Netflix Business Case EDA

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

Exploration of Data

```
df = pd.read_csv('netflix.csv')
```

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

	show_id	type	title	director	cast
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalan
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj

```
df.tail(5)
```

	show_id	type	title	director	cast
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey
8803	s8804	TV Show	Zombie Dumb	NaN	NaN
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Ston
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Char

```
df.columns
```

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

```
df.shape
```

```
(8807, 12)
```

```
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
```

```
df.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

```
df.describe(include='object')
```

	show_id	type	title	director	cast	country	date_added	r
count	8807	8807	8807	6173	7982	7976	8797	8
unique	8807	2	8807	4528	7692	748	1767	1
top	s8807	Movie	Zubaan	Rajiv Chilaka	David Attenborough	United States	January 1, 2020	T
freq	1	6131	1	19	19	2818	109	3

```
df.describe(include='all')
```

	show_id	type	title	director	cast	country	date_added	r
count	8807	8807	8807	6173	7982	7976	8797	8
unique	8807	2	8807	4528	7692	748	1767	1
top	s8807	Movie	Zubaan	Rajiv Chilaka	David Attenborough	United States	January 1, 2020	T
freq	1	6131	1	19	19	2818	109	3
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2

```
df.sample()
```

	show_id	type	title	director	cast	countr
7063	s7064	Movie	Incoming	Eric Zaragosa	Scott Adkins, Aaron McCusker, Vahldin Prelic, ...	Serbia

```
df.dtypes
```

	0
show_id	object
type	object
title	object
director	object
cast	object
country	object
date_added	object
release_year	int64

	0
rating	object
duration	object
listed_in	object
description	object

```
df[df.duplicated()]
```

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in

Data Wrangling

Unnesting the columns (directors , casts , countrys , listed_in)

```
unnesting = ['director','cast','country','listed_in']

for column in unnesting:
    df[column] = df[column].str.split(',')
    df = df.explode(column)
```

```
df.shape
```

```
(202065, 12)
```

```
df.dtypes
```

	0
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

```
df.reset_index(drop=True,inplace=True)
```

```
df
```

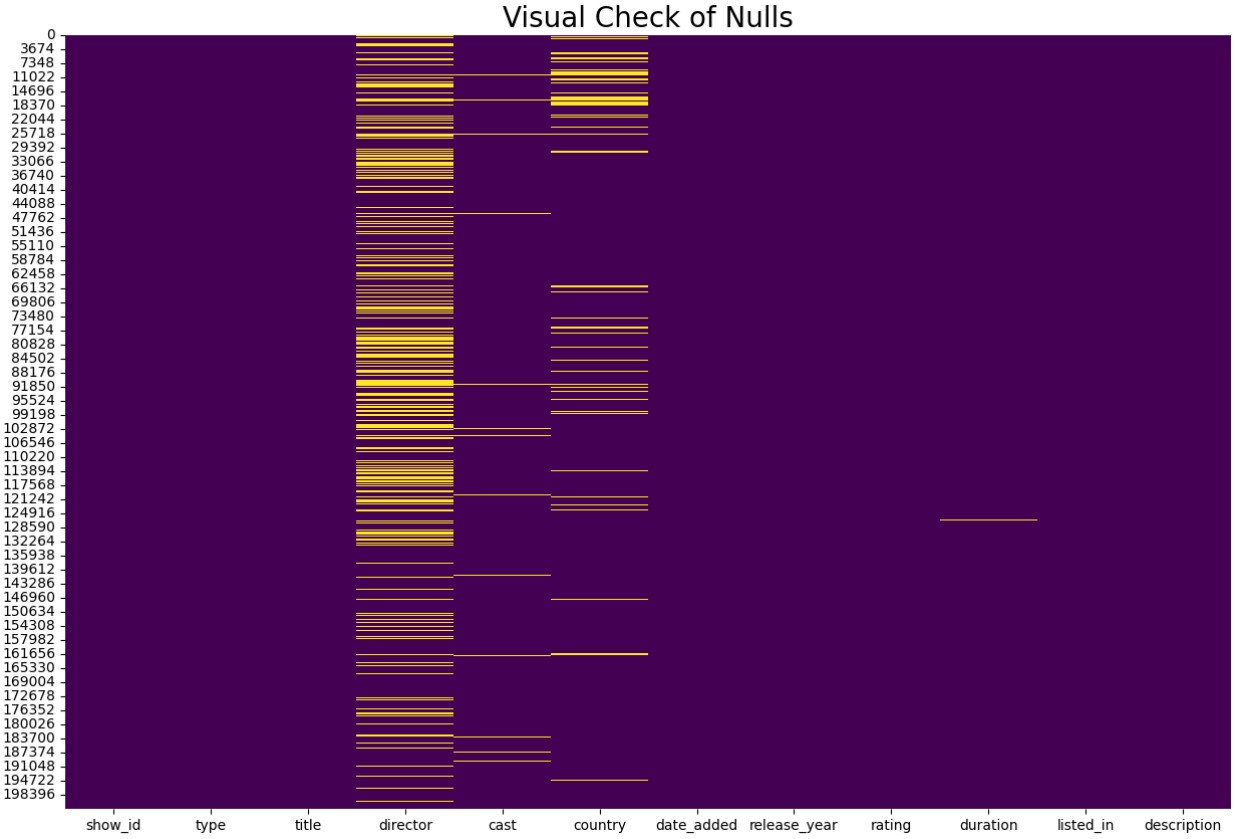
	show_id	type	title	director	cast	country
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United State
1	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa
2	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa
3	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa
4	s2	TV Show	Blood & Water	NaN	Khosi Ngema	South Africa
...	...	...	...	...	...	...
202060	s8807	Movie	Zubaan	Mozez Singh	Anita Shabdish	India
202061	s8807	Movie	Zubaan	Mozez Singh	Anita Shabdish	India
202062	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India
202063	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India
202064	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India

```
df.nunique()
```

	0
show_id	8807
type	2
title	8807
director	5120
cast	39296
country	197
date_added	1767
release_year	74
rating	17
duration	220
listed_in	73
description	8775

Treating Nulls

```
plt.figure(figsize=(15,10))
sns.heatmap(df.isnull(),cbar=False,cmap='viridis')
plt.title('Visual Check of Nulls',fontsize=20)
plt.show()
```




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

	0
director	50643
country	11897
cast	2149
date_added	158
rating	67
duration	3
show_id	0
type	0
title	0
release_year	0
listed_in	0
description	0

```
for i in df.columns:
    null_pct = (df[i].isna().sum() / df.shape[0]) *100
    if null_pct > 0 :
        print(f'Null_pct of {i} is {round(null_pct,3)} %')
```

```
Null_pct of director is 25.063 %
Null_pct of cast is 1.064 %
Null_pct of country is 5.888 %
Null_pct of date_added is 0.078 %
Null_pct of rating is 0.033 %
Null_pct of duration is 0.001 %
```

```
df[df.date_added.isna()]
```

	show_id	type	title	director	cast
136940	s6067	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Daniel Radcliffe
136941	s6067	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Daniel Radcliffe
136942	s6067	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Daniel Radcliffe
136943	s6067	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Jon Hamm
136944	s6067	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Jon Hamm
...	...	...	...	...	...
186965	s8183	TV Show	The Adventures of Figaro Pho	NaN	Charlotte Hamlyn
186966	s8183	TV Show	The Adventures of Figaro Pho	NaN	Stavroula Mountzour
186967	s8183	TV Show	The Adventures of Figaro Pho	NaN	Stavroula Mountzour
186968	s8183	TV Show	The Adventures of Figaro Pho	NaN	Aletheia Burney
186969	s8183	TV Show	The Adventures of Figaro Pho	NaN	Aletheia Burney

```
df['date_added'] = pd.to_datetime(df['date_added'], format='%B %d, %Y',
errors='coerce')
```

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

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

	0
director	50643
country	11897
cast	2149
rating	67
duration	3
title	0
show_id	0
type	0
release_year	0
date_added	0
listed_in	0
description	0

```
df.dtypes
```

	0
show_id	object
type	object
title	object
director	object
cast	object
country	object
date_added	datetime64[ns]
release_year	int64
rating	object
duration	object
listed_in	object
description	object

```
df['year_added'] = df['date_added'].dt.year
```

```
df.sample()
```

	show_id	type	title	director	cast	country	date_added	release_year
78713	s3271	TV Show	Inheritors	NaN	Yoon Jin-seo	South Korea	2019-11-15	2013

```
df[df.rating.isna() | df.duration.isna()]
```

	show_id	type	title	director	cast
126582	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K
131648	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K
131782	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K
135172	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava ...	NaN	Oprah W
135173	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava ...	NaN	Ava DuVe
...	...	...	...	...	...
172016	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Francesco
172017	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Albrecht
172018	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Giulia Di
172019	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Alessandr
172020	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Andreas S

```
df["country"].fillna("Unknown",inplace=True)
df["cast"].fillna("Unknown actors",inplace=True)
df["director"].fillna("Unknown director",inplace=True)
df["rating"].fillna("Unknown",inplace=True)
```

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

	0
show_id	0
type	0
title	0
director	0
cast	0
country	0
date_added	0
release_year	0
rating	0
duration	3
listed_in	0
description	0
year_added	0

```
df[df.duration.isna()]
```

	show_id	type	title	director	cast	country	c
126582	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	2
131648	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	2
131782	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	2

```
mask = df['director'] == 'Louis C.K.'
df.loc[mask, 'duration'] = df.loc[mask, 'duration'].fillna(df.loc[mask,
'rating'])
```

```
df.loc[df['director'] == 'Louis C.K.', 'rating'] = 'Unknown'
```

```
df[df.director=='Louis C.K.']
```

	show_id	type	title	director	cast	country	c
126582	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	2

	show_id	type	title	director	cast	country	
131648	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	2
131782	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	2

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

	0
show_id	0
type	0
title	0
director	0
cast	0
country	0
date_added	0
release_year	0
rating	0
duration	0
listed_in	0
description	0
year_added	0

Data Segregation

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

type	count
Movie	145917
TV Show	56148

```
movies_data = df[df['type'] == 'Movie']
tvshows_data = df[df['type'] == 'TV Show']
```

```
movies_data.shape , tvshows_data.shape
```

```
((145917, 13), (56148, 13))
```

```
movies_data['runtime_in_mins'] = movies_data['duration'].str.split(' ').str[0]
tvshows_data['no_of_seasons'] = tvshows_data['duration'].str.split(' ').str[0]
```

```
movies_data['runtime_in_mins'] = movies_data['runtime_in_mins'].astype(int)
tvshows_data['no_of_seasons'] = tvshows_data['no_of_seasons'].astype(int)
```

```
movies_data = movies_data.drop(columns=['description',
'duration']).reset_index(drop=True)
tvshows_data = tvshows_data.drop(columns=['description',
'duration']).reset_index(drop=True)
```

```
movies_data.sample()
```

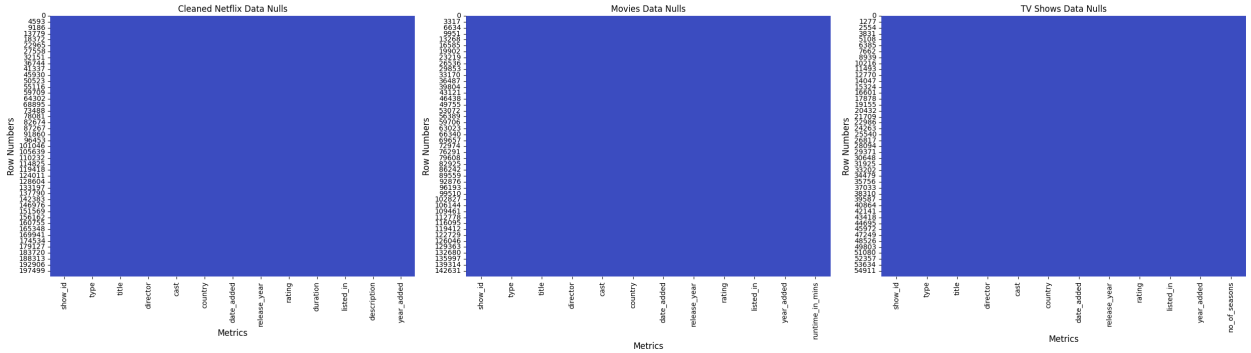
	show_id	type	title	director	cast	country	date_ad
42259	s2597	Movie	What a Girl Wants	Dennie Gordon	Amanda Bynes	United Kingdom	2020-05-0

```
tvshows_data.sample()
```

	show_id	type	title	director	cast	country	date_added
33798	s4007	TV Show	If I Hadn't Met You	Unknown director	Sergi López	Spain	2019-03-15

