In [106]:
```
pip install WordCloud
```

```
Requirement already satisfied: WordCloud in c:\users\mahes\appdata\l
ocal\programs\python\python311\lib\site-packages (1.9.2)
Requirement already satisfied: numpy>=1.6.1 in c:\users\mahes\appdat
a\local\programs\python\python311\lib\site-packages (from WordCloud)
(1.25.1)
Requirement already satisfied: pillow in c:\users\mahes\appdata\loca
l\programs\python\python311\lib\site-packages (from WordCloud) (10.
0.0)
Requirement already satisfied: matplotlib in c:\users\mahes\appdata
\local\programs\python\python311\lib\site-packages (from WordCloud)
(3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\mahes\ap
pdata\local\programs\python\python311\lib\site-packages (from matplo
tlib->WordCloud) (1.1.0)
Requirement already satisfied: cycler>=0.10 in c:\users\mahes\appdat
a\local\programs\python\python311\lib\site-packages (from matplotlib
->WordCloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\mahes\a
ppdata\local\programs\python\python311\lib\site-packages (from matpl
```

Netflix project By mahesh sharma

In [107]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
```

In [108]:
```
df=pd.read_csv('original_netflix.csv')
```

```
df.head()
```

In [109]:
```
! dir
```

```
 Volume in drive C is Windows
 Volume Serial Number is BE60-998E

 Directory of C:\Users\mahes\Desktop\NetflixEDA-main

07-09-2023  18:32    <DIR>          .
04-09-2023  19:10    <DIR>          ..
02-09-2023  21:33    <DIR>          .ipynb_checkpoints
02-09-2023  00:04            13,981 height weight gaussio.ipynb
07-09-2023  18:24           121,121 netflixeda.ipynb
18-08-2023  18:35         2,765,361 netflixeda.ipynb - Colaboratory.pd
f
18-08-2023  18:38         3,399,671 original_netflix.csv
07-09-2023  15:38             2,070 Untitled.ipynb
18-08-2023  18:48                72 Untitled1.ipynb
07-09-2023  18:32         2,007,095 Untitled2.ipynb
20-08-2023  06:24               617 Untitled3.ipynb
20-08-2023  06:30               617 Untitled4.ipynb
23-08-2023  21:12             3,340 Untitled5.ipynb
28-08-2023  19:57             1,051 Untitled6.ipynb
28-08-2023  17:43             2,889 Untitled7.ipynb
01-09-2023  20:42           428,120 weight-height.csv
              13 File(s)      8,746,005 bytes
               3 Dir(s)  304,866,480,128 bytes free
```

In [110]:
```
df.shape
```

Out[110]:
```
(8807, 12)
```

In [111]:
```
df.columns
```

Out[111]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'dat
e_added',
       'release_year', 'rating', 'duration', 'listed_in', 'descriptio
n'],
      dtype='object')
```

Dropping the show_id column as it may be of no use in the analysis

In [112]:
```
df.drop('show_id',axis =1,inplace=True)
```

In [113]:
```
df.columns
```

Out[113]:
```
Index(['type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'descriptio
n'],
      dtype='object')
```

In [114]:
```
df.duplicated().sum()
```

Out[114]:
```
0
```

In [115]:
```
df.nunique().sort_values(ascending =False)
```

Out[115]:
```
title           8807
description     8775
cast            7692
director        4528
date_added      1767
country          748
listed_in        514
duration         220
release_year      74
rating            17
type               2
dtype: int64
```

In [116]:
```
df.isna().sum().sort_values(ascending = False)
```

Out[116]:
```
director        2634
country          831
cast             825
date_added        10
rating             4
duration           3
type               0
title              0
release_year       0
listed_in          0
description        0
dtype: int64
```

In [ ]:

In [ ]:

In [117]:
```
percentage = round(df.isna().mean()*100,2).sort_values(ascending = False
null = df.isna().sum().sort_values(ascending=False)
missing_value = pd.concat([null,percentage],axis= 1,keys= ['Missing_valu
print('Total records present-------------', df.shape[0])
print('missing_value_count--------',missing_value[missing_value['Missing
print('-------------------------------------------------')
print('total                        ',missing_value['Missing_values_count'
```

```
Total records present------------- 8807
missing_value_count--------             Missing_values_count  percenta
ge
director                     2634       29.91
country                       831        9.44
cast                          825        9.37
date_added                     10        0.11
rating                          4        0.05
duration                        3        0.03
-------------------------------------------------
total                        4307       48.91
```

Dropping the rows which have null values in date_added column as there are only 10 values which sums up to only 0.11% of the total values

In [118]:
```
df.dropna(subset='date_added',inplace=True)
```

In [119]:
```
df.date_added.isna().sum()
```

Out[119]:
```
0
```

Dropping the rows which have null values in rating column as there are only 4 values which sums up to only 0.05% of the total values

In [120]:
```
df.dropna(subset='rating',inplace=True)
```

In [121]:
```
df.rating.isna().sum()
```

Out[121]:
```
0
```

Dropping the rows which have null values in duration column as there are only 3 values which sums up to only 0.03% of the total values

In [122]:
```
df.dropna(subset='duration',inplace=True)
```

In [123]:
```
df.duration.isna().sum()
```

Out[123]:
```
0
```

In [124]:
```
percentage = round(df.isna().mean()*100,2).sort_values(ascending = False
null = df.isna().sum().sort_values(ascending=False)
missing_value = pd.concat([null,percentage],axis= 1,keys= ['Missing_valu
print('Total records present-------------', df.shape[0])
print('missing_value_count--------',missing_value[missing_value['Missing
print('-------------------------------------------------')
print('total                    ',missing_value['Missing_values_count'].
```

```
Total records present------------- 8790
missing_value_count--------          Missing_values_count  percentage
director                   2621       29.82
country                     829        9.43
cast                        825        9.39
-------------------------------------------------
total                      4275       48.64
```

In [125]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8790 entries, 0 to 8806
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   type          8790 non-null   object
 1   title         8790 non-null   object
 2   director      6169 non-null   object
 3   cast          7965 non-null   object
 4   country       7961 non-null   object
 5   date_added    8790 non-null   object
 6   release_year  8790 non-null   int64 
 7   rating        8790 non-null   object
 8   duration      8790 non-null   object
 9   listed_in     8790 non-null   object
 10  description   8790 non-null   object
dtypes: int64(1), object(10)
memory usage: 824.1+ KB
```

In [126]:
```
df['date_added']=pd.to_datetime(df['date_added'],format='mixed')
```

In [127]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8790 entries, 0 to 8806
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   type          8790 non-null   object
 1   title         8790 non-null   object
 2   director      6169 non-null   object
 3   cast          7965 non-null   object
 4   country       7961 non-null   object
 5   date_added    8790 non-null   datetime64[ns]
 6   release_year  8790 non-null   int64 
 7   rating        8790 non-null   object
 8   duration      8790 non-null   object
 9   listed_in     8790 non-null   object
 10  description   8790 non-null   object
dtypes: datetime64[ns](1), int64(1), object(9)
memory usage: 824.1+ KB
```

Adding day, month, year, month_name, day_name as seperate columns to the dataframe as these will help us in analysis

In [128]:
```
df['day']=df['date_added'].dt.day
df['month']=df['date_added'].dt.month
df['month_name']=df['date_added'].dt.month_name()
df['year']=df['date_added'].dt.year
df['day_name']=df['date_added'].dt.day_name()
```

In [129]:
```
df.columns
```

Out[129]:
```
Index(['type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'descriptio
n', 'day',
       'month', 'month_name', 'year', 'day_name'],
      dtype='object')
```

In [130]:
```
df.rating.unique()
```

Out[130]:
```
array(['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7', 'R',
       'TV-G', 'G', 'NC-17', 'NR', 'TV-Y7-FV', 'UR'], dtype=object)
```

In [131]:
```
df.rating.replace(['TV-Y','TV-Y7','G','TV-G','PG','TV-PG','TV-Y7-FV'],'K
```

In [132]:
```
df['rating'].replace(['PG-13','TV-14'],'Teens',inplace=True)
```

In [133]:
```
df.rating.replace(['R','TV-MA','NC-17'],'Adults',inplace=True)
```

In [134]:
```
df.rating.replace(['NR','UR'],np.nan,inplace=True)
```

In [135]:
```
df.rating.unique()
```

Out[135]:
```
array(['Teens', 'Adults', 'Kids', nan], dtype=object)
```

From here it is confirmed that values have been replaced successfully and now we will delete rows that have null values.

In [136]:
```
df.rating.isnull().sum()
```

Out[136]:
```
82
```

In [137]:
```
df.dropna(subset='rating',inplace=True)
```

In [138]:
```
df.rating.isnull().sum()
```

Out[138]:
```
0
```

82 values were null, whhich was only about 1% of the data, so we deleted those rows.

In [139]:
```
percentage = round(df.isna().mean()*100,2).sort_values(ascending = False
null = df.isna().sum().sort_values(ascending=False)
missing_value = pd.concat([null,percentage],axis= 1,keys= ['Missing_valu
print('Total records present-------------', df.shape[0])
print('missing_value_count--------',missing_value[missing_value['Missing
print('-------------------------------------------------')
print('total                    ',missing_value['Missing_values_count'].
```

```
Total records present------------- 8708
missing_value_count--------           Missing_values_count  percentage
director                   2617       30.05
country                     829        9.52
cast                        808        9.28
-------------------------------------------------
total                      4254       48.85
```

Now, we still have about 48% missing data. So, we will check in which rows the director, country and cast all three are null, if all these 3 are null then there is no use of that row.

In [140]:
```
df.loc[(df['director'].isna()) & (df['country'].isna()) & (df['cast'].is
```

Out[140]: 96

We found out that total 96 such rows are there which dont have director, country and cast as null, so deleting those rows.

