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

# 1 Load the dataset
df = pd.read_csv("/content/netflix_titles (1).csv")
df
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...
...	...	...	...	...	...	...	...	...	...	...	...	...
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	2007	R	158 min	Cult Movies, Dramas, Thrillers	A political cartoonist, a crime reporter and a...
8803	s8804	TV	Zombie	NaN	NaN	NaN	July 1, 2019	2018	TV-Y7	2	Kids' TV, Korean TV	While living alone in a

```
#2 Display first 5 rows
df.head()
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
												Feuds,

```
#3 Display last five rows
df.tail()
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	2007	R	158 min	Cult Movies, Dramas, Thrillers	A political cartoonist, a crime reporter and a...
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	2018	TV-Y7	2 Seasons	Kids' TV, Korean TV Shows, TV Comedies	While living alone in a spooky town, a young g...
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	2009	R	88 min	Comedies, Horror Movies	Looking to survive in a world taken over by zo...
					Tim Allen,							

```
# 4 Check for missing values
df.isnull().sum()
```

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

dtype: int64

```
#5 Fill missing values
df['rating'].fillna('TV-MA', inplace=True)
df['country'].fillna('United States', inplace=True)
```

```
#6 Check for missing values again
df.isnull().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

dtype: int64

```
#7 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5697 entries, 2 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       5697 non-null   object
 1   type          5697 non-null   object
 2   title         5697 non-null   object
 3   director      5697 non-null   object
 4   cast          5697 non-null   object
 5   country       5697 non-null   object
 6   date_added    5697 non-null   object
 7   release_year  5697 non-null   int64 
 8   rating        5697 non-null   object
 9   duration      5697 non-null   object
 10  listed_in     5697 non-null   object
 11  description   5697 non-null   object
dtypes: int64(1), object(11)
memory usage: 578.6+ KB
```

```
#Get some description of the data - ONLY release_year is NUMBER(S) VALUE(S)

print(df['release_year'].describe())
```

```
count    5697.000000
mean     2012.978936
std         9.564384
min      1942.000000
25%      2012.000000
50%      2016.000000
75%      2018.000000
max      2021.000000
Name: release_year, dtype: float64
```

```
# Columns in dataset
print(df.columns)
```

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

```
# Shape of the data set
print(df.shape)
```

```
(5697, 12)
```

```
#10. How many years of Data (1966-2021) (release_year)
print(df['release_year'].nunique())
```

```
72
```

```
# 11. MOVIES vs TV SHOWS (type)
print(df['type'].value_counts())
```

```
type
Movie      5519
TV Show     178
Name: count, dtype: int64
```

```
# Check for null values in all columns of dataset

df.isnull().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

dtype: int64

```
#Replace NAN value(s) with TV-MA

df['rating'].fillna('TV-MA', inplace=True)
df['rating']
```

	rating
2	TV-MA
5	TV-MA
6	PG
7	TV-MA
8	TV-14
...	...
8801	TV-MA
8802	R
8804	R
8805	PG
8806	TV-14

5697 rows × 1 columns

dtype: object

```
#Replace NAN value(s) with United States

df['country'].fillna('United States', inplace=True)
df['country']
```

	country
2	United States
5	United States
6	United States
7	United States, Ghana, Burkina Faso, United Kin...
8	United Kingdom
...	...
8801	United Arab Emirates, Jordan
8802	United States
8804	United States
8805	United States
8806	India

5697 rows × 1 columns

dtype: object

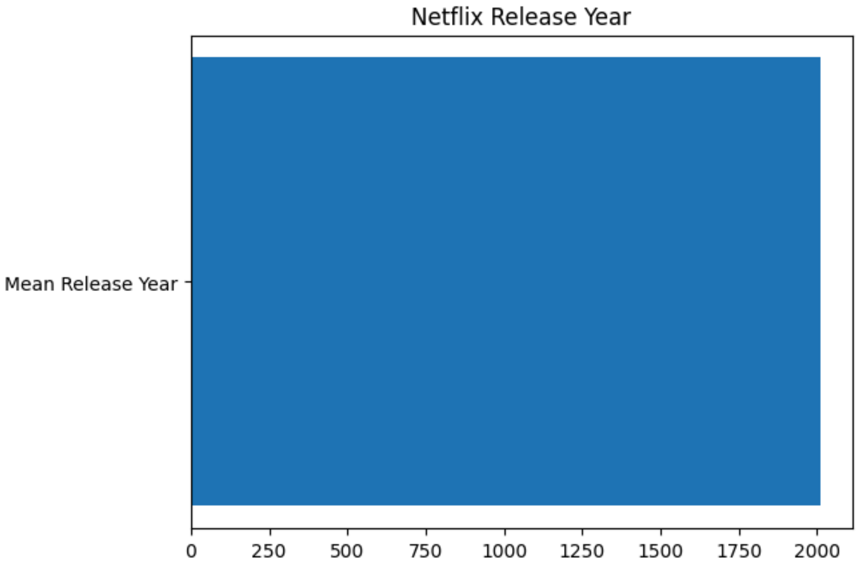
```
# Rename the Column “listed_in” to “genre”

df.rename(columns={'listed_in': 'genre'}, inplace=True)
print(df.columns)
```

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

Double-click (or enter) to edit

```
Start coding or generate with AI.
```

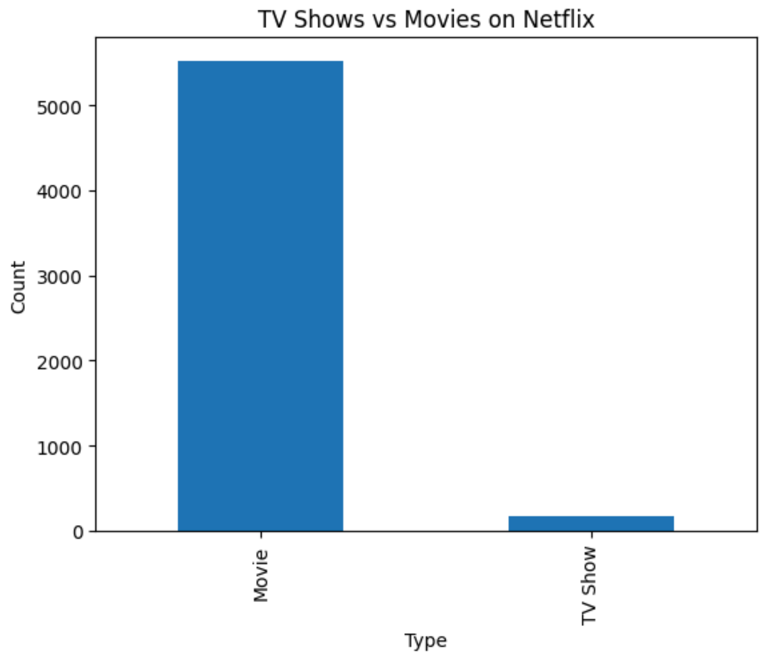
```
# Visualization of data using matplotlib and seaborn libraries
 #Calculating MEAN of release_year and show it using barh plot (Put title: Netflix release year)
import matplotlib.pyplot as plt
import seaborn as sns
mean_release_year = df['release_year'].mean()
plt.barh(['Mean Release Year'], [mean_release_year])
plt.title('Netflix Release Year')
plt.show()
```



Mean Release Year: This gives an idea of the average release year of content on Netflix.bold text

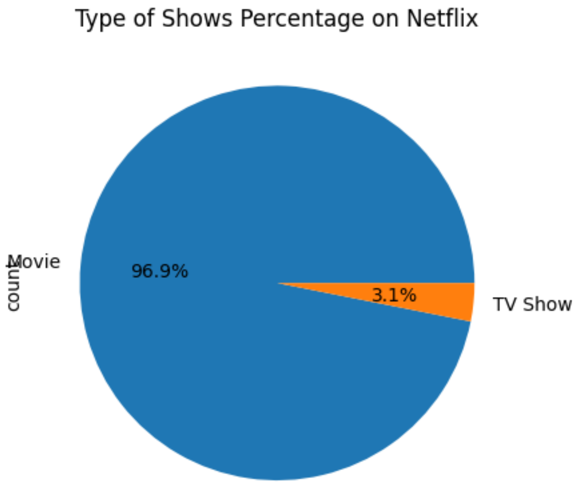
Insight: This plot gives average of 2000 movies or TV shows released in every year on Netflix

```
# TVshows vs Movies Netflix-Barchart
df['type'].value_counts().plot(kind='bar')
plt.title('TV Shows vs Movies on Netflix')
plt.xlabel('Type')
plt.ylabel('Count')
plt.show()
```



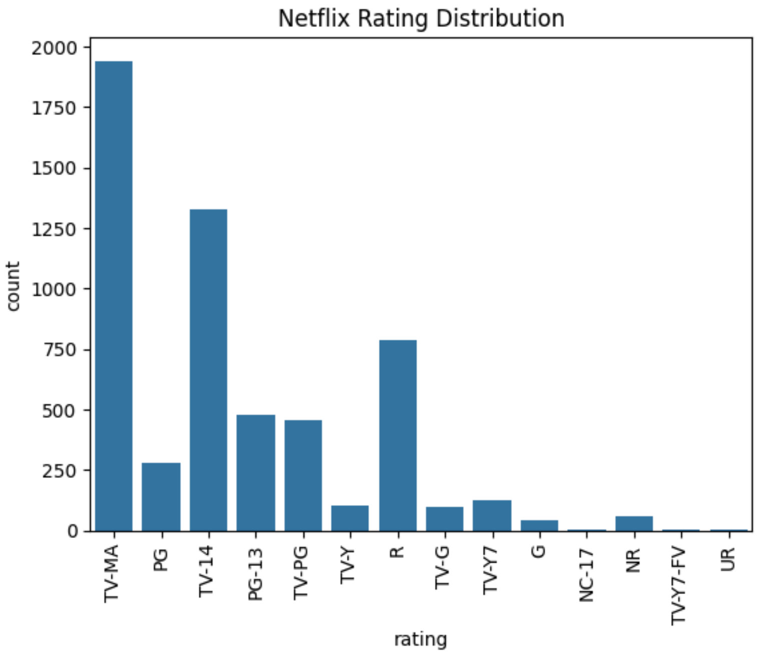
TV Shows vs Movies: Helps understand the distribution of content types.**bold text**

```
#3 Type of shows percentage on Netflix using piechart
df['type'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Type of Shows Percentage on Netflix')
plt.show()
```



Insight: This pie chart shows movies have more no of percentage than TV shows

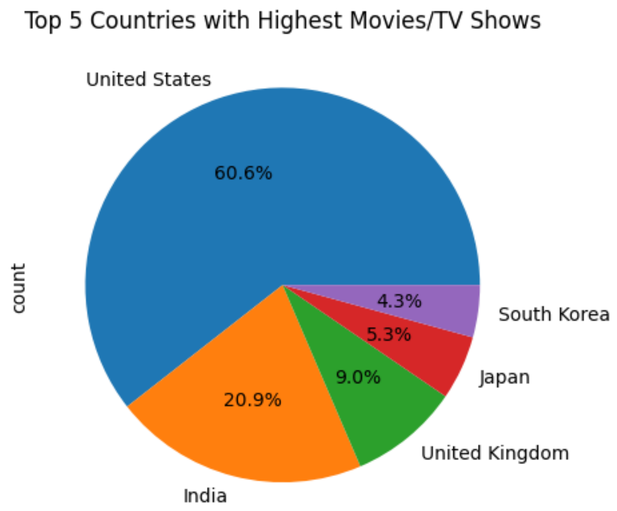
```
#4.Show Netflix Rating Distribution, visualised by a COUNT PLOT.

sns.countplot(data=df, x='rating')
plt.title('Netflix Rating Distribution')
plt.xticks(rotation=90)
plt.show()
```



Insight: This Shows have more number of Ratings in TV-MA

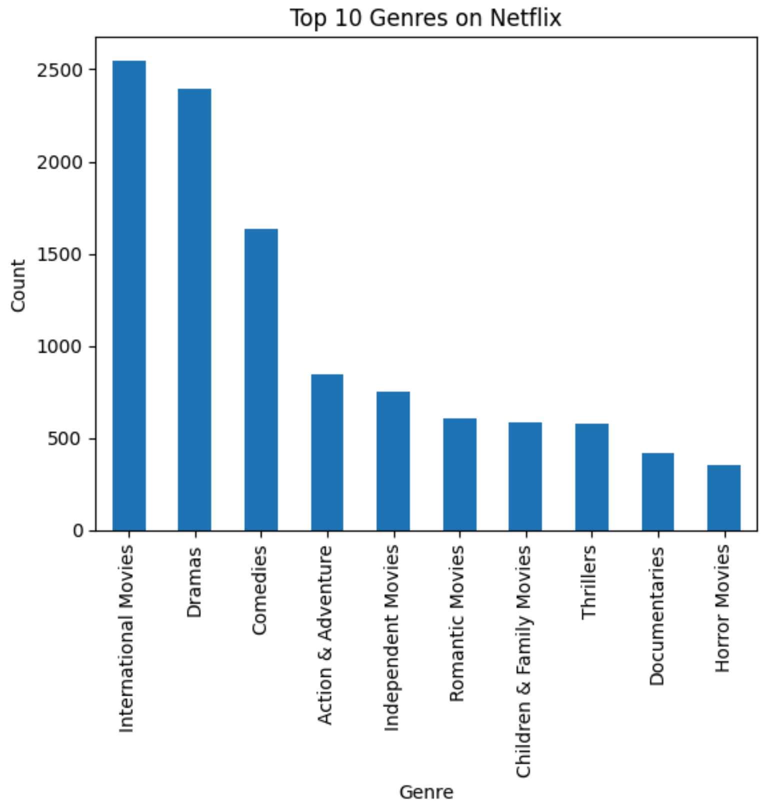
Rating Distribution: Shows the spread of different ratings, indicating content maturity levels.

```
#5. Show the Top 5 countries with Highest Movies/TV Shows, visualised by a PIE CHART.
top_countries = df['country'].value_counts().head(5)
top_countries.plot(kind='pie', autopct='%1.1f%%')
plt.title('Top 5 Countries with Highest Movies/TV Shows')
plt.show()
```



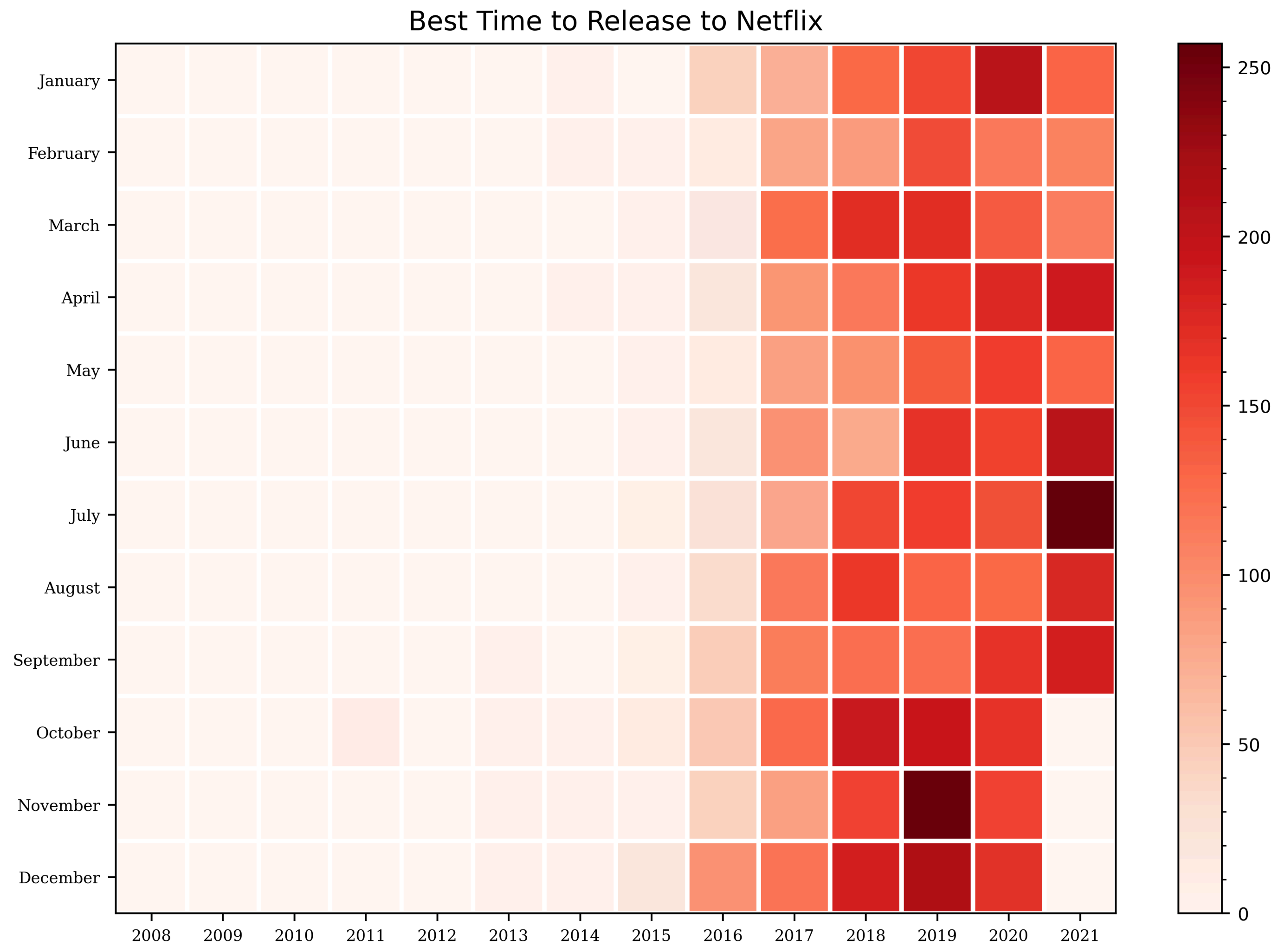
Insight: This Pie chart shows Netflix is more popular in United states

```
#7. Show the top 10 genres of movies and also tv shows.
top_genres = df['genre'].str.split(', ', expand=True).stack().value_counts().head(10)
top_genres.plot(kind='bar')
plt.title('Top 10 Genres on Netflix')
plt.xlabel('Genre')
plt.ylabel('Count')
plt.show()
```



Insight:This Bar plot shows more number of movies are viewed under International Movies category

```
df = df[['date_added']].dropna()
df['year'] = df['date_added'].apply(lambda x : x.split(', ')[-1])
df['month'] = df['date_added'].apply(lambda x : x.lstrip().split(' ')[0])

# Adding the Months and Grouping
month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'][::-1
df = df.groupby('year')['month'].value_counts().unstack().fillna(0)[month_order].T

# Customizing the figure design etc.
plt.figure(figsize=(10, 7), dpi=500)
plt.pcolor(df, cmap='Reds', edgecolors='white', linewidths=2) # heatmap

# Adding y and x ticks
plt.xticks(np.arange(0.5, len(df.columns), 1), df.columns, fontsize=7, fontfamily='serif')
plt.xticks(np.arange(0.5, len(df.columns), 1), df.columns, fontsize=7, fontfamily='serif')
plt.yticks(np.arange(0.5, len(df.index), 1), df.index, fontsize=7, fontfamily='serif')

# Adding the Title
plt.title('Best Time to Release to Netflix', fontsize=12, position=(0.50, 1.0+0.02))

# Adding the Colorbar
cbar = plt.colorbar()

cbar.ax.tick_params(labelsize=8)
cbar.ax.minorticks_on()
plt.show()
```



Insight:Best time to release any TV or movies in Netflix either by choosing January,July,october or November...

Best Release Time: Helps in understanding the optimal time for releasing new content based on historical data.**bold text**