# NETFLIX Case Study

### This case study is to analyse the Movies and TV shows on Netfilx.

### Insights from the data ananlysis and the recomendations could help Netflix in deciding which type of shows/movies to release on the OTT platform and how they can grow the business in different countries

## Importing libraries

In [119]:

```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import seaborn as sns
import warnings
warnings.simplefilter(action='ignore',category=FutureWarning)
```

#### Bar plot formate

In [120]:

```
def show_values_on_bars(axs, h_v="v", space=1):
    def _show_on_single_plot(ax):
        if h_v == "v":
            for p in ax.patches:
                _x = p.get_x() + p.get_width() / 2
                _y = p.get_y() + p.get_height()
                value = int(p.get_height())
                ax.text(_x, _y, value, ha="center")
        elif h_v == "h":
            for p in ax.patches:
                _x = p.get_x() + p.get_width() + float(space)
                _y = p.get_y() + p.get_height()
                value = int(p.get_width())
                ax.text(_x, _y, value, ha="left")

    if isinstance(axs, np.ndarray):
        for idx, ax in np.ndenumerate(axs):
            _show_on_single_plot(ax)
    else:
        _show_on_single_plot(axs)
```

#### Reading file and preparing window

In [121]:

```
# reading data and getting pd window ready
df = pd.read_csv('Nextflix.csv')
pd.set_option('display.width', 1000)
pd.set_option('display.max_columns', 120)
pd.set_option('display.max_row', 4000)
```

## Getting starter information

In [122]:

```python
# getting starter information
"""
* data shape
* data info
* data head
* data tail
* missing values
* data description
* check duplicates
* drop duplicates
* drop unnecessary columns
"""
cs_name = 'Netflix case study'
print(f'{cs_name}, shape is {df.shape}')
print()
print()
print(f"{cs_name} basic information")
print()
print(df.info())
print()
print()
print(f"{cs_name} Null value count percentage:")
print()
print(df.isnull().sum(axis=0)/len(df)*100)
print()
print()
print(f"{cs_name} Description:")
print()
print(df.describe())
print()
print()
print(f"{cs_name} Deep Description:")
print()
print(df.describe(include='object').T)
print()
print()
print(f"{cs_name} Duplicate values:")
print()
print()
print(f"{cs_name} Duplicate values:")
print()
print(df.loc[df.duplicated()])
```

```
Netflix case study, shape is (8807, 13)


Netflix case study basic information

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0    8807 non-null   int64
 1   show_id       8807 non-null   object
 2   type          8807 non-null   object
```

```
 3   title         8807 non-null   object
 4   director      6173 non-null   object
 5   cast          7982 non-null   object
 6   country       7976 non-null   object
 7   date_added    8797 non-null   object
 8   release_year  8807 non-null   int64
 9   rating        8803 non-null   object
 10  duration      8804 non-null   object
 11  listed_in     8807 non-null   object
 12  description   8807 non-null   object
dtypes: int64(2), object(11)
memory usage: 894.6+ KB
None


Netflix case study Null value count percentage:

Unnamed: 0       0.000000
show_id          0.000000
type             0.000000
title            0.000000
director        29.908028
cast             9.367549
country          9.435676
date_added       0.113546
release_year     0.000000
rating           0.045418
duration         0.034064
listed_in        0.000000
description      0.000000
dtype: float64


Netflix case study Description:

        Unnamed: 0  release_year
count  8807.000000   8807.000000
mean   4403.000000   2014.180198
std    2542.506244      8.819312
min       0.000000   1925.000000
25%    2201.500000   2013.000000
50%    4403.000000   2017.000000
75%    6604.500000   2019.000000
max    8806.000000   2021.000000


Netflix case study Deep Description:

            count unique                                                    t
op  freq
show_id      8807   8807
s1     1
type         8807      2                                                 Mov
ie  6131
title        8807   8807                                 Dick Johnson Is De
ad     1
director     6173   4528                                        Rajiv Chila
```

```
ka     19
cast          7982   7692                                David Attenborou
gh     19
country       7976    748                                     United Stat
es   2818
date_added    8797   1767                                   January 1, 20
20    109
rating        8803     17                                             TV-
MA   3207
duration      8804    220                                         1 Seas
on   1793
listed_in     8807    514                       Dramas, International Movi
es    362
description   8807   8775  Paranormal activity at a lush, abandoned prop
e...     4


Netflix case study Duplicate values:


Netflix case study Duplicate values:

Empty DataFrame
Columns: [Unnamed: 0, show_id, type, title, director, cast, country, date
_added, release_year, rating, duration, listed_in, description]
Index: []
```

# Cleaning the data

## Dropping unnecessary columns

In [123]:

```
# dropping unnecessary columns
df.drop("description", axis=1, inplace=True)
```

## Filling NA with Unknow

In [124]:

```
# fill na with unknow

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

## Datetime formate change in date columns

In [125]:

```
# datetime formate change in date columns
df['date_added'] = pd.DatetimeIndex(df['date_added'])

# released year to int type
a = df['release_year'].to_list()
a = np.array(a, dtype= int)
df['release_year'] = a

# making new column year added
df['year_added'] = pd.DatetimeIndex(df['date_added']).year

# making new column month added
df['month_added'] = pd.DatetimeIndex(df['date_added']).month
```

## Cleaning the cat column

In [126]:

```
# cleaning the cast column
bulk_cast = df['cast'].apply(lambda x: str(x).split(", ")).to_list()
new_df_cast = pd.DataFrame(bulk_cast, index=df['title'])
new_df_cast = new_df_cast.stack()
new_df_cast = pd.DataFrame(new_df_cast)
new_df_cast.reset_index(inplace=True)
new_df_cast = new_df_cast[['title', 0]]
new_df_cast.columns = ['title', 'actors']
```

## cleaning listed_in column

In [127]:

```
# cleaning list_in
bulk_list_in = df['listed_in'].apply(lambda x: str(x).split(", ")).to_list()
new_df_listed_in = pd.DataFrame(bulk_list_in, index=df['title'])
new_df_listed_in = new_df_listed_in.stack()
new_df_listed_in = pd.DataFrame(new_df_listed_in)
new_df_listed_in.reset_index(inplace=True)
new_df_listed_in = new_df_listed_in[['title', 0]]
new_df_listed_in.columns = ['title', 'genre']
```

## Cleaning directors

In [128]:

```
# cleaning directors
bulk_directors = df['director'].apply(lambda x: str(x).split(", ")).to_list()
new_df_directors = pd.DataFrame(bulk_directors, index=df['title'])
new_df_directors = new_df_directors.stack()
new_df_directors = pd.DataFrame(new_df_directors)
new_df_directors.reset_index(inplace=True)
new_df_directors = new_df_directors[['title', 0]]
new_df_directors.columns = ['title', 'directors']
```

## Cleaning Country

In [129]:

```
# cleaning country
bulk_countries = df['country'].apply(lambda x: str(x).split(", ")).to_list()
new_df_country = pd.DataFrame(bulk_countries, index=df['title'])
new_df_country = new_df_country.stack()
new_df_country = pd.DataFrame(new_df_country)
new_df_country.reset_index(inplace=True)
new_df_country = new_df_country[['title', 0]]
new_df_country.columns = ['title', 'countries']
```

## Joing the new DataFrames

In [130]:

```
#joining new DFs
merged_one = new_df_cast.merge(new_df_directors, on='title')
merged_two = new_df_country.merge(new_df_listed_in, on="title")
merged_df = merged_one.merge(merged_two, on='title')
df = df.merge(merged_df, on='title')
```

## Dropping old unnecessary columns

In [131]:

```
# dropping old unnecessary columns
drop_list = ['Unnamed: 0','show_id','director','listed_in']
df.drop(drop_list, axis=1, inplace = True)
df.drop('cast',axis=1,inplace=True)
df.drop('country',axis=1,inplace=True)
```

## Making movie duration and tv show duration colums with int type

In [132]:

