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

```
df = pd.read_csv("/content/netflix_titles.csv")
```

```
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
```

Un-nesting of Nested Columns

Un-nesting the columns that have cells with multiple comma separated values by creating multiple rows

```
def unnest_columns(df, columns):
    for col in columns:
        df[col] = df[col].str.split(', ')
        df = df.explode(col)
    return df
```

```
Nested_columns = ['director', 'cast', 'country', 'listed_in']
df = unnest_columns(df, Nested_columns)
df.head(10)
```

	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	South Africa	September 24, 2021	2021
1	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	September 24, 2021	2021

Double-click (or enter) to edit

After unnesting the column index ranges increased compare to previous

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 201991 entries, 0 to 8806
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   director      151348 non-null  object
 4   cast          199845 non-null  object
 5   country       190094 non-null  object
 6   date_added    201833 non-null  object
 7   release_year  201991 non-null  int64
 8   rating        201924 non-null  object
 9   duration      201988 non-null  object
 10  listed_in     201991 non-null  object
 11  description   201991 non-null  object
dtypes: int64(1), object(11)
memory usage: 20.0+ MB
```

Missing values

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

```
show_id             0
type                0
title               0
director        50643
cast             2146
country         11897
date_added        158
release_year        0
rating             67
duration            3
listed_in           0
description         0
dtype: int64
```

Handling null values

```
df["director"].fillna("Unknown Director", inplace=True)
df["cast"].fillna("Unknown Cast", inplace=True)
df["country"].fillna("Unknown Country", inplace=True)
df["rating"].fillna("Unknown Rating", inplace=True)
df["duration"].fillna("Unknown Duration", inplace=True)
df["date_added"].fillna(0, inplace=True)
```

```
#There is no null value
df.isna().sum()
```

```
show_id             0
type                0
title               0
director        50643
cast             2146
country         11897
date_added        158
release_year        0
rating             67
duration            3
listed_in           0
description         0
dtype: int64
```

```
df.shape
```

```
(201991, 17)
```

```
df.ndim
```

```
2
```

Non-Graphical Analysis: Value counts and unique attributes

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

```
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
74 min          1
84 min          1
66 min          1
Name: rating, dtype: int64
```

```
df["cast"].nunique()
```

```
36439
```

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

```
127
```

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

```
42
```

```
df["release_year"].nunique()
```

```
74
```

```
df["rating"].nunique()
```

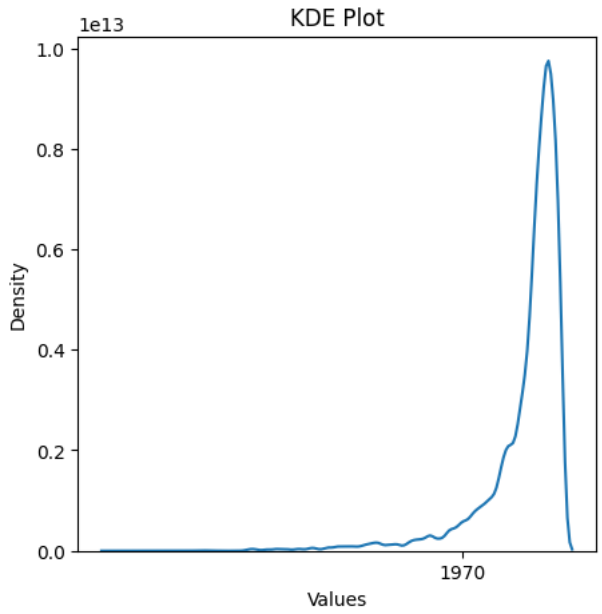
```
17
```

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

```
8807
```

Visual Analysis - Univariate, Bivariate after pre-processing of the data

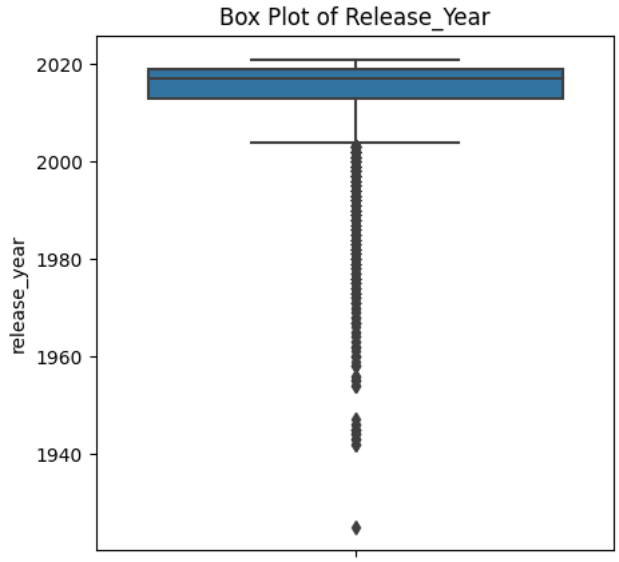
```
plt.figure(figsize=(5, 5))
sns.kdeplot(df['release_year'])
plt.title('KDE Plot')
plt.xlabel('Values')
plt.ylabel('Density')
plt.show()
```



Outlier

```
 # Plot box plots for numerical columns to identify outliers
plt.figure(figsize=(5, 5))
sns.boxplot(y=df['release_year'])
plt.title("Box Plot of Release_Year")
```

```
plt.xticks(rotation=0)
plt.show()
```



Heat Map

```
df.corr()
```

```
<ipython-input-159-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_on
  df.corr()
```

	release_year
release_year	1.0

```
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(),cmap='Blues', annot=True)
plt.title('Heatmap - Correlation Matrix')
plt.show()
```

```
<ipython-input-162-13f7ea8763d3>:2: FutureWarning: The default value of numeric_on
  sns.heatmap(df.corr(),cmap='Blues', annot=True)
```

1. Find the counts of each categorical variable both using graphical and non- graphical analysis.

a. For Non-graphical Analysis:

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

```
Movie      145843
TV Show     56148
Name: type, dtype: int64
```

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

```
Martin Scorsese        419
Youssef Chahine        409
Cathy Garcia-Molina    356
Steven Spielberg       355
Lars von Trier         336
                      ...
Richard Maurice          1
Richard E. Norman        1
Spencer Williams         1
Oscar Micheaux           1
Kirsten Johnson          1
Name: director, Length: 4993, dtype: int64
```

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

```
Liam Neeson             161
Alfred Molina           160
John Krasinski          139
Salma Hayek             130
Frank Langella          128
                       ...
Doug Averill              1
Lance Lewman              1
Ashleigh Aston Moore      1
Nanao                     1
Emily Rios                1
Name: cast, Length: 36439, dtype: int64
```

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

```
United States     59349
India             22814
United Kingdom    12945
Japan              8679
France             8254
                  ...
Palestine             2
Kazakhstan            1
Nicaragua             1
United States,        1
Uganda                1
Name: country, Length: 127, dtype: int64
```

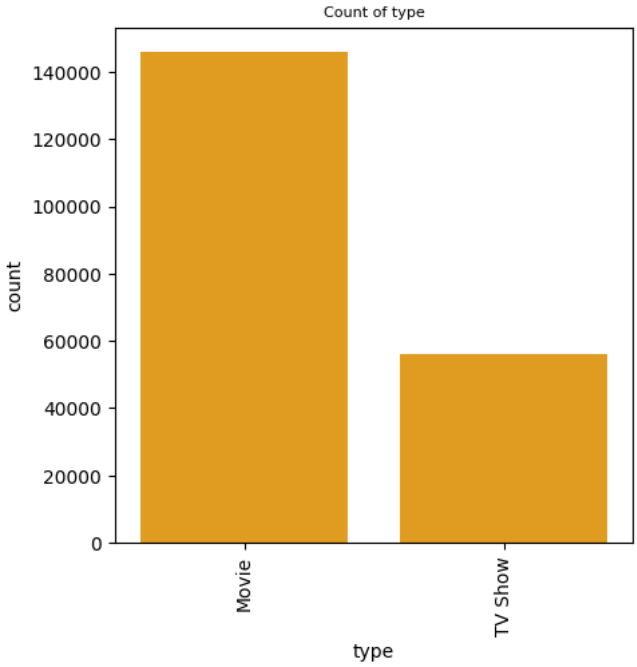
```
df['listed_in'].value_counts()
```

```
Dramas                          29775
International Movies            28211
Comedies                        20829
International TV Shows          12845
Action & Adventure              12216
Independent Movies               9834
Children & Family Movies         9771
TV Dramas                        8942
Thrillers                        7107
Romantic Movies                  6412
TV Comedies                      4963
Crime TV Shows                   4733
Horror Movies                    4571
Kids' TV                         4568
Sci-Fi & Fantasy                 4037
Music & Musicals                 3077
Romantic TV Shows                3049
Documentaries                    2407
Anime Series                     2313
TV Action & Adventure            2288
Spanish-Language TV Shows        2126
British TV Shows                 1808
Sports Movies                    1531
```

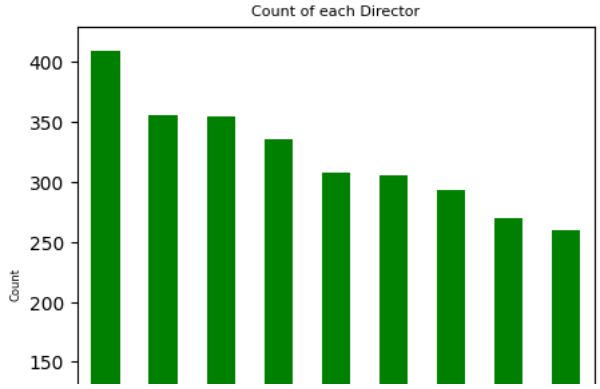
```
Classic Movies                  1434
TV Mysteries                    1281
Korean TV Shows                 1122
Cult Movies                     1077
TV Sci-Fi & Fantasy             1045
Anime Features                  1045
TV Horror                        941
Docuseries                       845
LGBTQ Movies                     838
TV Thrillers                     768
Teen TV Shows                    742
Reality TV                       735
Faith & Spirituality             719
Stand-Up Comedy                  540
Movies                           412
TV Shows                         337
Classic & Cult TV                272
Stand-Up Comedy & Talk Shows     268
Science & Nature TV              157
Name: listed_in, dtype: int64
```

b. For graphical analysis:

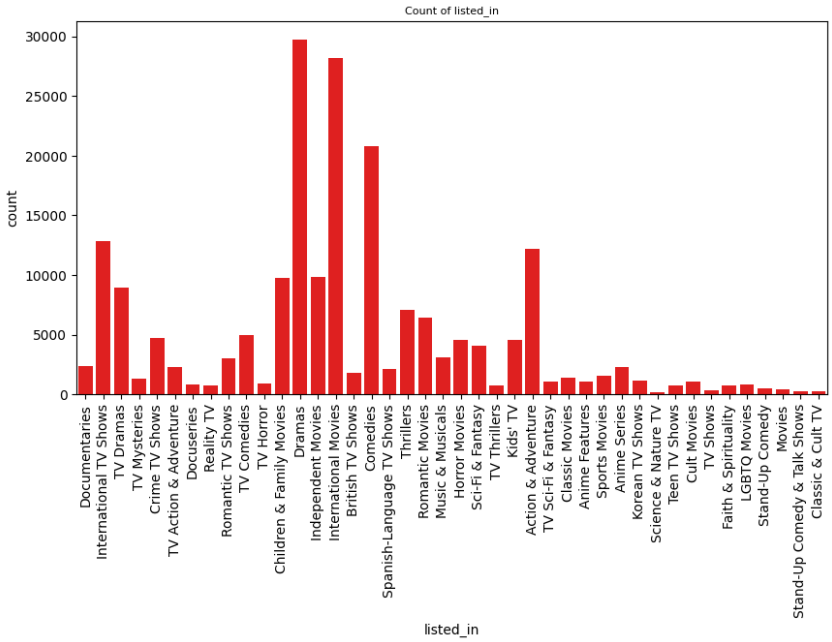
```
plt.figure(figsize=(5,5))
sns.countplot(x= 'type', data=df, color= 'orange')
plt.xticks(rotation=90)
plt.title('Count of type', fontsize=8)
plt.show()
```



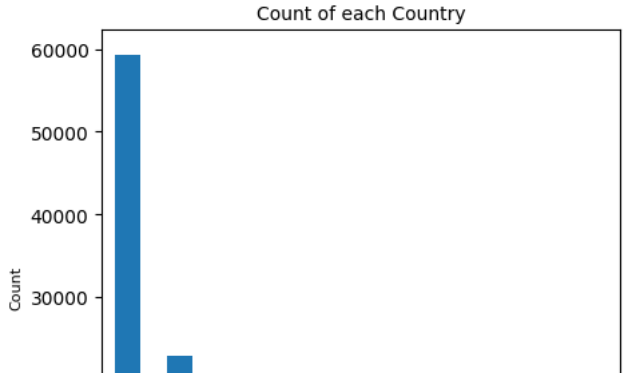

```
top_10_director=df['director'].value_counts().head(10)
plt.figure(figsize=(5,5))
top_10_director[1:10].plot(kind='bar', color='g')
plt.xlabel('Directors', fontsize=6)
plt.ylabel('Count', fontsize=6)
plt.title('Count of each Director', fontsize=8)
plt.show()
```




```
plt.figure(figsize=(10,5))
sns.countplot(x= 'listed_in', data=df, color= 'r')
plt.title('Count of listed_in', fontsize=8)
plt.xticks(rotation=90)
plt.show()
```




```
plt.figure(figsize=(5, 5))
top_10_country= df['country'].value_counts().head(10)
top_10_country.plot(kind='bar')
plt.xlabel('Countries', fontsize=8)
plt.ylabel('Count', fontsize=8)
plt.title('Count of each Country', fontsize=10)
plt.show()
```



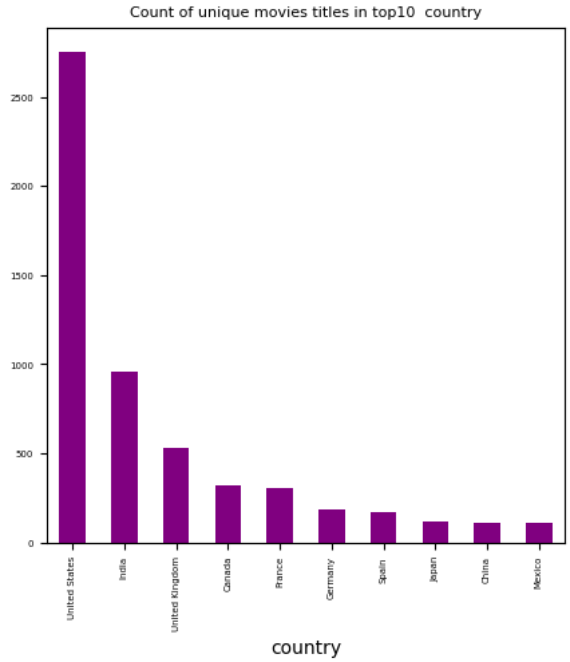
2. Comparison of tv shows vs. movies.

a. Find the number of movies produced in each country and pick the top 10 countries.

```
## Group by "country" and calculated the count of unique titles of movies produced in each country and picked up the top 10 countries
df_movies = df[df['type'] == 'Movie']
top10_Movies_country = df_movies.groupby('country')['title'].nunique().sort_values(ascending=False).head(10)
top10_Movies_country
```

```
country
United States     2751
India              962
United Kingdom     532
Canada             319
France             303
Germany            182
Spain              171
Japan              119
China              114
Mexico             111
Name: title, dtype: int64
```

```
plt.figure(figsize=(5, 5))
top10_Movies_country.plot(kind='bar', color='purple')
plt.title('Count of unique movies titles in top10  country', fontsize=8)
plt.xticks(fontsize=5)
plt.yticks(fontsize=5)
plt.show()
```

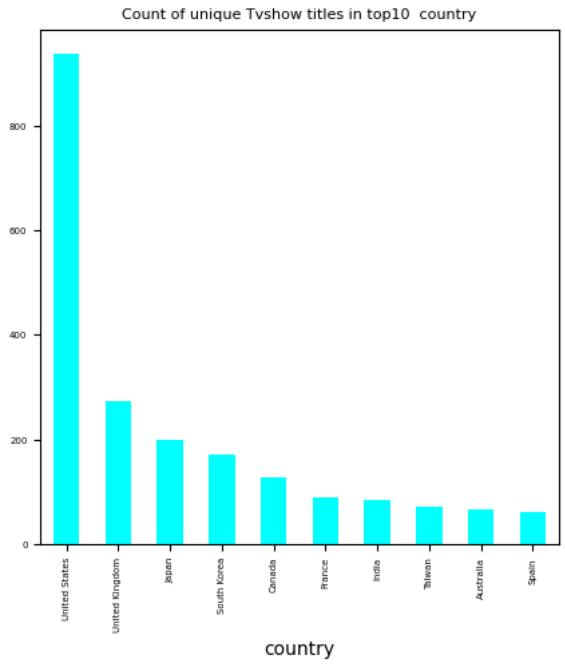


b. Find the number of Tv-Shows produced in each country and pick the top 10 countries.

```
df_TV_Show = df[df['type'] == 'TV Show']
top10_TV_Show_Country = df_TV_Show.groupby('country')['title'].nunique().sort_values(ascending=False).head(10)
top10_TV_Show_Country
```

```
country
United States     938
United Kingdom    272
Japan             199
South Korea       170
Canada            126
France             90
India              84
Taiwan             70
Australia          66
Spain              61
Name: title, dtype: int64
```

```
plt.figure(figsize=(5, 5))
top10_TV_Show_Country.plot(kind='bar', color='cyan')
plt.title('Count of unique Tvshow titles in top10  country', fontsize=8)
plt.xticks(fontsize=5)
plt.yticks(fontsize=5)
plt.show()
```



3. What is the best time to launch a TV show?

```
#created new column for timestamp,month,week
df['Timestamp']=pd.to_datetime(df['date_added'])
df['Month']=df['Timestamp'].dt.month
df['Month']=df['Month'].astype('Int64')
df['Week']=df['Timestamp'].dt.isocalendar().week
df.head()
```

	show_id	type	title	director	cast	country	date_added	release_year	r
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	P
1	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	September 24, 2021	2021	T

a. Find which is the best week to release the Tv-show or the movie. Do the analysis separately for Tv-shows and Movies

Best week to release tv show/Movie

```
Tv_show_weekly_count=df[df['type'] == 'TV Show']["Week"].value_counts()
Tv_show_weekly_count
```

```
27    1977
35    1945
24    1702
26    1662
31    1646
13    1554
48    1513
5     1386
44    1380
18    1364
40    1362
15    1230
51    1194
50    1182
19    1181
33    1180
46    1155
22    1150
52    1130
38    1128
37    1127
49    1074
53    1071
21    1044
1     1018
12    1002
7     1001
8      976
32     968
23     964
20     939
11     927
42     900
25     896
36     879
17     864
45     855
34     834
14     828
9      826
2      812
29     797
4      788
41     764
10     743
39     743
30     731
47     678
6      613
3      601
28     586
43     566
16     554
Name: Week, dtype: Int64
```

```
#Best week to release the Tv-show is week-27
Tv_show_weekly_count.idxmax()
```

```
27
```

```
Movie_weekly_count = df[df['type'] == 'Movie']['Week'].value_counts()
Movie_weekly_count
```

```
1     8456
44    5563
9     5094
35    5048
26    4931
40    4878
31    4388
27    3808
48    3737
18    3686
13    3503
39    3502
30    3262
22    3237
23    3164
5     3148
15    3083
28    2744
7     2636
17    2627
14    2609
```

```
36    2585
25    2568
37    2559
43    2521
10    2515
50    2463
33    2418
29    2335
34    2332
16    2323
51    2276
11    2225
49    2181
42    2105
38    2086
3     2031
24    1920
52    1840
20    1829
41    1807
47    1740
6     1649
19    1630
2     1618
21    1606
8     1538
46    1519
12    1431
53    1413
45    1396
32    1233
4     1047
Name: Week, dtype: Int64
```

```
#Best week to release the movie is week 1
Movie_weekly_count.idxmax()
```

```
1
```

b. Find which is the best month to release the Tv-show or the movie. Do the analysis separately for Tv-shows and Movies

Best month to release tv show/Movie

```
Tv_show_monthly_count= df[df['type'] == 'TV Show']['Month'].value_counts()
Tv_show_monthly_count
```

```
12    5498
7     5227
8     5162
6     5043
9     4900
4     4543
11    4532
3     4352
1     4307
10    4255
5     4248
2     3923
Name: Month, dtype: Int64
```

```
#Best month to release Tv show is "12"
Tv_show_monthly_count.idxmax()
```

```
12
```

```
Movie_monthly_counts = df[df['type'] == 'Movie']['Month'].value_counts()
Movie_monthly_counts
```

```
7     15049
1     13947
10    13514
9     13219
12    12768
4     12538
8     11924
6     11616
3     11489
11    11063
5      9579
2      9137
Name: Month, dtype: Int64
```

```
#Best month to release Movie is "7"
Movie_monthly_counts.idxmax()
```

```
7
```

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

a. Identify the top 10 Actors who have appeared in most movies or TV shows.

```
df_TV_Show = df[df['type'] == 'TV Show']
df_movies = df[df['type'] == 'Movie']
```

```
Top10_Actor_TV_Show=df_TV_Show.groupby('cast')['title'].nunique().sort_values(ascending=False).head(10).reset_index(name="TV_show title")
Top10_Actor_Movie=df_movies.groupby('cast')['title'].nunique().sort_values(ascending=False).head(10).reset_index(name="Movie Title")
```

```
#Top 10 actors appeared in most TV shows where we group by each actor thereby finding the count of unique titles of Tv-shows
Top10_Actor_TV_Show
```

	cast	TV_show title
0	Takahiro Sakurai	25
1	Yuki Kaji	19
2	Junichi Suwabe	17
3	Daisuke Ono	17
4	Ai Kayano	17
5	Yuichi Nakamura	16
6	Yoshimasa Hosoya	15
7	Jun Fukuyama	15
8	David Attenborough	14
9	Kana Hanazawa	13

```
#Top 10 actors appeared in most movies where we group by each actor thereby finding the count of unique titles of movies
Top10_Actor_Movie
```

	cast	Movie Title
0	Anupam Kher	42
1	Shah Rukh Khan	35
2	Naseeruddin Shah	32
3	Akshay Kumar	30
4	Om Puri	30
5	Paresh Rawal	28
6	Amitabh Bachchan	28
7	Julie Tejwani	28
8	Boman Irani	27
9	Rupa Bhimani	27

b. Identify the top 10 directors who have appeared in most movies or TV shows.

```
Top10_Director_TV_Show=df_TV_Show.groupby('director')['title'].nunique().sort_values(ascending=False).head(10).reset_index(name="TV_show
Top10_Director_Movie=df_movies.groupby('director')['title'].nunique().sort_values(ascending=False).head(10).reset_index(name="Movie Title
```

```
#Top 10 Directors appeared in most TV shows where we group by each actor thereby finding the count of unique titles of Tv-shows
Top10_Director_TV_Show
```

	director	TV_show title
0	Ken Burns	3
1	Alastair Fothergill	3
2	Stan Lathan	2
3	Jung-ah Im	2
4	Joe Berlinger	2
5	Hsu Fu-chun	2
6	Gautham Vasudev Menon	2
7	Lynn Novick	2

```
##Top 10 Directors appeared in most movies where we group by each actor thereby finding the count of unique titles of movies
Top10_Director_Movie
```

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

5. Which genre movies are more popular or produced more

```
from wordcloud import WordCloud


df_genre=df['listed_in'].value_counts()
df_genre.head(10)
```

```
Dramas                      29775
International Movies        28211
Comedies                    20829
International TV Shows      12845
Action & Adventure          12216
Independent Movies           9834
Children & Family Movies     9771
TV Dramas                    8942
Thrillers                    7107
Romantic Movies              6412
Name: listed_in, dtype: int64
```

```
#Drama genre are more popular or produced more
df_genre.head(10).plot(kind='pie')
```

```
<Axes: ylabel='listed in'>
```

Using WordCloud the Drama genre are more popular

```
genre=df['listed_in'].value_counts().to_dict()
```

```
wc = WordCloud().generate_from_frequencies(genre)
plt.imshow(wc)
plt.show()
```

6. Find After how many days the movie will be added to Netflix after the release of the movie (you can consider the recent past data)

```
#It takes 18261 days for the movie will be added to Netflix after the release of the movie.
df['date_added'] = pd.to_datetime(df['date_added'])
df['release_year']=pd.to_datetime(df['release_year'])
df['days_to_added']=(df['date_added']-df['release_year']).dt.days
mode_of_difference=df['days_to_added'].mode().astype('Int64')
mode_of_difference
```

```
0    18261
Name: days_to_added, dtype: Int64
```