Business Case Study: `Netflix`

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

Our goal is to analyze 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.

Importing libraries & Data

In [1]:

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

In [2]:

```
df = pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940/original/netflix.csv')
df.head()
```

Out[2]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train l...

Understanding data

In [3]:

```
print("No. of rows:",df.shape[0],"\nNo. of columns",df.shape[1])
```

```
No. of rows: 8807
No. of columns 12
```

Based on the above info we have data related to 8,807 movies/TV shows. We will confirm this again based on the unique `show_id` values

In [4]:

```
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
```

We are also provided with a **data dictionary**:

- `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

Notes:

1. Based on the above, we have some inconcistencies with the **data types** of certain columns. Some columns need to be converted to categorical types and `Date_added` needs to be converted to *DATETIME* format.
2. We can also observe that some columns have missing values.

Null values check

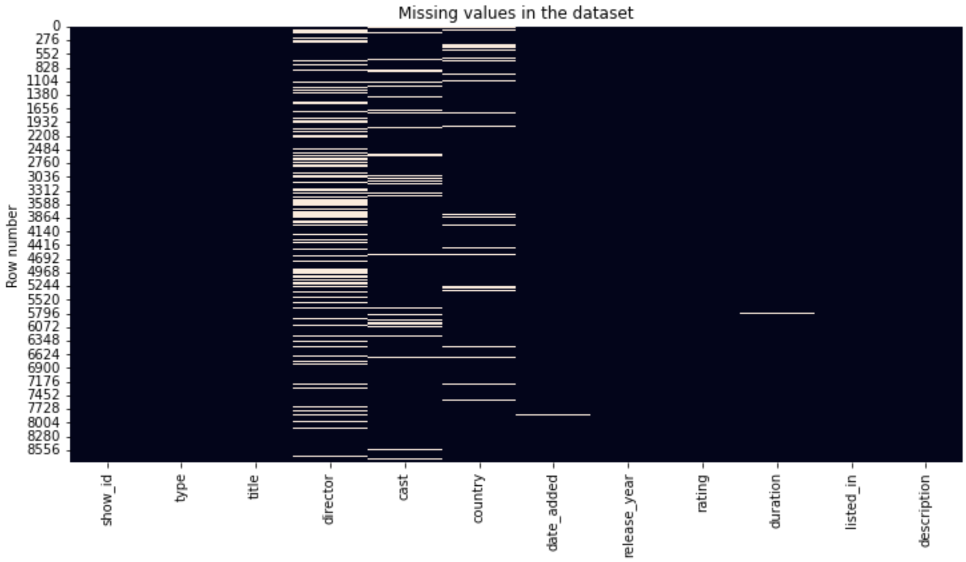
In [5]:

```
#Creating a function to check null values by percent and counts (helpful tofetch by function when we treat missing values)
def nulls(df=df):
    nulls = pd.DataFrame({'count':df.isnull().sum(),'percent':round(df.isnull().sum()*100/len(df),2)})
    plt.figure(figsize=(12,6))
    plt.title('Missing values in the dataset')
    sns.heatmap(df.isnull(), cbar=False)
    plt.ylabel('Row number')
    return nulls[nulls['count']>0]

nulls()
```

Out[5]:

	count	percent
director	2634	29.91
cast	825	9.37
country	831	9.44
date_added	10	0.11
rating	4	0.05
duration	3	0.03



- From the above we can see that we have `6 columns` with missing values. We will decide on the treatement of the same when we explore the data individually by columns.
- Based on the heatmap, there is **no pattern found** with consistent missing values across columns. Each column can be treated individually.

Statistical Summary of data

In [6]:

```
#Year
df.describe(exclude='object').round()
```

Out[6]:

	release_year
count	8807.0
mean	2014.0
std	9.0
min	1925.0
25%	2013.0
50%	2017.0
75%	2019.0
max	2021.0

- Based on the above we have movies from the year *1925* to *2021*
- The **mean** suggests that most movies could be around the year *2014* (to be confirmed during `univariate analysis`)
- The **std** of `9` could mean that the distribution of years might not be widely seperated and significant gap between years might NOT be observed. (Except probably in the first quartile since the range is huge from *1925* to *2013*)

In [7]:

```
#All other categorical columns
df.describe(include='object')
```

Out[7]:

	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	8807	4528	7692	748	1767	17	220	514	8775
top	s1	Movie	Dick Johnson Is Dead	Rajiv Chilaka	David Attenborough	United States	January 1, 2020	TV-MA	1 Season	Dramas, International Movies	Paranormal activity at a lush, abandoned prope...
freq	1	6131	1	19	19	2818	109	3207	1793	362	4

We will have to seperate columns with multiple tags into seperate rows to analyze different entities of the dataset. This will be done while we perform visual analysis.

(eg. `cast` has many people which might cause a grouped analysis)

Changing data types

In [8]:

```
#Changing to date-time format
df['date_added'] = pd.to_datetime(df['date_added'])

#Changing to 'object' format
df['release_year'] = df['release_year'].astype('object')
```

In [9]:

```
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   datetime64[ns]
 7   release_year  8807 non-null   object        
 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), object(11)
memory usage: 825.8+ KB
```

Now that all data types have appropriately been typecasted we will continue with further analysis

Unique values count

In [10]:

```
for i in df:
    print(i+':',df[i].nunique())
```

```
show_id: 8807
type: 2
title: 8807
director: 4528
cast: 7692
country: 748
date_added: 1714
release_year: 74
rating: 17
duration: 220
listed_in: 514
description: 8775
```

- We have 8807 unique `show_id` and `title` which suggests that there is no duplication in the number of movies.
- Further inspection is required to understand other categories. We will try to understand the columns with reasonable categories.

In [11]:

```
# Type of show
pd.DataFrame({'count':df['type'].value_counts(),'percent':round(df['type'].value_counts(normalize=True)*100,2)})
```

Out[11]:

	count	percent
Movie	6131	69.62
TV Show	2676	30.38

We have 70% as movies and 30% as TV shows from the data

In [12]:

```
# Rating of show
pd.DataFrame({'count':df['rating'].value_counts(),'percent':round(df['rating'].value_counts(normalize=True)*100,2)})
```

Out[12]:

	count	percent
TV-MA	3207	36.43
TV-14	2160	24.54
TV-PG	863	9.80
R	799	9.08
PG-13	490	5.57
TV-Y7	334	3.79
TV-Y	307	3.49
PG	287	3.26
TV-G	220	2.50
NR	80	0.91
G	41	0.47
TV-Y7-FV	6	0.07
NC-17	3	0.03
UR	3	0.03
74 min	1	0.01
84 min	1	0.01
66 min	1	0.01

There are some ill-placed categories (giving us duration instead of rating) in `rating` column which will need to be treated

In [13]:

```
#Treating the wrongly placed values
df.loc[df['rating'].isin(['74 min','84 min','66 min']), 'duration'] = df.loc[df['rating'].isin(['74 min','84 min','66 min']),
df.loc[df['rating'].isin(['74 min','84 min','66 min']), 'rating'] = np.nan
pd.DataFrame({'count':df['rating'].value_counts(),'percent':round(df['rating'].value_counts(normalize=True)*100,2)})
```

Out[13]:

	count	percent
TV-MA	3207	36.44
TV-14	2160	24.55
TV-PG	863	9.81
R	799	9.08
PG-13	490	5.57
TV-Y7	334	3.80
TV-Y	307	3.49
PG	287	3.26
TV-G	220	2.50
NR	80	0.91
G	41	0.47
TV-Y7-FV	6	0.07
NC-17	3	0.03
UR	3	0.03

For columns which have multiple nested categories *(eg. cast, director, country, listed_in)* we will treat and analyze these during visual analysis.

Example below

In [14]:

```
# Rating of show
pd.DataFrame({'count':df['cast'].value_counts(),'percent':round(df['cast'].value_counts(normalize=True)*100,2)})
```

Out[14]:

	count	percent
David Attenborough	19	0.24
Vatsal Dubey, Julie Tejwani, Rupa Bhimani, Jigna Bhardwaj, Rajesh Kava, Mousam, Swapnil	14	0.18
Samuel West	10	0.13
Jeff Dunham	7	0.09
David Spade, London Hughes, Fortune Feimster	6	0.08
...	...	...
Michael Peña, Diego Luna, Tenoch Huerta, Joaquin Cosio, José María Yazpik, Matt Letscher, Alyssa Diaz	1	0.01
Nick Lachey, Vanessa Lachey	1	0.01
Takeru Sato, Kasumi Arimura, Haru, Kentaro Sakaguchi, Takayuki Yamada, Kendo Kobayashi, Ken Yasuda, Arata Furuta, Suzuki Matsuo, Koichi Yamadera, Arata Iura, Chikako Kaku, Kotaro Yoshida	1	0.01
Toyin Abraham, Sambasa Nzeribe, Chioma Chukwuka Akpotha, Chioma Omeruah, Chiwetalu Agu, Dele Odule, Femi Adebayo, Bayray McNwizu, Biodun Stephen	1	0.01
Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanana, Manish Chaudhary, Meghna Malik, Malkeet Rauni, Anita Shabdish, Chittaranjan Tripathy	1	0.01

7692 rows × 2 columns

Exploratory Data Analysis

Creating functions to ease-up further analysis steps

In [15]:

```
#Function to fetch value counts of columns and percentage parallely
def values_col(data):
    return pd.DataFrame({'count':data.value_counts(),'percent':round(data.value_counts(normalize=True)*100,2)})
```

In [16]:

```
#Creating function for converting to list
def make_list(x):
    if ',' in str(x):
        return x.split(', ')
    else:
        return x

#Creating a function to unnest columns with nested values
def unnest(col, df):
    return df[col].apply(make_list).explode().reset_index(drop=True)
```

In [17]:

```
#Testing above
unnest('cast', df)
```

Out[17]:

```
0                         NaN
1                  Ama Qamata
2                 Khosi Ngema
3               Gail Mabalane
4              Thabang Molaba
                  ...
64946         Manish Chaudhary
64947             Meghna Malik
64948            Malkeet Rauni
64949           Anita Shabdish
64950    Chittaranjan Tripathy
Name: cast, Length: 64951, dtype: object
```

Now that we have created a function which allows us to easily unnest, we will employ this to visualize and understand values in each column better.

We will combine univariate and bivariate analysis (with respect to type of the show) as and where it's feasible and would possibly get interesting insights.

`show_id`

In [18]:

```
df['show_id'].describe()
```

Out[18]:

```
count      8807
unique     8807
top          s1
freq          1
Name: show_id, dtype: object
```

We have **8807 unique show IDs**. This suggests that no shows are being duplicated and we have all unique records.

`title`

In [19]:

```
df['title'].describe()
```

Out[19]:

```
count                      8807
unique                     8807
top       Dick Johnson Is Dead
freq                          1
Name: title, dtype: object
```

We have **8807 unique titles**. This just confirms our previous observation with `show_id` that we have all unique titles in our data.

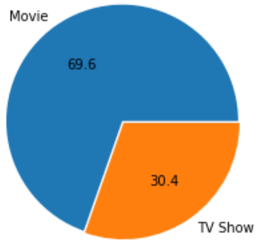
`type`

In [20]:

```
plt.pie(df['type'].value_counts(), labels=['Movie', 'TV Show'], explode=[0.02,0], autopct='%.1f')
values_col(df['type'])
```

Out[20]:

	count	percent
Movie	6131	69.62
TV Show	2676	30.38



Our data is divided into 2 categories essentially: **Movies** *(70%)* & **TV Shows** *(30%)*

We will divide our data seperately for a more thorough analysis since both segments need to be considered as seperate entitites.

In [21]:

```
movies = df[df['type']=='Movie']
tv = df[df['type']=='TV Show']
```

In [22]:

```
movies.shape, tv.shape
```

Out[22]:

```
((6131, 12), (2676, 12))
```

director

In [23]:

```
values_col(movies['director'])
```

Out[23]:

	count	percent
Rajiv Chilaka	19	0.32
Raúl Campos, Jan Suter	18	0.30
Suhas Kadav	16	0.27
Marcus Raboy	15	0.25
Jay Karas	14	0.24
...	...	...
Dennis Rovira van Boekholt	1	0.02
Naoto Amazutsumi	1	0.02
Jenny Gage	1	0.02
Kaila York	1	0.02
Mozez Singh	1	0.02

4354 rows × 2 columns

From the above, we can see that we need to unnest director column to identify the actual counts of directors.

This observation is based on the records of `Raúl Campos, Jan Suter`

In [24]:

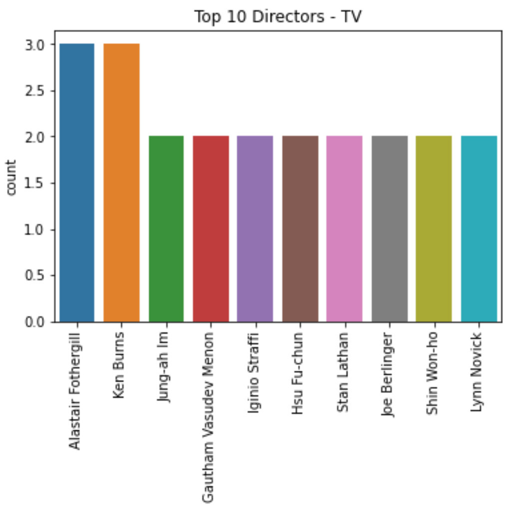
```
#tv
tv_directors = unnest('director', tv)

#plotting top 10
sns.barplot(x=values_col(tv_directors).index[:10], y=values_col(tv_directors)['count'][:10])
plt.xticks(rotation=90)
plt.title('Top 10 Directors - TV')
values_col(tv_directors)
```

Out[24]:

	count	percent
Alastair Fothergill	3	0.96
Ken Burns	3	0.96
Jung-ah Im	2	0.64
Gautham Vasudev Menon	2	0.64
Iginio Straffi	2	0.64
...	...	...
Jesse Vile	1	0.32
Ellena Wood	1	0.32
Picky Talarico	1	0.32
Pedro Waddington	1	0.32
Michael Cumming	1	0.32

299 rows × 2 columns



In [25]:

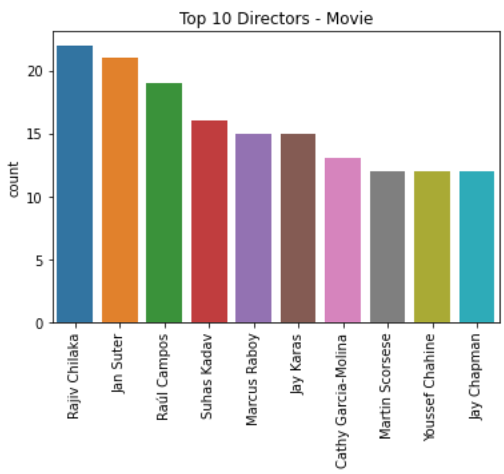
```
#movie
mv_directors = unnest('director', movies)

#plotting top 10
sns.barplot(x=values_col(mv_directors).index[:10], y=values_col(mv_directors)['count'][:10])
plt.xticks(rotation=90)
plt.title('Top 10 Directors - Movie')
values_col(mv_directors)
```

Out[25]:

	count	percent
Rajiv Chilaka	22	0.33
Jan Suter	21	0.32
Raúl Campos	19	0.29
Suhas Kadav	16	0.24
Marcus Raboy	15	0.23
...	...	...
Vrinda Samartha	1	0.02
Nicholaus Goossen	1	0.02
Stig Bergqvist	1	0.02
Paul Demeyer	1	0.02
Mozez Singh	1	0.02

4777 rows × 2 columns



Notes:

- Based on the ratios of moves vs. TV shows, there seems to be NO abnormality in the no. of directors whos Titles have been featured on Netflix
- We can do further analysis `country-wise` to get insights on director based releases.
- The top 5 directors are:

In [26]:

```
for n,i in enumerate(values_col(df['director']).index[:5]):
    print(n+1,')',i)
```

```
1 ) Rajiv Chilaka
2 ) Raúl Campos, Jan Suter
3 ) Marcus Raboy
4 ) Suhas Kadav
5 ) Jay Karas
```

cast

In [27]:

```
values_col(df['cast'])
```

Out[27]:

	count	percent
David Attenborough	19	0.24
Vatsal Dubey, Julie Tejwani, Rupa Bhimani, Jigna Bhardwaj, Rajesh Kava, Mousam, Swapnil	14	0.18
Samuel West	10	0.13
Jeff Dunham	7	0.09
David Spade, London Hughes, Fortune Feimster	6	0.08
...	...	...
Michael Peña, Diego Luna, Tenoch Huerta, Joaquin Cosio, José María Yazpik, Matt Letscher, Alyssa Diaz	1	0.01
Nick Lachey, Vanessa Lachey	1	0.01
Takeru Sato, Kasumi Arimura, Haru, Kentaro Sakaguchi, Takayuki Yamada, Kendo Kobayashi, Ken Yasuda, Arata Furuta, Suzuki Matsuo, Koichi Yamadera, Arata Iura, Chikako Kaku, Kotaro Yoshida	1	0.01
Toyin Abraham, Sambasa Nzeribe, Chioma Chukwuka Akpotha, Chioma Omeruah, Chiwetalu Agu, Dele Odule, Femi Adebayo, Bayray McNwizu, Biodun Stephen	1	0.01
Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanana, Manish Chaudhary, Meghna Malik, Malkeet Rauni, Anita Shabdish, Chittaranjan Tripathy	1	0.01

7692 rows × 2 columns

- We will have to unnest the column and analyze based on different types of shows.

In [28]:

```
#top 10 cast
values_col(unnest('cast', df))[:10]
```

Out[28]:

	count	percent
Anupam Kher	43	0.07
Shah Rukh Khan	35	0.05
Julie Tejwani	33	0.05
Naseeruddin Shah	32	0.05
Takahiro Sakurai	32	0.05
Rupa Bhimani	31	0.05
Akshay Kumar	30	0.05
Om Puri	30	0.05
Yuki Kaji	29	0.05
Paresh Rawal	28	0.04

In [29]:

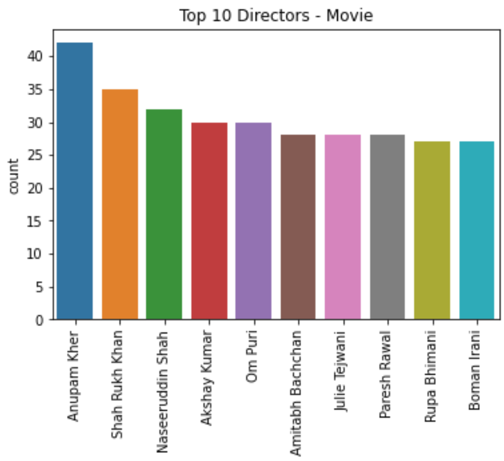
```
mv_cast = unnest('cast', movies)

#plotting top 10
sns.barplot(x=values_col(mv_cast).index[:10], y=values_col(mv_cast)['count'][:10])
plt.xticks(rotation=90)
plt.title('Top 10 Directors - Movie')
values_col(mv_cast)
```

Out[29]:

	count	percent
Anupam Kher	42	0.09
Shah Rukh Khan	35	0.08
Naseeruddin Shah	32	0.07
Akshay Kumar	30	0.07
Om Puri	30	0.07
...	...	...
Sushma Bakshi	1	0.00
Yusuf Hussain	1	0.00
Amarjeet Amle	1	0.00
Priya	1	0.00
Chittaranjan Tripathy	1	0.00

25951 rows × 2 columns



In [30]:

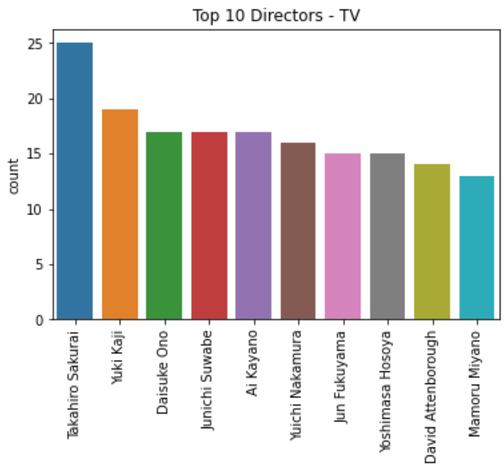
```
tv_cast = unnest('cast', tv)

#plotting top 10
sns.barplot(x=values_col(tv_cast).index[:10], y=values_col(tv_cast)['count'][:10])
plt.xticks(rotation=90)
plt.title('Top 10 Directors - TV')
values_col(tv_cast)
```

Out[30]:

	count	percent
Takahiro Sakurai	25	0.13
Yuki Kaji	19	0.10
Daisuke Ono	17	0.09
Junichi Suwabe	17	0.09
Ai Kayano	17	0.09
...	...	...
Bhumibhat Thavornsiri	1	0.01
Thanongsak Suphakan	1	0.01
Kanjanaporn Plodpai	1	0.01
Boonsong Nakphoo	1	0.01
Hina Khawaja Bayat	1	0.01

14863 rows × 2 columns



Based on the TOP casts:

- The top 10 cast in movies seem to be from *Bollywood Movies*
- While top 10 cast in TV seems to be from *Asian TV (Anime, KDrama, etc.)*

Overall top 10 cast suggests that there are many bollywood affiliated cast titles.

country

In [32]:

```
values_col(df['country'])
```

Out[32]:

	count	percent
United States	2818	35.33
India	972	12.19
United Kingdom	419	5.25
Japan	245	3.07
South Korea	199	2.49
...	...	...
Romania, Bulgaria, Hungary	1	0.01
Uruguay, Guatemala	1	0.01
France, Senegal, Belgium	1	0.01
Mexico, United States, Spain, Colombia	1	0.01
United Arab Emirates, Jordan	1	0.01

748 rows × 2 columns

From the above, we need to unnest this column as well

In [33]:

```
#top 10 countries
values_col(unnest('country', df))[:10]
```

Out[33]:

	count	percent
United States	3689	36.84
India	1046	10.45
United Kingdom	804	8.03
Canada	445	4.44
France	393	3.92
Japan	318	3.18
Spain	232	2.32
South Korea	231	2.31
Germany	226	2.26
Mexico	169	1.69

In [34]:

```
# No. of unique countries
unnest('country', df).unique().size
```

Out[34]:

```
128
```

In [35]:

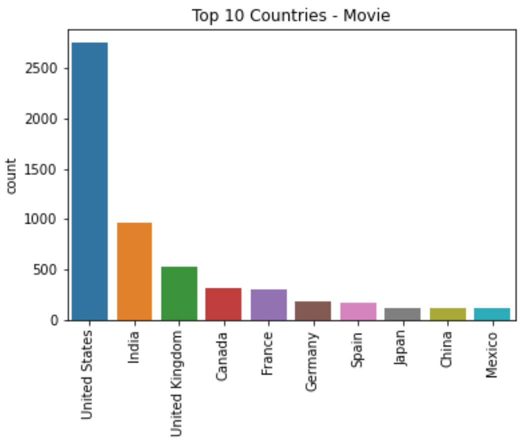
```
mv_country = unnest('country', movies)

#plotting top 10
sns.barplot(x=values_col(mv_country).index[:10], y=values_col(mv_country)['count'][:10])
plt.xticks(rotation=90)
plt.title('Top 10 Countries - Movie')
values_col(mv_country)
```

Out[35]:

	count	percent
United States	2751	37.31
India	962	13.05
United Kingdom	532	7.21
Canada	319	4.33
France	303	4.11
...	...	...
Bermuda	1	0.01
Angola	1	0.01
Armenia	1	0.01
Mongolia	1	0.01
Montenegro	1	0.01

122 rows × 2 columns



In [36]:

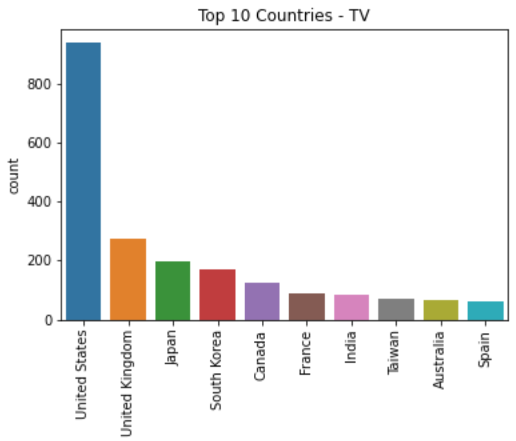
```
tv_country = unnest('country', tv)

#plotting top 10
sns.barplot(x=values_col(tv_country).index[:10], y=values_col(tv_country)['count'][:10])
plt.xticks(rotation=90)
plt.title('Top 10 Countries - TV')
values_col(tv_country)
```

Out[36]:

	count	percent
United States	938	35.53
United Kingdom	272	10.30
Japan	199	7.54
South Korea	170	6.44
Canada	126	4.77
...	...	...
Malta	1	0.04
Belarus	1	0.04
United Arab Emirates	1	0.04
Uruguay	1	0.04
Switzerland	1	0.04

66 rows × 2 columns



Notes:

- Our data has a total of **128 countries.**
- **United States** is leading by beng the most producing country in both the categories by heavily drastic numbers.
- **USA**, **UK** & **Canada** are leading in both segments by being in the top 5.

date_added

In [37]:

```
date_counts = df['date_added'].value_counts().reset_index()
date_counts
```

Out[37]:

	index	date_added
0	2020-01-01	110
1	2019-11-01	91
2	2018-03-01	75
3	2019-12-31	74
4	2018-10-01	71
...	...	...
1709	2017-02-21	1
1710	2017-02-07	1
1711	2017-01-29	1
1712	2017-01-25	1
1713	2020-01-11	1

1714 rows × 2 columns

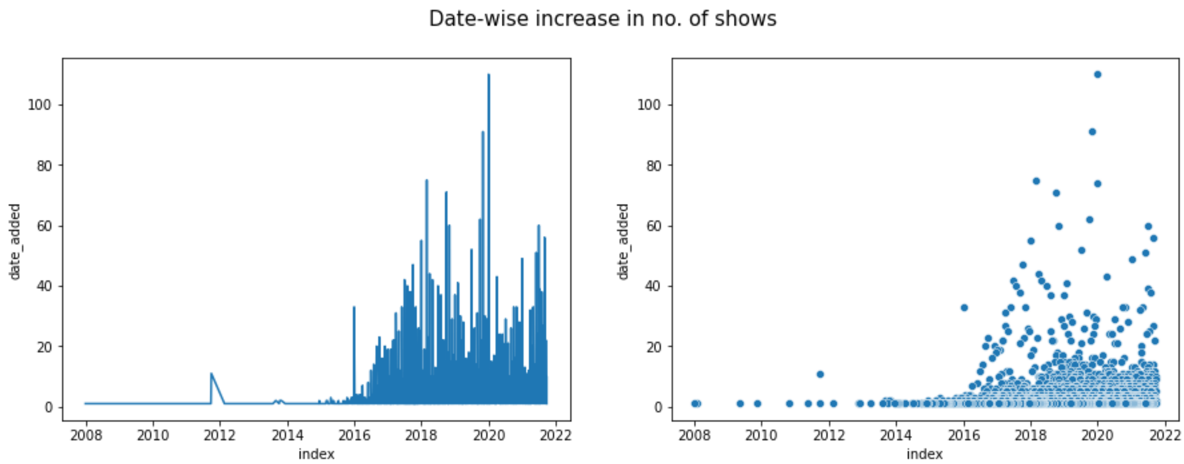
In [38]:

```
plt.figure(figsize=(15,5))

plt.subplot(1,2,1)
sns.lineplot(x = date_counts['index'], y=date_counts['date_added'])

plt.subplot(1,2,2)
sns.scatterplot(x = date_counts['index'], y=date_counts['date_added'])

plt.suptitle('Date-wise increase in no. of shows', fontsize=15)
plt.show()
```



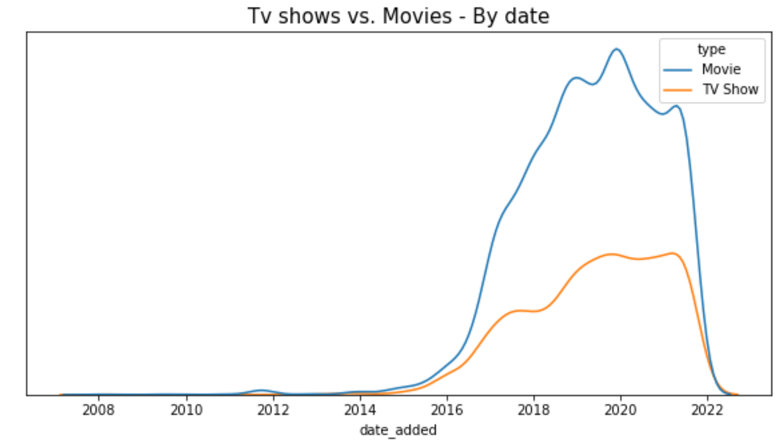
- The increase in no. of titles has increased drastically starting from 2014 onwards based on our data
- **2020** seems to have the highest no. of shows added.

In [128]:

```
plt.figure(figsize=(22,5))

plt.subplot(1,2,1)
sns.kdeplot(x = df['date_added'], hue = df['type'])
plt.ylabel(' ')
plt.yticks([])

plt.title('Tv shows vs. Movies - By date', fontsize=15)
plt.show()
```



- An increase in both segments is visible starting from 2015 oonwards
- The increase in counts of movies is significantly higher than tv shows based on our data
- Both segments seemed to peak at their respective max values at 2020

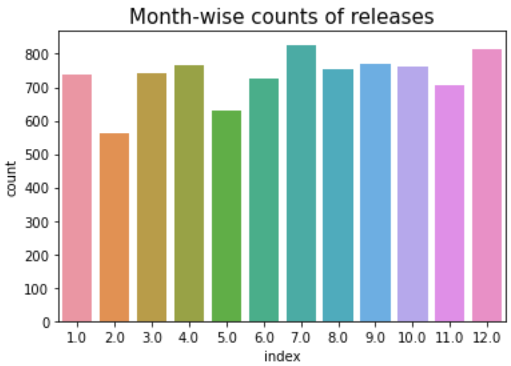
In [40]:

```
#Checking for month-wise increase
months = values_col(df['date_added'].dt.month)['count'].reset_index()
months = months.sort_values('index').reset_index(drop=True)

sns.barplot(x='index', y='count', data=months)
plt.title('Month-wise counts of releases', fontsize=15)
months
```

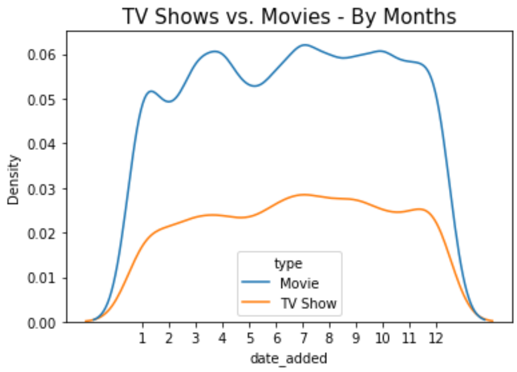
Out[40]:

	index	count
0	1.0	738
1	2.0	563
2	3.0	742
3	4.0	764
4	5.0	632
5	6.0	728
6	7.0	827
7	8.0	755
8	9.0	770
9	10.0	760
10	11.0	705
11	12.0	813



In [41]:

```
sns.kdeplot(x = df['date_added'].dt.month, hue=df['type'])
plt.xticks(np.arange(1,13))
plt.title('TV Shows vs. Movies - By Months', fontsize=15)
plt.show()
```



- The growth in movies and tv shows added to netflix seemed to be almost parallely increasing (or decreasing)
- The most shows were added in 3 sets of months:
  - December & January
  - July
  - March & April
- Least shows were added in February and May

release_year

In [42]:

```
release_date_counts = df['release_year'].value_counts().reset_index()
release_date_counts
```

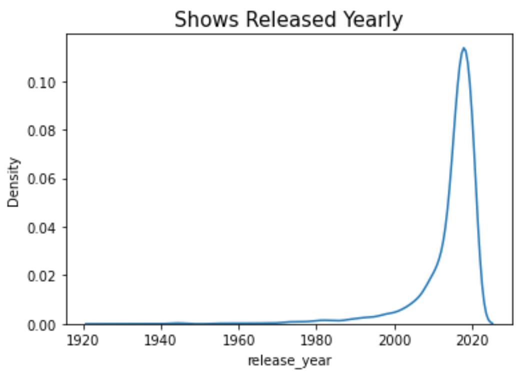
Out[42]:

	index	release_year
0	2018	1147
1	2017	1032
2	2019	1030
3	2020	953
4	2016	902
...	...	...
69	1959	1
70	1925	1
71	1961	1
72	1947	1
73	1966	1

74 rows × 2 columns

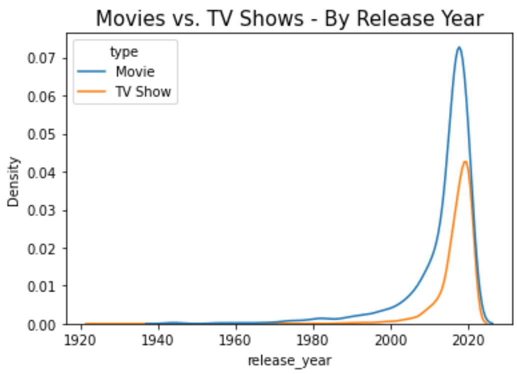
In [43]:

```
sns.kdeplot('release_year', data=df,)
plt.title('Shows Released Yearly', fontsize=15)
plt.show()
```



In [44]:

```
sns.kdeplot('release_year', data=df, hue='type')
plt.title('Movies vs. TV Shows - By Release Year', fontsize=15)
plt.show()
```



- Most shows were released after 2016
- This pattern is consistent across both segments - tv shows and movies.

`rating`

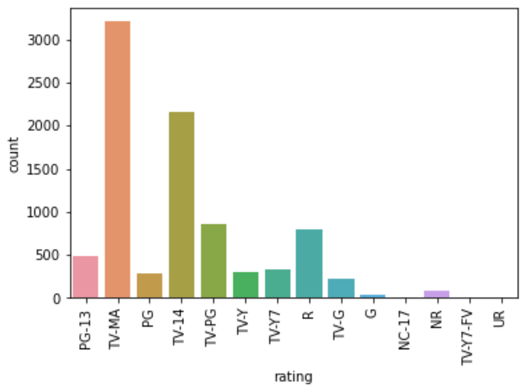
In [57]:

```
sns.countplot(df['rating'])
plt.xticks(rotation=90)
values_col(df['rating'])
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable
as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other argu
ments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[57]:

	count	percent
TV-MA	3207	36.44
TV-14	2160	24.55
TV-PG	863	9.81
R	799	9.08
PG-13	490	5.57
TV-Y7	334	3.80
TV-Y	307	3.49
PG	287	3.26
TV-G	220	2.50
NR	80	0.91
G	41	0.47
TV-Y7-FV	6	0.07
NC-17	3	0.03
UR	3	0.03



Notes:

- `TV-MA` and `TV-14` contribute to >50% of the data

- Some categories have extremely low counts, such as: `NR` , `G` , `TV-Y7-FV` , `NC-17` , `UR`

`duration`

In [58]:

```
df['duration']
```

Out[58]:

```
0          90 min
1       2 Seasons
2        1 Season
3        1 Season
4       2 Seasons
          ...
8802      158 min
8803    2 Seasons
8804       88 min
8805       88 min
8806      111 min
Name: duration, Length: 8807, dtype: object
```

This column consists different duration values for different **type of show**.

We will have to analyze them seperately.

In [61]:

```
# Ensuring the type of 'units' in the column
df['duration'].apply(lambda x: x.split(' ')[1]).unique()
```

Out[61]:

```
array(['min', 'Seasons', 'Season'], dtype=object)
```

There are 3 types of units:

- **Movie** : *min*
- **TV Show** : *Season* , *Seasons*

We will take this into consideriation when analyzing them.

In [64]:

```
#Ensuring movie dataset's column has only "min"

movies['duration'].apply(lambda x: x.split(' ')[1]).unique()
```

Out[64]:

```
array(['min'], dtype=object)
```

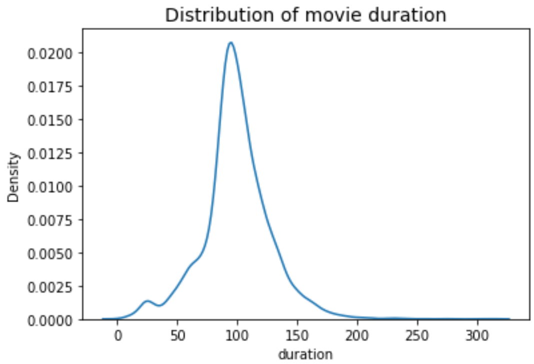
Looks good, lets visualize this.

In [89]:

```
sns.kdeplot(movies['duration'].apply(lambda x: int(x.split(' ')[0])))
plt.title('Distribution of movie duration', fontsize=14)
(movies['duration'].apply(lambda x: int(x.split(' ')[0]))).describe(percentiles=[0.025,0.05,0.1,0.25,0.5,0.75,0.9,0.95,0.97])
```

Out[89]:

```
count    6131.000000
mean       99.564998
std        28.289504
min         3.000000
2.5%       35.000000
5%         52.000000
10%        65.000000
25%        87.000000
50%        98.000000
75%       114.000000
90%       133.000000
95%       147.000000
97%       156.000000
max       312.000000
Name: duration, dtype: float64
```



Notes (movies):

- We can easily spot a **significant no. of outliers** with higher (and lower) duration of movies
  - *Specifically, movies >180 (**less than 3 percentile**) mins seem to fall under outliers*
  - There seems to be a good no. of movies which are <30 mins as well (***less than 2.5 percentile***). These could very well be short films treated as part of the wider movies section.
- Average duration of movies is 98% (based on the median)

In [74]:

```
#Ensuring tv dataset's column has only "Season"/"Seasons"

tv['duration'].apply(lambda x: x.split(' ')[1]).unique()
```

Out[74]:

```
array(['Seasons', 'Season'], dtype=object)
```

In [83]:

```
pd.DataFrame({'prefix':tv['duration'].apply(lambda x: x.split(' ')[1]),
              'no. of seasons':tv['duration'].apply(lambda x: x.split(' ')[0])}).groupby(['prefix','no. of seasons']).count()
```

Out[83]:

prefix	no. of seasons
Season	**1**
Seasons	**10**
	11
	12
	13
	15
	17
	2
	3
	4
	5
	6
	7
	8
	9

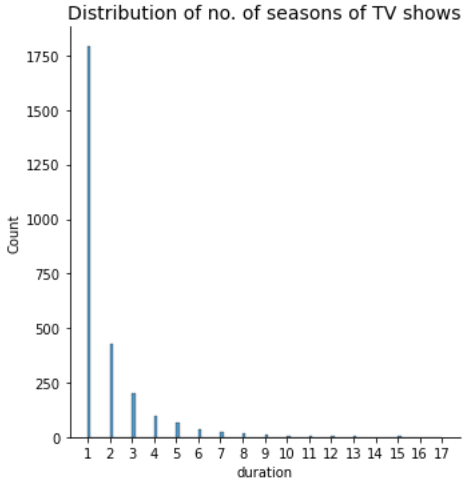
- From the above, it's evident that the technical "terminology" is appropriate.
  - *i.e. "Season" has only 1 season and "Seasons" has multiple*

This means there's no other difference and data quality is fair. This will be treated to extract the no. only and then analyzed.

In [101]:

```
sns.displot(tv['duration'].apply(lambda x: int(x.split(' ')[0])))
plt.title('Distribution of no. of seasons of TV shows', fontsize=14)
plt.xticks(np.arange(1,18))
plt.show()

values_col(tv['duration'].apply(lambda x: int(x.split(' ')[0])))
```



Out[101]:

	count	percent
1	1793	67.00
2	425	15.88
3	199	7.44
4	95	3.55
5	65	2.43
6	33	1.23
7	23	0.86
8	17	0.64
9	9	0.34
10	7	0.26
13	3	0.11
15	2	0.07
12	2	0.07
11	2	0.07
17	1	0.04

Notes (TV shows):

- The no. of seasons is inversely proportional to the count of tv shows with said no. of seasons.
  - _i.e. tv shows with less seasons are higher in number (and vice versa)
- Majority of the data has a **single season (67%)**
- TV shows >8 seasons seem to be very rare

listed_in

In [121]:

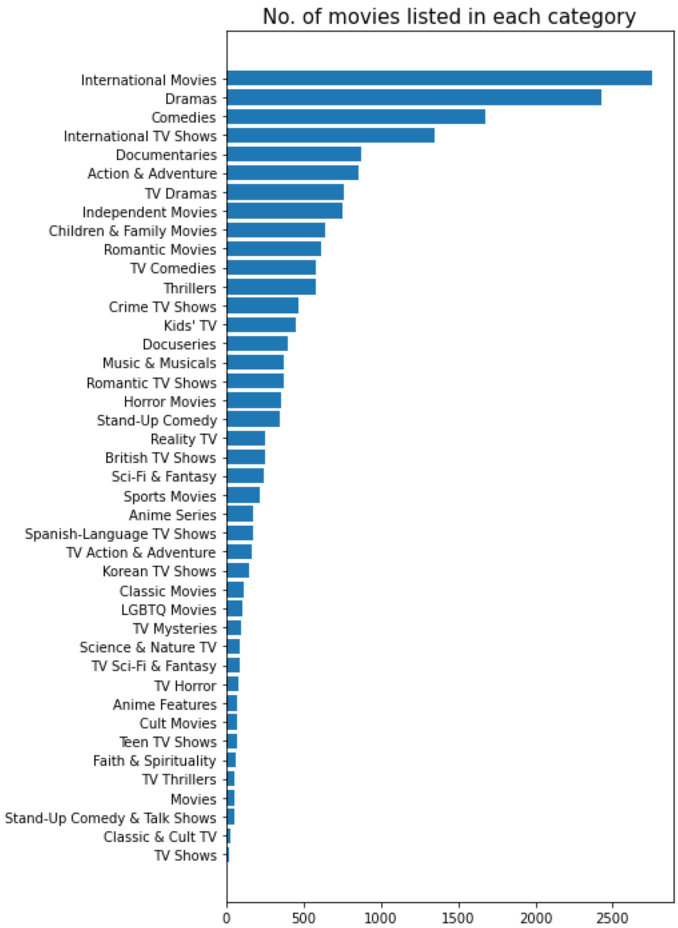
```
#Checing top 5
values_col(unnest('listed_in', df))[:5]
```

Out[121]:

	count	percent
International Movies	2752	14.24
Dramas	2427	12.56
Comedies	1674	8.66
International TV Shows	1351	6.99
Documentaries	869	4.50

In [120]:

```
plt.figure(figsize=(6,12))
plt.barh(values_col(unnest('listed_in', df)).index[::-1], values_col(unnest('listed_in', df))['count'][::-1])
plt.title('No. of movies listed in each category', fontsize=15)
plt.show()
```



Notes:

- **International movies & Dramas** seem to lead the chart by being the set with highest shows.
- We can observe that there are multiple instances where TV shows and movies have the same type of category names,
  - eg: `International Movies` & `Internation TV shows`

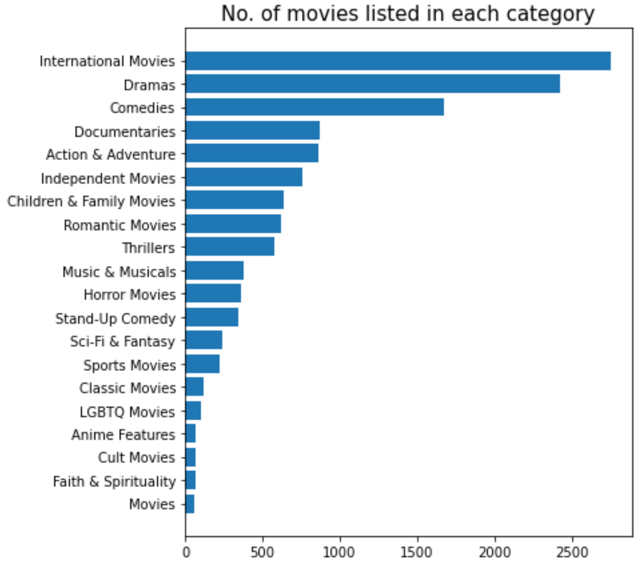
We will seperate the types and visualize futher

In [124]:

```
#Movies

plt.figure(figsize=(6,7))
plt.barh(values_col(unnest('listed_in', movies)).index[::-1], values_col(unnest('listed_in', movies))['count'][::-1])
plt.title('No. of movies listed in each category', fontsize=15)
plt.show()

#Top 5 counts
values_col(unnest('listed_in', movies))[:5]
```



Out[124]:

	count	percent
International Movies	2752	20.86
Dramas	2427	18.40
Comedies	1674	12.69
Documentaries	869	6.59
Action & Adventure	859	6.51

Notes:

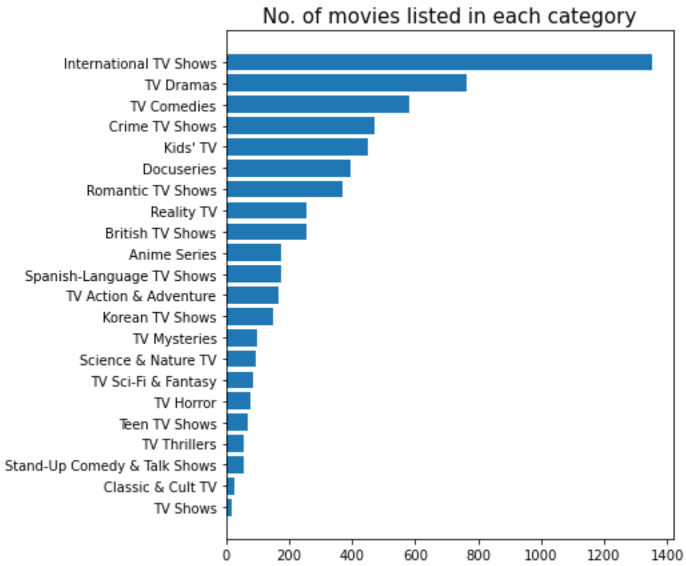
- The top 5 genres are as stated above.
- Top 3 genres seem to take up the highest portion (>50%) of the whole data.
  - i.e. **International movies, Dramas, Comedies**

In [125]:

```
#tv

plt.figure(figsize=(6,7))
plt.barh(values_col(unnest('listed_in', tv)).index[::-1], values_col(unnest('listed_in', tv))['count'][::-1])
plt.title('No. of movies listed in each category', fontsize=15)
plt.show()

#Top 5 counts
values_col(unnest('listed_in', tv))[:5]
```



Out[125]:

	count	percent
International TV Shows	1351	22.03
TV Dramas	763	12.44
TV Comedies	581	9.47
Crime TV Shows	470	7.66
Kids' TV	451	7.35

Notes:

- Top 5 shows are as shown above
- Although, its interesting to note that only **International TV shows** hold avery high % of tv shows (unlike 2 different categories in movies)
- To constitute majority of the data (i.e. ~50%) here we consider first 4 categories (instead of 3 like in movies)

rating by country

In [140]:

```
#Creating a dataframe to expand country-wise rating preferences
val = pd.DataFrame({'rating':df['rating'], 'country':df['country'].apply(make_list)}).explode('country').reset_index(drop=True
val
```

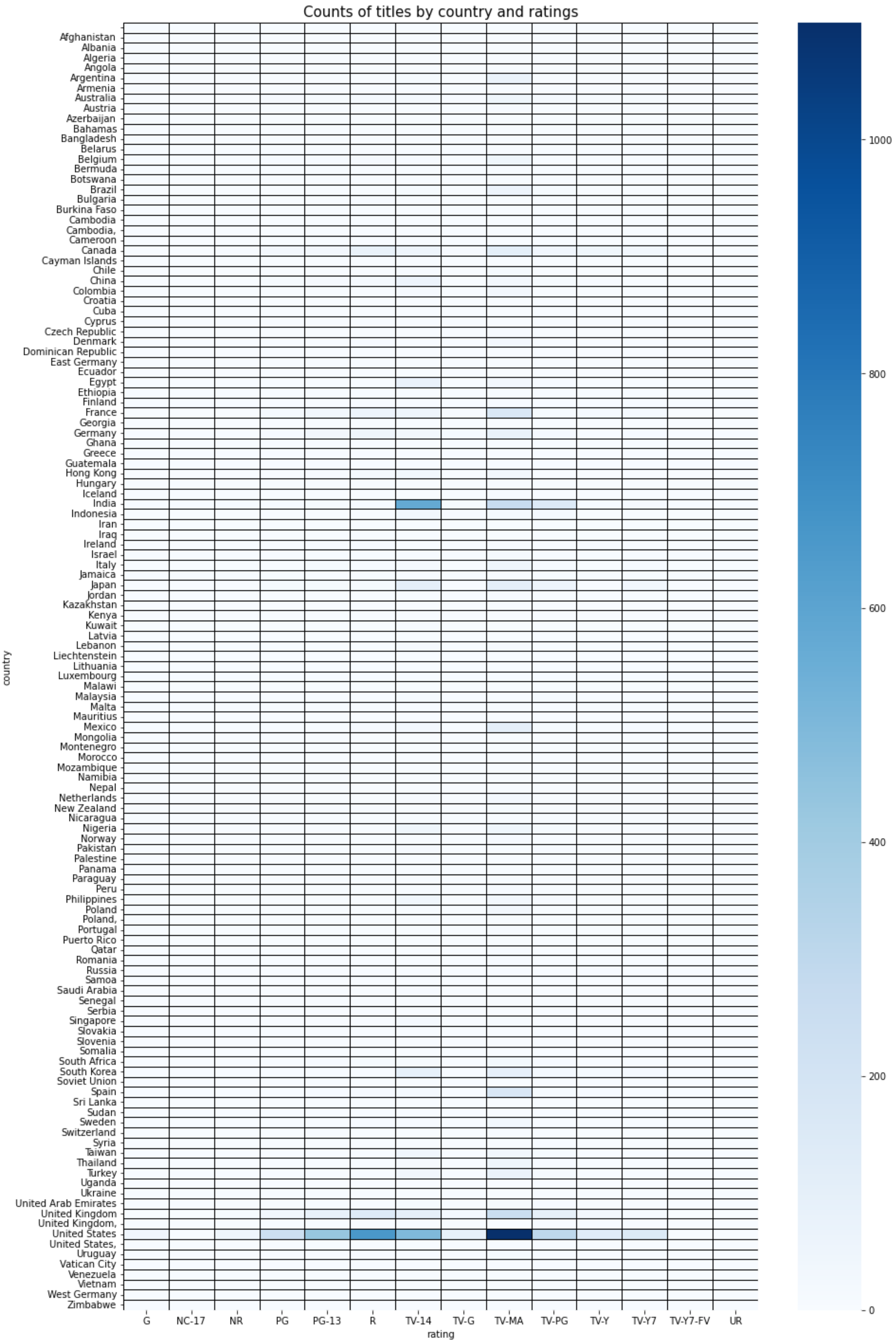
Out[140]:

	rating	country
0	PG-13	United States
1	TV-MA	South Africa
2	TV-MA	NaN
3	TV-MA	NaN
4	TV-MA	India
...	...	...
10840	R	United States
10841	TV-Y7	NaN
10842	R	United States
10843	PG	United States
10844	TV-14	India

10845 rows × 2 columns

In [158]:

```
plt.figure(figsize=(15,25))
sns.heatmap(pd.crosstab(val['country'], val['rating']), cmap='Blues', linecolor='k', linewidth=1)
plt.title('Counts of titles by country and ratings', fontsize=15)
plt.show()
```



Notes:

The most prominent sections seem to be:

- `India :`
  - **TV-14**
  - **TV-MA**
- `Japan :`
  - **TV-14**
  - **TV-MA**
- `France & Spain :`
  - **TV-MA**
- `UK :`
  - **TV-MA**
  - **R**
- `USA :`
  - **TV-MA**
  - **TV-14**
  - **PG** & **PG-13**
  - **R**
  - **TV-PG**

`listed_in by country`

In [161]:

```
#Creating a dataframe to expand country-wise rating preferences
val = pd.DataFrame({'genre':df['listed_in'].apply(make_list),
                    'country':df['country'].apply(make_list)}).explode('country').explode('genre').reset_index(drop=True)
val
```

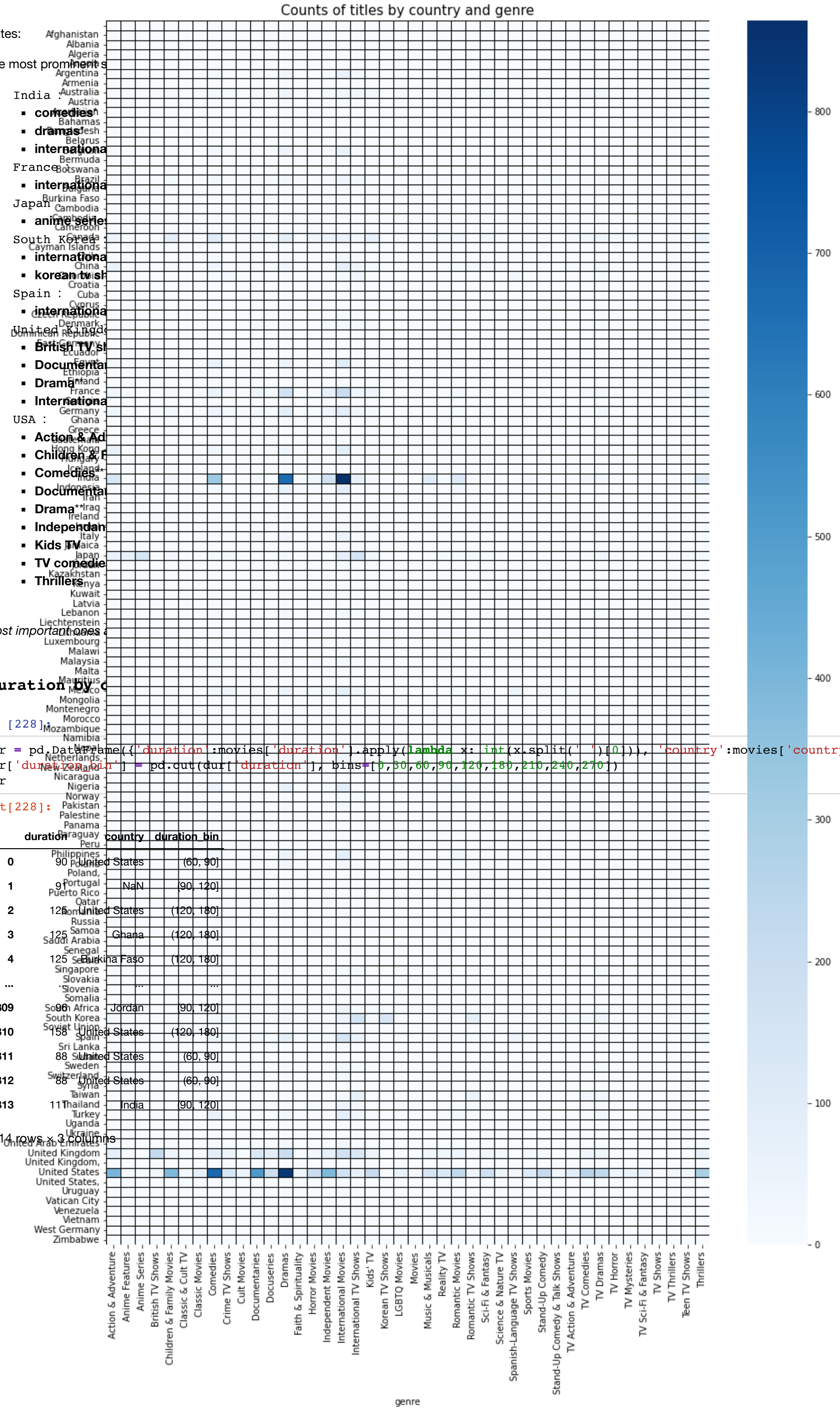
Out[161]:

	genre	country
0	Documentaries	United States
1	International TV Shows	South Africa
2	TV Dramas	South Africa
3	TV Mysteries	South Africa
4	Crime TV Shows	NaN
...	...	...
23749	Children & Family Movies	United States
23750	Comedies	United States
23751	Dramas	India
23752	International Movies	India
23753	Music & Musicals	India

23754 rows × 2 columns

In [231]:

```
plt.figure(figsize=(15,25))
sns.heatmap(pd.crosstab(val['country'], val['genre']), cmap='Blues', linecolor='k', linewidth=1)
plt.title('Counts of titles by country and genre', fontsize=15)
plt.show()
```



Notes:

The most prominent s

- India :
  - comedies
  - dramas
  - internationa
- France
  - internationa
- Japan
  - anime series
- South Korea :
  - internationa
  - korean tv sh
- Spain :
  - internationa
- United Kingdo
  - British TV Sh
  - Documenta
  - Drama**
  - Internationa
- USA :
  - Action & Ad
  - Children & F
  - Comedies**
  - Documenta
  - Drama**
  - Independen
  - Kids TV
  - TV comedie
  - Thrillers

Most important ones a

duration by c

In [228]:

```
dur = pd.DataFrame({'duration':movies['duration'].apply(lambda x: int(x.split(' ')[0])), 'country':movies['country'].apply(make
dur['duration_bin'] = pd.cut(dur['duration'], bins=[0,30,60,90,120,180,210,240,270])
dur
```

Out[228]:

	duration	country	duration_bin
0	90	United States	(60, 90]
1	91	NaN	(90, 120]
2	125	United States	(120, 180]
3	125	Ghana	(120, 180]
4	125	Burkina Faso	(120, 180]
...	...	...	...
7809	96	Jordan	(90, 120]
7810	158	United States	(120, 180]
7811	88	United States	(60, 90]
7812	88	United States	(60, 90]
7813	111	India	(90, 120]

7814 rows × 3 columns

In [229]:

```
temp = dur.groupby(['country','duration_bin']).count()
temp.reset_index().sort_values('duration', ascending=False)[:15]
```

Out[229]:

	country	duration_bin	duration
915	United States	(90, 120]	1358
914	United States	(60, 90]	859
348	India	(120, 180]	556
347	India	(90, 120]	323
899	United Kingdom	(90, 120]	283
916	United States	(120, 180]	270
913	United States	(30, 60]	187
275	France	(90, 120]	187
163	Canada	(90, 120]	158
162	Canada	(60, 90]	114
898	United Kingdom	(60, 90]	114
803	Spain	(90, 120]	112
291	Germany	(90, 120]	103
187	China	(90, 120]	82
900	United Kingdom	(120, 180]	81

With the above we get a good understanding of most preferred **duration of movies** and are popular across countries:

- `USA` : 90-120 mins, 60-90mins, 120-180mins
- `India` : 120-180mins, 90-120mins
- `UK` : 90-120mins

This goes to suggest that on a global scale, the most preferred duration for movies is **`90-120 mins`**

Business Insights & Suggestions

Based on all the observations mentioned after each analytical piece, we can come to a conclusion of a few steps that can be taken proactively and could be considered as benchmark for selection & curation of future content on **Netflix**

These suggestions are:

1. To select the content it would be important to consider the most popular **genres** across both the types of shows(TV/Movies), which are --
   - **`Drama, Comedy & International TV/Movies`**
2. Most preferred duration for movies is **`90-120 minutes`** and for TV shows it would be **`<4 Seasons`**
   - *Although movies **upto 180mins** and **atleast 60 mins** can as well be considered since it has a significant popularity as well.*
3. **`TV-MA`** or Content with ***Mature Rating*** is gloablly popular. This type of content can be consumed by age groups of 17+. Such content curation can be further increased upon by considering other suggestions parallely.
   - For *`USA`* considering increasing content for Kids & Teens can be considered since those *Genres* and *Ratings* seem to be pretty popular.
4. **Anime Shows** is quite well prefrred by consuemrs in Japan, this can further be focussed upon, similarly **Indian Movies and shows** for indian population.
   - These 2 countries seem to hold a lot of popularity and potential consumers hence focusing on this segment can improve customer experience drastically.
5. Some **cast and directors** who are extremely popular are also potentially preferred for content selection.

```
1. `Popular Cast`:
   - Anupam Kher
   - Shah Rukh Khan
   - Naseeruddin Shah

2. `Popular Directors`:
   - Rajiv Chilaka
   - Raúl Campos
   - Jan Suter
   - Suhas Kadav
```