In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import matplotlib.pyplot as plt
import ipywidgets as w
from IPython.display import display
```

In [ ]:

```
#Importing the dataset
```

In [2]:

```
df_p = pd.read_csv('C:\\Users\\prafu\\OneDrive\\Desktop\\netflix.csv')
df_p
```

Out[2]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...
...	...	...	...	...	...	...	...	...	...	...	...	...
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	2007	R	158 min	Cult Movies, Dramas, Thrillers	A political cartoonist, a crime reporter and a...
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	2018	TV-Y7	2 Seasons	Kids' TV, Korean TV Shows, TV Comedies	While living alone in a spooky town, a young g...
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	2009	R	88 min	Comedies, Horror Movies	Looking to survive in a world taken over by zo...
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	2006	PG	88 min	Children & Family Movies, Comedies	Dragged from civilian life, a former superhero...
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	2015	TV-14	111 min	Dramas, International Movies, Music & Musicals	A scrappy but poor boy worms his way into a ty...

8807 rows × 12 columns

In [3]:

```
#Having the info of the dataset
df_p.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64 
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

In [5]:

```
#statistical analysis
df_p.describe()
```

Out[5]:

	release_year
count	8807.000000
mean	2014.180198
std	8.819312
min	1925.000000
25%	2013.000000
50%	2017.000000
75%	2019.000000
max	2021.000000

In [4]:

```
#checking for Null values
df_p.isna().any()
```

Out[4]:

```
show_id         False
type            False
title           False
director         True
cast             True
country          True
date_added       True
release_year    False
rating           True
duration         True
listed_in       False
description     False
dtype: bool
```

In [6]:

```
#Displaying the first five rows in dataset
#Displaying the first five rows to analyse and identify, the data and datatype in different columns,
#As we see in current dataset, there ara nested datas present in cast, director,country,
#and listed_in columns, and there are many null values present. We need to breakdown each point to anlayse the data.
df_p.head()
```

Out[6]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...

In [9]:

```
#Checking the count of null values column wise
df_p.isna().sum()
```

Out[9]:

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

In [10]:

```
#Total number of Null values
df_p.isna().sum().sum()
```

Out[10]:

```
4307
```

In [ ]:

```
# It can be seen that we have a total of 4307 null values, out of which 2634 belongs to director column,
#825 belong to cast, 831 to country, 10 to date_added, 4 to rating and 3 to duration.
```

In [11]:

```
#Dealing with null values
#as we see, there are'nt any rows, where more than 40% data is missing, so dropping the row is not feasible in this case

#For country column, mode value is used for filling the missing data
#For director and cast column 'not known' and 'anonymous' values are used respectively for filling the director and cast co
#For Rating ,we input value as "unavailable" for null values
#For duration, we input "0" for duration column
```

In [13]:

```
df_p['country'] = df_p['country'].fillna(df_p['country'].mode()[0])
df_p['director'] = df_p['director'].fillna(value="not known")
df_p['cast'] = df_p['cast'].fillna(value="Anonymous")
df_p['date_added'] = df_p['date_added'].fillna(df_p['date_added'].mode()[0])
df_p['rating'] = df_p['rating'].fillna(value="unavailable")
df_p['duration'] = df_p['duration'].fillna(value="0")
```

In [14]:

```
#Splitting duration and adding only numbers in new column

df_p['new_duration']=df_p['duration'].str.split(' ').str[0]
```

In [ ]:

```
#Casting nested datas to un-nested data's
#As there are nested date's in cast,country,genre,director, we need to unnest and merge to single dataframe.
```

In [15]:

```
constraint=df_p['director'].apply(lambda x: str(x).split(', ')).tolist()
df_new=pd.DataFrame(constraint,index=df_p['title'])
df_new=df_new.stack()
df_director=pd.DataFrame(df_new)
df_director.reset_index(inplace=True)
df_director=df_director[['title',0]]
df_director.rename(columns={0:'director'})

constraint=df_p['cast'].apply(lambda x: str(x).split(', ')).tolist()
df_new=pd.DataFrame(constraint,index=df_p['title'])
df_new=df_new.stack()
df_cast=pd.DataFrame(df_new)
df_cast.reset_index(inplace=True)
df_cast=df_cast[['title',0]]
df_cast.rename(columns={0:'cast'})


constraint=df_p['country'].apply(lambda x: str(x).split(', ')).tolist()
df_new=pd.DataFrame(constraint,index=df_p['title'])
df_new=df_new.stack()
df_country=pd.DataFrame(df_new)
df_country.reset_index(inplace=True)
df_country=df_country[['title',0]]
df_country.rename(columns={0:'country'})


constraint=df_p['listed_in'].apply(lambda x: str(x).split(', ')).tolist()
df_new=pd.DataFrame(constraint,index=df_p['title'])
df_new=df_new.stack()
df_listedin=pd.DataFrame(df_new)
df_listedin.reset_index(inplace=True)
df_listedin=df_listedin[['title',0]]
df_listedin.rename(columns={0:'genre'})
```

Out[15]:

	title	genre
0	Dick Johnson Is Dead	Documentaries
1	Blood & Water	International TV Shows
2	Blood & Water	TV Dramas
3	Blood & Water	TV Mysteries
4	Ganglands	Crime TV Shows
...	...	...
19318	Zoom	Children & Family Movies
19319	Zoom	Comedies
19320	Zubaan	Dramas
19321	Zubaan	International Movies
19322	Zubaan	Music & Musicals

19323 rows × 2 columns

In [16]:

```
#Merging multiple dataframes to single
x= df_director.merge(df_cast,left_on="title",right_on="title",how="left")
x=x.rename(columns={'0_x':'director','0_y':'cast'})

y= x.merge(df_country,left_on="title",right_on="title",how="left")
y=y.rename(columns={0:'country'})

z= y.merge(df_listedin,left_on="title",right_on="title",how="left")
z=z.rename(columns={0:'genre'})

#Dropping columns in original dataset
df_p=df_p.drop(['director', 'cast','country','listed_in'], axis=1)

#Merge columns to Single dataset column

df= z.merge(df_p,left_on="title",right_on="title",how="left")
```

In [17]:

```
df_p
```

Out[17]:

	show_id	type	title	date_added	release_year	rating	duration	description	new_duration
0	s1	Movie	Dick Johnson Is Dead	September 25, 2021	2020	PG-13	90 min	As her father nears the end of his life, filmm...	90
1	s2	TV Show	Blood & Water	September 24, 2021	2021	TV-MA	2 Seasons	After crossing paths at a party, a Cape Town t...	2
2	s3	TV Show	Ganglands	September 24, 2021	2021	TV-MA	1 Season	To protect his family from a powerful drug lor...	1
3	s4	TV Show	Jailbirds New Orleans	September 24, 2021	2021	TV-MA	1 Season	Feuds, flirtations and toilet talk go down amo...	1
4	s5	TV Show	Kota Factory	September 24, 2021	2021	TV-MA	2 Seasons	In a city of coaching centers known to train I...	2
...	...	...	...	...	...	...	...	...	...
8802	s8803	Movie	Zodiac	November 20, 2019	2007	R	158 min	A political cartoonist, a crime reporter and a...	158
8803	s8804	TV Show	Zombie Dumb	July 1, 2019	2018	TV-Y7	2 Seasons	While living alone in a spooky town, a young g...	2
8804	s8805	Movie	Zombieland	November 1, 2019	2009	R	88 min	Looking to survive in a world taken over by zo...	88
8805	s8806	Movie	Zoom	January 11, 2020	2006	PG	88 min	Dragged from civilian life, a former superhero...	88
8806	s8807	Movie	Zubaan	March 2, 2019	2015	TV-14	111 min	A scrappy but poor boy worms his way into a ty...	111