In [141]:
```
df.dropna(subset=['director','country','cast'],how='all',inplace=True)
```

In [142]:
```
percentage = round(df.isna().mean()*100,2).sort_values(ascending = False
null = df.isna().sum().sort_values(ascending=False)
missing_value = pd.concat([null,percentage],axis= 1,keys= ['Missing_valu
print('Total records present-------------', df.shape[0])
print('missing_value_count--------',missing_value[missing_value['Missing
print('-----------------------------------------------')
print('total                    ',missing_value['Missing_values_count'].
```

```
Total records present------------- 8612
missing_value_count--------          Missing_values_count  percentage
director                  2521       29.27
country                    733        8.51
cast                       712        8.27
-----------------------------------------------
total                     3966       46.05
```

As the missing value percentage of these rows are significant, we cant delete them so we will dealwith those later

and now lets focus

on one more major problem in the dataset which is NESTED DATA in director, cast, country, and genre.

In [143]:
```
df.head()
```

Out[143]:

	type	title	director	cast	country	date_added	release_year	rating	dur
0	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	2020	Teens	9
1	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021	Adults	Sea
2	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	2021-09-24	2021	Adults	Se
4	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	2021-09-24	2021	Adults	Sea
5	TV Show	Midnight Mass	Mike Flanagan	Kate Siegel, Zach Gilford, Hamish Linklater, H...	NaN	2021-09-24	2021	Adults	Se

Now we will split all the nested data individually and save them in a new dataframe, and then we will merge all of them so get the final dataframe.

In the process we will lose the format of null values from np.NaN to string NaN, so that requires one additional step which is done in the last 2 lines of each block We can confirm the count of null values from the above block

In [144]:
```
df_cast = pd.DataFrame(df['cast'].apply(lambda x: str(x).split(', ')).to
df_cast = df_cast.stack()
df_cast = pd.DataFrame(df_cast)
df_cast.reset_index(inplace=True)
df_cast = df_cast[['title',0]]
df_cast.columns =['title','cast']
df_cast.replace('nan',np.NaN,inplace=True)
df_cast.isna().sum()
```

Out[144]:
```
title      0
cast     712
dtype: int64
```

In [145]:
```
df.rename(columns={'listed_in' : 'genre'},inplace=True)
df.head()
```

Out[145]:

	type	title	director	cast	country	date_added	release_year	rating	dur
0	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	2020	Teens	9
1	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021	Adults	Sea
2	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	2021-09-24	2021	Adults	Se
4	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	2021-09-24	2021	Adults	Sea
5	TV Show	Midnight Mass	Mike Flanagan	Kate Siegel, Zach Gilford, Hamish Linklater, H...	NaN	2021-09-24	2021	Adults	Se

In [146]:
```
df_genre = pd.DataFrame(df['genre'].apply(lambda x: str(x).split(', ')).
df_genre = df_genre.stack()
df_genre = pd.DataFrame(df_genre)
df_genre.reset_index(inplace=True)
df_genre = df_genre[['title',0]]
df_genre.columns =['title','genre']
df_genre.replace('nan',np.NaN,inplace=True)
df_genre.isna().sum()
```

Out[146]:
```
title    0
genre    0
dtype: int64
```

In [147]:
```
df_director = pd.DataFrame(df['director'].apply(lambda x: str(x).split('
df_director = df_director.stack()
df_director = pd.DataFrame(df_director)
df_director.reset_index(inplace=True)
df_director = df_director[['title',0]]
df_director.columns =['title','director']
df_director.replace('nan',np.NaN,inplace=True)
df_director.isna().sum()
```

Out[147]:
```
title          0
director    2521
dtype: int64
```

In [148]:
```
df_country = pd.DataFrame(df['country'].apply(lambda x: str(x).split(',
df_country = df_country.stack()
df_country = pd.DataFrame(df_country)
df_country.reset_index(inplace=True)
df_country = df_country[['title',0]]
df_country.columns =['title','country']
df_country.replace('nan',np.NaN,inplace=True)
df_country.isna().sum()
```

Out[148]:
```
title        0
country    733
dtype: int64
```

In [149]:
```
df12 = df_cast.merge(df_genre, on = 'title')
df12
```

Out[149]:

	title	cast	genre
0	Dick Johnson Is Dead	NaN	Documentaries
1	Blood & Water	Ama Qamata	International TV Shows
2	Blood & Water	Ama Qamata	TV Dramas
3	Blood & Water	Ama Qamata	TV Mysteries
4	Blood & Water	Khosi Ngema	International TV Shows
...	...	...	...
147927	Zubaan	Anita Shabdish	International Movies
147928	Zubaan	Anita Shabdish	Music & Musicals
147929	Zubaan	Chittaranjan Tripathy	Dramas
147930	Zubaan	Chittaranjan Tripathy	International Movies
147931	Zubaan	Chittaranjan Tripathy	Music & Musicals

147932 rows × 3 columns

In [150]:
```
df123 = df_director.merge(df12, on = 'title')
df123
```

Out[150]:

	title	director	cast	genre
0	Dick Johnson Is Dead	Kirsten Johnson	NaN	Documentaries
1	Blood & Water	NaN	Ama Qamata	International TV Shows
2	Blood & Water	NaN	Ama Qamata	TV Dramas
3	Blood & Water	NaN	Ama Qamata	TV Mysteries
4	Blood & Water	NaN	Khosi Ngema	International TV Shows
...	...	...	...	...
159583	Zubaan	Mozez Singh	Anita Shabdish	International Movies
159584	Zubaan	Mozez Singh	Anita Shabdish	Music & Musicals
159585	Zubaan	Mozez Singh	Chittaranjan Tripathy	Dramas
159586	Zubaan	Mozez Singh	Chittaranjan Tripathy	International Movies
159587	Zubaan	Mozez Singh	Chittaranjan Tripathy	Music & Musicals

159588 rows × 4 columns

In [151]:
```
df1234 = df_country.merge(df123, on = 'title')
df1234
```

Out[151]:

	title	country	director	cast	genre
0	Dick Johnson Is Dead	United States	Kirsten Johnson	NaN	Documentaries
1	Blood & Water	South Africa	NaN	Ama Qamata	International TV Shows
2	Blood & Water	South Africa	NaN	Ama Qamata	TV Dramas
3	Blood & Water	South Africa	NaN	Ama Qamata	TV Mysteries
4	Blood & Water	South Africa	NaN	Khosi Ngema	International TV Shows
...	...	...	...	...	...
199945	Zubaan	India	Mozez Singh	Anita Shabdish	International Movies
199946	Zubaan	India	Mozez Singh	Anita Shabdish	Music & Musicals
199947	Zubaan	India	Mozez Singh	Chittaranjan Tripathy	Dramas
199948	Zubaan	India	Mozez Singh	Chittaranjan Tripathy	International Movies
199949	Zubaan	India	Mozez Singh	Chittaranjan Tripathy	Music & Musicals

199950 rows × 5 columns

In [152]:
```
df_new = df.merge(df1234, on = 'title')
df_new.duplicated().sum()
```

Out[152]:
```
55
```

In [153]:
```
df_new.columns
```

Out[153]:
```
Index(['type', 'title', 'director_x', 'cast_x', 'country_x', 'date_add
ed',
       'release_year', 'rating', 'duration', 'genre_x', 'description',
'day',
       'month', 'month_name', 'year', 'day_name', 'country_y', 'direct
or_y',
       'cast_y', 'genre_y'],
      dtype='object')
```

In [ ]:

In [154]:
```
df_new.drop(columns=['director_x','country_x','cast_x','genre_x'],axis =
```

In [155]:
```
df_new.rename(columns={'country_y':'country','director_y': 'director','c
```

In [156]:
```
df_new.duplicated().sum()
```

Out[156]:
```
55
```

In [157]:
```
df_new.shape
```

Out[157]:
```
(199950, 16)
```

This is the new dataframe which has 199950 rows and 16 columns after unnesting the data

In [158]:
```
percentage = round(df_new.isna().mean()*100,2).sort_values(ascending = F
null = df_new.isna().sum().sort_values(ascending=False)
missing_value = pd.concat([null,percentage],axis= 1,keys= ['Missing_valu
print('Total records present-------------', df_new.shape[0])
print('missing_value_count--------',missing_value[missing_value['Missing
print('-------------------------------------------------')
print('total                    ',missing_value['Missing_values_count'].
```

```
Total records present------------- 199950
missing_value_count--------          Missing_values_count  percentage
director                 50116       25.06
country                  11710        5.86
cast                      1894        0.95
-------------------------------------------------
total                    63720       31.869999999999997
```

In [159]:
```
df_new['director']=df_new['director'].replace(np.nan,'unknown')
df_new['country']=df_new['country'].replace(np.nan,'unknown')
df_new['cast']=df_new['cast'].replace(np.nan,'unknown')
```

In [160]:
```
percentage = round(df_new.isna().mean()*100,2).sort_values(ascending = F
null = df_new.isna().sum().sort_values(ascending=False)
missing_value = pd.concat([null,percentage],axis= 1,keys= ['Missing_valu
print('Total records present-------------', df_new.shape[0])
print('missing_value_count--------',missing_value[missing_value['Missing
print('-------------------------------------------------')
print('total                    ',missing_value['Missing_values_count'].
```

```
Total records present------------- 199950
missing_value_count-------- Empty DataFrame
Columns: [Missing_values_count, percentage]
Index: []
-------------------------------------------------
total                    0       0.0
```

After replacing we can confirm no more null values

In [161]:
```
df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199950 entries, 0 to 199949
Data columns (total 16 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   type          199950 non-null  object
 1   title         199950 non-null  object
 2   date_added    199950 non-null  datetime64[ns]
 3   release_year  199950 non-null  int64
 4   rating        199950 non-null  object
 5   duration      199950 non-null  object
 6   description   199950 non-null  object
 7   day           199950 non-null  int32
 8   month         199950 non-null  int32
 9   month_name    199950 non-null  object
 10  year          199950 non-null  int32
 11  day_name      199950 non-null  object
 12  country       199950 non-null  object
 13  director      199950 non-null  object
 14  cast          199950 non-null  object
 15  genre         199950 non-null  object
dtypes: datetime64[ns](1), int32(3), int64(1), object(11)
memory usage: 22.1+ MB
```

In [162]:
```
df.duplicated().sum()
```

Out[162]:
```
0
```

Creating 2 more dataframes for better analysis by dividing the present dataframe on the basis of type i.e Movie and TV Show and naming them df_movies and df_tvs.

In [163]:
```
df_tv_show = df_new.loc[df_new['type']=='TV Show']
```

In [164]:
```
df_Movie = df_new.loc[df_new['type']=='Movie']
```

In [165]:
```
df_Movie.shape
```

Out[165]:
```
(144295, 16)
```

In [166]:
```
df_tv_show.shape
```

Out[166]:
```
(55655, 16)
```

In [167]:
```
df_new.shape
```

Out[167]:
```
(199950, 16)
```

In [168]:
```
df_new.type.unique()
```

Out[168]:
```
array(['Movie', 'TV Show'], dtype=object)
```

In [169]:
```
df_tv_show['duration'].unique()
```

Out[169]:
```
array(['2 Seasons', '1 Season', '9 Seasons', '4 Seasons', '5 Seasons',
       '3 Seasons', '6 Seasons', '7 Seasons', '10 Seasons', '8 Season
s',
       '17 Seasons', '13 Seasons', '15 Seasons', '12 Seasons',
       '11 Seasons'], dtype=object)
```

In [170]:
```
df_Movie['duration'].unique()
```

Out[170]:
```
array(['90 min', '91 min', '125 min', '104 min', '127 min', '67 min',
       '94 min', '161 min', '61 min', '166 min', '147 min', '103 min',
       '97 min', '106 min', '111 min', '110 min', '105 min', '96 min',
       '124 min', '116 min', '98 min', '23 min', '115 min', '122 min',
       '99 min', '88 min', '100 min', '102 min', '93 min', '95 min',
       '85 min', '83 min', '113 min', '13 min', '182 min', '48 min',
       '145 min', '87 min', '92 min', '80 min', '117 min', '128 min',
       '119 min', '143 min', '114 min', '118 min', '108 min', '63 mi
n',
       '121 min', '142 min', '154 min', '120 min', '82 min', '109 mi
n',
       '101 min', '86 min', '229 min', '76 min', '89 min', '156 min',
       '112 min', '107 min', '129 min', '135 min', '136 min', '165 mi
n',
       '150 min', '133 min', '70 min', '84 min', '140 min', '78 min',
       '64 min', '59 min', '139 min', '69 min', '148 min', '189 min',
       '141 min', '130 min', '138 min', '81 min', '132 min', '123 mi
n',
       '65 min', '68 min', '66 min', '62 min', '74 min', '131 min',
       '39 min', '46 min', '38 min', '126 min', '155 min', '159 min',
       '137 min', '12 min', '273 min', '36 min', '34 min', '77 min',
       '60 min', '49 min', '58 min', '72 min', '204 min', '212 min',
       '25 min', '73 min', '47 min', '32 min', '35 min', '71 min',
       '149 min', '33 min', '15 min', '54 min', '224 min', '162 min',
       '37 min', '75 min', '79 min', '55 min', '158 min', '164 min',
       '173 min', '181 min', '185 min', '21 min', '24 min', '51 min',
       '151 min', '42 min', '22 min', '134 min', '177 min', '52 min',
       '14 min', '53 min', '8 min', '57 min', '28 min', '50 min', '9 m
in',
       '26 min', '45 min', '171 min', '27 min', '44 min', '29 min',
       '146 min', '20 min', '157 min', '17 min', '203 min', '41 min',
       '30 min', '194 min', '233 min', '237 min', '230 min', '195 mi
n',
       '253 min', '152 min', '190 min', '160 min', '208 min', '180 mi
n',
       '144 min', '5 min', '174 min', '170 min', '192 min', '209 min',
       '187 min', '172 min', '16 min', '186 min', '11 min', '193 min',
       '176 min', '56 min', '169 min', '40 min', '10 min', '3 min',
       '168 min', '312 min', '153 min', '214 min', '31 min', '163 mi
n',
       '19 min', '179 min', '43 min', '200 min', '196 min', '167 min',
       '178 min', '228 min', '18 min', '205 min', '201 min', '191 mi
n'],
      dtype=object)
```

In [171]:
```
df_Movie['duration'] = df_Movie['duration'].str.replace('min',"")
df_Movie.head()
```

```
C:\Users\mahes\AppData\Local\Temp\ipykernel_10844\3712849509.py:1: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
  df_Movie['duration'] = df_Movie['duration'].str.replace('min',"")
```

Out[171]:

	type	title	date_added	release_year	rating	duration	description	day	mo
0	Movie	Dick Johnson Is Dead	2021-09-25	2020	Teens	90	As her father nears the end of his life, filmm...	25	
157	Movie	My Little Pony: A New Generation	2021-09-24	2021	Kids	91	Equestria's divided. But a bright-eyed hero be...	24	
158	Movie	My Little Pony: A New Generation	2021-09-24	2021	Kids	91	Equestria's divided. But a bright-eyed hero be...	24	
159	Movie	My Little Pony: A New Generation	2021-09-24	2021	Kids	91	Equestria's divided. But a bright-eyed hero be...	24	
160	Movie	My Little Pony: A New Generation	2021-09-24	2021	Kids	91	Equestria's divided. But a bright-eyed hero be...	24	

In [172]:

```
df_tv_show['duration'] = df_tv_show['duration'].str.replace('Seasons',"")
df_tv_show['duration'] = df_tv_show['duration'].str.replace('Season',"")
df_tv_show.head()
```

```
C:\Users\mahes\AppData\Local\Temp\ipykernel_10844\194583511.py:1: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
  df_tv_show['duration'] = df_tv_show['duration'].str.replace('Season
s',"")
C:\Users\mahes\AppData\Local\Temp\ipykernel_10844\194583511.py:2: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
  df_tv_show['duration'] = df_tv_show['duration'].str.replace('Seaso
n',"")
```

Out[172]:

	type	title	date_added	release_year	rating	duration	description	day	month	mo
1	TV Show	Blood & Water	2021-09-24	2021	Adults	2	After crossing paths at a party, a Cape Town t...	24	9	
2	TV Show	Blood & Water	2021-09-24	2021	Adults	2	After crossing paths at a party, a Cape Town t...	24	9	
3	TV Show	Blood & Water	2021-09-24	2021	Adults	2	After crossing paths at a party, a Cape Town t...	24	9	
4	TV Show	Blood & Water	2021-09-24	2021	Adults	2	After crossing paths at a party, a Cape Town t...	24	9	
5	TV Show	Blood & Water	2021-09-24	2021	Adults	2	After crossing paths at a party, a Cape Town t...	24	9	

In [173]:
```
df_tv_show.rename(columns={'duration':'seasons'},inplace=True)
```

```
C:\Users\mahes\AppData\Local\Temp\ipykernel_10844\931575804.py:1: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
  df_tv_show.rename(columns={'duration':'seasons'},inplace=True)
```

Finally, we have 4 dataframes, we will use all of these according to the analysis required ahead

df = Cleaned data before unnesting.

df_new = Cleaned data after unnesting.

df_movies = Cleaned data of type - movie after unnesting.

df_tvs = Cleaned data of type - TV Show after unnesting

We are ready with out dataframes, and also ready to begin our EDA!

Lets start with checking the time period of our data.

In [174]:
```
print(df['year'].max())
```

```
2021
```

In [175]:
```
print(df['year'].min())
```

```
2008
```

- The data lies between the year 2008 and 2021.

Now lets see the count of content available on Netflix.

In [190]:
```
df.shape[0]
```

Out[190]:
```
8612
```

- The Netflix library has 8612 movies or shows to watch.

Now, lets see the count of movies and TV Shows individually.

In [191]:
```
df_director['director'].nunique()
```

Out[191]:
```
4925
```

- There are a total of 4925 directors present in the data

Now, lets see the count of actors present.

In [192]:
```
df_cast['cast'].nunique()
```

Out[192]:
```
36148
```

- There are a total of 36148 actors present in the data.

Lets check in how many countries in the data distributed in.

In [193]:
```
df_country['country'].nunique()
```

Out[193]:
```
127
```

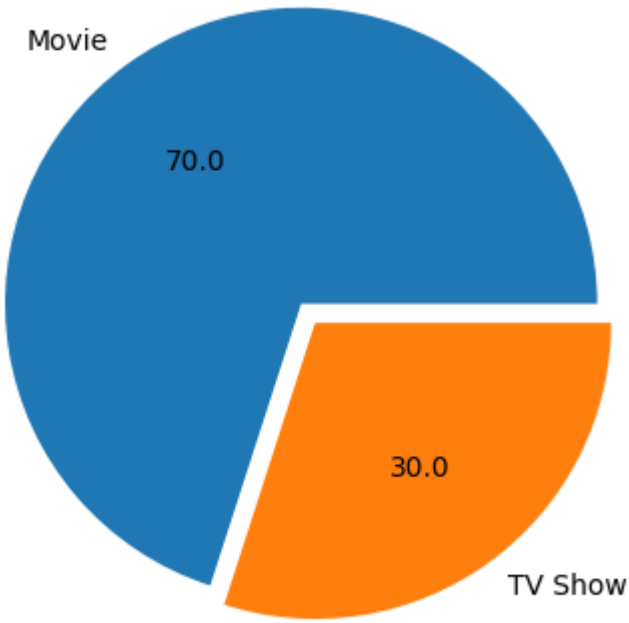
- The content is distributed across 127 countries.

Lets start our visualisation with seeing the distribution of content on Netflix. Here we will use df as we only need to count the row once.

In [194]:
```
labels = df['type'].unique().tolist()
```

In [195]:
```
plt.pie(df.groupby('type')['type'].count().tolist(),explode=(0.08,0),lab
```

Out[195]:
```
([<matplotlib.patches.Wedge at 0x221693a7bd0>,
  <matplotlib.patches.Wedge at 0x221693986d0>],
 [Text(-0.6944916677933262, 0.9539818254902158, 'Movie'),
  Text(0.6474074036632277, -0.8893051521733351, 'TV Show')],
 [Text(-0.4002155373724252, 0.5497522384180904, '70.0'),
  Text(0.3531313110890333, -0.4850755375490918, '30.0')])
```



Inference

- Netflix has 70% of its content as movies.

- TV Shows are clearly lesser than Movies.

Recommendations

- More TV Shows should be added as they create more suspense and have more story than a 100 minute movie.
- Users like to binge watch a particular story for a longer time.

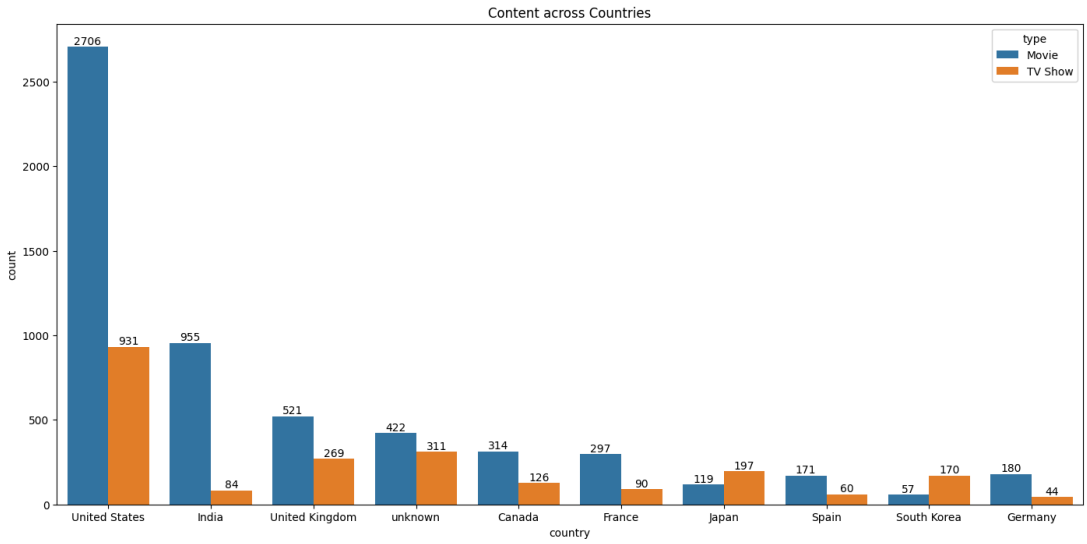
Lets see the content distribution across countries.

Here we need the unnested data for countries which is in df_new and we need to count the titles. After unnesting there is only one dataframe which has multiple rows because we also unnested the director, cast and genre, so hence we will do a drop_duplicate function on the df_new so that we will only get one row of the title and country group.

In [ ]:

In [196]:

```
df_temp = df_new.drop_duplicates(subset = ['country', 'title'])
x = df_temp['country'].value_counts().head(10)

plt.figure(figsize = (17,8))
plt.title('Content across Countries')
label = sns.countplot(data = df_temp, x = 'country', hue = 'type', order
for i in label.containers:
    label.bar_label(i)
plt.show()
```



Inference

- US has the most content for movies followed by India and UK.
- TV Shows are mostly created in the US and UK.
- We can also observe that all the countries have more movies than TV Shows, whereas Japan and South Korea have more TV shows than movies.

Recommendation**

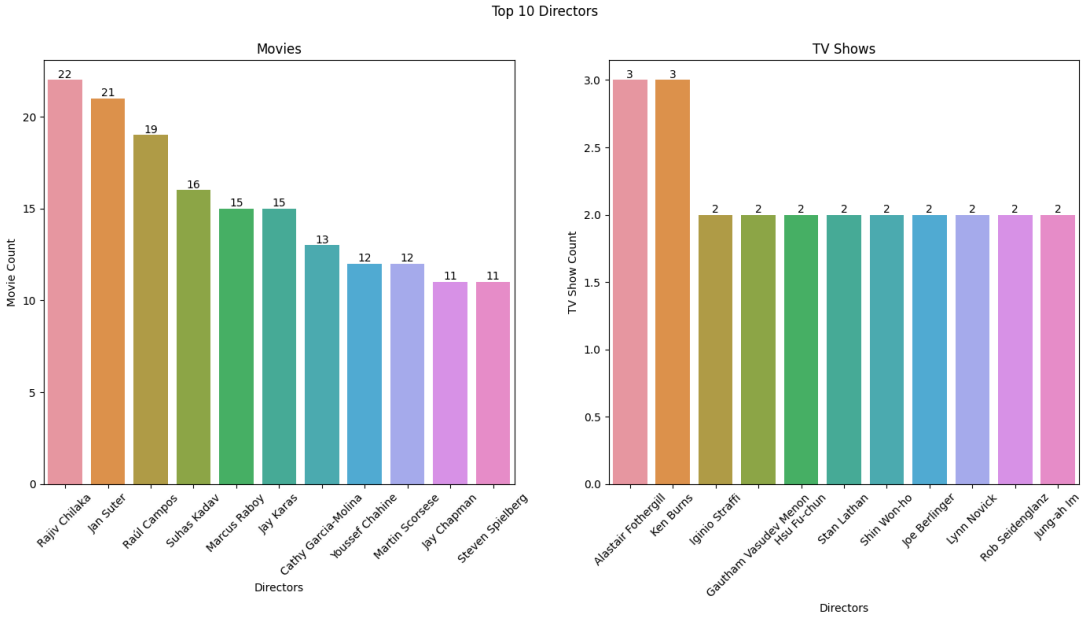
- The difference between the number of movies and Tv shows for all the countries is very high especially for India.
- This should be minimised by adding more TV Shows as TV Shows can keep a user engaged for 2-3 seasons rated than a 100 minute movie.
- TV Shows have also been very popular in recent times and are the new and demanded versions of a movie.

Lets see the top 10 Directors.

Here we need unnested data of directors only so we will drop duplicates on director, title group in df_movies and df_tv_show.

In [197]:

```
df_movies_temp = df_Movie.drop_duplicates(subset = ['director','title'])
df_tvs_temp = df_tv_show.drop_duplicates(subset = ['director','title'])

plt.figure(figsize = (17,7))
plt.suptitle('Top 10 Directors')

plt.subplot(1,2,1)
label = sns.countplot(data=df_movies_temp, x='director', order = df_movi
for i in label.containers:
    label.bar_label(i)
plt.title("Movies")
plt.xticks(rotation=45)
plt.xlabel('Directors')
plt.ylabel('Movie Count')

plt.subplot(1,2,2)
label = sns.countplot(data=df_tvs_temp, x='director', order = df_tvs_tem
for i in label.containers:
    label.bar_label(i)
plt.title("TV Shows")
plt.xticks(rotation=45)
plt.xlabel('Directors')
plt.ylabel('TV Show Count')

plt.show()
```



Rajiv Chilaka, Jan Suter, Raul Campos are the most active directors with 22, 21 and 19 movies

Whereas for TV Shows all the directos have directed around 2-3 shows only.

Lets see the top 10 Genres

Here we need unnested data of genres only

So we will drop duplicates on genre, title group in df_movies and df_tv_show.

In [198]:
```
df_Movie.loc[df_Movie[['genre','title']].duplicated()]
```

Out[198]:

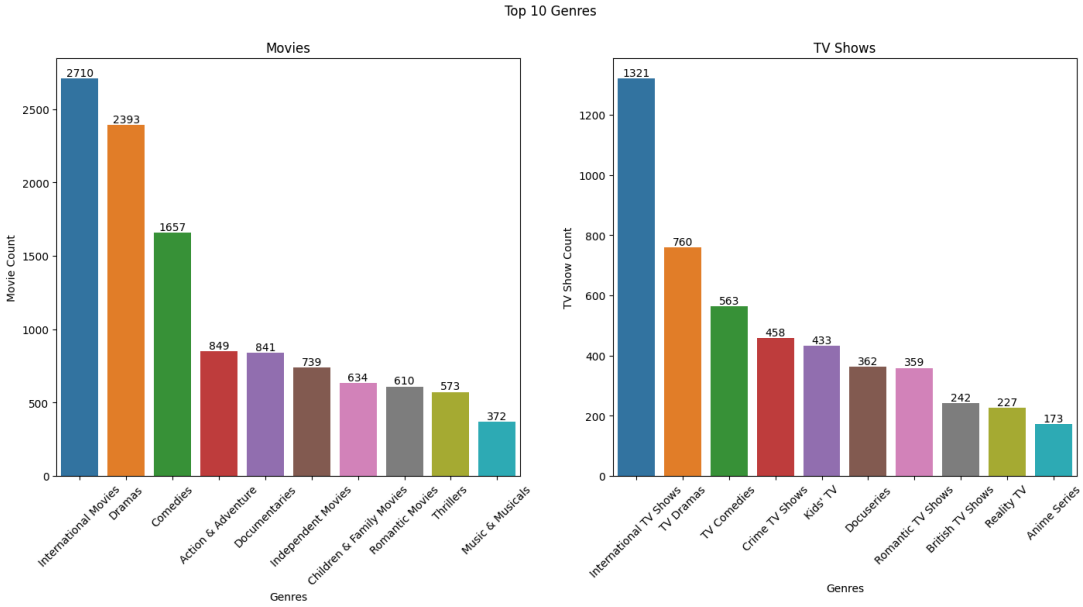
	type	title	date_added	release_year	rating	duration	description	day
158	Movie	My Little Pony: A New Generation	2021-09-24	2021	Kids	91	Equestria's divided. But a bright-eyed hero be...	24
159	Movie	My Little Pony: A New Generation	2021-09-24	2021	Kids	91	Equestria's divided. But a bright-eyed hero be...	24
160	Movie	My Little Pony: A New Generation	2021-09-24	2021	Kids	91	Equestria's divided. But a bright-eyed hero be...	24
161	Movie	My Little Pony: A New Generation	2021-09-24	2021	Kids	91	Equestria's divided. But a bright-eyed hero be...	24
162	Movie	My Little Pony: A New Generation	2021-09-24	2021	Kids	91	Equestria's divided. But a bright-eyed hero be...	24
...	...	...	...	...	...	...	...	...
199945	Movie	Zubaan	2019-03-02	2015	Teens	111	A scrappy but poor boy worms his way into a ty...	2
199946	Movie	Zubaan	2019-03-02	2015	Teens	111	A scrappy but poor boy worms his way into a ty...	2
199947	Movie	Zubaan	2019-03-02	2015	Teens	111	A scrappy but poor boy worms his way into a ty...	2
199948	Movie	Zubaan	2019-03-02	2015	Teens	111	A scrappy but poor boy worms his way into a ty...	2
199949	Movie	Zubaan	2019-03-02	2015	Teens	111	A scrappy but poor boy worms his way into a ty...	2

131304 rows × 16 columns

In [ ]:

In [199]:

```
df_movies_temp = df_Movie.drop_duplicates(subset = ['genre','title'])
df_tvs_temp = df_tv_show.drop_duplicates(subset = ['genre','title'])

plt.figure(figsize = (17,7))
plt.suptitle('Top 10 Genres')

plt.subplot(1,2,1)
label = sns.countplot(data= df_movies_temp, x='genre', order = df_movies
for i in label.containers:
    label.bar_label(i)
plt.title("Movies")
plt.xticks(rotation=45)
plt.xlabel('Genres')
plt.ylabel('Movie Count')

plt.subplot(1,2,2)
label = sns.countplot(data=df_tvs_temp, x='genre', order = df_tvs_temp['
for i in label.containers:
    label.bar_label(i)
plt.title("TV Shows")
plt.xticks(rotation=45)
plt.xlabel('Genres')
plt.ylabel('TV Show Count')

plt.show()
```



In [ ]:

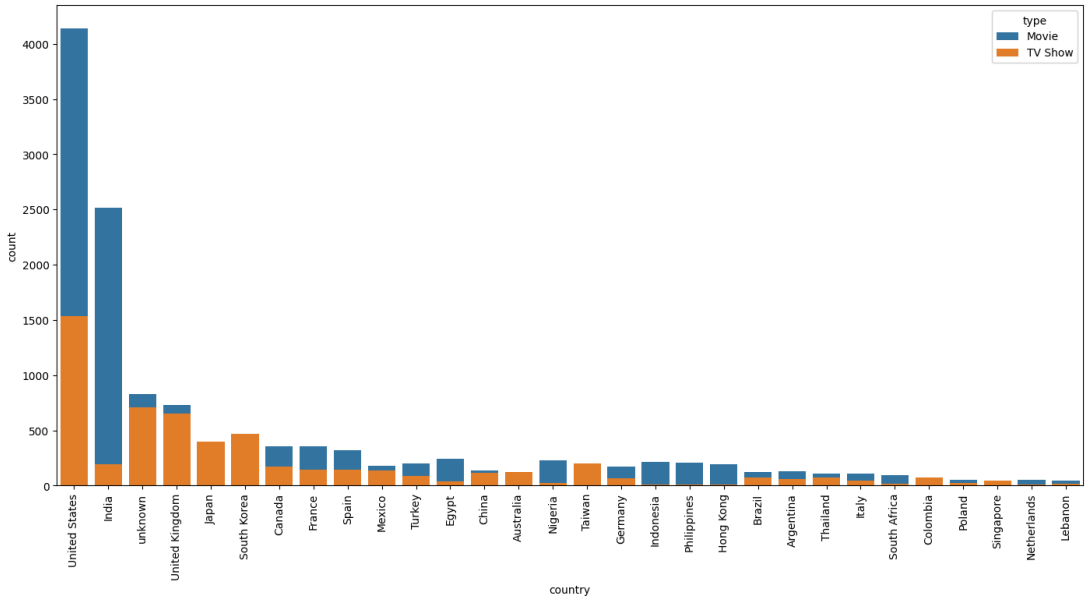
Inference

- From the above graph, it is inferred that most of the content fall under International followed by Drama and Comedy genre

Recommendations

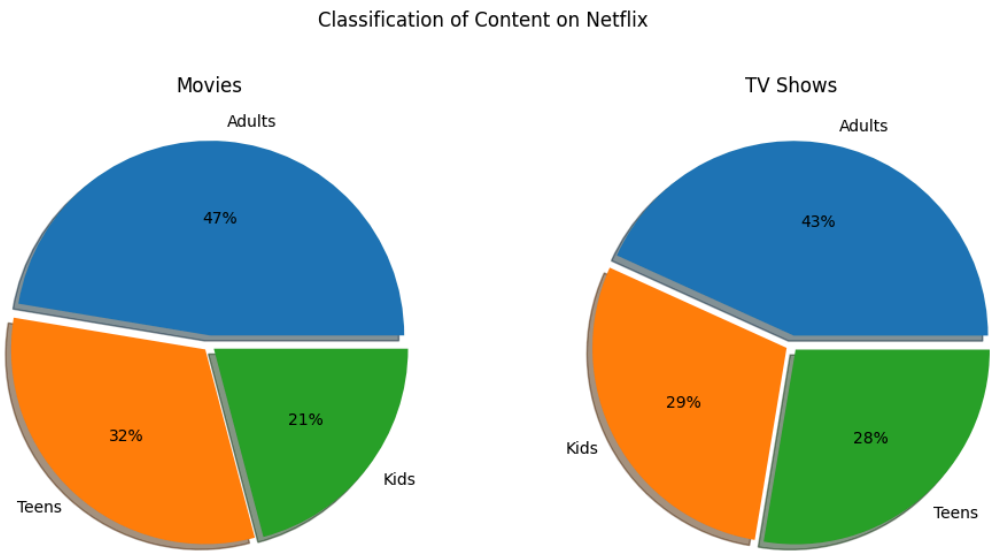
- The count of International genre is very high as compared to other genres for TV Shows,
- Netflix should try adding more content of different genres as well.

In [200]:
```
df_temp = df_new.drop_duplicates(subset = ['genre','title'])

plt.figure(figsize = (17,8))
plt.xticks(rotation=90)
sns.countplot(data = df_temp, x='country', hue = 'type', dodge = False,
plt.show()
```



Lets see the content available age group wise

In [201]:
```
df_movies_temp = df_Movie.drop_duplicates(subset = ['rating','title'])
df_tvs_temp = df_tv_show.drop_duplicates(subset = ['rating','title'])

plt.figure(figsize = (12,6))
plt.suptitle('Classification of Content on Netflix')

plt.subplot(1,2,1)
plt.pie(df_movies_temp['rating'].value_counts(), labels = df_movies_temp
plt.title('Movies')

plt.subplot(1,2,2)
plt.pie(df_tvs_temp['rating'].value_counts(), labels = df_tvs_temp['rati
plt.title('TV Shows')

plt.show()
```



Inference

- Most content on Netflix is for the adults followed by Teens and Kids.

Recommedation

- More content should be added for the teens so as to increase the viewership.

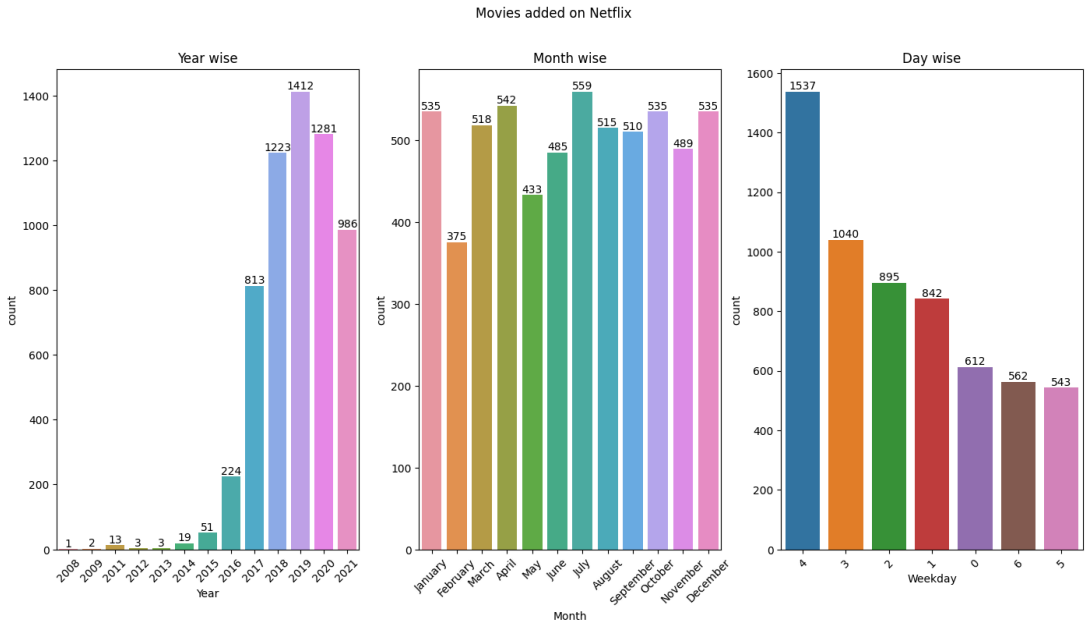
Now, lets see when does netflix add movies and tv shows the most

In [ ]:

In [202]:

```
df_movies_temp = df_Movie.drop_duplicates(subset = ['year','title'])
df_movies_temp['weekday']=pd.to_datetime(df_movies_temp['date_added']).d


mv_year = df_movies_temp['year'].value_counts()
mv_year.sort_index(inplace=True)

month_order = ['January', 'February', 'March', 'April', 'May', 'June', '
mv_month = df_movies_temp['month_name'].value_counts().loc[month_order]

day_order = ['Monday','Tuesday','Wednesday','Thursday','Friday','Saturda
mv_day = df_movies_temp['weekday'].value_counts()


plt.figure(figsize=(17,8))
plt.suptitle('Movies added on Netflix')

plt.subplot(1,3,1)
label = sns.countplot(data=df_movies_temp, x='year', order = mv_year.inc
for i in label.containers:
    label.bar_label(i)
plt.xticks(rotation=45)
plt.xlabel('Year')
plt.title('Year wise')

plt.subplot(1,3,2)
label = sns.countplot(data=df_movies_temp, x='month_name', order = mv_mo
for i in label.containers:
    label.bar_label(i)
plt.xticks(rotation=45)
plt.xlabel('Month')
plt.title('Month wise')

plt.subplot(1,3,3)
label = sns.countplot(data=df_movies_temp, x='weekday', order = mv_day.i
for i in label.containers:
    label.bar_label(i)
plt.xticks(rotation=45)
plt.xlabel('Weekday')
plt.title('Day wise')


plt.show()
```

```
C:\Users\mahes\AppData\Local\Temp\ipykernel_10844\1773502885.py:2: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
  df_movies_temp['weekday']=pd.to_datetime(df_movies_temp['date_adde
d']).dt.weekday
```



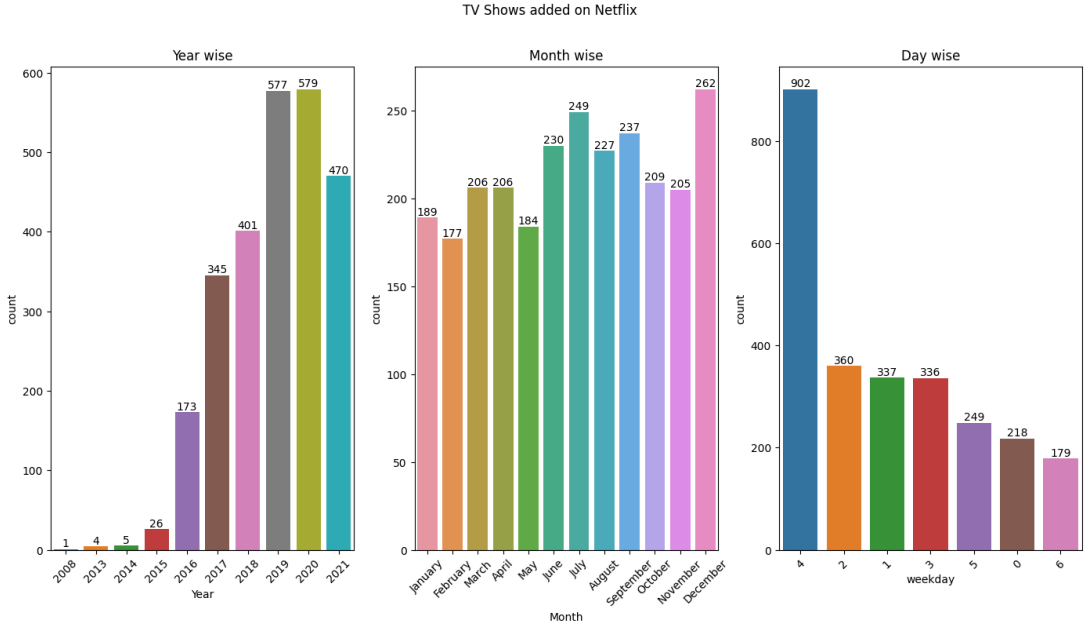
Inference

- As per the data, most movies were released in the year 2019 and 2020.
- They were released the most in July.
- We can also see that most movies were released on Friday.

In [203]:

```
df_tvs_temp = df_tv_show.drop_duplicates(subset = ['year','title'])
df_tv_show['weekday']=pd.to_datetime(df_tv_show['date_added']).dt.weekda

tv_year = df_tvs_temp['year'].value_counts()
tv_year.sort_index(inplace=True)

month_order = ['January', 'February', 'March', 'April', 'May', 'June', '
tv_month = df_tvs_temp['month_name'].value_counts().loc[month_order]

day_order = ['Monday','Tuesday','Wednesday','Thursday','Friday','Saturda
tv_day = df_tvs_temp['weekday'].value_counts()


plt.figure(figsize=(17,8))
plt.suptitle('TV Shows added on Netflix')

plt.subplot(1,3,1)
label = sns.countplot(data=df_tvs_temp, x='year', order = tv_year.index)
for i in label.containers:
    label.bar_label(i)
plt.xticks(rotation=45)
plt.xlabel('Year')
plt.title('Year wise')

plt.subplot(1,3,2)
label = sns.countplot(data=df_tvs_temp, x='month_name', order = tv_month
for i in label.containers:
    label.bar_label(i)
plt.xticks(rotation=45)
plt.xlabel('Month')
plt.title('Month wise')

plt.subplot(1,3,3)
label = sns.countplot(data=df_tvs_temp, x='weekday', order = tv_day.inde
for i in label.containers:
    label.bar_label(i)
plt.xticks(rotation=45)
plt.xlabel('weekday')
plt.title('Day wise')


plt.show()
```

```
C:\Users\mahes\AppData\Local\Temp\ipykernel_10844\3198309574.py:2: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
  df_tv_show['weekday']=pd.to_datetime(df_tv_show['date_added']).dt.we
ekday
```



Inference

- As per the data, most TV Shows were released in the year 2019 and 2020.
- They were released the most in December followed by July and September.
- We can also see that most shows were released on Friday.

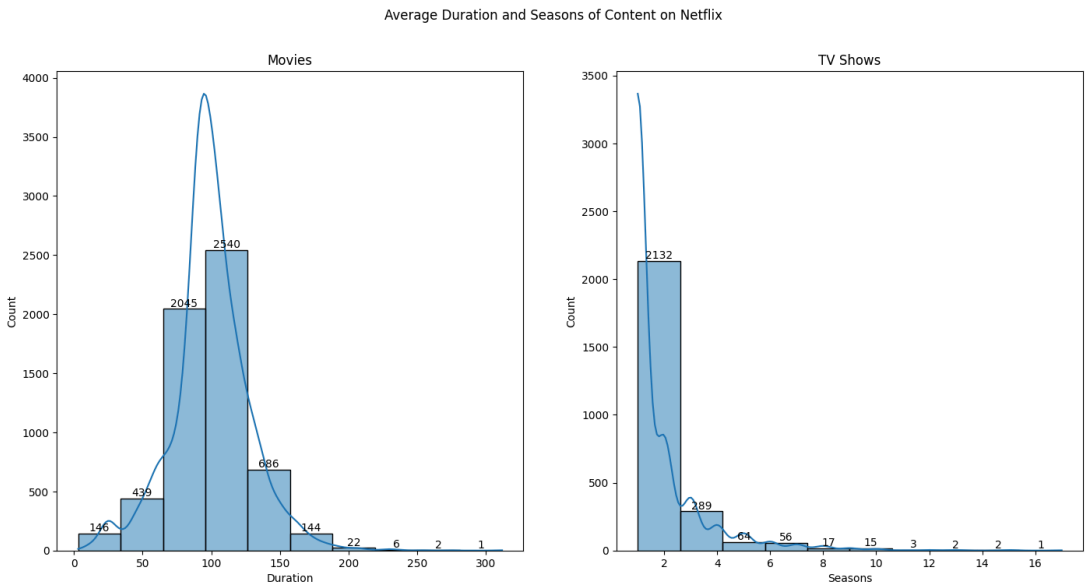
Recommendation

- To increase viewership in India, more shows should be released during the vacation or festival seasons which are around April-May and between October and December.

Lets see the average duration of a movie and a average season of TV Show

In [204]:

```
df_movies_temp = df_Movie.drop_duplicates(subset = ['duration','title'])
df_tvs_temp = df_tv_show.drop_duplicates(subset = ['seasons','title'])


plt.figure(figsize=(17,8))
plt.suptitle('Average Duration and Seasons of Content on Netflix')

plt.subplot(1,2,1)
label = sns.histplot(df_movies_temp['duration'].astype(int), bins=10, kd
for i in label.containers:
    label.bar_label(i)
plt.xlabel('Duration')
plt.title('Movies')

plt.subplot(1,2,2)
label = sns.histplot(df_tvs_temp['seasons'].astype(int), bins=10, kde =
for i in label.containers:
    label.bar_label(i)
plt.xlabel('Seasons')
plt.title('TV Shows')

plt.show()
```



Inference

- Most (Around 4500) movies have duration between 65 and 125 minutes.
- Most(Around 2200) TV Shows have been produced for around 2 seasons.

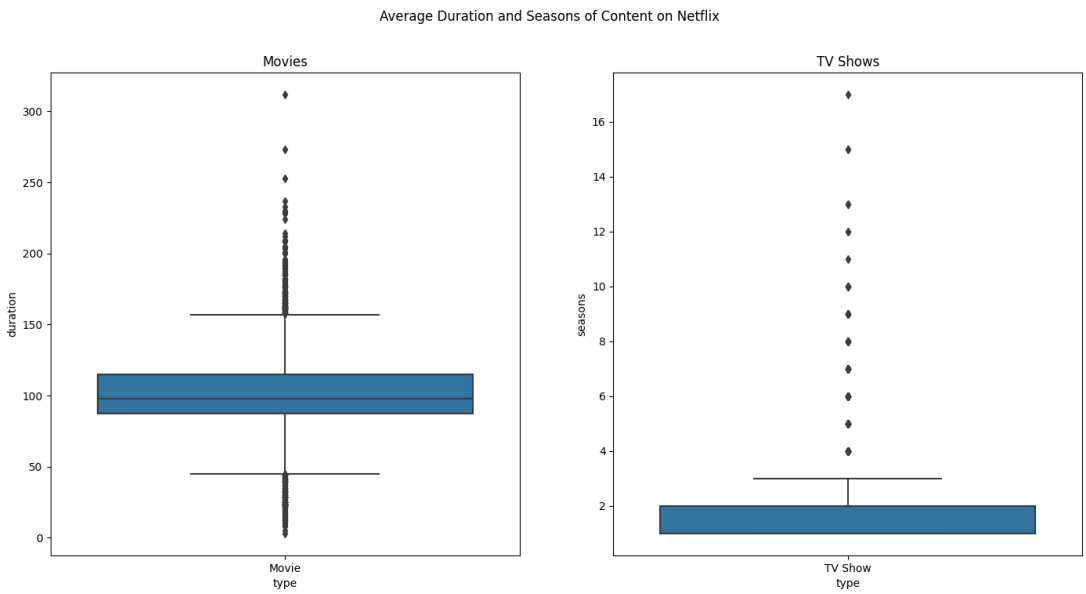
Recommendations

- Duration must be kep between 65 and 125 minutes for a movie.
- TV Show should have around 2-3 seasons

Lets see the range of most movie's duration and tv show's seasons lie.

In [205]:
```
df_movies_temp['duration'] = df_movies_temp['duration'].astype(int)
df_tvs_temp['seasons'] = df_tvs_temp['seasons'].astype(int)

plt.figure(figsize=(17,8))
plt.suptitle('Average Duration and Seasons of Content on Netflix')

plt.subplot(1,2,1)
sns.boxplot(df_movies_temp, x='type', y='duration')
plt.title('Movies')

plt.subplot(1,2,2)
sns.boxplot(df_tvs_temp, x='type', y='seasons')
plt.title('TV Shows')

plt.show()
```

```
C:\Users\mahes\AppData\Local\Temp\ipykernel_10844\785406378.py:1: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
  df_movies_temp['duration'] = df_movies_temp['duration'].astype(int)
C:\Users\mahes\AppData\Local\Temp\ipykernel_10844\785406378.py:2: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
  df_tvs_temp['seasons'] = df_tvs_temp['seasons'].astype(int)
```



Inference

- The median duration of a movie on Netflix is around 100 minutes, whereas the median season of a TV Show is 1 season.

- Most movies have duration length between 50 and 160 minutes.
- Most TV shows have either 1,2 or 3 seasons.

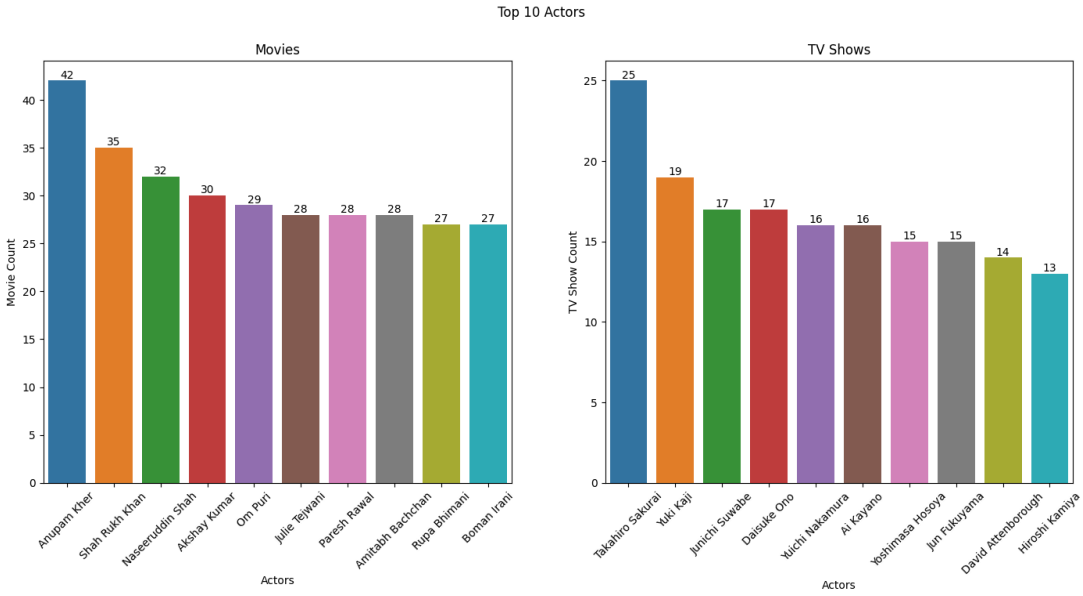
Recommendation

- In order to keep the audience engaged, it is recommended to keep the movie length upto 160 minutes and upto 3 seasons for a TV Show.

Lets see most active actors for movies and tv shows seperately.

In [206]:

```
df_movies_temp = df_Movie.drop_duplicates(subset = ['cast','title'])
df_tvs_temp = df_tv_show.drop_duplicates(subset = ['cast','title'])

plt.figure(figsize = (17,7))
plt.suptitle('Top 10 Actors')

plt.subplot(1,2,1)
label = sns.countplot(data=df_movies_temp, x='cast', order = df_movies_t
for i in label.containers:
    label.bar_label(i)
plt.title("Movies")
plt.xticks(rotation=45)
plt.xlabel('Actors')
plt.ylabel('Movie Count')

plt.subplot(1,2,2)
label = sns.countplot(data=df_tvs_temp, x='cast', order = df_tvs_temp['c
for i in label.containers:
    label.bar_label(i)
plt.title("TV Shows")
plt.xticks(rotation=45)
plt.xlabel('Actors')
plt.ylabel('TV Show Count')

plt.show()
```



Inference

- We can clearly see that Anupam Kher has done the most amount of movies followed by Shah Rukh Khan and Naseeruddin Shah.

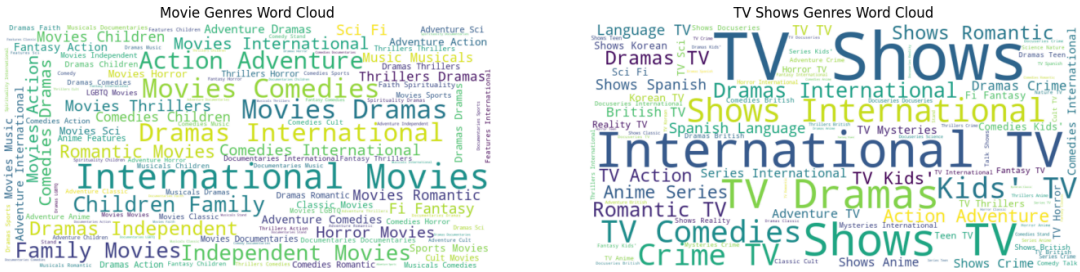
- The top 10 actors for movies are all Indians, whereas the top 10 actors for TV Shows are from different countries.
- 8 out of the top 10 actors for movies are males and 2 are females.

Recommendations

- TV Shows have very few Indian actors as compared to other actors.
- To increase viewership of TV Shows in India, Netflix should add more TV Shows with Indian actors in it.

In [207]:

```
genre_text = " ".join(df_Movie["genre"])
wordcloud = WordCloud(width=800, height=400, background_color="white").g

plt.figure(figsize=(17, 10))

plt.subplot(1,2,1)
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Movie Genres Word Cloud")

genre_text = " ".join(df_tv_show["genre"])
wordcloud = WordCloud(width=800, height=400, background_color="white").g

plt.subplot(1,2,2)
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("TV Shows Genres Word Cloud")

plt.show()
```



Lets see the correlation in our data.

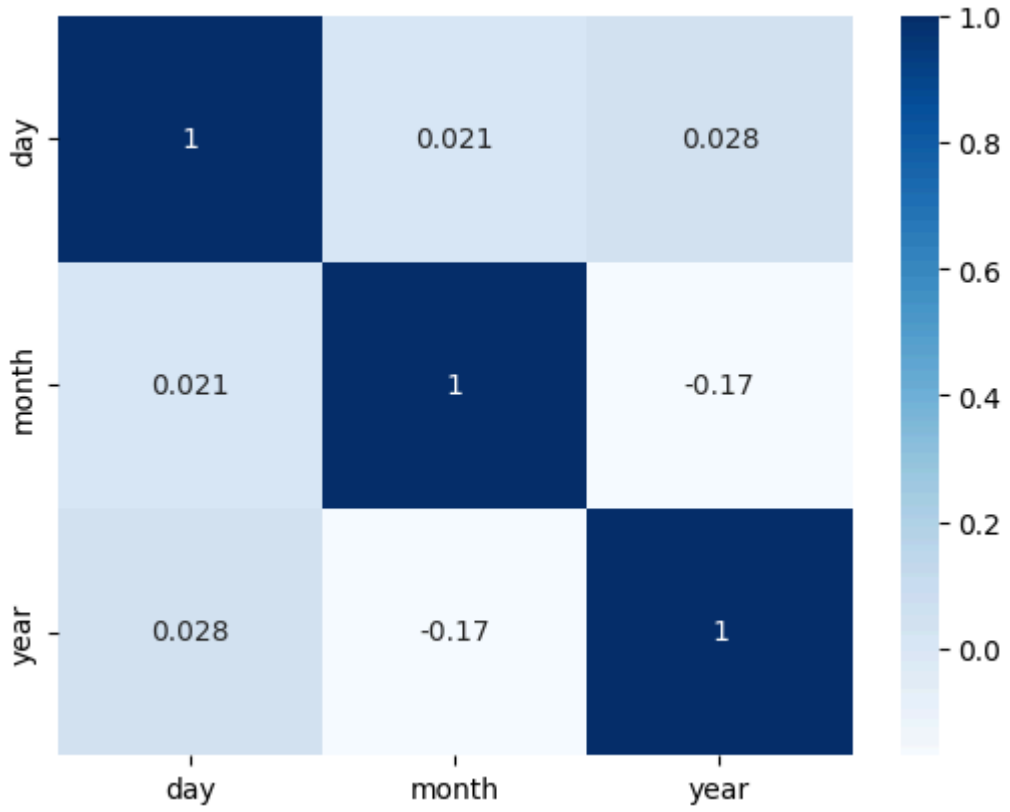
In [208]:

```
df_new.columns
```

Out[208]:

```
Index(['type', 'title', 'date_added', 'release_year', 'rating', 'durat
ion',
       'description', 'day', 'month', 'month_name', 'year', 'day_nam
e',
       'country', 'director', 'cast', 'genre'],
      dtype='object')
```

In [209]:
```
sns.heatmap(df_new[['day','month','year']].corr(), cmap = 'Blues', annot
plt.show()
```



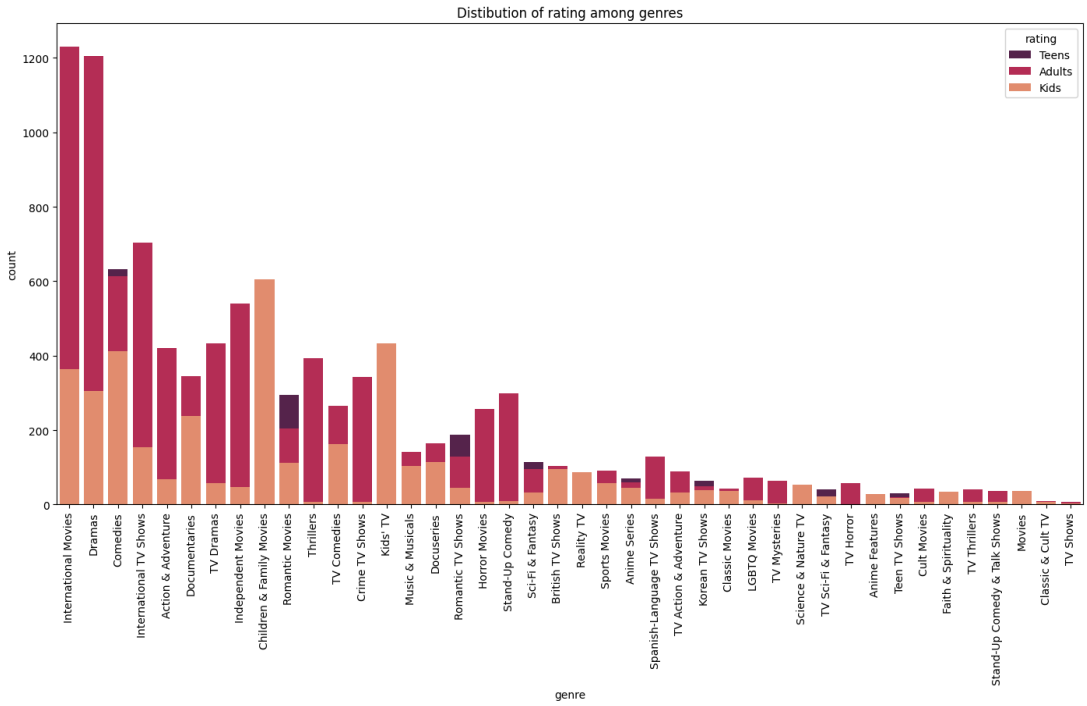
Inference

- The heatmap shows the relation between numerical values of the data.
- Heatmap in our case gives us no interpretation as the only numerical values which we have in our data are the day, month and year of the movie or tv show.

Lets see distribution of genre and rating among themselves.

In [210]:

```
df_temp = df_new.drop_duplicates(subset = ['genre','title'])

plt.figure(figsize=(17,8))
plt.title('Distibution of rating among genres')
sns.countplot(data=df_temp, x='genre', hue = 'rating', dodge=False, orde
plt.xticks(rotation=90)
plt.show()
```



Inference

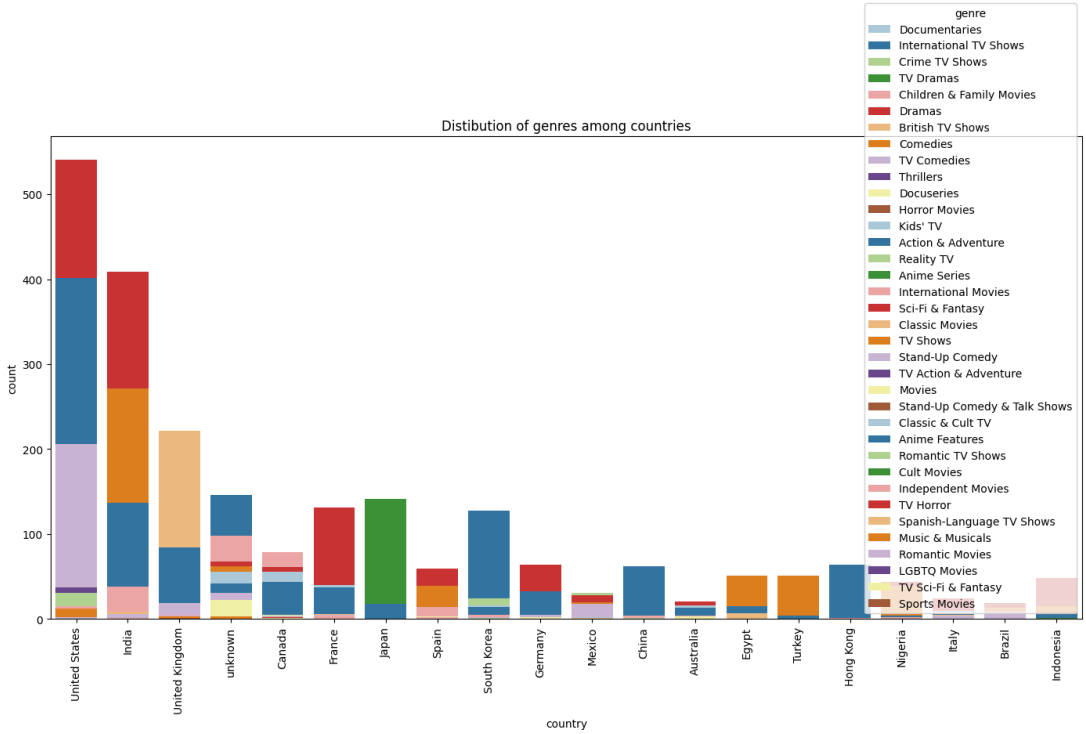
- We can clearly see that most content(both movies and tv shows) are made for adults.

Recommendations

- To increase more users, netflix should diversify the content for teens and kids as well.

Lets see distribution of genres across countries.

In [211]:
```
df_temp = df_new.drop_duplicates(subset = ['country','title'])

plt.figure(figsize=(17,8))
plt.title('Distibution of genres among countries')
sns.countplot(data=df_temp, x='country', hue = 'genre', dodge=False, orc
plt.xticks(rotation=90)
plt.show()
```



Inference

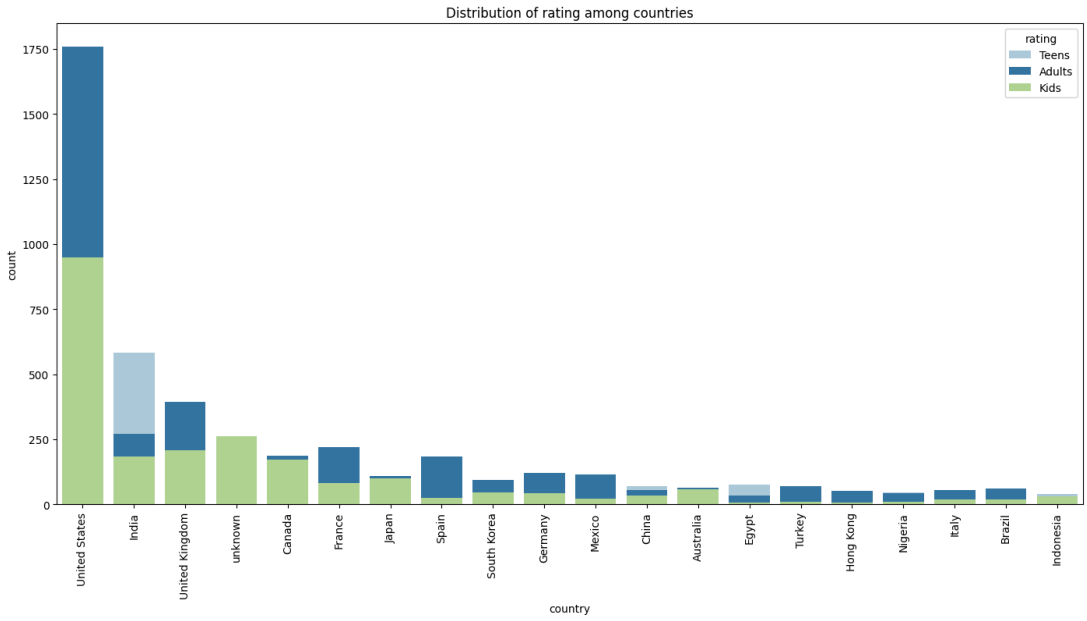
- We can observe that International(Blue color) and Drama(Red color) are the most type of content available on Netflix.

Recommendations

- Netflix should produce more different genres also in order to attract and increase viewership.

Lets see countrywise content rating classification.

In [212]:
```
df_temp = df_new.drop_duplicates(subset = ['country','title'])

plt.figure(figsize=(17,8))
plt.title('Distribution of rating among countries')
sns.countplot(data=df_temp, x='country', hue = 'rating', dodge=False, or
plt.xticks(rotation=90)
plt.show()
```



Inference

- We can see the in US and UK there is no content specially made for the teens, whereas in India we can see the that most of the content is made for teens.

Recommendation

- More content for teens should be added to atract newer audiences.

EXPLORATORY DATA ANALYSIS IS ALMOST FINISHED PLEASE SUGGEST ANY DEFICIENCIES AS THIS IS MY FIRST PROJECT

In [ ]:

In [ ]: