```python
# Importing libraries for our purpose.
#print(pd.__version__)
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Reading the netflix csv file
netflix_data = pd.read_csv('netflix.csv')
netflix_data.head()
```

```
{"summary":"{\n  \"name\": \"netflix_data\",\n  \"rows\": 8807,\n
\"fields\": [\n    {\n      \"column\": \"show_id\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 8807,\n        \"samples\": [\n
\"s4971\",\n          \"s3363\",\n          \"s5495\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"type\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"TV Show\",\n          \"Movie\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"title\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 8807,\n        \"samples\": [\n          \"Game
Over, Man!\",\n          \"Arsenio Hall: Smart & Classy\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"director\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4528,\n        \"samples\": [\n
\"Kanwal Sethi\",\n          \"R\\u00e9my Four, Julien War\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"cast\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 7692,\n        \"samples\": [\n          \"Tzi
Ma, Christine Ko, Hong-Chi Lee, Hayden Szeto, Kunjue Li, Fiona Fu,
James Saito, Joan Chen\",\n          \"Priyanshu Painyuli,
Chandrachoor Rai, Shadab Kamal, Rajeev Siddhartha, Sheetal Thakur,
Ninad Kamat, Swati Semwal, Eijaz Khan\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"country\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
748,\n        \"samples\": [\n          \"United States, United
Kingdom, Denmark, Sweden\",\n          \"United Kingdom, Hong Kong\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"date_added\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 1767,\n        \"samples\": [\n
\"October 22, 2018\",\n          \"January 29, 2021\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"release_year\",\n
```

\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
8,\n   \"min\": 1925,\n   \"max\": 2021,\n
\"num_unique_values\": 74,\n   \"samples\": [\n   1996,\n
1969\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"rating\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 17,\n
\"samples\": [\n   \"PG-13\",\n   \"TV-MA\"\n   ],\
n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"duration\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 220,\n   \"samples\": [\n   \"37
min\",\n   \"177 min\"\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"listed_in\",\n   \"properties\": {\n
\"dtype\": \"category\",\n   \"num_unique_values\": 514,\n
\"samples\": [\n   \"Crime TV Shows, International TV Shows, TV
Mysteries\",\n   \"Children & Family Movies, Classic Movies,
Dramas\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"description\",\n   \"properties\": {\n   \"dtype\":
\"string\",\n   \"num_unique_values\": 8775,\n
\"samples\": [\n   \"A heedless teen drifter who falls for a
small-town waitress makes the mistake of robbing a drug lord, putting
his life and newfound love in jeopardy.\",\n   \"Twelve-year-
old Calvin manages to join the navy and serves in the battle of
Guadalcanal. But when his age is revealed, the boy is sent to the
brig.\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"netflix_data"}

```python
# Checking Null values in all the columns.
netflix_data.isna().sum().sort_values(ascending=False)
```

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

```python
# Checking data type of all the columns.
netflix_data.dtypes
```

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

```python
# Checking unique values in all the columns.
for i in netflix_data.columns:
  print(i, ':', netflix_data[i].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
```

```python
# Percentage of null values in our Dataset
null_percentage = (netflix_data.isna().sum()/len(netflix_data)*100) \
.sort_values(ascending=False)
formatted_percentage = null_percentage.apply(lambda x: f"{x:.2f}%")
formatted_percentage
```

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

```python
# Un nesting the columns # df.loc[:,col] --> Not to get confused
SettingWithCopyWarning

def un_nest_col(df, col):
  df = df.dropna(subset=[col])
  df.loc[:, col] = df[col].str.split(', ')
  return df.explode(col).reset_index(drop=True)

# These 4 columns have multiple values in a column,so we need to
explode and make it to one.
categorical_columns = ['director','cast','country','listed_in']

# taking copy of netflix data(Deep Copy)
netflix_data_copy = netflix_data.copy()

for col in categorical_columns:
  netflix_data = un_nest_col(netflix_data, col)

# Checking number of rows after exploding
netflix_data.shape, netflix_data_copy.shape
```

```
((143102, 12), (8807, 12))
```

```python
# Displaying first 10 rows after exploding
netflix_data.head(10)
```

{"type":"dataframe","variable_name":"netflix_data"}

```python
netflix_data.isna().sum().sort_values(ascending=False)
```

```
rating          7
duration        3
show_id         0
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
listed_in       0
description     0
dtype: int64
```

```python
filtered_data = ['74 min','66 min','84 min']

# Here if we observe we have duration values present in rating column.
netflix_data[netflix_data['rating'].isin(filtered_data)]
```

{"repr_error":"0","type":"dataframe"}

```
# Taking a copy and assign it to mask.
# COPY()--> Pandas not to get confused with whether to update to
# original DF or to mask.
mask = netflix_data[netflix_data['rating'].isin(filtered_data)].copy()
```

```
mask
```

{"repr_error":"0","type":"dataframe","variable_name":"mask"}

```
mask['rating']
```

```
80168    74 min
83632    84 min
83661    66 min
Name: rating, dtype: object
```

```
mask['duration'] = mask['rating']
```

```
mask
```

{"repr_error":"0","type":"dataframe","variable_name":"mask"}

```
mask['rating'] = 'NR'
```

```
mask
```

{"repr_error":"0","type":"dataframe","variable_name":"mask"}

```
netflix_data.update(mask)
```

```
netflix_data.loc[netflix_data['rating'] == 'NR']
```

{"repr_error":"0","type":"dataframe"}

```
# Filling null values in Rating column with NR--> 'No rating'
netflix_data['rating'].fillna('NR', inplace=True)
```

```
<ipython-input-138-2eeb7bad725b>:2: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  netflix_data['rating'].fillna('NR', inplace=True)
```

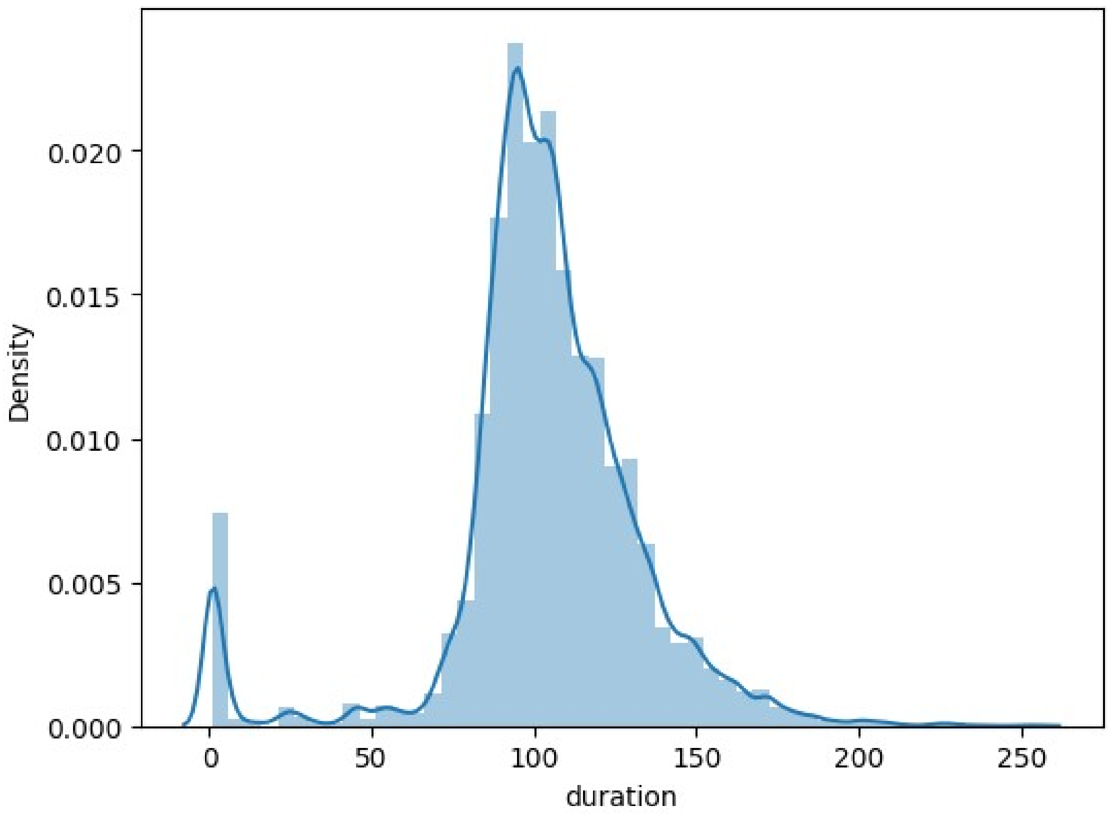
```
# checking for null values.
netflix_data.isna().sum().sum()

0

# Removing 'min' from duration column, left with only number
netflix_data['duration'] = netflix_data['duration'].str.replace('
min','')

netflix_data
```

{"type":"dataframe","variable_name":"netflix_data"}

```
netflix_data['duration'] = netflix_data['duration']. \
str.split('          ').str[0]

# netflix_data.to_csv('netflix_updated.csv', index=False)

# from google.colab import files
# files.download('netflix_updated.csv')

netflix_data.loc[netflix_data['duration'].str.contains('Season',
case=False, \
na=False), 'duration'] = netflix_data['duration']. \
apply(lambda x: x.split(' ')[0] if 'Season' in x else x)
```

Univariate Analysis

```
# Frequency of movies/series for different duration ranges.
sns.distplot(netflix_data['duration'], kde=True, hist=True)
plt.show()

<ipython-input-146-158f7ef5ab10>:2: 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(netflix_data['duration'], kde=True, hist=True)
```

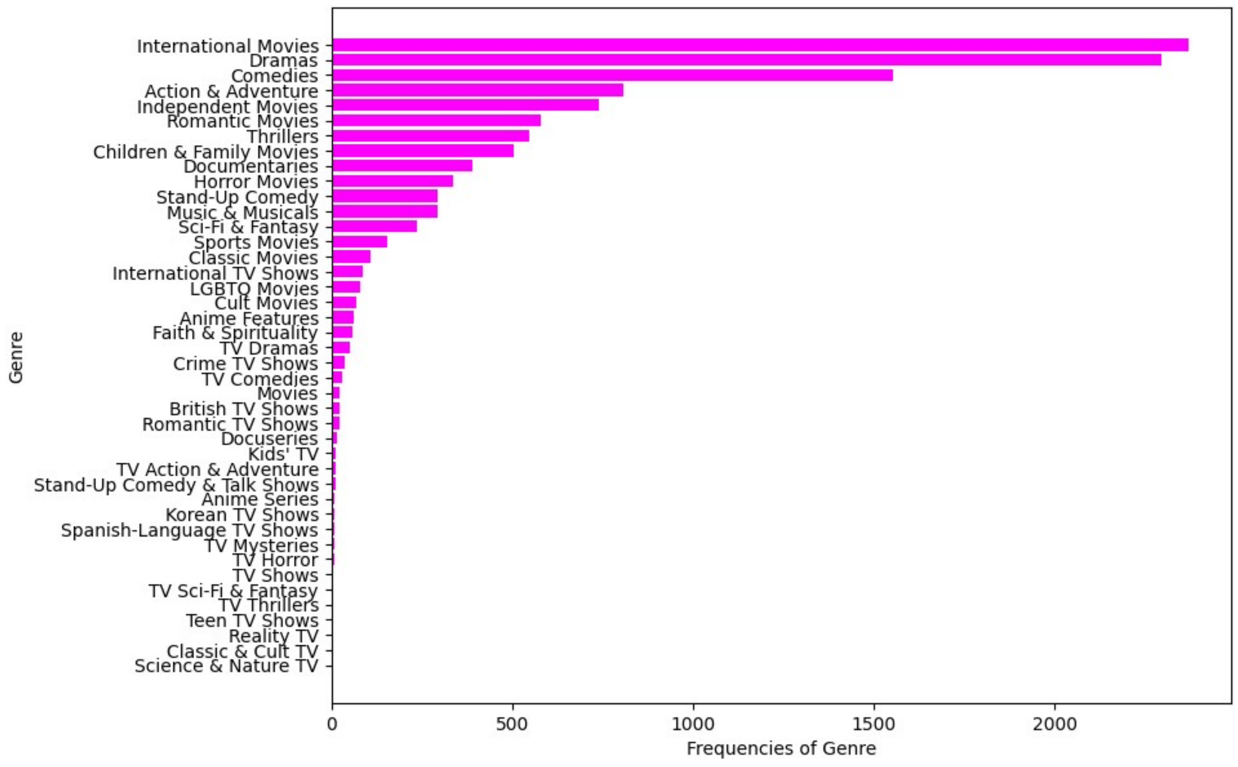


```
# Number of distinct titles on the basis of genre(Listed_in)
netflix_data.groupby('listed_in')['title'].nunique(). \
sort_values(ascending=False)
```

```
listed_in
International Movies            2369
Dramas                          2294
Comedies                        1553
Action & Adventure               806
Independent Movies               740
Romantic Movies                  579
Thrillers                        547
Children & Family Movies         503
Documentaries                    391
Horror Movies                    336
Stand-Up Comedy                  294
Music & Musicals                 292
Sci-Fi & Fantasy                 236
Sports Movies                    156
Classic Movies                   108
International TV Shows            87
LGBTQ Movies                      80
Cult Movies                       69
Anime Features                    61
```

```
Faith & Spirituality                 58
TV Dramas                            52
Crime TV Shows                       38
TV Comedies                          30
Movies                               23
British TV Shows                     21
Romantic TV Shows                    21
Docuseries                           14
Kids' TV                             13
TV Action & Adventure                13
Stand-Up Comedy & Talk Shows         11
Anime Series                         10
Korean TV Shows                      10
Spanish-Language TV Shows            10
TV Mysteries                          8
TV Horror                             7
TV Shows                              5
TV Sci-Fi & Fantasy                   4
TV Thrillers                          3
Teen TV Shows                         3
Reality TV                            3
Classic & Cult TV                     3
Science & Nature TV                   1
Name: title, dtype: int64
```

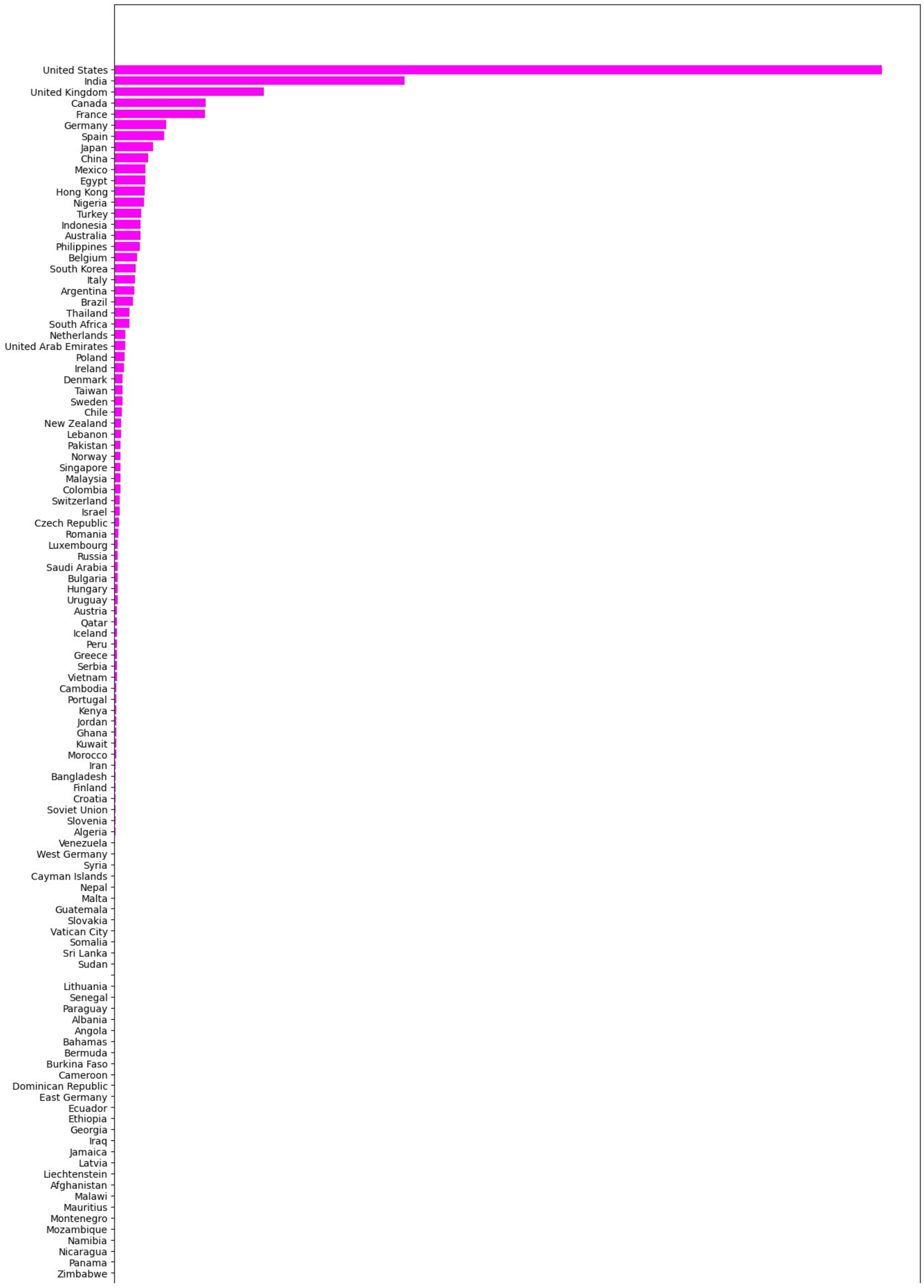
```python
new_netflix_genre = netflix_data.groupby(['listed_in']) \
.agg({'title':'nunique'}).reset_index().sort_values(by='title',ascendi
ng=False)

# Frequency of Movies or TV Shows across Genre
plt.figure(figsize=(9,7))
plt.barh(new_netflix_genre[::-1]['listed_in'], new_netflix_genre[::-1]
\
['title'],color=['magenta'])
plt.xlabel('Frequencies of Genre', size=10)
plt.ylabel('Genre')
plt.show()
```



Insights: International movies, Dramas, Comedies are most popular.

```
netflix_data['country'] = netflix_data['country'].str.replace(',','')

new_data_country =
netflix_data.groupby('country').agg({'title':'nunique'}) \
.reset_index().sort_values(by='title',ascending=False)

plt.figure(figsize=(15,25))
plt.barh(new_data_country[::-1]['country'], new_data_country[::-1] \
['title'], color='magenta')
```

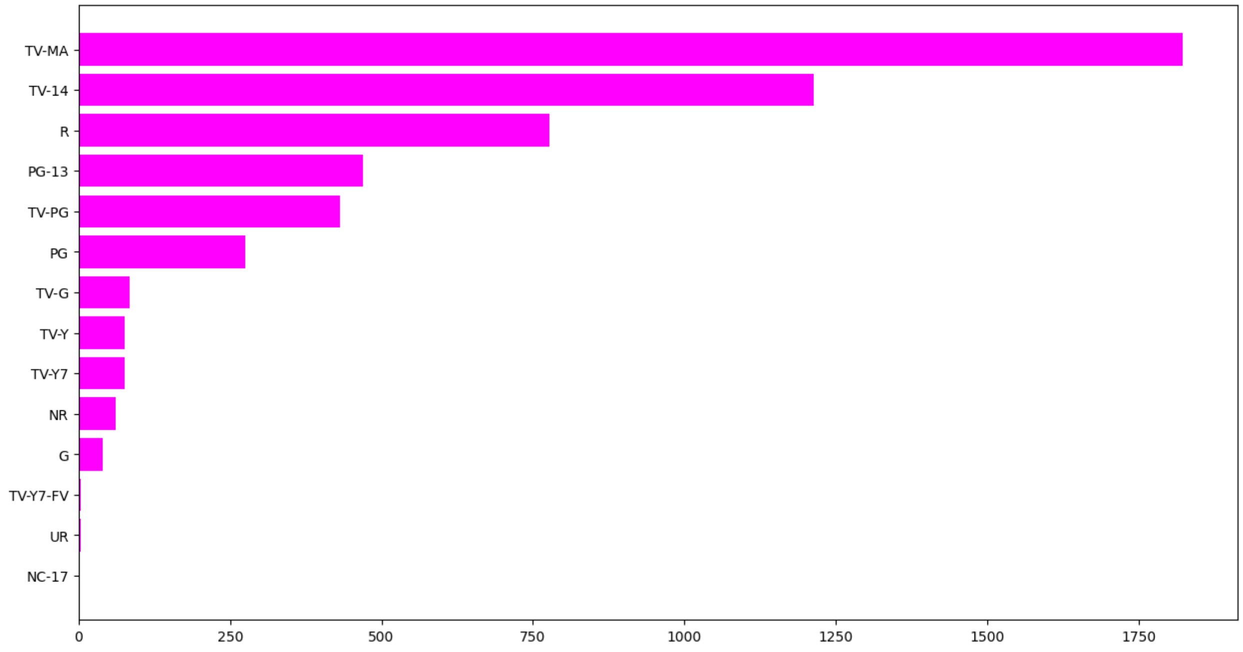
```
<BarContainer object of 110 artists>
```


United States
India
United Kingdom
Canada
France
Germany
Spain
Japan
China
Mexico
Egypt
Hong Kong
Nigeria
Turkey
Indonesia
Australia
Philippines
Belgium
South Korea
Italy
Argentina
Brazil
Thailand
South Africa
Netherlands
United Arab Emirates
Poland
Ireland
Denmark
Taiwan
Sweden
Chile
New Zealand
Lebanon
Pakistan
Norway
Singapore
Malaysia
Colombia
Switzerland
Israel
Czech Republic
Romania
Luxembourg
Russia
Saudi Arabia
Bulgaria
Hungary
Uruguay
Austria
Qatar
Iceland
Peru
Greece
Serbia
Vietnam
Cambodia
Portugal
Kenya
Jordan
Ghana
Kuwait
Morocco
Iran
Bangladesh
Finland
Croatia
Soviet Union
Slovenia
Algeria
Venezuela
West Germany
Syria
Cayman Islands
Nepal
Malta
Guatemala
Slovakia
Vatican City
Somalia
Sri Lanka
Sudan
Lithuania
Senegal
Paraguay
Albania
Angola
Bahamas
Bermuda
Burkina Faso
Cameroon
Dominican Republic
East Germany
Ecuador
Ethiopia
Georgia
Iraq
Jamaica
Latvia
Liechtenstein
Afghanistan
Malawi
Mauritius
Montenegro
Mozambique
Namibia
Nicaragua
Panama
Zimbabwe

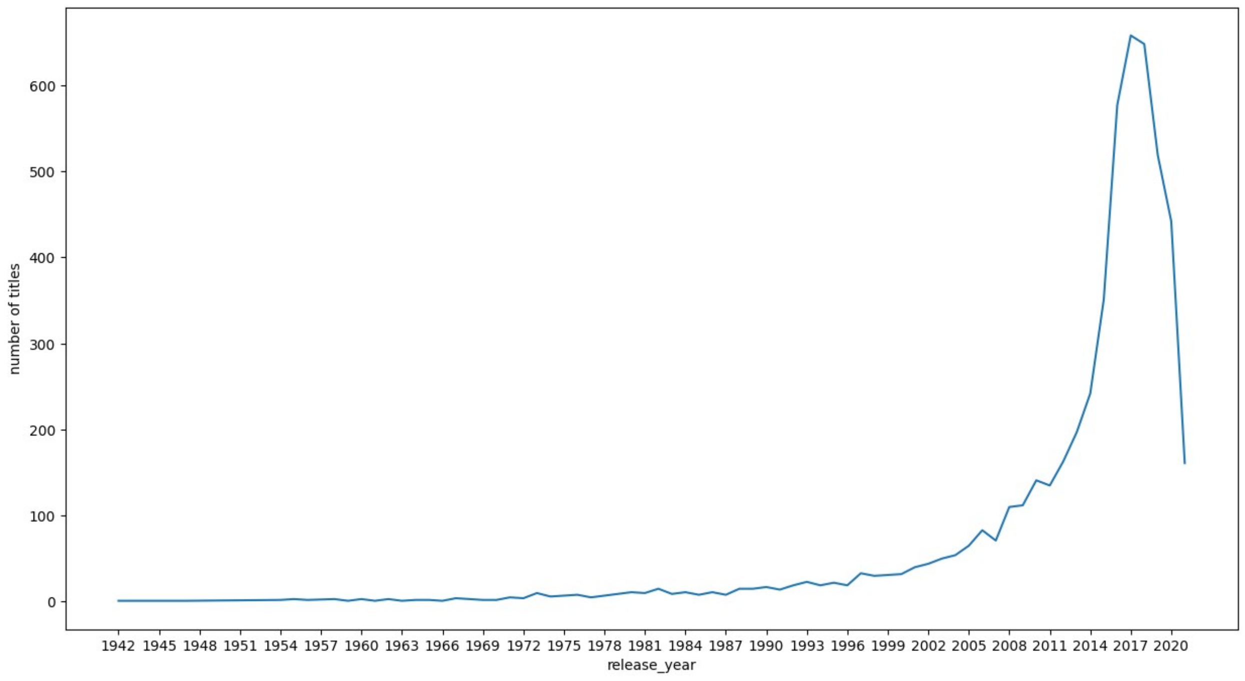
United States, India, United Kingdom, Canada and France are leading countries in content creation on Netflix

```
#number of distinct titles on the basis of rating

new_data_rating =
netflix_data.groupby('rating').agg({'title':'nunique'}) \
.reset_index().sort_values(by='title', ascending=False)

plt.figure(figsize=(15,8))
plt.barh(new_data_rating[::-1]['rating'], new_data_rating[::-1] \
['title'], color='magenta')
```

```
<BarContainer object of 14 artists>
```



```
# number of distinct titles on the basis of year

new_data_year =
netflix_data.groupby('release_year').agg({'title':'nunique'}) \
.reset_index().sort_values(by='release_year', ascending=False)

plt.figure(figsize=(15,8))
sns.lineplot(data=new_data_year, x='release_year',y='title')
plt.xticks(np.arange(new_data_year['release_year'].min(), \
new_data_year['release_year'].max()+1, 3))
plt.ylabel('number of titles')
plt.show()
```



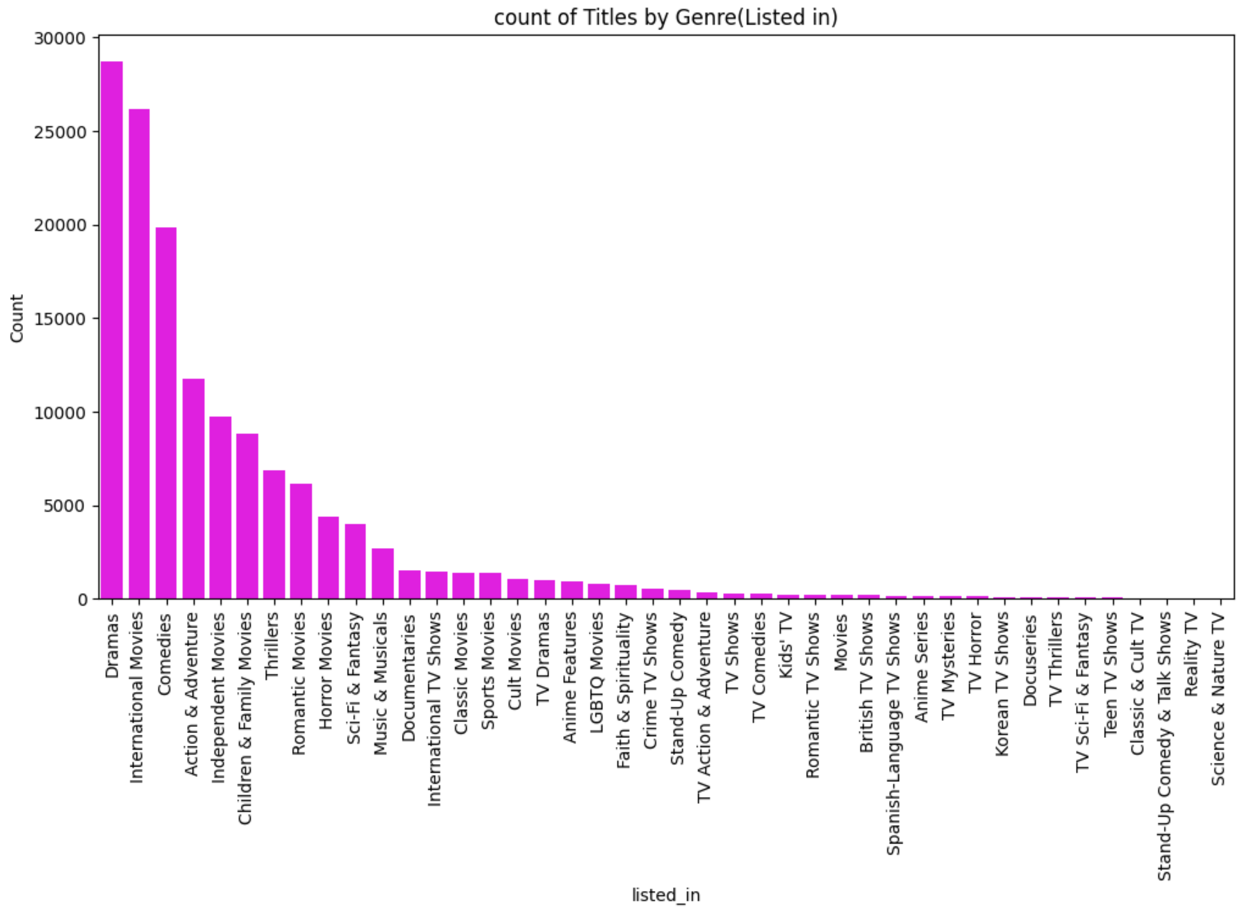
The amount of Content across Nteflix has increased from 1942 to 2020. Then started decreasing from 2017(may be due to Covid).

```
# counts of each categorical variable both using graphical and
# non-graphical analysis.
netflix_categorical =
['director','cast','country','listed_in','rating','type']
for col in netflix_categorical:
  print(f'\nvalue counts of column {col}')
  print(netflix_data[col].value_counts())


value counts of column director
director
Martin Scorsese        419
Youssef Chahine        409
Cathy Garcia-Molina    356
Steven Spielberg       355
Lars von Trier         336
                      ...
Julián Gaviria           1
Greg Whiteley            1
Rachel Lears             1
Wyatt Cenac              1
Daniel McCabe            1
Name: count, Length: 4313, dtype: int64

value counts of column cast
cast
Liam Neeson           161
```

```
Alfred Molina         154
John Krasinski        138
Salma Hayek           130
Frank Langella        128
                     ...
Stephanie Honoré        1
Andrew Fiscella         1
Shantel VanSanten       1
Bobby Campo             1
Charlotte McKinney      1
Name: count, Length: 25465, dtype: int64

value counts of column country
country
United States     45773
India             21762
United Kingdom     9010
France             6816
Canada             5793
                  ...
Vatican City          3
Sri Lanka             2
Afghanistan           2
Panama                2
Nicaragua             1
Name: count, Length: 110, dtype: int64

value counts of column listed_in
listed_in
Dramas                                28698
International Movies                  26191
Comedies                              19834
Action & Adventure                    11761
Independent Movies                     9712
Children & Family Movies               8796
Thrillers                              6857
Romantic Movies                        6145
Horror Movies                          4416
Sci-Fi & Fantasy                       3980
Music & Musicals                       2717
Documentaries                          1492
International TV Shows                 1450
Classic Movies                         1407
Sports Movies                          1389
Cult Movies                            1071
TV Dramas                              1023
Anime Features                          934
LGBTQ Movies                            769
Faith & Spirituality                    699
```

```
Crime TV Shows                            547
Stand-Up Comedy                           443
TV Action & Adventure                     308
TV Shows                                  286
TV Comedies                               265
Kids' TV                                  236
Romantic TV Shows                         232
Movies                                    231
British TV Shows                          226
Spanish-Language TV Shows                 174
Anime Series                              128
TV Mysteries                              123
TV Horror                                 119
Korean TV Shows                           111
Docuseries                                 87
TV Thrillers                               78
TV Sci-Fi & Fantasy                        51
Teen TV Shows                              48
Classic & Cult TV                          29
Stand-Up Comedy & Talk Shows               25
Reality TV                                  7
Science & Nature TV                         7
Name: count, dtype: int64

value counts of column rating
rating
TV-MA       44931
TV-14       27720
R           25593
PG-13       16078
PG          10802
TV-PG        9870
TV-Y7        1926
TV-Y         1574
G            1528
TV-G         1441
NR           1360
NC-17         137
UR             86
TV-Y7-FV       56
Name: count, dtype: int64

value counts of column type
type
Movie      137542
TV Show      5560
Name: count, dtype: int64
```

```
plt.figure(figsize=(12, 6))
sns.countplot(data=netflix_data, x='listed_in', \
order=netflix_data['listed_in'].value_counts().index,color='magenta')
plt.ylabel('Count')
plt.title('count of Titles by Genre(Listed in)')
plt.xticks(rotation=90)
plt.show()
```



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

```
netflix_movies = netflix_data.loc[netflix_data['type']=='Movie']

netflix_data.groupby('country').agg({'title':'count'}) \
.sort_values(by='title',ascending=False).head(10)
```

{"summary":"{\n \"name\": \"netflix_data\",\n \"rows\": 10,\n
\"fields\": [\n   {\n   \"column\": \"country\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 10,\n   \"samples\": [\n
\"China\",\n   \"India\",\n   \"Japan\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"title\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 13663,\n
\"min\": 2341,\n   \"max\": 45773,\n

```
\"num_unique_values\": 10,\n        \"samples\": [\n          2399,\n
21762,\n          3818\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

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

```
netflix_tv_show = netflix_data.loc[netflix_data['type'] == 'TV Show']

netflix_tv_show.groupby(by='country')['title'].agg('count')\
.sort_values(ascending=False).head(10)

country
United States     894
United Kingdom    596
Taiwan            495
India             494
South Korea       358
Spain             357
Japan             307
Thailand          285
France            263
Canada            166
Name: title, dtype: int64
```

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

For TV Shows

```
netflix_data.head(2)

{"type":"dataframe","variable_name":"netflix_data"}

# Converted data type of Date_added column to date_time format
netflix_data['date_added'] = pd.to_datetime(netflix_data['date_added']
\
                                            ,errors='coerce')

netflix_tv_show.head(3)
```

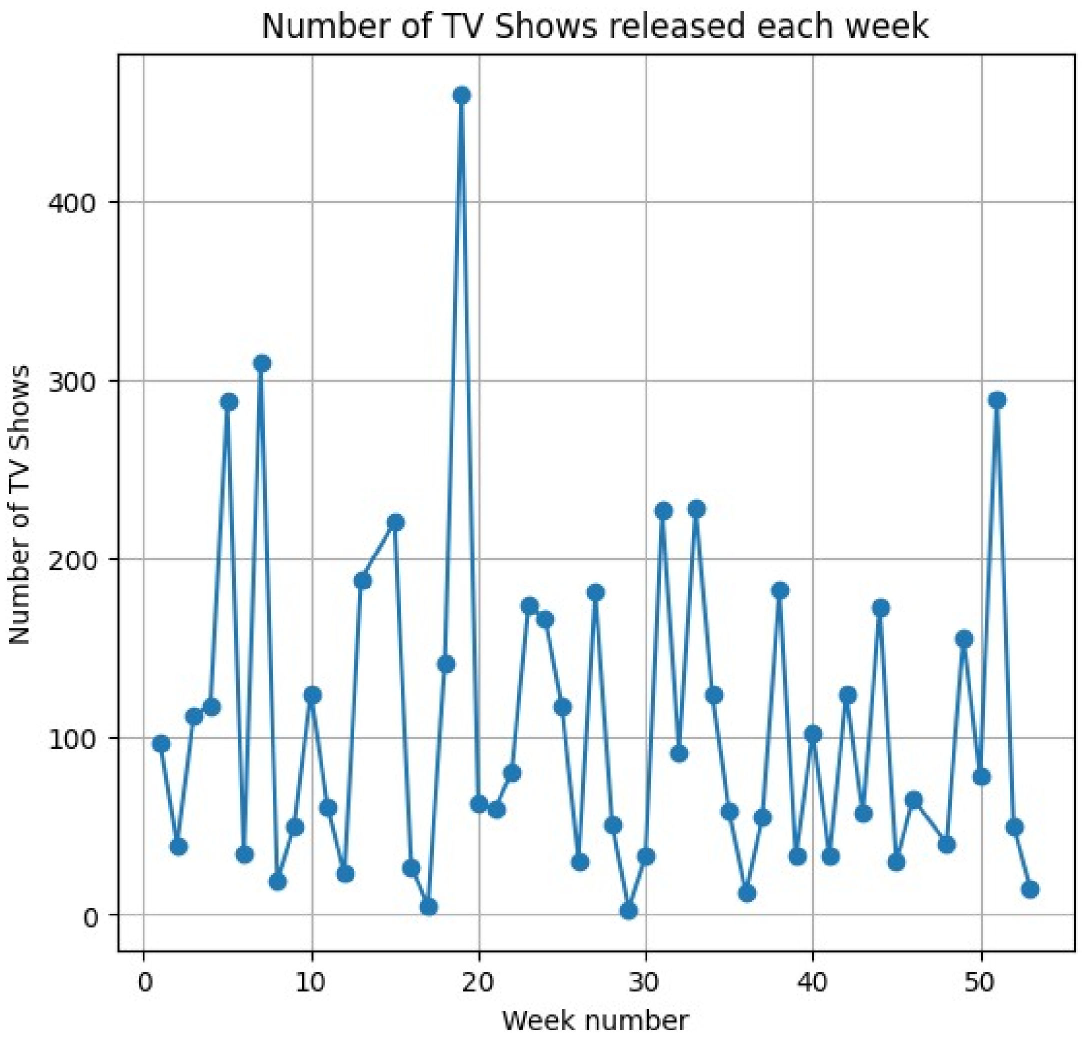
```
{"summary":"{\n  \"name\": \"netflix_tv_show\",\n  \"rows\": 5560,\n
\"fields\": [\n    {\n      \"column\": \"show_id\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 147,\n        \"samples\": [\n
\"s6811\",\n          \"s2554\",\n          \"s7749\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"type\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 1,\n
\"samples\": [\n          \"TV Show\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"title\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\":
147,\n        \"samples\": [\n          \"Frozen Planet\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"director\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 191,\n        \"samples\": [\n
\"Olivier Jean-Marie\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"cast\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 1324,\n        \"samples\": [\n          \"Kim
Tae-hun\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"country\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 34,\n
\"samples\": [\n          \"Brazil\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"date_added\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 134,\n        \"samples\": [\n          \"April
22, 2015\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"release_year\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 3,\n        \"min\": 1990,\n
\"max\": 2021,\n        \"num_unique_values\": 15,\n
\"samples\": [\n          2016\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"rating\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          \"TV-14\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"duration\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 10,\n
\"samples\": [\n          \"8\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"listed_in\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 22,\n        \"samples\": [\n
\"British TV Shows\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"description\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 147,\n
\"samples\": [\n          \"Go on a journey through the Arctic and
Antarctic with this visually stunning program that explores these
wildernesses and their inhabitants.\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"netflix_tv_show"}
```

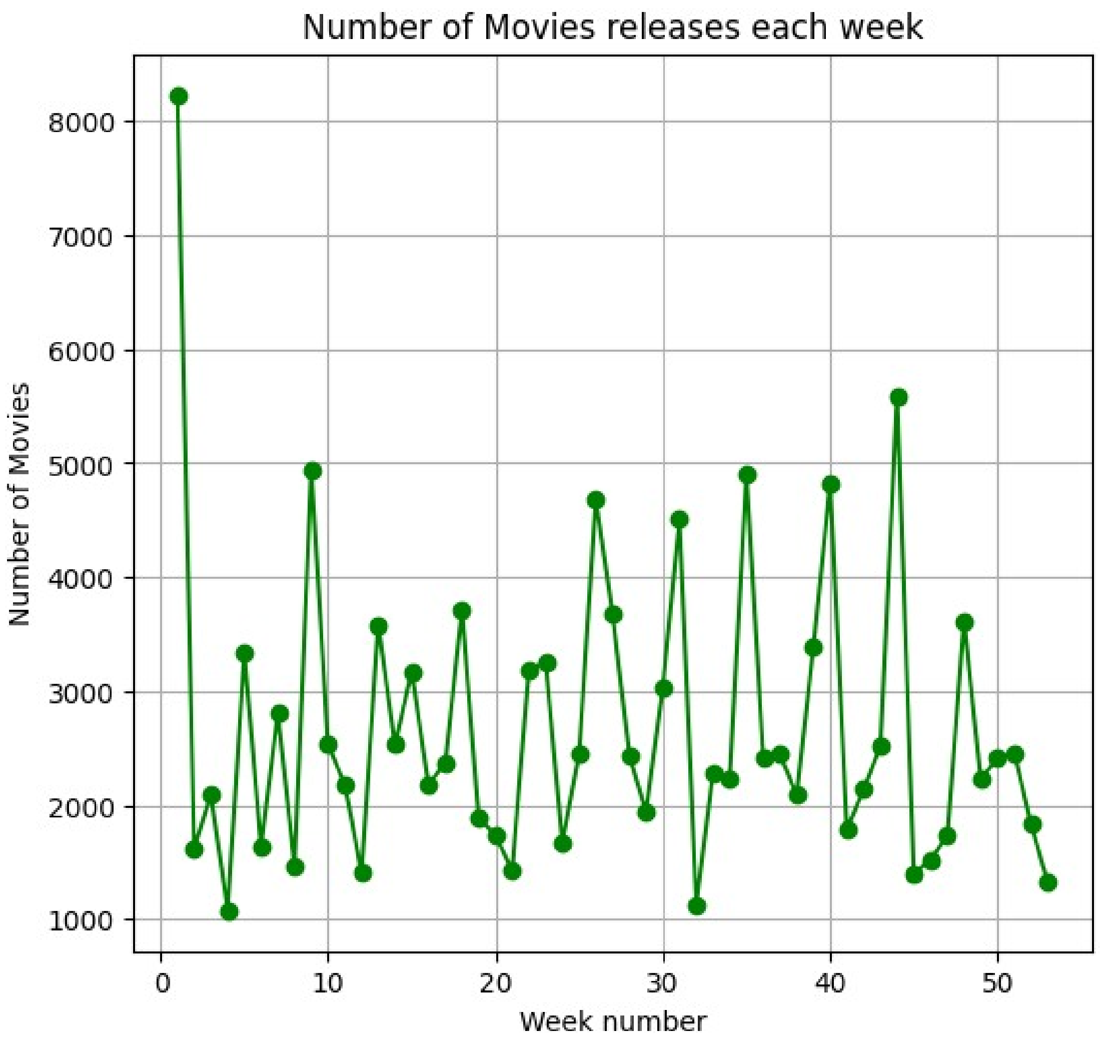
```
# Extracted the week of which this particular data is added
netflix_data['week'] =
netflix_data['date_added'].dt.isocalendar().week

netflix_TVshow = netflix_data.loc[netflix_data['type'] == 'TV Show']

tv_show_per_week = netflix_TVshow.groupby(by='week')['title'] \
.apply('count').reset_index()
tv_show_per_week.columns = ['week','No of TV shows']

tv_show_per_week.head(2)
```

{"summary":"{\n \"name\": \"tv_show_per_week\",\n \"rows\": 51,\n \"fields\": [\n   {\n   \"column\": \"week\",\n   \"properties\": {\n   \"dtype\": \"UInt32\",\n   \"num_unique_values\": 51,\n   \"samples\": [\n   45,\n   42,\n   49\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"No of TV shows\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 93,\n   \"min\": 3,\n   \"max\": 460,\n   \"num_unique_values\": 45,\n   \"samples\": [\n   65,\n   51,\n   3\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"tv_show_per_week"}

```
movies_per_week = netflix_data.groupby('week')['title'] \
.apply('count').reset_index()
movies_per_week.columns = ['week','No of Movies']



plt.figure(figsize=(14,6))
plt.subplot(1,2,1)
plt.plot(tv_show_per_week['week'],tv_show_per_week['No of TV
shows'],marker='o')
plt.title('Number of TV Shows released each week')
plt.xlabel('Week number')
plt.ylabel('Number of TV Shows')
plt.grid(True)

plt.figure(figsize=(14,6))
plt.subplot(1,2,2)
plt.plot(movies_per_week['week'],movies_per_week['No of Movies'] \
         ,marker='o',color='green')
plt.title('Number of Movies releases each week')
plt.xlabel('Week number')
plt.ylabel('Number of Movies')
plt.grid(True)
```

```
#plt.tight_layout()
plt.show()
```

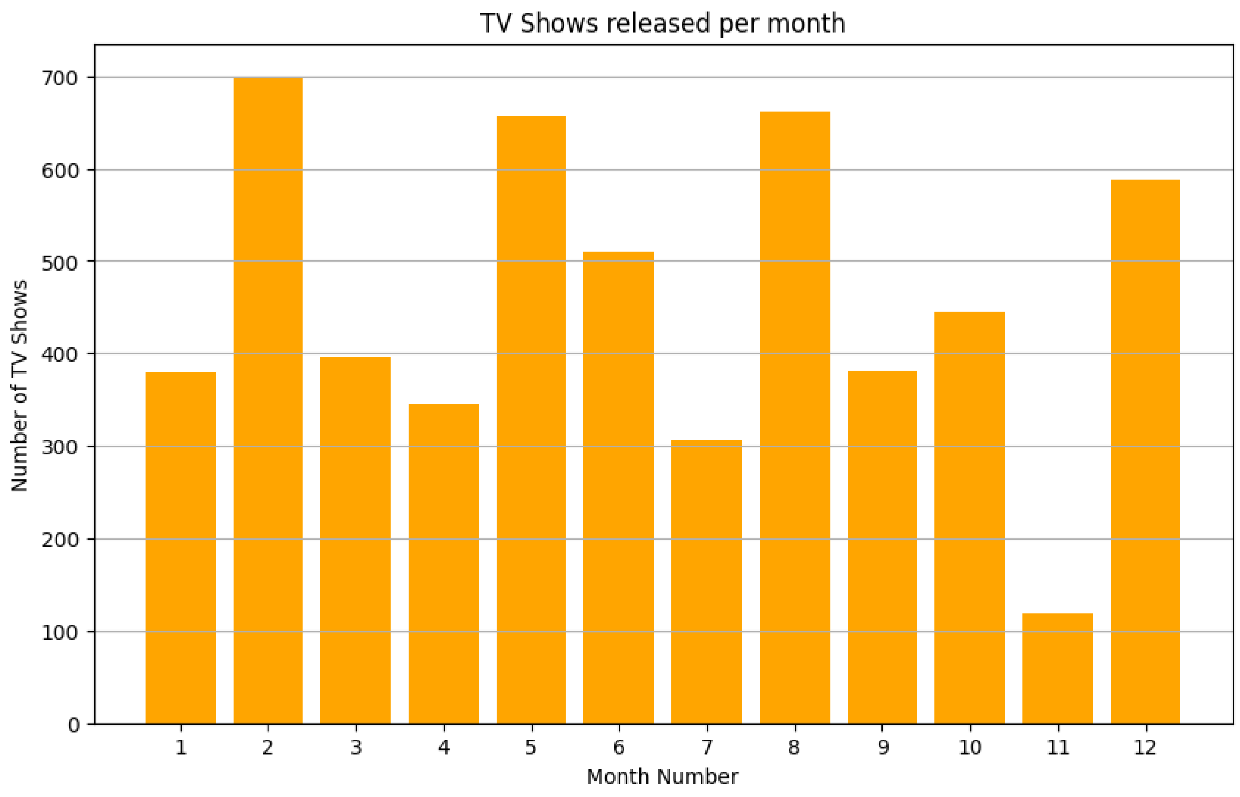




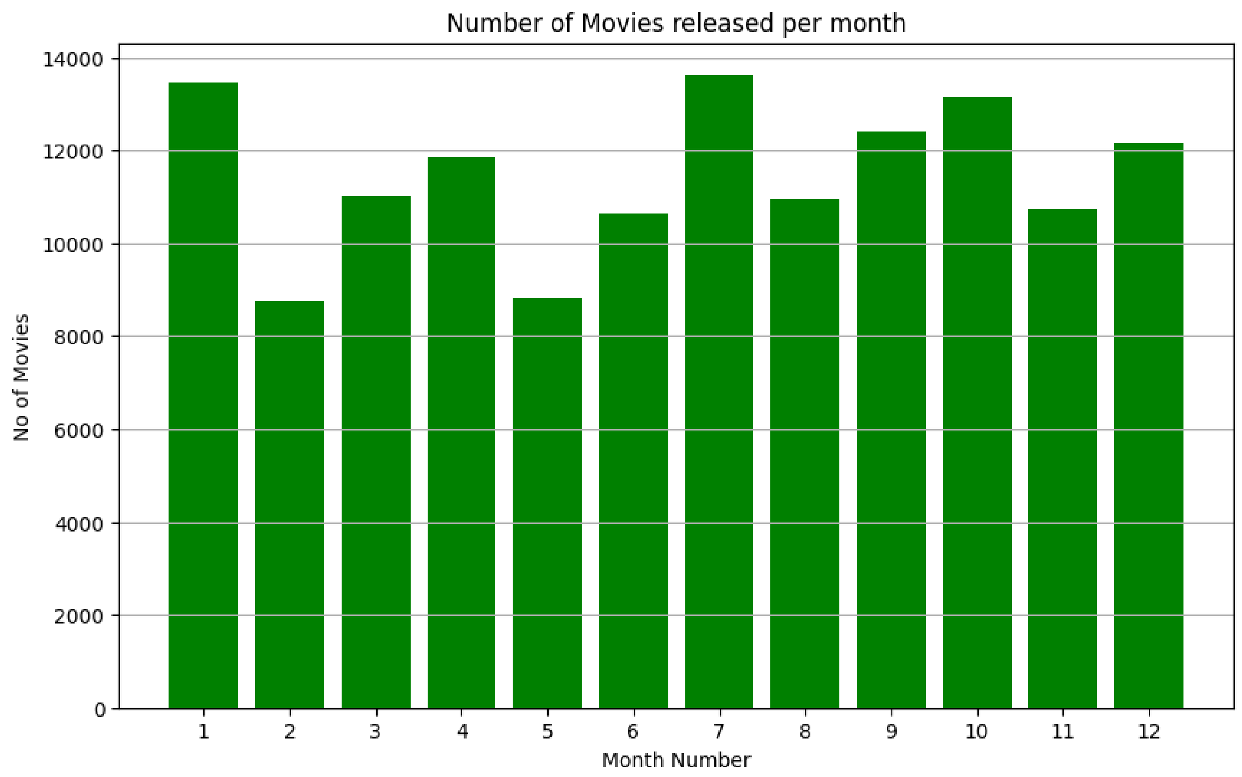
Find which is the best month to release the Tv-show or the movie. Do the analysis separately for Tv-shows and Movies

```
netflix_data['month'] = netflix_data['date_added'].dt.month

netflix_movies_month = netflix_data.loc[netflix_data['type'] ==
'Movie']

netflix_tv_show_month = netflix_data.loc[netflix_data['type'] == 'TV
Show']

tv_shows_per_month = netflix_tv_show_month.groupby(by='month')
['title'] \
.apply('count').reset_index()
tv_shows_per_month.columns = ['month','No of TV Shows']

movies_per_month = netflix_movies_month.groupby('month')['title'] \
.apply('count').reset_index()
movies_per_month.columns = ['month','No of Movies']
```

```
plt.figure(figsize=(10,6))
plt.bar(tv_shows_per_month['month'],tv_shows_per_month['No of TV
Shows'] \
        ,color='orange')
plt.title('TV Shows released per month')
plt.xlabel('Month Number')
plt.ylabel('Number of TV Shows')
plt.grid(axis='y')
plt.xticks(range(1,13))
plt.show()
```

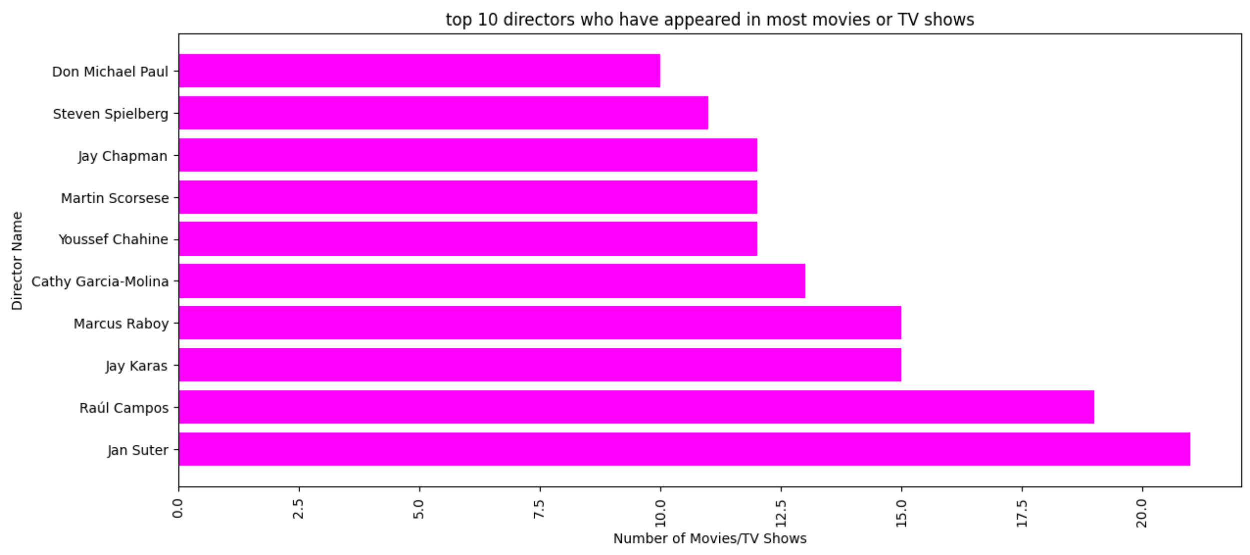


```
plt.figure(figsize=(10,6))
plt.bar(movies_per_month['month'],movies_per_month['No of Movies'], \
        color='green')
plt.grid(axis='y')
plt.xlabel('Month Number')
plt.ylabel('No of Movies')
plt.title('Number of Movies released per month')
plt.xticks(range(1,13))
plt.show()
```



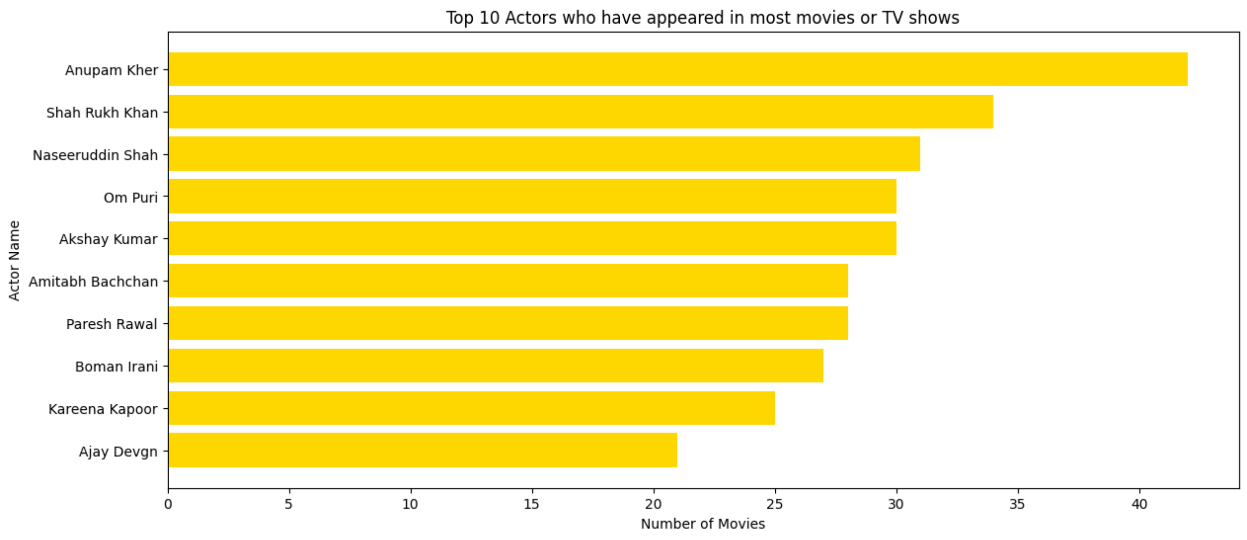
##Identify the top 10 directors who have appeared in most movies or TV shows.

```
netflix_data.head(2)

{"type":"dataframe","variable_name":"netflix_data"}

netflix_director = netflix_data.groupby(by='director')['title'] \
.nunique().reset_index()
netflix_director.columns = ['director','No of Titles']

top_10_directors = netflix_director.sort_values(by='No of Titles', \
                          ascending=False).head(10)

plt.figure(figsize=(14,6))
plt.barh(top_10_directors['director'],top_10_directors['No of Titles']
\
         ,color='magenta')
plt.xlabel('Number of Movies/TV Shows')
plt.ylabel('Director Name')
plt.title('top 10 directors who have appeared in most movies or TV
shows')
plt.xticks(rotation=90)
plt.show()
```



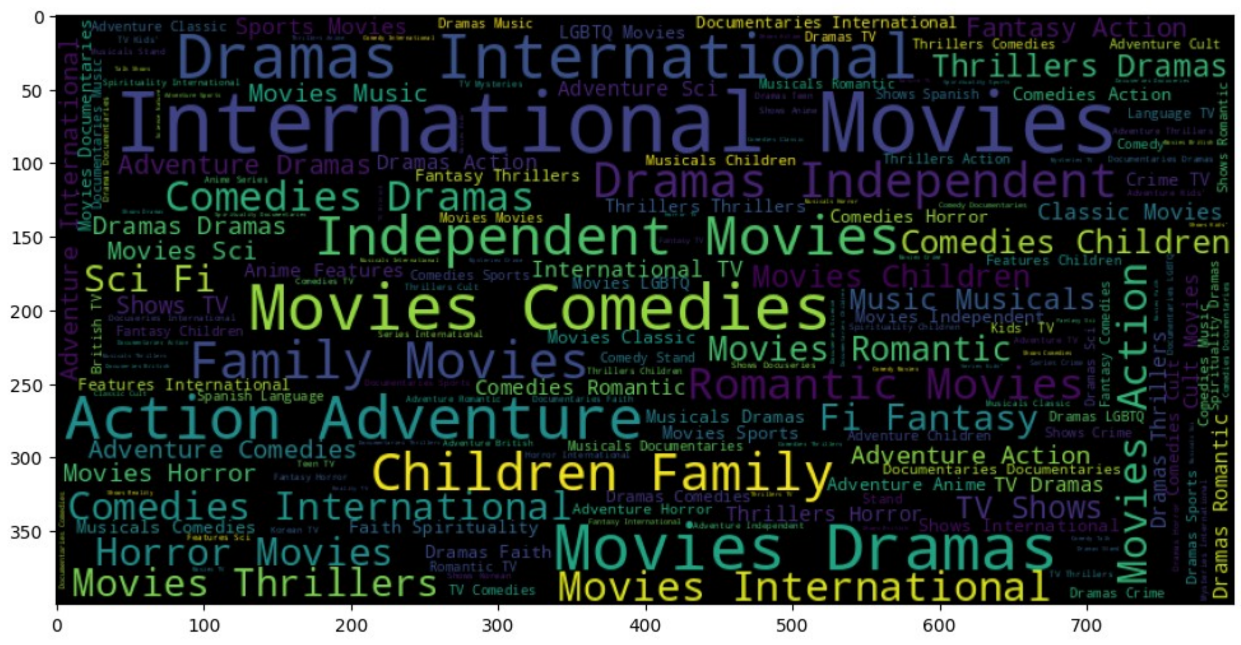
##Identify the top 10 directors who have appeared in most movies or TV shows.

```
netflix_actors = netflix_data.groupby(by='cast')['title'] \
.nunique().reset_index()
netflix_actors.columns = ['Actors','No_of_titles']

top_10_Actors = netflix_actors.sort_values(by='No_of_titles', \
                                           ascending=False).head(10)

plt.figure(figsize=(14,6))
plt.barh(top_10_Actors['Actors'][::-1],top_10_Actors['No_of_titles']
[::-1] \
         ,color='gold')
plt.title('Top 10 Actors who have appeared in most movies or TV
shows')
plt.xlabel('Number of Movies')
plt.ylabel('Actor Name')
plt.show()
```



Which genre movies are more popular or produced more

```
netflix_data['listed_in'] = netflix_data['listed_in'].astype(str)
genres_text = " ".join(netflix_data['listed_in'])

from wordcloud import WordCloud
plt.figure(figsize=(14,6))
word_cloud = WordCloud(width=800, height=400).generate(genres_text)
plt.imshow(word_cloud)
plt.show()
```


International Movies
Dramas International
Independent Movies
Movies Comedies
Action Adventure
Children Family
Movies Dramas
Family Movies
Comedies Dramas
Dramas Independent
Romantic Movies
Movies International
Comedies International
Horror Movies
Movies Thrillers
Sci Fi
Thrillers Dramas
TV Shows