About NETFLIX

Netflix is one of the most popular media and video streaming platforms in the world. As of 2021, Netflix has over 222 million subscribers worldwide. The company offers a wide variety of content, including movies, TV shows, documentaries, and stand-up comedy specials. Netflix is available in over 190 countries and territories. 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.

Netflix is a major player in the media and video streaming industry. The company is well-positioned for continued growth in the years to come.

Business Problem

Analyze the Netflix dataset to identify trends in viewership and demographics, which can help Netflix make decisions about which type of shows/movies to produce and how to grow the business in different countries.

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

These are the basic imports will allow us to use the NumPy, Pandas, Matplotlib, and Seaborn libraries in Python.

NumPy is a library for scientific computing in Python. It provides a high-performance multidimensional array object, along with tools for working with arrays.

Pandas is a library for data analysis in Python. It provides a high-level interface for working with dataframes, which are data structures that are similar to spreadsheets.

Matplotlib is a library for plotting data in Python. It provides a wide range of plotting options, including line plots, bar charts, and pie charts.

Seaborn is a library for statistical plotting in Python. It builds on Matplotlib and provides a number of high-level plotting functions that make it easy to create beautiful and informative plots.

```
df = pd.read_csv('netflix.csv', index_col = 'show_id')
```

This code will read the netflix.csv file and create a Pandas DataFrame. The index_col parameter specifies that the show_id column should be used as the index of the DataFrame.

The netflix.csv file contains data on Netflix shows, including the show's title, release date, Listed_in, and rating ,etc.

```
df.head(5)
```

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
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...
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...
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...
s4	TV Show	Jailbirds New	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go

```
df.shape
```

```
(8807, 11)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8807 entries, s1 to s8807
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
```

```
 0   type          8807 non-null   object
 1   title         8807 non-null   object
 2   director      6173 non-null   object
 3   cast          7982 non-null   object
 4   country       7976 non-null   object
 5   date_added    8797 non-null   object
 6   release_year  8807 non-null   int64
 7   rating        8803 non-null   object
 8   duration      8804 non-null   object
 9   listed_in     8807 non-null   object
 10  description   8807 non-null   object
dtypes: int64(1), object(10)
memory usage: 825.7+ KB
```

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

```
United States     2818
India              972
United Kingdom     419
Japan              245
South Korea        199
Name: country, dtype: int64
```

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

```
show_id
s1     Dick Johnson Is Dead
s2             Blood & Water
s3                Ganglands
s4    Jailbirds New Orleans
s5              Kota Factory
Name: title, dtype: object
```

```
# Top 10 directors
df.director.value_counts().head(8)
```

```
Rajiv Chilaka             19
Raúl Campos, Jan Suter    18
Marcus Raboy              16
Suhas Kadav               16
Jay Karas                 14
Cathy Garcia-Molina       13
Martin Scorsese           12
Youssef Chahine           12
Name: director, dtype: int64
```

```
df['listed_in'].nunique()
```

```
514
```

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

```
2
```

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

```
Movie      6131
TV Show    2676
Name: type, dtype: int64
```

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

```
listed_in           country
Action & Adventure  United States                                64
                    United Kingdom, United States                 8
                    United Kingdom                                7
                    Canada                                        5
                    United States, Canada                         3
                                                                 ..
Thrillers           United States, Hong Kong                      1
                    United States, India, France                  1
                    United States, Italy                          1
                    United States, Malta, France, United Kingdom  1
                    United States, Spain                          1
Name: country, Length: 2644, dtype: int64
```

Handling missing values

```
null_values = df.isna().sum().sort_values(ascending = False)
null_values
```

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

```
percentage_of_null_values = round(df.isna().sum() / df.shape[0] * 100 , 2).sort_values(ascending = False)
percentage_of_null_values
```

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

The percentage of null values in 'director', 'country' and 'cast' columns are significantly greater than the null values in other columns. Therefore, we are going to replace null values with "Unknown" for these three columns only and drop the rows with null values of other columns as it is not going to affect the analysis of this large dataset.

```
df['director'].fillna('Unknown Director', inplace = True)
df['cast'].fillna('Unknown Cast', inplace = True)
df['country'].fillna('Unknown Country', inplace = True)
```

```
# Dropping rows with null values present in columns : 'date_added' (0.11 % null values), 'rating' (0.05 % null values) and 'duration' (0.
df.dropna(subset = ['date_added', 'rating', 'duration'], axis = 0, inplace = True)
```

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

```
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64
```

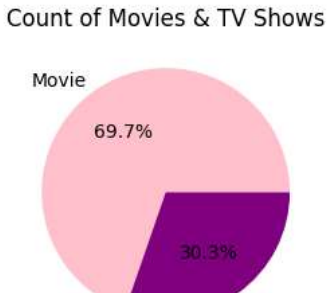
```
df.shape
```

```
(8790, 11)
```

All the null values have been treated !

