```
# Importing essential libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Load CSV file into dataframe
df = pd.read_csv('netflix.csv')
```

```
# Having a glance into dataset for first 5 rows
df.head(5)
```

	show_id	type	title	director	cast	country	date_added	release_year	rat
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas,	NaN	September 24, 2021	2021	TV-

```
# Gathering more information on dataset to check non-null count and datatype of each column
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
```

```
# To identify no of rows and columns
df.shape
```

```
(8807, 12)
```

```
# To get overall statistical report on dataset
df.describe()
```

	release_year
count	8807.000000

```
# To get overall statistical report on dataset including objects
df.describe(include = 'object').T
```

	count	unique	top	freq
show_id	8807	8807	s1	1
type	8807	2	Movie	6131
title	8807	8807	Dick Johnson Is Dead	1
director	6173	4528	Rajiv Chilaka	19
cast	7982	7692	David Attenborough	19
country	7976	748	United States	2818
date_added	8797	1767	January 1, 2020	109
rating	8803	17	TV-MA	3207
duration	8804	220	1 Season	1793
listed_in	8807	514	Dramas, International Movies	362
description	8807	8775	Paranormal activity at a lush, abandoned prope...	4

```
# Checking percentage of missing data
df.isnull().sum()/len(df)*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
```

```
# Checking missing value for duration
df[df.isnull()['duration'] == True]
```

	show_id	type	title	director	cast	country	date_added	release_year	rating
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	2017	74 min
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	September 16, 2016	2010	84 min

```
# Replacing rating with duration value as it is interchanged
index = list(df[df.isnull()['duration'] == True].index)
df.loc[index, 'duration'] = df.loc[index, 'rating']
df.loc[index,'rating'] = np.nan
df.loc[index,:]
```

	show_id	type	title	director	cast	country	date_added	release_year	rating
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	2017	NaN
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	September 16, 2016	2010	NaN

```
# Rechecking percentage of missing data
df.isnull().sum()/len(df)*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           0.079482
duration         0.000000
listed_in        0.000000
description      0.000000
dtype: float64
```

```
# Checking missing value for rating
df[df.isnull()['rating'] == True]
```

	show_id	type	title	director	cast	country	date_added	release_yea
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	20
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	September 16, 2016	20
5813	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	August 15, 2016	20
5989	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava ...	NaN	Oprah Winfrey, Ava DuVernay	NaN	January 26, 2017	20
6827	s6828	TV Show	Gargantia on the Verdurous Planet	NaN	Kaito Ishikawa, Hisako Kanemoto, Ai Kayano, Ka...	Japan	December 1, 2016	20

```
# Impute missing 'rating' values

mode_rating = df['rating'].mode()[0]
df['rating'].fillna(mode_rating, inplace=True)
```

```
# Checking missing value for date_added
df[df.isnull()['date_added'] == True]
```

	show_id	type	title	director	cast	country	date_added	release_year
6066	s6067	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Daniel Radcliffe, Jon Hamm, Adam Godley, Chris...	United Kingdom	NaN	2013
6174	s6175	TV Show	Anthony Bourdain: Parts Unknown	NaN	Anthony Bourdain	United States	NaN	2018
		TV			Kelsey Grammer, Jane	United		

```
# Impute missing 'date_added' values as 1st Jan of release year
index = list(df[df.isnull()['date_added'] == True].index)
df.loc[index,'date_added'] = ['January 1, ' + str(i) for i in df.loc[index,'release_year']]
```

Jennifer

```
# Checking missing value for country
df[df.isnull()['country'] == True]
```

	show_id	type	title	director	cast	country	date_added	release_year
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	202
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	202
5	s6	TV Show	Midnight Mass	Mike Flanagan	Kate Siegel, Zach Gilford, Hamish Linklater, H...	NaN	September 24, 2021	202
6	s7	Movie	My Little Pony: A New Generation	Robert Cullen, José Luis Ucha	Vanessa Hudgens, Kimiko Glenn, James Marsden, ...	NaN	September 24, 2021	202
10	s11	TV Show	Vendetta: Truth, Lies and The Mafia	NaN	NaN	NaN	September 24, 2021	202
...	...	...	...	...	...	...	...	..
8718	s8719	Movie	Westside vs. the World	Michael Fahey	Ron Perlman, Louie Simmons	NaN	August 9, 2019	201

```
# Impute missing 'country' values based on director country
for i in df[df['country'].isnull()]['director'].unique():
    if i in df[~df['country'].isnull()]['director'].unique():
        imp=df[df['director']==i]['country'].mode().values[0]
        df.loc[df['director']==i,'country']=df.loc[df['director']==i,'country'].fillna(imp)
```

```
# Impute missing 'country' values based on cast country
for i in df[df['country'].isnull()]['cast'].unique():
    if i in df[~df['country'].isnull()]['cast'].unique():
```

```
    imp=df[df['cast']==i]['country'].mode().values[0]
    df.loc[df['cast']==i,'country']=df.loc[df['cast']==i,'country'].fillna(imp)
```

```
# Remaining values to be replace by Unknown Country
df['country'].fillna('Unknown Country',inplace=True)
df.isnull().sum()
```

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

```
# Impute missing 'director' values with the mode within each group
df['director'] = df.groupby(['title', 'listed_in', 'country'])['director'].transform(lambda x: x.fillna(x.mode().iloc[0] if not x.mode().empt
```

```
# Impute missing 'cast' values with the mode within each group
df['cast'] = df.groupby(['country', 'director', 'title', 'listed_in'])['cast'].transform(lambda x: x.fillna(x.mode().iloc[0] if not x.mode().
```

```
# Rechecking percentage of missing data
df.isnull().sum()/len(df)*100
```

```
show_id         0.0
type            0.0
title           0.0
director        0.0
cast            0.0
country         0.0
date_added      0.0
release_year    0.0
rating          0.0
duration        0.0
listed_in       0.0
description     0.0
dtype: float64
```

```
#unnesting the directors column

constraint1=df['director'].apply(lambda x:str(x).split(', ' )).tolist()
df_new1=pd.DataFrame(constraint1,index=df['title'])
df_new1=df_new1.stack()
df_new1=pd.DataFrame(df_new1.reset_index())
df_new1.rename(columns={0:'Directors'},inplace=True)
df_new1.drop(['level_1'],axis=1,inplace=True)
df_new1.head()
```

	title	Directors
0	Dick Johnson Is Dead	['Kirsten Johnson']
1	Blood & Water	['Unknown']
2	Ganglands	['Julien Leclercq']
3	Jailbirds New Orleans	['Unknown']
4	Kota Factory	['Unknown']

```
#unnesting the cast column,

constraint2=df['cast'].apply(lambda x: str(x).split(', ')).tolist()
df_new2=pd.DataFrame(constraint2,index=df['title'])
df_new2=df_new2.stack()
df_new2=pd.DataFrame(df_new2.reset_index())
df_new2.rename(columns={0:'cas'},inplace=True)
df_new2.drop(['level_1'],axis=1,inplace=True)
df_new2.head()
```

	title	Actors
0	Dick Johnson Is Dead	nan
1	Blood & Water	nan
2	Ganglands	nan
3	Jailbirds New Orleans	nan
4	Kota Factory	nan

```
#unnesting the listed_in column,

constraint3=df['listed_in'].apply(lambda x: str(x).split(', ')).tolist()
df_new3=pd.DataFrame(constraint3,index=df['title'])
df_new3=df_new3.stack()
df_new3=pd.DataFrame(df_new3.reset_index())
df_new3.rename(columns={0:'Genre'},inplace=True)
df_new3.drop(['level_1'],axis=1,inplace=True)
df_new3.head()
```

	title	Genre
0	Dick Johnson Is Dead	['Documentaries']
1	Blood & Water	['International TV Shows'
2	Blood & Water	' TV Dramas'
3	Blood & Water	' TV Mysteries']
4	Ganglands	['Crime TV Shows'

```
#unnesting the country column,

constraint4=df['country'].apply(lambda x: str(x).split(', ')).tolist()
df_new4=pd.DataFrame(constraint4,index=df['title'])
df_new4=df_new4.stack()
df_new4=pd.DataFrame(df_new4.reset_index())
df_new4.rename(columns={0:'country'},inplace=True)
df_new4.drop(['level_1'],axis=1,inplace=True)
df_new4.head()
```

	title	country
0	Dick Johnson Is Dead	['United States']
1	Blood & Water	['South Africa']
2	Ganglands	['France']
3	Jailbirds New Orleans	['Unknown Country']
4	Kota Factory	['India']

```
#merging the unnested director data with unnested cast data
df_new5=df_new2.merge(df_new1,on=['title'],how='inner')

#merging the above merge data with unnested listed_in data
df_new6=df_new5.merge(df_new3,on=['title'],how='inner')

#merging the above merged data with unnested country data
df_new=df_new6.merge(df_new4,on=['title'],how='inner')

#replacing nan values of director and actor by Unknown Actor and Director
df_new['Actors'].replace(['nan'],['Unknown Actor'],inplace=True)
df_new.head()
```

title	Actors	Directors	Genre	country

```
#merging our unnested data with the original data

df_final=df_new.merge(df[['show_id', 'type', 'title', 'date_added',
       'release_year', 'rating', 'duration']],on=['title'],how='left')
df_final.head()
```

	title	Actors	Directors	Genre	country	show_id	type	date_added
0	Dick Johnson Is Dead	Unknown Actor	['Kirsten Johnson']	['Documentaries']	['United States']	s1	Movie	September 25, 2021
1	Blood & Water	Unknown Actor	['Unknown']	['International TV Shows'	['South Africa']	s2	TV Show	September 24, 2021
2	Blood &	Unknown	['Unknown']	' TV Dramas'	['South	s2	TV	September

```
# Rechecking nulls
df_final.isnull().sum()
```

```
title           0
Actors          0
Directors       0
Genre           0
country         0
show_id         0
type            0
date_added      0
release_year    0
rating          0
duration        0
dtype: int64
```

```
# Adding extra coulmns for time, week, day, month analysis

from datetime import datetime
from dateutil.parser import parse
arr=[]
for i in df_final['date_added'].values:
    dt1=parse(i)
    arr.append(dt1.strftime('%Y-%m-%d'))
df_final['Modified_Added_date']=arr
df_final['Modified_Added_date']=pd.to_datetime(df_final['Modified_Added_date'])
df_final['month_added']=df_final['Modified_Added_date'].dt.month
df_final['week_Added']=df_final['Modified_Added_date'].dt.week
df_final['day_Added']=df_final['Modified_Added_date'].dt.day
df_final['year']=df_final['Modified_Added_date'].dt.year
df_final['Weekday_added'] = df_final['Modified_Added_date'].apply(lambda x: parse(str(x)).strftime("%A"))
df_final.head()
```

```
ca5>:10: FutureWarning: Series.dt.weekofyear and Series.dt.
final['Modified_Added_date'].dt.week
```

ctors	Genre	country	show_id	type	date_added	rel
irsten hnson	Documentaries	United States	s1	Movie	September 25, 2021	
nown	International TV Shows	South Africa	s2	TV Show	September 24, 2021	
nown	TV Dramas	South Africa	s2	TV Show	September 24, 2021	
nown	TV Mysteries	South Africa	s2	TV Show	September 24, 2021	
Julien clercq	Crime TV Shows	France	s3	TV Show	September 24, 2021	

```
# Removing jargons like ''[] from columns

df_final['title']=df_final['title'].str.replace(r"\(.*\)","")
df_final['Genre']=df_final['Genre'].str.replace(" Kids' TV", "Kids' TV")
```

```
df_final['country']=df_final['country'].str.replace(r"\(.*\)","")
df_final['Directors']=df_final['Directors'].str.replace(r"\(.*\)","")
df_final['Genre'] = df_final['Genre'].str.strip("['" "']")
df_final['country'] = df_final['country'].str.strip("['" "']")
df_final['Directors'] = df_final['Directors'].str.strip("['" "']")
df_final.head()
```

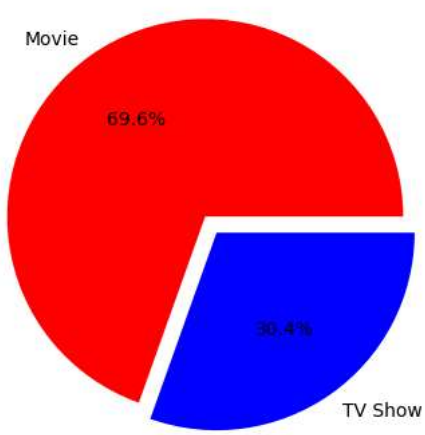
```
<ipython-input-97-cfb5d4f47f1c>:1: FutureWarning: The default value of regex will change
  df_final['title']=df_final['title'].str.replace(r"\(.*\)","")
<ipython-input-97-cfb5d4f47f1c>:3: FutureWarning: The default value of regex will change
  df_final['country']=df_final['country'].str.replace(r"\(.*\)","")
<ipython-input-97-cfb5d4f47f1c>:4: FutureWarning: The default value of regex will change
  df_final['Directors']=df_final['Directors'].str.replace(r"\(.*\)","")
```

	title	Actors	Directors	Genre	country	show_id	type	date_added	rel
0	Dick Johnson Is Dead	Unknown Actor	Kirsten Johnson	Documentaries	United States	s1	Movie	September 25, 2021	
1	Blood & Water	Unknown Actor	Unknown	International TV Shows	South Africa	s2	TV Show	September 24, 2021	
2	Blood & Water	Unknown Actor	Unknown	TV Dramas	South Africa	s2	TV Show	September 24, 2021	
3	Blood & Water	Unknown Actor	Unknown	TV Mysteries	South Africa	s2	TV Show	September 24, 2021	
4	Ganglands	Unknown Actor	Julien Leclercq	Crime TV Shows	France	s3	TV Show	September 24, 2021	

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

df_final.groupby(['type']).agg({"title":'nunique'})
```

type	title
Movie	6115
TV Show	2676

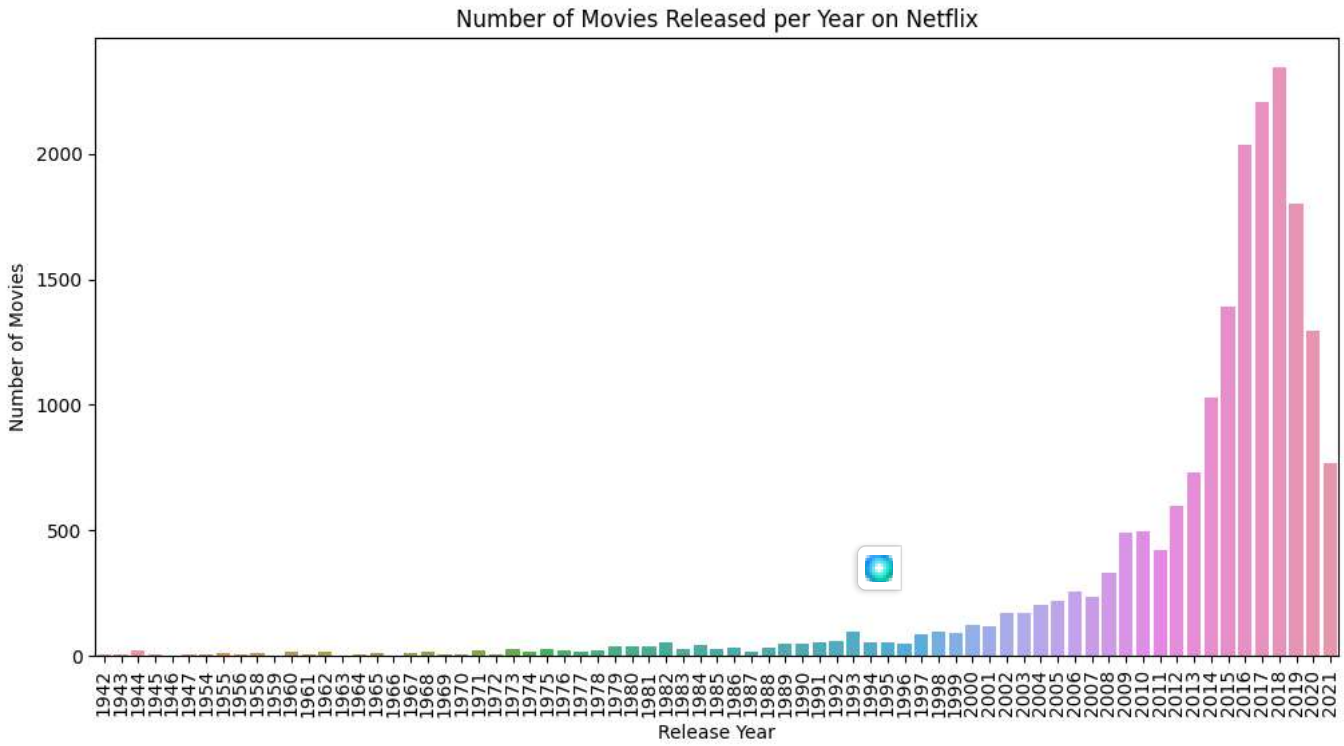
```
# Plotting pie chart for 'type' category

df_type=df_final.groupby(['type']).agg({"title":"nunique"}).reset_index()
plt.pie(df_type['title'],explode=(0.05,0.05),labels=df_type['type'],colors=['red','blue'],autopct='%.1f%%')
plt.show()
```



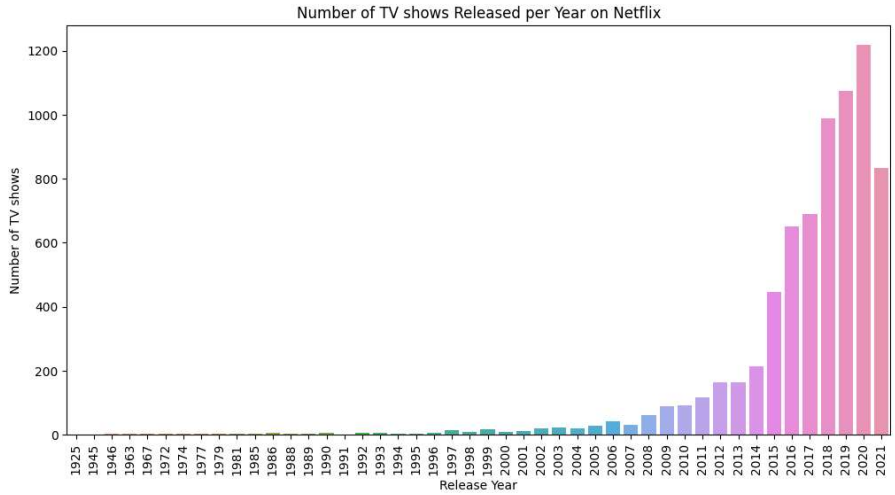
Netflix has 70% of its content as movies.

Movies are clearly more popular on Netflix than TV shows.

```
# Filter for movies only

movies_data = df_final[df_final['type'] == 'Movie']

# Group the data by release year and count the number of movies released each year
movie_count_by_year = movies_data['release_year'].value_counts().sort_index().reset_index()
movie_count_by_year.columns = ['Release Year', 'Count']


# Create a bar chart using Seaborn
plt.figure(figsize=(12, 6))
sns.barplot(x='Release Year', y='Count', data=movie_count_by_year)
plt.title('Number of Movies Released per Year on Netflix')
plt.xlabel('Release Year')
plt.ylabel('Number of Movies')
plt.xticks(rotation=90)
plt.show()
```

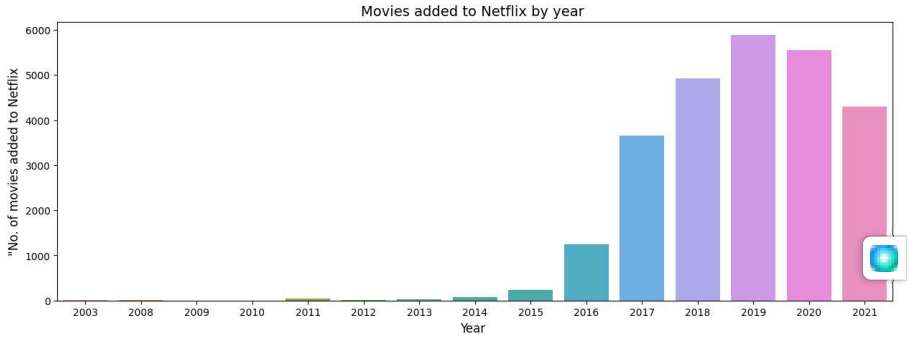



```
# Filter for TV shows only

shows_data = df_final[df_final['type'] == 'TV Show']

# Group the data by release year and count the number of TV shows released each year
shows_count_by_year = shows_data['release_year'].value_counts().sort_index().reset_index()
shows_count_by_year.columns = ['Release Year', 'Count']


# Create a bar chart using Seaborn
plt.figure(figsize=(12, 6))
sns.barplot(x='Release Year', y='Count', data=shows_count_by_year)
plt.title('Number of TV shows Released per Year on Netflix')
plt.xlabel('Release Year')
plt.ylabel('Number of TV shows')
plt.xticks(rotation=90)
plt.show()
```




```
# Checking number of new Contents added yearly

fig = plt.figure(figsize=(15,5))
sns.countplot(data=df_final,x = 'year')
plt.title('Movies added to Netflix by year ', fontsize=14)
plt.ylabel('"No. of movies added to Netflix', fontsize=12)
plt.xlabel('Year ', fontsize=12)
```

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



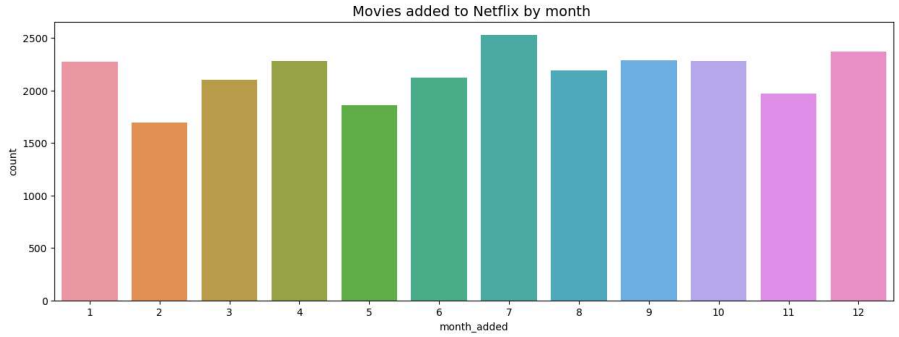
Netflix has started adding content since 2016.

In the last 5 years, so we're seeing a increase in content being added.

Movies and TV shows added in the year 2019 was highest until date.

```
# Checking number of new Contents on months

fig = plt.figure(figsize=(15,5))
sns.countplot(data=df_final,x = 'month_added')
plt.title('Movies added to Netflix by month ', fontsize=14)
```

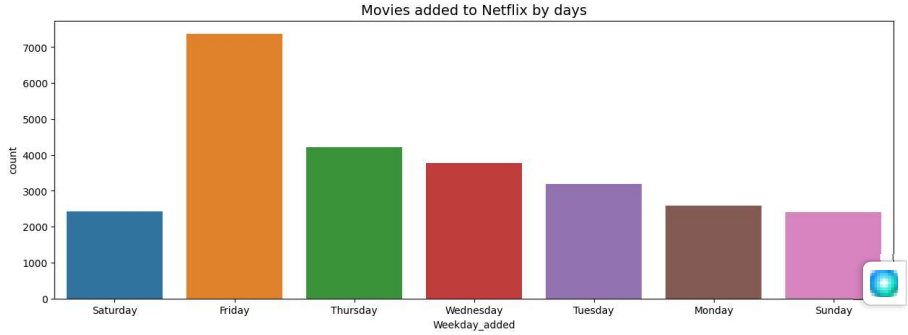
```
Text(0.5, 1.0, 'Movies added to Netflix by month ')
```



Consistent content additions in every month

```
# Checking number of new Contents on weekends

fig = plt.figure(figsize=(15,5))
sns.countplot(data=df_final,x = 'Weekday_added')
plt.title('Movies added to Netflix by days', fontsize=14)
```

```
Text(0.5, 1.0, 'Movies added to Netflix by days')
```



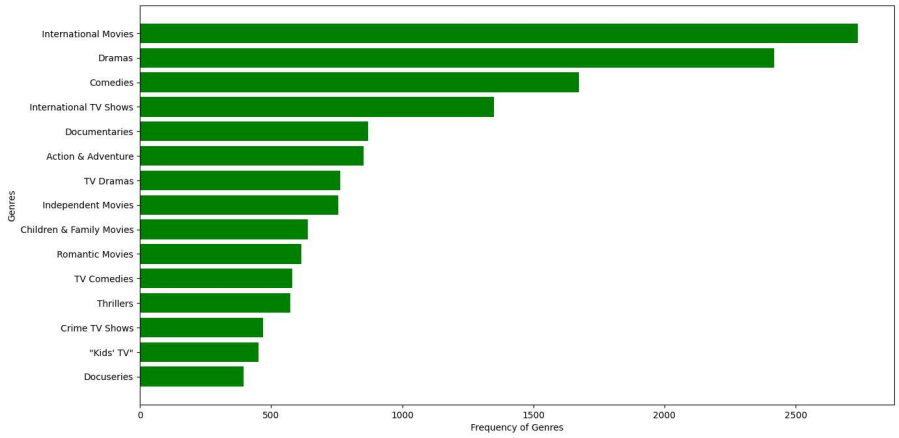
Netflix adds most of its content on Thursdays and Fridays.

On Friday, new content added is highest.

```
# No. of distinct titles on the basis of listed_in

df_final.groupby(['Genre']).agg({"title":"nunique"})
```

Genre	title
"Kids' TV"	451
Action & Adventure	854
Anime Features	71
Anime Series	176
British TV Shows	253
Children & Family Movies	639
Classic & Cult TV	28
Classic Movies	116
Comedies	1673
Crime TV Shows	470
Cult Movies	71
Documentaries	869
Docuseries	395
Dramas	2418
Faith & Spirituality	65
Horror Movies	353
Independent Movies	756
International Movies	2738
International TV Shows	1351
Korean TV Shows	151
LGBTQ Movies	102
Movies	57
Music & Musicals	372
Reality TV	255
Romantic Movies	615
Romantic TV Shows	370
Sci-Fi & Fantasy	243
Science & Nature TV	92
Spanish-Language TV Shows	174
Sports Movies	219
Stand-Up Comedy	343
Stand-Up Comedy & Talk Shows	56
TV Action & Adventure	168
TV Comedies	581
TV Dramas	[illegible]

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



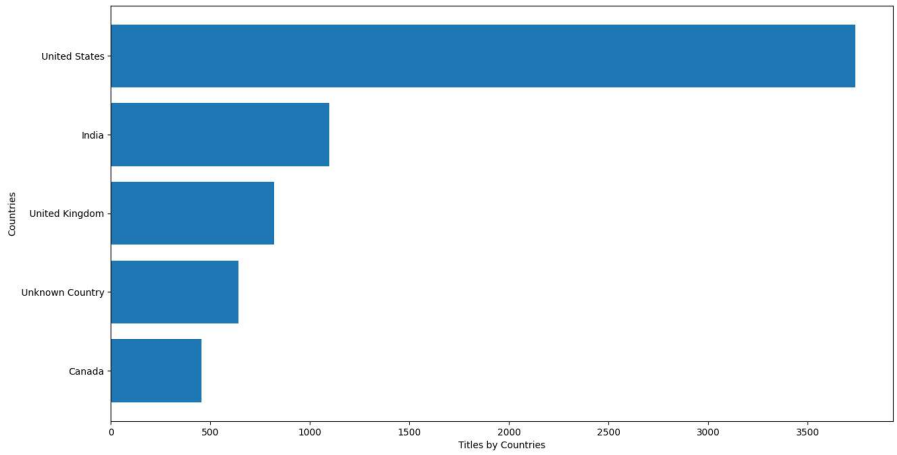
International Movies, Dramas and Comedies are the most popular genres on NETFLIX.

```
# No of distinct titles on the basis of country

df_final.groupby(['country']).agg({"title":"nunique"})
```

country	title
	8
Afghanistan	1
Albania	1
Algeria	4
Angola	1
...	...
Vatican City	1
Venezuela	4
Vietnam	7
West Germany	5
Zimbabwe	3

124 rows × 1 columns

```
df_country=df_final.groupby(['country']).agg({"title":"nunique"}).reset_index().sort_values(by=['title'],ascending=False)[:5]
plt.figure(figsize=(15,8))
plt.barh(df_country[::-1]['country'],df_country[::-1]['title'])
plt.xlabel('Titles by Countries')
plt.ylabel('Countries')
plt.show()
```

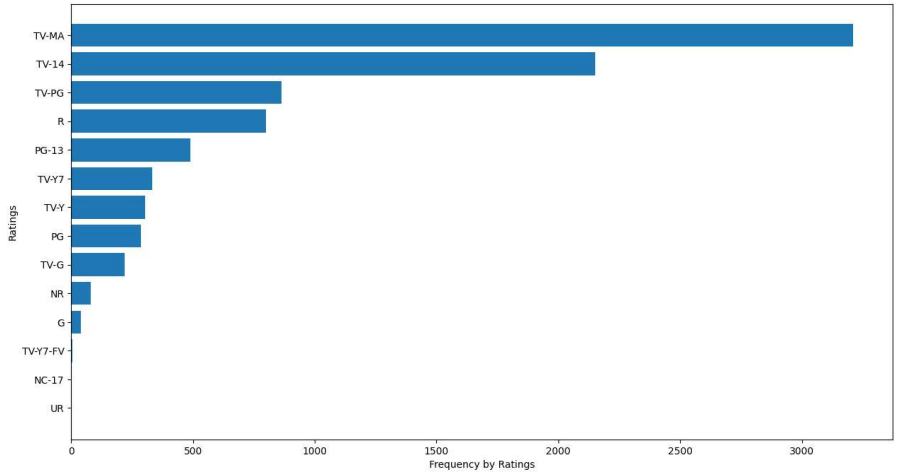


US,India,UK,Canada are leading countries in Content Creation on Netflix

```
# No of distinct titles on the basis of rating

df_final.groupby(['rating']).agg({"title":"nunique"})
```

rating	title
G	41
NC-17	3
NR	80
PG	287
PG-13	490
R	799
TV-14	2151
TV-G	220
TV-MA	3211
TV-PG	863
TV-Y	305
TV-Y7	334
TV-Y7-FV	6
UR	3

```
df_rating=df_final.groupby(['rating']).agg({"title":"nunique"}).reset_index().sort_values(by=['title'],ascending=False)[:15]
plt.figure(figsize=(15,8))
plt.barh(df_rating[::-1]['rating'],df_rating[::-1]['title'])
plt.xlabel('Frequency by Ratings')
plt.ylabel('Ratings')
plt.show()
```



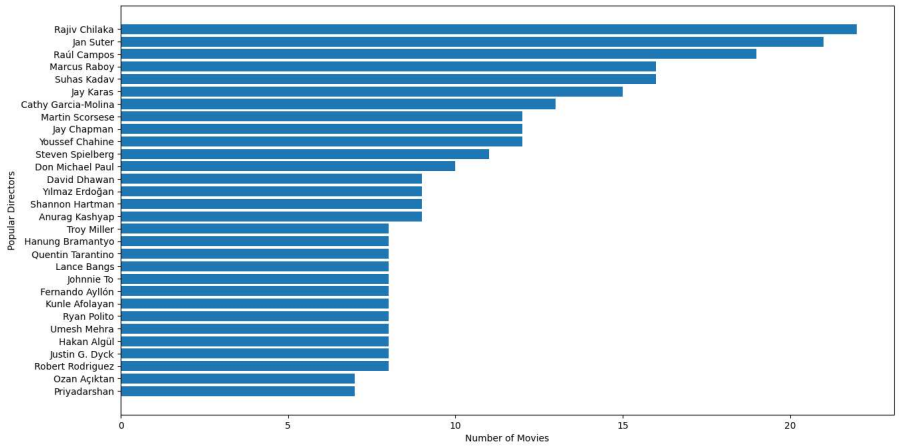
Most of content on Netflix is intended for Mature Audiences, R Rated, content not intended for audience under 14 and those which require Parental Guidance.

```
# No of distinct titles on the basis of Directors

df_final.groupby(['Directors']).agg({"title":"nunique"})
```

Directors	title
" Frank O'Connor"	1
"Alma Har'el"	1
"André D'Elia"	1
"Anthony D'Souza"	2
"Bill D'Elia"	2
...	...
Éric Warin	1
Ísold Uggadóttir	1
Óskar Thór Axelsson	1
Ömer Faruk Sorak	3
Şenol Sönmez	2

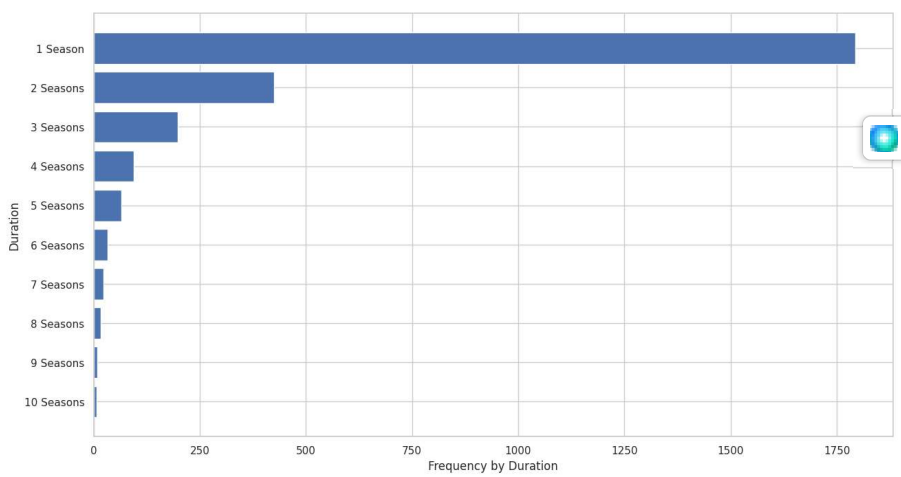
4994 rows × 1 columns

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



Rajiv Chilaka, Jan Suter and Raul Campos are the most popular directors.

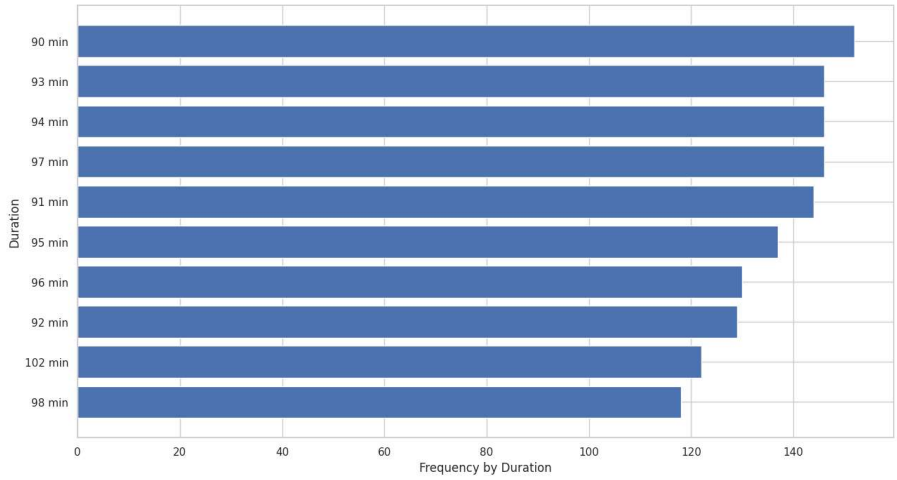
```
df_duration=shows_data.groupby(['duration']).agg({"title":"nunique"}).reset_index().sort_values(by=['title'],ascending=False)[:10]
plt.figure(figsize=(15,8))
plt.barh(df_duration[::-1]['duration'], df_duration[::-1]['title'])
plt.xlabel('Frequency by Duration')
plt.ylabel('Duration')
plt.show()
```



1 Season are common as soon as the season length increases, the number of shows decrease.

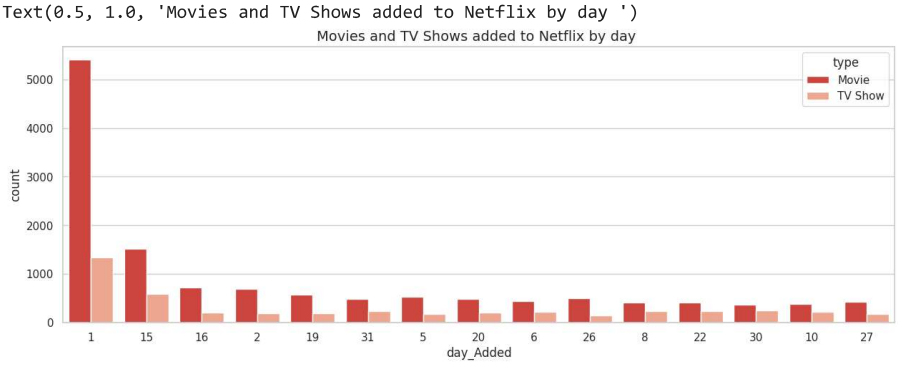
```
df_duration=movies_data.groupby(['duration']).agg({"title":"nunique"}).reset_index().sort_values(by=['title'],ascending=False)[:10]
plt.figure(figsize=(15,8))
```

```
plt.barh(df_duration[::-1]['duration'], df_duration[::-1]['title'])
plt.xlabel('Frequency by Duration')
plt.ylabel('Duration')
plt.show()
```




```
#Bi-variate analysis-1

fig = plt.figure(figsize=(15,5))
sns.countplot(data=df_final,x = 'day_Added',hue = 'type',palette ="Reds_r",
              order = df_final['day_Added'].value_counts().index[0:15])
plt.title('Movies and TV Shows added to Netflix by day ', fontsize=14)
```

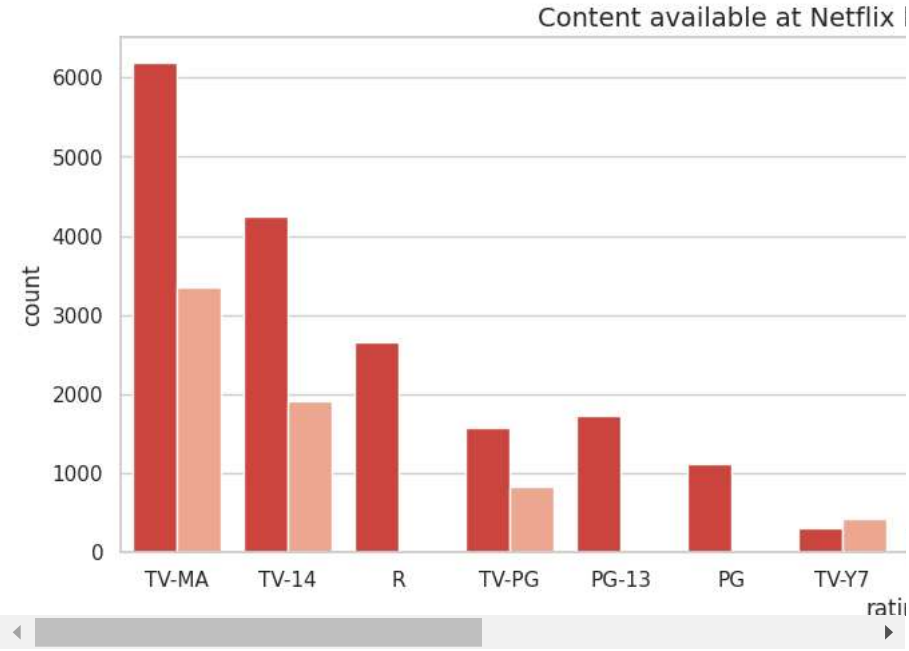
```
Text(0.5, 1.0, 'Movies and TV Shows added to Netflix by day ')
```



It was evident that 1st of every month was when the most content was added. It is recommended that Movies be added at the beginning of every month.

```
# Bi-variate analysis-2

fig = plt.figure(figsize=(15,5))
sns.countplot(data=df_final,x = 'rating',hue = 'type',palette ="Reds_r",
              order = df_final['rating'].value_counts().index[0:15])
plt.title('Content available at Netflix based on the Maurity level ', fontsize=14)
```

```
Text(0.5, 1.0, 'Content available at Netflix based on the Maurity level ')
```



Summary of Final Recommendations:

Invest in Classic Movies and TV Shows: Only 25% of Netflix's content consists of movies and TV shows released before 2013. To attract more subscribers, Netflix should consider investing in classic movies and TV shows that have appeal.

Release Content on Thursdays and Fridays: Netflix adds approx 45% of its content on Thursdays and Fridays, likely because people tend to watch more content on weekends. It is recommended for content creators or Netflix to release new content on these days to maximize viewership.

Schedule Monthly Movie Releases: The 1st of every month sees the most content added to Netflix. It is highly recommended to prioritize adding movies at the beginning of each month. Additionally, Netflix can use this knowledge to plan server upgrades or maintenance ahead of time, minimizing disruptions.

Opportunities for New Content Creators: Netflix started adding content in 2015 and continues to encourage content creators to share new content on the platform. This suggests that new content creators have significant opportunities to showcase their work on Netflix's platform.

Increase Kid-Friendly Content: Approx 48% of Netflix's content is inclined towards adults, and over 60% of TV and show content is not suitable for kids. To attract more subscribers with families, Netflix should consider expanding its library of kid-friendly content.

These recommendations aim to help Netflix optimize its content strategy to cater to a wider audience and improve subscriber numbers.