```
fig = plt.figure(figsize=(25, 8))
fig.suptitle('Visual checks of Nulls', fontsize=20, fontweight="bold",
fontfamily='serif')

plt.subplot(1, 3, 1)
sns.heatmap(df.isnull(), cmap="coolwarm", cbar=False)
plt.title('Cleaned Netflix Data Nulls', fontsize=12)
plt.xlabel('Metrics', fontsize=12)
plt.ylabel('Row Numbers', fontsize=12)

plt.subplot(1, 3, 2)
sns.heatmap(movies_data.isnull(), cmap="coolwarm", cbar=False)
plt.title('Movies Data Nulls', fontsize=12)
plt.xlabel('Metrics', fontsize=12)
plt.ylabel('Row Numbers', fontsize=12)

plt.subplot(1, 3, 3)
sns.heatmap(tvshows_data.isnull(), cmap="coolwarm", cbar=False)
plt.title('TV Shows Data Nulls', fontsize=12)
plt.xlabel('Metrics', fontsize=12)
plt.ylabel('Row Numbers', fontsize=12)

plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```

Visual checks of Nulls




```
# Save DataFrames to CSV files with proper formatting

df.to_csv('netflix_cleaned_data.csv', index=False)
movies_data.to_csv('cleaned_movies_data.csv', index=False)
tvshows_data.to_csv('cleaned_tvshows_data.csv', index=False)
```

Exploratory Data Analysis (EDA):

```
nx = pd.read_csv('netflix_cleaned_data.csv')
md = pd.read_csv('cleaned_movies_data.csv')
tvd = pd.read_csv('cleaned_tvshows_data.csv')
```

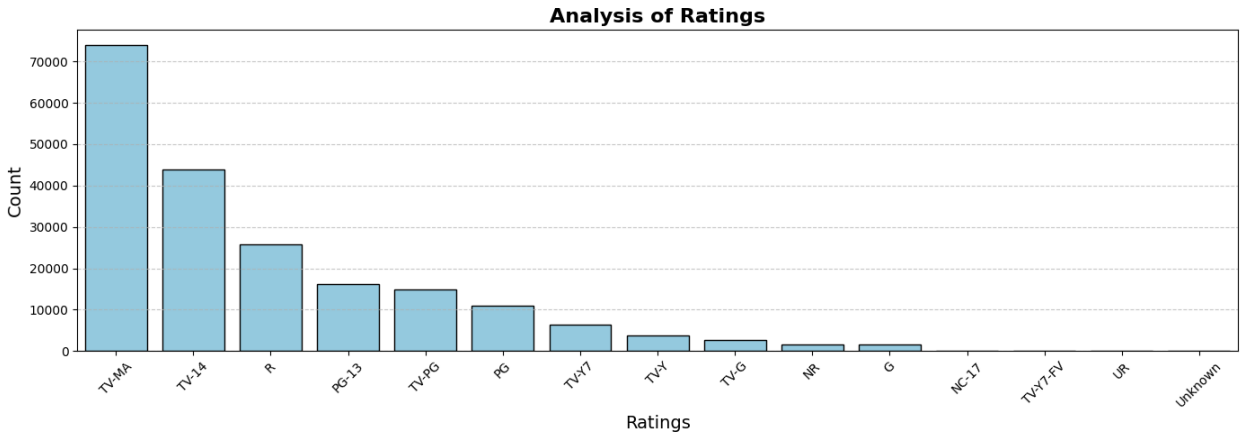
Q. How is content distributed based on ratings on Netflix?

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

rating	count
TV-MA	73915
TV-14	43957
R	25860
PG-13	16246
TV-PG	14926
PG	10919
TV-Y7	6304
TV-Y	3665
TV-G	2779
NR	1573
G	1530

rating	count
NC-17	149
TV-Y7-FV	86
UR	86
Unknown	70

```
plt.figure(figsize=(14, 5))

sns.barplot(x=rvc.index, y=rvc.values, color='skyblue', edgecolor='black' )

plt.title('Analysis of Ratings', fontsize=16, fontweight='bold')
plt.xlabel('Ratings', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.xticks(rotation=45)

plt.grid(axis='y', linestyle='--', alpha=0.7)

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



Insights :

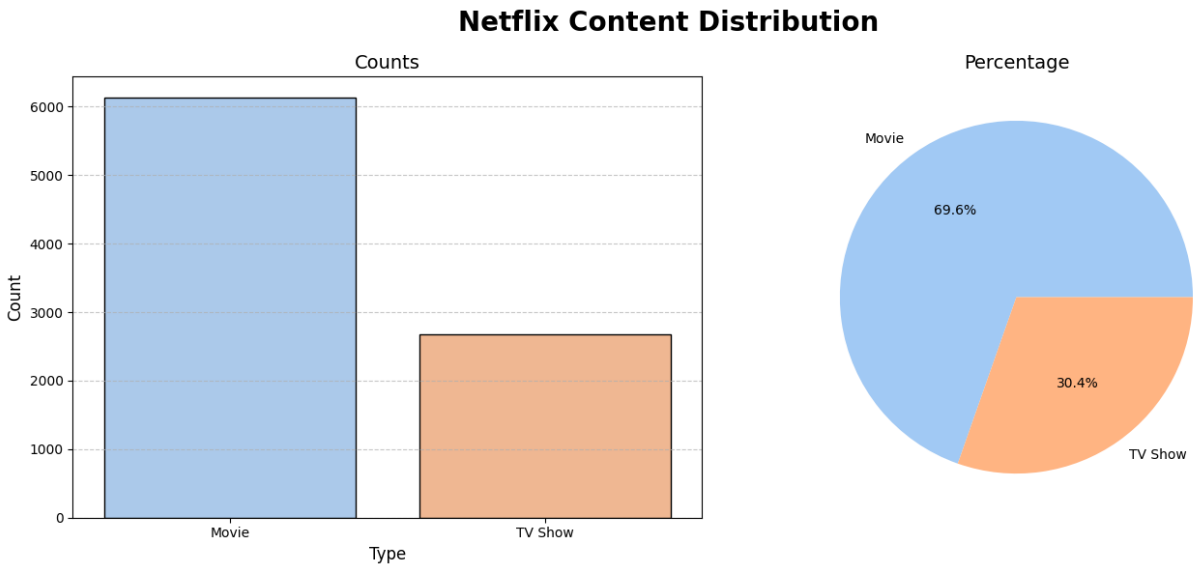
- **While Netflix does offer family-friendly movies and shows, they prioritize content for mature and teenage audiences as majority of content falls under `TV-MA` and `TV-14`.**

Q. How are contents distributed in Netflix Platform ?

```
pg = df.groupby('type')['show_id'].nunique()
pg
```

type	show_id
Movie	6131
TV Show	2676

```
plt.figure(figsize=(14, 6))
font = {'weight':'bold',
        'family':'serif'}
plt.suptitle("Netflix Content Distribution", fontweight='bold', fontsize=20)

plt.subplot(1, 2, 1)
ax = sns.barplot(x=pg.index, y=pg.values, palette="pastel", edgecolor='black')

plt.title('Counts', fontsize=14)
plt.xlabel('Type', fontsize=12)
plt.ylabel('Count', fontsize=12)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.subplot(1, 2, 2)
plt.pie(pg, labels=pg.index, autopct='%1.1f%%',
colors=sns.color_palette("pastel"))
plt.title('Percentage', fontsize=14)

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



Insights :

- **we can clearly interpret that nearly *70%* contents are `Movies` whereas *30%* are `Tvshows` contents in Netflix content library.**

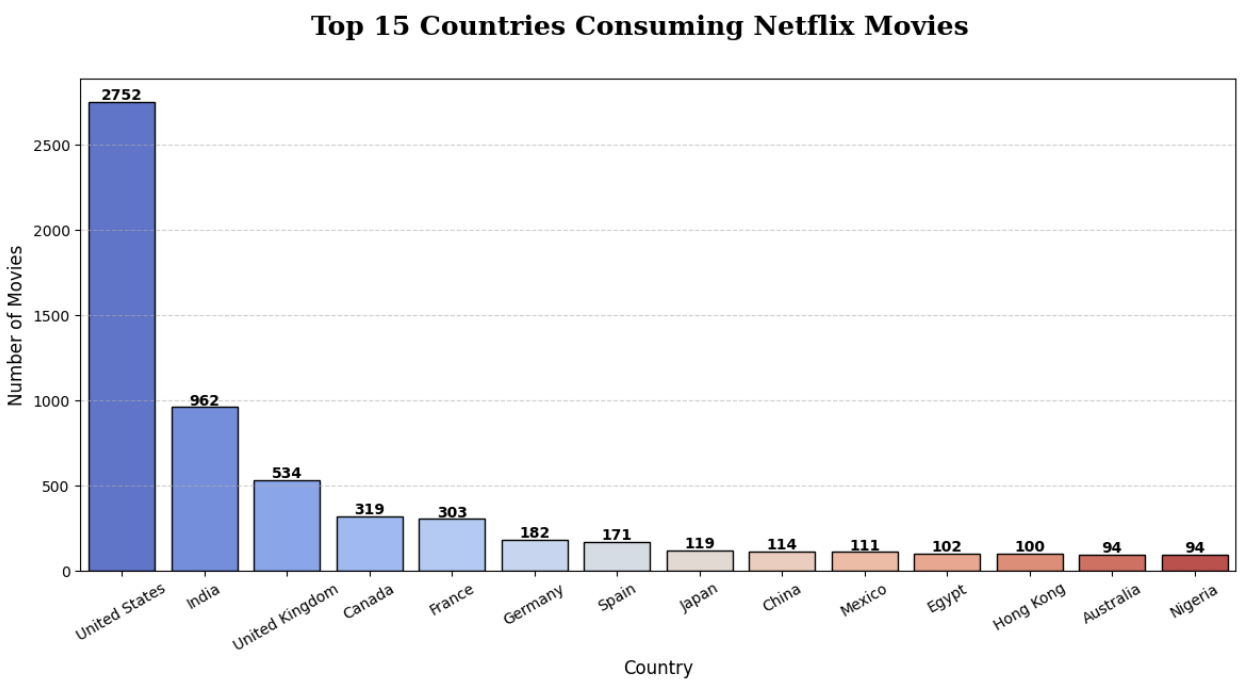
Q. What are the top 15 countries producing the most Movies and TV Shows on Netflix?

```
# Clean the 'country' column before grouping
md['country'] = md['country'].str.strip().str.title()  # Strip spaces and
title-case

# countries consumption of movies
cm = md.groupby('country')['show_id'].nunique().sort_values(ascending=False)[:15]
cmdf = cm[cm.index != 'Unknown']
cmdf
```

country	show_id
United States	2752
India	962
United Kingdom	534
Canada	319
France	303
Germany	182
Spain	171
Japan	119
China	114
Mexico	111
Egypt	102
Hong Kong	100
Australia	94
Nigeria	94

```
plt.figure(figsize=(14, 6))
plt.suptitle('Top 15 Countries Consuming Netflix Movies',
             fontsize=18, fontweight="bold", fontfamily='serif')

# Access the values of the 'show_id' column directly for the y-axis
sns.barplot(x=cmdf.index, y=cmdf.values, palette="coolwarm", edgecolor='black')

plt.xlabel('Country', fontsize=12)
plt.ylabel('Number of Movies', fontsize=12)
plt.xticks(rotation=30)
```

```
plt.grid(axis='y', linestyle='--', alpha=0.6)

for i, value in enumerate(cmdf.values): # iterate through the show_id column
    plt.text(i, value + 10, str(value), ha='center', fontsize=10,
fontweight="bold")

plt.show()
```