8807 rows × 9 columns

In [18]:

```
#Tackling with the unneccesary spaces in the date_added
df['date_added'] = df['date_added'].str.replace(" ","")
```

In [21]:

```
#Checking the new date_added format
df['date_added']
```

Out[21]:

```
0         September25,2021
1         September24,2021
2         September24,2021
3         September24,2021
4         September24,2021
               ...
201986         March2,2019
201987         March2,2019
201988         March2,2019
201989         March2,2019
201990         March2,2019
Name: date_added, Length: 201991, dtype: object
```

In [39]:

```
#Date_added ,changing the type to date
df['date'] = pd.to_datetime(df['date_added'],format="%B%d,%Y")
df['year'] = df['date'].apply(lambda datetime: datetime.year)
df['month'] = df['date'].apply(lambda datetime: datetime.month)
```

In [23]:

```
df['date']
```

Out[23]:

```
0        2021-09-25
1        2021-09-24
2        2021-09-24
3        2021-09-24
4        2021-09-24
            ...
201986   2019-03-02
201987   2019-03-02
201988   2019-03-02
201989   2019-03-02
201990   2019-03-02
Name: date, Length: 201991, dtype: datetime64[ns]
```

In [28]:

```
#Non- Graphical Analysis
```

In [33]:

```
#1) Total Content available in Netflix
```

In [34]:

```
df['title'].drop_duplicates(keep='last').value_counts().value_counts()[1]
```

Out[34]:

```
8807
```

In [35]:

```
#2) Total Content released in Summer holidays
#To predict ,whether summer holidays is the best time to release movie. In this prediction , summer month is assumed as May
```

In [44]:

```
mdm=df[df['type']=='Movie'][['title','month']]
mdm=mdm.drop_duplicates(keep='last')
mdt=df[df['type']=='TV Show'][['title','month']]
mdt=mdt.drop_duplicates(keep='last')
mdm=mdm[mdm['month']==5].value_counts().value_counts()[1]
mdt=mdt[mdt['month']==5].value_counts().value_counts()[1]
```

In [37]:

```
#Total movies released in May month
```

In [45]:

```
mdm
```

Out[45]:

```
439
```

In [46]:

```
#Total Tv shows released in May month
```

In [47]:

```
mdt
```

Out[47]:

```
193
```

In [48]:

```
#DATA VISUALIZATION
```

In [49]:

```
# For Exporting graphs while downloading as PDF
import plotly.io as pio
pio.renderers.default = "notebook+pdf"  # Renderer for Notebook and HTML exports + Renderer for PDF exports

import plotly.offline as pyo
pyo.init_notebook_mode()
```

In [50]:

```
#Category wise content
md=df[df['type']=='Movie']['title']
md=md.drop_duplicates(keep='last').value_counts()
td=df[df['type']=='TV Show']['title']
td=td.drop_duplicates(keep='last').value_counts()
```

In [51]:

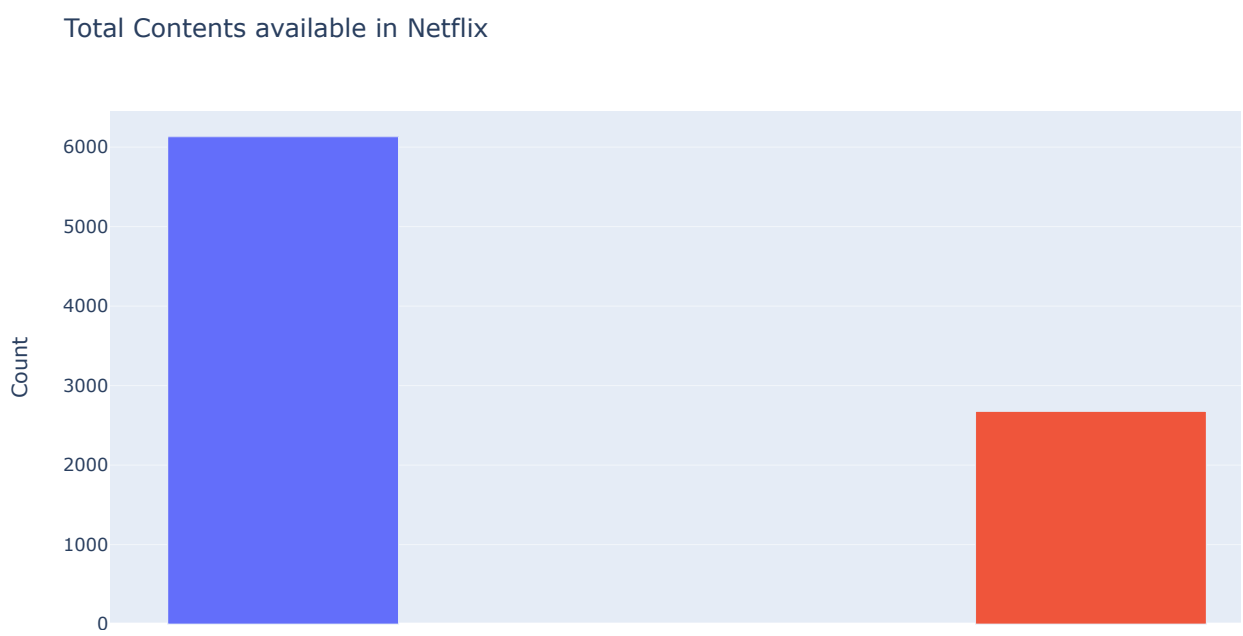
```
data_dict1 = {'Count':[md.value_counts()[1], td.value_counts()[1]], 'type': ['Movie','TV Show']}
```

In [52]:

```
df_b = pd.DataFrame(data=data_dict1, columns=['Count','type'])
```

In [53]:

```
px.bar(data_frame=df_b, x="type", y="Count", color="type", barmode="group",title="Total Contents available in Netflix")
```



In [54]:

```
#Top Countries contributing to Netflix
```

In [55]:

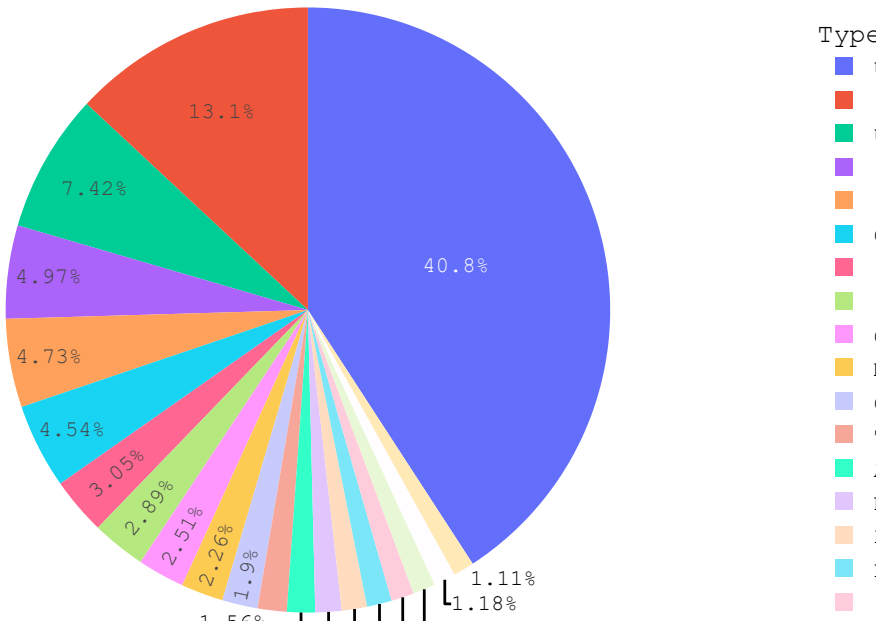
```
data_dict1 = {'country': df.groupby('country').size().sort_values(ascending=False)[:20].index,
              'Number of content': df.groupby('country').size().sort_values(ascending=False)[:20].values
              }
```

In [56]:

```
df_P = pd.DataFrame(data=data_dict1, columns=['country', 'Number of content'])
```

In [57]:

```
fig = px.pie(df_P, values='Number of content', names='country',title="Top 20 Contries Contributing to Netflix")
fig.update_layout(xaxis_title="Year",
                  yaxis_title="Number of content",
                  legend_title='Type of Content',
                  height=600,
    title=dict(
        text='<b>Top 20 Contries Contributing to Netflix</b>',
        x=0.25,
        y=0.96,
        font=dict(
            family="Arial",
            size=25,
            color='#000000'
        )
    ),
    font=dict(
        family="Courier New, Monospace",
        size=15,
        color='#000000'
    )
)
fig.show()
```

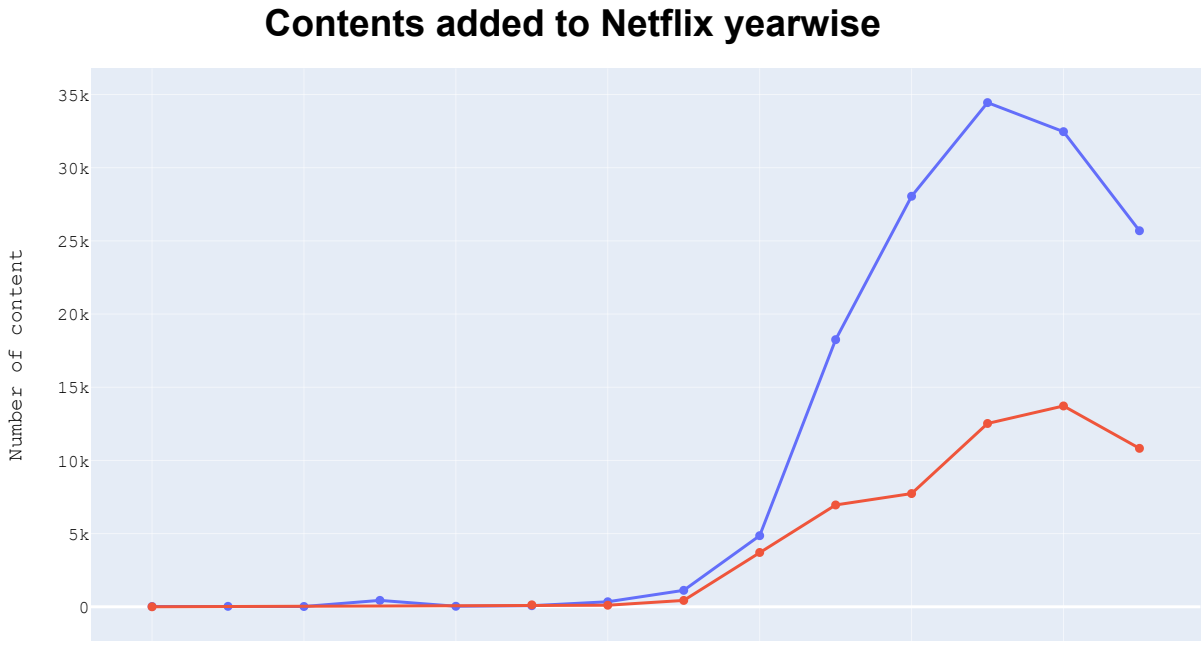


In [58]:

```
##Yearwise Content added to netflix
```

In [59]:

```
type_of_contents=df.groupby('type').size().index.tolist()
df6=df.loc[df['type'].isin(type_of_contents)]
df_6_upd=df6.groupby('year')['type'].value_counts().reset_index(name='counts')
fig = px.line(df_6_upd, x="year", y="counts", color='type',
              markers=True)
fig.update_layout(xaxis_title="Year",
                  yaxis_title="Number of content",
                  legend_title='Type of Content',
    title=dict(
        text='<b>Contents added to Netflix yearwise</b>',
        x=0.20,
        y=0.96,
        font=dict(
            family="Arial",
            size=25,
            color='#000000'
        )
    ),
    font=dict(
        family="Courier New, Monospace",
        size=12,
        color='#000000'
    )
)
```

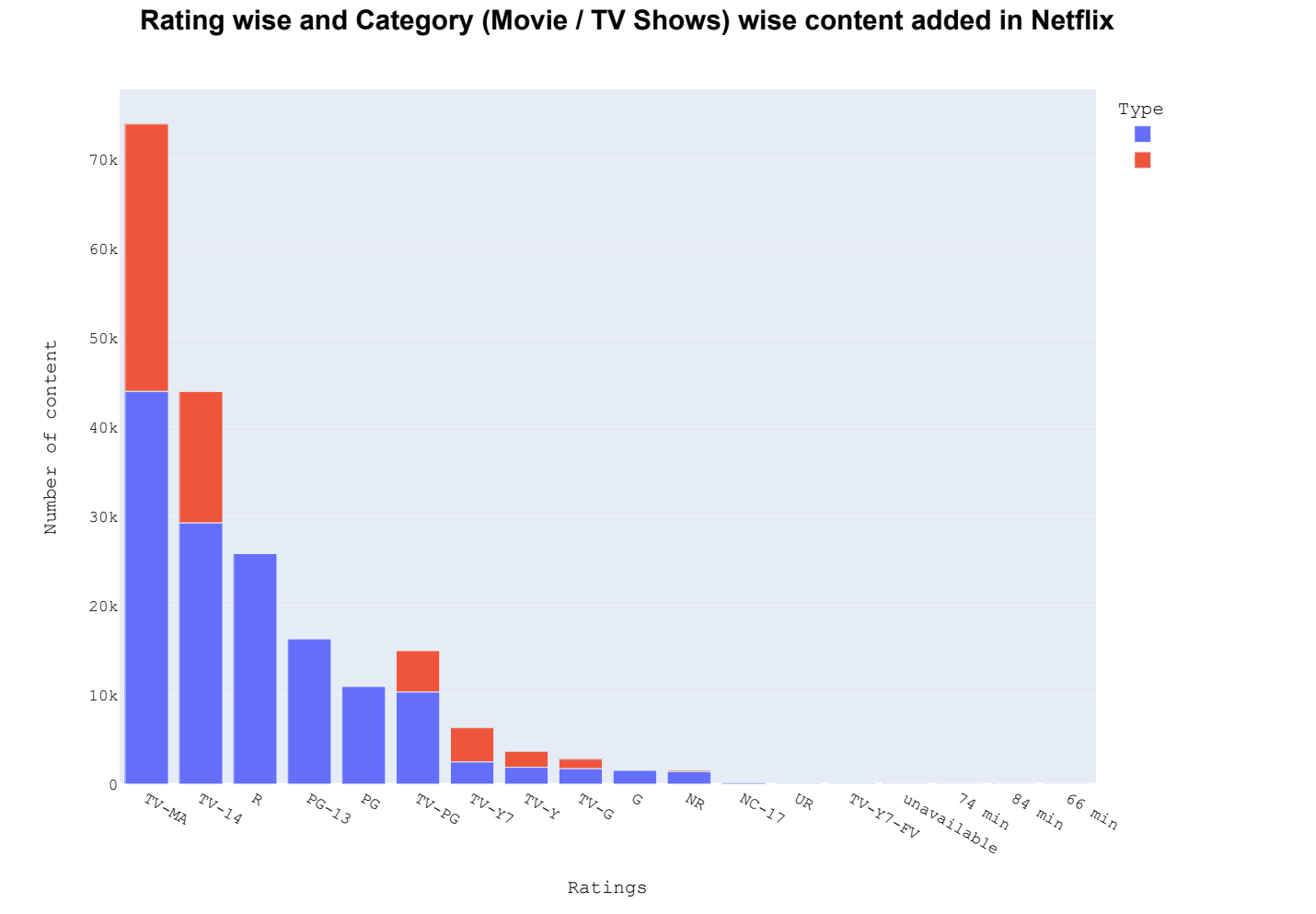


In [60]:

```
#contents added in Category wise and  Rating wise
```

In [61]:

```
type_of_rating=df.groupby('rating').size().index.tolist()
df7=df.loc[df['rating'].isin(type_of_rating)]
df7=df7.groupby(['type'])['rating'].value_counts().reset_index(name='counts')
data_dict1 = {'Ratings': df7['rating'],
              'Number of content': df7['counts'],'type': df7['type']
             }
df_R = pd.DataFrame(data=data_dict1, columns=['Ratings', 'Number of content','type'])
fig = px.bar(df_R, x="Ratings",
                   y="Number of content",
                   title="Rating wise and Category (Movie / TV Shows) wise content added in Netflix",
                   color='type')
fig.update_layout(autosize=False, width=950, height=700, xaxis_title="Ratings",
                  yaxis_title="Number of content",
                  legend_title='Type of Content',
    title=dict(
        text='<b>Rating wise and Category (Movie / TV Shows) wise content added in Netflix</b>',
        x=0.10,
        y=0.94,
        font=dict(
            family="Arial",
            size=20,
            color='#000000'
        )
    ),
    font=dict(
        family="Courier New, Monospace",
        size=12,
        color='#000000'
    )
)


fig.show()
```

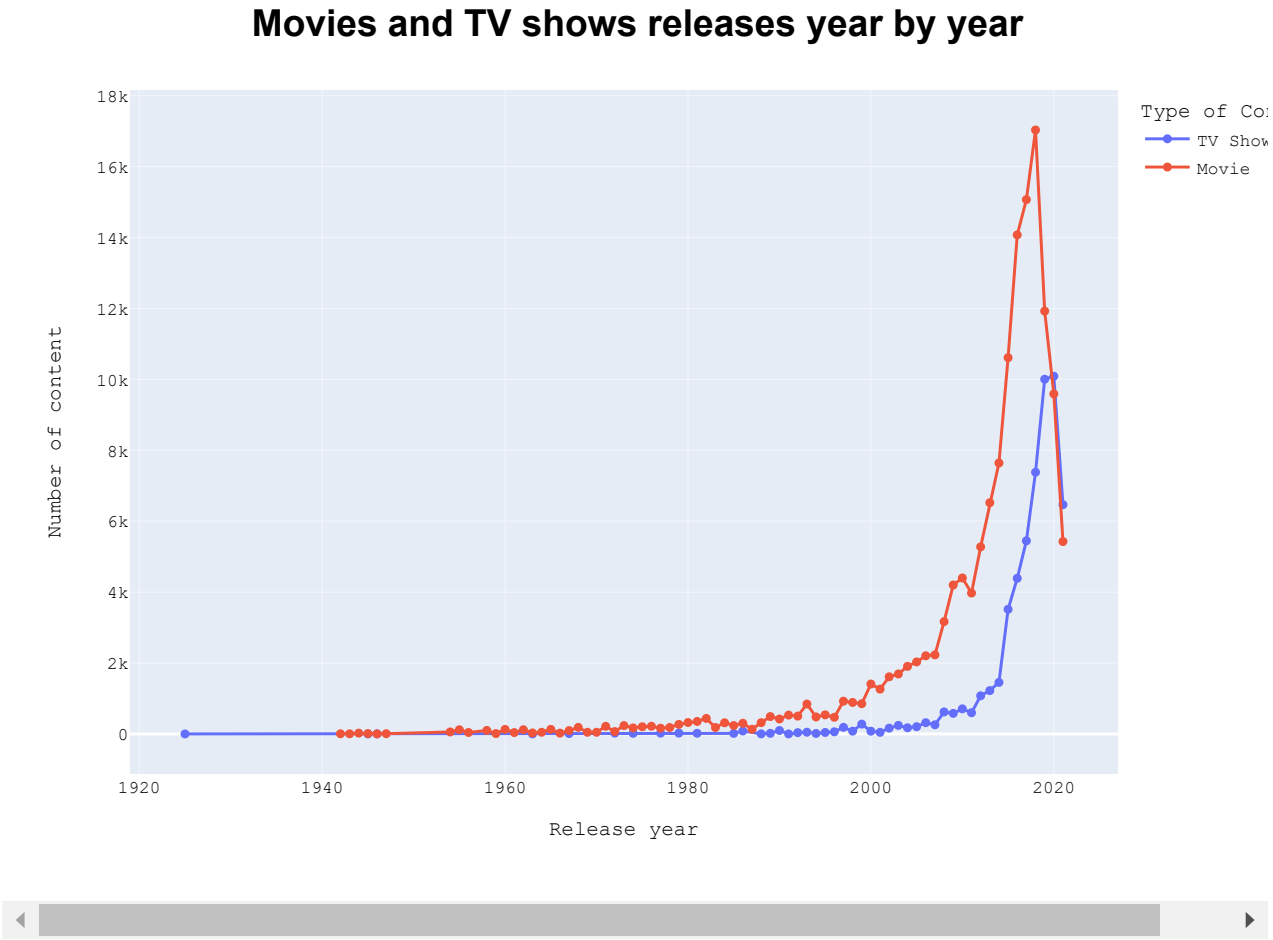


In [62]:

```
# Movies and TV shows Releases year by year
```

In [63]:

```
type_of_contents=df.groupby('type').size().index.tolist()
df6=df.loc[df['type'].isin(type_of_contents)]
df_6_upd=df6.groupby('release_year')['type'].value_counts().reset_index(name='counts')
fig = px.line(df_6_upd, x="release_year", y="counts", color='type',
              title='',
             markers=True)
fig.update_layout(xaxis_title="Release year",
                  yaxis_title="Number of content",
                  legend_title='Type of Content',
    height=600,
    width=900,
    title=dict(
        text='<b>Movies and TV shows releases year by year</b>',
        x=0.18,
        y=0.99,
        font=dict(
            family="Arial",
            size=25,
            color='#000000'
        )
    ),
    font=dict(
        family="Courier New, Monospace",
        size=12,
        color='#000000'
    )
)
```



In [64]:

```
#As we see in the above graph, the following points can be infered:

#Movies released more than TV Shows
#TV shows and movies contents are released more after 2015
#And during pandemic period- 2021, the movie and tv shows are released less and the curve dropping depicts the same
```

In [ ]:

```
#Total Watch hours content available
#We need to identify , how much watch hour content is available in Movies and TV shows,
#In movies, we can easily find it. But in case of TV shows , we can only count the number of seasons available,
#since we dont have duration of episodes in TV shows
```

In [65]:

```
fd=df[df['type']=='Movie'][['title','new_duration']]
fd.duplicated().sum()
fd.loc[fd.duplicated(), :]
fd=fd.drop_duplicates(keep='last')

#changing the datatype
fd=fd.astype({'new_duration': 'int32'})

moviehrs=fd['new_duration'].sum()
```

In [66]:

```
#Total seasons released in TV Shows category
```

In [67]:

```
od=df[df['type']=='TV Show'][['title','new_duration']]
od.duplicated().sum()
od.loc[od.duplicated(), :]
od=od.drop_duplicates(keep='last')

#changing the datatype
od=od.astype({'new_duration': 'int32'})

tvseasons=od['new_duration'].sum()
```

In [ ]:

```
#Pie Chart to display content available for TV shows an Movies
```

In [68]:

```
dic_hrs={'type':['Movies','Seasons'],'Total Count':[moviehrs,tvseasons]}
df_pie = pd.DataFrame(data=dic_hrs, columns=['type', 'Total Count'])
```

In [69]:

```
#Changing Duration of Movies from minutes to hours approximately %60
df_pie['Total Count'][0]=df_pie['Total Count'][0]//6
```

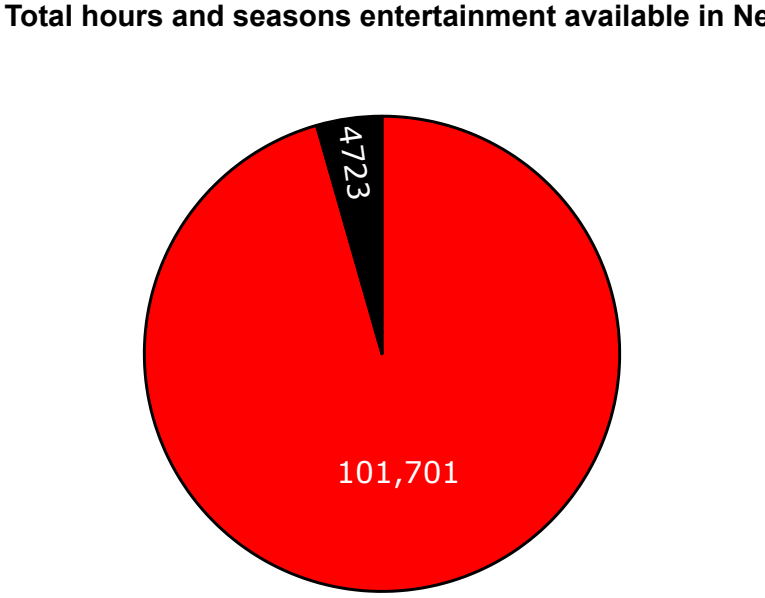
```
C:\Users\prafu\AppData\Local\Temp\ipykernel_7480\2001946693.py:2: SettingWithCopyWarning:


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)
```

In [70]:

```
#We plot the data of total hour content in movies and seasons in tv shows using plotly.
#Plotly helps to plot the pie charts in an interactive manner.
#Here we ll be using graph_objects library of plotly to plot the same
```

In [72]:

```
colors = ['red', 'black']
fig = go.Figure(go.Pie(
    name = "",
    values = df_pie['Total Count'],
    labels = df_pie['type'],
    text = ["Total Hour content in movies", "Total Seasons in TV Shows"],
    hovertemplate = "%{label}: <br>Content available: %{value} </br> %{text}",
    marker=dict(colors=colors, line=dict(color='#000000', width=2)))
)
fig.update_traces(textinfo='value',textfont_size=20)
fig.update_layout(
    height=500,
    title=dict(
        text='<b>Total hours and seasons entertainment available in Netflix</b>',
        x=0.5,
        y=0.95,
        font=dict(
            family="Arial",
            size=20,
            color='#000000'
        )
    ),

)
fig.show()
```

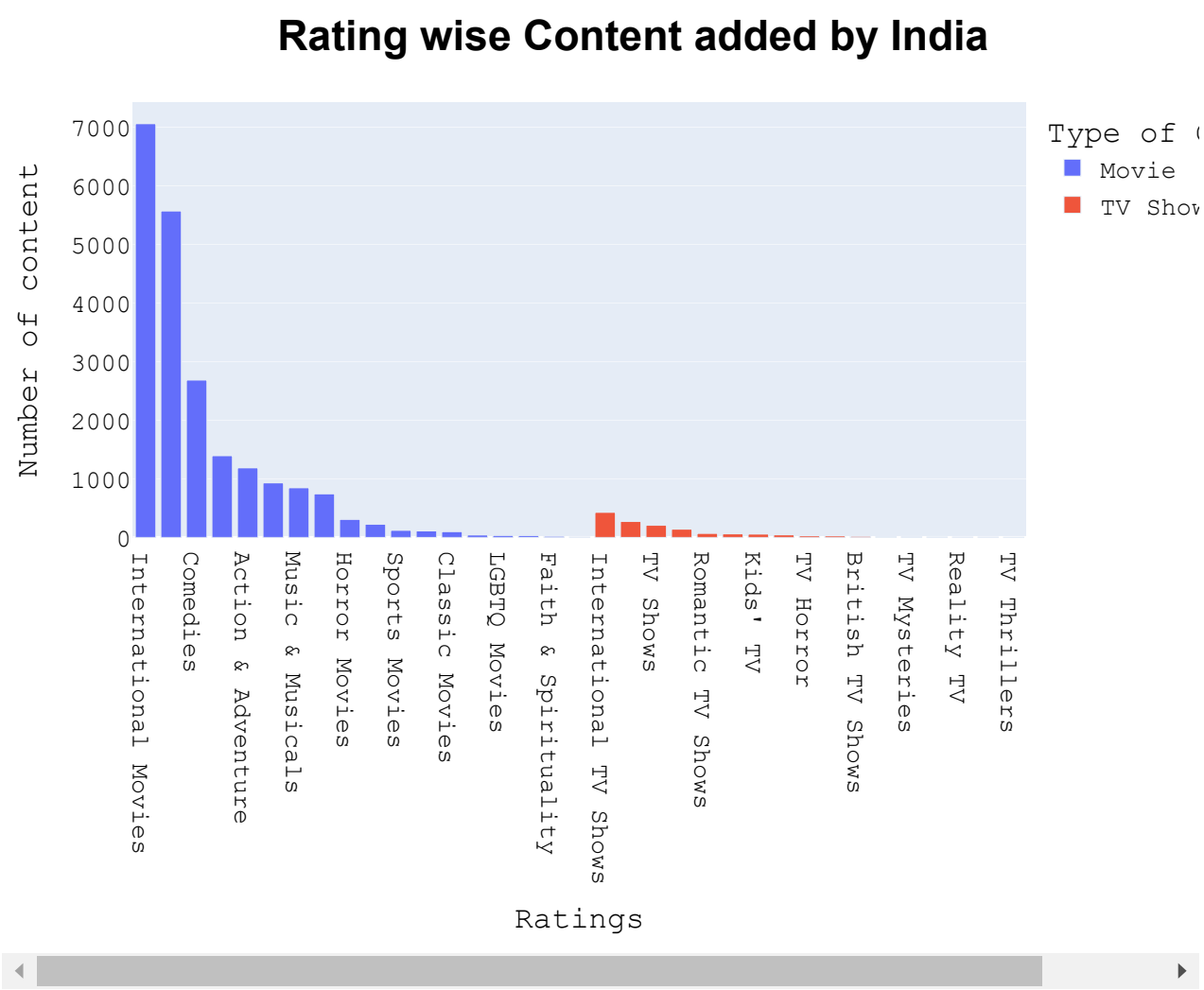


In [73]:

```
#Total count of movies and TV shows released in India - Genre wise
```

In [74]:

```
ir=df[df['country']=="India"]
type_of_rating=ir.groupby('genre').size().index.tolist()
df7=ir.loc[ir['genre'].isin(type_of_rating)]
df7=df7.groupby(['type'])['genre'].value_counts().reset_index(name='counts')
data_dict1 = {'Genre': df7['genre'],
              'Number of content': df7['counts'],'type': df7['type']
              }
df_R = pd.DataFrame(data=data_dict1, columns=['Genre', 'Number of content','type'])
fig = px.bar(df_R, x="Genre",
                   y="Number of content",
                   title="Genre wise and Category (Movie / TV Shows) wise content added in Netflix",
                   color='type')
fig.update_layout(autosize=False, width=950, height=700, xaxis_title="Ratings",
                  yaxis_title="Number of content",
                  legend_title='Type of Content',
    title=dict(
        text='<b>Rating wise Content added by India</b>',
        x=0.20,
        y=0.94,
        font=dict(
            family="Arial",
            size=30,
            color='#000000'
        )
    ),
    font=dict(
        family="Courier New, Monospace",
        size=18,
        color='#000000'
    )
)
fig.show()
```

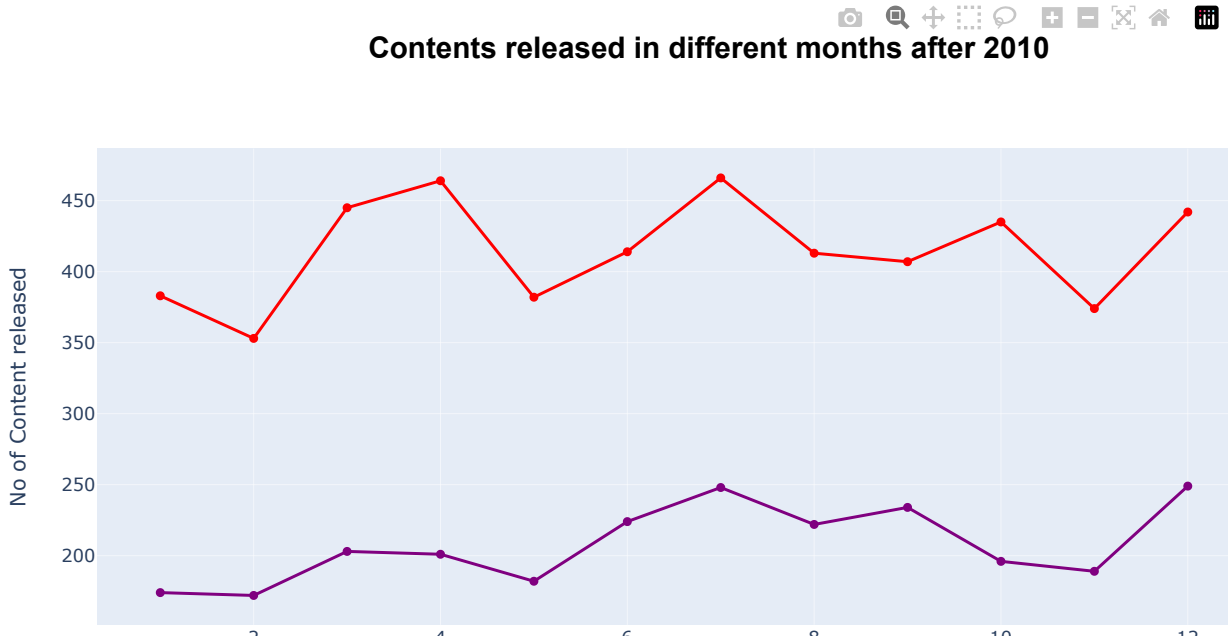


In [ ]:

```
#Contents released in different months after 2010
```

In [80]:

```
mdmm=df[(df['type']=='Movie')& (df["release_year"] >= 2010)][['title','month']]
mdmm=mdmm.drop_duplicates(keep='last')
mdtt=df[(df['type']=='TV Show')& (df["release_year"] >= 2010)][['title','month']]
mdtt=mdtt.drop_duplicates(keep='last')
movie_month=mdmm.groupby(["month"])["title"].count()
tvshow_month=mdtt.groupby(["month"])["title"].count()

fig = go.Figure()
fig.add_trace(go.Scatter(
x= [1,2,3,4,5,6,7,8,9,10,11,12],
y= movie_month,
showlegend=True,
text = mdmm,
name='Movie',
marker_color='Red'
))
fig.add_trace(go.Scatter(
x=[1,2,3,4,5,6,7,8,9,10,11,12],
y= tvshow_month,
showlegend=True,
text = movie_month,
name='TV Show',
marker_color='Purple'
))
fig.update_layout(xaxis_title="Months", yaxis_title="No of Content released ",
height=500,
title=dict(
    text='<b>Contents released in different months after 2010</b>',
    x=0.5,
    y=0.95,
    font=dict(
        family="Arial",
        size=20,
        color='#000000'
    )
),
)
fig.show()
```

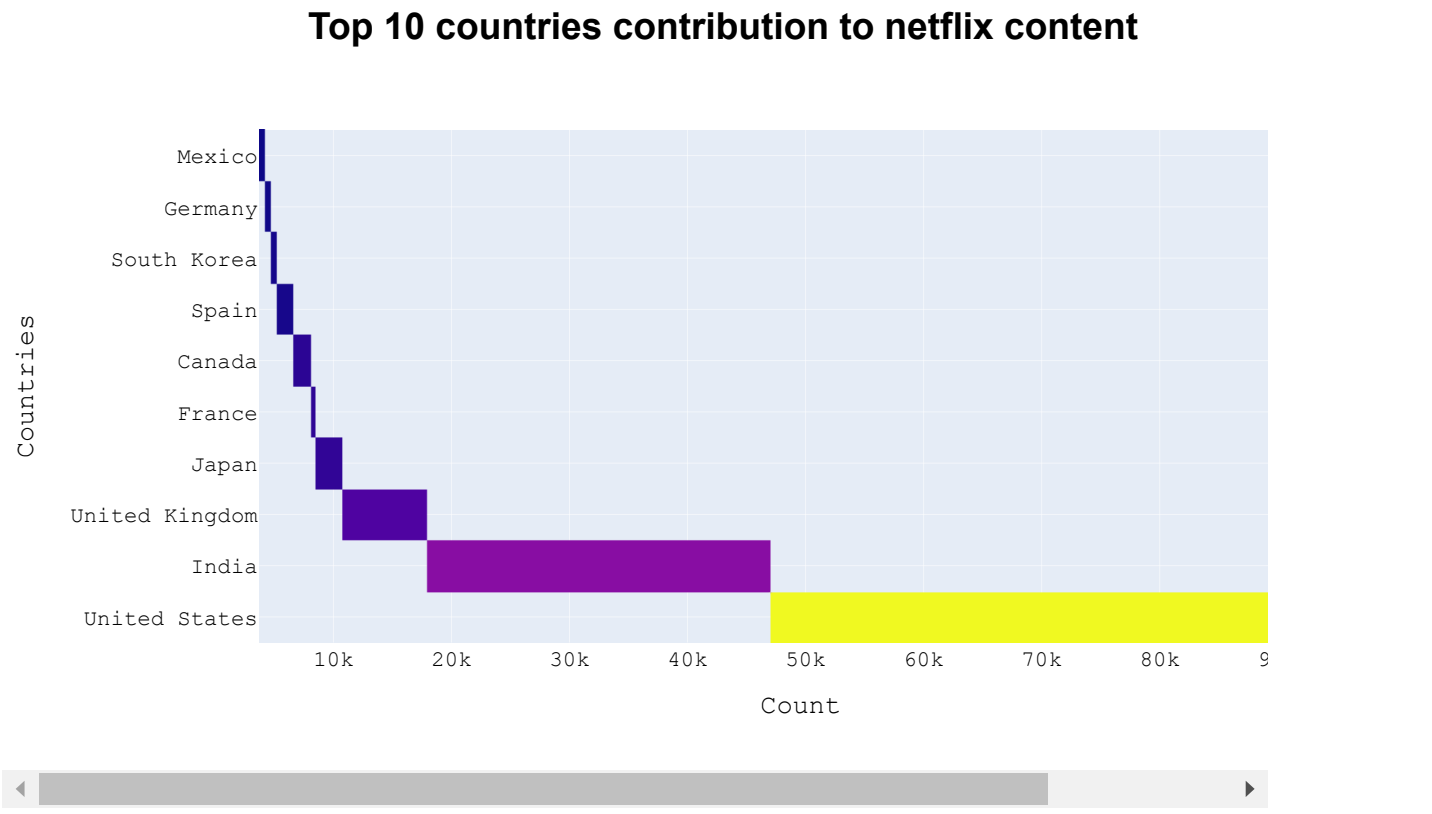


In [75]:

```
#Top 10 countries contribution to netflix content
#Top 10 Countries contributes most of the content in netflix content.
#And this is shown is heatmap. As from the heatmap, we infer, most of the content is from USA.
```

In [76]:

```
data_heat = {'country': df.groupby('country').size().sort_values(ascending=False)[:10].index,
             'Number of content': df.groupby('country').size().sort_values(ascending=False)[:10].values
             }
df_heat = pd.DataFrame(data=data_heat, columns=['country', 'Number of content'])
fig = go.Figure(data=go.Heatmap(
                   z=df_heat['Number of content'],
                   x=df_heat['Number of content'],
                   y=df_heat['country'],
                   hoverongaps = False))

fig.update_layout(xaxis_title="Count",
                  yaxis_title="Countries",
                  width=1000,

    title=dict(
        text='<b>Top 10 countries contribution to netflix content</b>',
        x=0.20,
        y=0.96,
        font=dict(
            family="Arial",
            size=25,
            color='#000000'
        )
    ),
    font=dict(
        family="Courier New, Monospace",
        size=15,
        color='#000000'
    )
)
fig.show()
```



In [77]:

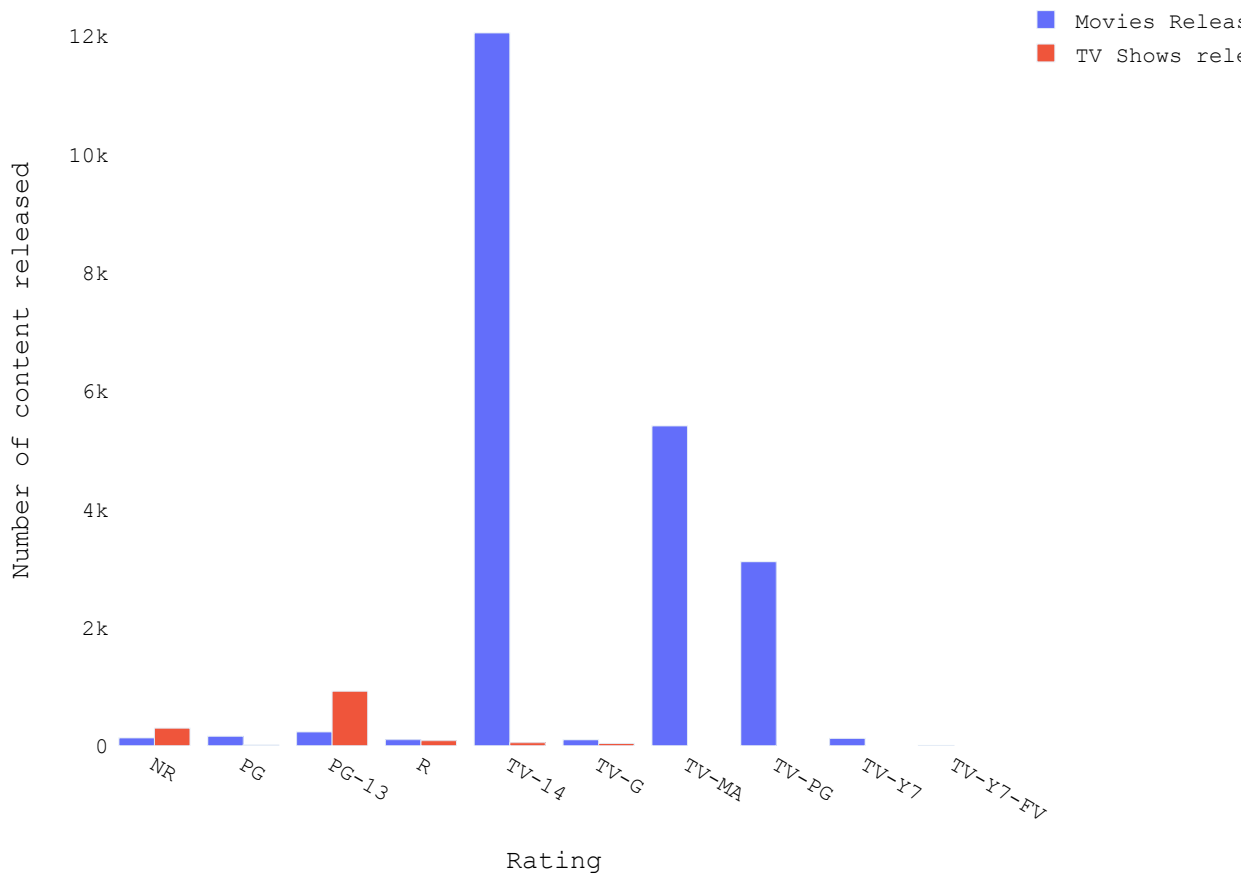
```
#Rating wise content added by top 20 countries
#This Graph contains Rating wise content added by multiple countries.
#This graph is one of the greatest feature available in plotly library. And has great feature added which is drop down integ
#This has been implemented here for analysing the contents of different countries
```

In [78]:

```
ir=df[df['country']=="India"]
type_of_rating=ir.groupby('rating').size().index.tolist()

df7_m=ir[ir['type']=="Movie"].loc[ir['rating'].isin(type_of_rating)]
df7_m=df7_m.groupby(['rating'])['type'].value_counts().reset_index(name='counts')

df7_t=ir[ir['type']=="TV Show"].loc[ir['rating'].isin(type_of_rating)]
df7_t=df7_t.groupby(['rating'])['type'].value_counts().reset_index(name='counts')


data_dict1 = {'Ratings': df7_m['rating'],
              'Movie': df7_m['counts'],'TV Shows': df7_t['counts']
             }

df_R = pd.DataFrame(data=data_dict1, columns=['Ratings', 'Movie','TV Shows'])


x  = 'Ratings'
y = 'Movie'
y1='TV Shows'

trace1 = {
    'x': df_R['Ratings'],
    'y': df_R['Movie'],
    'type': 'bar',
    'name':'Movies Released'
}
trace2 = {
    'x': df_R['Ratings'],
    'y': df_R['TV Shows'],
    'type': 'bar',
    'name':'TV Shows released'
}

data = [trace1,trace2]

# Create layout for the plot
layout=dict(


    title=dict(
        text='<b>Ratingwise content released in differnt countries</b>',
        x=0.25,
        y=0.96,
        font=dict(
            family="Arial",
            size=25,
            color='#000000'
        )
    ),
    font=dict(
        family="Courier New, Monospace",
        size=15,
        color='#000000'
    ),
    width=900, height=700, title_x=0.5,
    paper_bgcolor='#fff',
    plot_bgcolor="#fff",
    xaxis=dict(
        title='Rating',
        gridcolor='rgb(255,255,255)',
        zeroline= True,
    ),
    yaxis=dict(
        title='Number of content released',
        zeroline= False
            )
        )

fig = go.FigureWidget(data=data, layout=layout)


def update_fig(change):
    dc=change['new']
    ir=df[df['country']==dc[0]]
    type_of_rating=ir.groupby('rating').size().index.tolist()

    df7_m=ir[ir['type']=="Movie"].loc[ir['rating'].isin(type_of_rating)]
    df7_m=df7_m.groupby(['rating'])['type'].value_counts().reset_index(name='counts')
```

```
    df7_t=ir[ir['type']=="TV Show"].loc[ir['rating'].isin(type_of_rating)]
    df7_t=df7_t.groupby(['rating'])['type'].value_counts().reset_index(name='counts')


    data_dict1 = {'Ratings': df7_m['rating'],
              'Movie': df7_m['counts'],'TV Shows': df7_t['counts']
             }

    df_R = pd.DataFrame(data=data_dict1, columns=['Ratings', 'Movie','TV Shows'])

    with fig.batch_update():
        for trace, column in zip(fig.data,["Movie","TV Shows"]):
            trace.y = df_R[column]

drop = w.Dropdown(options=[
    ('India', ['India']),
    ('Japan', ['Japan']),
    ('United States', ['United States']),
    ('United Kingdom', ['United Kingdom']),
    ('France', ['France']),
    ('Canada', ['Canada']),
    ('South Korea', ['South Korea']),
    ('Germany', ['Germany']),
    ('Mexico', ['Mexico']),
    ('Turkey', ['Turkey']),
    ('Mexico', ['Mexico']),
    ('Australia', ['Australia']),
    ('Nigeria', ['Nigeria']),
    ('Hong Kong', ['Hong Kong']),
    ('Egypt', ['Egypt']),
    ('Indonesia', ['Indonesia']),
    ('Taiwan', ['Taiwan']),
    ('Belgium', ['Belgium']),
    ('Thailand', ['Thailand']),
    ('China', ['China'])

])
drop.observe(update_fig, names='value')
display(w.VBox([drop, fig]))
```

India


Ratingwise content released in differnt countries
Movies Releas
TV Shows rele
12k
10k
8k
6k
4k
2k
0
Number of content released
NR
PG
PG-13
R
TV-14
TV-G
TV-MA
TV-PG
TV-Y7
TV-Y7-FV
Rating

In [ ]: