Analysis objective

- Netflix got more movies or TV shows
- Number of films and television programs added to the platform yearly
- What is the annual average number of movies and tv shows added to the platform
- How long it takes for a movie or tv-show to be added to NETFLIX after its relaese date
- Time interval difference distribution between date added on NETFLIX and release date of Movies and TV shows
- Movies and TV Shows duration distribution
- Movies and TV Shows Ratings
- Locations where movies and tv shows were produced

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
data = pd.read_csv(r"C:\Users\pc\Documents\Projet\Projet Python\NETFLIX\netflix_titles.c
```

In [3]:
```
data.head(2)
```

Out[3]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentar
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	Internatio TV Shows, Dramas, Myste

Columns definition

- Country, is where the movie or TV show was produced
- Date added, is when the movie or TV was added on NETFLIX

Data exploration

In [4]:
```
data.shape
#8 807 records and 12 columns
```

Out[4]:
```
(8807, 12)
```

In [5]:
```
data.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 [6]:
```
data.isnull().sum()
# the values below tell us the number of empty records we have for the displayed columns
```

Out[6]:
```
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 [7]:
```
data["country"].value_counts()
# Some films were produced in several countries
```

Out[7]:
```
United States                                2818
India                                         972
United Kingdom                                419
Japan                                         245
South Korea                                   199
                                             ...
Romania, Bulgaria, Hungary                      1
Uruguay, Guatemala                              1
France, Senegal, Belgium                        1
Mexico, United States, Spain, Colombia          1
United Arab Emirates, Jordan                    1
Name: country, Length: 748, dtype: int64
```

In [8]:
```
data["rating"].value_counts()
# Rating tell us the type of audience to which the show or movie is adressed or not
# Some ratings with different names mean the same thing, we will further give them the s
# Depending on the analysis question we will delete or not, records with values 74 min,
```

Out[8]:
```
TV-MA       3207
TV-14       2160
TV-PG        863
R            799
PG-13        490
TV-Y7        334
TV-Y         307
PG           287
TV-G         220
NR            80
G             41
```

```
TV-Y7-FV        6
NC-17           3
UR              3
74 min          1
84 min          1
66 min          1
Name: rating, dtype: int64
```

Data cleaning

> We clean the data (column types, fill missing values, delete in necessary)
> Add some columns for future visuals
> Reshape the dataframe depending on the analysis objective and visuals

- **Date_added column": data type**

In [9]:
```
data["date_added"] = pd.to_datetime(data["date_added"])
```

In [10]:
```
data["date_added"].dtypes
```

Out[10]:
```
dtype('<M8[ns]')
```

- **Rating column: cleaning**

> **observation**

- When exploring the data above, we saw that, the rating column has the values ["74 min","84 min","66 min"]
- Display the records for the values.

In [11]:
```
data[data["rating"].isin(["74 min","84 min","66 min"])]
# The duration column has no values for this condition,
# These values type matches perfectly with the data type of the duration column
# We will fill in the duration empty cells with these records...thereby rendering our du
```

Out[11]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	2017-04-04	2017	74 min	NaN	Movies	o
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	2016-09-16	2010	84 min	NaN	Movies	
5813	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	2016-08-15	2015	66 min	NaN	Movies	T h

In [12]:
```
# filling the duration column
data.loc[((data["rating"]== "74 min") & (data["duration"].isnull())),"duration" ]="74 mi
data.loc[((data["rating"]== "84 min") & (data["duration"].isnull())),"duration" ]="84 mi
data.loc[((data["rating"]=="66 min") & (data["duration"].isnull())),"duration" ]="66 min
```

In [13]:
```
# replace the values ["74 min","84 min","66 min"] by nulls  in the rating column
data.loc[data["rating"].isin(["74 min","84 min","66 min"]), "rating"]=np.nan
```

- **Search on the internet for the rating type for the records with no rating value**

In [15]:
```
data["rating"].isnull().sum() #7 records
```

Out[15]:
```
7
```

In [16]:
```
data.loc[data["title"].str.contains("Louis C.K."), "rating"] = "TV-MA"
data.loc[data["title"].str.contains("Conversation with Oprah"), "rating"] = "TV-PG"
data.loc[data["title"].str.contains("Gargantia on the Verdurous Planet"), "rating"] = "P
data.loc[data["title"].str.contains("Little Lunch"), "rating"] = "TV-PG"
data.loc[data["title"].str.contains("My Honor Was Loyalty"), "rating"] = "PG-13"
```

In [17]:
```
data[data["rating"].isnull()]
# we are good no null cells for this column
```

Out[17]:

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description

In [18]:
```
data["rating"].value_counts()
```

Out[18]:
```
TV-MA       3210
TV-14       2160
TV-PG        867
R            799
PG-13        492
TV-Y7        332
TV-Y         307
PG           287
TV-G         220
NR            80
G             41
TV-Y7-FV       6
NC-17          3
UR             3
Name: rating, dtype: int64
```

- **Unify values in the Rating column**

> from the results some ratings give thesame information: UR=NR, G = TV-PG, PG = TV-PG, TV-Y7 = TV-Y.
> Let us unify this column

In [19]:
```
data["rating"]=data["rating"].replace(["UR","G","PG","TV-Y7"],["NR","TV-PG","TV-PG","TV-
```

In [20]:
```
data["rating"].value_counts()
```

Out[20]:
```
TV-MA       3210
TV-14       2160
TV-PG       1195
R            799
TV-Y         639
PG-13        492
TV-G         220
NR            83
TV-Y7-FV       6
```

```
NC-17           3
Name: rating, dtype: int64
```

In [21]:
```
data.isnull().sum()
```

Out[21]:
```
show_id            0
type               0
title              0
director        2634
cast             825
country          831
date_added        10
release_year       0
rating             0
duration           0
listed_in          0
description        0
dtype: int64
```

- **Observation**

> Well if we want information about directors we will not be able to have information for 2 634 records.
> Same for the cast, country, date_added columns

- **Country column**

> We see that some movies and TV shows are done in more than one countries(a cell contains morethan one country).
> If we are interested to see the number of movies or TV shows in each country we have to shape our data

In [22]:
```
data["country"].nunique()
```

Out[22]:
```
748
```

In [23]:
```
data_country = data.copy()
```

In [24]:
```
data_country = data_country[~(data_country["country"].isnull())]
#remove records where country is null
```

- **Split the country column**

> We do this to have columns with a single country par cell.
> We will later on delete spaces at the begining and at the end in each cells for the country column

In [25]:
```
data_country[["country_0", "country_1","country_2","country_3","country_4","country_5","
```

In [26]:
```
data_country["country_0"]=data_country["country_0"].str.strip(" ")
data_country["country_1"]=data_country["country_1"].str.strip(" ")
data_country["country_2"]=data_country["country_2"].str.strip(" ")
data_country["country_3"]=data_country["country_3"].str.strip(" ")
data_country["country_4"]=data_country["country_4"].str.strip(" ")
data_country["country_5"]=data_country["country_5"].str.strip(" ")
```

```
data_country["country_6"]=data_country["country_6"].str.strip(" ")
data_country["country_7"]=data_country["country_7"].str.strip(" ")
data_country["country_8"]=data_country["country_8"].str.strip(" ")
data_country["country_9"]=data_country["country_9"].str.strip(" ")
data_country["country_10"]=data_country["country_10"].str.strip(" ")
data_country["country_11"]=data_country["country_11"].str.strip(" ")
```

- **The number of movies and TV shows produced in eacu country**

> Pivot table for each columns from our strsplit function
> This to easily visualise the data in a choropleth map
> To answer analysis questions like..which countries is used the most to produced movies or TV shows etc

In [27]:
```
data_country_0 = data_country.pivot_table(index="country_0", columns = "type",aggfunc={"
data_country_1 = data_country.pivot_table(index="country_1", columns = "type",aggfunc={"
data_country_2 = data_country.pivot_table(index="country_2", columns = "type",aggfunc={"
data_country_3 = data_country.pivot_table(index="country_3", columns = "type",aggfunc={"
data_country_4 = data_country.pivot_table(index="country_4", columns = "type",aggfunc={"
data_country_5 = data_country.pivot_table(index="country_5", columns = "type",aggfunc={"
data_country_6 = data_country.pivot_table(index="country_6", columns = "type",aggfunc={"
data_country_7 = data_country.pivot_table(index="country_7", columns = "type",aggfunc={"
data_country_8 = data_country.pivot_table(index="country_8", columns = "type",aggfunc={"
data_country_9 = data_country.pivot_table(index="country_9", columns = "type",aggfunc={"
data_country_10 = data_country.pivot_table(index="country_10", columns = "type",aggfunc=
data_country_11 = data_country.pivot_table(index="country_11", columns = "type",aggfunc=
```

- **Concatenate the different dataframe objects**

In [28]:
```
data_country= data_country_0.append([data_country_1, data_country_2,data_country_3,data_
```