Movies and TV shows

```
show_type = df['type'].value_counts().reset_index()
show_type.columns = ['type', 'count']

plt.figure(figsize=(4,3))
plt.pie(show_type['count'], labels=show_type['type'], autopct='%1.1f%%', colors = ['pink','purple'])
plt.title('Count of Movies & TV Shows')

plt.show()
```




```
type_movie = df.groupby('type').get_group('Movie')
type_movie.head(2)
```

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	Unknown Cast	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
s7	Movie	My Little Pony: A New	Robert Cullen, José Luis	Vanessa Hudgens, Kimiko Glenn, James	Unknown Country	September 24, 2021	2021	PG	91 min	Children & Family Movies	Equestria's divided. But a bright-eyed

```
type_tvshow = df.groupby('type').get_group('TV Show')
type_tvshow.head(2)
```

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
s2	TV Show	Blood & Water	Unknown Director	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...
s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas	Unknown Country	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International	To protect his family from a powerful drug

```
# As we can see in the 'duration' column, the datatype is object, but to operate some functions on it, we should convert it to integer.


type_movie['duration'] = type_movie['duration'].apply(lambda x : x.replace(' min', '') if 'min' in x else x).astype(int)
type_movie.head(2)
```

```
<ipython-input-23-0c73d77f202f>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  type_movie['duration'] = type_movie['duration'].apply(lambda x : x.replace(' min', '') if 'min' in x else x).astype(int)
```

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	Unknown Cast	United States	September 25, 2021	2020	PG-13	90	Documentaries	As her father nears the end of his life, filmm...
s7	Movie	My Little Pony: A	Robert Cullen,	Vanessa Hudgens, Kimiko Glenn	Unknown	September	2021	PG	91	Children &	Equestria's divided. But a

```
type_tvshow['duration'] = type_tvshow['duration'].apply(lambda x : x.replace(" Seasons", "") if "Seasons" in x else x)
type_tvshow['duration'] = type_tvshow['duration'].apply(lambda x : x.replace(" Season", "") if "Season" in x else x).astype(int)
type_tvshow.head(2)
```

```
<ipython-input-24-d5cb2e54f3b6>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  type_tvshow['duration'] = type_tvshow['duration'].apply(lambda x : x.replace(" Seasons", "") if "Seasons" in x else x)
<ipython-input-24-d5cb2e54f3b6>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  type_tvshow['duration'] = type_tvshow['duration'].apply(lambda x : x.replace(" Season", "") if "Season" in x else x).astype(int)
```

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
				Ama Qamata,						International	After crossing

```
type_movie['duration'].dtypes
```

```
dtype('int64')
```

```
type_tvshow['duration'].dtypes
```

```
dtype('int64')
```

The 'duration' column for both subsets of dataframe, df : *type_movie* and *type_tvshow* are now converted to int64 datatypes.

```
type_movie.describe()
```

	release_year	duration
count	6126.000000	6126.000000
mean	2013.120144	99.584884
std	9.681723	28.283225
min	1942.000000	3.000000
25%	2012.000000	87.000000
50%	2016.000000	98.000000
75%	2018.000000	114.000000
max	2021.000000	312.000000

```
type_tvshow.describe()
```

	release_year	duration
count	2664.000000	2664.000000
mean	2016.627628	1.751877
std	5.735194	1.550622
min	1925.000000	1.000000
25%	2016.000000	1.000000
50%	2018.000000	1.000000
75%	2020.000000	2.000000
max	2021.000000	17.000000

```
#longest tv show
type_tvshow.loc[type_tvshow['duration'] == 17]
```

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
s549	TV Show	Grey's Anatomy	Unknown Director	Ellen Pompeo, Sandra Oh, Katherine Heigl, Just...	United States	July 3, 2021	2020	TV-14	17	Romantic TV Shows, TV Dramas	Intern (and eventual resident) Meredith Grey f...

```
#longest movie
type_movie.loc[type_movie['duration'] == 312]
```

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
s4254	Movie	Black Mirror: Bandersnatch	Unknown Director	Fionn Whitehead, Will Poulter, Craig Parkinson...	United States	December 28, 2018	2018	TV-MA	312	Dramas, International Movies, Sci-Fi & Fantasy	In 1984, a young programmer begins to question...

```
# shortest movie
type_movie.loc[type_movie['duration'] == 3]
```

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
s3778	Movie	Silent	Limbert Fabian, Brandon Oldenburg	Unknown Cast	United States	June 4, 2019	2014	TV-Y	3	Children & Family Movies, Sci-Fi & Fantasy	"Silent" is an animated short film created by ...

```
type_tvshow['duration'].value_counts()
```

```
1     1791
2      421
3      198
4       94
5       64
6       33
7       23
8       17
9        9
10       6
13       2
15       2
12       2
17       1
11       1
Name: duration, dtype: int64
```

```
type_movie['release_year'].value_counts().sort_values(ascending = False).head(10)
```

```
2018    767
2017    765
2016    658
2019    633
2020    517
2015    396
2021    277
2014    264
2013    225
2012    173
Name: release_year, dtype: int64
```

```
type_movie['release_year'].min()
```

```
1942
```

```
type_movie['release_year'].max()
```

```
2021
```

```
type_tvshow['release_year'].value_counts().sort_values(ascending = False).head(10)
```

```
2020    436
2019    397
2018    379
2021    315
2017    265
2016    243
2015    159
2014     88
2012     63
2013     61
Name: release_year, dtype: int64
```

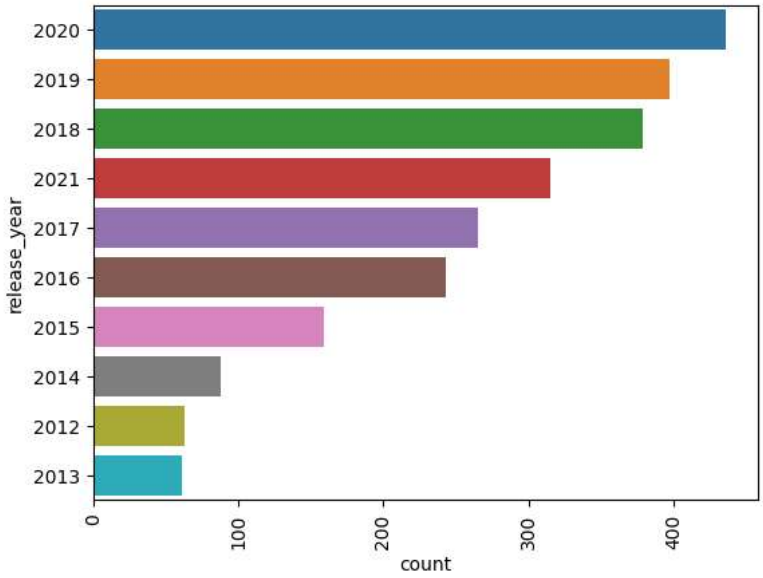
```
type_tvshow['release_year'].min()
```

```
1925
```

```
type_tvshow['release_year'].max()
```

```
2021
```

```
# Plotting a bar graph to show the year in which the maximum number of TV shows were released
sns.countplot(data = type_tvshow, y = 'release_year', order = type_tvshow['release_year'].value_counts().sort_values(ascending = False).h
plt.xticks(rotation = 90, fontsize = 10)
plt.show()
```



The top 10 release years for movies on Netflix are:

The top 10 release years for TV shows on Netflix are:

1. 2020 (436 shows)
2. 2019 (397 shows)
3. 2018 (379 shows)
4. 2021 (315 shows)
5. 2017 (265 shows)
6. 2016 (243 shows)
7. 2015 (159 shows)
8. 2014 (88 shows)
9. 2012 (63 shows)
10. 2013 (61 shows)

```
# Plotting a bar graph to show the years in which maximum number of movies were released
sns.countplot(data = type_movie, y = 'release_year', order = type_movie['release_year'].value_counts().sort_values(ascending = False).hea
plt.xticks(rotation = 90, fontsize = 10)
plt.show()
```

The top 10 release years for movies on Netflix are:

1. 2018 (767 movies)
2. 2017 (765 movies)
3. 2016 (658 movies)
4. 2019 (633 movies)
5. 2020 (517 movies)
6. 2015 (396 movies)
7. 2021 (277 movies)
8. 2014 (264 movies)
9. 2013 (225 movies)
10. 2012 (173 movies)

The bar chart created also shows that the most recent years have the most number of movies released on Netflix. This could be because Netflix is investing more in original content, or because it is becoming more popular as a streaming platform.

It would be interesting to see how this data changes over time. For example, it would be interesting to see if the number of movies released on Netflix continues to increase, or if it plateaus at some point. It would also be interesting to see if the distribution of movies released by year changes over time.

```
release_year_count = pd.pivot_table(df, values='title', index='release_year',
                                    columns='type', aggfunc='count',
                                    dropna=True).reset_index()
release_year_count.fillna(0)
```

type	release_year	Movie	TV Show
0	1925	0.0	1.0
1	1942	2.0	0.0
2	1943	3.0	0.0
3	1944	3.0	0.0
4	1945	3.0	1.0
...	...	...	...
69	2017	765.0	265.0
70	2018	767.0	379.0
71	2019	633.0	397.0
72	2020	517.0	436.0
73	2021	277.0	315.0

74 rows × 3 columns

```
plt.figure(figsize=(8,4))
plt.plot(release_year_count.loc[release_year_count['release_year']>=1990,
                                'release_year'],
         release_year_count.loc[release_year_count['release_year']>=1990,
                                'Movie'],ms=3)
plt.plot(release_year_count.loc[release_year_count['release_year']>=1990,
                                'release_year'],
         release_year_count.loc[release_year_count['release_year']>=1990,
                                'TV Show'],ms=3)

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()
```

Observations that can be made from the data:

There is a clear trend of increasing number of movies and TV shows released on Netflix over time.

The most popular release years for both movies and TV shows are 2018 and 2019.

There is a slight difference in the distribution of movies and TV shows released by year.

Movies are more evenly distributed across the years, while TV shows have a higher concentration in the most recent years.

These are just a few of the insights that can be gained from analyzing the data.

By exploring the data further, it is possible to gain a deeper understanding of the trends in the release of movies and TV shows on Netflix.

Rating Distribution

Each TV show and movie on Netflix is assigned a maturity rating to help members make informed choices for themselves and their children. Maturity ratings are either determined by Netflix or by a local standards organization. Netflix determines maturity ratings by the frequency and impact of mature content in a TV show or movie. TV show ratings reflect the overall maturity level of the whole series.

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

```
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            79
G             41
TV-Y7-FV       6
NC-17          3
UR             3
Name: rating, dtype: int64
```

The different types of ratings:

TV-MA: MA stands for 'Mature Audiences' and typically contains strong language, violence, and sexual content.

TV-14: This rating is for parents to be advised, and typically contains some material that may not be suitable for children under 14.

TV-PG: This rating is for parental guidance suggested, and typically contains some material that may not be suitable for younger children.

R: This rating is for restricted, and typically contains strong violence, sex, or language.

PG-13: This rating is for parents strongly cautioned, and typically contains some material that may be inappropriate for children under 13.