```
# making movie duration and tv show duration colums with int type

df['movie_duration'] = df[df['type'] == "Movie"]['duration'].str.replace(' mi
movie_duration_in_list = df['movie_duration'].to_list()
movie_duration_in_list = np.array(movie_duration_in_list, dtype='float')
movie_duration_in_list

df['movie_duration'] = movie_duration_in_list


df['TV_Show_duration'] = df[df['type'] == "TV Show"]['duration'].str.replace(
df['TV_Show_duration'] = df['TV_Show_duration'].str.replace(' Season',"")
df['TV_Show_duration'].astype("float")
df.drop('duration', axis=1, inplace=True)
```

## New column content for

In [133]:

```
# new column content for
df["content_for"] = df["rating"].replace({"TV-MA":"Adults",
                      "TV-14" : "Teens",
                      "TV-PG" : "Older Kids",
                       "R":"Adults",
                       "PG-13":"Teens",
                       "TV-Y7":"Older Kids",
                       "TV-Y":"Kids",
                       "PG":"Older Kids",
                       "TV-G":"Kids",
                       "NR":"Adults",
                       "G":"Kids",
                       "TV-Y7-FV":"Older Kids",
                       "NC-17":"Adults",
                       "UR":"Adults"
                      })
```

In [ ]:

# Analysing the data

### Movie and TV shows distribution in the data

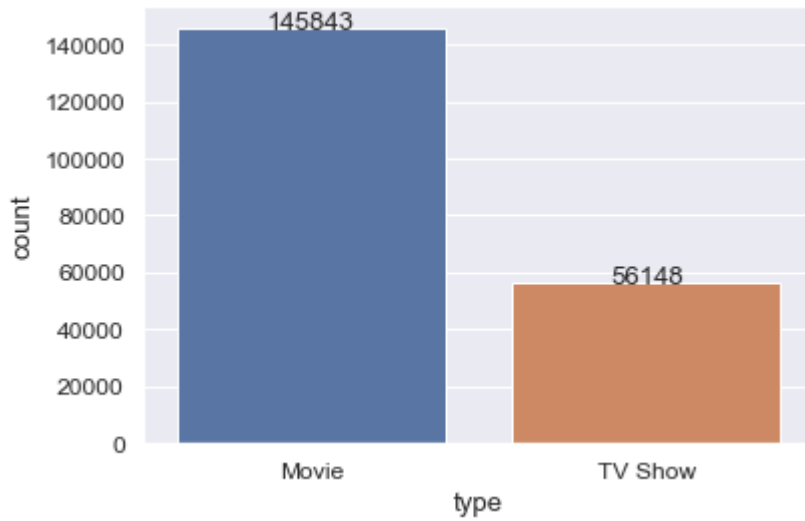
In [134]:

```
# movie and tv shows distribution in the data
type_distribution = df["type"].value_counts(normalize=True)*100
print(type_distribution)
```

```
Movie      72.202722
TV Show    27.797278
Name: type, dtype: float64
```

In [135]:

```
# movie and tv shows distribution in the data plot

sns.set(font_scale = 1.1)
types = sns.countplot(data = df , x = "type")

show_values_on_bars(types,h_v="v",space=1)
```




**Distribution on the bases of rating, and the countplot graph**

In [136]:

```
# content distribtion on rating bases
rating_base_distribution = df['rating'].value_counts()
print(rating_base_distribution)
print('_____________________________________________________________________
rating_base_distribution_percentage = (df['rating'].value_counts(normalize=Tr
print(rating_base_distribution_percentage)
print('_____________________________________________________________________
```

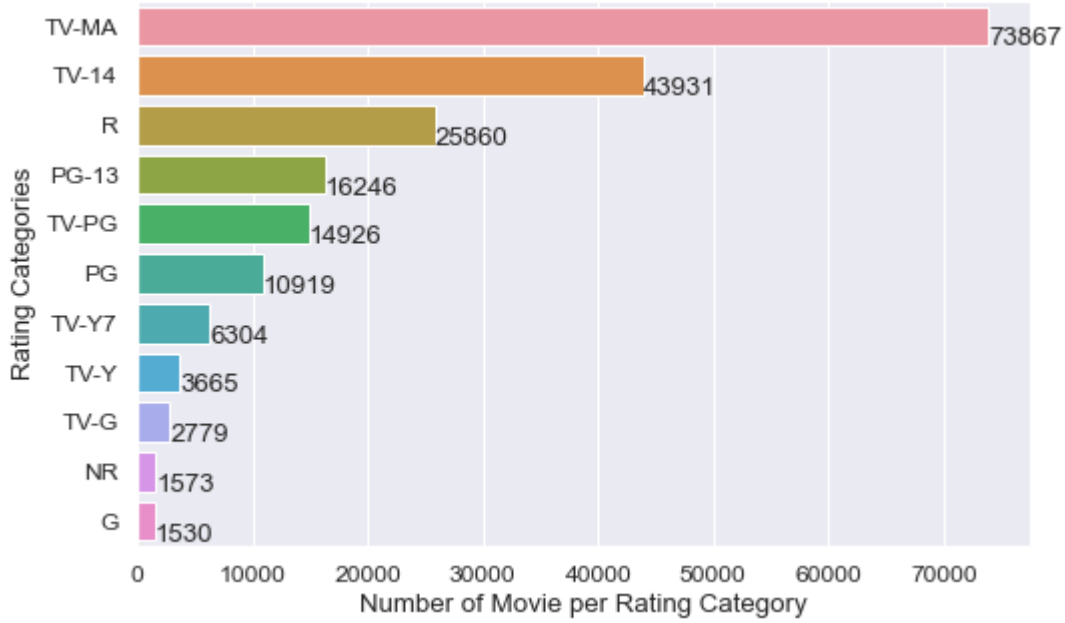
```
TV-MA       73867
TV-14       43931
R           25860
PG-13       16246
TV-PG       14926
PG          10919
TV-Y7        6304
TV-Y         3665
TV-G         2779
NR           1573
G            1530
NC-17         149
TV-Y7-FV       86
UR             86
74 min          1
84 min          1
66 min          1
Name: rating, dtype: int64
_____________________________________________________________________
_____________________________________
TV-MA       36.581585
TV-14       21.756205
R           12.806799
PG-13        8.045601
TV-PG        7.391890
PG           5.407480
TV-Y7        3.121967
TV-Y         1.815039
TV-G         1.376260
NR           0.779006
G            0.757711
NC-17        0.073790
TV-Y7-FV     0.042590
UR           0.042590
Name: rating, dtype: float64
_____________________________________________________________________
_____________________________________
```

In [137]:

```
#Observation:
#TV-MA and TV-14 covers more the 50% of the content rating
```

In [138]:

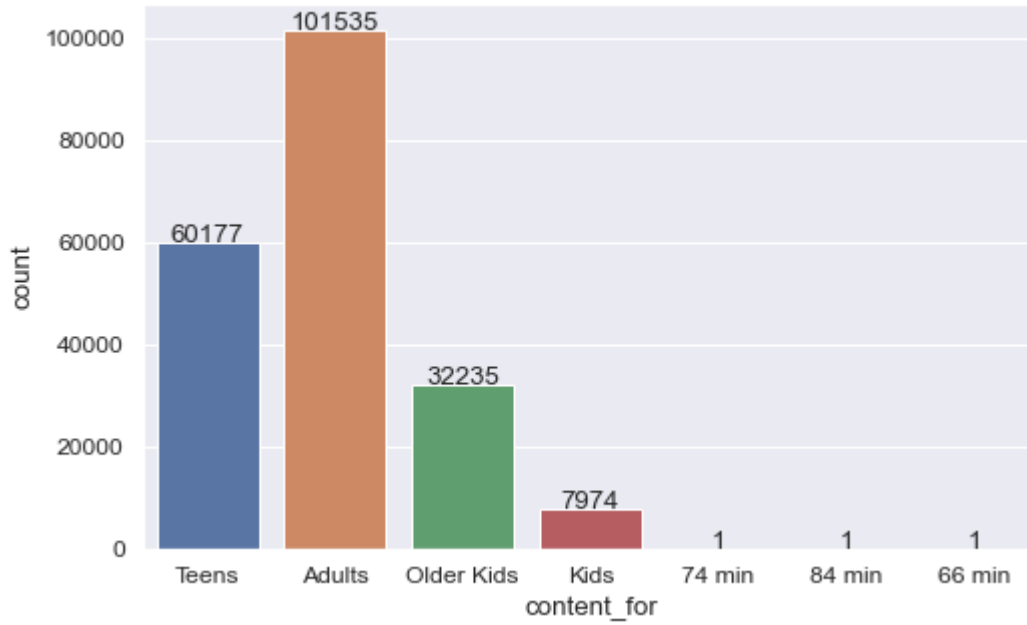
```
plt.figure(figsize=(8,5))
show_values_on_bars(sns.barplot( x = df["rating"].value_counts().head(11), y
plt.xlabel("Number of Movie per Rating Category")
plt.ylabel("Rating Categories")
plt.show()
```




**Distribution on the bases of age groups, and the countplot graph**

In [139]:

```
age_base_distribution = df["content_for"].value_counts()
age_base_distribution
age_base_distribution_in_percent = df["content_for"].value_counts(normalize=T
age_base_distribution_in_percent


plt.figure(figsize=(8,5))
show_values_on_bars(sns.countplot(x = df["content_for"]))
```




In [140]:

```
# observation:
# movies and tv shows on Netflix are mostly for adults and teen.
```

## Movies and Tv shows per year

In [141]:

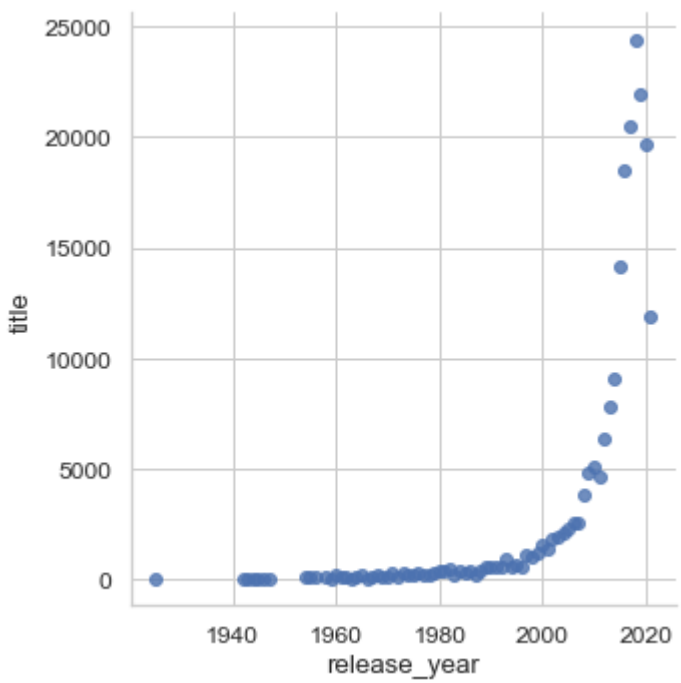
```
# movies_ans_tvshows_per_year

movies_ans_tvshows_per_year = df.groupby('release_year')['title'].count().sor
movies_ans_tvshows_per_year.head()
```

Out[141]:

| | release_year | title |
|---|---|---|
| **0** | 1925 | 1 |
| **1** | 1943 | 5 |
| **2** | 1942 | 6 |
| **3** | 1946 | 6 |
| **4** | 1947 | 8 |

In [142]:

```
# movies_ans_tvshows_per_year plot

sns.set_style('whitegrid')
sns.lmplot(x='release_year', y='title', data=movies_ans_tvshows_per_year,fit_
plt.show()
```




**Number of movies and tv shows per year for last 30 year**

In [143]:

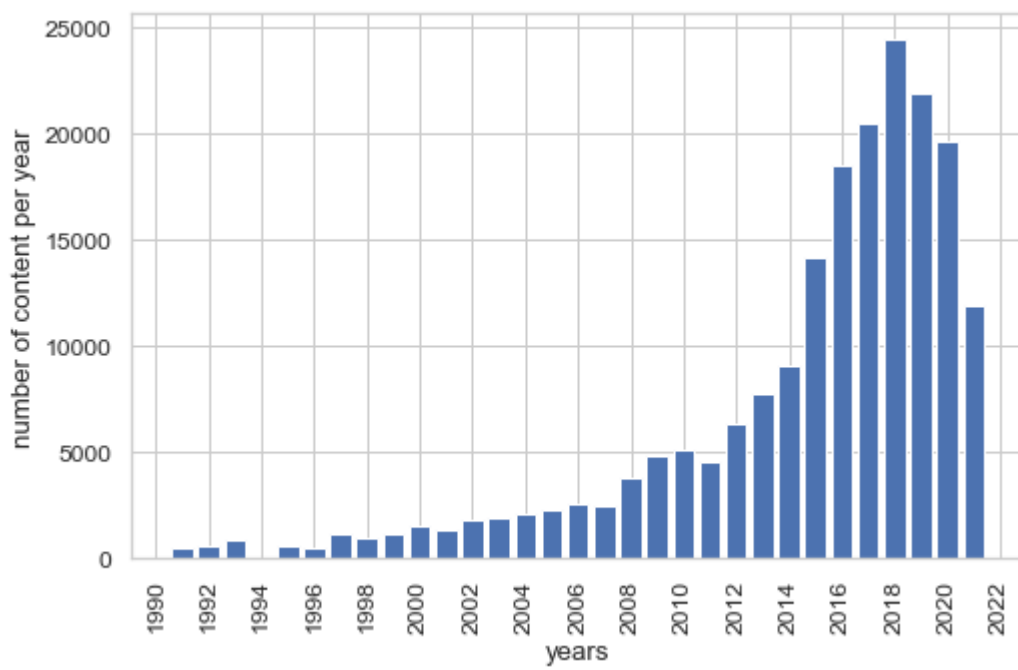
```
# number of movies and tv shows per year for last 30 year

in_last_30_year_movies_per_year = movies_ans_tvshows_per_year.tail(30)
in_last_30_year_movies_per_year.head()
```

Out[143]:

| | release_year | title |
|---|---|---|
| **44** | 1996 | 532 |
| **45** | 1991 | 533 |
| **46** | 1992 | 542 |
| **47** | 1995 | 582 |
| **48** | 1993 | 893 |

In [144]:

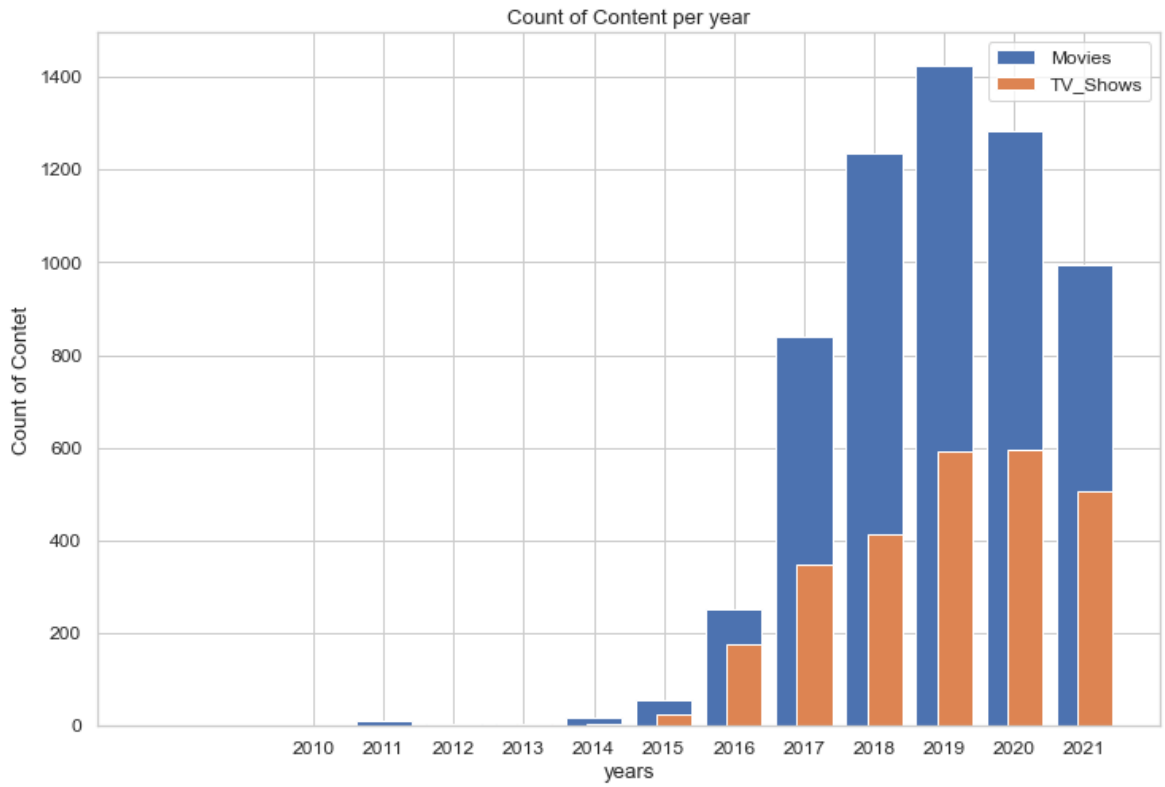
```
# number of movies and tv shows per year for last 30 year plot
plt.figure(figsize=(8,5))
plt.bar(in_last_30_year_movies_per_year['release_year'],in_last_30_year_movie
plt.xlabel('years')
plt.ylabel('number of content per year')
plt.xticks(np.arange(1990,2023,2), rotation=90)
plt.show()
```




## Analyzing TV shows and Movies data seperately

In [145]:

```
TV_shows = df.loc[df["type"]=="TV Show"]
Movies = df.loc[df["type"]=="Movie"]

plt.figure(figsize=(12,8))
plt.bar(Movies.groupby("year_added")["title"].nunique().index,Movies.groupby(
plt.bar((TV_shows.groupby("year_added")["title"].nunique().index)+0.15,TV_sho
 width=0.5 )
plt.xticks(np.arange(2010,2022),rotation = 0)
plt.title("Count of Content per year")
plt.xlabel("years")
plt.ylabel("Count of Contet")
plt.legend(["Movies","TV_Shows"])
plt.show()
```




## Best director, number of movies

In [146]:

```
# best director, number of movies
top_10_directors_global = df.groupby('directors')['title'].count().sort_value
top_10_directors_global = top_10_directors_global.drop(0,axis=0)
top_10_directors_global

top_director_global = top_10_directors_global.head(1)
top_director_global
```

Out[146]:

| | directors | title |
|---|---|---|
| **1** | Martin Scorsese | 419 |

## Best actor, based on number of movies

In [147]:

```
# best actor, number of movies
top_10_actors_global = df.groupby('actors')['title'].count().sort_values(asce
top_10_actors_global = top_10_actors_global.drop(0,axis=0)
top_10_actors_global
top_actors_global = top_10_actors_global.head(1)
top_actors_global
```

Out[147]:

| | actors | title |
|---|---|---|
| **1** | Liam Neeson | 161 |

## Number of Content per country, and its plot

In [148]:

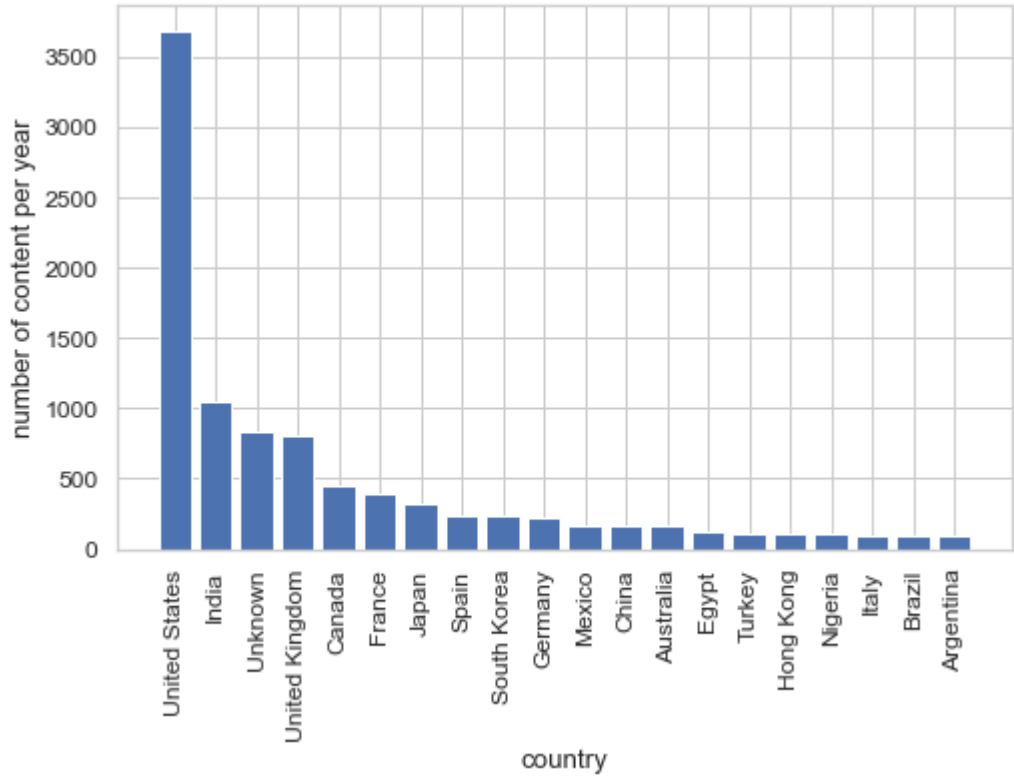
```
# number of movies per country

no_of_content_per_country = df.groupby("countries")["title"].nunique().sort_v
no_of_content_per_country.head()
```

Out[148]:

| | countries | title |
|---|---|---|
| **0** | United States | 3689 |
| **1** | India | 1046 |
| **2** | Unknown | 831 |
| **3** | United Kingdom | 804 |
| **4** | Canada | 445 |

In [149]:

```
# # number of movies per country plot

plt.figure(figsize=(8,5))
plt.bar(no_of_content_per_country['countries'],no_of_content_per_country['tit
plt.xlabel('country')
plt.ylabel('number of content per year')
plt.xticks(rotation=90)
plt.show()
```




## Top 20 countries with highest content

In [150]:

```
# top 20 countries
top_20_countries = no_of_content_per_country.head(20)
top_20_countries.head()
```

Out[150]:

| | countries | title |
|---|---|---|
| **0** | United States | 3689 |
| **1** | India | 1046 |
| **2** | Unknown | 831 |
| **3** | United Kingdom | 804 |
| **4** | Canada | 445 |

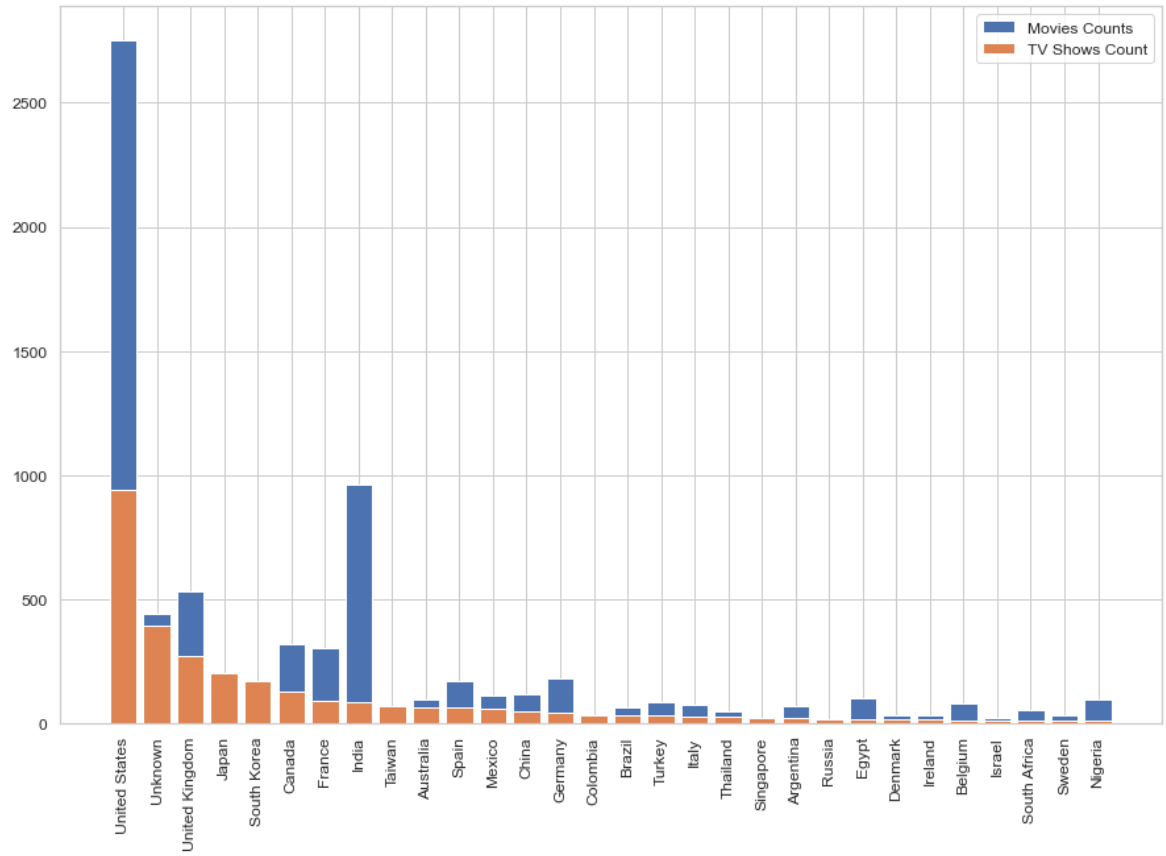
## Movies Counts - TV Shows Count as per Top Countires:

In [151]:

```
tvcount = TV_shows.groupby("countries")["title"].nunique().sort_values(ascend

mcount = Movies.groupby("countries")["title"].nunique().sort_values(ascending

tvVSMovies_per_country = tvcount.merge(mcount,on="countries",how = "outer")

tvVSMovies_per_country.rename({"title_x":"TV Shows Count", "title_y":"Movies
```

In [152]:

```
plt.figure(figsize=(15,10))
plt.bar(tvVSMovies_per_country["countries"].head(30),tvVSMovies_per_country["
plt.bar(tvVSMovies_per_country["countries"].head(30),tvVSMovies_per_country["
plt.xticks(rotation = 90)
plt.legend(["Movies Counts","TV Shows Count"])
plt.show()
```




## Most popular genre

In [153]:

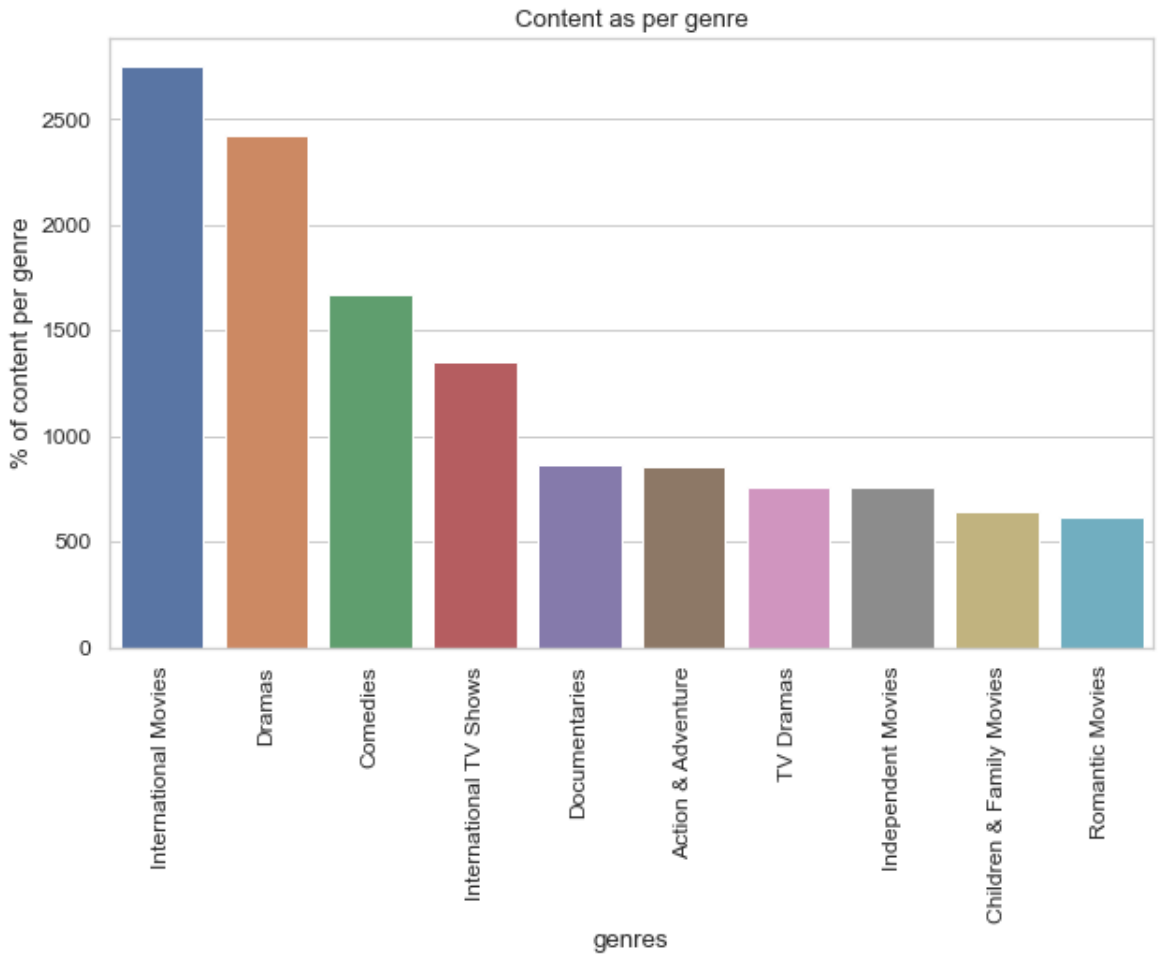
```
# most popular genre top 10
top_genre = df.groupby('genre')['title'].nunique().sort_values(ascending = Fa
top_genre
```

Out[153]:

| | genre | title |
|---|---|---|
| **0** | International Movies | 2752 |
| **1** | Dramas | 2427 |
| **2** | Comedies | 1674 |
| **3** | International TV Shows | 1351 |
| **4** | Documentaries | 869 |
| **5** | Action & Adventure | 859 |
| **6** | TV Dramas | 763 |
| **7** | Independent Movies | 756 |
| **8** | Children & Family Movies | 641 |
| **9** | Romantic Movies | 616 |

In [154]:

```
# most popular genre top 10 polt

plt.figure(figsize=(10,6))
sns.barplot(x = "genre" , y="title" , data = top_genre )
plt.title("Content as per genre")
plt.xlabel("genres")
plt.ylabel("% of content per genre")
plt.xticks(rotation = 90)
plt.show()
```




## Top 20 actors who have worked in more then one country

In [181]:

```
# top 20 actors who have worked in more the counties

top_20_actors_working_in_muliple_country = df.groupby('actors')['countries'].
top_20_actors_working_in_muliple_country = top_20_actors_working_in_muliple_c
top_20_actors_working_in_muliple_country.head(20)
```

Out[181]:

| | actors | countries |
|---|---|---|
| **1** | Alfred Molina | 14 |
| **2** | Paul Giamatti | 14 |
| **3** | Mads Mikkelsen | 13 |
| **4** | Ben Whishaw | 13 |
| **5** | James Franco | 13 |
| **6** | Eddie Marsan | 13 |
| **7** | Vincent Tong | 13 |
| **8** | Sylvester Stallone | 13 |
| **9** | John Cleese | 13 |
| **10** | Nicole Kidman | 13 |
| **11** | Stellan Skarsgård | 12 |
| **12** | Jeremy Irvine | 12 |
| **13** | Alicia Vikander | 12 |
| **14** | Brian Drummond | 12 |
| **15** | Christopher Lee | 12 |
| **16** | Natalie Dormer | 11 |
| **17** | Bruno Ganz | 11 |
| **18** | Guy Pearce | 11 |
| **19** | Léa Seydoux | 11 |
| **20** | Jim Broadbent | 11 |

## Top 20 directors who have worked in more then one country

In [180]:

```
# top 20 directors who have worked in more the counties

top_20_directors_working_in_muliple_country = df.groupby('directors')['countr
top_20_directors_working_in_muliple_country = top_20_directors_working_in_mul
top_20_directors_working_in_muliple_country.head(20)
```

Out[180]:

| | directors | countries |
|---|---|---|
| **1** | Matthew Salleh | 12 |
| **2** | Joshua Oppenheimer | 10 |
| **3** | Farhad Safinia | 8 |
| **4** | Martin Campbell | 8 |
| **5** | James Watkins | 7 |
| **6** | Aaron Woodley | 7 |
| **7** | Renny Harlin | 7 |
| **8** | Nora Twomey | 7 |
| **9** | Brian De Palma | 7 |
| **10** | Olivier Assayas | 7 |
| **11** | Ari Folman | 7 |
| **12** | Tom Hooper | 7 |
| **13** | Philippe Aractingi | 6 |
| **14** | Petra Costa | 6 |
| **15** | Alastair Fothergill | 6 |
| **16** | Juan Zapata | 6 |
| **17** | Martin Scorsese | 6 |
| **18** | Pablo Larraín | 6 |
| **19** | Liam O'Donnell | 6 |
| **20** | Paul Greengrass | 6 |

## Top actors in popular genres

In [157]:

```
# for each genre top actor


most_popular_genre = top_genre[0:11]


for genre in most_popular_genre['genre']:
    print()
    print(genre)
    print("Actor                number of movies")
    name = (df[df['genre']== genre].groupby('actors')['title'].nunique().sort
    print(name)
```

```
International Movies
Actor                number of movies
actors
Unknown        178
Anupam Kher     38
Name: title, dtype: int64

Dramas
Actor                number of movies
actors
Anupam Kher       28
Shah Rukh Khan    28
Name: title, dtype: int64

Comedies
Actor                number of movies
actors
Anupam Kher     20
Paresh Rawal    18
Name: title, dtype: int64

International TV Shows
Actor                number of movies
actors
Unknown             109
Takahiro Sakurai     22
Name: title, dtype: int64

Documentaries
Actor                number of movies
actors
Unknown        424
Samuel West     10
Name: title, dtype: int64

Action & Adventure
Actor                number of movies
actors
Bruce Willis        13
Amitabh Bachchan    12
Name: title, dtype: int64

TV Dramas
```

```
Actor               number of movies
actors
Tay Ping Hui    10
Jeanette Aw      7
Name: title, dtype: int64

Independent Movies
Actor               number of movies
actors
Naseeruddin Shah    10
Rajit Kapoor         8
Name: title, dtype: int64

Children & Family Movies
Actor               number of movies
actors
Unknown          33
Julie Tejwani    26
Name: title, dtype: int64

Romantic Movies
Actor               number of movies
actors
Akshay Kumar    8
Salman Khan     7
Name: title, dtype: int64
```

## Top Directors in popular genres

In [158]:

```
# for each genre top director

for genre in most_popular_genre['genre']:
    print()
    print(genre)
    print("Director                number of movies")
    name = (df[df['genre']== genre].groupby('directors')['title'].nunique().s
    print(name)
```

```
International Movies
Director                number of movies
directors
Unknown                 47
Cathy Garcia-Molina     13
Name: title, dtype: int64

Dramas
Director                number of movies
directors
Unknown             25
Youssef Chahine     12
Name: title, dtype: int64

Comedies
Director                number of movies
directors
Unknown          32
David Dhawan      9
Name: title, dtype: int64

International TV Shows
Director                number of movies
directors
Unknown                 1223
Alastair Fothergill        3
Name: title, dtype: int64

Documentaries
Director                number of movies
directors
Unknown        57
Vlad Yudin      6
Name: title, dtype: int64

Action & Adventure
Director                number of movies
directors
Don Michael Paul     9
Unknown              8
Name: title, dtype: int64

TV Dramas
Director                number of movies
directors
Unknown              702
Abhishek Chaubey       1
```

```
Name: title, dtype: int64

Independent Movies
Director                number of movies
directors
Noah Baumbach            5
Paul Thomas Anderson     5
Name: title, dtype: int64

Children & Family Movies
Director                number of movies
directors
Unknown           36
Rajiv Chilaka     22
Name: title, dtype: int64

Romantic Movies
Director                number of movies
directors
Unknown                 11
Cathy Garcia-Molina      8
Name: title, dtype: int64
```

**movie duration from shortest to longest**

In [159]:

```
movie_dur = df.groupby("movie_duration")['title'].unique().reset_index()
movie_dur.head()
```

Out[159]:

| | movie_duration | title |
|---|---|---|
| **0** | 3.0 | [Silent] |
| **1** | 5.0 | [Sol Levante] |
| **2** | 8.0 | [Cops and Robbers] |
| **3** | 9.0 | [Canvas] |
| **4** | 10.0 | [American Factory: A Conversation with the Oba... |

## Top 20 Versatile Actors

In [160]:

```
df.groupby("actors")["genre","title"].aggregate({
 "genre":"nunique",
 "title":"nunique"
}).sort_values(by = ["genre","title"],ascending=[False,False]).head(20)[1:]
```

Out[160]:

| actors | genre | title |
|---|---|---|
| Ron Perlman | 17 | 13 |
| Gary Cole | 16 | 11 |
| Kiernan Shipka | 16 | 9 |
| Glenn Close | 15 | 12 |
| Anupam Kher | 14 | 43 |
| Samuel L. Jackson | 14 | 24 |
| Rajesh Sharma | 14 | 18 |
| Antonio Banderas | 14 | 15 |
| Ben Kingsley | 14 | 15 |
| Keith David | 14 | 15 |
| Guy Pearce | 14 | 14 |
| Jim Broadbent | 14 | 13 |
| John Leguizamo | 14 | 12 |
| Carla Gugino | 14 | 11 |
| Jay Baruchel | 14 | 11 |
| Keegan-Michael Key | 14 | 11 |
| Mae Whitman | 14 | 9 |
| Rosamund Pike | 14 | 9 |
| Nicholas Hoult | 14 | 8 |

## Top 20 versatile Directors

In [161]:

```
df.groupby("directors")["genre","title"].aggregate({
 "genre":"nunique",
 "title":"nunique"
}).sort_values(by = ["genre","title"],ascending=[False,False]).head(20)[1:]
```

Out[161]:

| directors | genre | title |
|---|---|---|
| **Martin Scorsese** | 9 | 12 |
| **Anurag Kashyap** | 9 | 9 |
| **Priyadarshan** | 8 | 7 |
| **Abhishek Chaubey** | 8 | 5 |
| **Vishal Bhardwaj** | 8 | 5 |
| **Vikramaditya Motwane** | 8 | 4 |
| **Ifa Isfansyah** | 8 | 3 |
| **David Dhawan** | 7 | 9 |
| **Clint Eastwood** | 7 | 7 |
| **Ron Howard** | 7 | 7 |
| **Brad Anderson** | 7 | 4 |
| **Jeremy Saulnier** | 7 | 4 |
| **BB Sasore** | 7 | 3 |
| **Guillermo del Toro** | 7 | 3 |
| **Jalil Lespert** | 7 | 3 |
| **Julien Leclercq** | 7 | 3 |
| **Manolo Caro** | 7 | 3 |
| **Tim Burton** | 7 | 3 |
| **Youssef Chahine** | 6 | 12 |

## Shortest movie on Netflix (in min)

In [162]:

```
# Shortest movie on Netflix

shortest_movie = min(movie_dur['movie_duration'])
shortest_movie = movie_dur[movie_dur['movie_duration'] == shortest_movie]
shortest_movie
```

Out[162]:

| | movie_duration | title |
|---|---|---|
| **0** | 3.0 | [Silent] |

## longest movie on Netflix (in min)

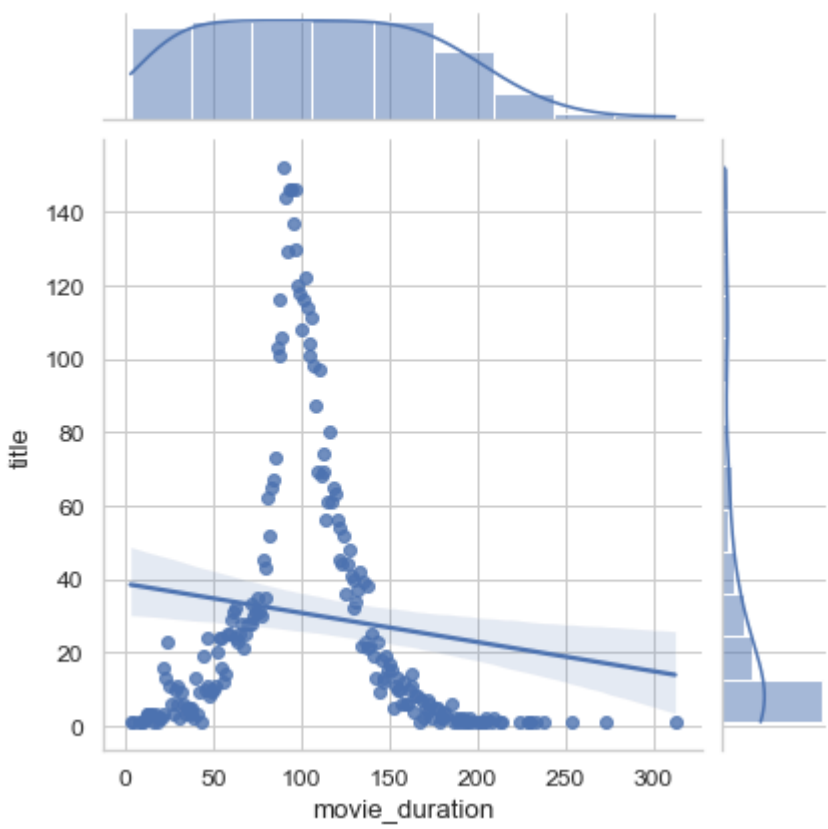
In [163]:
```
#longest movie on Netflix
longest_movie = max(movie_dur['movie_duration'])
longest_movie = movie_dur[movie_dur['movie_duration'] == longest_movie]
longest_movie
```

Out[163]:

| | movie_duration | title |
|---|---|---|
| **204** | 312.0 | [Black Mirror: Bandersnatch] |

## Movie duration density and graph

In [164]:
```
movie_duration_density =  df.groupby("movie_duration")['title'].nunique().res
movie_duration_density


sns.jointplot(x='movie_duration', y='title', data=movie_duration_density, kin
plt.show()
```




## Most popular movie duration

In [165]:

```
most_popular_movie_length = movie_duration_density[movie_duration_density['ti
most_popular_movie_length
```

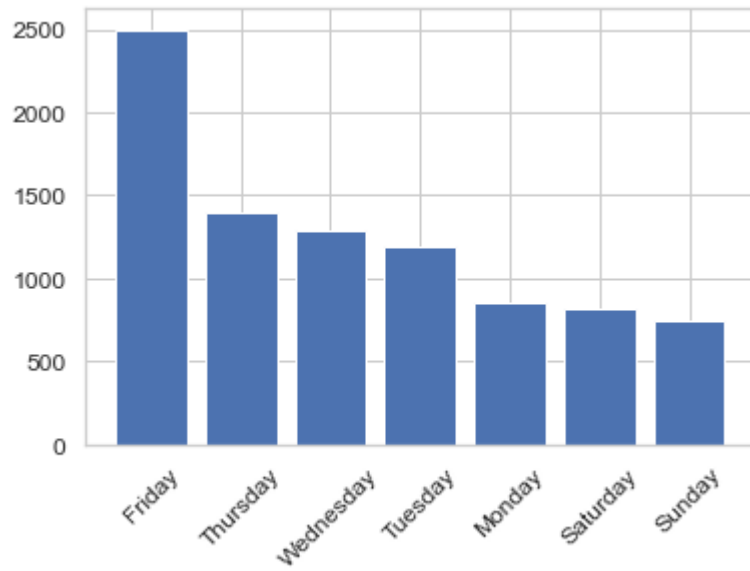
Out[165]:

| | movie_duration | title |
|---|---|---|
| **84** | 90.0 | 152 |

## Best time to launch a content

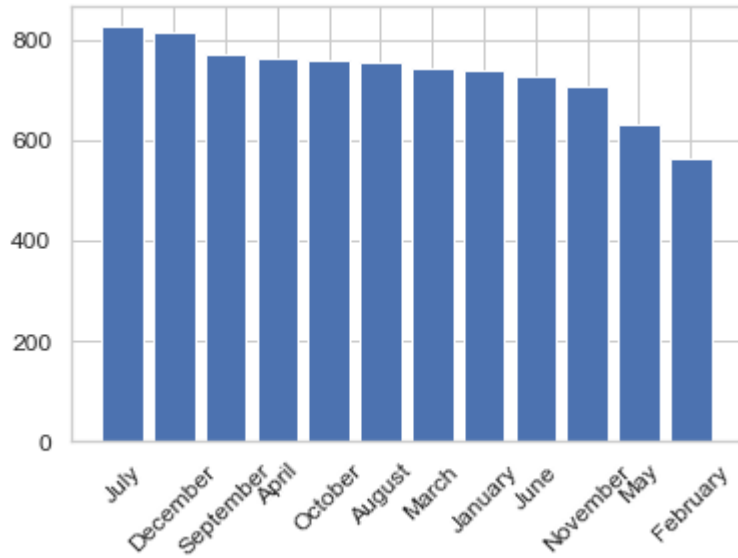
In [166]:

```
plt.bar(df.groupby(df["date_added"].dt.day_name())["title"].nunique().sort_va
 df.groupby(df["date_added"].dt.day_name())["title"].nunique().sort_values(as
plt.xticks(rotation = 45)
plt.show()
```



In [167]:

```
plt.bar(df.groupby(df["date_added"].dt.month_name())["title"].nunique().sort_
 df.groupby(df["date_added"].dt.month_name())["title"].nunique().sort_values(
plt.xticks(rotation = 45)
plt.show()
```




## Top actors in India

In [177]:

```
# top actors in India

top_10_actors_India = df.groupby(['countries','actors'])['title'].count().res
top_10_actors_India[top_10_actors_India['countries'] == 'India'].sort_values(
```

Out[177]:

| | countries | actors | title |
|---|---|---|---|
| **14233** | India | Anupam Kher | 113 |
| **16873** | India | Shah Rukh Khan | 98 |
| **17460** | India | Unknown | 97 |
| **15879** | India | Naseeruddin Shah | 95 |
| **14001** | India | Akshay Kumar | 87 |
| **16297** | India | Radhika Apte | 86 |
| **16079** | India | Paresh Rawal | 85 |
| **14073** | India | Amitabh Bachchan | 85 |
| **16029** | India | Om Puri | 79 |
| **15324** | India | Kareena Kapoor | 76 |

## Top directors in India

In [176]:

```
# top actors in India

top_10_directors_India = df.groupby(['countries','directors'])['title'].count
top_10_directors_India[top_10_directors_India['countries'] == 'India'].sort_v
```

Out[176]:

| | countries | directors | title |
|---|---|---|---|
| **2335** | India | Unknown | 995 |
| **1802** | India | David Dhawan | 270 |
| **1731** | India | Anurag Kashyap | 219 |
| **1819** | India | Dibakar Banerjee | 192 |
| **2268** | India | Sooraj R. Barjatya | 180 |
| **2386** | India | Zoya Akhtar | 168 |
| **2333** | India | Umesh Mehra | 162 |
| **2129** | India | Ram Gopal Varma | 158 |
| **2078** | India | Priyadarshan | 156 |
| **1906** | India | Karan Johar | 155 |

## genre wise distribution in India

In [175]:

```
# genre wise distribution in India
genre_wise_dis_in_india = df.groupby(['countries','genre'])['title'].count().
genre_wise_dis_in_india= genre_wise_dis_in_india[genre_wise_dis_in_india
                                                 ['countries'] == "India"].sc
genre_wise_dis_in_india
```

Out[175]:

| | countries | genre | title |
|---|---|---|---|
| **526** | India | International Movies | 7059 |
| **522** | India | Dramas | 5569 |
| **517** | India | Comedies | 2685 |
| **525** | India | Independent Movies | 1394 |
| **513** | India | Action & Adventure | 1187 |
| **532** | India | Romantic Movies | 931 |
| **530** | India | Music & Musicals | 847 |
| **547** | India | Thrillers | 743 |
| **527** | India | International TV Shows | 428 |
| **524** | India | Horror Movies | 307 |
| **540** | India | TV Dramas | 272 |
| **515** | India | Children & Family Movies | 225 |
| **544** | India | TV Shows | 207 |
| **539** | India | TV Comedies | 141 |
| **535** | India | Sports Movies | 121 |
| **534** | India | Sci-Fi & Fantasy | 111 |
| **516** | India | Classic Movies | 98 |
| **533** | India | Romantic TV Shows | 68 |
| **518** | India | Crime TV Shows | 61 |
| **528** | India | Kids' TV | 57 |
| **538** | India | TV Action & Adventure | 44 |
| **519** | India | Cult Movies | 42 |
| **529** | India | LGBTQ Movies | 33 |
| **520** | India | Documentaries | 32 |
| **541** | India | TV Horror | 28 |
| **543** | India | TV Sci-Fi & Fantasy | 27 |
| **523** | India | Faith & Spirituality | 20 |
| **514** | India | British TV Shows | 19 |
| **521** | India | Docuseries | 15 |
| **542** | India | TV Mysteries | 11 |
| **537** | India | Stand-Up Comedy & Talk Shows | 8 |
| **536** | India | Stand-Up Comedy | 7 |

| | countries | genre | title |
|---|---|---|---|
| **531** | India | Reality TV | 7 |
| **546** | India | Teen TV Shows | 7 |
| **545** | India | TV Thrillers | 3 |

## Top 10 genre in India

In [171]:

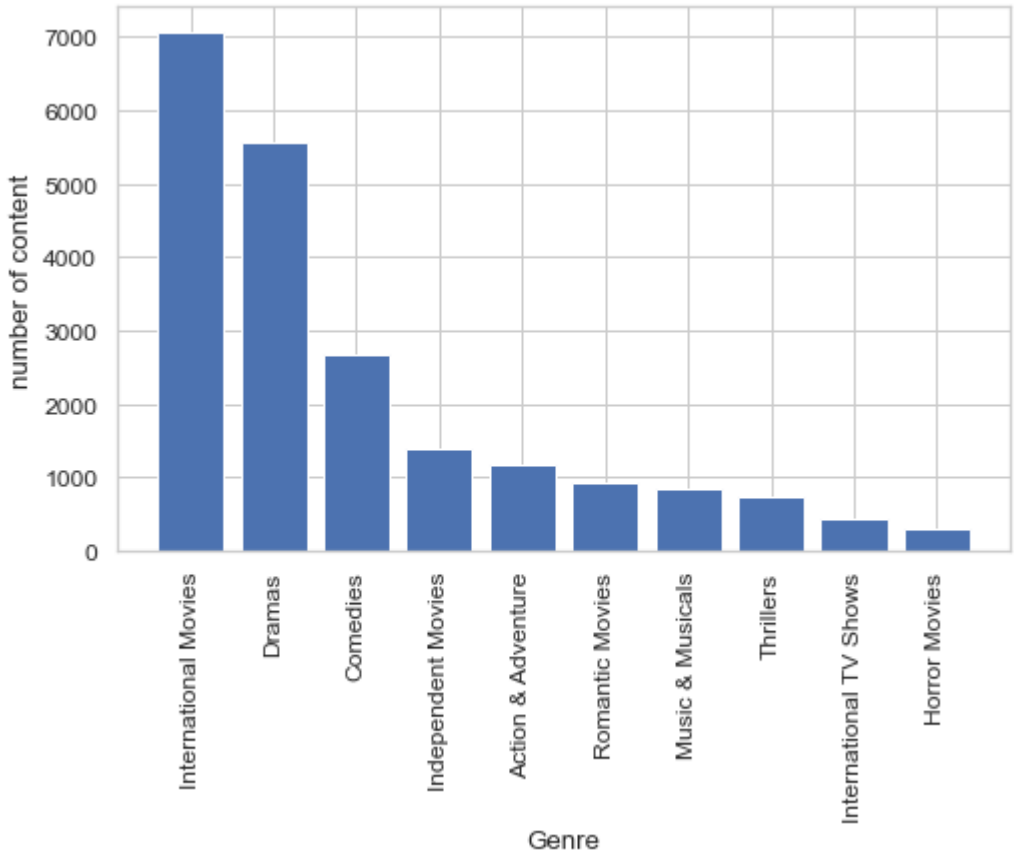
```
top_10_genre_india =  genre_wise_dis_in_india.head(10)
top_10_genre_india
```

Out[171]:

| | countries | genre | title |
|---|---|---|---|
| **526** | India | International Movies | 7059 |
| **522** | India | Dramas | 5569 |
| **517** | India | Comedies | 2685 |
| **525** | India | Independent Movies | 1394 |
| **513** | India | Action & Adventure | 1187 |
| **532** | India | Romantic Movies | 931 |
| **530** | India | Music & Musicals | 847 |
| **547** | India | Thrillers | 743 |
| **527** | India | International TV Shows | 428 |
| **524** | India | Horror Movies | 307 |

In [172]:

```
# most popular genre in india top 10 polt

plt.figure(figsize=(8,5))
plt.bar(top_10_genre_india['genre'],top_10_genre_india['title'])
plt.xlabel('Genre')
plt.ylabel('number of content')
plt.xticks(rotation=90)
plt.show()
```



In [173]:
```
df.head()
```

Out[173]:

| | type | title | date_added | release_year | rating | year_added | month_added | actors | dire |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Movie | Dick Johnson Is Dead | 2021-09-25 | 2020 | PG-13 | 2021.0 | 9.0 | Unknown | K Jo |
| **1** | TV Show | Blood & Water | 2021-09-24 | 2021 | TV-MA | 2021.0 | 9.0 | Ama Qamata | Unk |
| **2** | TV Show | Blood & Water | 2021-09-24 | 2021 | TV-MA | 2021.0 | 9.0 | Ama Qamata | Unk |
| **3** | TV Show | Blood & Water | 2021-09-24 | 2021 | TV-MA | 2021.0 | 9.0 | Ama Qamata | Unk |
| **4** | TV Show | Blood & Water | 2021-09-24 | 2021 | TV-MA | 2021.0 | 9.0 | Khosi Ngema | Unk |

In [ ]:

In [ ]:

In [ ]: