Defining Problem Statement and Analysing basic metrics (10 Points)

Problem Statement

Which type of TV/movie programs should Netflix produce to increase its profitability by gaining better viewership?

```
#import the pandas library
import pandas as pd
```

```
#Load Netflix data file into panda dataframe
data=pd.read_csv('sample_data/netflix.csv')
```

Carry out basic metrics analysis

```
#get to know the shape of the dataset
data.shape
```

```
(8807, 12)
```

The ***shape*** output suggests that the dataset has **8807 rows** and **12 columns**

```
#List the top 5 records from the dataset
data.head()
```

	show_id	type	title	director	cast	country	date_added	release_year
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

```
#backup original dataframe in another dataframe
back_df=data
```

```
#List the last 5 records from the dataset
data.tail()
```

	show_id	type	title	director	cast	country	date_added	release_ye
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	2
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	2

Lets find out all the columns in the dataset

```
data.columns
```

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

Lets find out more information about these columns in the dataset

```
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
```

Its important to note that* **all columns(Series):*** exepct the "release_year" column are read as ***"object"*** data types while ***"release_year"*** is read as an ***intege***r

Double-click (or enter) to edit

Now, lets find out information about the ***non-null value***s in this dataset

```
data.count()
```

```
show_id         8807
type            8807
title           8807
director        6173
cast            7982
country         7976
date_added      8797
release_year    8807
rating          8803
duration        8804
listed_in       8807
description     8807
dtype: int64
```

Lets find out about the index information

```
data.index
```

```
RangeIndex(start=0, stop=8807, step=1)
```

As we can see , the first index position start at 0 and the last index position ends at 8807

Now, lets apply the describe function on the dataset to see its effect.

```
data.describe()
```

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

```
data.describe(include='all').T
```

	count	unique	top	freq	mean	std	min	25%
show_id	8807	8807	s1	1	NaN	NaN	NaN	NaN
type	8807	2	Movie	6131	NaN	NaN	NaN	NaN
title	8807	8807	Dick Johnson Is Dead	1	NaN	NaN	NaN	NaN
director	6173	4528	Rajiv Chilaka	19	NaN	NaN	NaN	NaN
cast	7982	7692	David Attenborough	19	NaN	NaN	NaN	NaN
country	7976	748	United States	2818	NaN	NaN	NaN	NaN
date_added	8797	1767	January 1, 2020	109	NaN	NaN	NaN	NaN
release_year	8807.0	NaN	NaN	NaN	2014.180198	8.819312	1925.0	2013.0
rating	8803	17	TV-MA	3207	NaN	NaN	NaN	NaN
duration	8804	220	1 Season	1793	NaN	NaN	NaN	NaN

Lets look at the ***number of unique values*** in each Series of the dataset

```
data.nunique()
```

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

Now, lets find out the total number of null and NaN values in the dataset

```
def missing_values_count(column_name):
  total_count=data[column_name].nunique()
  unique_count=data[column_name].nunique()
  nan_count = data[column_name].isna().sum().sum()
  null_count = data[column_name].isnull().sum().sum()
  total_missing_count=nan_count+null_count

  #print(f"Total values in column  {column_name} is {Total_count} and total number of missing values is {Total_missing_count} out of w
  print(f"Column name : {column_name.ljust(15) }           Total Values: {str(total_count).ljust(6)} Total Unique: {str(unique_count).
```

Lets do this analysis for each column/Series

```
for column_name in data.columns:
  missing_values_count(column_name)
```

```
Column name : show_id                    Total Values: 8807   Total Unique: 8807   Total Missing : 0       Null Count :0      NaN Co
Column name : type                       Total Values: 2      Total Unique: 2      Total Missing : 0       Null Count :0      NaN Co
Column name : title                      Total Values: 8807   Total Unique: 8807   Total Missing : 0       Null Count :0      NaN Co
Column name : director                   Total Values: 4528   Total Unique: 4528   Total Missing : 5268    Null Count :2634   NaN Co
Column name : cast                       Total Values: 7692   Total Unique: 7692   Total Missing : 1650    Null Count :825    NaN Co
Column name : country                    Total Values: 748    Total Unique: 748    Total Missing : 1662    Null Count :831    NaN Co
Column name : date_added                 Total Values: 1767   Total Unique: 1767   Total Missing : 20      Null Count :10     NaN Co
Column name : release_year               Total Values: 74     Total Unique: 74     Total Missing : 0       Null Count :0      NaN Co
Column name : rating                     Total Values: 17     Total Unique: 17     Total Missing : 8       Null Count :4      NaN Co
Column name : duration                   Total Values: 220    Total Unique: 220    Total Missing : 6       Null Count :3      NaN Co
Column name : listed_in                  Total Values: 514    Total Unique: 514    Total Missing : 0       Null Count :0      NaN Co
Column name : description                Total Values: 8775   Total Unique: 8775   Total Missing : 0       Null Count :0      NaN Co
```

```
#Lets look at the percentage wise missing values for each column
data.isnull().sum()/len(data)*100
```

```
show_id          0.000000
type             0.000000
title            0.000000
director        29.908028
cast             9.367549
country          9.435676
date_added       0.113546
release_year     0.000000
rating           0.045418
duration         0.034064
listed_in        0.000000
description      0.000000
dtype: float64
```

As you can see the director column has close to 30% missing values followed by case and country columns with almost 9.5% missing values

```
data['description'].nunique()
```

```
8775
```

Univariate data analysis

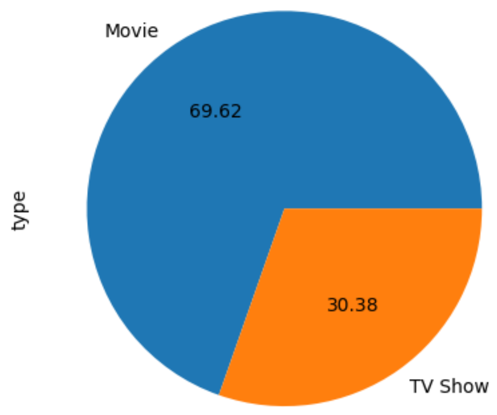
```
data['type'].value_counts(normalize=True)*100
```

```
Movie      69.615079
TV Show    30.384921
Name: type, dtype: float64
```

Shows that 69% of the content is Movies while 30.38% is TV shows.

```

data['type'].value_counts().plot(kind='pie',autopct='%.2f')

```

```
<Axes: ylabel='type'>
```



UNNEST columns with multiple values in same column

```
# director_list=data['cast'].apply(lambda x: str(x).split(', ')).tolist()
# data_new=pd.DataFrame(director_list,index=data['title'])
# data_new=data_new.stack()
# data_new=pd.DataFrame(data_new)
# data_new.reset_index(inplace=True)
# data_new=data_new[['title',0]]
# data_new.columns=['title','cast']
```

```
def unnest_column(column_name,id_column_name):
    data_list=data[column_name].apply(lambda x: str(x).split(', ')).tolist()
    new_df=pd.DataFrame(data_list,index=data[id_column_name])
    new_df=new_df.stack()
    new_df=pd.DataFrame(new_df)
    new_df.reset_index(inplace=True)
    new_df=new_df[[id_column_name,0]]
    new_df.columns=[id_column_name,column_name]
    return new_df
```

```
director_df=unnest_column('director','title')
cast_df=unnest_column('cast','title')
country_df=unnest_column('country','title')
listed_in_df=unnest_column('listed_in','title')
```

```
data.columns
```

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

```
len(data.index)
```

```
8807
```

```
main_data_df=data[['show_id', 'type', 'title',  'date_added', 'release_year', 'rating', 'duration',  'description']]
```

```
len(main_data_df.index)
```

```
8807
```

```
final_df=main_data_df.merge(director_df,on='title').merge(country_df,on='title').merge(listed_in_df,on='title').merge(cast_df,on='titl
final_df["date_added"]=pd.to_datetime(final_df["date_added"])
final_df
```

	show_id	type	title	date_added	release_year	rating	duration	descript
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	90 min	As her fa nears end of life, film
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2 Seasons	A cross paths party, a C Town
2	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2 Seasons	A cross paths party, a C Town
3	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2 Seasons	A cross paths party, a C

**Identify incorrect values*

```
final_df['rating'].unique()
```

```
array(['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7', 'R',
       'TV-G', 'G', 'NC-17', '74 min', '84 min', '66 min', 'NR', nan,
       'TV-Y7-FV', 'UR'], dtype=object)
```

```

```

We can see that rating column has 3 invalid ratings which are actually duration column values. We should therefore replace these with Null and place the duration values in correct column

Update values in correct column

Lets update the column values manually

```
import numpy as np
final_df.loc[final_df['rating']=='66 min','duration']='66 min'
final_df.loc[final_df['rating']=='66 min','rating']=np.nan
```

```
final_df.loc[final_df['rating']=='74 min','duration']='74 min'
final_df.loc[final_df['rating']=='74 min','rating']=np.nan
```

```
final_df.loc[final_df['rating']=='84 min','duration']='84 min'
final_df.loc[final_df['rating']=='84 min','rating']=np.nan
```

Fix Nan and other values using imputation

```

```

Perform imputatations for country column based on director **information**

```
#country column is imputed on the basis of director,i.e- suppose there's a null for country
#when we have a director whose other movies have a country given.So below piece of code just checks the mode of
#country for the director
# and imputes in place of nulls the corresponding mode

for i in final_df[final_df['country'].isnull()]['director'].unique(): # all the places where corresponding country is missing for a di
  if i in final_df[~final_df['country'].isnull()]['director'].unique():
    imp=final_df[final_df['director']==i]['country'].mode().values[0]
    final_df.loc[final_df['director']==i,'country']=final_df.loc[final_df['director']==i,'country'].fillna(imp)
```

```
final_df['rating'].unique()
```

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

```
#Remove minutes from duration
final_df['duration']=final_df['duration'].str.replace(" min","")
final_df.head()
```

	show_id	type	title	date_added	release_year	rating	duration	description
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	90	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2 Seasons	After crossing paths at a

```
#Remove season from duration for TV shows
final_df['duration']=final_df['duration'].str.replace(" Seasons","")
final_df['duration']=final_df['duration'].str.replace(" Season","")
final_df.head()
```

	show_id	type	title	date_added	release_year	rating	duration	description
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	90	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2	After crossing paths at a

```
#Convert duration to int
final_df['duration']=final_df['duration'].astype('int')
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 201991 entries, 0 to 201990
Data columns (total 12 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   show_id       201991 non-null  object
 1   type          201991 non-null  object
 2   title         201991 non-null  object
 3   date_added    201833 non-null  datetime64[ns]
 4   release_year  201991 non-null  int64
 5   rating        201921 non-null  object
 6   duration      201991 non-null  int64
 7   description   201991 non-null  object
 8   director      201991 non-null  object
 9   country       201991 non-null  object
 10  listed_in     201991 non-null  object
 11  cast          201991 non-null  object
```

```
dtypes: datetime64[ns](1), int64(2), object(9)
memory usage: 20.0+ MB
```

Visualize

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

```
final_df
```

	show_id	type	title	date_added	release_year	rating	duration	descript
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	90	As her fat nears end of life, film
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2	A cross paths party, a C Town
2	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2	A cross paths party, a C Town
3	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2	A cross paths party, a C

```
release_year_counts=final_df["release_year"].value_counts()
release_year_counts
```

```
2018    24414
2019    21931
2017    20516
2020    19679
2016    18465
        ...
1947        8
1946        6
1942        6
1943        5
1925        1
Name: release_year, Length: 74, dtype: int64
```

Bivariate data analysis

```
plt.bar(release_year_counts.index,release_year_counts.values,width=.8)
plt.title('Count of number of releases every year')
plt.xlabel('Year')
plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```

Count of number of releases every year

```
tv_show_df=final_df.loc[final_df['type']=='TV Show']
tv_show_df

```

	show_id	type	title	date_added	release_year	rating	duration	descripti
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2	Aft crossi paths at party, a Ca Town t
2	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2	Aft crossi paths at party, a Ca Town t
3	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2	Aft crossi paths at party, a Ca Town t
4	s2	TV Show	Blood &	2021-09-24	2021	TV-MA	2	Aft crossi paths at

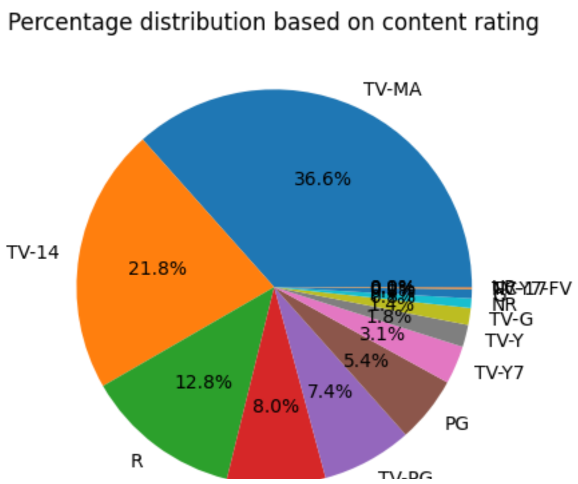
```
movies_df=final_df.loc[final_df['type']=='Movie']
movies_df
```

	show_id	type	title	date_added	release_year	rating	duration	desc
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	90	As (lif
159	s7	Movie	My Little Pony: A New Generation	2021-09-24	2021	PG	91	E div a br
160	s7	Movie	My Little Pony: A New Generation	2021-09-24	2021	PG	91	E div a br
161	s7	Movie	My Little Pony: A New Generation	2021-09-24	2021	PG	91	E div a br
			My Little					E

```
ratings_sum=final_df['rating'].value_counts()

ratings_sum
```

```
TV-MA       73867
TV-14       43931
R           25860
PG-13       16246
TV-PG       14926
PG          10919
TV-Y7        6304
TV-Y         3665
TV-G         2779
NR           1573
G            1530
NC-17         149
TV-Y7-FV       86
UR             86
Name: rating, dtype: int64
```

```
plt.pie(ratings_sum,labels=ratings_sum.index,autopct='%1.1f%%')
plt.title('Percentage distribution based on content rating')
plt.show()
```



```
tv_rating_sum=final_df.loc[final_df['type']=='TV Show','rating'].value_counts()
tv_rating_sum

```

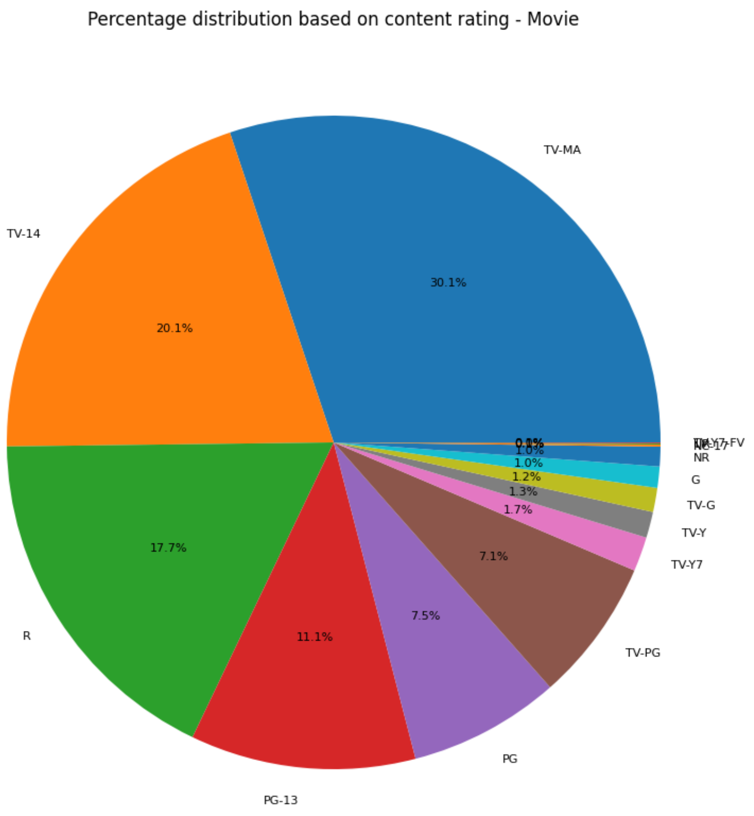
```
TV-MA       29906
TV-14       14691
TV-PG        4614
TV-Y7        3818
TV-Y         1787
TV-G         1041
NR            155
R              54
TV-Y7-FV       24
Name: rating, dtype: int64
```

```
plt.figure(figsize=(10,10))
plt.pie(tv_rating_sum,labels=tv_rating_sum.index,autopct='%1.1f%%',textprops={'fontsize': 8})
plt.title('Percentage distribution based on content rating - TV Show')
plt.show()
```

```
movie_rating_sum=final_df.loc[final_df['type']=='Movie','rating'].value_counts()
movie_rating_sum
```

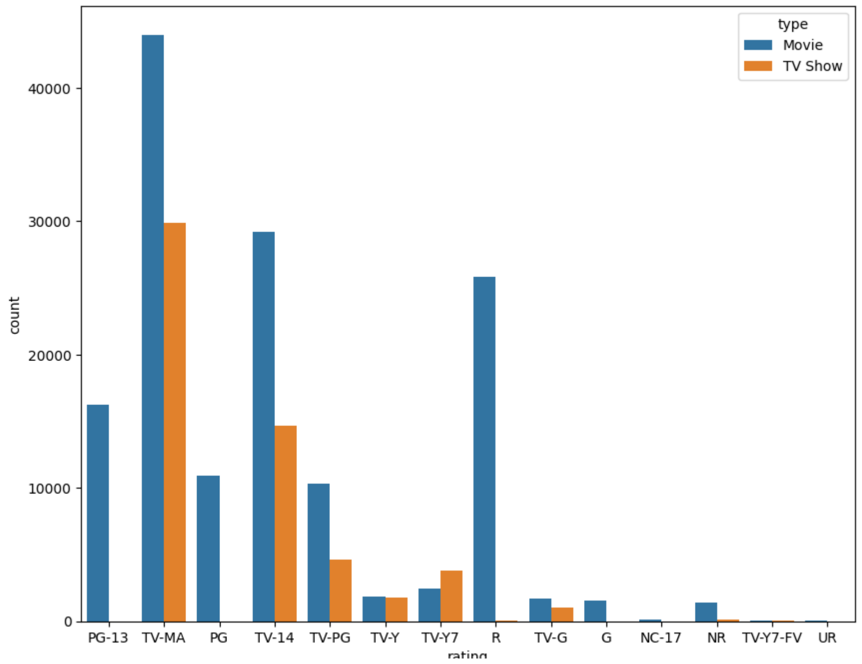
```
TV-MA       43961
TV-14       29240
R           25806
PG-13       16246
PG          10919
TV-PG       10312
TV-Y7        2486
TV-Y         1878
TV-G         1738
G            1530
NR           1418
NC-17         149
UR             86
TV-Y7-FV       62
Name: rating, dtype: int64
```

```
plt.figure(figsize=(10,10))
plt.pie(movie_rating_sum,labels=movie_rating_sum.index,autopct='%1.1f%%',textprops={'fontsize': 8})
plt.title('Percentage distribution based on content rating - Movie')
plt.show()
```



```
plt.figure(figsize=(10,8))
sns.countplot(x='rating', hue='type', data=final_df)
```

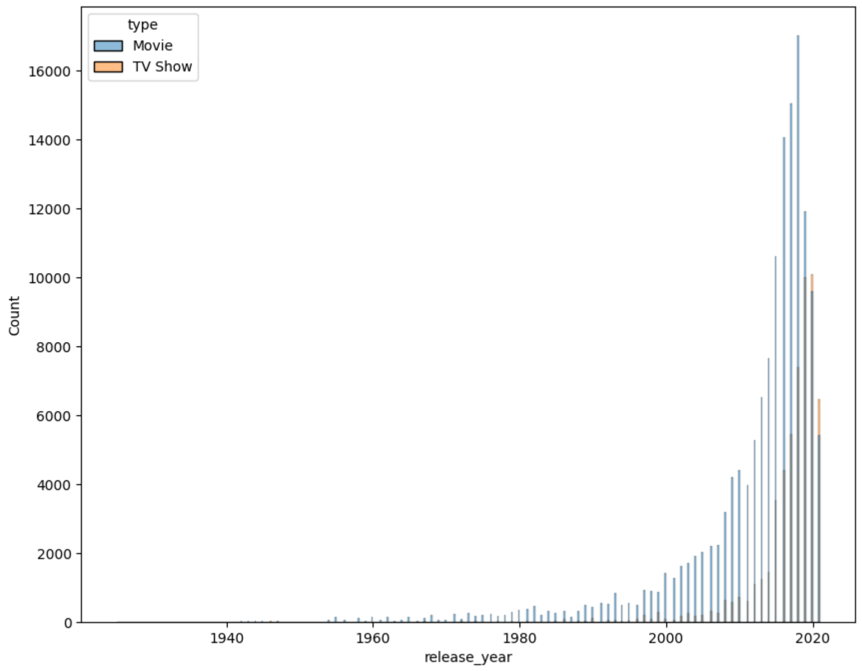
```
<Axes: xlabel='rating', ylabel='count'>
```



We can notice that most content is under the adult category and is meant for audience above 17 years

```
plt.figure(figsize=(10,8))
sns.histplot(x='release_year',hue='type',data=final_df)
```

```
<Axes: xlabel='release_year', ylabel='Count'>
```

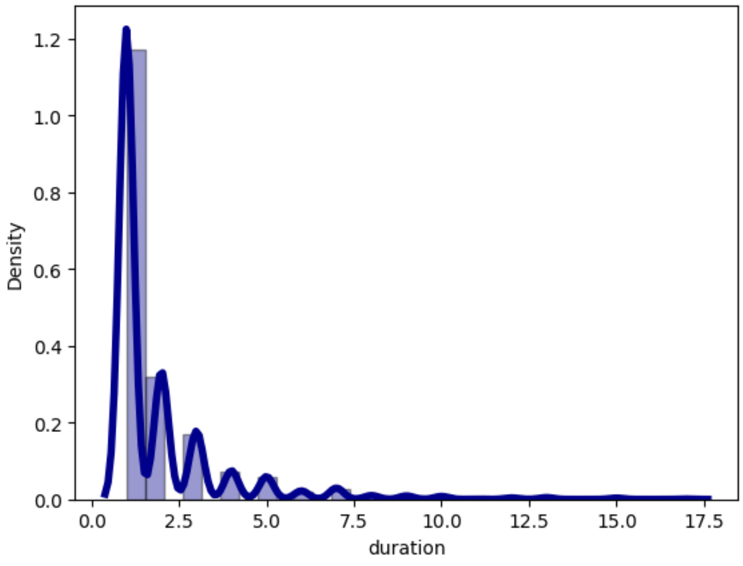


```
tv_df=final_df.loc[final_df['type']=='TV Show']
tv_df.head()
```

```
show_id  type  title  date_added  release_year  rating  duration  description  dir
                                                                  After
```

```
# lets look at the duration of the TV shows
tv_df=final_df.loc[final_df['type']=='TV Show']
tv_df.head()

sns.distplot(tv_df['duration'], hist=True, kde=True,
bins=int(30), color = 'darkblue',
hist_kws={'edgecolor':'black'},
kde_kws={'linewidth': 4})
plt.show()
```

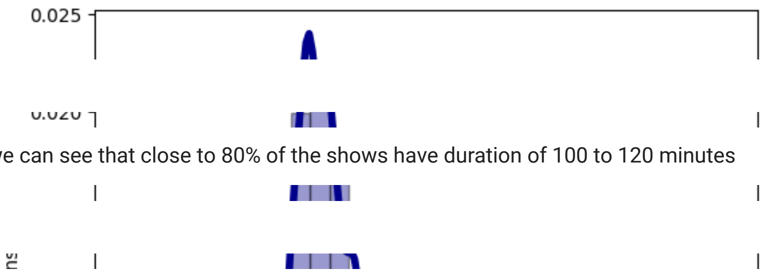
```
<ipython-input-115-3172cc84439b>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(tv_df['duration'], hist=True, kde=True,
```



We can see that most TV shows have 1 **season**

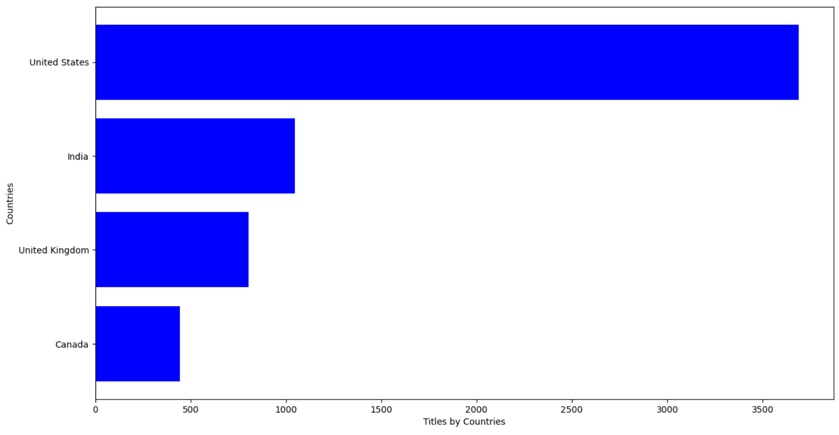
```
#Lets look at duration of movies

movie_df=final_df.loc[final_df['type']=='Movie']
movie_df.head()
sns.distplot(movie_df['duration'], hist=True, kde=True,
bins=int(30), color = 'darkblue',
hist_kws={'edgecolor':'black'},
kde_kws={'linewidth': 4})
plt.show()
```

```
<ipython-input-116-093e4cd265a5>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(movie_df['duration'], hist=True, kde=True,
```



here we can see that close to 80% of the shows have duration of 100 to 120 minutes

```
#Now, lets look at the country wise distribution
df_country=final_df.groupby(['country']).agg({"title":"nunique"}).reset_index().sort_values(by=['title'],ascending=False)[:5]
df_country=df_country[df_country['country']!='nan']
plt.figure(figsize=(15,8))
plt.barh(df_country[::-1]['country'], df_country[::-1]['title'],color=['blue'])
plt.xlabel('Titles by Countries')
plt.ylabel('Countries')
plt.show()
```



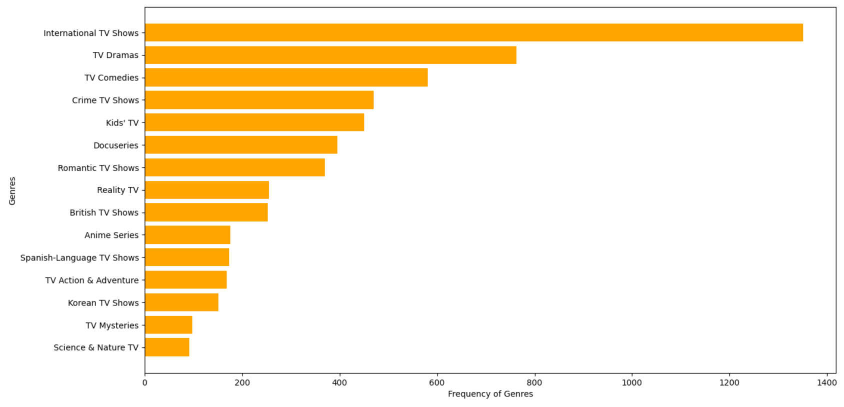
USA, UK, India are top content producing nations

Now, lets look at the top Genre is TV shows

```
tv_show_df.head()
```

show_id	type	title	date_added	release_year	rating	duration	description
	TV	Blood					After crossing

```
df_genre=tv_show_df.groupby(['listed_in']).agg({"title":"nunique"}).reset_index().sort_values(by=['title'],ascending=False)[:15]
plt.figure(figsize=(15,8))
plt.barh(df_genre[::-1]['listed_in'], df_genre[::-1]['title'],color=['orange'])
plt.xlabel('Frequency of Genres in TV Shows')
plt.ylabel('Genres - TV Shows')
plt.show()
```

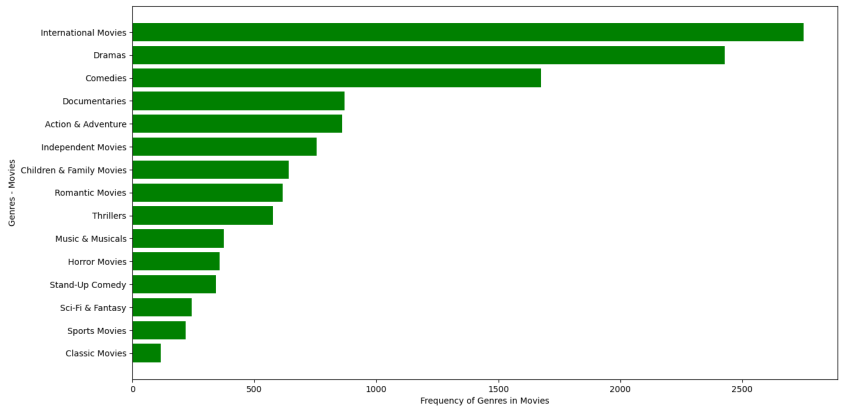


International TV shows dominate the TV Shows category followed by Drama and Comedy Genre

```
1
```

Now, lets see the most popular Genre in Movies

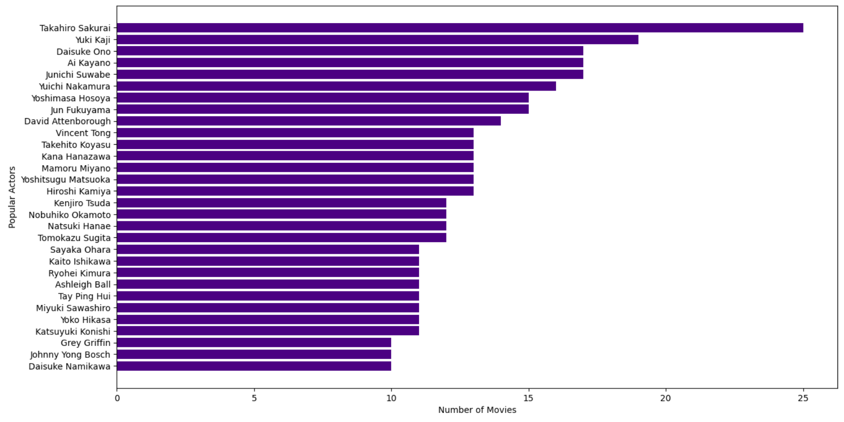
```
df_genre=movie_df.groupby(['listed_in']).agg({"title":"nunique"}).reset_index().sort_values(by=['title'],ascending=False)[:15]
plt.figure(figsize=(15,8))
plt.barh(df_genre[::-1]['listed_in'], df_genre[::-1]['title'],color=['green'])
plt.xlabel('Frequency of Genres in Movies')
plt.ylabel('Genres - Movies')
plt.show()
```



Here as well International movies leads the popularity charts, closely followed by Drama and Comedy movies

Now lets look at the popular actors on Netflix TV Shows

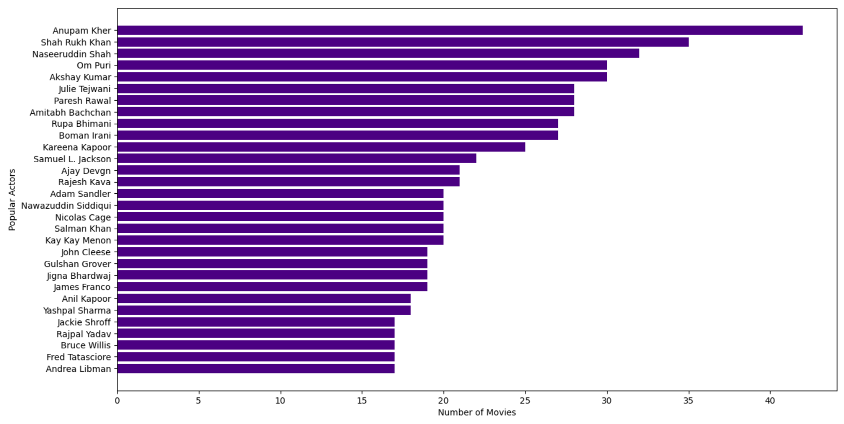
```
df_actors=tv_show_df.groupby(['cast']).agg({"title":"nunique"}).reset_index().sort_values(by=['title'],ascending=False)[:31]
df_actors=df_actors[df_actors['cast']!='nan']
plt.figure(figsize=(15,8))
plt.barh(df_actors[::-1]['cast'], df_actors[::-1]['title'],color=['indigo'])
plt.xlabel('Number of Movies')
plt.ylabel('Popular Actors')
plt.show()
```



Takahiro Sakurai leads the popular actors followed by Yuki Kaji and Daisuke Ono and Ai Kayano and Junichi Suwabe

Now lets look at the popular actors on Netflix Movies

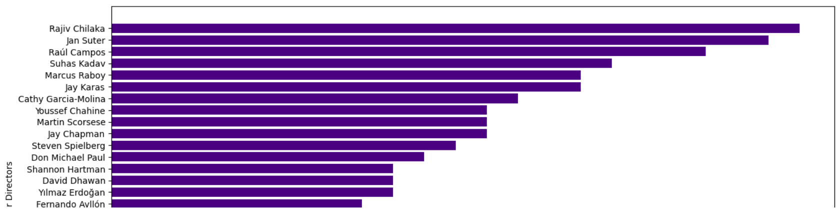
```
df_actors=movie_df.groupby(['cast']).agg({"title":"nunique"}).reset_index().sort_values(by=['title'],ascending=False)[:31]
df_actors=df_actors[df_actors['cast']!='nan']
plt.figure(figsize=(15,8))
plt.barh(df_actors[::-1]['cast'], df_actors[::-1]['title'],color=['indigo'])
plt.xlabel('Number of Movies')
plt.ylabel('Popular Actors')
plt.show()
```



Anupam Kher is the most popular actor in netflix movies followed by Shahrukh Khan , Nasseruddin Shah

Now lets look at the popular directors on Netflix Movies

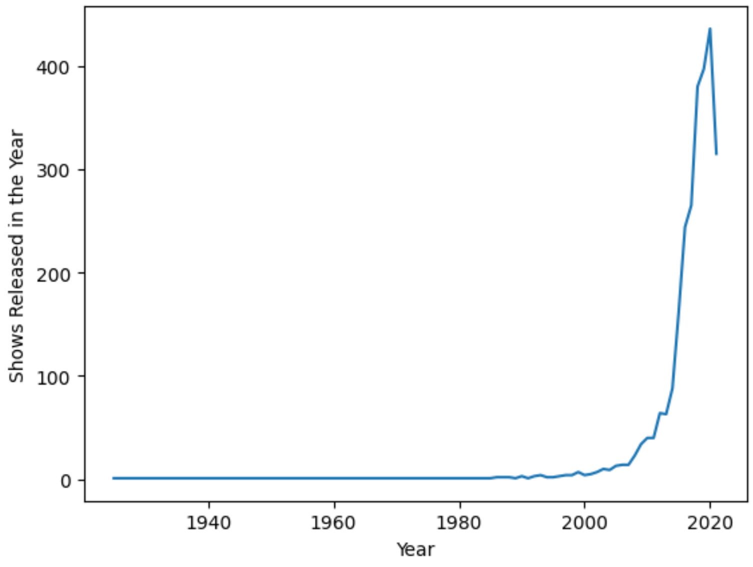
```
df_directors=movie_df.groupby(['director']).agg({"title":"nunique"}).reset_index().sort_values(by=['title'],ascending=False)[:31]
df_directors=df_directors[df_directors['director']!='nan']
plt.figure(figsize=(15,8))
plt.barh(df_directors[::-1]['director'], df_directors[::-1]['title'],color=['indigo'])
plt.xlabel('Number of Movies')
plt.ylabel('Popular Directors')
plt.show()
```



Rajiv Chilaka is the most popular director

Now lets see the release pattern of TV shows year on year

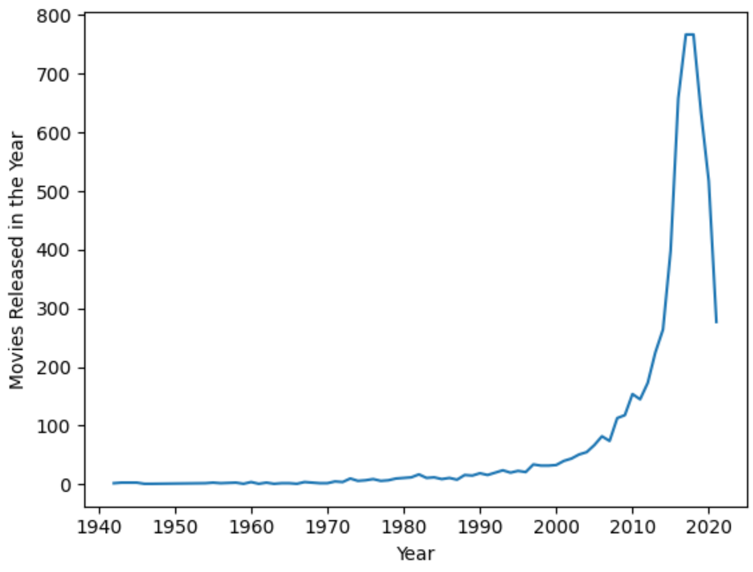
```
df_year=tv_show_df.groupby(['release_year']).agg({"title":"nunique"}).reset_index()
sns.lineplot(data=df_year, x='release_year', y='title')
plt.ylabel("Shows Released in the Year")
plt.xlabel("Year")
plt.show()
```



2018 and 2019 had the maximum number of releases as far as TV Shows are concerned.

Now, lets review the numbers for movies

```
df_year=movie_df.groupby(['release_year']).agg({"title":"nunique"}).reset_index()
sns.lineplot(data=df_year, x='release_year', y='title')
plt.ylabel("Movies Released in the Year")
plt.xlabel("Year")
plt.show()
```



There is an upward trend starting from 2010 which peaks around 2018 and takes a dip in 2020

1) The most popular Genres across the countries and in both TV Shows and Movies are is International shows , Drama and Comedy. Continuing to invest in shows and movies in these Genres is recommmended

2)For TV Shows Single season shows seem to be in demand and should be continued that way. Indicates, audience prefers new content over new seasons of existing TV shows.

3) The movies with duration between 100 to 120 are the most popular and content producers should ensure their content remains in this viewing duration

4) There has been a decline in new movies since 2019 and Netflix should look at filling this gap to retain its movie audience base

5) While creating content, popular actors/director combination should be taken into account along with regional preferences