The other ratings on Netflix are less common, but they are still used for some titles. These ratings include:

TV-Y7: for children ages 7 and up

TV-Y: for children ages 2 and up

PG: for parental guidance suggested

TV-G: for general audiences

NR: This rating means that the rating is not yet known

Unknown rating: The rating is unknown

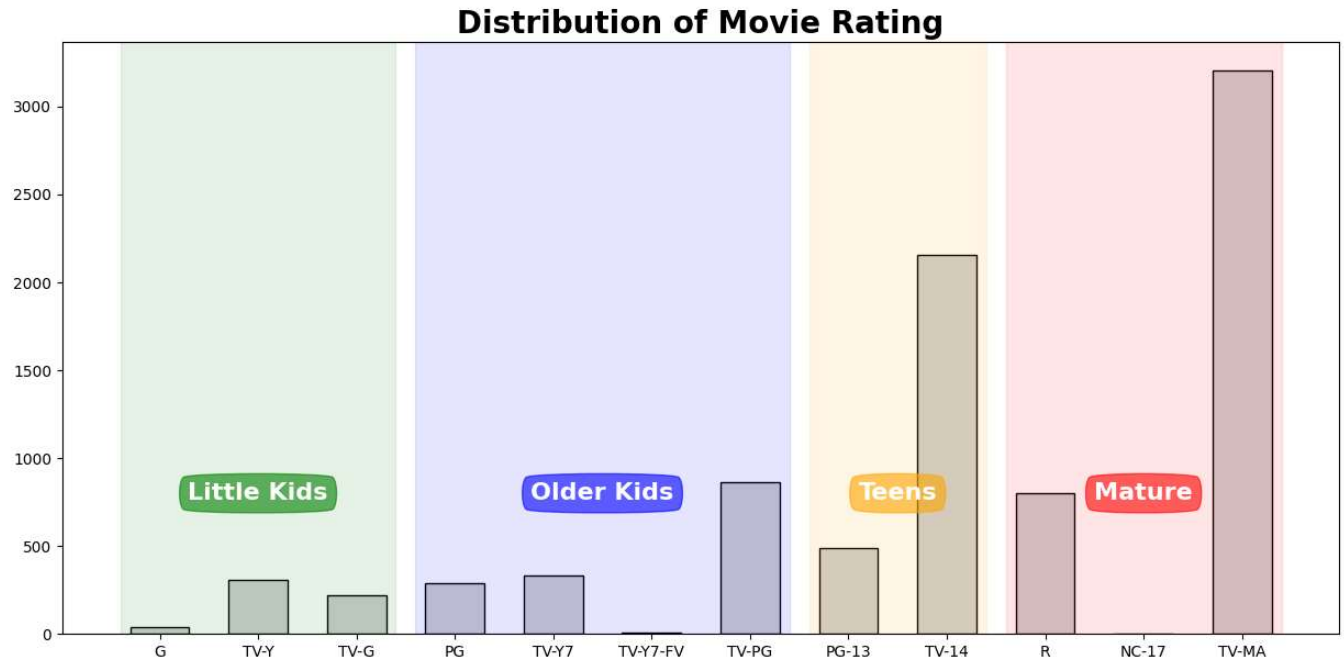
NC-17: This rating is for adults only, and typically contains strong violence, sex, or language

UR: This means Unrated

```
rating_order_movie =  ['G', 'TV-Y', 'TV-G', 'PG', 'TV-Y7', 'TV-Y7-FV', 'TV-PG', 'PG-13', 'TV-14', 'R', 'NC-17', 'TV-MA']
rating_order_tv =  ['G', 'TV-Y', 'TV-G', 'TV-Y7', 'TV-Y7-FV', 'TV-PG', 'TV-14', 'R', 'TV-MA']
movie_rating = df['rating'].value_counts()[rating_order_movie]
tv_rating = df['rating'].value_counts()[rating_order_tv].fillna(0)


def rating_barplot(data, title, height, h_lim=None):
    fig, ax = plt.subplots(1,1, figsize=(15, 7))
    if h_lim :
        ax.set_ylim(0, h_lim)
    ax.bar(data.index, data,  color="#d0d0d0", width=0.6, edgecolor='black')

    color =  ['green',  'blue',  'orange',  'red']
    span_range = [[0, 2], [3,  6], [7, 8], [9, 11]]

    for idx, sub_title in enumerate(['Little Kids', 'Older Kids', 'Teens', 'Mature']):
        ax.annotate(sub_title,
                    xy=(sum(span_range[idx])/2 ,height),
                    xytext=(0,0), textcoords='offset points',
                    va="center", ha="center",
                    color="w", fontsize=16, fontweight='bold',
                    bbox=dict(boxstyle='round4', pad=0.4, color=color[idx], alpha=0.6))
        ax.axvspan(span_range[idx][0]-0.4,span_range[idx][1]+0.4,  color=color[idx], alpha=0.1)

    ax.set_title(f'Distribution of {title} Rating', fontsize=20, fontweight='bold', position=(0.5, 1.0+0.03))
    plt.show()


Movie_ratingwise = df[df['type']== 'Movie']
Movie_ratingwise.rating.value_counts()
```

```
TV-MA       2062
TV-14       1427
R            797
TV-PG        540
PG-13        490
PG           287
TV-Y7        139
TV-Y         131
TV-G         126
NR            75
G             41
TV-Y7-FV       5
NC-17          3
UR             3
Name: rating, dtype: int64
```

```
rating_barplot(movie_rating,'Movie', 800)
```

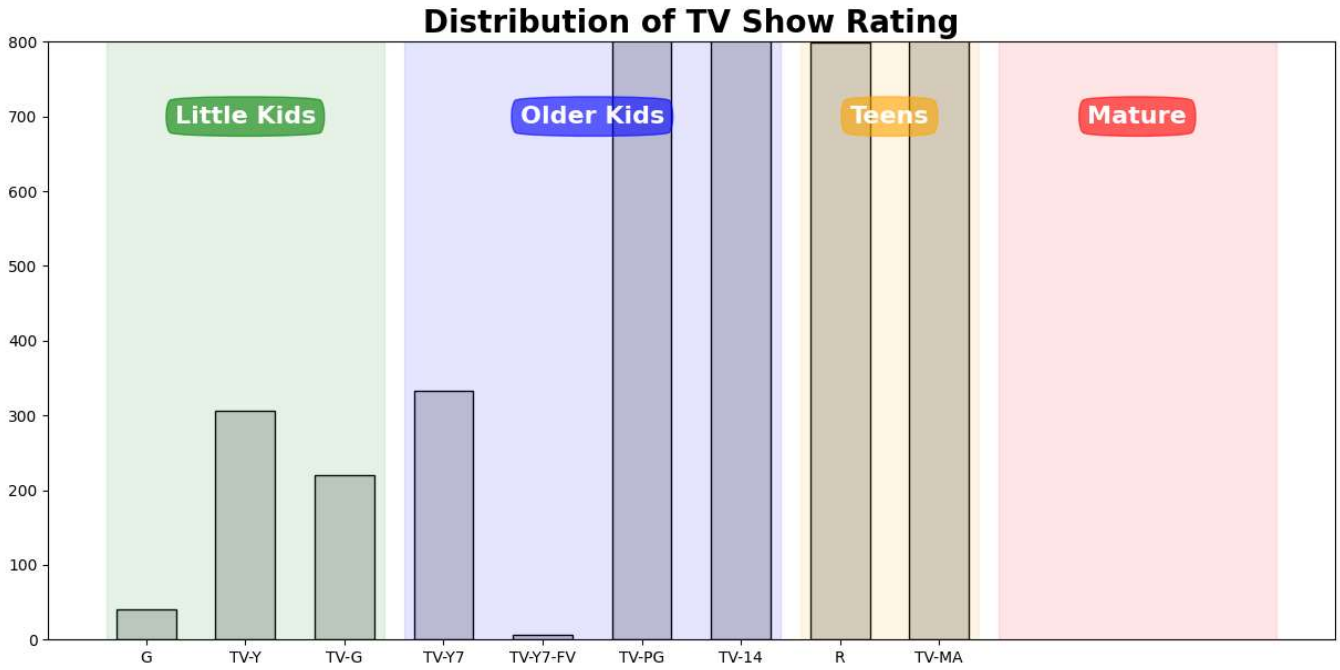



```
Tv_Show_ratingwise = df[df['type']== 'TV Show']
```

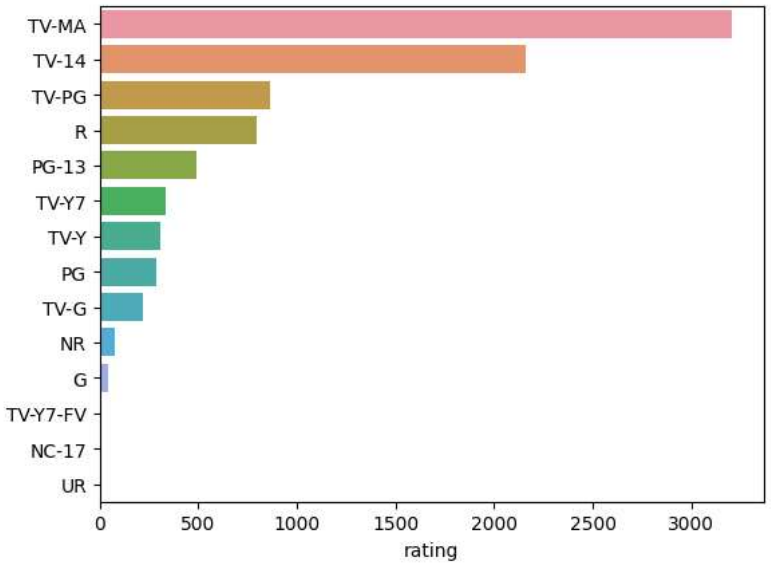
```
Tv_Show_ratingwise.rating.value_counts()
```

```
TV-MA       1143
TV-14        730
TV-PG        321
TV-Y7        194
TV-Y         175
TV-G          94
NR             4
R              2
TV-Y7-FV       1
Name: rating, dtype: int64
```

```
rating_barplot(tv_rating,'TV Show' , 700, 800)
```

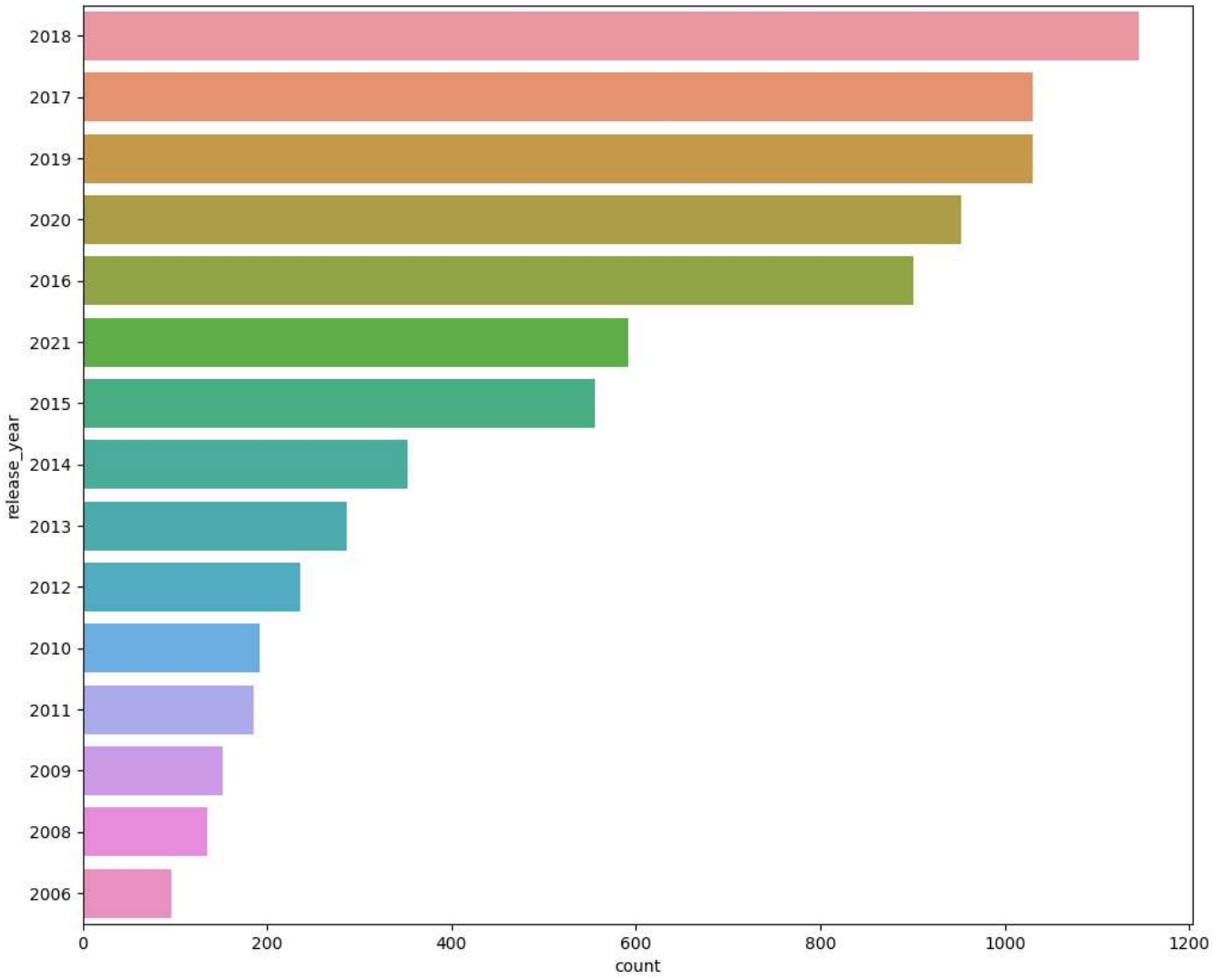



```
sns.barplot(x = df.rating.value_counts(), y = df.rating.value_counts().index, data = df, orient = 'h')
plt.show()
```



YEAR WISE COUNT

```
plt.figure(figsize=(12,10))
Release_Year = sns.countplot(y='release_year',data = df,order = df['release_year'].value_counts().index[0:15])
```



Highest releases were in 2018 followed by 2017 and 2019.

Different categories in which Titles are listed in :

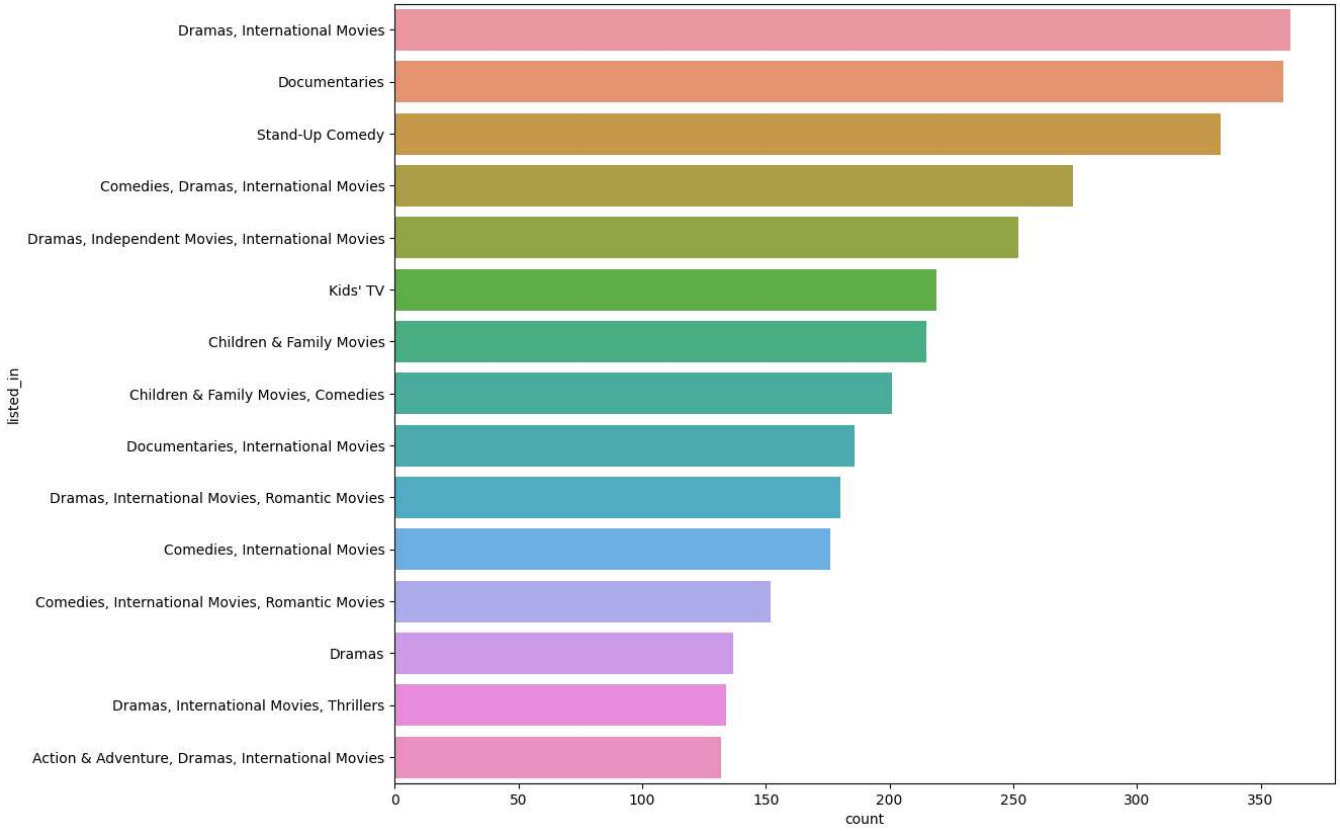
```
df.listed_in.value_counts().head(10)
```

```
Dramas, International Movies                        362
Documentaries                                       359
Stand-Up Comedy                                     334
Comedies, Dramas, International Movies              274
Dramas, Independent Movies, International Movies    252
Kids' TV                                            219
Children & Family Movies                            215
Children & Family Movies, Comedies                  201
Documentaries, International Movies                 186
Dramas, International Movies, Romantic Movies       180
Name: listed_in, dtype: int64
```

```
df.listed_in.value_counts().tail()
```

```
Crime TV Shows, International TV Shows, TV Sci-Fi & Fantasy    1
International TV Shows, TV Horror, TV Sci-Fi & Fantasy         1
Crime TV Shows, Kids' TV                                       1
Horror Movies, International Movies, Sci-Fi & Fantasy          1
Cult Movies, Dramas, Thrillers                                 1
Name: listed_in, dtype: int64
```

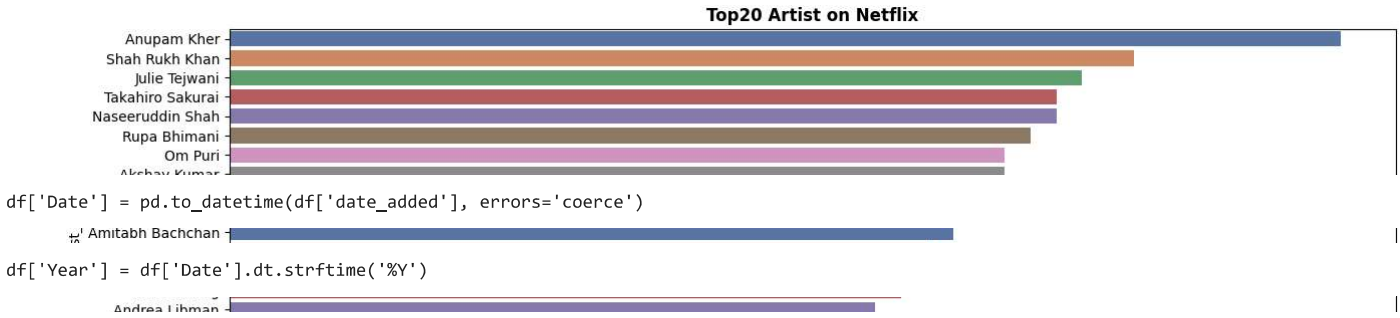
```
plt.figure(figsize = (12,10))
ax = sns.countplot( y = 'listed_in', data = df, order = df.listed_in.value_counts().index[0:15])
```



It's interesting to see that the top 3 genres are all related to documentaries, dramas, and international movies. This suggests that Netflix subscribers are interested in learning about different cultures and seeing stories from all around the world. It's also interesting to see that stand-up comedy is a popular genre on Netflix. This suggests that Netflix subscribers are looking for shows that are both entertaining and informative.

Cast

```
df['cast_name'] = df['cast'].apply(lambda x :  x.replace(' ,',',').replace(', ',',').split(','))
cast_count = []
for i in df['cast_name']: cast_count += i

cast_dict = dict((i, cast_count.count(i)) for i in cast_count)

df_cast_count = pd.DataFrame(cast_dict.values(),cast_dict.keys()).reset_index().sort_values(0,ascending=False).rename(
    columns = {'index' : 'cast_name', 0 : 'count'}).iloc[1:21]


plt.figure(figsize=(15,5))
sns.barplot(y='cast_name',x='count',data=df_cast_count,palette="deep")
plt.title("Top20 Artist on Netflix",fontweight="bold")
plt.xticks(rotation=90)
plt.show()
```




```
df['Date'] = pd.to_datetime(df['date_added'], errors='coerce')
```

```
df['Year'] = df['Date'].dt.strftime('%Y')
```

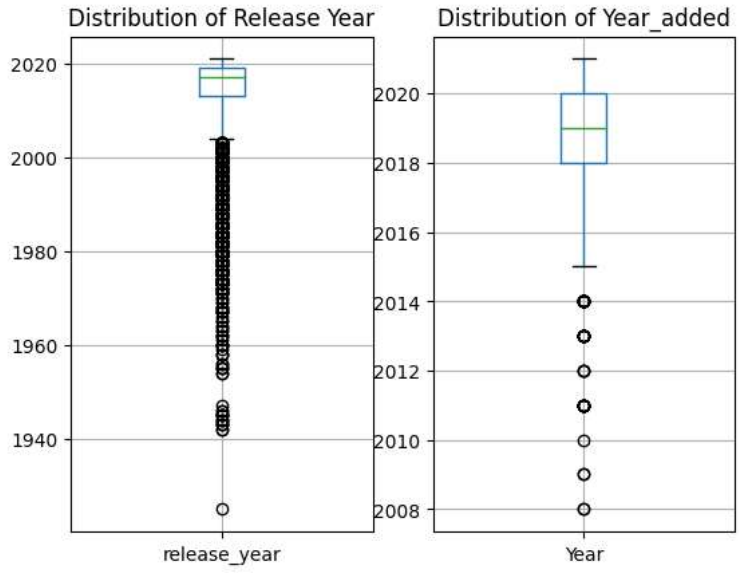
```
df['Year'] = df['Year'].astype(np.float).astype("Int64")
```

```
<ipython-input-53-2e00136c2e6b>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this wa
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  df['Year'] = df['Year'].astype(np.float).astype("Int64")
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8790 entries, s1 to s8807
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   type          8790 non-null   object        
 1   title         8790 non-null   object        
 2   director      8790 non-null   object        
 3   cast          8790 non-null   object        
 4   country       8790 non-null   object        
 5   date_added    8790 non-null   object        
 6   release_year  8790 non-null   int64         
 7   rating        8790 non-null   object        
 8   duration      8790 non-null   object        
 9   listed_in     8790 non-null   object        
 10  description   8790 non-null   object        
 11  cast_name     8790 non-null   object        
 12  Date          8790 non-null   datetime64[ns]
 13  Year          8790 non-null   Int64         
dtypes: Int64(1), datetime64[ns](1), int64(1), object(11)
memory usage: 1.3+ MB
```

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

df.boxplot(column='release_year', ax=ax[0])
ax[0].set_title('Distribution of Release Year')

df.boxplot(column='Year', ax=ax[1])
ax[1].set_title('Distribution of Year_added')

plt.show()
```



The boxplots shows the distribution of release year and year added for the shows and movies on Netflix. The release year boxplot shows that the most common release year for shows and movies on Netflix is 2018. There is also a long tail in the distribution, which suggests that there are a few shows and movies that were released in earlier years.

From 2015 more number of movies and TV shows were started adding.

The year added boxplot shows that the most common year that shows and movies were added to Netflix is 2020. There is also a long tail in the distribution, which suggests that there are a few shows and movies that were added to Netflix in earlier years.

The difference in the distributions of release year and year added suggests that Netflix is adding more recent shows and movies to its library. This is likely because Netflix is trying to keep up with the latest trends and to provide its subscribers with the most up-to-date content.

```
from sklearn.preprocessing import MultiLabelBinarizer # Similar to One-Hot Encoding

def relation_heatmap(df, title):
    df['genre'] = df['listed_in'].apply(lambda x :  x.replace(' ,',',').replace(', ',',').split(','))
    Types = []
    for i in df['genre']: Types += i
    Types = set(Types)
    print(f"There are {len(Types)} types in the Netflix {title} Dataset")
    test = df['genre']
    mlb = MultiLabelBinarizer()
    res = pd.DataFrame(mlb.fit_transform(test), columns=mlb.classes_, index=test.index)
    corr = res.corr()
    mask = np.zeros_like(corr, dtype=np.bool)
    mask[np.triu_indices_from(mask)] = True
    fig, ax = plt.subplots(figsize=(15, 14))
    pl = sns.heatmap(corr, mask=mask, cmap= "coolwarm", vmax=.5, vmin=-.5, center=0, square=True, linewidths=.7, cbar_kws={"shrink": 0.6}

    plt.show()


relation_heatmap(type_movie, 'Movie')
```

```
<ipython-input-56-536c8eaaef0c>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['genre'] = df['listed_in'].apply(lambda x :  x.replace(' ,',',').replace(', ',',').split(','))
<ipython-input-56-536c8eaaef0c>:13: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To silence this war
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  mask = np.zeros_like(corr, dtype=np.bool)
There are 20 types in the Netflix Movie Dataset
```

In film, the negative relationship between drama and documentary is remarkable. You can also see that there are many dramas for independent and international films.

```
relation_heatmap(type_tvshow, 'TV Show')
```

```
<ipython-input-56-536c8eaaef0c>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['genre'] = df['listed_in'].apply(lambda x :  x.replace(' ,',',').replace(', ',',').split(','))
<ipython-input-56-536c8eaaef0c>:13: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To silence this war
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  mask = np.zeros_like(corr, dtype=np.bool)
There are 22 types in the Netflix TV Show Dataset
```


TV shows are more clearly correlated than movies.

The most obvious is the relationship between kids and International (Could it be that kids' content is important to their culture?), Science & Natural and Docuseries.

BUSINESS INSIGHTS

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.

RECOMMENDATIONS

1) Add more content for young/general audiences: 80% of the content on netflix (7022 out of 8790 titles) is either for mature audiences or requires parental guidance. In order to expand its target audience, Netflix needs to add more shows for family and children.

2) Add more variety of content in countries other than US/UK: While US and UK have a healthy mix of all categories of content, other contries do not. At least other english speaking countries like Australia and India should have an equally good mix of content. A wider population can be targetted with more titles in genres like documentaries, horror, standup comedy, crime and musicals for these countries.

3) Country-specific genres: Just like Korean drama and Anime are available in Korea/Japan, more country-specific niches should be created in order to build more customisation. French and German shows would increase business in Europe. India also has blockbusters in many regional languages.

4) Promotion & Customer Acquisition in top 5 countries: Since Netflix already has a lot of content for countries like US, UK, India, Canada, France, Germany, Japan and South Korea; the focus in these countries should be on customer acquisition. Tie-ups with local businesses in these countries which already have many subscribers (food delivery/telecom/editorial) can help increase customer volume.

5) Addition of more content in the next 5 countries: Countries like China, Indonesia, Mexico and Brazil are some of the most populous countries of the world. Content needs to be the primaryfocus to grow business in these set of countries.

6) Target older Population: 75% of the content on the platform is released after 2014. Older population can be targeted by adding more content released in the 1970s-1990s