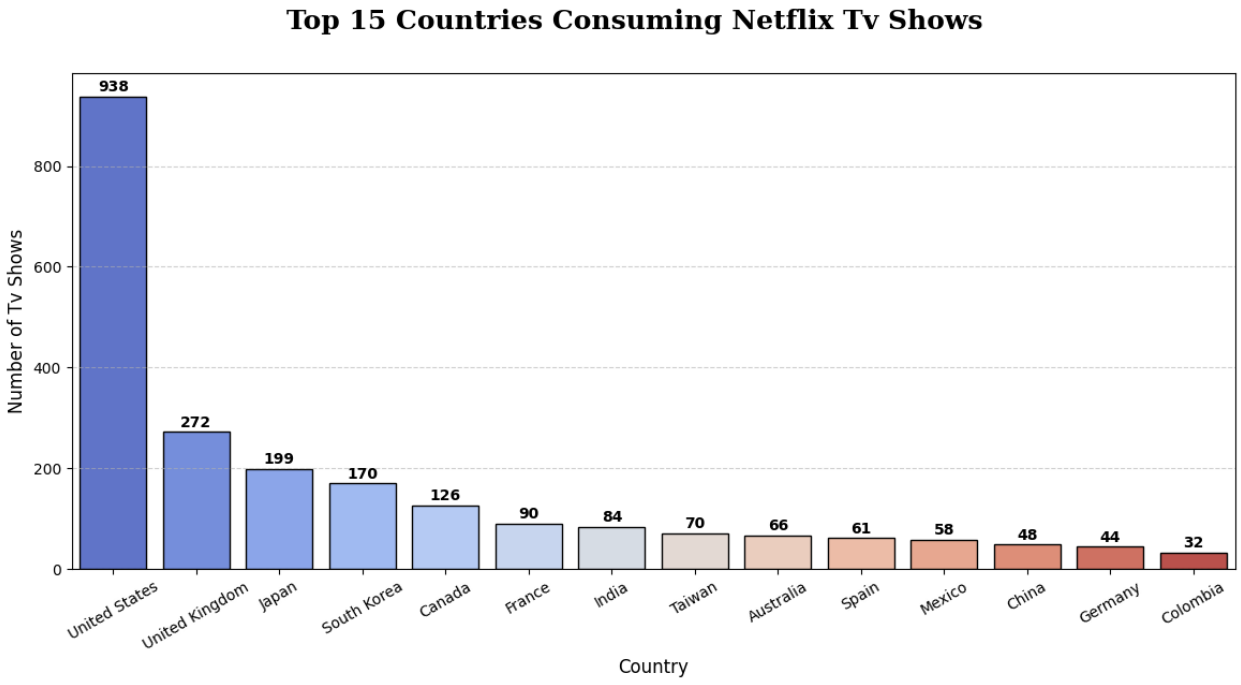



```
# Clean the 'country' column before grouping
tvd['country'] = tvd['country'].str.strip().str.title()  # Strip spaces and
title-case

# countries consumption of tv shows
ctv
=tvd.groupby('country')['show_id'].nunique().sort_values(ascending=False)[:15]
ctvdf = ctv[ctv.index != 'Unknown']
ctvdf
```

country	show_id
United States	938
United Kingdom	272
Japan	199
South Korea	170
Canada	126

country	show_id
France	90
India	84
Taiwan	70
Australia	66
Spain	61
Mexico	58
China	48
Germany	44
Colombia	32

```
plt.figure(figsize=(14, 6))
plt.suptitle('Top 15 Countries Consuming Netflix Tv Shows',
             fontsize=18, fontweight="bold", fontfamily='serif')

# Access the values of the 'show_id' column directly for the y-axis
sns.barplot(x=ctvdf.index, y=ctvdf.values, palette="coolwarm", edgecolor='black')

plt.xlabel('Country', fontsize=12)
plt.ylabel('Number of Tv Shows', fontsize=12)
plt.xticks(rotation=30)
plt.grid(axis='y', linestyle='--', alpha=0.6)

for i, value in enumerate(ctvdf.values): # iterate through the show_id column
    plt.text(i, value + 10, str(value), ha='center', fontsize=10,
fontweight="bold")

plt.show()
```



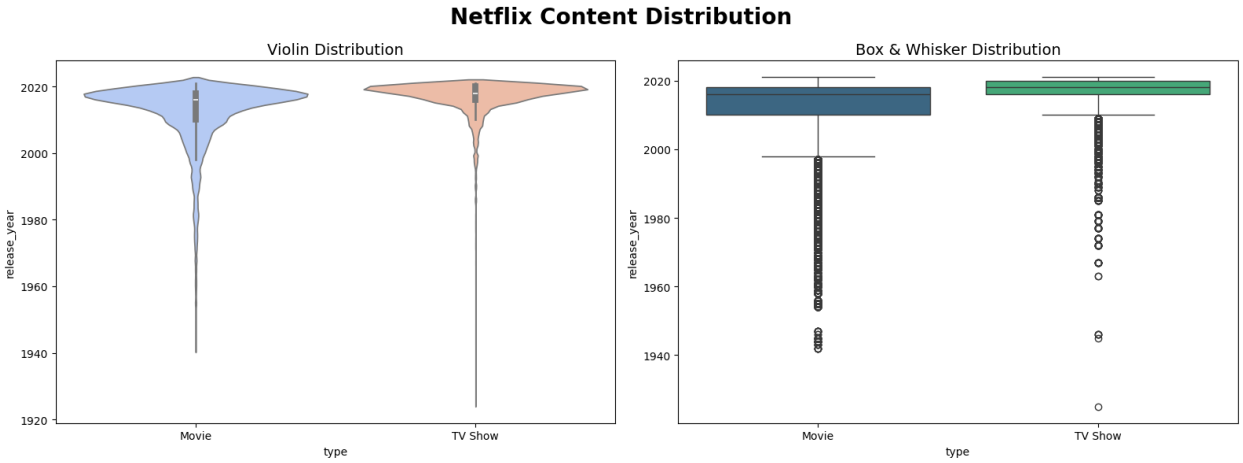
Insights :

- The top 5 countries producing the highest count of Movies are United States, India, United Kingdom, Canada and France.
-

The top 5 countries producing the highest count of TV shows are United States, United Kingdom, Japan, South Korea and Canada.

Q. Outliers check:

```
plt.figure(figsize=(16, 6))
plt.suptitle("Netflix Content Distribution", fontweight='bold', fontsize=20)

# Violin Plot
plt.subplot(1, 2, 1)
sns.violinplot(data=nx, x='type', y='release_year', palette="coolwarm")
plt.title("Violin Distribution", fontsize=14)

# Box Plot
plt.subplot(1, 2, 2)
sns.boxplot(data=nx, x='type', y='release_year', palette="viridis")
plt.title("Box & Whisker Distribution", fontsize=14)

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



Insights :

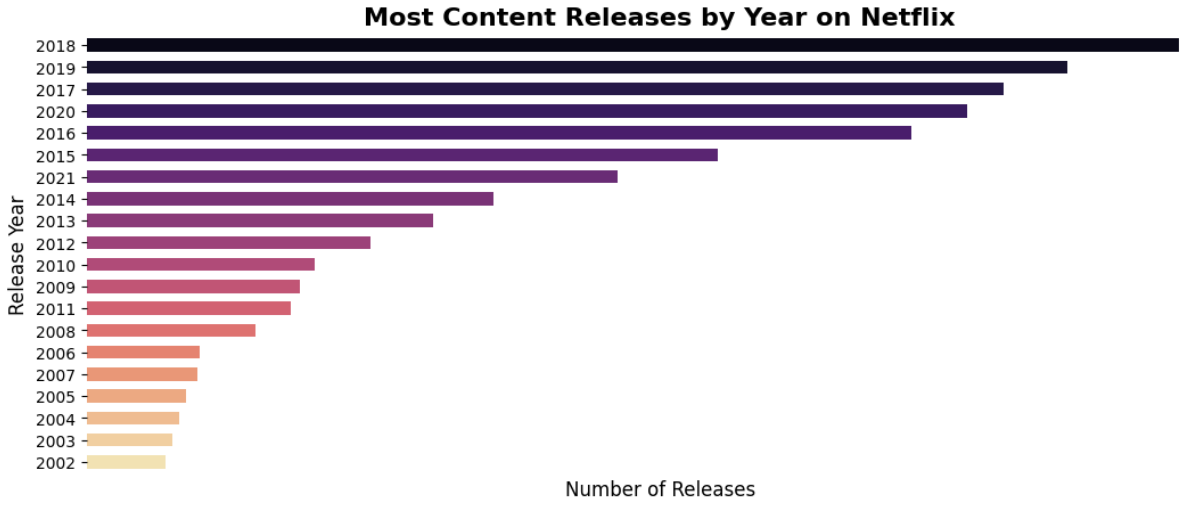
-

Here we can identify that the contents released before 1960 are the outliers.

Q. In which year maximum contents got released ?

```
ryvc = nx.release_year.value_counts()[:20]
```

```
plt.figure(figsize=(13, 5))

sns.countplot(data=nx, y='release_year', order=ryvc.index, palette="magma",
width=0.6)
sns.despine(left=True, bottom=True)

plt.xticks([])
plt.xlabel('Number of Releases', fontsize=12)
plt.ylabel('Release Year', fontsize=12)
plt.title('Most Content Releases by Year on Netflix', fontsize=16,
fontweight='bold')

plt.show()
```



Insights :

- **Maximum contents got released in 2018 followed by 2019.**

Multivariate Analisys

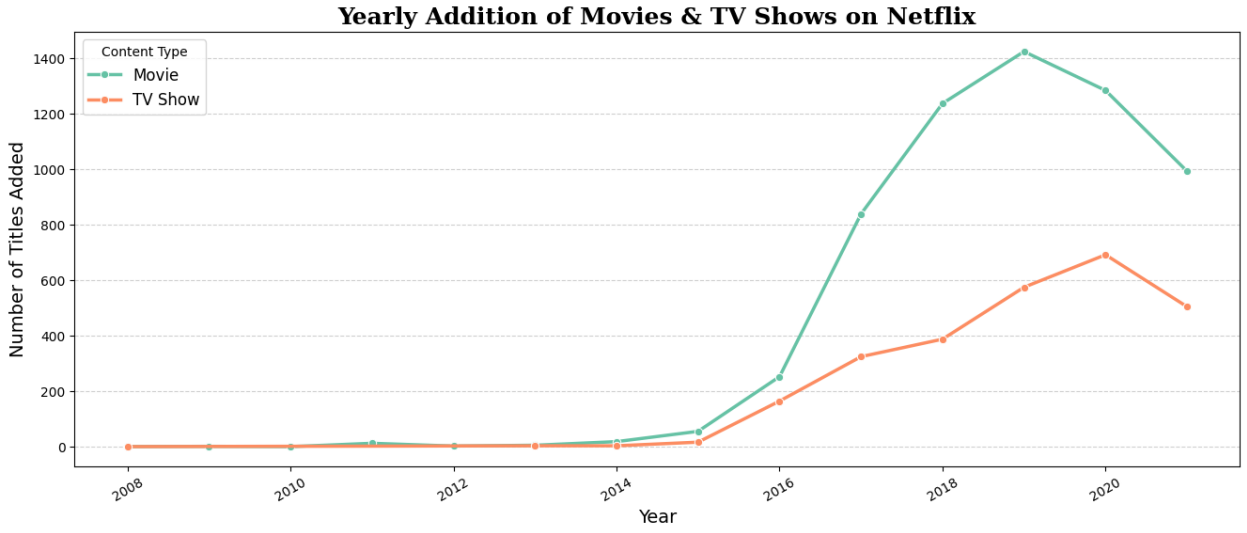
Q. How much contents are added every year in netflix ?

```
yc = nx.groupby(['year_added','type'])[['show_id']].nunique().reset_index()
yc.sort_values(by='show_id',ascending=False)
```

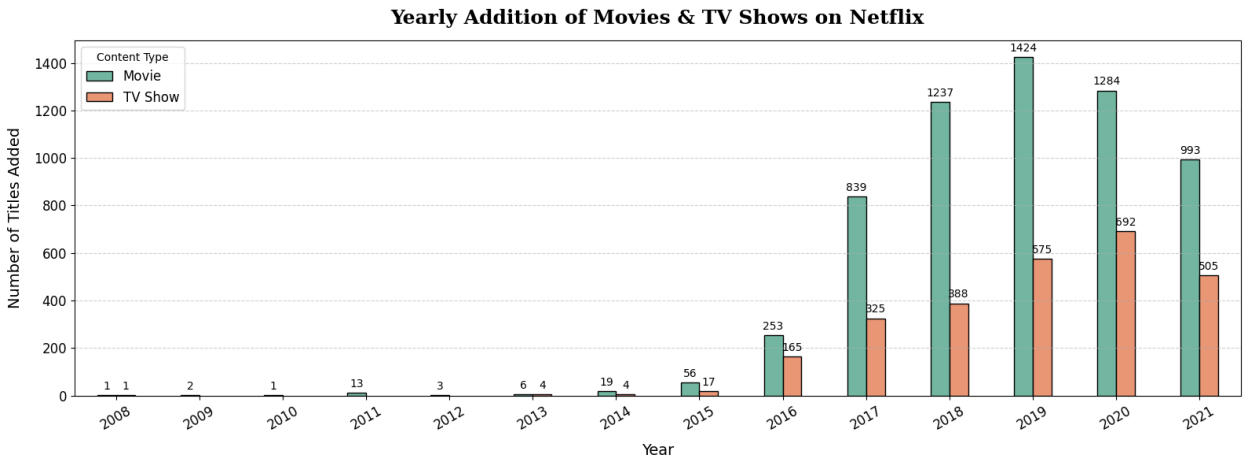
	year_added	type	show_id
18	2019	Movie	1424
20	2020	Movie	1284
16	2018	Movie	1237
22	2021	Movie	993
14	2017	Movie	839
21	2020	TV Show	692
19	2019	TV Show	575
23	2021	TV Show	505
17	2018	TV Show	388
15	2017	TV Show	325
12	2016	Movie	253
13	2016	TV Show	165
10	2015	Movie	56
8	2014	Movie	19
11	2015	TV Show	17

	year_added	type	show_id
4	2011	Movie	13
6	2013	Movie	6
7	2013	TV Show	4
9	2014	TV Show	4
5	2012	Movie	3
2	2009	Movie	2
1	2008	TV Show	1
3	2010	Movie	1
0	2008	Movie	1

```
plt.figure(figsize=(16, 6))
plt.style.use('seaborn-v0_8-bright')

sns.lineplot(data=yc, x='year_added', y='show_id', hue='type',
             marker='o', linewidth=2.5, palette='Set2')

plt.title('Yearly Addition of Movies & TV Shows on Netflix', fontsize=18,
fontweight="bold", fontfamily='serif')
plt.xlabel('Year', fontsize=14)
plt.ylabel('Number of Titles Added', fontsize=14)
plt.xticks(rotation=30)
plt.grid(axis='y', linestyle='--', alpha=0.6)

plt.legend(title="Content Type", loc='upper left', fontsize=12)

plt.show()
```



```
plt.figure(figsize=(16, 6))
plt.style.use('seaborn-v0_8-bright')

# Improved bar plot
c = sns.barplot(data=yc, x='year_added', y='show_id', hue='type',
                palette='Set2', width=0.45, edgecolor='black')

# Title & Labels
plt.title('Yearly Addition of Movies & TV Shows on Netflix',
          fontsize=18, fontweight="bold", fontfamily='serif', pad=15)
plt.xlabel('Year', fontsize=14, labelpad=10)
plt.ylabel('Number of Titles Added', fontsize=14, labelpad=10)
plt.xticks(rotation=30, fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.6)

# Bar labels for better readability
for container in c.containers:
    c.bar_label(container, label_type='edge', fontsize=10, padding=3)

# Enhanced legend placement
plt.legend(title="Content Type", loc='upper left', fontsize=12, frameon=True)

# Adjust layout for better spacing
plt.tight_layout()
plt.show()
```



Insights :

- Netflix's content library grew significantly after 2016, peaking in 2019 with over 1,400 movies and 575 TV shows added.
-

Movies consistently outnumber TV Shows, but TV content saw rapid growth from 2017 onward.

Q. How much contents gets released every year?

```
rc = nx.groupby(['release_year','type'])[['show_id']].nunique().reset_index()
rc.sort_values(by='show_id',ascending=False)

print(rc.to_string())
```

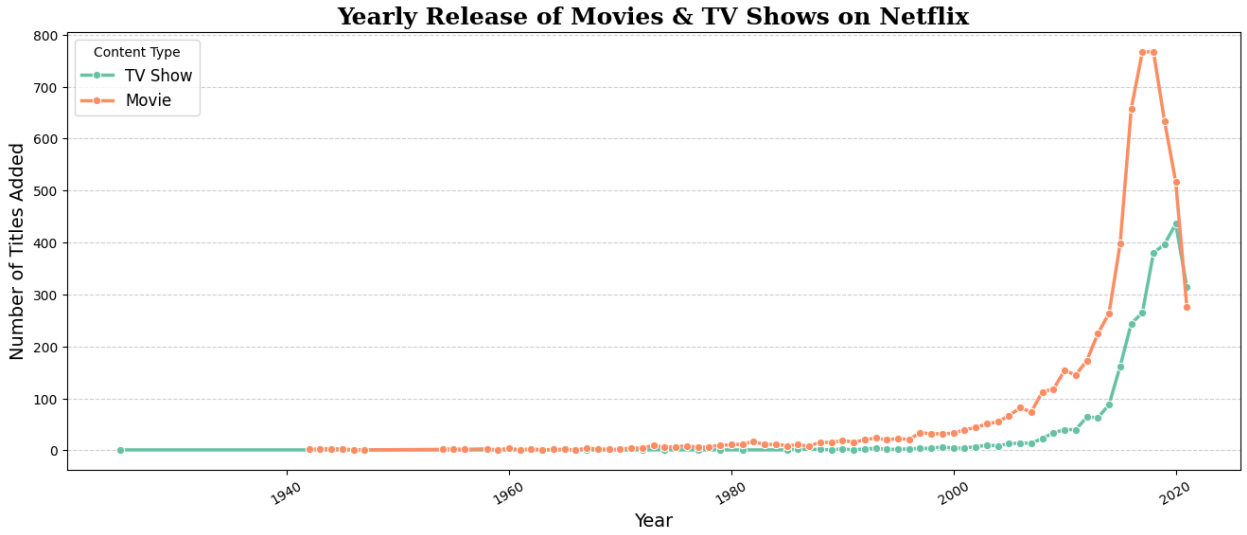
```
     release_year     type  show_id
0            1925  TV Show        1
1            1942    Movie        2
2            1943    Movie        3
3            1944    Movie        3
4            1945    Movie        3
5            1945  TV Show        1
6            1946    Movie        1
7            1946  TV Show        1
8            1947    Movie        1
9            1954    Movie        2
10           1955    Movie        3
11           1956    Movie        2
12           1958    Movie        3
13           1959    Movie        1
14           1960    Movie        4
15           1961    Movie        1
16           1962    Movie        3
17           1963    Movie        1
18           1963  TV Show        1
19           1964    Movie        2
20           1965    Movie        2
21           1966    Movie        1
22           1967    Movie        4
23           1967  TV Show        1
24           1968    Movie        3
25           1969    Movie        2
26           1970    Movie        2
27           1971    Movie        5
28           1972    Movie        4
29           1972  TV Show        1
30           1973    Movie       10
31           1974    Movie        6
32           1974  TV Show        1
33           1975    Movie        7
34           1976    Movie        9
35           1977    Movie        6
36           1977  TV Show        1
```

37	1978	Movie	7
38	1979	Movie	10
39	1979	TV Show	1
40	1980	Movie	11
41	1981	Movie	12
42	1981	TV Show	1
43	1982	Movie	17
44	1983	Movie	11
45	1984	Movie	12
46	1985	Movie	9
47	1985	TV Show	1
48	1986	Movie	11
49	1986	TV Show	2
50	1987	Movie	8
51	1988	Movie	16
52	1988	TV Show	2
53	1989	Movie	15
54	1989	TV Show	1
55	1990	Movie	19
56	1990	TV Show	3
57	1991	Movie	16
58	1991	TV Show	1
59	1992	Movie	20
60	1992	TV Show	3
61	1993	Movie	24
62	1993	TV Show	4
63	1994	Movie	20
64	1994	TV Show	2
65	1995	Movie	23
66	1995	TV Show	2
67	1996	Movie	21
68	1996	TV Show	3
69	1997	Movie	34
70	1997	TV Show	4
71	1998	Movie	32
72	1998	TV Show	4
73	1999	Movie	32
74	1999	TV Show	7
75	2000	Movie	33
76	2000	TV Show	4
77	2001	Movie	40
78	2001	TV Show	5
79	2002	Movie	44
80	2002	TV Show	7
81	2003	Movie	51
82	2003	TV Show	10
83	2004	Movie	55
84	2004	TV Show	9

```
85           2005    Movie       67
86           2005  TV Show       13
87           2006    Movie       82
88           2006  TV Show       14
89           2007    Movie       74
90           2007  TV Show       14
91           2008    Movie      113
92           2008  TV Show       23
93           2009    Movie      118
94           2009  TV Show       34
95           2010    Movie      154
96           2010  TV Show       40
97           2011    Movie      145
98           2011  TV Show       40
99           2012    Movie      173
100          2012  TV Show       64
101          2013    Movie      225
102          2013  TV Show       63
103          2014    Movie      264
104          2014  TV Show       88
105          2015    Movie      398
106          2015  TV Show      162
107          2016    Movie      658
108          2016  TV Show      244
109          2017    Movie      767
110          2017  TV Show      265
111          2018    Movie      767
112          2018  TV Show      380
113          2019    Movie      633
114          2019  TV Show      397
115          2020    Movie      517
116          2020  TV Show      436
117          2021    Movie      277
118          2021  TV Show      315
```

```
plt.figure(figsize=(16, 6))
plt.style.use('seaborn-v0_8-bright')

sns.lineplot(data=rc, x='release_year', y='show_id', hue='type',
                         marker='o', linewidth=2.5, palette='Set2')

plt.title('Yearly Release of Movies & TV Shows on Netflix', fontsize=18,
fontweight="bold", fontfamily='serif')
plt.xlabel('Year', fontsize=14)
plt.ylabel('Number of Titles Added', fontsize=14)
plt.xticks(rotation=30)
plt.grid(axis='y', linestyle='--', alpha=0.6)
```

```
plt.legend(title="Content Type", loc='upper left', fontsize=12)

plt.show()
```



Insights :

- Netflix has few movies and TV shows from before 2010 compared to the vast collection from more recent years.
-

A large portion of Netflix's available content comes from 2017 and 2018, making them the most predominant years in the library.

Univariate Analisys

Q. What are the top 10 genres in Netflix ?

```
mg = md.groupby(['listed_in'])[['title']].nunique().sort_values(by='title'
,ascending=False)[:10]
mg = mg.reset_index()
mg
```

	listed_in	title
0	International Movies	2624
1	Dramas	1600
2	Comedies	1210
3	Action & Adventure	859

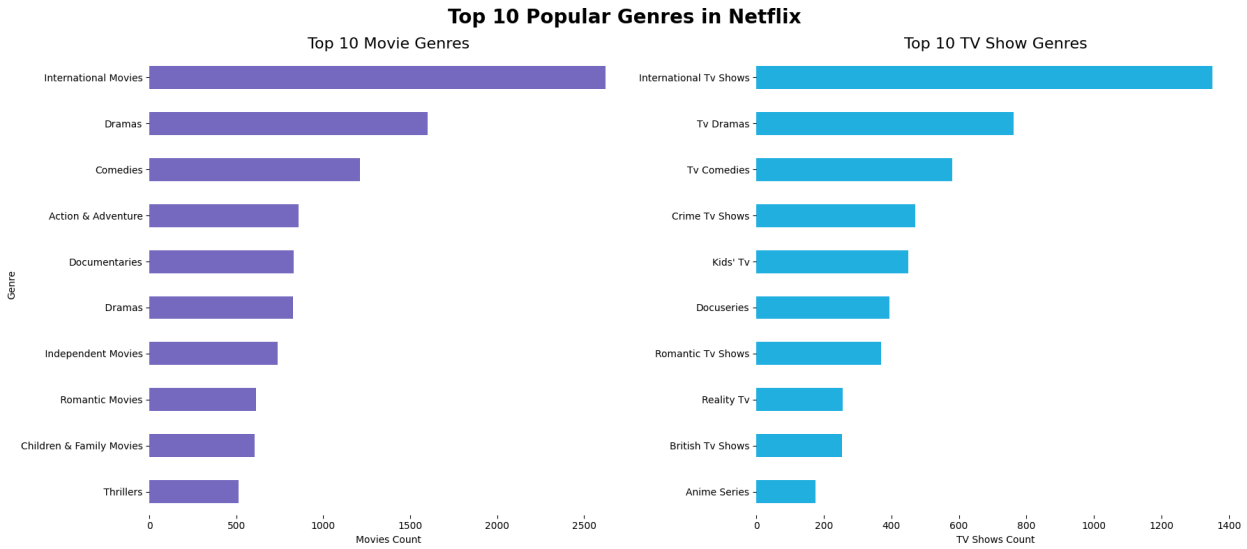
	listed_in	title
4	Documentaries	829
5	Dramas	827
6	Independent Movies	736
7	Romantic Movies	613
8	Children & Family Movies	605
9	Thrillers	512

```
# Clean the 'listed_in' column before grouping
tvd['listed_in'] = tvd['listed_in'].str.strip().str.title()

# Now perform the groupby and other operations
tvg = tvd.groupby(['listed_in'])[['title']].nunique()
tvg = tvg.sort_values(by='title',ascending=False)[:10]
tvg = tvg.reset_index()

tvg
```

	listed_in	title
0	International Tv Shows	1351
1	Tv Dramas	763
2	Tv Comedies	581
3	Crime Tv Shows	470
4	Kids' Tv	451
5	Docuseries	395
6	Romantic Tv Shows	370
7	Reality Tv	255
8	British Tv Shows	253
9	Anime Series	176

```
plt.figure(figsize=(18, 8))
plt.suptitle('Top 10 Popular Genres in Netflix',
             fontsize=20, fontweight="bold")

# Movie Genre (Horizontal Bar)
plt.subplot(1, 2, 1)
sns.barplot(data=mg, y='listed_in', x='title', color='slateblue', width=0.5)
sns.despine(left=True, bottom=True, trim=True)
plt.title('Top 10 Movie Genres', fontsize=16)
plt.xlabel('Movies Count')
plt.ylabel('Genre')

# TV Show Genre (Horizontal Bar)
plt.subplot(1, 2, 2)
```

```
sns.barplot(data=tvg, y='listed_in', x='title', color='deepskyblue', width=0.5)
sns.despine(left=True, bottom=True, trim=True)
plt.title('Top 10 TV Show Genres', fontsize=16)
plt.xlabel('TV Shows Count')
plt.ylabel('')

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



Insights :

- International content is a major part of Netflix's library across both movies and TV shows.
-

Drama, Comedy, and Action remain the most in-demand genres.

Q. what genre's are more preferred by directors ?

```
mdgd = md.groupby(['listed_in'])[['director']].nunique().sort_values(by=
'director',ascending=False)[:10]
mdgd
```

listed_in	director
International Movies	2170
Dramas	1429
Comedies	1057

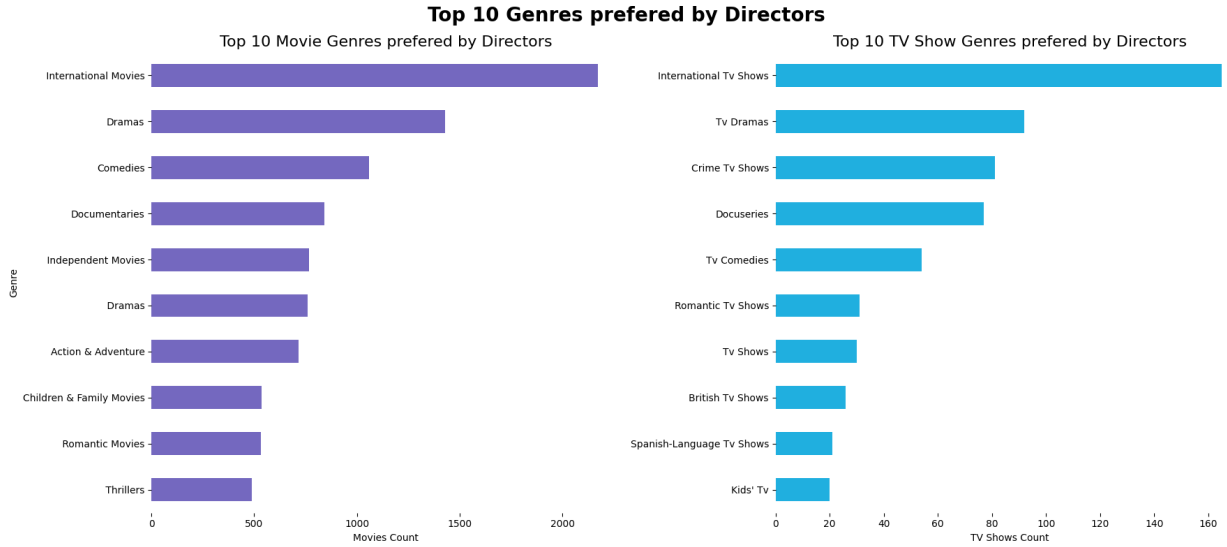
listed_in	director
Documentaries	840
Independent Movies	765
Dramas	759
Action & Adventure	715
Children & Family Movies	536
Romantic Movies	531
Thrillers	487

```
tvdgd = tvd.groupby(['listed_in'])[['director']].nunique().sort_values(by=
'director',ascending=False)[:10]
tvdgd
```

listed_in	director
International Tv Shows	165
Tv Dramas	92
Crime Tv Shows	81
Docuseries	77
Tv Comedies	54
Romantic Tv Shows	31
Tv Shows	30
British Tv Shows	26
Spanish-Language Tv Shows	21
Kids' Tv	20

```
plt.figure(figsize=(18, 8))
plt.suptitle('Top 10 Genres prefered by Directors',
             fontsize=20, fontweight="bold")

# Movie Genre (Horizontal Bar)
plt.subplot(1, 2, 1)
sns.barplot(data=mdgd, y='listed_in', x='director', color='slateblue', width=0.5)
sns.despine(left=True, bottom=True, trim=True)
plt.title('Top 10 Movie Genres prefered by Directors', fontsize=16)
plt.xlabel('Movies Count')
plt.ylabel('Genre')

# TV Show Genre (Horizontal Bar)
plt.subplot(1, 2, 2)
sns.barplot(data=tvdgd, y='listed_in', x='director', color='deepskyblue',
width=0.5)
sns.despine(left=True, bottom=True, trim=True)
```

```
plt.title('Top 10 TV Show Genres prefered by Directors', fontsize=16)
plt.xlabel('TV Shows Count')
plt.ylabel('')

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



Insights :

-

Genres like Dramas, Comedies, Crime,and Documentaries are prefered by directors across Movies and Tv Shows.

Q. How are contents distributed based on Runtime & Seasons

```
mrm = md.groupby(['runtime_in_mins'])[['title']].nunique().sort_values(by=
'title',ascending=False)
mrm
print(mrm.to_string())
```

```
                 title
runtime_in_mins
90                 152
94                 146
93                 146
97                 146
91                 144
95                 137
```

96	130
92	129
102	122
98	120
99	118
88	116
101	116
103	114
106	111
100	108
89	106
104	104
86	103
87	101
105	101
107	98
110	97
108	87
116	80
112	74
85	73
109	69
113	69
111	68
84	68
83	65
118	65
119	63
81	62
115	61
117	61
120	56
114	56
121	54
124	52
82	52
127	48
122	45
78	45
123	44
126	44
80	43
133	42
128	41
130	40
135	39
137	38
132	37

125	36
75	35
79	35
131	34
72	33
63	32
129	32
74	32
61	31
76	31
73	30
77	30
66	29
60	29
69	28
71	28
70	28
140	25
59	25
68	25
65	25
58	25
62	24
54	24
46	24
53	24
64	23
143	23
24	23
136	23
139	22
134	22
67	21
138	21
52	20
148	19
141	19
44	19
145	18
150	17
55	16
22	16
149	15
151	15
162	14
57	14
142	13
154	13

23	13
40	13
146	13
56	12
158	12
147	12
153	11
163	11
25	11
29	11
51	11
47	11
50	10
45	10
28	10
156	10
155	10
161	10
49	9
144	9
42	9
32	9
165	8
166	8
48	8
171	7
168	7
173	6
159	6
157	6
160	6
33	6
185	6
30	6
26	6
35	5
38	5
170	5
36	5
176	5
177	5
152	5
164	4
181	4
172	4
12	3
14	3
27	3

21	3
13	3
17	3
15	3
37	3
34	3
41	3
182	3
209	2
174	2
180	2
187	2
204	2
195	2
192	2
11	2
31	2
39	2
20	2
19	2
190	2
179	2
169	2
43	1
10	1
8	1
5	1
3	1
9	1
16	1
18	1
178	1
167	1
189	1
186	1
194	1
196	1
193	1
191	1
201	1
200	1
205	1
203	1
208	1
212	1
214	1
224	1
228	1

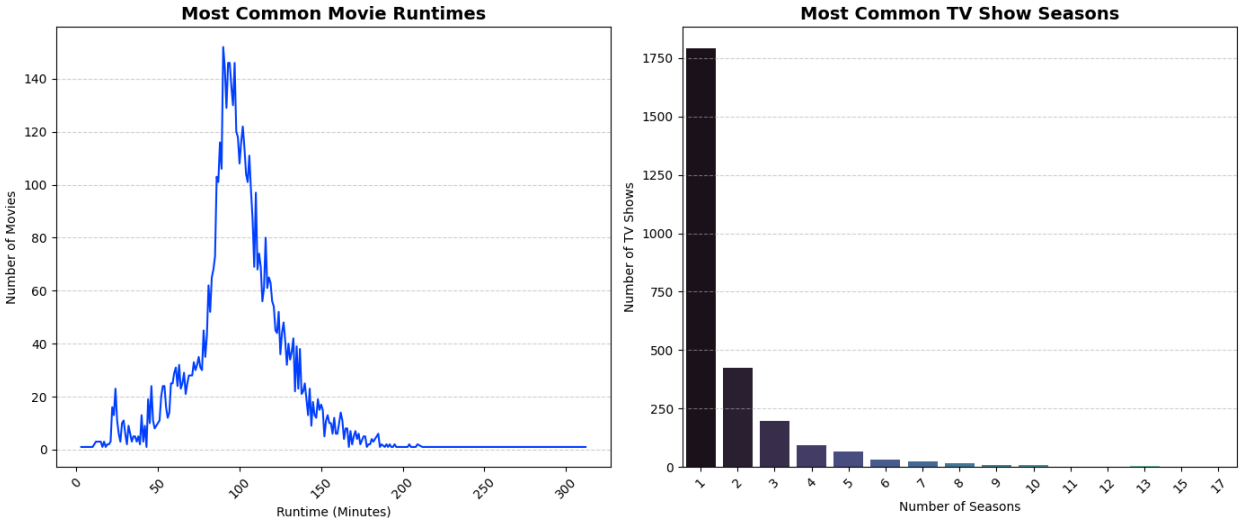
229	1
230	1
233	1
237	1
253	1
273	1
312	1

```
tvs = tvd.groupby(['no_of_seasons'])[['title']].nunique().sort_values(by='title'
,ascending=False)
tvs
```

no_of_seasons	title
1	1793
2	425
3	199
4	95
5	65
6	33
7	23
8	17
9	9
10	7
13	3
11	2
12	2
15	2
17	1

```
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
sns.lineplot(data =mrm,  x='runtime_in_mins', y='title', palette="viridis")
plt.title("Most Common Movie Runtimes", fontsize=14, fontweight="bold")
plt.xlabel("Runtime (Minutes)")
plt.ylabel("Number of Movies")
plt.xticks(rotation=45)
plt.grid(axis="y", linestyle="--", alpha=0.6)

plt.subplot(1, 2, 2)
sns.barplot(data=tvs, x='no_of_seasons', y='title', palette="mako")
plt.title("Most Common TV Show Seasons", fontsize=14, fontweight="bold")
plt.xlabel("Number of Seasons")
plt.ylabel("Number of TV Shows")
plt.xticks(rotation=45)
```

```
plt.grid(axis="y", linestyle="--", alpha=0.6)

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



Insights :

- The most frequent movie runtimes are between 90-100 minutes, suggesting an optimal duration for audience engagement.
-

  Most TV shows have 1-3 seasons, indicating a trend toward shorter series or limited runs.

Q. What are the ratings given for the contents uploaded on netflix ?

```
movie_rating = md.groupby(['rating'])[['title']].nunique().reset_index()
movie_rating = movie_rating.sort_values(by='title',ascending=False)
movie_rating
```

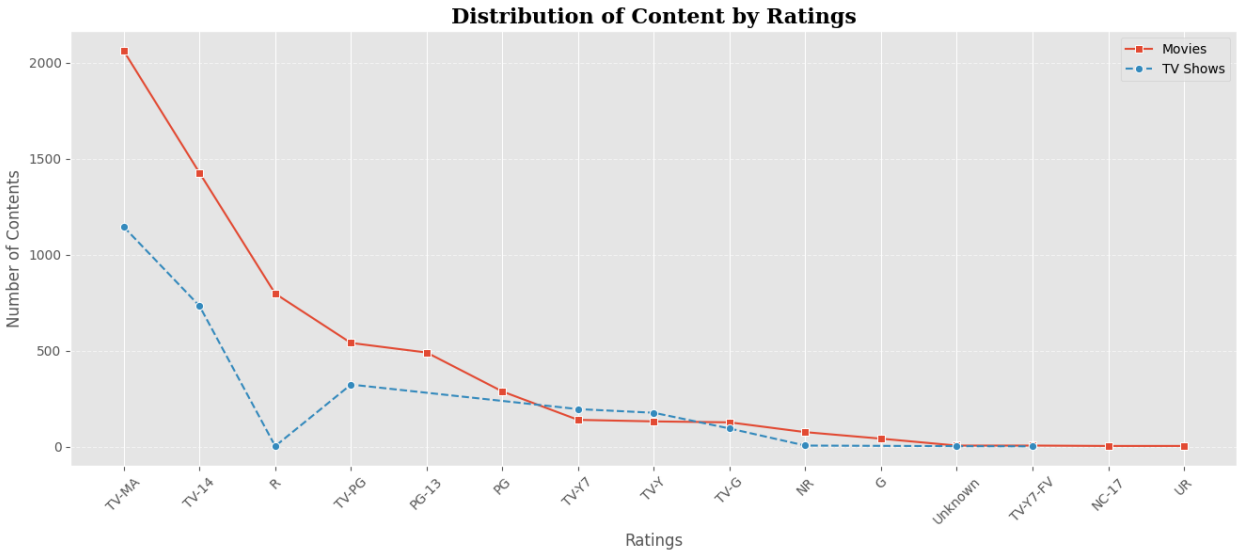
	rating	title
8	TV-MA	2062
6	TV-14	1427
5	R	797
9	TV-PG	540
4	PG-13	490
3	PG	287

	rating	title
11	TV-Y7	139
10	TV-Y	131
7	TV-G	126
2	NR	75
0	G	41
14	Unknown	5
12	TV-Y7-FV	5
1	NC-17	3
13	UR	3

```
tv_rating = tvd.groupby(['rating'])[['title']].nunique().reset_index()
tv_rating = tv_rating.sort_values(by='title',ascending=False)
tv_rating
```

	rating	title
4	TV-MA	1145
2	TV-14	733
5	TV-PG	323
7	TV-Y7	195
6	TV-Y	176
3	TV-G	94
0	NR	5
1	R	2
9	Unknown	2
8	TV-Y7-FV	1

```
plt.figure(figsize=(16,6))
plt.style.use('ggplot')

sns.lineplot(data=movie_rating, x='rating', y='title', label='Movies',
marker='s', linestyle='-')
sns.lineplot(data=tv_rating, x='rating', y='title', label='TV Shows', marker='o',
linestyle='--')

plt.title('Distribution of Content by Ratings', fontsize=16, fontweight="bold",
fontfamily='serif')
plt.ylabel('Number of Contents')
plt.xlabel('Ratings')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.6)

plt.legend(loc='upper right', frameon=True)

plt.show()
```



Insights :

Movies - The most frequent rating is "TV-MA", with 2,062 titles, indicating that a large portion of Netflix's content is tailored for mature audiences.

- "TV-14", with 1,427 titles, is the second most common rating, signifying content appropriate for viewers aged 14 and older.
- "R" (Restricted) comes in third with 797 titles, meaning viewers under 17 require parental or adult supervision.

TV Shows

- "TV-MA" leads with 1,145 titles, showing that a significant number of TV shows are designed for mature viewers.
- "TV-14", with 733 titles, follows closely behind, catering to teen audiences.
-

"TV-PG" ranks third with 323 titles, suggesting a notable selection of family-friendly shows that may require parental guidance.

Q. Top actors with the most appearances in Netflix content.

```
movies_cast = md.groupby('cast')[['title']].nunique().sort_values(by='title'
,ascending=False)[1:11]
movies_cast
```

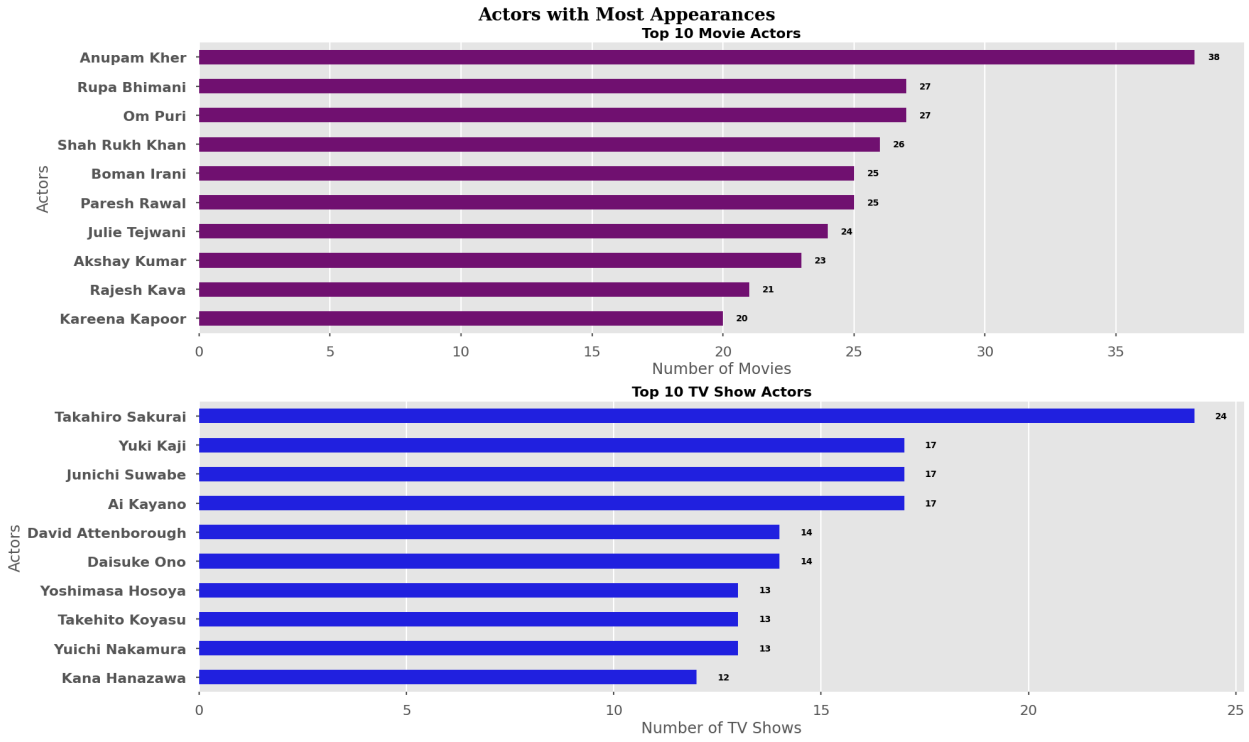
cast	title
Anupam Kher	38
Rupa Bhimani	27
Om Puri	27
Shah Rukh Khan	26
Boman Irani	25
Paresh Rawal	25
Julie Tejwani	24
Akshay Kumar	23
Rajesh Kava	21
Kareena Kapoor	20

```
tv_cast = tvd.groupby('cast')[['title']].nunique().sort_values(by='title'
,ascending=False)[1:11]
tv_cast
```

cast	title
Takahiro Sakurai	24
Yuki Kaji	17
Junichi Suwabe	17
Ai Kayano	17
David Attenborough	14
Daisuke Ono	14
Yoshimasa Hosoya	13
Takehito Koyasu	13
Yuichi Nakamura	13
Kana Hanazawa	12

```
plt.figure(figsize=(20,12))
plt.suptitle('Actors with Most Appearances',
             fontsize=20, fontweight="bold", fontfamily='serif')
plt.style.use('seaborn-v0_8-poster')

plt.subplot(2,1,1)
c1 = sns.barplot(y=movies_cast.index[:10], x=movies_cast.title[:10],
color="purple", width=0.5)
sns.despine(left=True, bottom=True, trim=True)
plt.title('Top 10 Movie Actors', fontsize=16, fontweight="bold")
plt.xlabel('Number of Movies')
plt.ylabel('Actors')
plt.yticks(fontweight='bold')
```

```
for i, v in enumerate(movies_cast.title[:10]):
    plt.text(v + 0.5, i, str(v), color='black', ha='left', va='center',
fontweight='bold')

plt.subplot(2,1,2)
c2 = sns.barplot(y=tv_cast.index[:10], x=tv_cast.title[:10], color="blue",
width=0.5)
sns.despine(left=True, bottom=True, trim=True)
plt.title('Top 10 TV Show Actors', fontsize=16, fontweight="bold")
plt.xlabel('Number of TV Shows')
plt.ylabel('Actors')
plt.yticks(fontweight='bold')

for i, v in enumerate(tv_cast.title[:10]):
    plt.text(v + 0.5, i, str(v), color='black', ha='left', va='center',
fontweight='bold')

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



Insights :

Movies - Anupam Kher has appeared in the highest number of Netflix movies.

- Rupa Bhimani and Om Puri follow closely.

- Shah Rukh Khan, Boman Irani, and Paresh Rawal have also been featured prominently.

TV Shows

- Takahiro Sakurai leads in TV appearances.
-

Yuki Kaji, Junichi Suwabe, and Ai Kayano follow closely.

Q. Top directors with the most content in Netflix.

```
movie_director = md.groupby('director')[['show_id']].nunique().sort_values(by=
'show_id',ascending=False)[1:11]
movie_director
```

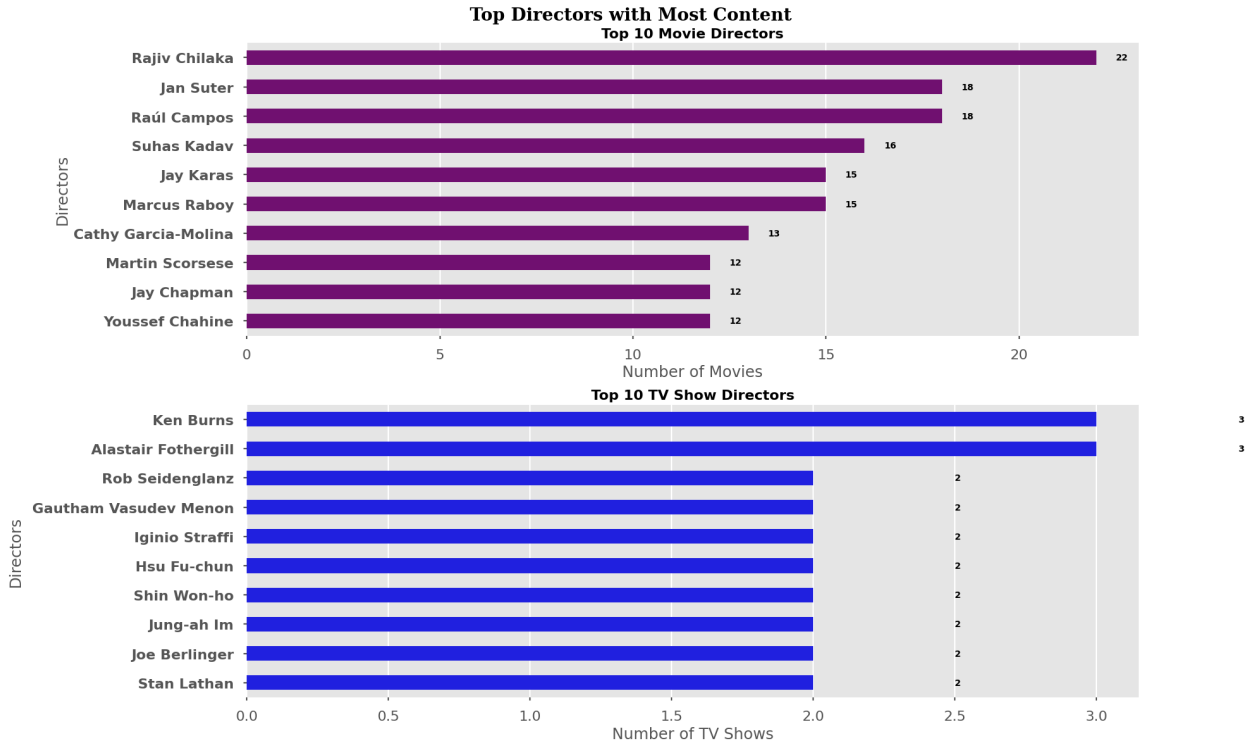
director	show_id
Rajiv Chilaka	22
Jan Suter	18
Raúl Campos	18
Suhas Kadav	16
Jay Karas	15
Marcus Raboy	15
Cathy Garcia-Molina	13
Martin Scorsese	12
Jay Chapman	12
Youssef Chahine	12

```
tv_director = tvd.groupby('director')[['show_id']].nunique().sort_values(by=
'show_id',ascending=False)[1:11]
tv_director
```

director	show_id
Ken Burns	3
Alastair Fothergill	3
Rob Seidenglanz	2
Gautham Vasudev Menon	2
Iginio Straffi	2
Hsu Fu-chun	2
Shin Won-ho	2
Jung-ah Im	2
Joe Berlinger	2

director	show_id
Stan Lathan	2

```
plt.figure(figsize=(20,12))
plt.suptitle('Top Directors with Most Content',
             fontsize=20, fontweight="bold", fontfamily='serif')
plt.style.use('seaborn-v0_8-poster')

plt.subplot(2,1,1)
c1 = sns.barplot(y=movie_director.index[:10], x=movie_director.show_id[:10],
color="purple", width=0.5)
sns.despine(left=True, bottom=True, trim=True)
plt.title('Top 10 Movie Directors', fontsize=16, fontweight="bold")
plt.xlabel('Number of Movies')
plt.ylabel('Directors')
plt.yticks(fontweight='bold')

for i, v in enumerate(movie_director.show_id[:10]):
    plt.text(v + 0.5, i, str(v), color='black', ha='left', va='center',
fontweight='bold')

plt.subplot(2,1,2)
c2 = sns.barplot(y=tv_director.index[:10], x=tv_director.show_id[:10],
color="blue", width=0.5)
sns.despine(left=True, bottom=True, trim=True)
plt.title('Top 10 TV Show Directors', fontsize=16, fontweight="bold")
plt.xlabel('Number of TV Shows')
plt.ylabel('Directors')
plt.yticks(fontweight='bold')

for i, v in enumerate(tv_director.show_id[:10]):
    plt.text(v + 0.5, i, str(v), color='black', ha='left', va='center',
fontweight='bold')

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



Insights :

Movies - Rajiv Chilaka has directed the most Netflix movies.

TV Shows

- **Ken Burns and Alastair Fothergill lead in TV show direction.**

Q. What is the best time to launch a movie?

```
movie_release = md[['show_id','title','date_added']]
movie_release = movie_release.reset_index(drop=True)
movie_release
```

	show_id	title	date_added
0	s1	Dick Johnson Is Dead	2021-09-25
1	s7	My Little Pony: A New Generation	2021-09-24
2	s7	My Little Pony: A New Generation	2021-09-24
3	s7	My Little Pony: A New Generation	2021-09-24
4	s7	My Little Pony: A New Generation	2021-09-24
...	...	...	...
145912	s8807	Zubaan	2019-03-02

	show_id	title	date_added
145913	s8807	Zubaan	2019-03-02
145914	s8807	Zubaan	2019-03-02
145915	s8807	Zubaan	2019-03-02
145916	s8807	Zubaan	2019-03-02

```
movie_release.dtypes
```

	0
show_id	object
title	object
date_added	object

```
movie_release['date_added'] = pd.to_datetime(movie_release['date_added'])
```

```
movie_release.dtypes
```

	0
show_id	object
title	object
date_added	datetime64[ns]

```
movie_release.isna().sum()
```

	0
show_id	0
title	0
date_added	0

```
movie_release['uploaded_week'] =
movie_release['date_added'].dt.isocalendar().week
movie_release['uploaded_weekday'] = movie_release['date_added'].dt.strftime('%A')
movie_release['uploaded_month'] = movie_release['date_added'].dt.strftime('%B')
```

```
month_order = ['January', 'February', 'March', 'April', 'May',
               'June', 'July', 'August', 'September',
               'October', 'November', 'December']
movie_release['uploaded_month']= pd.Categorical(movie_release['uploaded_month'],
                                        categories=month_order, ordered=True)
```

```
movie_release
```

	show_id	title	date_added	uploaded_week	uploaded_weekday	u
0	s1	Dick Johnson Is Dead	2021-09-25	38	Saturday	S
1	s7	My Little Pony: A New Generation	2021-09-24	38	Friday	S
2	s7	My Little Pony: A New Generation	2021-09-24	38	Friday	S
3	s7	My Little Pony: A New Generation	2021-09-24	38	Friday	S
4	s7	My Little Pony: A New Generation	2021-09-24	38	Friday	S
...	...	...	...	...	...	.
145912	s8807	Zubaan	2019-03-02	9	Saturday	I
145913	s8807	Zubaan	2019-03-02	9	Saturday	I
145914	s8807	Zubaan	2019-03-02	9	Saturday	I
145915	s8807	Zubaan	2019-03-02	9	Saturday	I
145916	s8807	Zubaan	2019-03-02	9	Saturday	I

```
week_movie_release=movie_release.groupby('uploaded_week')['show_id'].nunique()
week_movie_release=week_movie_release.reset_index()
week_movie_release
```

	uploaded_week	show_id
0	1	316
1	2	78
2	3	81
3	4	56
4	5	135
5	6	64
6	7	106
7	8	72
8	9	207
9	10	107
10	11	115
11	12	67
12	13	174
13	14	124
14	15	100
15	16	124
16	17	109
17	18	173
18	19	73
19	20	85
20	21	76
21	22	146
22	23	112

	uploaded_week	show_id
23	24	89
24	25	101
25	26	195
26	27	154
27	28	89
28	29	94
29	30	116
30	31	185
31	32	73
32	33	105
33	34	102
34	35	189
35	36	97
36	37	114
37	38	88
38	39	111
39	40	215
40	41	84
41	42	90
42	43	88
43	44	243
44	45	61
45	46	83
46	47	85
47	48	139
48	49	95
49	50	119
50	51	86
51	52	80
52	53	61

```
week_movie_release.sum()
```

	0
uploaded_week	1431.0
show_id	6131.0

```
monthly_movie_release=movie_release.groupby('uploaded_month')['show_id'
].nunique()
monthly_movie_release = monthly_movie_release.reset_index()
monthly_movie_release = monthly_movie_release.sort_values(by='uploaded_month')
monthly_movie_release.reset_index(drop=True)
monthly_movie_release
```

	uploaded_month	show_id
0	January	546
1	February	382
2	March	529
3	April	550
4	May	439
5	June	492
6	July	565
7	August	519
8	September	519
9	October	545
10	November	498
11	December	547

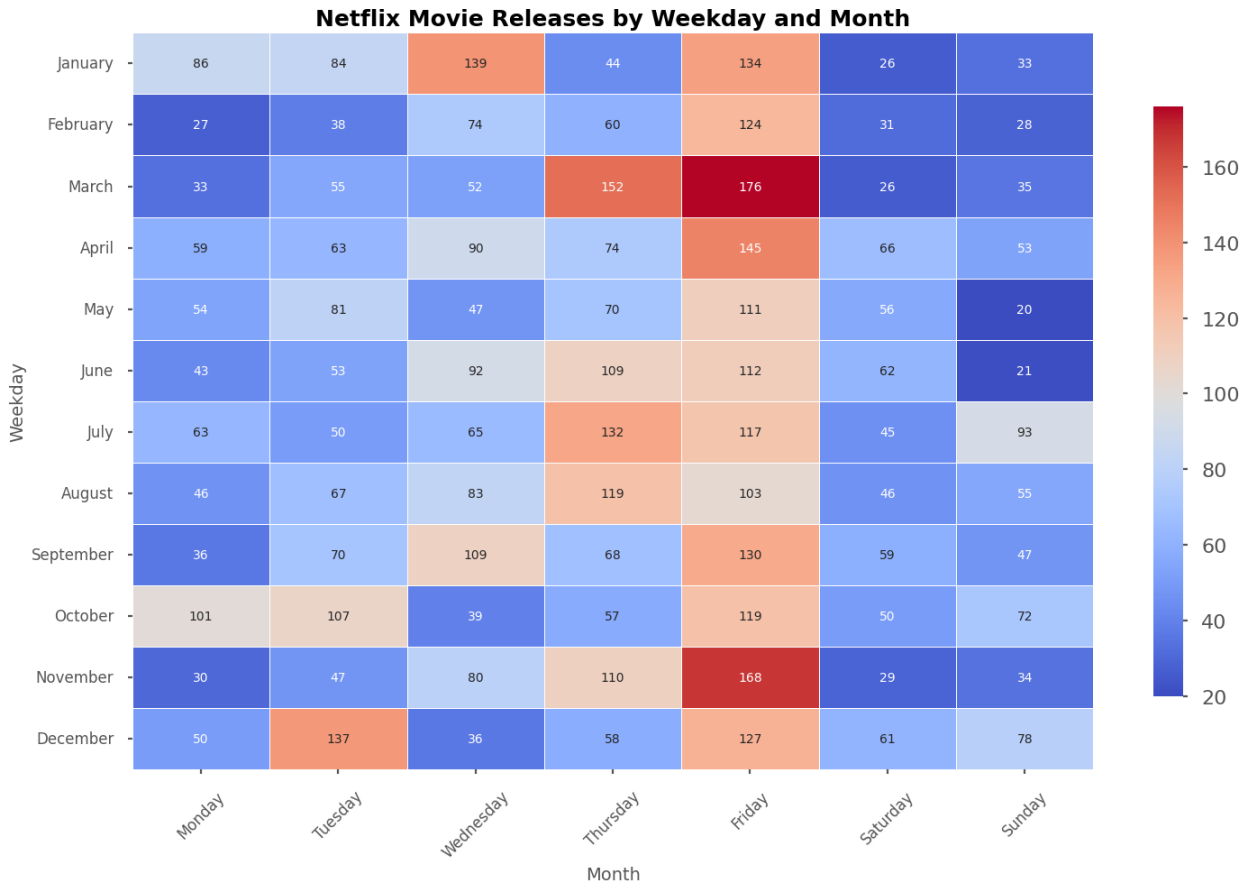
```
monthly_movie_release.show_id.sum()
```

```
np.int64(6131)
```

```
movies_release_pivot = movie_release.pivot_table(index='uploaded_month',
                                                 columns='uploaded_weekday',
                                                 values='show_id',
                                                 aggfunc=pd.Series.nunique)

day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday',
             'Friday', 'Saturday','Sunday']
movies_release_pivot = movies_release_pivot[day_order]
movies_release_pivot
```

uploaded_weekday uploaded_month	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
January	86	84	139	44	134	26	33
February	27	38	74	60	124	31	28
March	33	55	52	152	176	26	35
April	59	63	90	74	145	66	53
May	54	81	47	70	111	56	20
June	43	53	92	109	112	62	21
July	63	50	65	132	117	45	93
August	46	67	83	119	103	46	55
September	36	70	109	68	130	59	47
October	101	107	39	57	119	50	72
November	30	47	80	110	168	29	34
December	50	137	36	58	127	61	78

```
plt.figure(figsize=(15, 10))

sns.heatmap(movies_release_pivot, cmap='coolwarm',
            annot=True, fmt='d', linewidth=0.5, cbar_kws={"shrink": 0.8})

plt.title("Netflix Movie Releases by Weekday and Month", fontsize=18,
fontweight='bold')
plt.xlabel("Month", fontsize=14)
plt.ylabel("Weekday", fontsize=14)

plt.xticks(fontsize=12, rotation=45)
plt.yticks(fontsize=12, rotation=0)

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




```
movies_release_pivot.sum().sort_values(ascending=False)
```

uploaded_weekday	0
Friday	1566
Thursday	1053
Wednesday	906
Tuesday	852
Monday	628
Sunday	569
Saturday	557

```
movies_release_pivot.sum().sum()
```

```
np.int64(6131)
```

Q. What is the best time to launch a tv show?

```
tvs_release = tvd[['show_id','title','date_added']]
tvs_release = tvs_release.reset_index(drop=True)
tvs_release
```

	show_id	title	date_added
0	s2	Blood & Water	2021-09-24
1	s2	Blood & Water	2021-09-24
2	s2	Blood & Water	2021-09-24
3	s2	Blood & Water	2021-09-24
4	s2	Blood & Water	2021-09-24
...	...	...	...
56143	s8801	Zindagi Gulzar Hai	2016-12-15
56144	s8801	Zindagi Gulzar Hai	2016-12-15
56145	s8804	Zombie Dumb	2019-07-01
56146	s8804	Zombie Dumb	2019-07-01
56147	s8804	Zombie Dumb	2019-07-01

```
tvs_release.dtypes
```

	0
show_id	object
title	object
date_added	object

```
tvs_release.isna().sum()
```

	0
show_id	0
title	0
date_added	0

```
tvs_release['date_added'].fillna(tvs_release['date_added'].mode()[0],inplace=
True)
```

```
tvs_release['date_added'] = pd.to_datetime(tvs_release['date_added'])
```

```
tvs_release['date_added'].dtypes
```

```
dtype('<M8[ns]')
```

```
tvs_release['uploaded_week'] = tvs_release['date_added'].dt.isocalendar().week
tvs_release['uploaded_weekday'] = tvs_release['date_added'].dt.strftime('%A')
tvs_release['uploaded_month'] = tvs_release['date_added'].dt.strftime('%B')
```

```
month_order = ['January', 'February', 'March', 'April', 'May',
               'June', 'July', 'August', 'September',
               'October', 'November', 'December']
tvs_release['uploaded_month']= pd.Categorical(tvs_release['uploaded_month'],
                                    categories=month_order, ordered=True)
```

```
tvs_release
```

	show_id	title	date_added	uploaded_week	uploaded_weekday	uploaded_month
0	s2	Blood & Water	2021-09-24	38	Friday	September
1	s2	Blood & Water	2021-09-24	38	Friday	September
2	s2	Blood & Water	2021-09-24	38	Friday	September
3	s2	Blood & Water	2021-09-24	38	Friday	September
4	s2	Blood & Water	2021-09-24	38	Friday	September
...	...	...	...	...	...	...
56143	s8801	Zindagi Gulzar Hai	2016-12-15	50	Thursday	December
56144	s8801	Zindagi Gulzar Hai	2016-12-15	50	Thursday	December
56145	s8804	Zombie Dumb	2019-07-01	27	Monday	July
56146	s8804	Zombie Dumb	2019-07-01	27	Monday	July
56147	s8804	Zombie Dumb	2019-07-01	27	Monday	July

```
tvs_release.groupby('uploaded_week')['show_id'].nunique().sum()
```

```
np.int64(2676)
```

```
week_release = tvs_release.groupby('uploaded_week')['show_id'].nunique()
week_release = week_release.reset_index()
week_release
```

	uploaded_week	show_id
0	1	150
1	2	26
2	3	31
3	4	31
4	5	68
5	6	33
6	7	41
7	8	37
8	9	46
9	10	28
10	11	46
11	12	40
12	13	73
13	14	48
14	15	50
15	16	34
16	17	45
17	18	60
18	19	43
19	20	44
20	21	39
21	22	56
22	23	39
23	24	75
24	25	42
25	26	69
26	27	85
27	28	40
28	29	44
29	30	43
30	31	79
31	32	49
32	33	47
33	34	40
34	35	73
35	36	44

	uploaded_week	show_id
36	37	67
37	38	50
38	39	55
39	40	69
40	41	31
41	42	45
42	43	28
43	44	67
44	45	36
45	46	51
46	47	35
47	48	56
48	49	44
49	50	65
50	51	48
51	52	50
52	53	41

```
month_release = tvs_release.groupby('uploaded_month')['show_id'].nunique()
month_release = month_release.reset_index()
month_release
```

	uploaded_month	show_id
0	January	279
1	February	175
2	March	205
3	April	209
4	May	187
5	June	232
6	July	254
7	August	230
8	September	246
9	October	210
10	November	199
11	December	250

```
tvs_release_pivot = tvs_release.pivot_table(
            index='uploaded_month' ,
            columns='uploaded_weekday' ,
            values='show_id' ,
            aggfunc=pd.Series.nunique
    )
```

```
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday',
                  'Friday', 'Saturday','Sunday']
tvs_release_pivot = tvs_release_pivot[day_order]
tvs_release_pivot
```

uploaded_weekday uploaded_month	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
January	13	19	128	19	85	5	10
February	20	13	28	27	63	17	7
March	11	25	32	22	88	16	11
April	20	20	38	38	58	21	14
May	21	26	25	15	74	17	9
June	22	16	35	29	77	47	6
July	22	50	32	29	82	29	10
August	23	41	30	26	80	18	12
September	13	29	44	34	87	15	24
October	19	25	21	35	64	26	20
November	6	29	27	21	85	11	20
December	27	37	28	39	67	24	28

```
tvs_release_pivot.sum(axis=1)
```

uploaded_month	0
January	279
February	175
March	205
April	209
May	187
June	232
July	254
August	230
September	246
October	210
November	199
December	250

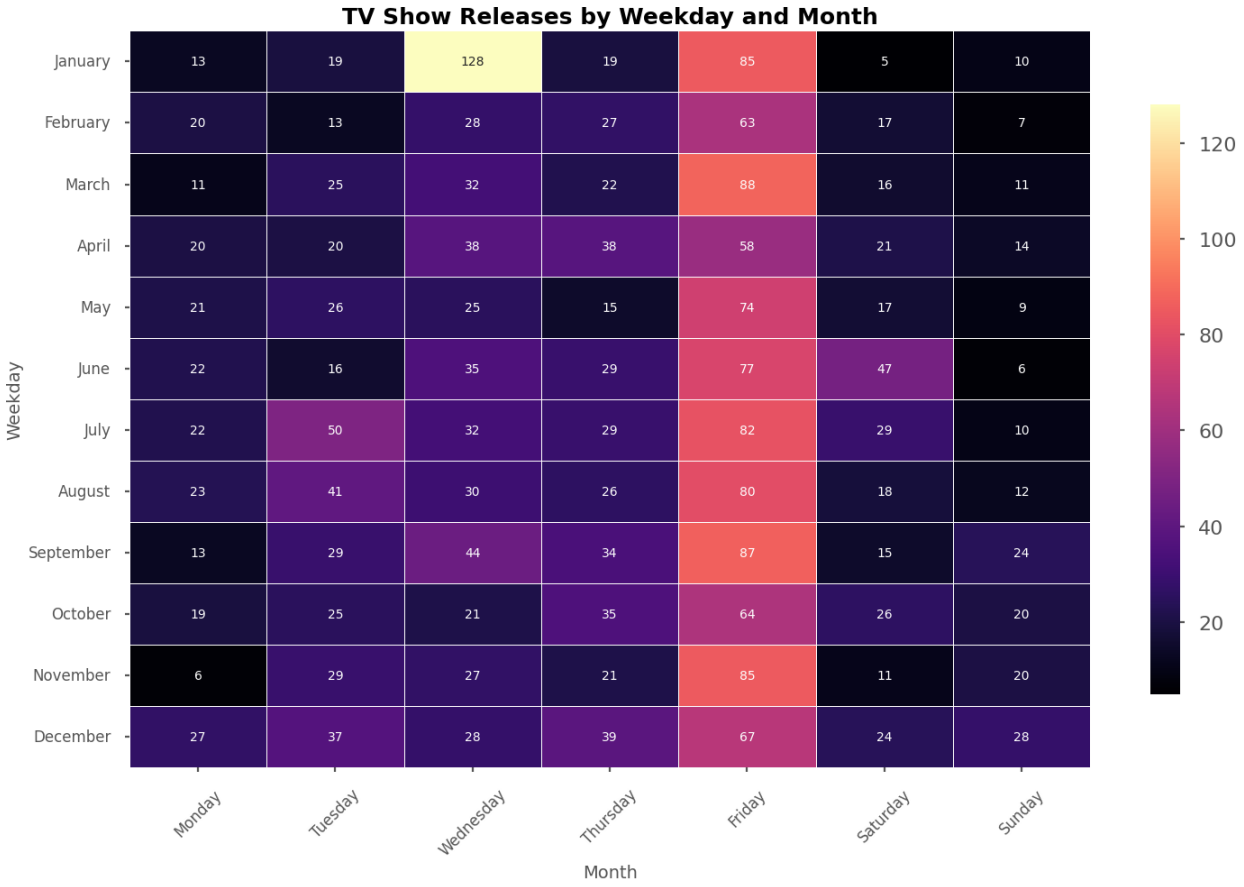
```
tvs_release_pivot.sum().sort_values(ascending=False)
```

uploaded_weekday	0
Friday	910
Wednesday	468
Thursday	334
Tuesday	330
Saturday	246
Monday	217
Sunday	171

```
tvs_release_pivot.sum().sum()
```

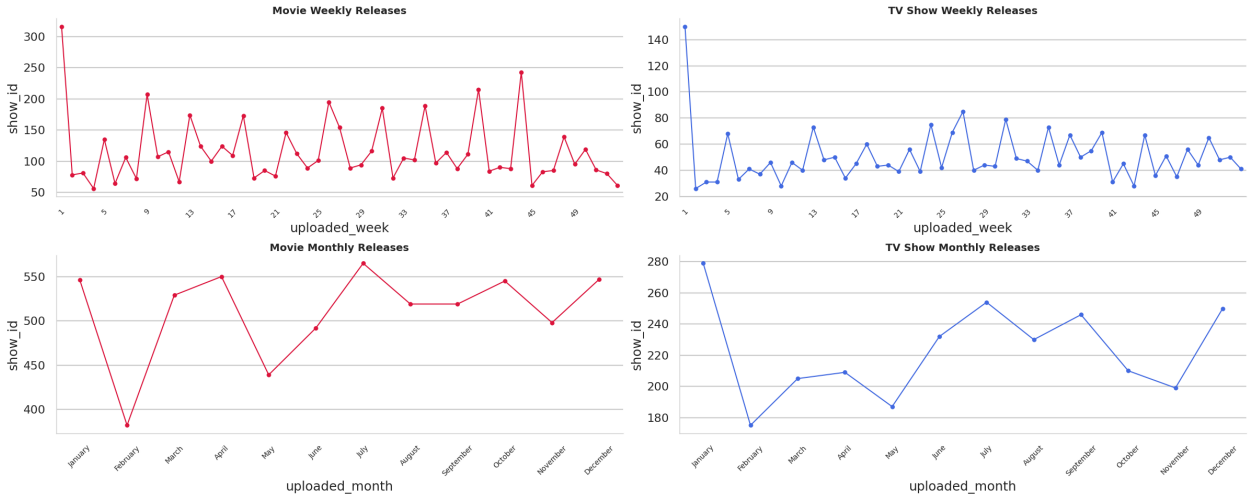
```
np.int64(2676)
```

```
plt.figure(figsize=(15, 10))

sns.heatmap(tvs_release_pivot, cmap='magma', annot=True,
            fmt='d', linewidth=0.5, cbar_kws={"shrink": 0.8})

plt.title("TV Show Releases by Weekday and Month", fontsize=18,
fontweight='bold')
plt.xlabel("Month", fontsize=14)
plt.ylabel("Weekday", fontsize=14)

plt.xticks(fontsize=12, rotation=45)
plt.yticks(fontsize=12, rotation=0)

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




```
sns.set_style("whitegrid")
plt.figure(figsize=(24, 10), facecolor='white')

plt.subplot(2, 2, 1)
sns.pointplot(data=week_movie_release, x='uploaded_week', y='show_id',
              color="crimson", linewidth=1.5, markersize=5)
plt.title('Movie Weekly Releases', fontsize=14, fontweight='bold')

ticks = np.arange(0, 52, 4)
plt.xticks(ticks, fontsize=10, rotation=45)
sns.despine()

plt.subplot(2, 2, 3)
sns.pointplot(data=monthly_movie_release, x='uploaded_month', y='show_id',
              color="crimson", linewidth=1.5, markersize=5)
plt.title('Movie Monthly Releases', fontsize=14, fontweight='bold')

plt.xticks(fontsize=10, rotation=45)
sns.despine()

plt.subplot(2, 2, 2)
sns.pointplot(data=week_release, x='uploaded_week', y='show_id',
```

```
                color="royalblue", linewidth=1.5, markersize=5)
plt.title('TV Show Weekly Releases', fontsize=14, fontweight='bold')

plt.xticks(ticks, fontsize=10, rotation=45)
sns.despine()

plt.subplot(2, 2, 4)
sns.pointplot(data=month_release, x='uploaded_month', y='show_id',
                color="royalblue", linewidth=1.5, markersize=5)
plt.title('TV Show Monthly Releases', fontsize=14, fontweight='bold')

plt.xticks(fontsize=10, rotation=45)
sns.despine()

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



Insights :

Movies - Friday of every Week,Month throughout the year, has peaked stating that weekends are great time for movie releases.

- July, October, April have topped at releasing because of the holidays.
- December & January also have recorded high releases on holidays.

TV Shows

- Wednesday , Thrusday and Friday seems to be good days releasing new Tv Shows.
-

July,September, December, January are also having a higher than average TvShow releases.

Business Proposal: Strategic Recommendations for Netflix Growth

Introduction
Netflix has established itself as a global leader in the streaming industry, offering a vast content library and innovative viewing experiences.

However, with increasing competition from platforms such as Disney+, Amazon Prime, and HBO Max, it is essential for Netflix to continually evolve.

This proposal outlines key strategic recommendations that will help Netflix attract new subscribers, enhance engagement, and maintain its competitive edge.

---

1. Expanding Family-Friendly Content

- While Netflix primarily caters to mature and teenage audiences, increasing family-friendly options (**TV-PG, G-rated content**) can expand its subscriber base.
- Investment in **animated films and educational series** will attract parents and younger viewers.

---

2. Balancing Movie and TV Show Offerings

- Netflix's content library consists of approximately **70% movies and 30% TV shows**.
- Increasing high-quality **TV series, particularly limited series (1-3 seasons)**, will enhance viewer engagement and retention.

---

3. Strengthening Global Content Strategy

- With major content-producing countries being the **U.S., U.K., India, Canada, Japan, and South Korea**, Netflix should continue investing in international productions.
- Expanding **Asian content**, especially localized productions in **Japan and South Korea**, will tap into growing audiences.
- Strengthen partnerships with **regional film industries** for unique storytelling opportunities.

---

4. **Optimizing Release Strategy for Maximum Engagement**

- Maintain **movie releases on Fridays** to capitalize on weekend viewership.
- Continue **TV show releases on Wednesdays, Thursdays, and Fridays** for peak audience engagement.
- Focus content releases around high-traffic months (**July, October, December, January**) to leverage holiday streaming trends.

---

5. **Investing in Popular Genres**

- **Drama, Comedy, Crime, and Documentaries** are in high demand across both movies and TV shows.
- Increase investment in **crime-thriller series** and **comedy specials** to sustain audience interest.

---

6. **Leveraging Data for Ideal Movie Runtime and TV Show Length**

- Most successful movies fall within the **90-100 minute range**, aligning new productions with this format for optimal engagement.
- **Shorter TV series (1-3 seasons)** align with viewer preferences and reduce content fatigue.

---

7. **Collaborating with High-Profile Actors and Directors**

- Actors such as **Anupam Kher, Shah Rukh Khan, and Takahiro Sakurai** have significant appeal.
- Strengthening collaborations with top directors like **Rajiv Chilaka (movies) and Ken Burns (TV shows)** can attract a larger audience.

---

8. **Strengthening Older Content Library**

- A significant portion of Netflix's content comes from **2017-2018**, with fewer older films.
- Acquiring **classic films and TV series from before 2010** can cater to nostalgia-driven audiences.

---

9. Experimenting with Limited Releases for Nostalgic Content

- Offering **older movies and TV shows as limited-time exclusives** can create urgency and attract older audiences.

---

10. Enhancing Regional Pricing & Subscription Models

- Introduce **region-specific pricing** to attract price-sensitive markets such as **India, Southeast Asia, and Latin America**.
- Expand **mobile-only and ad-supported subscription plans** to increase accessibility and affordability.

---

11. Leveraging AI & Personalization for Better User Engagement

- Utilize **AI-driven recommendations** to enhance personalized content discovery.
- Introduce **interactive content similar to Bandersnatch** to create immersive experiences.

Conclusion

While Netflix is already a market leader, there is always room for improvement.

By implementing these strategic recommendations, Netflix can continue growing, attract new subscribers, solidify its current success and also unlock new opportunities for expansion and innovation.