```
!gdown 1bd9IujcsBhTO38WTqrsVAhwKDhHn9_QR
```

```
Downloading...
From: https://drive.google.com/uc?id=1bd9IujcsBhTO38WTqrsVAhwKDhHn9_QR
To: /content/netflix.csv
   0% 0.00/3.34M [00:00<?, ?B/s] 100% 3.34M/3.34M [00:00<00:00,
144MB/s]
```

```
from typing import NewType
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
netflix_data = pd.read_csv("netflix.csv")
netflix_data
```


```
{"summary":"{\n  \"name\": \"netflix_data\",\n  \"rows\": 8807,\n
\"fields\": [\n    {\n      \"column\": \"show_id\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 8807,\n        \"samples\": [\n
\"s4971\",\n          \"s3363\",\n          \"s5495\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"type\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"TV Show\",\n          \"Movie\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"title\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 8804,\n        \"samples\": [\n          \"The
Circle\",\n          \"The Old Thieves: The Legend of Artegios\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"director\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4528,\n        \"samples\": [\n
\"Kanwal Sethi\",\n          \"R\\u00e9my Four, Julien War\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"cast\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 7692,\n        \"samples\": [\n          \"Tzi
Ma, Christine Ko, Hong-Chi Lee, Hayden Szeto, Kunjue Li, Fiona Fu,
James Saito, Joan Chen\",\n          \"Priyanshu Painyuli,
Chandrachoor Rai, Shadab Kamal, Rajeev Siddhartha, Sheetal Thakur,
Ninad Kamat, Swati Semwal, Eijaz Khan\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"country\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
748,\n        \"samples\": [\n          \"United States, United
Kingdom, Denmark, Sweden\",\n          \"United Kingdom, Hong Kong\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"date_added\",\n
```

\"properties\": {\n   \"dtype\": \"object\",\n
\"num_unique_values\": 1767,\n   \"samples\": [\n   \"22-
Oct-18\",\n   \"29-Jan-21\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"release_year\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
8,\n   \"min\": 1925,\n   \"max\": 2021,\n
\"num_unique_values\": 74,\n   \"samples\": [\n   1996,\n
1969\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"rating\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 17,\n
\"samples\": [\n   \"PG-13\",\n   \"TV-MA\"\n   ],\
n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"duration\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 220,\n   \"samples\": [\n   \"37
min\",\n   \"177 min\"\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"listed_in\",\n   \"properties\": {\n
\"dtype\": \"category\",\n   \"num_unique_values\": 514,\n
\"samples\": [\n   \"Crime TV Shows, International TV Shows, TV
Mysteries\",\n   \"Children & Family Movies, Classic Movies,
Dramas\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"description\",\n   \"properties\": {\n   \"dtype\":
\"string\",\n   \"num_unique_values\": 8775,\n
\"samples\": [\n   \"A heedless teen drifter who falls for a
small-town waitress makes the mistake of robbing a drug lord, putting
his life and newfound love in jeopardy.\",\n   \"Twelve-year-
old Calvin manages to join the navy and serves in the battle of
Guadalcanal. But when his age is revealed, the boy is sent to the
brig.\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"netflix_data"}

#Counts of each categorical variable

##Non-Graphical Analysis: Value counts for categorical variables

---

```
# Count for 'type' (Movie/TV Show)
type_counts = netflix_data['type'].value_counts()
type_counts
```

```
type
Movie      6131
```

```
TV Show    2676
Name: count, dtype: int64
```

```
# count for rating

rating_counts = netflix_data['rating'].value_counts()
rating_counts
```

```
rating
TV-MA       3207
TV-14       2160
TV-PG        863
R            799
PG-13        490
TV-Y7        334
TV-Y         307
PG           287
TV-G         220
NR            80
G             41
TV-Y7-FV       6
NC-17          3
UR             3
74 min         1
84 min         1
66 min         1
Name: count, dtype: int64
```

```
# Count for 'country' (Top 10 countries)

country_counts = netflix_data['country'].value_counts().head(10)
country_counts
```

```
country
United States     2818
India              972
United Kingdom     419
Japan              245
South Korea        199
Canada             181
Spain              145
France             124
Mexico             110
Egypt              106
Name: count, dtype: int64
```

```
# Count for 'listed_in' (Top 10 genres)

genre_counts = netflix_data['listed_in'].value_counts().head(10)
genre_counts
```

```
listed_in
Dramas, International Movies                            362
Documentaries                                           359
Stand-Up Comedy                                         334
Comedies, Dramas, International Movies                  274
Dramas, Independent Movies, International Movies        252
Kids' TV                                                220
Children & Family Movies                                215
Children & Family Movies, Comedies                      201
Documentaries, International Movies                     186
Dramas, International Movies, Romantic Movies           180
Name: count, dtype: int64
```

```
# Display the counts

print("Counts Of 'type':\n", type_counts)
print("\nCounts for 'rating':\n", rating_counts)
print("\nTop 10 counts for 'country':\n", country_counts)
print("\nTop 10 counts for 'listed_in':\n", genre_counts)
```

```
Counts Of 'type':
 type
Movie      6131
TV Show    2676
Name: count, dtype: int64

Counts for 'rating':
 rating
TV-MA       3207
TV-14       2160
TV-PG        863
R            799
PG-13        490
TV-Y7        334
TV-Y         307
PG           287
TV-G         220
NR            80
G             41
TV-Y7-FV       6
NC-17          3
UR             3
74 min         1
84 min         1
66 min         1
Name: count, dtype: int64

Top 10 counts for 'country':
 country
United States     2818
```

```
India                 972
United Kingdom        419
Japan                 245
South Korea           199
Canada                181
Spain                 145
France                124
Mexico                110
Egypt                 106
Name: count, dtype: int64

Top 10 counts for 'listed_in':
 listed_in
Dramas, International Movies                            362
Documentaries                                           359
Stand-Up Comedy                                         334
Comedies, Dramas, International Movies                  274
Dramas, Independent Movies, International Movies        252
Kids' TV                                                220
Children & Family Movies                                215
Children & Family Movies, Comedies                      201
Documentaries, International Movies                     186
Dramas, International Movies, Romantic Movies           180
Name: count, dtype: int64
```

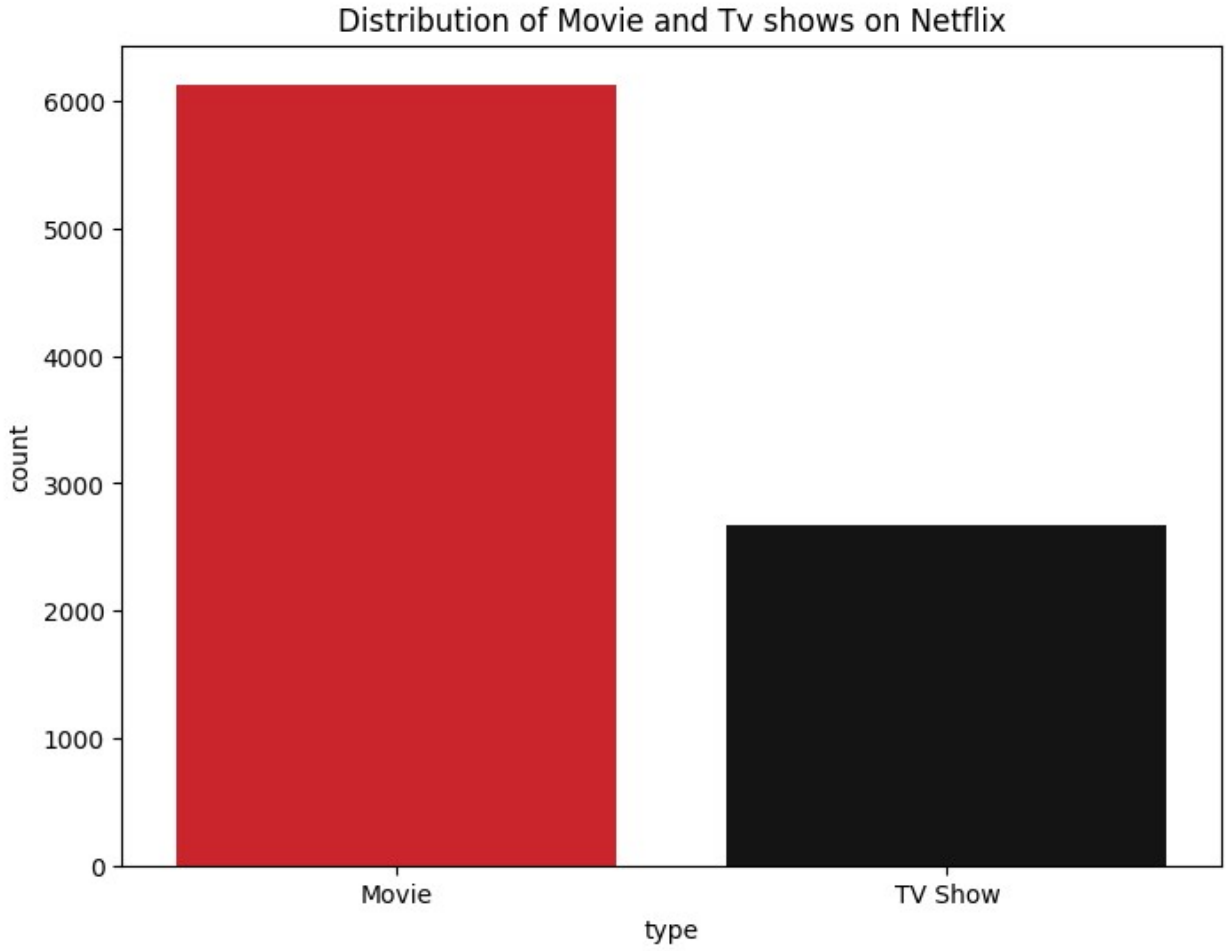
From the analysis, we can see that Netflix has a significantly higher number of movies compared to TV shows. Additionally, 'TV-MA' and 'TV-14' ratings are the most common, indicating Netflix caters to a mature audience.

##Graphical analysis : Value counts for categorical variables

```
# Count plot for 'type' (Movie/TV Show)

plt.figure(figsize=(8,6))
sns.countplot(data=netflix_data, x='type', palette = ['#E50914',
'#141414', '#B3B3B3'])
plt.title('Distribution of Movie and Tv shows on Netflix')
plt.show()
```

```
<ipython-input-83-777fcf32a68d>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(data=netflix_data, x='type', palette = ['#E50914',
'#141414', '#B3B3B3'])
<ipython-input-83-777fcf32a68d>:4: UserWarning: The palette list has
more values (3) than needed (2), which may not be intended.
```

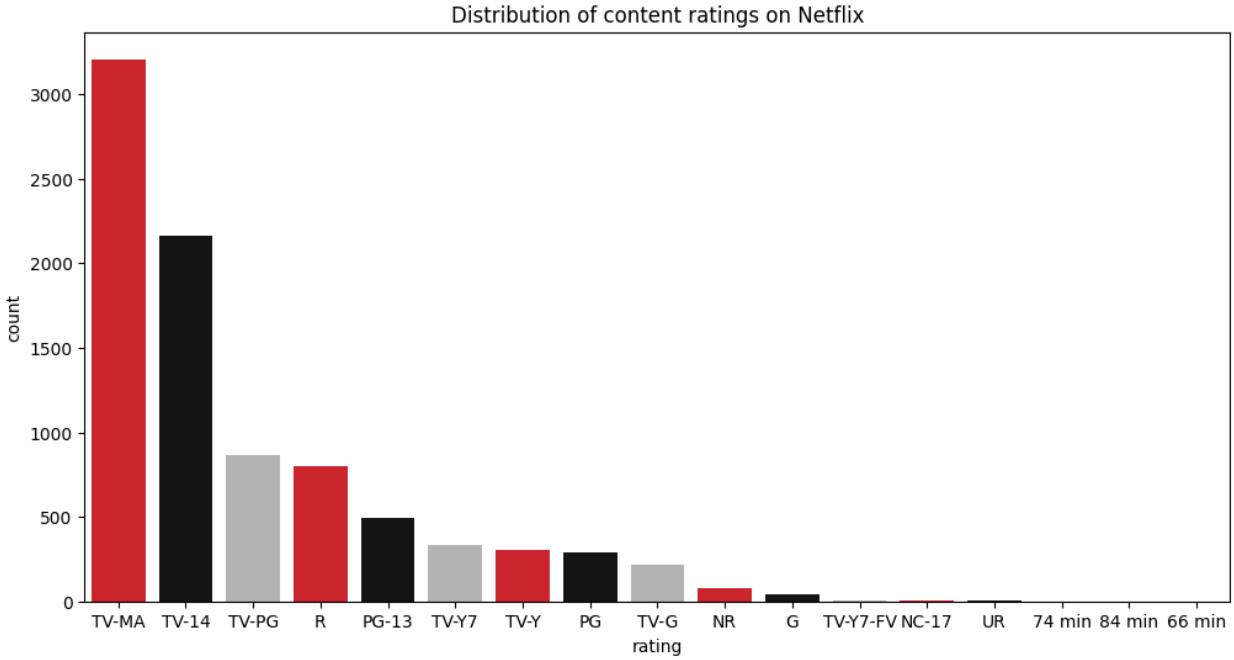
```
  sns.countplot(data=netflix_data, x='type', palette = ['#E50914',
'#141414', '#B3B3B3'])
```




```
# Count plot for 'rating'

plt.figure(figsize=(12,6))
sns.countplot(data=netflix_data, x= 'rating', palette =['#E50914',
'#141414', '#B3B3B3'],
order = netflix_data['rating'].value_counts().index)
plt.title('Distribution of content ratings on Netflix')
plt.show()
```

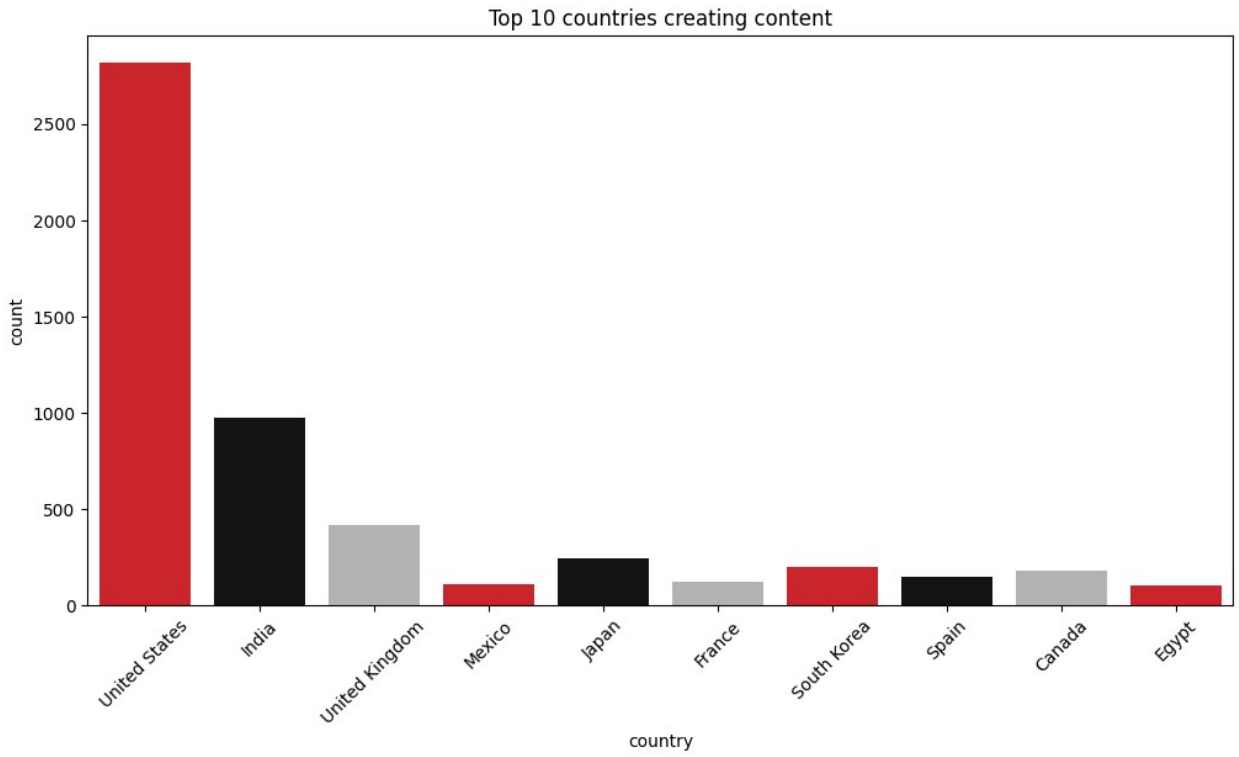
```
<ipython-input-84-169fcf5d25e7>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(data=netflix_data, x= 'rating', palette =['#E50914',
'#141414', '#B3B3B3'],
```

```
<ipython-input-84-169fcf5d25e7>:4: UserWarning:
The palette list has fewer values (3) than needed (17) and will cycle,
which may produce an uninterpretable plot.
  sns.countplot(data=netflix_data, x= 'rating', palette =['#E50914',
'#141414', '#B3B3B3'],
```




```
# Count plot for 'country' (Top 10 countries)

top_10_countries =
netflix_data['country'].value_counts().nlargest(10).index
plt.figure(figsize=(12,6))
sns.countplot(
    data=netflix_data[netflix_data['country'].isin(top_10_countries)],
    x='country',
    palette=['#E50914', '#141414', '#B3B3B3'])
plt.title('Top 10 countries creating content')
plt.xticks(rotation = 45)
plt.show()

<ipython-input-85-bc276d7da531>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(
<ipython-input-85-bc276d7da531>:5: UserWarning:
The palette list has fewer values (3) than needed (10) and will cycle,
```

```
which may produce an uninterpretable plot.
  sns.countplot(
```



Comparison of tv shows vs. movies

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

##Non-Graphical Analysis: Top 10 Countries Producing Movies:

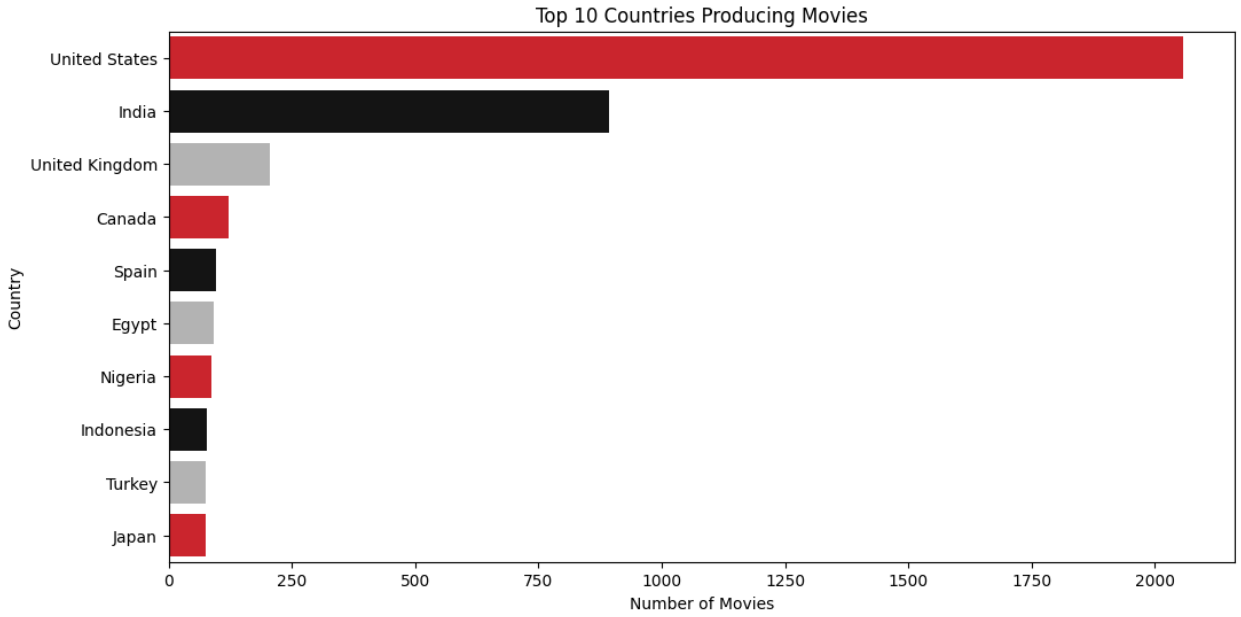
```
#Filter the data to include only 'Movies'

movies_data = netflix_data[netflix_data['type']== 'Movie']

# Group by 'country' and count unique 'title for each country

movies_by_country = movies_data.groupby('country')
['title'].count().sort_values(ascending = False).head(10)
print("Top 10 Countries Producting Movies:\n", movies_by_country)

Top 10 Countries Producting Movies:
 country
United States     2058
India              893
```

```
United Kingdom     206
Canada             122
Spain               97
Egypt               92
Nigeria             86
Indonesia           77
Turkey              76
Japan               76
Name: title, dtype: int64
```

##Graphical Analysis: Bar Plot of Top 10 Countries Producing Movies:

```
plt.figure(figsize=(12,6))
sns.barplot(x=movies_by_country.values, y= movies_by_country.index,
palette=['#E50914', '#141414', '#B3B3B3'])
plt.title('Top 10 Countries Producing Movies')
plt.xlabel('Number of Movies')
plt.ylabel('Country')
plt.show()
```

```
<ipython-input-88-f781b83eb6ec>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=movies_by_country.values, y= movies_by_country.index,
palette=['#E50914', '#141414', '#B3B3B3'])
<ipython-input-88-f781b83eb6ec>:2: UserWarning:
The palette list has fewer values (3) than needed (10) and will cycle,
which may produce an uninterpretable plot.
  sns.barplot(x=movies_by_country.values, y= movies_by_country.index,
palette=['#E50914', '#141414', '#B3B3B3'])
```



The analysis shows that the United States dominates movie production on Netflix, followed by India and the United Kingdom. This trend reflects Netflix's focus on content from these major film industries.

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

####Non Graphical analysis

```
#Filter the data to include only 'TV Shows'

tv_shows_data = netflix_data[netflix_data['type']=='TV Show']

# Group by 'country' and count unique 'title' for each country

tv_shows_by_country = tv_shows_data.groupby('country')
['title'].count().sort_values(ascending=False).head(10)

print("Top 10 countries producing TV shows:\n", tv_shows_by_country)
```

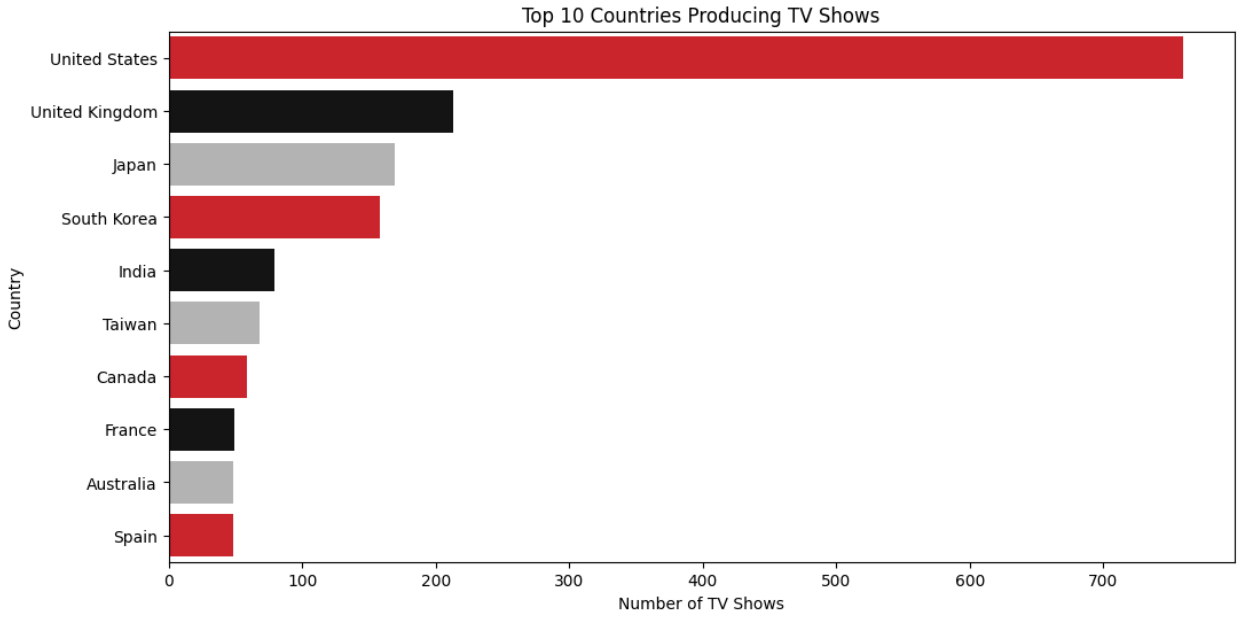
```
Top 10 countries producing TV shows:
 country
United States     760
United Kingdom    213
Japan             169
South Korea       158
India              79
Taiwan             68
```

```
Canada              59
France              49
Australia           48
Spain               48
Name: title, dtype: int64
```

####Graphical analysis

```
plt.figure(figsize=(12,6))
sns.barplot(x=tv_shows_by_country.values, y=tv_shows_by_country.index,
palette=['#E50914', '#141414', '#B3B3B3'])
plt.title('Top 10 Countries Producing TV Shows')
plt.xlabel('Number of TV Shows')
plt.ylabel('Country')
plt.show()
```

```
<ipython-input-90-6643b5c94c98>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=tv_shows_by_country.values,
y=tv_shows_by_country.index, palette=['#E50914', '#141414',
'#B3B3B3'])
<ipython-input-90-6643b5c94c98>:2: UserWarning:
The palette list has fewer values (3) than needed (10) and will cycle,
which may produce an uninterpretable plot.
  sns.barplot(x=tv_shows_by_country.values,
y=tv_shows_by_country.index, palette=['#E50914', '#141414',
'#B3B3B3'])
```



The United States leads in producing TV shows on Netflix with, followed by the United Kingdom and Japan. This indicates Netflix's strong focus on the US and UK markets for both movies and TV shows, while also tapping into the Asian content market with Japan and South Korea.

#What is the best time to launch a TV show

##Analysis for TV Shows:

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

```
# Converting 'date_added' to datetime
netflix_data['date_added'] =
pd.to_datetime(netflix_data['date_added'], errors='coerce')

# Create a new column 'week_added' to extract the week of the year
netflix_data['week_added'] =
netflix_data['date_added'].dt.isocalendar().week
```

```
<ipython-input-91-5adc78c33120>:2: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  netflix_data['date_added'] =
pd.to_datetime(netflix_data['date_added'], errors='coerce')
```

```
tv_shows_data = netflix_data[netflix_data['type']=='TV Show']

tv_show_by_week = tv_shows_data.groupby('week_added')
['title'].count().sort_values(ascending = False)
```

```
# Display the week with the highest number of TV show releases
print("Best week to release TV shows:\n", tv_show_by_week.head(1))
```

```
Best week to release TV shows:
 week_added
27    86
Name: title, dtype: int64
```

```
# Display the top 10 weeks for releasing TV shows
print("\nTop 10 Weeks to release TV Shoes:\n",
tv_show_by_week.head(10))
```

```
Top 10 Weeks to release TV Shoes:
 week_added
27    86
31    83
13    76
44    75
24    75
35    74
5     73
26    73
40    72
50    70
Name: title, dtype: int64
```

#What is the best time to launch a Movie

```
# Filter the data to include only 'Movies'

movies_data = netflix_data[netflix_data['type']=='Movie']

# Group by 'week_added' and count the number of movies added in each
week
movies_by_week = movies_data.groupby('week_added')
['title'].count().sort_values(ascending = False)

# Display the week with the highest number of movie releases

print("Best week to release movies:\n", movies_by_week.head(1))

# Display the top 10 weeks for releasing movies

print("Top 10 weeks to Release Movies:\n", movies_by_week.head(10))
```

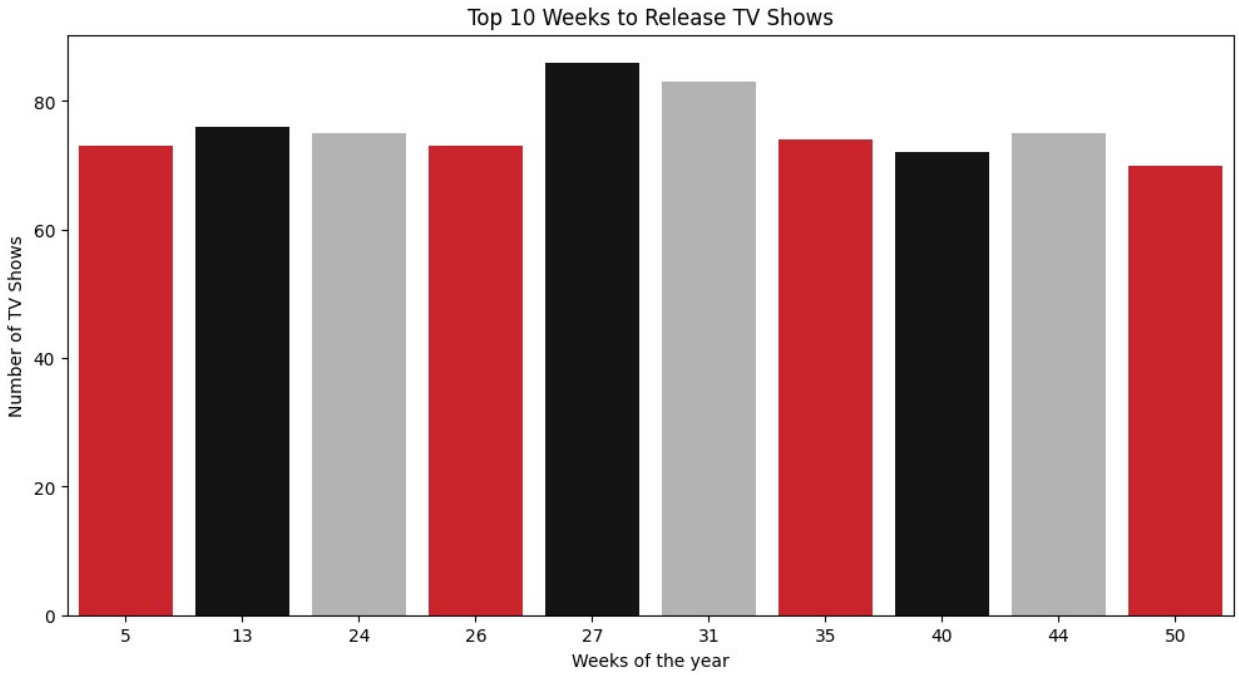
```
Best week to release movies:
 week_added
1    316
Name: title, dtype: int64
```

```
Top 10 weeks to Release Movies:
 week_added
1     316
44    243
40    215
9     207
26    195
35    189
31    185
13    174
18    173
27    154
Name: title, dtype: int64
```

##Graphical Representation:

```
# the best weeks for TV Shows

plt.figure(figsize=(12,6))
sns.barplot(x=tv_show_by_week.head(10).index, y =
tv_show_by_week.head(10).values, palette=['#E50914', '#141414',
'#B3B3B3'])
plt.title('Top 10 Weeks to Release TV Shows')
plt.xlabel('Weeks of the year')
plt.ylabel('Number of TV Shows')
plt.show()
```

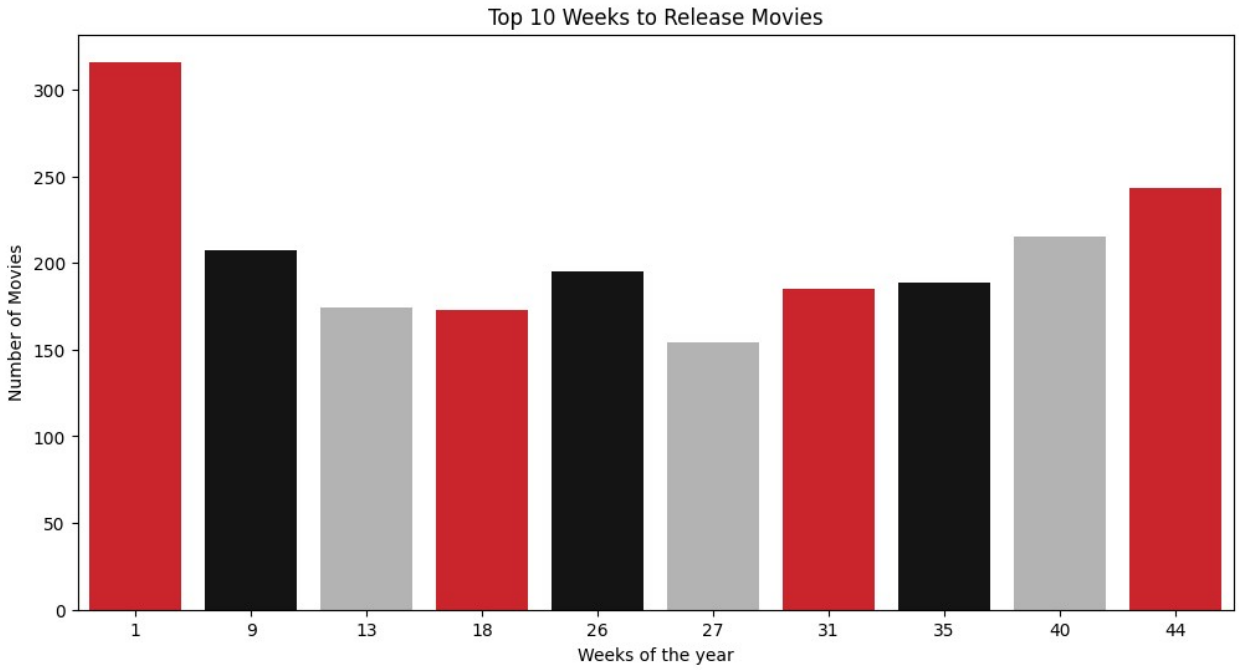
```
<ipython-input-95-89944328c731>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=tv_show_by_week.head(10).index, y =
tv_show_by_week.head(10).values, palette=['#E50914', '#141414',
'#B3B3B3'])
<ipython-input-95-89944328c731>:4: UserWarning:
The palette list has fewer values (3) than needed (10) and will cycle,
which may produce an uninterpretable plot.
  sns.barplot(x=tv_show_by_week.head(10).index, y =
tv_show_by_week.head(10).values, palette=['#E50914', '#141414',
'#B3B3B3'])
```



##Analysis for Movies

```
plt.figure(figsize=(12,6))
sns.barplot(x=movies_by_week.head(10).index, y =
movies_by_week.head(10).values, palette=['#E50914', '#141414',
'#B3B3B3'])
plt.title('Top 10 Weeks to Release Movies')
plt.xlabel('Weeks of the year')
plt.ylabel('Number of Movies')
plt.show()
```

```
<ipython-input-96-7bfbdcf345ad>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=movies_by_week.head(10).index, y =
movies_by_week.head(10).values, palette=['#E50914', '#141414',
'#B3B3B3'])
<ipython-input-96-7bfbdcf345ad>:2: UserWarning:
The palette list has fewer values (3) than needed (10) and will cycle,
which may produce an uninterpretable plot.
  sns.barplot(x=movies_by_week.head(10).index, y =
movies_by_week.head(10).values, palette=['#E50914', '#141414',
'#B3B3B3'])
```



Week 27 and 31 appears to be the best time to release TV shows and week 1 appears to be the best time to release a movie.

Likely corresponding to the holiday season when viewership spikes. Week 5,13, 40 are also high-performing

Find which is the best month to release the Tv-show or the movie.

```
#Strip whitespace from 'date_added' and convert to datetime

# Convert all values in 'date_added' to strings first to safely
apply .str.strip()
netflix_data['date_added'] =
netflix_data['date_added'].astype(str).str.strip()

# Then, convert to datetime with error handling for invalid formats
netflix_data['date_added'] =
pd.to_datetime(netflix_data['date_added'], errors='coerce')


# Create a new column 'month_added' to extract the month from
'date_added'

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

# Separate the data for TV shows and movies

tv_shows_data = netflix_data[netflix_data['type'] == 'TV Show']
movies_data = netflix_data[netflix_data['type'] == 'Movie']
```

```
# Analysis for TV Shows

tv_show_by_month = tv_shows_data.groupby('month_added')
['title'].count().sort_values(ascending = False)

# Analysis for Movies

movies_by_month = movies_data.groupby('month_added')
['title'].count().sort_values(ascending = False)

#print the results

print("Best months to release TV Shows")
print(tv_show_by_month)

print("\nBest months to release Movies")
print(movies_by_month)
```

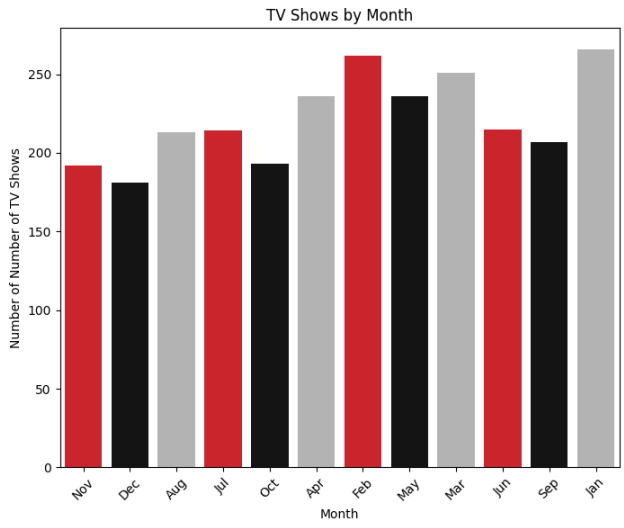
```
Best months to release TV Shows
month_added
12.0    266
7.0     262
9.0     251
6.0     236
8.0     236
10.0    215
4.0     214
3.0     213
11.0    207
5.0     193
1.0     192
2.0     181
Name: title, dtype: int64

Best months to release Movies
month_added
7.0     565
4.0     550
12.0    547
1.0     546
10.0    545
3.0     529
8.0     519
9.0     519
11.0    498
6.0     492
5.0     439
2.0     382
Name: title, dtype: int64
```

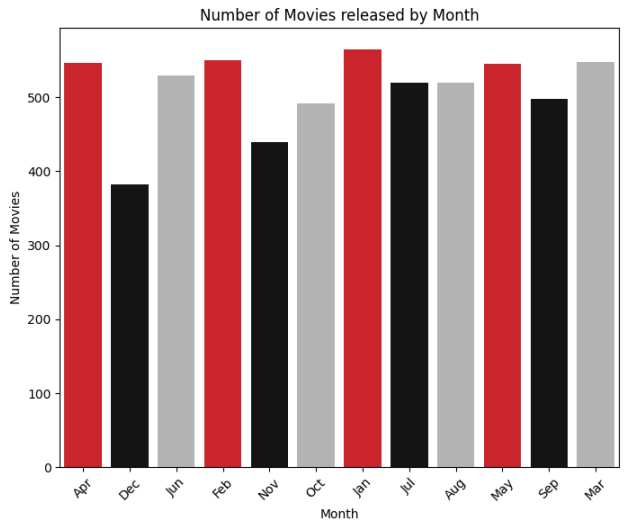
###Graphical Representation

```
netflix_data['date_added'] =
netflix_data['date_added'].astype(str).str.strip()


netflix_data['date_added'] =
pd.to_datetime(netflix_data['date_added'], errors='coerce')


# Create a new column 'month_added' to extract the month from
'date_added'

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

# Separate the data for TV shows and movies
tv_shows_data = netflix_data[netflix_data['type'] == 'TV Show']
movies_data = netflix_data[netflix_data['type'] == 'Movie']

#Analysis for TV Shows

tv_show_by_month = tv_shows_data.groupby('month_added')
['title'].count().sort_values(ascending = False)

# Analysis for Movie

movies_by_month = movies_data.groupby('month_added')
['title'].count().sort_values(ascending = False)

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

#Bar plot for TV Shows

plt.subplot(1,2,1) # (row, columns, panel number)
sns.barplot(x = tv_show_by_month.index, y = tv_show_by_month.values,
palette=['#E50914', '#141414', '#B3B3B3'])
plt.title('TV Shows by Month')
plt.xlabel('Month')
plt.ylabel('Number of Number of TV Shows')
plt.xticks(tv_show_by_month.index -1, labels = ['Jan', 'Feb', 'Mar',
'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],
rotation = 45)


#Bar plot for Movies
plt.subplot(1,2,2)
sns.barplot(x=movies_by_month.index, y=movies_by_month.values,
palette=['#E50914', '#141414', '#B3B3B3'])
plt.title('Number of Movies released by Month')
plt.xlabel('Month')
```

```
plt.ylabel('Number of Movies')
plt.xticks(ticks= movies_by_month.index -1, labels = ['Jan', 'Feb',
'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],
rotation = 45)

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

```
<ipython-input-101-a5e122b7d06f>:14: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x = tv_show_by_month.index, y = tv_show_by_month.values,
palette=['#E50914', '#141414', '#B3B3B3'])
<ipython-input-101-a5e122b7d06f>:14: UserWarning:
The palette list has fewer values (3) than needed (12) and will cycle,
which may produce an uninterpretable plot.
  sns.barplot(x = tv_show_by_month.index, y = tv_show_by_month.values,
palette=['#E50914', '#141414', '#B3B3B3'])
<ipython-input-101-a5e122b7d06f>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=movies_by_month.index, y=movies_by_month.values,
palette=['#E50914', '#141414', '#B3B3B3'])
<ipython-input-101-a5e122b7d06f>:23: UserWarning:
The palette list has fewer values (3) than needed (12) and will cycle,
which may produce an uninterpretable plot.
  sns.barplot(x=movies_by_month.index, y=movies_by_month.values,
palette=['#E50914', '#141414', '#B3B3B3'])
```





The analysis reveals that December is the best month for releasing both TV shows and movies. The trends observed indicate that releases during the holiday season attract higher viewership and engagement.

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

```
# Unnest the 'cast' column by creating multiple rows for actors

actors_data = netflix_data.assign(cast =
netflix_data['cast'].str.split(",")).explode('cast')

# Remove any leading/trailing whitespace from actor names

actors_data['cast'] = actors_data['cast'].str.strip()


# Count unique titles for each actor

top_actors = actors_data.groupby("cast")
['title'].nunique().sort_values(ascending= False).head(10)

print("Top 10 Actors who have appeared in the most number of Movies or
TV shows:")
print(top_actors)
```

```
Top 10 Actors who have appeared in the most number of Movies or TV
shows:
cast
Anupam Kher          43
Shah Rukh Khan       35
Julie Tejwani        33
Naseeruddin Shah     32
Takahiro Sakurai     32
Rupa Bhimani         31
Akshay Kumar         30
Om Puri              30
Yuki Kaji            29
Amitabh Bachchan     28
Name: title, dtype: int64
```

This analysis identifies the top 10 actors who have appeared in the most movies and TV shows on Netflix. By counting the unique titles associated with each actor, we can understand which actors are most prominent in Netflix's catalog, potentially guiding casting decisions for future productions.

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

```
# Count unique titles for each director


Top_directors = netflix_data.groupby('director')
```

```
['title'].nunique().sort_values(ascending = False).head(10)

print("\nTop 10 Directors who have directed most Movies/TVshows:")
print(Top_directors)
```

```
Top 10 Directors who have directed most Movies/TVshows:
director
Rajiv Chilaka             19
Raúl Campos, Jan Suter    18
Marcus Raboy              16
Suhas Kadav               16
Jay Karas                 14
Cathy Garcia-Molina       13
Jay Chapman               12
Martin Scorsese           12
Youssef Chahine           12
Steven Spielberg          11
Name: title, dtype: int64
```

This analysis highlights the top 10 directors who have directed the most movies and TV shows available on Netflix. By examining the unique titles attributed to each director, we gain insights into directorial trends and can identify key figures whose work could enhance Netflix's content strategy.

#Which genre movies are more popular or produced more

```
import pandas as pd
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Combine all genres into a single string
all_genres = ' '.join(netflix_data['listed_in'].dropna().astype(str))

# Create a word cloud
wordcloud = WordCloud(
    width=800,
    height=400,
    background_color='black',  # Netflix-inspired background color
    colormap='Reds',  # Netflix-themed color palette
    max_words=200  # Limit the number of words to keep it neat
).generate(all_genres)

# Plot the word cloud
plt.figure(figsize=(12, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')  # Turn off axes for a clean look
plt.title(
    "Popular Netflix Genres",
    fontsize=30, color='#E50914', weight='bold', pad=20
```

```
)
plt.gca().patch.set_facecolor('#141414')  # Add Netflix black
background for the plot
plt.tight_layout(pad=2)
plt.show()
```

The word cloud illustrates the frequency of various genres produced on Netflix. Larger font sizes indicate more frequently produced genres, while smaller sizes represent those that are less common. This visualization helps identify popular genres, guiding Netflix in making informed decisions about future content production and potential areas for growth.

Find After how many days the movie will be added to Netflix after the release of the movie

```
netflix_data['date_added'] =
pd.to_datetime(netflix_data['date_added'], errors = 'coerce')

#creating a new column 'release_date' by combining 'release_year' with
a default date


netflix_data['release_date'] =
pd.to_datetime(netflix_data['release_year'].astype(str) + '-01-01')

# Calculate the difference in days between 'date_added' and
```

```
'release_date'

netflix_data['days_to_add'] = (netflix_data['date_added']-
netflix_data['release_date']).dt.days

# Get the mode of the difference

mode_days_to_add = netflix_data['days_to_add'].mode()[0]

print(f"The mode of days taken to add a movie to Netflix after its
release: {mode_days_to_add}")
```

```
The mode of days taken to add a movie to Netflix after its release:
334.0
```

The analysis shows that movies are typically added to Netflix 334 days after their release. This insight can guide Netflix in optimizing its content acquisition strategy and managing viewer expectations, ultimately enhancing engagement and competitiveness in the streaming market.

#Insights and Recommendations

Popular Genres

###Insight: Certain genres like "Drama," "Comedy," or "Action" are most watched on Netflix.

###Recommendation: Focus on adding more shows and movies in these genres since they attract the most viewers. You can also test introducing lesser-known genres to explore new audience interests.

##Best Time to Release Content

###Insight: Specific months and weeks see more releases, indicating high audience activity.

###Recommendation: Plan big releases during popular times (like weekends or holidays) to get maximum views. Release new shows or movies when people are most active.

##Top Directors and Actors

###Insight: Certain directors and actors appear in the most successful content on Netflix.

###Recommendation: Work with these directors and actors for future projects to create more hits and attract loyal fans.

##Country-Specific Content

###Insight: Countries like the US, India, and the UK produce a large number of Netflix shows and movies.

###Recommendation: Create more region-specific content to cater to local audiences. For example, add more Indian dramas for Indian viewers.

##Ratings and Audience Preferences

###Insight: Shows and movies with specific ratings (like PG-13 or R) perform better depending on the audience.

###Recommendation: Ensure there's a good mix of family-friendly and mature content so everyone can find something to watch.

##Timing for Adding Movies

###Insight: Movies are typically added to Netflix after a certain number of days from their theatrical release.

###Recommendation: Try adding trending movies sooner to keep the excitement alive and attract more subscribers.

##Word Cloud of Genres

###Insight: The most frequent words in genres highlight what people enjoy watching, like "Romantic Comedies" or "Thrillers."

###Recommendation: Use this data to improve Netflix's recommendations so users can quickly find content they love.

##Niche Genres

###Insight: Some genres are underrepresented but could attract smaller, dedicated audiences.

###Recommendation: Experiment with adding niche content to engage new viewers who don’t find their interests represented often.

##Local Content for Emerging Markets

###Insight: Growing markets show potential for more Netflix users.

###Recommendation: Create affordable subscription plans and more local-language content to attract new users in emerging countries.