64(2), int64(3), object(8)
memory usage: 1.0+ MB
```

After pre-processing, we have:

3 int columns (year_added, month_added and release_year)

2 category columns (type and rating)

1 datetime column (date_added)

2 float column (movie_duration,tv_seasons)

Outlier Check

In [35]:

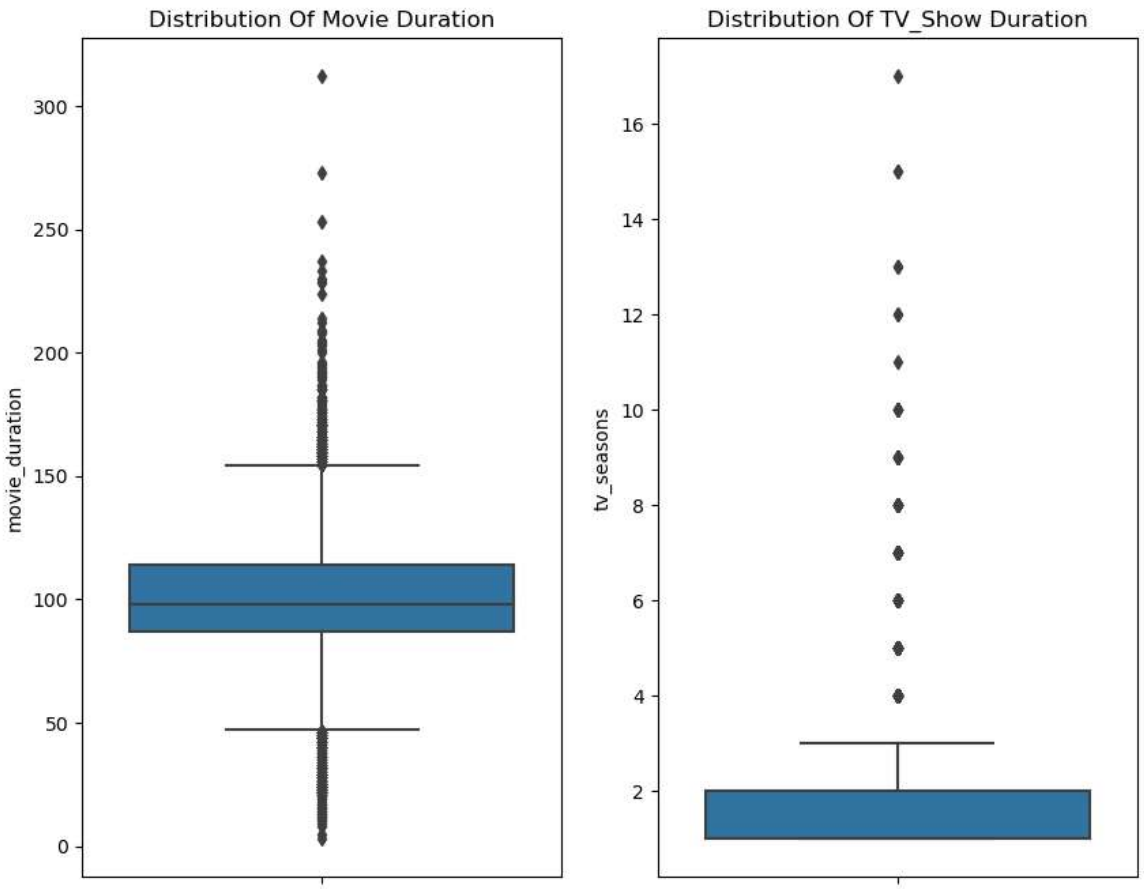
```
1 data.dtypes
```

Out[35]:

```
show_id                  object
type                   category
title                    object
director                 object
cast                     object
country                  object
date_added        datetime64[ns]
release_year              int64
rating                 category
duration                 object
listed_in                object
description              object
movie_duration          float64
tv_seasons              float64
year_added                int64
month_added               int64
dtype: object
```

In [36]:

```
#Outlier check for Duration of Movie/TV Show

plt.figure(figsize=(10,8))
plt.subplot(1,2,1)
sns.boxplot(y=data["movie_duration"])
plt.title("Distribution Of Movie Duration")

plt.subplot(1,2,2)
sns.boxplot(y=data["tv_seasons"])
plt.title("Distribution Of TV_Show Duration")

plt.show()
```

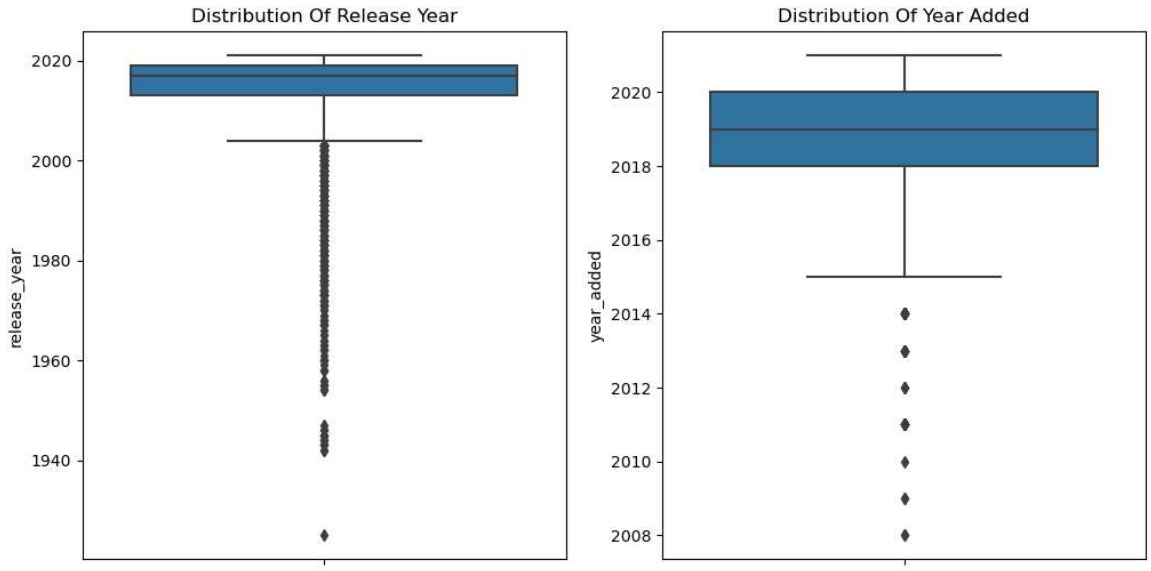


50 mins - 150 mins is the range for movie duration excluding potential outliers (values lying outside the whiskers)

1-3 seasons is the range for TV shows excluding potential outliers

In [37]:

```
#Outlier check for Release_Year and Year_Added


plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
sns.boxplot(y=data["release_year"])
plt.title("Distribution Of Release Year")

plt.subplot(1,2,2)
sns.boxplot(y=data["year_added"])
plt.title("Distribution Of Year Added")

plt.show()
```



The release year for shows is concentrated in the range 2000-2021

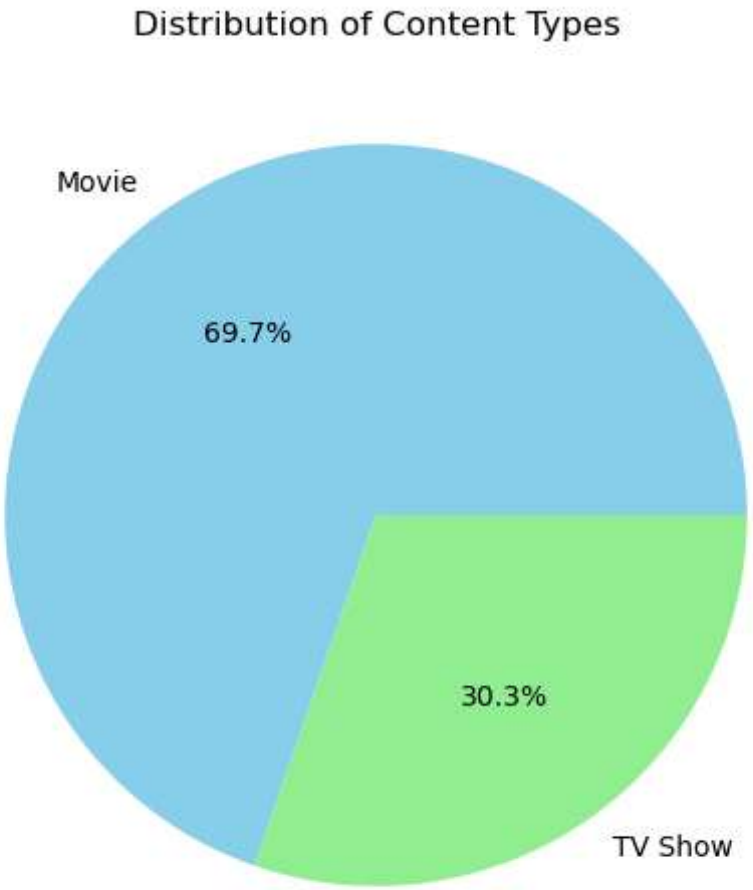
The year of adding shows is concentrated in the range 2015-2021

Visual Analysis: Univariate

TYPE OF SHOW

In [38]:

```
1 plt.figure(figsize=(8, 6))
2 data['type'].value_counts().plot(kind='pie', autopct='%1.1f%%', colors=['skyblue', '
3 plt.title('Distribution of Content Types')
4 plt.ylabel('')
5 plt.show()
```

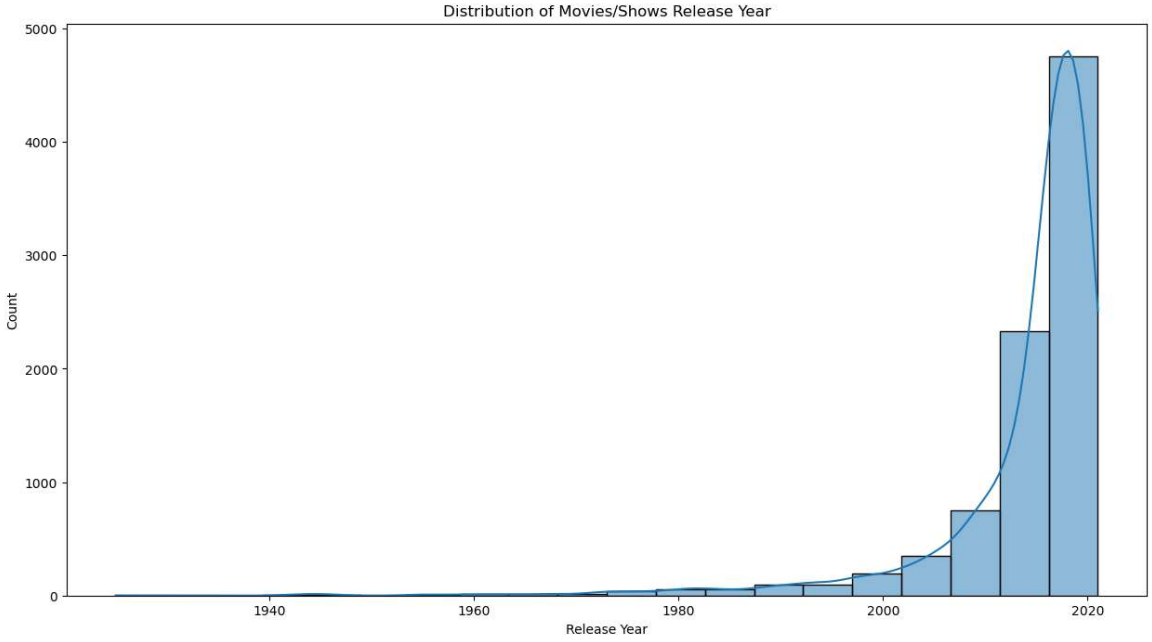


70% shows on Netflix are movies and only 30% are TV shows

RELEASE YEAR

In [39]:

```
plt.figure(figsize=(15,8))
sns.histplot(data["release_year"], bins=20,kde=True)
plt.xlabel("Release Year")
plt.ylabel("Count")
plt.title("Distribution of Movies/Shows Release Year")
plt.show()
```



The left skewed graph shows as the release year increases, the movie count also increases.

In [40]:

```
(data["release_year"].max(),data["release_year"].min())
```

Out[40]:

```
(2021, 1925)
```

In [41]:

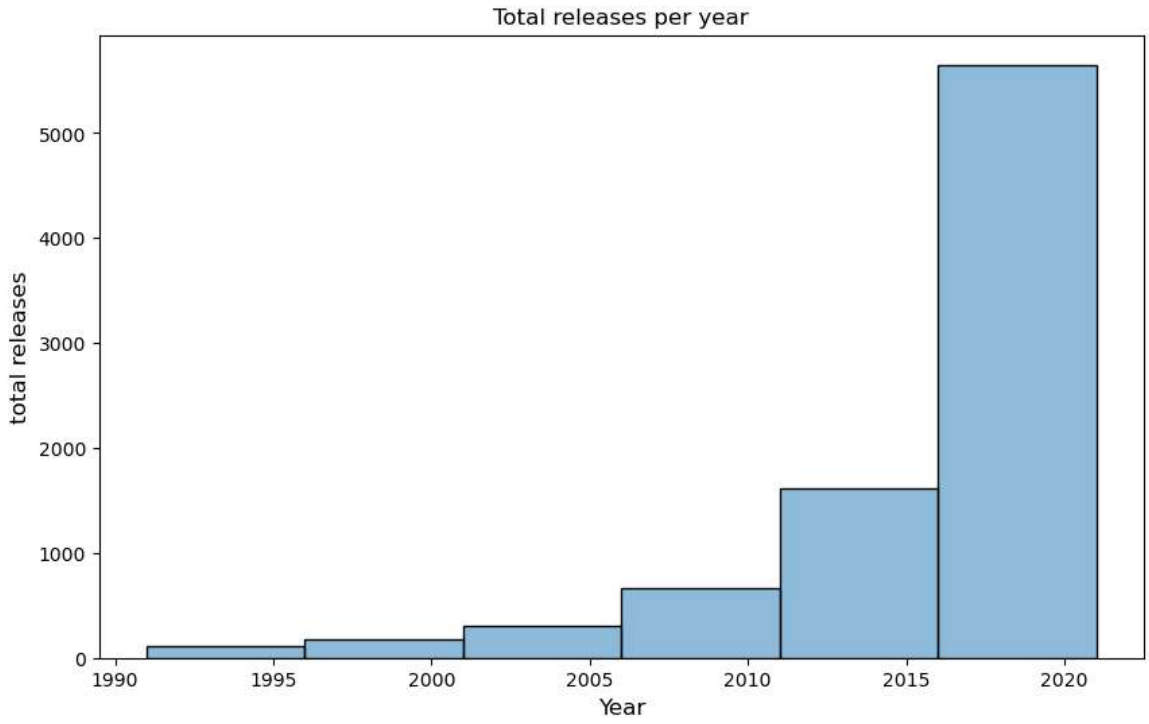
```
k=data[data["release_year"]>1990]
k
k.groupby("release_year").count()["show_id"].sort_index().sort_values(ascending=False
```

Out[41]:

```
release_year
2018    1146
2019    1030
2017    1030
2020     953
2016     901
2021     592
2015     555
2014     352
2013     286
2012     236
2010     192
2011     185
2009     152
2008     135
2006      96
2007      88
2005      80
2004      64
2003      59
2002      51
2001      45
1999      39
1997      38
2000      37
1998      36
1993      28
1995      25
1996      24
1992      23
1994      22
1991      17
Name: show_id, dtype: int64
```

In [42]:

```
plt.figure(figsize = (10,6))
sns.histplot([k['release_year']] , bins = 6 , legend = False)
plt.xlabel('Year' , fontsize = 12)
plt.ylabel('total releases' , fontsize = 12)
plt.title('Total releases per year')
plt.show()
```



The above graph shows how number of movies/shows released per year changed over the last 20-30 years(Directly Proportional)

In [43]:

```
release_year_count = pd.pivot_table(data, values='show_id', index='release_year',
                                    columns='type', aggfunc='count',
                                    dropna=True).reset_index()
release_year_count
```

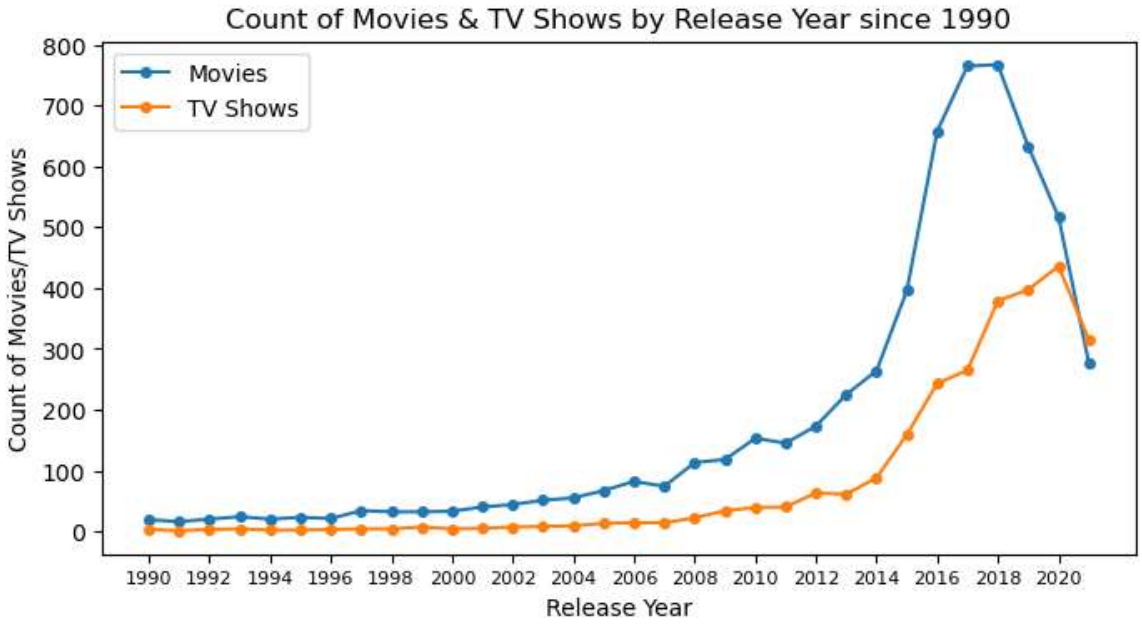
Out[43]:

type	release_year	Movie	TV Show
0	1925	0	1
1	1942	2	0
2	1943	3	0
3	1944	3	0
4	1945	3	1
...	...	...	...
69	2017	765	265
70	2018	767	379
71	2019	633	397
72	2020	517	436
73	2021	277	315

74 rows × 3 columns

In [162]:

```
# Count of movies/TV Shows by release year

plt.figure(figsize=(8,4))
plt.plot(release_year_count.loc[release_year_count['release_year']>=1990, 'release_ye
         release_year_count.loc[release_year_count['release_year']>=1990,'Movie'],mar

plt.plot(release_year_count.loc[release_year_count['release_year']>=1990,'release_yea
         release_year_count.loc[release_year_count['release_year']>=1990,'TV Show'],r

plt.xlabel('Release Year')
plt.ylabel('Count of Movies/TV Shows')

plt.title('Count of Movies & TV Shows by Release Year since 1990')
plt.legend(['Movies','TV Shows'])
plt.xticks(np.arange(1990,2021,2), fontsize=8)

plt.show();
```



The above graph shows how number of movies and shows respectively released per year changed over the last 20-30 years(Directly Proportional)

2018 marks the highest number of movie and TV show releases

The yearly number of releases has surged drastically from 2015.

The period of 2005-2015 shows a gradual increase in the number of releases per year

RATING

In [163]:

```
value_count_ratings=data["rating"].value_counts()
value_count_ratings
```

Out[163]:

```
TV-MA       3205
TV-14       2157
TV-PG        861
R            799
PG-13        490
TV-Y7        333
TV-Y         306
PG           287
TV-G         220
NR            82
G             41
TV-Y7-FV       6
NC-17          3
Name: rating, dtype: int64
```

In [164]:

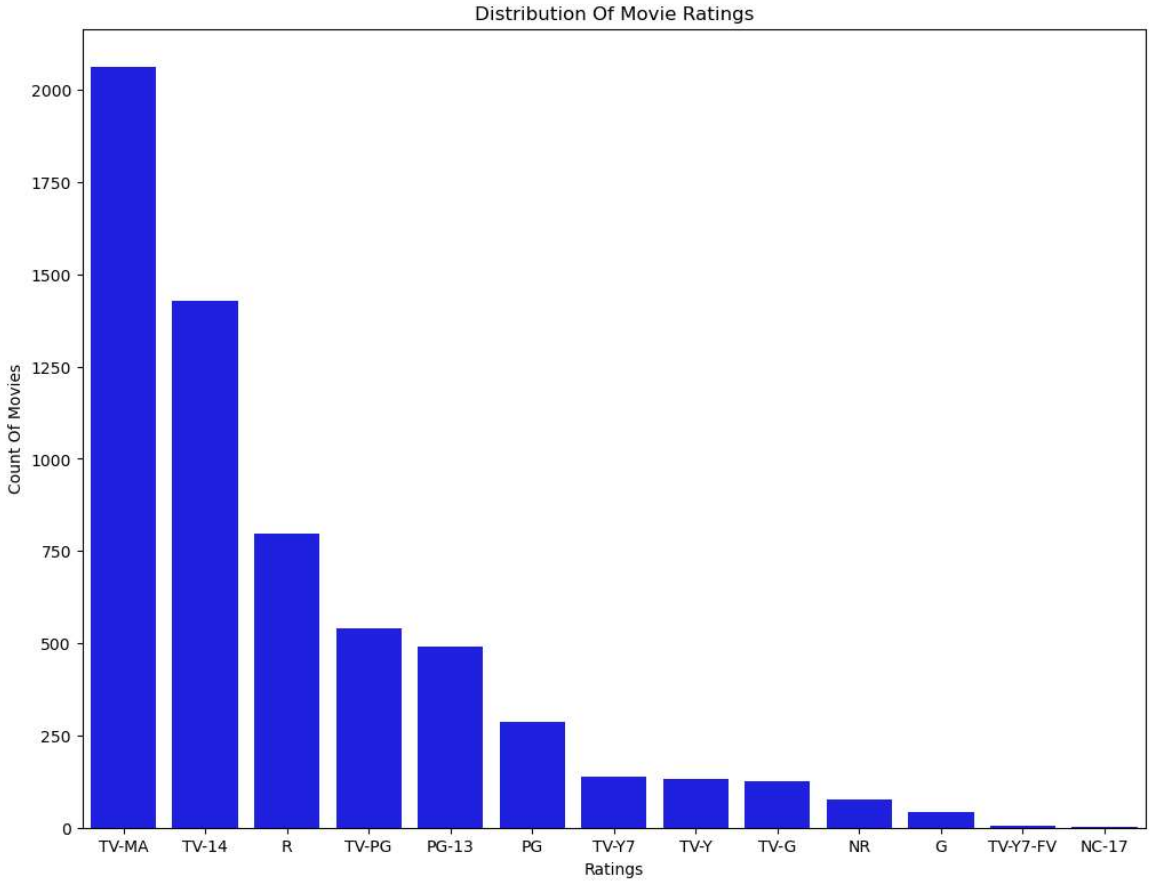
```
movie_data = data.loc[data['type']=='Movie']
tv_data = data.loc[data['type']=='TV Show']
```

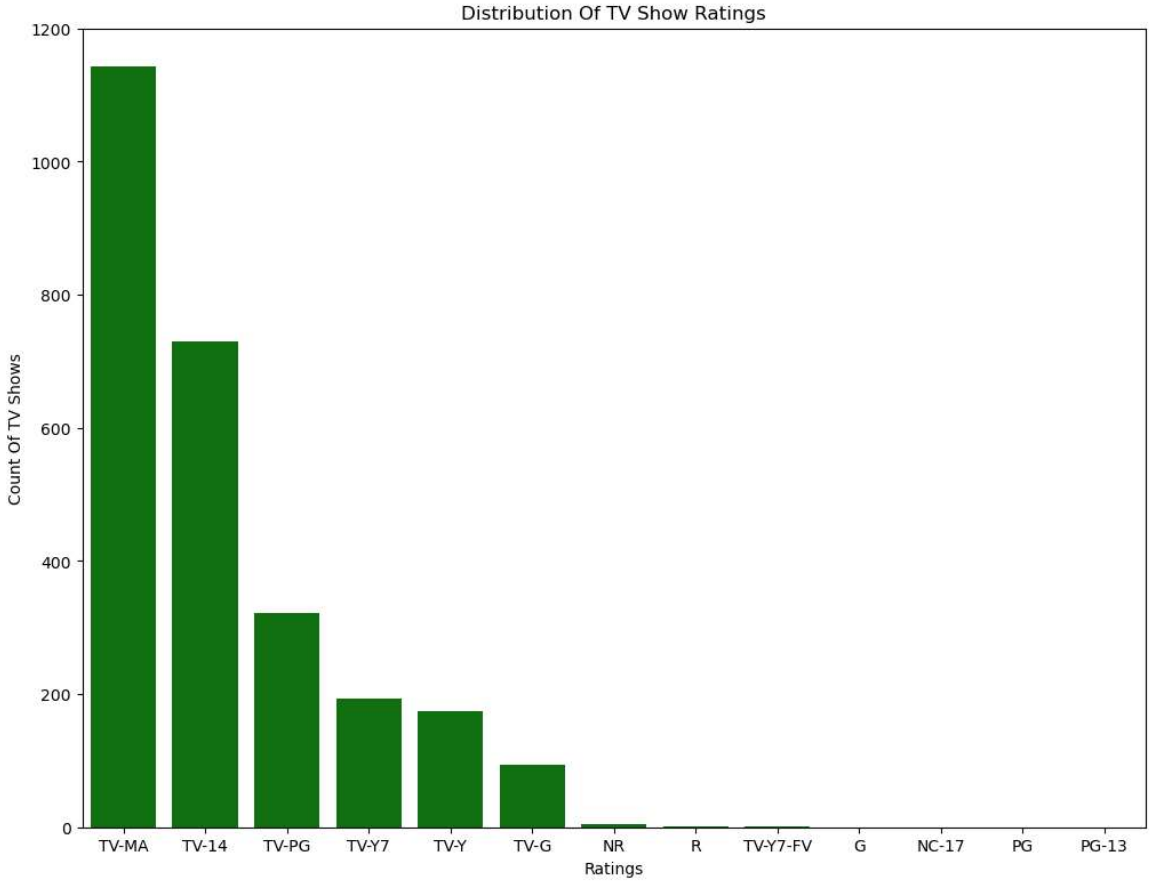
In [165]:

```
movie_rating = movie_data.groupby('rating')['show_id'].count().sort_values()
tv_rating = tv_data .groupby('rating')['show_id'].count().sort_values()
```

In [167]:

```
plt.figure(figsize=(12,20))

plt.subplot(2,1,1)
order = movie_data["rating"].value_counts().index.tolist()
sns.countplot(movie_data, x="rating",order=order,color="Blue")
plt.xlabel("Ratings")
plt.ylabel("Count Of Movies")
plt.title("Distribution Of Movie Ratings")

plt.subplot(2,1,2)
Order = tv_data["rating"].value_counts().index.tolist()
sns.countplot(tv_data, x="rating",order=Order,color="Green")
plt.xlabel("Ratings")
plt.ylabel("Count Of TV Shows")
plt.title("Distribution Of TV Show Ratings")

plt.show()
```


Distribution Of Movie Ratings
Count Of Movies
2000
1750
1500
1250
1000
750
500
250
0
TV-MA
TV-14
R
TV-PG
PG-13
PG
TV-Y7
TV-Y
TV-G
NR
G
TV-Y7-FV
NC-17
Ratings

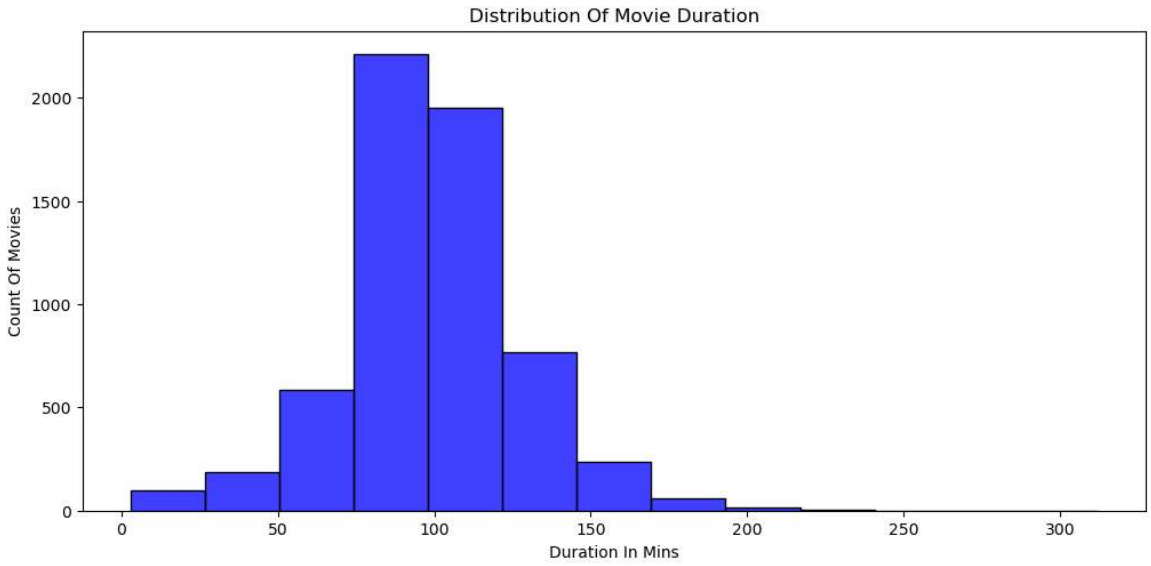

Distribution Of TV Show Ratings
Count Of TV Shows
1200
1000
800
600
400
200
0
TV-MA
TV-14
TV-PG
TV-Y7
TV-Y
TV-G
NR
R
TV-Y7-FV
G
NC-17
PG
PG-13
Ratings

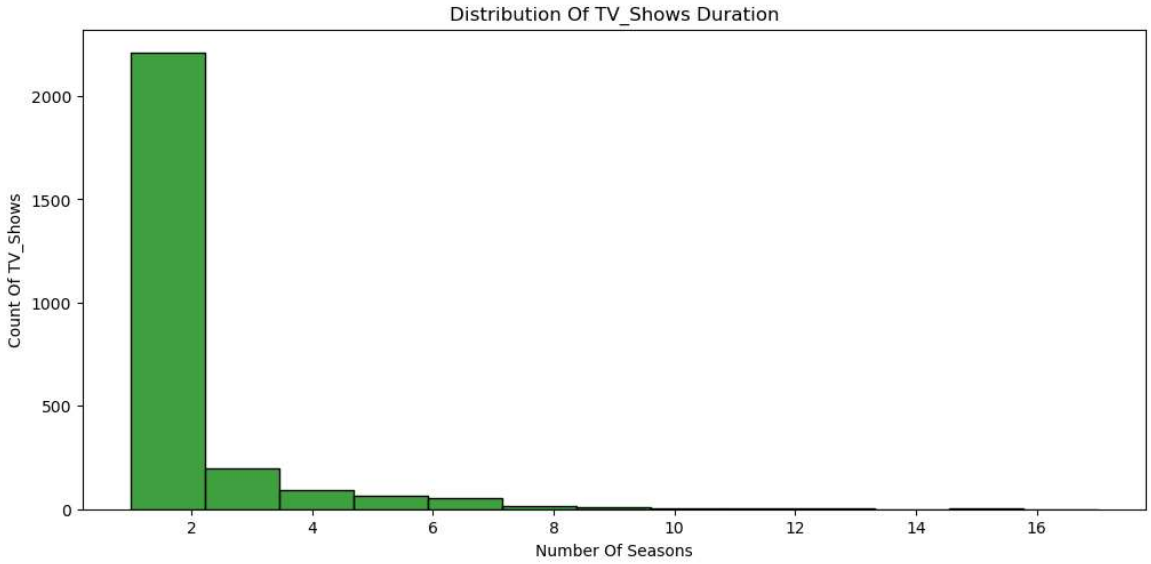
Highest number of movies and TV shows are rated TV-MA (for mature audiences), followed by TV-14 & R/TV-PG

DURATION

In [169]:

```
plt.figure(figsize=(12,12))

plt.subplot(2,1,1)

sns.histplot(data, x="movie_duration",bins=13 ,color="Blue")
plt.xlabel("Duration In Mins")
plt.ylabel("Count Of Movies")
plt.title("Distribution Of Movie Duration")

plt.subplot(2,1,2)
sns.histplot(data, x="tv_seasons",bins=13 ,color="Green")
plt.xlabel("Number Of Seasons")
plt.ylabel("Count Of TV_Shows")
plt.title("Distribution Of TV_Shows Duration")

plt.show()

```





Maximum number of movies have a duration of 75-125 mins

Maximum number of TV shows are 1-2 seasons long

Date_Added

In [170]:

```
k=data.loc[data["year_added"]>=2015]
p=k.groupby(['year_added','month_added'])['show_id'].count().reset_index()
p.rename({"show_id":"Number Of Show/Movie"},axis=1,inplace=True)
p
```

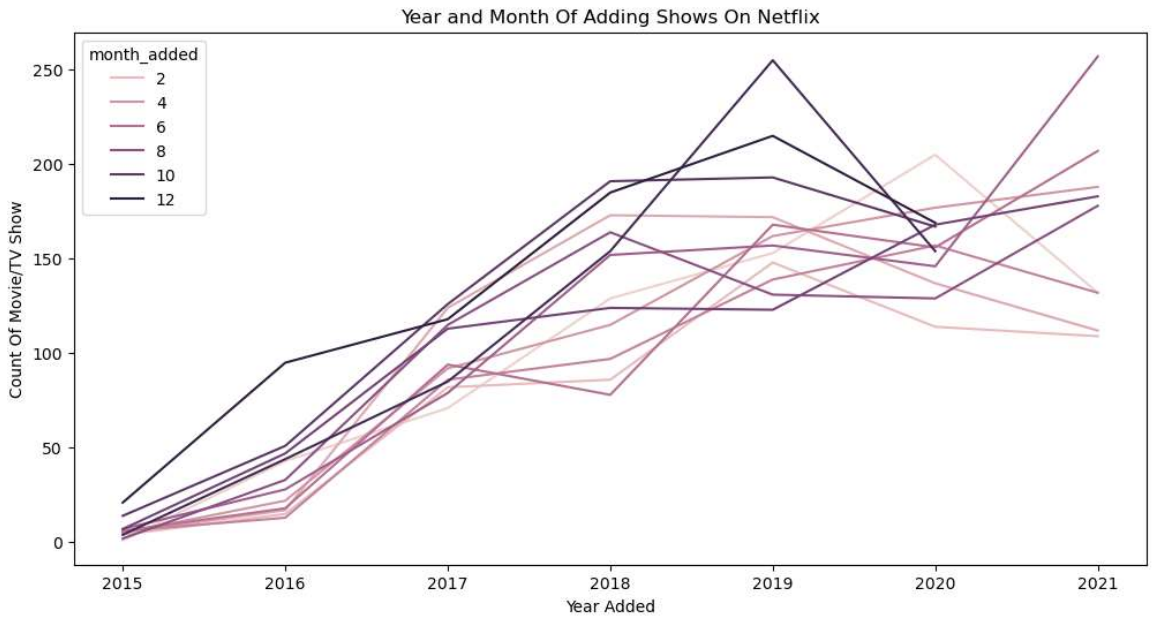
Out[170]:

	year_added	month_added	Number Of Show/Movie
0	2015	1	1
1	2015	2	4
2	2015	3	5
3	2015	4	5
4	2015	5	6
...	...	...	...
76	2021	5	132
77	2021	6	207
78	2021	7	257
79	2021	8	178
80	2021	9	183

81 rows × 3 columns

In [171]:

```
plt.figure(figsize=(12,6))



sns.lineplot(p, x="year_added",y="Number Of Show/Movie" ,hue="month_added")
plt.xlabel("Year Added")
plt.ylabel("Count Of Movie/TV Show")
plt.title("Year and Month Of Adding Shows On Netflix")
plt.show()
```



The number of shows getting added is increasing with each year until 2020

Also, months in the last quarter of the year (Oct-Dec) have more shows being added than the other months of the year. This could be because US has its festive season in Dec and India also has Diwali in Oct-Nov.

COUNTRY

In [45]:

```
data["country"].value_counts()
```

Out[45]:

```
United States                                2809
India                                         972
United Kingdom                                418
Japan                                         243
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: country, Length: 748, dtype: int64
```

We see that many movies are released in more than 1 country. Hence, the country column has comma separated values of countries.

This makes it difficult to analyse how many movies were released in each country. We can use explode function in pandas to split the country column into different rows.

In [46]:

```
country_tb =data[["show_id","type","country"]]
country_tb
```

Out[46]:

	show_id	type	country
0	s1	Movie	United States
1	s2	TV Show	South Africa
2	s3	TV Show	NaN
3	s4	TV Show	NaN
4	s5	TV Show	India
...	...	...	...
8802	s8803	Movie	United States
8803	s8804	TV Show	NaN
8804	s8805	Movie	United States
8805	s8806	Movie	United States
8806	s8807	Movie	India

8790 rows × 3 columns

In [47]:

```
country_tb.isna().sum()
```

Out[47]:

```
show_id      0
type         0
country    829
dtype: int64
```

There are 829 null values in the country column , first we will remove them

In [48]:

```
1 country_tb.dropna(subset=["country"],inplace=True)
```

```
C:\Users\NABINA\AppData\Local\Temp\ipykernel_11712\3866619214.py:1: Settin
gWithCopyWarning:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  country_tb.dropna(subset=["country"],inplace=True)
```

In [49]:

```
1 country_tb.isna().sum()
```

Out[49]:

```
show_id    0
type       0
country    0
dtype: int64
```

Unnesting The Country Column

In [50]:

```
country_tb["country"] = country_tb["country"].str.split(", ")
country_tb = country_tb.explode("country")
country_tb
```

```
C:\Users\NABINA\AppData\Local\Temp\ipykernel_11712\2064027723.py:1: Settin
gWithCopyWarning:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  country_tb["country"] = country_tb["country"].str.split(", ")
```

Out[50]:

	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

9999 rows × 3 columns

In [51]:

```
country_count = country_tb.groupby(["type","country"]).count()["show_id"].sort_values
country_count.rename({"show_id":"Number Of Movie/Show"},axis=1,inplace=True)
country_count
```

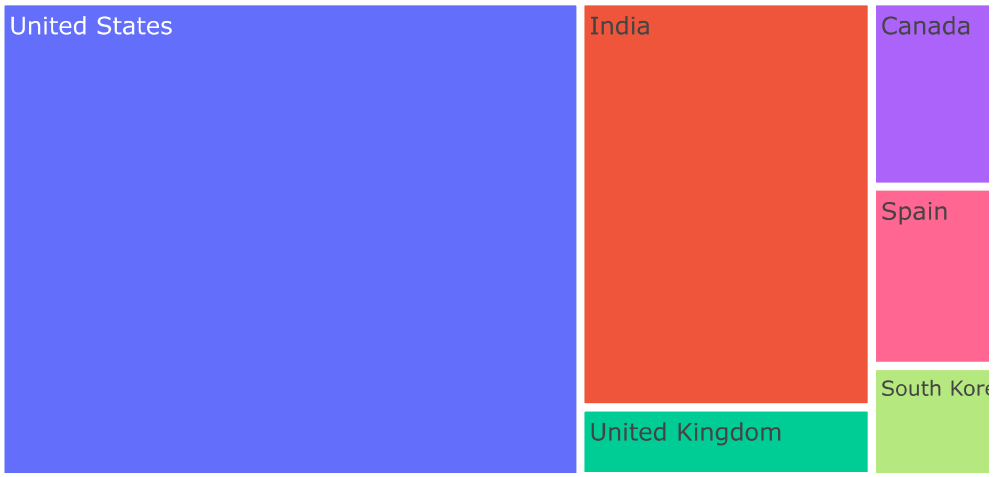
Out[51]:

	type	country	Number Of Movie/Show
0	Movie	United States	2748
1	Movie	India	962
2	TV Show	United States	932
3	Movie	United Kingdom	532
4	Movie	Canada	319
...	...	...	...
249	TV Show	Kenya	0
250	TV Show	Kazakhstan	0
251	TV Show	Jamaica	0
252	TV Show	Iraq	0
253	TV Show	Zimbabwe	0

254 rows × 3 columns

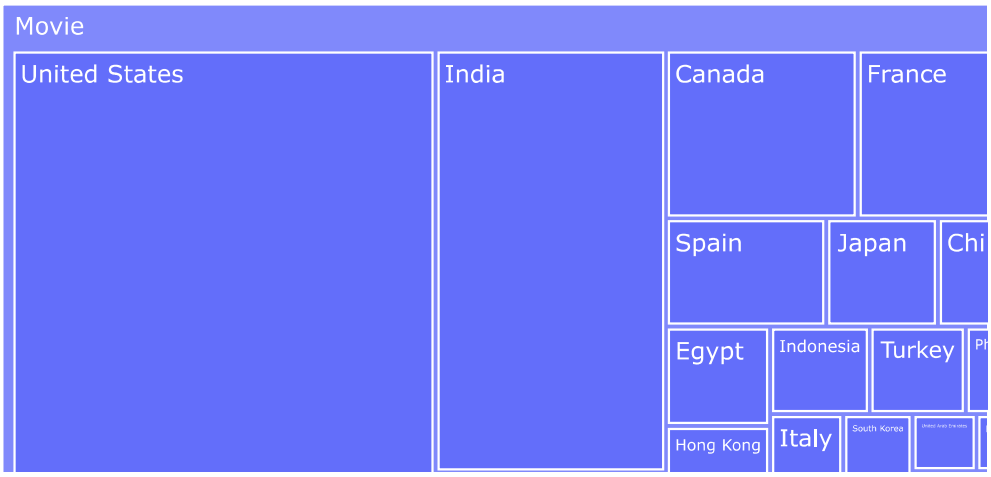
In [52]:

```
px.treemap(
    country_count,
    values='Number Of Movie/Show',
    path=['country']
)

```



In [53]:

```
px.treemap(
    country_count,
    values='Number Of Movie/Show',
    path=['type','country']
)


```



USA, followed by India, UK, Canada, France have the highest number of movie listings.

USA, followed by UK, Japan, South Korea and Canada have the highest number of TV show listings

In [54]:

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

Out[54]:

127

In [55]:

```
#We can use the following top 6 countries for analysis
Top_6 = ['United States', 'United Kingdom', 'India',
         'Canada', 'France', 'Japan']
```

In [56]:

```
Top_6
```

Out[56]:

```
['United States', 'United Kingdom', 'India', 'Canada', 'France', 'Japan']
```

Visual Analysis: Bivariate/Multivariate

POPULAR GENRES BY COUNTRY

In [184]:

```
genre_tb = data[["show_id","listed_in"]]
genre_tb
```

Out[184]:

	show_id	listed_in
0	s1	Documentaries
1	s2	International TV Shows, TV Dramas, TV Mysteries
2	s3	Crime TV Shows, International TV Shows, TV Act...
3	s4	Docuseries, Reality TV
4	s5	International TV Shows, Romantic TV Shows, TV ...
...	...	...
8802	s8803	Cult Movies, Dramas, Thrillers
8803	s8804	Kids' TV, Korean TV Shows, TV Comedies
8804	s8805	Comedies, Horror Movies
8805	s8806	Children & Family Movies, Comedies
8806	s8807	Dramas, International Movies, Music & Musicals

8790 rows × 2 columns

In [185]:

```
genre_tb.isna().sum()
```

Out[185]:

```
show_id      0
listed_in    0
dtype: int64
```

In [186]:

```
# Splitting the listed_in column

genre_tb["listed_in"] = genre_tb["listed_in"].str.split(", ")
genre_tb = genre_tb.explode("listed_in")
genre_tb
```

```
C:\Users\NABINA\AppData\Local\Temp\ipykernel_15468\1189571133.py:3: Settin
gWithCopyWarning:


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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
```

Out[186]:

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

19294 rows × 2 columns

In [187]:

```
genre_wise=genre_tb.groupby("listed_in").count()['show_id'].sort_values(ascending=Fal
genre_wise.rename({"listed_in":"Genre","show_id":"Number of Movie/Shows"},axis=1,inpl
genre_wise
```

Out[187]:

	Genre	Number of Movie/Shows
0	International Movies	2752
1	Dramas	2426
2	Comedies	1674
3	International TV Shows	1349
4	Documentaries	869
5	Action & Adventure	859
6	TV Dramas	762
7	Independent Movies	756
8	Children & Family Movies	641
9	Romantic Movies	616
10	Thrillers	577
11	TV Comedies	573
12	Crime TV Shows	469
13	Kids' TV	448
14	Docuseries	394
15	Music & Musicals	375
16	Romantic TV Shows	370
17	Horror Movies	357
18	Stand-Up Comedy	343
19	Reality TV	255
20	British TV Shows	252
21	Sci-Fi & Fantasy	243
22	Sports Movies	219
23	Anime Series	174
24	Spanish-Language TV Shows	173
25	TV Action & Adventure	167
26	Korean TV Shows	151
27	Classic Movies	116
28	LGBTQ Movies	102
29	TV Mysteries	98
30	Science & Nature TV	92
31	TV Sci-Fi & Fantasy	83
32	TV Horror	75
33	Anime Features	71
34	Cult Movies	71
35	Teen TV Shows	69
36	Faith & Spirituality	65

	Genre	Number of Movie/Shows
37	TV Thrillers	57
38	Stand-Up Comedy & Talk Shows	56
39	Movies	53
40	Classic & Cult TV	26
41	TV Shows	16

TOP 10 Genres Based ON MOVIE/TV Shows

In [188]:

```
1 genre_wise[:10]
```

Out[188]:

	Genre	Number of Movie/Shows
0	International Movies	2752
1	Dramas	2426
2	Comedies	1674
3	International TV Shows	1349
4	Documentaries	869
5	Action & Adventure	859
6	TV Dramas	762
7	Independent Movies	756
8	Children & Family Movies	641
9	Romantic Movies	616

Top 20 countries by count of titles

In [234]:

```
1 merged_df_country = pd.merge(data,country_tb,on="show_id").drop(['country_x' , 'type_
2
```

In [235]:

```
1 merged_df_country.merge(genre_tb,on="show_id").drop('listed_in_x' , axis = 1).rename({'lis
2
```

In [236]:

```
#Top 20 countries by count of titles

top_20 = merged_df_country["country"].value_counts().reset_index()
top_20 = top_20.loc[:20,'index'].to_list()
top_20
top_20_df = merged_df_country_genre.loc[merged_df_country_genre ['country'].isin(top_
```

In [192]:

```
data_details= pd.pivot_table(top_20_df, index = top_20_df['listed_in'],
                        columns = 'country',
                        values = 'show_id', aggfunc='count')
data_details
```

Out[192]:

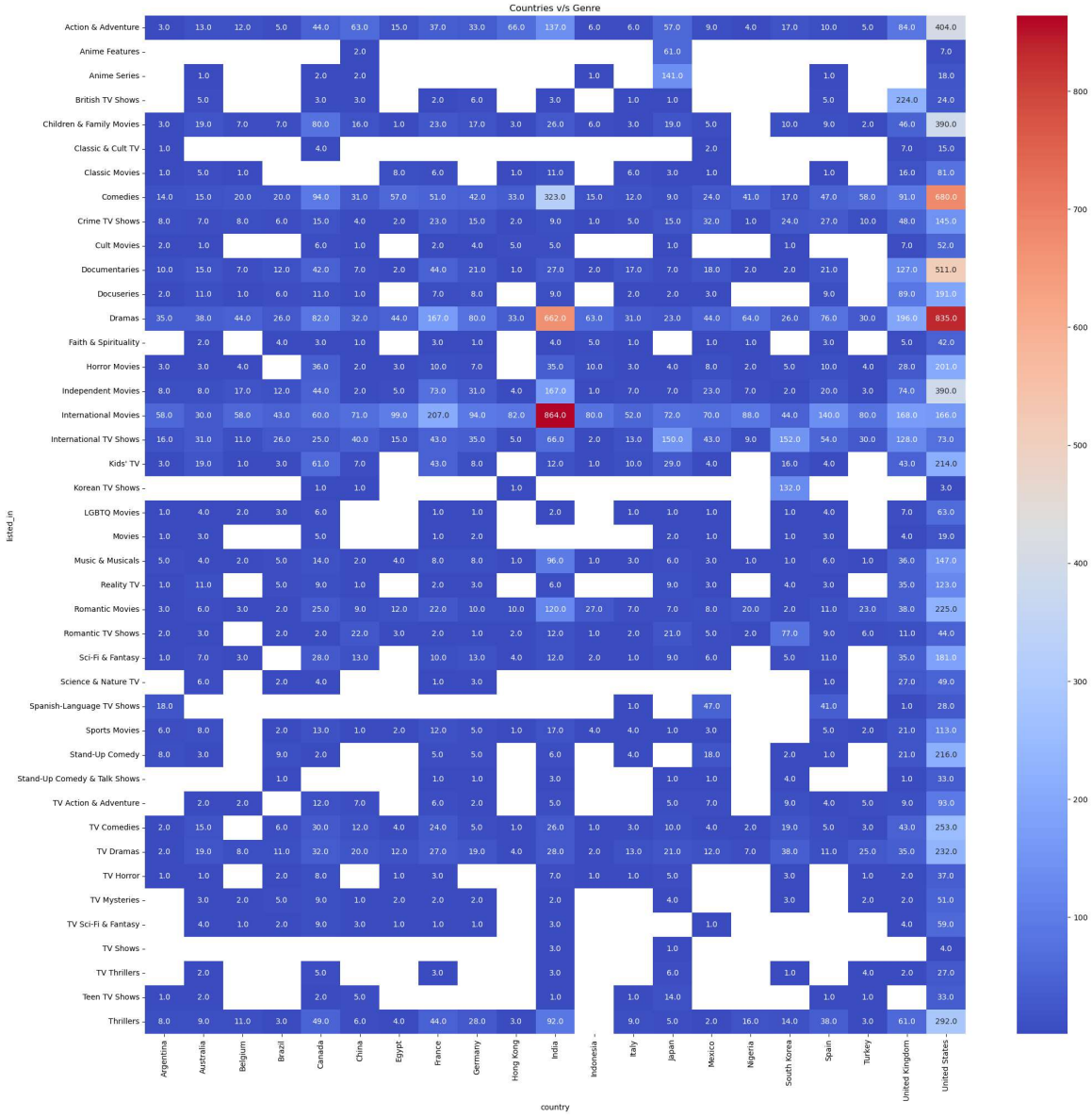
country listed_in	Argentina	Australia	Belgium	Brazil	Canada	China	Egypt	France	German
Action & Adventure	3.0	13.0	12.0	5.0	44.0	63.0	15.0	37.0	33.
Anime Features	NaN	NaN	NaN	NaN	NaN	2.0	NaN	NaN	Nal
Anime Series	NaN	1.0	NaN	NaN	2.0	2.0	NaN	NaN	Nal
British TV Shows	NaN	5.0	NaN	NaN	3.0	3.0	NaN	2.0	6.
Children & Family Movies	3.0	19.0	7.0	7.0	80.0	16.0	1.0	23.0	17.
Classic & Cult TV	1.0	NaN	NaN	NaN	4.0	NaN	NaN	NaN	Nal
Classic Movies	1.0	5.0	1.0	NaN	NaN	NaN	8.0	6.0	Nal
Comedies	14.0	15.0	20.0	20.0	94.0	31.0	57.0	51.0	42.
Crime TV Shows	8.0	7.0	8.0	6.0	15.0	4.0	2.0	23.0	15.
Cult Movies	2.0	1.0	NaN	NaN	6.0	1.0	NaN	2.0	4.
Documentaries	10.0	15.0	7.0	12.0	42.0	7.0	2.0	44.0	21.
Docuseries	2.0	11.0	1.0	6.0	11.0	1.0	NaN	7.0	8.
Dramas	35.0	38.0	44.0	26.0	82.0	32.0	44.0	167.0	80.
Faith & Spirituality	NaN	2.0	NaN	4.0	3.0	1.0	NaN	3.0	1.
Horror Movies	3.0	3.0	4.0	NaN	36.0	2.0	3.0	10.0	7.
Independent Movies	8.0	8.0	17.0	12.0	44.0	2.0	5.0	73.0	31.
International Movies	58.0	30.0	58.0	43.0	60.0	71.0	99.0	207.0	94.
International TV Shows	16.0	31.0	11.0	26.0	25.0	40.0	15.0	43.0	35.
Kids' TV	3.0	19.0	1.0	3.0	61.0	7.0	NaN	43.0	8.
Korean TV Shows	NaN	NaN	NaN	NaN	1.0	1.0	NaN	NaN	Nal
LGBTQ Movies	1.0	4.0	2.0	3.0	6.0	NaN	NaN	1.0	1.
Movies	1.0	3.0	NaN	NaN	5.0	NaN	NaN	1.0	2.
Music & Musicals	5.0	4.0	2.0	5.0	14.0	2.0	4.0	8.0	8.
Reality TV	1.0	11.0	NaN	5.0	9.0	1.0	NaN	2.0	3.
Romantic Movies	3.0	6.0	3.0	2.0	25.0	9.0	12.0	22.0	10.
Romantic TV Shows	2.0	3.0	NaN	2.0	2.0	22.0	3.0	2.0	1.
Sci-Fi & Fantasy	1.0	7.0	3.0	NaN	28.0	13.0	NaN	10.0	13.

country	Argentina	Australia	Belgium	Brazil	Canada	China	Egypt	France	German
listed_in									
Science & Nature TV	NaN	6.0	NaN	2.0	4.0	NaN	NaN	1.0	3.
Spanish-Language TV Shows	18.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
Sports Movies	6.0	8.0	NaN	2.0	13.0	1.0	2.0	12.0	5.
Stand-Up Comedy	8.0	3.0	NaN	9.0	2.0	NaN	NaN	5.0	5.
Stand-Up Comedy & Talk Shows	NaN	NaN	NaN	1.0	NaN	NaN	NaN	1.0	1.
TV Action & Adventure	NaN	2.0	2.0	NaN	12.0	7.0	NaN	6.0	2.
TV Comedies	2.0	15.0	NaN	6.0	30.0	12.0	4.0	24.0	5.
TV Dramas	2.0	19.0	8.0	11.0	32.0	20.0	12.0	27.0	19.
TV Horror	1.0	1.0	NaN	2.0	8.0	NaN	1.0	3.0	Nal
TV Mysteries	NaN	3.0	2.0	5.0	9.0	1.0	2.0	2.0	2.
TV Sci-Fi & Fantasy	NaN	4.0	1.0	2.0	9.0	3.0	1.0	1.0	1.
TV Shows	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
TV Thrillers	NaN	2.0	NaN	NaN	5.0	NaN	NaN	3.0	Nal
Teen TV Shows	1.0	2.0	NaN	NaN	2.0	5.0	NaN	NaN	Nal
Thrillers	8.0	9.0	11.0	3.0	49.0	6.0	4.0	44.0	28.

42 rows × 21 columns

In [193]:

```
plt.figure(figsize=(24,24))
sns.heatmap(data_details, cmap='coolwarm', annot=True, fmt=".1f", cbar=True)
plt.title("Countries v/s Genre")
plt.show()
```



In [194]:

```
Res=merged_df_country_genre.groupby(["country","listed_in"])["show_id"].count().sort_
Res.rename({"listed_in":"Genre","show_id":"Number Of Show/Movie"},axis=1,inplace=True
Res
```

Out[194]:

	country	Genre	Number Of Show/Movie
0	India	International Movies	864
1	United States	Dramas	835
2	United States	Comedies	680
3	India	Dramas	662
4	United States	Documentaries	511
...	...	...	...
1417	Mauritius	International TV Shows	1
1418	Mauritius	TV Dramas	1
1419	Mexico	Classic Movies	1
1420	Mexico	Faith & Spirituality	1
1421	Zimbabwe	Romantic Movies	1

1422 rows × 3 columns

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

Country-wise Rating of Content

In [195]:

```
1 top_20_df.groupby(["rating","country"])["show_id"].count().sort_values(ascending=Fals
2
```

Out[195]:

	rating	country	show_id
0	TV-MA	United States	2023
1	TV-14	India	1569
2	R	United States	1268
3	TV-14	United States	885
4	PG-13	United States	854
...	...	...	...
268	NC-17	Australia	0
269	PG	Turkey	0
270	TV-Y	Indonesia	0
271	NC-17	Brazil	0
272	PG-13	Nigeria	0

273 rows × 3 columns

In [196]:

```
#Preparing  the data for visualisation

data_check = pd.pivot_table(top_20_df, index = top_20_df['rating'],
                        columns = top_20_df['country'],
                        values = 'show_id', aggfunc='count')
data_check
```

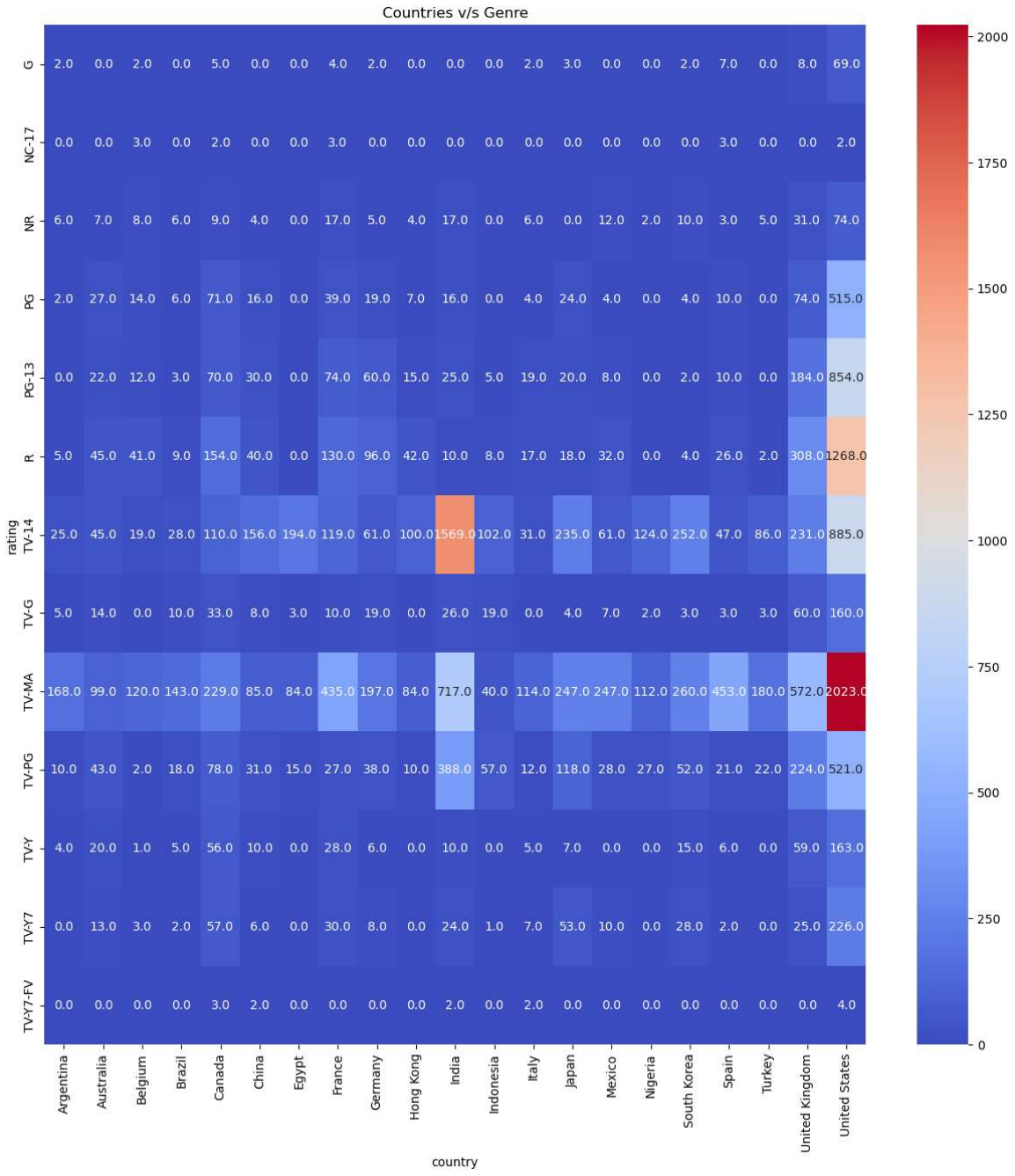
Out[196]:

country rating	Argentina	Australia	Belgium	Brazil	Canada	China	Egypt	France	Germany	Hon Kon
G	2	0	2	0	5	0	0	4	2	
NC-17	0	0	3	0	2	0	0	3	0	
NR	6	7	8	6	9	4	0	17	5	
PG	2	27	14	6	71	16	0	39	19	
PG-13	0	22	12	3	70	30	0	74	60	1
R	5	45	41	9	154	40	0	130	96	4
TV-14	25	45	19	28	110	156	194	119	61	10
TV-G	5	14	0	10	33	8	3	10	19	
TV-MA	168	99	120	143	229	85	84	435	197	8
TV-PG	10	43	2	18	78	31	15	27	38	1
TV-Y	4	20	1	5	56	10	0	28	6	
TV-Y7	0	13	3	2	57	6	0	30	8	
TV-Y7-FV	0	0	0	0	3	2	0	0	0	

13 rows × 21 columns

In [197]:

```
#Heatmap using Seaborn

plt.figure(figsize=(15,15))
sns.heatmap(data_check, cmap='coolwarm', annot=True, fmt=".1f", cbar=True)
plt.title("Countries v/s Genre")
plt.show()
```



Overall, Netflix has an unproportionately 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

Popular Actors by Country

In [198]:

```
cast_df = merged_df_country_genre.copy()

```

In [199]:

```
#Checking For Nulls

cast_df.isna().sum()
```

Out[199]:

```
show_id              0
type                 0
title                0
director          5693
cast              1547
date_added           0
release_year         0
rating               0
duration             0
description          0
movie_duration    5994
tv_seasons       16011
year_added           0
month_added          0
country              0
listed_in            0
dtype: int64
```

In [200]:

```
#Removing Nulls

cast_df.dropna(subset=["cast"],inplace=True)

```

In [201]:

```
# Splitting the cast column and Explode the rows with multiple actors to different ro

cast_df["cast"] = cast_df["cast"].str.split(", ")
cast_df = cast_df.explode("cast")
```

In [202]:

```
country_names= Top_6
country_names
```

Out[202]:

```
['United States', 'United Kingdom', 'India', 'Canada', 'France', 'Japan']
```

In [203]:

```
v_names = np.array(['india_actors', 'uk_actors', 'us_actors',
        'canada_actors', 'france_actors', 'japan_actors'])
```

In [204]:

```
v_names=list(v_names)
v_names
```

Out[204]:

```
['india_actors',
 'uk_actors',
 'us_actors',
 'canada_actors',
 'france_actors',
 'japan_actors']
```

In [205]:

```
cast_df.loc[(cast_df['type']=='Movie') &(cast_df['country']=="United States")].groupb


```

Out[205]:

	cast	show_id
0	Samuel L. Jackson	21
1	Adam Sandler	20
2	James Franco	19
3	Nicolas Cage	18
4	Seth Rogen	15
...	...	...
11798	Jacky Cheung	1
11799	Jackée Harry	1
11800	Jaclyn Ngan	1
11801	Jacob Artist	1
11802	Óscar Jaenada	1

11803 rows × 2 columns

In [206]:

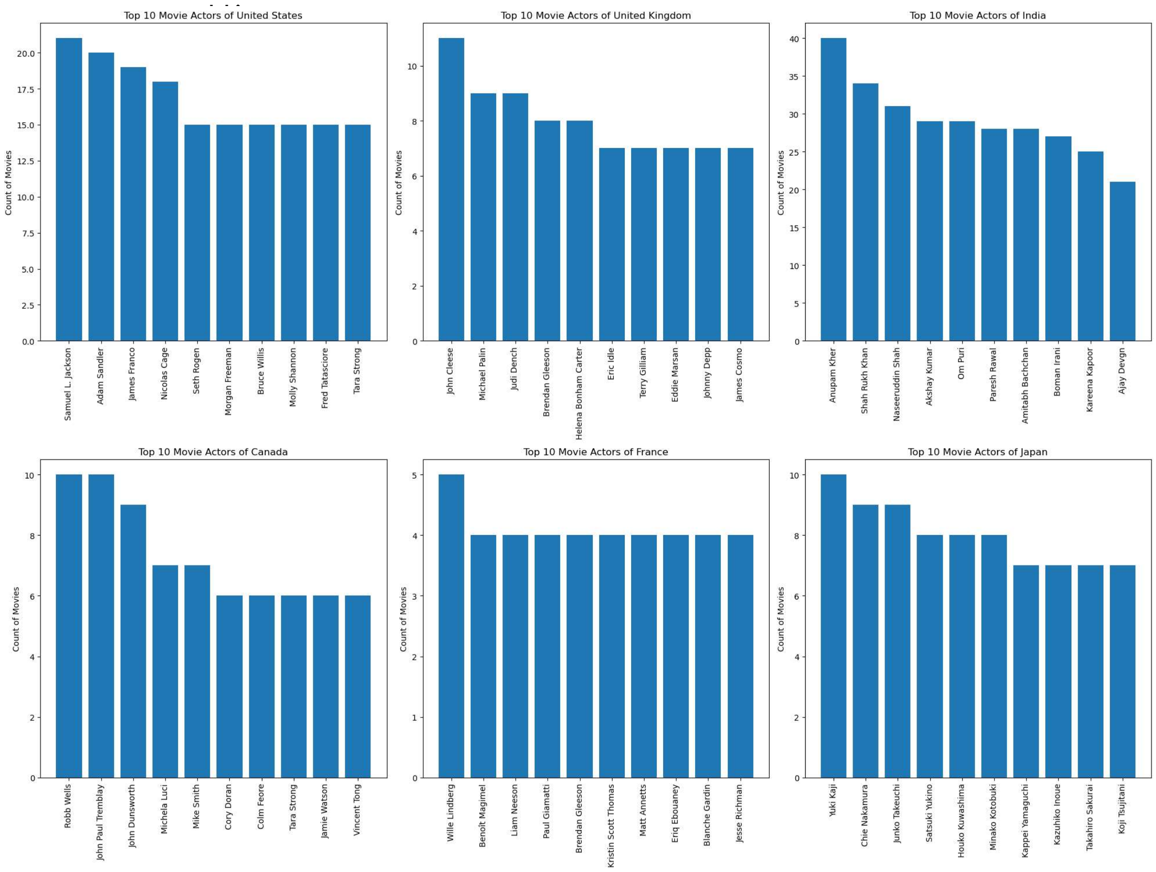
```
#Preparing Data for visualisation

datas=[]

def country_wise_actors(country_name, var_name):
  var_name = cast_df.loc[(cast_df['type']=='Movie') &
                                (cast_df['country']==country_name)].groupby(
                                    'cast')['show_id'].nunique().sort_values(
                                        ascending=False).reset_index().head(10)
  datas.append(var_name)
  return data

for country_name, var_name in zip(country_names, v_names):
  country_wise_actors(country_name, var_name)

datas
```

Out[206]:

```
[                 cast  show_id
 0  Samuel L. Jackson       21
 1       Adam Sandler       20
 2       James Franco       19
 3       Nicolas Cage       18
 4         Seth Rogen       15
 5     Morgan Freeman       15
 6       Bruce Willis       15
 7     Molly Shannon        15
 8    Fred Tatasciore       15
 9        Tara Strong       15,
                  cast  show_id
 0         John Cleese       11
 1       Michael Palin        9
 2          Judi Dench        9
 3     Brendan Gleeson        8
 4  Helena Bonham Carter      8
 5           Eric Idle        7
 6        Terry Gilliam       7
```

In [208]:

```
fig, axes = plt.subplots(2, 3, figsize=(20,15))
for i, ax in enumerate(axes.flatten()):
    ax.bar(datas[i]["cast"], datas[i]["show_id"])
    ax.set_xticks(np.arange(0,10), labels=datas[i]['cast'], rotation='vertical')
    ax.set_title(f'Top 10 Movie Actors of {country_names[i]}')
    ax.set_ylabel('Count of Movies')
plt.tight_layout()
plt.show();
```




```
 0       Wille Lindberg        5
 1       Benoît Magimel        4
 2          Liam Neeson        4
 3        Paul Giamatti        4
 4      Brendan Gleeson        4
 5  Kristin Scott Thomas       4
 6         Matt Annetts        4
 7       Eriq Ebouaney         4
 8       Blanche Gardin        4
 9        Jesse Richman        4,
               cast  show_id
 0        Yuki Kaji       10
 1    Chie Nakamura        9
 2   Junko Takeuchi        9
 3   Satsuki Yukino        8
 4  Houko Kuwashima        8
```

Popular Directors by genre

In [209]:

```
director_df = merged_df_country_genre.copy()
```

```
5   Minako Kotobuki         8
6   Kappei Yamaguchi        7
7     Kazuhiko Inoue        7
8   Takahiro Sakurai        7
9     Koji Tsujitani        7]
```

In [211]:

```
#Removing Nulls.

director_df.dropna(subset=["director"],inplace=True)
```

In [212]:

```
director_df.isna().sum()
```

Out[212]:

```
show_id               0
type                  0
title                 0
director              0
cast                993
date_added            0
release_year          0
rating                0
duration              0
description           0
movie_duration      554
tv_seasons        15758
year_added            0
month_added           0
country               0
listed_in             0
dtype: int64
```

In [213]:

```
# Splitting the cast column and Explode the rows with multiple actors to different ro

director_df["director"] = director_df["director"].str.split(", ")
director_df = director_df.explode("director")
```

In [214]:

```
#Preparing Data for visualisation

genre_names = ['Action & Adventure', 'Children & Family Movies', 'Comedies',
               'Dramas', 'International Movies', 'International TV Shows',
               'Sci-Fi & Fantasy', 'Thrillers']

v_names = ['action', 'family', 'comedies', 'dramas',
        'int_movies', 'int_tv', 'scifi', 'thrillers']

datas=[]

def genre_wise_directors(genre_name, var_name):
  var_name = director_df.loc[director_df['listed_in']==genre_name].groupby('director'

  datas.append(var_name)
  return datas

for genre_name, var_name in zip(genre_names, v_names):
  genre_wise_directors(genre_name, var_name)

datas

```

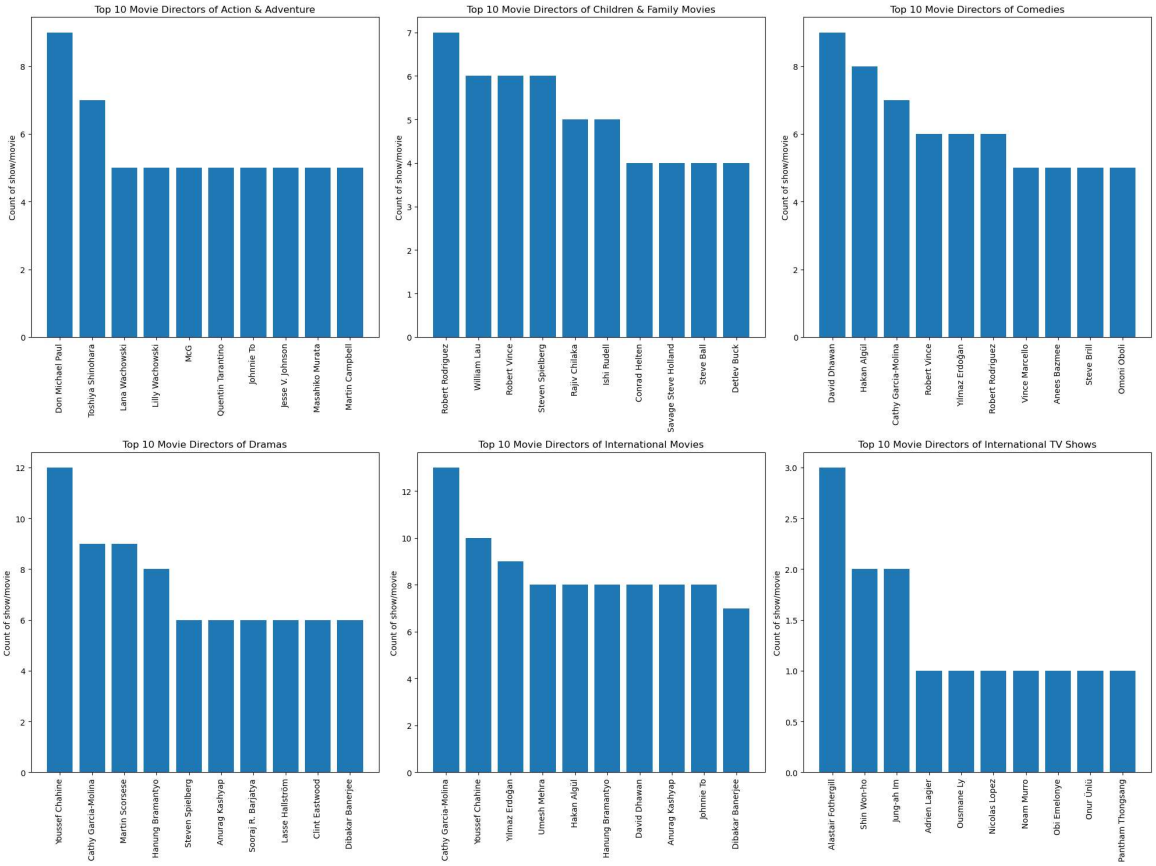
Out[214]:

```
[            director  show_id
 0   Don Michael Paul        9
 1  Toshiya Shinohara        7
 2     Lana Wachowski        5
 3    Lilly Wachowski        5
 4                McG        5
 5  Quentin Tarantino        5
 6         Johnnie To        5
 7   Jesse V. Johnson        5
 8    Masahiko Murata        5
 9    Martin Campbell        5,
               director  show_id
 0      Robert Rodriguez        7
 1           William Lau        6
 2          Robert Vince        6
 3      Steven Spielberg        6
 4         Rajiv Chilaka        5
 5           Ishi Rudell        5
 6         Conrad Helten        4
 7  Savage Steve Holland        4
 8            Steve Ball        4
 9           Detlev Buck        4,
                director  show_id
 0         David Dhawan        9
 1          Hakan Algül        8
 2  Cathy Garcia-Molina        7
 3         Robert Vince        6
 4       Yılmaz Erdoğan        6
 5     Robert Rodriguez        6
 6       Vince Marcello        5
 7         Anees Bazmee        5
 8          Steve Brill        5
 9          Omoni Oboli        5,
                director  show_id
 0      Youssef Chahine       12
 1  Cathy Garcia-Molina        9
 2      Martin Scorsese        9
 3     Hanung Bramantyo        8
 4     Steven Spielberg        6
 5       Anurag Kashyap        6
 6   Sooraj R. Barjatya        6
 7      Lasse Hallström        6
 8       Clint Eastwood        6
 9     Dibakar Banerjee        6,
               director  show_id
 0  Cathy Garcia-Molina       13
 1      Youssef Chahine       10
 2       Yılmaz Erdoğan        9
 3          Umesh Mehra        8
 4          Hakan Algül        8
 5     Hanung Bramantyo        8
 6         David Dhawan        8
 7       Anurag Kashyap        8
 8           Johnnie To        8
 9     Dibakar Banerjee        7,
               director  show_id
 0  Alastair Fothergill        3
 1          Shin Won-ho        2
 2           Jung-ah Im        2
 3        Adrien Lagier        1
 4           Ousmane Ly        1
```

```
5        Nicolas Lopez         1
6           Noam Murro         1
7          Obi Emelonye        1
8            Onur Ünlü         1
9     Pantham Thongsang        1
          director  show_id
0       Lana Wachowski        4
1    Lilly Wachowski          4
2  Guillermo del Toro         3
3         Peter Jackson        3
4  Paul W.S. Anderson         3
5      Barry Sonnenfeld       3
```

In [215]:

```
fig, axes = plt.subplots(2, 3, figsize=(20,15))
for i, ax in enumerate(axes.flatten()):
    ax.bar(datas[i]["director"],datas[i]["show_id"])
    ax.set_xticks(np.arange(0,10), labels=datas[i]['director'], rotation='vertical')
    ax.set_title(f'Top 10 Movie Directors of {genre_names[i]}')
    ax.set_ylabel('Count of show/movie')
plt.tight_layout()
plt.show();
```



In [216]:

```
director_details=data.copy()
```

In [217]:

```
director_details["director"] = director_details["director"].str.split(", ")
director_details = director_details.explode("director")
```

In [218]:

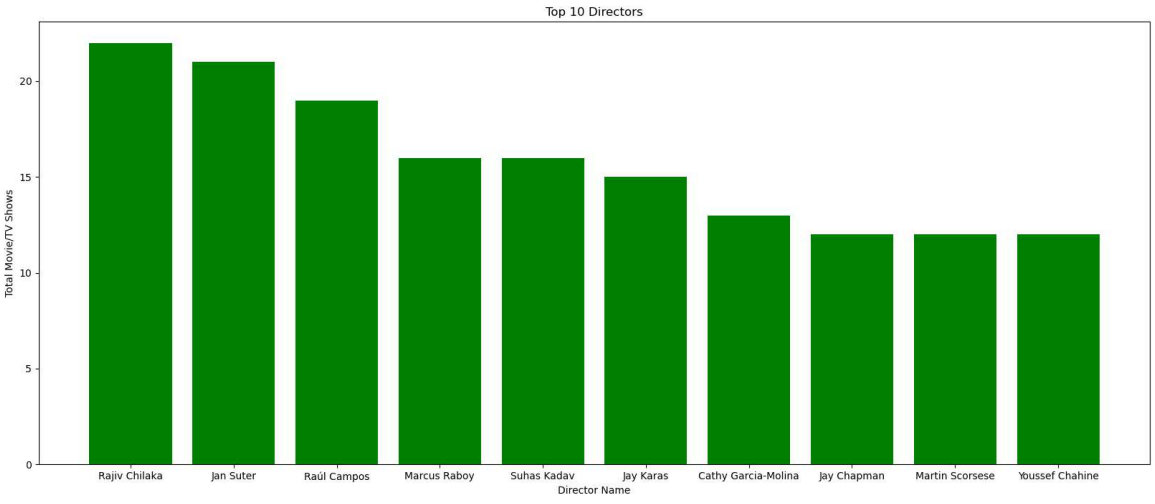
```
k=director_details.groupby('director').nunique()['show_id'].sort_values(ascending=Fal
top_10_directors=k[:10].reset_index()
top_10_directors
categories=list(top_10_directors["director"])
categories
values=list(top_10_directors["show_id"])
values
top_10_directors
```

Out[218]:

	director	show_id
0	Rajiv Chilaka	22
1	Jan Suter	21
2	Raúl Campos	19
3	Marcus Raboy	16
4	Suhas Kadav	16
5	Jay Karas	15
6	Cathy Garcia-Molina	13
7	Jay Chapman	12
8	Martin Scorsese	12
9	Youssef Chahine	12

In [219]:

```
plt.figure(figsize=(20,8))
plt.bar(categories, values,color="Green")
plt.xlabel("Director Name")
plt.ylabel("Total Movie/TV Shows")
plt.title("Top 10 Directors")
plt.show()
```



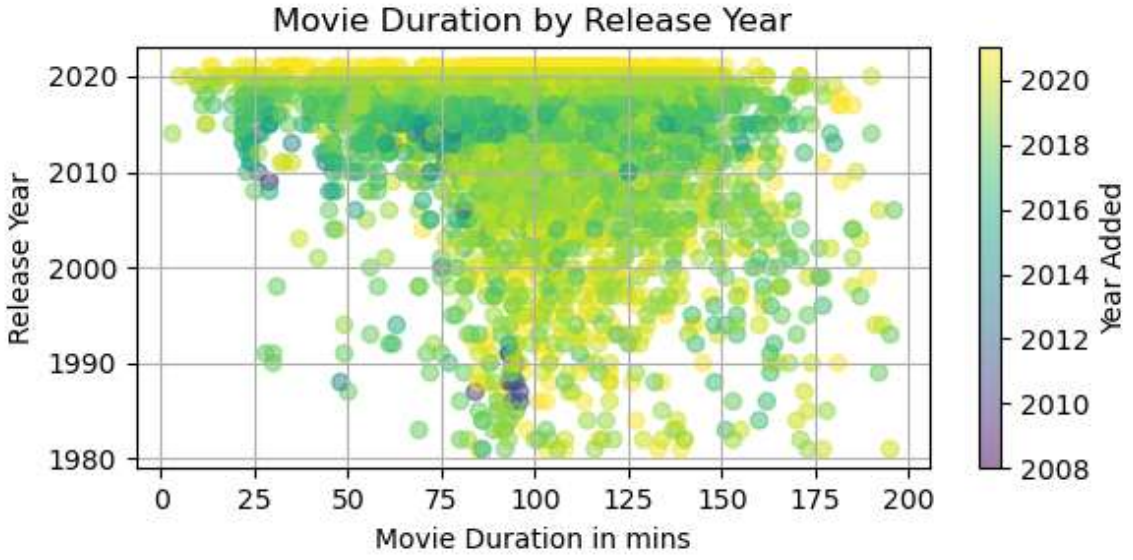
Trend in Duration by Release Year

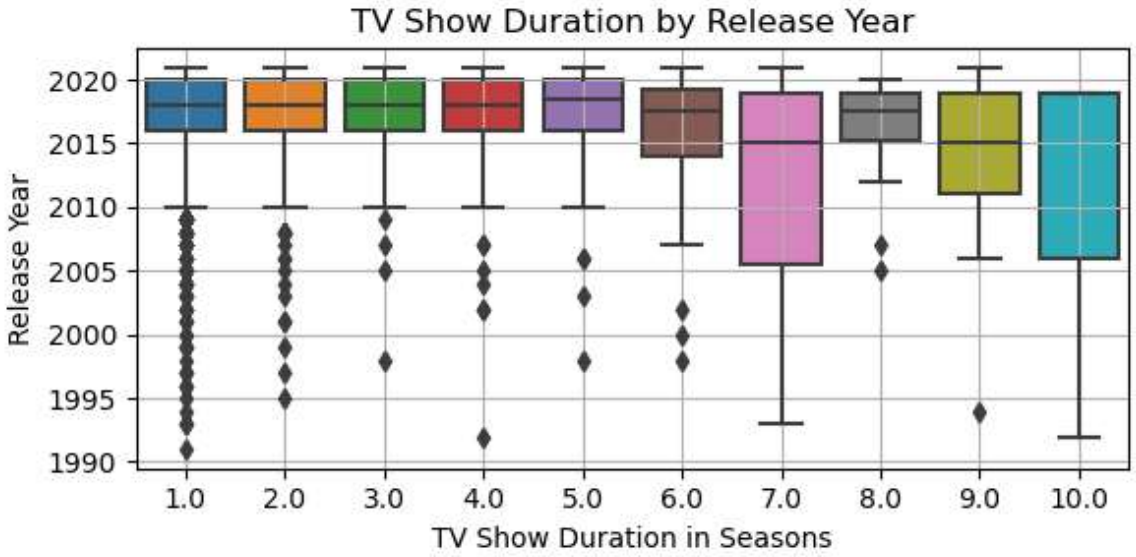
In [223]:

```
# Remove Outliers

movie_data = data[(data['movie_duration']<200) &
                      (data['release_year']>1980)]

tv_data = data[(data['tv_seasons']<=10) &
                (data['release_year']>1990)]
```

In [225]:

```
fig, ax = plt.subplots(2, 1, figsize=(6,6))

scatter1 = ax[0].scatter(movie_data['movie_duration'], movie_data['release_year'],
                         c = movie_data['year_added'], alpha=0.5)
ax[0].set_xlabel('Movie Duration in mins')
ax[0].set_ylabel('Release Year')
ax[0].set_title('Movie Duration by Release Year')
ax[0].grid(True)

sns.boxplot(data = tv_data, x='tv_seasons', y='release_year', ax = ax[1])
ax[1].set_xlabel('TV Show Duration in Seasons')
ax[1].set_ylabel('Release Year')
ax[1].set_title('TV Show Duration by Release Year')
ax[1].grid(True)


fig.tight_layout()

# Add color legend
cbar = fig.colorbar(scatter1, ax=ax[0])
cbar.set_label('Year Added')

plt.show();

```





Best Month To Release A Movie /TV Show

In [231]:

```
1 Tv = data.loc[data["type"]== "TV Show"]
```

In [232]:

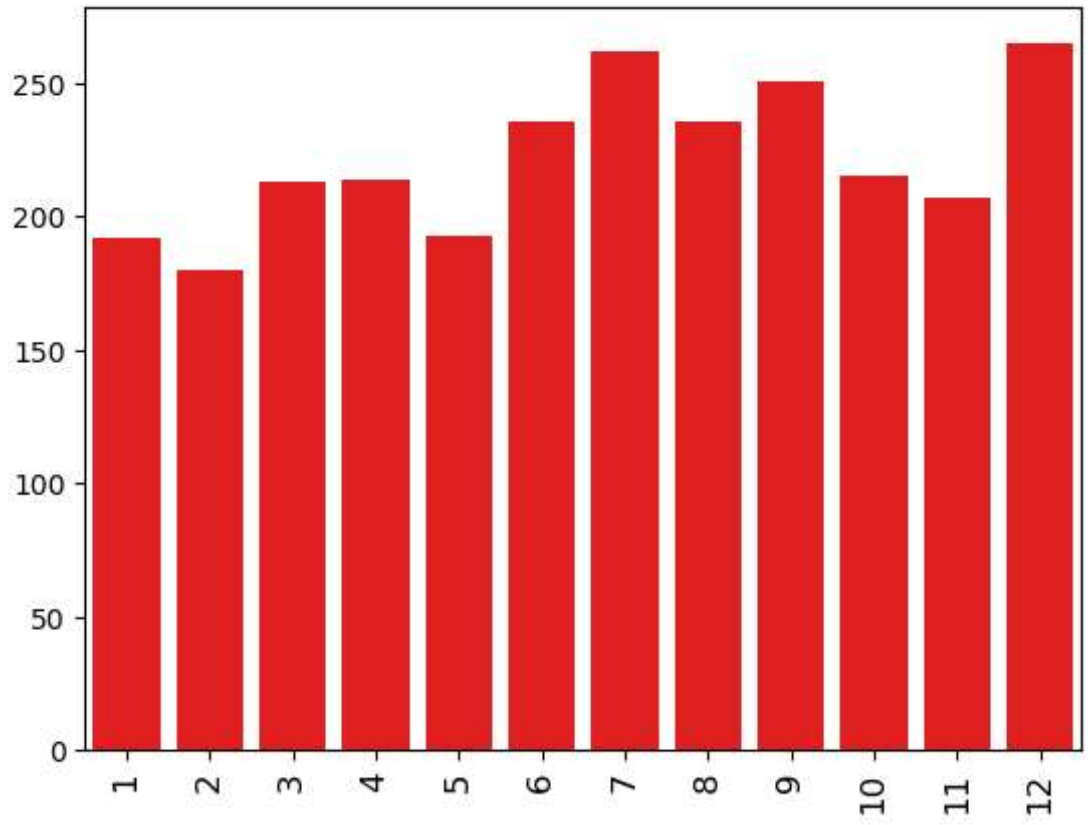
```
1 months=Tv["month_added"].value_counts()
2 months
```

Out[232]:

```
12    265
7     262
9     251
8     236
6     236
10    215
4     214
3     213
11    207
5     193
1     192
2     180
Name: month_added, dtype: int64
```

In [233]:

```
1 sns.barplot(data, x=months.index,y = months.values,  color='red')
2 plt.xticks(rotation=90, fontsize=12)
3 plt.show()
```



December is the best month to release a Tv Show

BUSINESS INSIGHTS

Range of Attributes:

Type of Show: Approx 70% shows are movies and remaining 30% are TV shows

The release year for shows is concentrated in the range 2000-2021

The year of adding shows is concentrated in the range 2015-2021

Movie Duration: 50 mins - 150 mins is the range excluding potential outliers (values lying outside the whiskers of boxplot)

TV Show Duration: 1-3 seasons is the range for TV shows excluding potential outliers

Countries: Out of 128 countries present in the dataset, only 23 countries have more than 50 movie titles and only 11 countries have more than 50 TV Shows

Ratings: There are 12 different kinds of ratings based on the relevance of the content for different age-groups

The dataset consists of 36,392 actors and 4,991 directors

Visual Analysis:

Release Year & Year/Month of Addition to Netflix:

2018 marks the highest number of movie and TV show releases

The yearly number of releases has surged drastically from 2015, so has the number of movies being added.

Months in the last quarter of the year (Oct-Dec) have more shows being added than the other months of the year. This could be because US has its festive season in Dec and India also has Diwali in Oct-Nov.

Type of content across Countries:

USA, followed by India, UK, Canada, France have the highest number of movie listings.

USA, followed by UK, Japan, South Korea and Canada have the highest number of TV show listings

Only US, Canada, UK, France and Japan have content for young audiences (TV-Y & TV-Y7)

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

Content Rating:

Highest number of movies and TV shows are rated TV-MA (for mature audiences), followed by TV-14 (14 yrs & above) & R/TV-PG (Restricted/Parental Guidance). Overall, Netflix has an unproportionately large amount of adult content across all countries.

There is scarce content for general audience (TV-G & G) across all countries except US

Genre:

Most popular genres: Action & Adventure, Children & Family Movies, Comedies, Dramas, International Movies & TV Shows, TV Dramas, Thrillers

Duration:

There is a surge in the number of short duration movies (less than 75 mins) post 2010

All 1-5 season TV shows are concentrated in 2010-2020 release year window. Older TV Shows have more number of seasons.

In [ ]:

```
1
```

In [ ]:

```
1
```