```
C:\Users\pc\AppData\Local\Temp\ipykernel_4556\423964477.py:1: FutureWarning: The frame.a
ppend method is deprecated and will be removed from pandas in a future version. Use pand
as.concat instead.
  data_country= data_country_0.append([data_country_1, data_country_2,data_country_3,dat
a_country_4,data_country_5,data_country_6,data_country_7,data_country_8,data_country_9,d
ata_country_10,data_country_11])
```

In [29]:
```
data_country= data_country.reset_index()
data_country = data_country[~(data_country["index"]=="")]
# the column coutry had data like "poland," or ",southafrica"
# when applying the split we have columns with cells with no value (no country name)
# we delete it from our data because it yields no information
```

In [31]:
```
data_country["index"].nunique() # 122 countries
```

Out[31]:
```
122
```

In [33]:
```
data_country[data_country["index"]=="Argentina"]
```

Out[33]:

	index	type	
type		Movie	TV Show
1	Argentina	56.0	20.0
89	Argentina	12.0	NaN
181	Argentina	3.0	NaN

In [34]:
```
data_country.groupby('index').sum().loc[:,["type"]].head(5)
```

```
C:\Users\pc\AppData\Local\Temp\ipykernel_4556\2458670135.py:1: PerformanceWarning: dropp
ing on a non-lexsorted multi-index without a level parameter may impact performance.
  data_country.groupby('index').sum().loc[:,["type"]].head(5)
```

Out[34]:

	type
type	Movie	TV Show
index		
Afghanistan	1.0	0.0
Albania	1.0	0.0
Algeria	3.0	0.0
Angola	1.0	0.0
Argentina	71.0	20.0

In [35]:
```
data_country_sum = data_country.groupby('index').sum().loc[:,["type"]]
```

```
C:\Users\pc\AppData\Local\Temp\ipykernel_4556\1883398543.py:1: PerformanceWarning: dropp
ing on a non-lexsorted multi-index without a level parameter may impact performance.
  data_country_sum = data_country.groupby('index').sum().loc[:,["type"]]
```

In [36]:
```
data_country_sum.stack().head(5)
```

Out[36]:

		type
index	type	
Afghanistan	Movie	1.0
	TV Show	0.0
Albania	Movie	1.0
	TV Show	0.0
Algeria	Movie	3.0

In [37]:
```
t = data_country_sum.stack()
t.rename(columns = {'type':"total"}, inplace = True)
```

In [38]:
```
t.head(3)
```

Out[38]:

		total
index	type	
Afghanistan	Movie	1.0
	TV Show	0.0
Albania	Movie	1.0

In [39]:
```
data_choro = t.reset_index()
data_choro.head(3)
```

Out[39]:

	index	type	total
0	Afghanistan	Movie	1.0

1	Afghanistan	TV Show	0.0
2	Albania	Movie	1.0

- **Create column ISO CODE for index column (country)**

> data_choro doesnot contain the three letter ISO CODE for countries, we have to add a column with ISO CODE values

In [40]:
```
import pycountry
```

In [41]:
```
def country_code(country_name):
    try:
        return pycountry.countries.get(name=country_name).alpha_3
    except:
        return("NA")

data_choro["country_iso_code"]=data_choro["index"].apply(lambda row : country_code(row))
```

In [42]:
```
data_choro.head(2)
```

Out[42]:

	index	type	total	country_iso_code
0	Afghanistan	Movie	1.0	AFG
1	Afghanistan	TV Show	0.0	AFG

- **Some countries do not have an ISO CODE**

> We shall use the map function to complete to replace the NA values

In [43]:
```
data_choro["index"][data_choro["country_iso_code"]=="NA"].unique()
```

Out[43]:
```
array(['Czech Republic', 'East Germany', 'Iran', 'Palestine', 'Russia',
       'South Korea', 'Soviet Union', 'Syria', 'Taiwan', 'Vatican City',
       'Venezuela', 'Vietnam', 'West Germany'], dtype=object)
```

In [44]:
```
dico = {'Czech Republic' : "CZE" , 'East Germany':"DEU", 'Iran':"IRN", 'Palestine':"PS",
       'South Korea':"KOR", 'Soviet Union': "SUN", 'Syria':"SYR", 'Taiwan':"TWN", 'Vatic
       'Venezuela':"VEN", 'Vietnam':"VNM", 'West Germany':"DEU"}
```

In [45]:
```
data_choro["country_iso_code"] = np.where(data_choro["country_iso_code"]=="NA",data_chor
```

In [46]:
```
data_choro[data_choro["country_iso_code"]=="NA"]  # zero records we are good
```

Out[46]:

	index	type	total	country_iso_code

- **Duration column**

> This column has two units **min (Movie)** and **seasons (TV shows)**
> If we want to do some calculations on it we cannot because it is of string type, we will see how to convert it to an int type

In [48]:
```
data.head(2)
```

Out[48]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	2020	PG-13	90 min	Documentar
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021	TV-MA	2 Seasons	Internatio TV Shows, Dramas, Myste

In [49]:
```
data_movie = data[data["type"]=="Movie"]
data_movie["duration"].isnull().sum()
```

Out[49]:
```
0
```

In [50]:
```
data_movie[["duration_value","duration_unit"]]= data_movie["duration"].str.split(" ", ex
```

```
C:\Users\pc\AppData\Local\Temp\ipykernel_4556\1670387490.py:1: SettingWithCopyWarning:
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
  data_movie[["duration_value","duration_unit"]]= data_movie["duration"].str.split(" ",
expand = True)
C:\Users\pc\AppData\Local\Temp\ipykernel_4556\1670387490.py:1: SettingWithCopyWarning:
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
  data_movie[["duration_value","duration_unit"]]= data_movie["duration"].str.split(" ",
expand = True)
```

In [51]:
```
data_movie["duration_value"]=data_movie["duration_value"].astype(int)
```

```
C:\Users\pc\AppData\Local\Temp\ipykernel_4556\3410939696.py:1: SettingWithCopyWarning:
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
  data_movie["duration_value"]=data_movie["duration_value"].astype(int)
```

In [52]:
```
data_show = data[data["type"]=="TV Show"]
data_show["duration"].isnull().sum()
```

Out[52]:
```
0
```

In [53]:
```
data_show[["duration_value","duration_unit"]]= data_show["duration"].str.split(" ", expa
```

```
C:\Users\pc\AppData\Local\Temp\ipykernel_4556\2848077729.py:1: SettingWithCopyWarning:
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
```

```
  data_show[["duration_value","duration_unit"]]= data_show["duration"].str.split(" ", ex
pand = True)
C:\Users\pc\AppData\Local\Temp\ipykernel_4556\2848077729.py:1: SettingWithCopyWarning:
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
  data_show[["duration_value","duration_unit"]]= data_show["duration"].str.split(" ", ex
pand = True)
```

In [54]:
```
data_show["duration_value"]=data_show["duration_value"].astype(int)
```

```
C:\Users\pc\AppData\Local\Temp\ipykernel_4556\513797792.py:1: SettingWithCopyWarning:
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
  data_show["duration_value"]=data_show["duration_value"].astype(int)
```

- **Time interval difference between Movies & TV Shows release_year and date_added on NETFLIX**

In [55]:
```
data["date_added"].dtypes
```

Out[55]:
```
dtype('<M8[ns]')
```

In [56]:
```
data_interval = data.copy()
```

In [57]:
```
data_interval.insert(7,'year_date_added',data_interval['date_added'].dt.year)
```

In [58]:
```
data_interval.insert(9,'time',data_interval['year_date_added'] - data_interval['release_
```

In [60]:
```
data_interval[data_interval["time"]<0].shape
# we have 14 records where date added to netflix is < release year, kind of weird
# 2 records are Movies, the remaining TV shows
```

Out[60]:
```
(14, 14)
```

In [61]:
```
data_interval[data_interval["time"].isnull()].shape
# we have 10 records where both date added and date release is null, these records are o
```

Out[61]:
```
(10, 14)
```

In [62]:
```
len(data_interval[(data_interval["time"]<0) | (data_interval["time"].isnull())])
# we were expecting 24 records, we are good
```

Out[62]:
```
24
```

In [63]:
```
data_interval[(data_interval["time"]<0) | (data_interval["time"].isnull())].head(3)
```

Out[63]:

	show_id	type	title	director	cast	country	date_added	year_date_added	release_year	time	rat
1551	s1552	TV Show	Hilda	NaN	Bella Ramsey, Ameerah Falzon-Ojo, Oliver Nelso...	United Kingdom, Canada, United States	2020-12-14	2020.0	2021	-1.0	T

1696	s1697	TV Show	Polly Pocket	NaN	Emily Tennant, Shannon Chan-Kent, Kazumi Evans...	Canada, United States, Ireland	2020-11-15	2020.0	2021	-1.0	T
2920	s2921	TV Show	Love Is Blind	NaN	Nick Lachey, Vanessa Lachey	United States	2020-02-13	2020.0	2021	-1.0	

Business questions and visuals

- **Netflix got more movies or TV shows?**

In [64]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

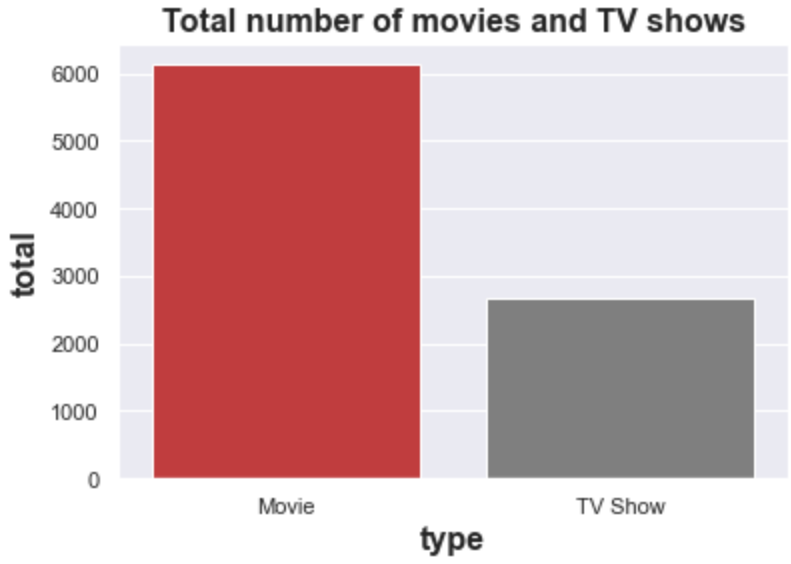
In [65]:
```
data.groupby("type").count().loc[:,["show_id"]].reset_index()
```

Out[65]:

	type	show_id
0	Movie	6131
1	TV Show	2676

In [66]:
```
t_1 = data.groupby("type").count().loc[:,["show_id"]].reset_index()
t_1.rename(columns={"show_id": "total"}, inplace = True)
```

In [67]:
```
sns.set()
sns.barplot(data=t_1, x="type", y="total", color = 'red', palette = ['tab:red', 'tab:gre
plt.xlabel('type', fontsize=16, fontweight = "bold");
plt.ylabel('total', fontsize=16,fontweight = "bold");
plt.title ("Total number of movies and TV shows",fontsize=16,fontweight = "bold")
plt.show()
```



- **What can we say**

> Visually we observe that we have **more Movies than TV shows added on NETFLIX**
> Well, we can't give a conclusion because we are doing exploratory analysis
> In order to confidently answer this question we have to do some statistical test like hypothesis testing....

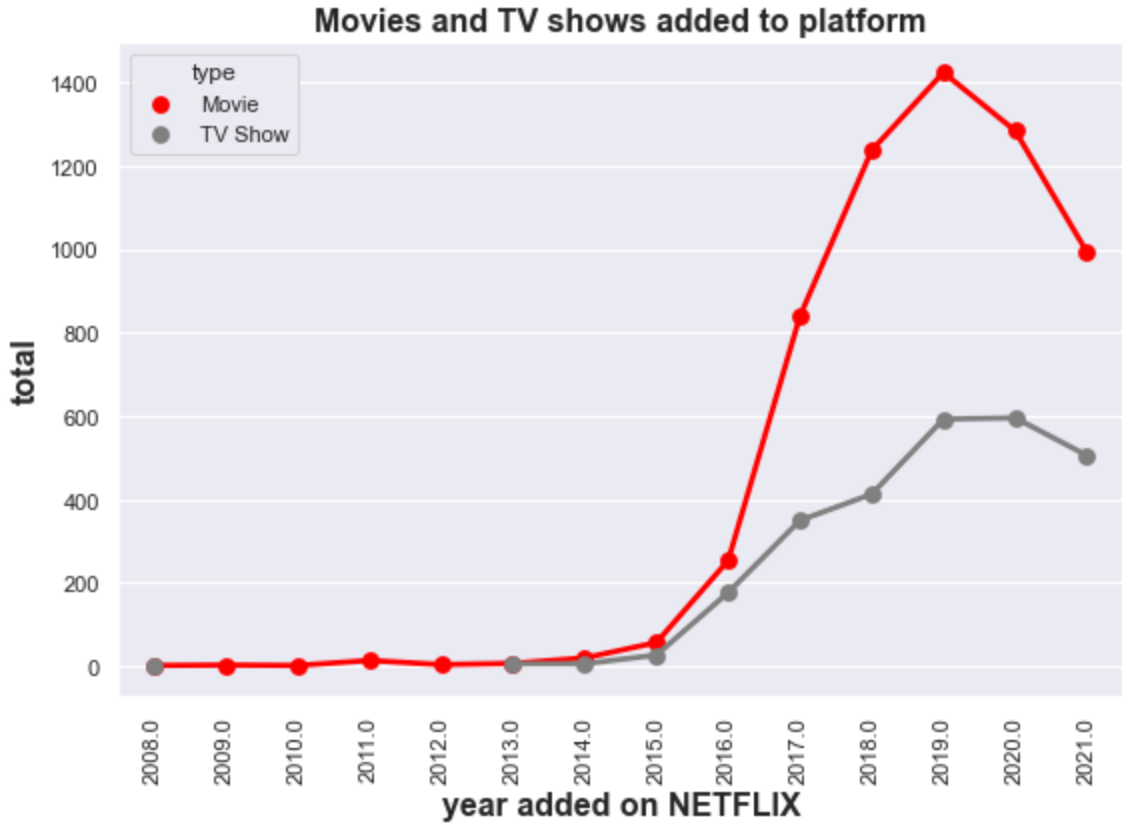
- **What is the trend of number of films or television programs added to the platform?**

In [68]:
```
data_interval.groupby(["year_date_added","type"]).count().loc[:,["show_id"]].reset_index
```

Out[68]:

	year_date_added	type	show_id
19	2019.0	TV Show	592
20	2020.0	Movie	1284
21	2020.0	TV Show	595
22	2021.0	Movie	993
23	2021.0	TV Show	505

In [69]:
```
t_2 = data_interval.groupby(["year_date_added","type"]).count().loc[:,["show_id"]].reset
sns.set()
plt.figure(figsize=(9,6))
sns.pointplot(x='year_date_added', y='show_id', data = t_2, hue='type', palette=["red","
plt.xlabel('year added on NETFLIX', fontsize=16, fontweight = "bold");
plt.ylabel('total', fontsize=16,fontweight = "bold");
plt.title ("Movies and TV shows added to platform",fontsize=16,fontweight = "bold")
plt.xticks(rotation = 90)
plt.show()
```



- **What can we say**

> We have **a lot of variations throughout the years**. We first have **very few and constant addings** from **[2008 - 2014]** for Movies, a sudden **rise** from **[2014 - 2018]** then a **fall** from **[2018 - 2021]**
>
> For TV shows, we have **addings** in **2008**, then **no addings** up to **2013**. From **2013** we experience a **rise** and a **fall** from **2019**
>
> More to that we have 10 records of TV-shows, where we don't have a date added value

- **What is the annual average number of movies and tv shows added to the platform?**

> This calculation is done for a time range [2008 - 2021] from our dataset

In [70]:
```
t_2.rename(columns={'show_id' : 'total'}, inplace = True)
```

- **Movies**

In [71]:
```
round(t_2["total"][t_2["type"]=="Movie"].sum()/len(t_2["year_date_added"][t_2["type"]=="
```

Out[71]:
```
438.0
```

- **TV-Shows**

In [72]:
```
t_2["total"][t_2["type"]=="TV Show"].sum()
```

Out[72]:
```
2666
```

In [73]:
```
(data[(data["type"]=="TV Show") & (data["date_added"].isnull())]).shape
```

Out[73]:
```
(10, 12)
```

- **Observation**

> Previously, we saw that the total number of TV shows is equal to 2 670
> here we have 2 666, where is the remaining 10 tv shows?. It is because we have 10 TV shows that have no date_added values

In [74]:
```
# TV shows average
round(t_2["total"][t_2["type"]=="TV Show"].sum()/len(t_2["year_date_added"][t_2["type"]=
```

Out[74]:
```
267.0
```

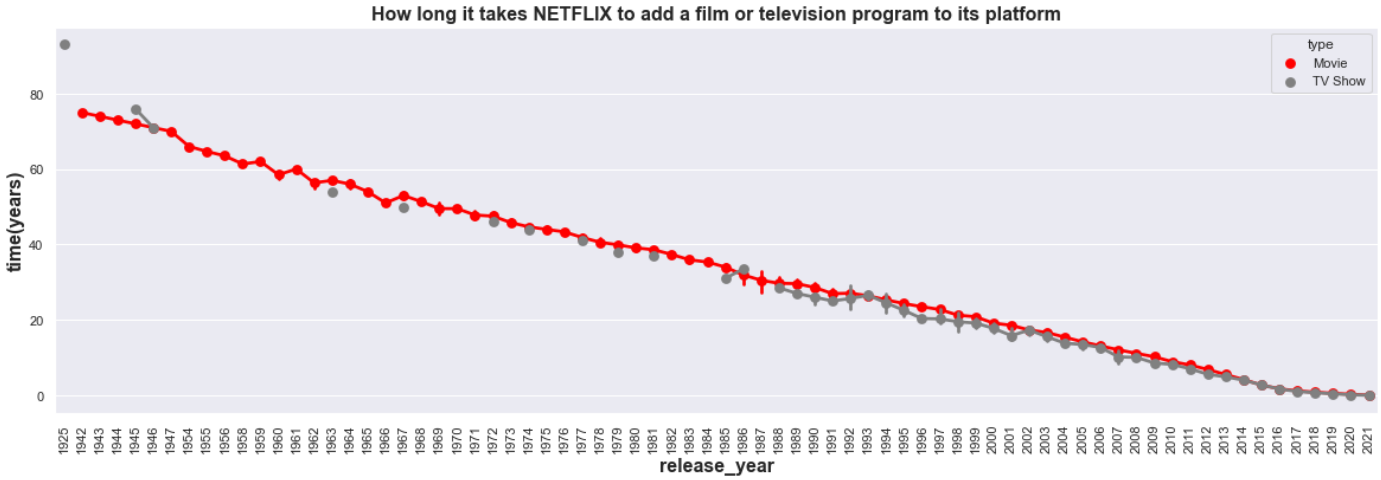
- **What can we say?**

> On average (yearly) we have **267 TV shows** and **438 Movies** added on NETFLIX from our sample data (dataset)
> From the adding trend above, there is alot of variation throughout the years, the yearly average varies a lot, we can't give a firm conclusion, that each year we will fairly expect the above calculated average values.

- **How long it takes for a movie or tv-show to be added to NETFLIX after its relaese date**

> We previously saw that, we have 10 records where we have no date added and dte release values
> We also say that we have 14 records where date added is < date release, this is weird

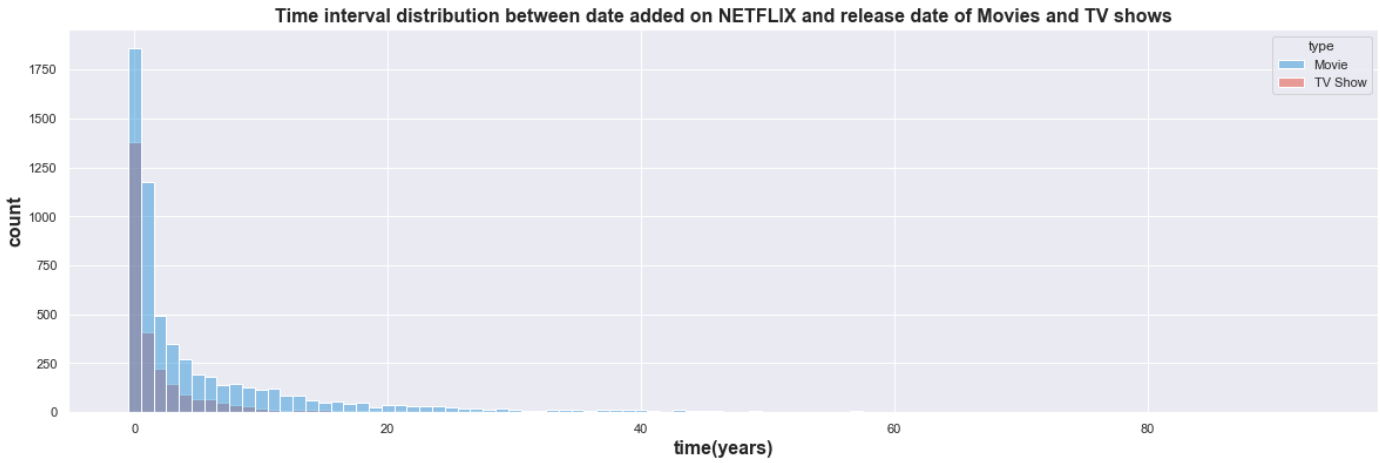
In [75]:
```
t_3 = data_interval[["type","year_date_added","release_year","time"]]
sns.set()
plt.figure(figsize=(20,6))
sns.pointplot(x='release_year', y='time', data = t_3, hue='type', palette=["red","grey"]
plt.xlabel('release_year', fontsize=16, fontweight = "bold");
plt.ylabel('time(years)', fontsize=16,fontweight = "bold");
plt.title ("How long it takes NETFLIX to add a film or television program to its platfor
plt.xticks(rotation = 90)
plt.show()
```



- **What can we say?**

> We have a **negative linear relationship**, as the years increase the time NETFLIX takes to add a movie or TV-show to its platform reduces, this is pretty good
> By the year 2013 we have approxiamtely 1 year for an adding, for the years >2016 the addings are done in less than a year.

In [76]:
```
palette = sns.color_palette(["#3498db",'#e74c3c'])
sns.set_palette(palette)
plt.figure(figsize=(20,6))
sns.histplot(data= data_interval[~((data_interval["time"]<0) | (data_interval["time"].is
plt.title("Time interval distribution between date added on NETFLIX and release date of
plt.xlabel("time(years)",fontsize=16, fontweight = "bold")
plt.ylabel("count",fontsize=16, fontweight = "bold")
plt.show()
```



- **What can we say**

> We have very few addings for a time interval difference of 40+ years
> For an interval difference of less than a year we have the highest addings for both movies and tv-shows

- **Movies & TV shows duration**

- **Top 05 Movies with the longest duration**

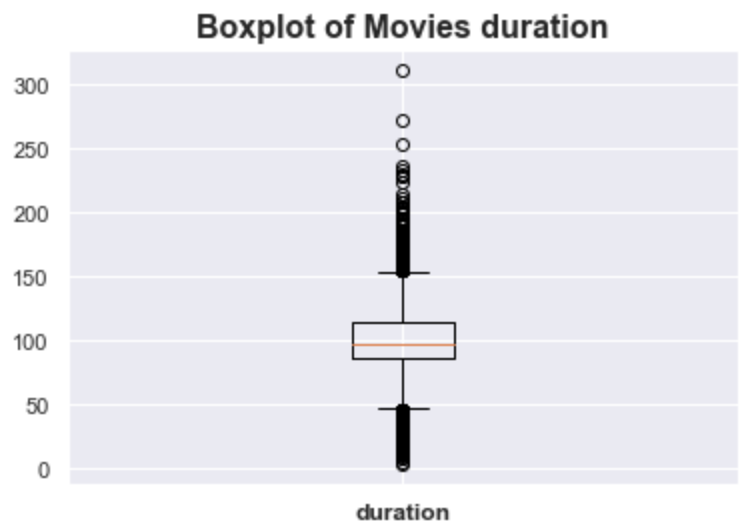
In [77]:
```
data_movie[["type","title","country","duration_value"]].sort_values(["duration_value"],
```

Out[77]:

	type	title	country	duration_value
4253	Movie	Black Mirror: Bandersnatch	United States	312
717	Movie	Headspace: Unwind Your Mind	NaN	273
2491	Movie	The School of Mischief	Egypt	253
2487	Movie	No Longer kids	Egypt	237
2484	Movie	Lock Your Girls In	NaN	233

- **Distribution of Movies' duration**

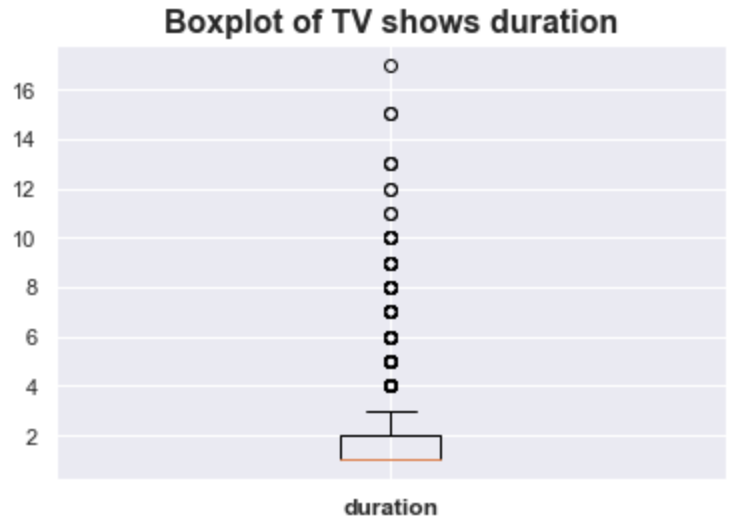
In [78]:
```
sns.set()
plt.boxplot(data_movie["duration_value"])
plt.title("Boxplot of Movies duration", fontsize=16, fontweight="bold")
plt.xticks([1],["duration"], fontsize=12, fontweight="bold")
plt.show()
```



- **What can we say?**

> The **bottom 50 % of movies** has a duration **< 95 min** and the **upper 50 % of movies** has a duration **> 95 min**
> The **bottom 25 % of movies** has a duration **< 85 min** and the upper **25 % of movies** have a duration **>115 min**
> There is **a lot of variation** in the data, the **median does not cut exactly at half of the box**, the data is skewed to the right... thereby dragging the mean value to a higher duration
> We also have **outliers** represented by **extreme congested circles**, these are Movies that have duration **< 45 min** and **> 150 min**

In [79]:
```
sns.set()
plt.boxplot(data_show["duration_value"],showfliers=True )
plt.title("Boxplot of TV shows duration", fontsize=16, fontweight="bold")
plt.xticks([1],["duration"], fontsize=12, fontweight="bold")
plt.show()
```



- **What can we say?**

> The **bottom 50 % of TV shows** has **1 season**, there is **no variation** that is why we have **no box or tail below the median (line in red)**
> The **upper 50 % of TV shows** has more than **1 seasons**

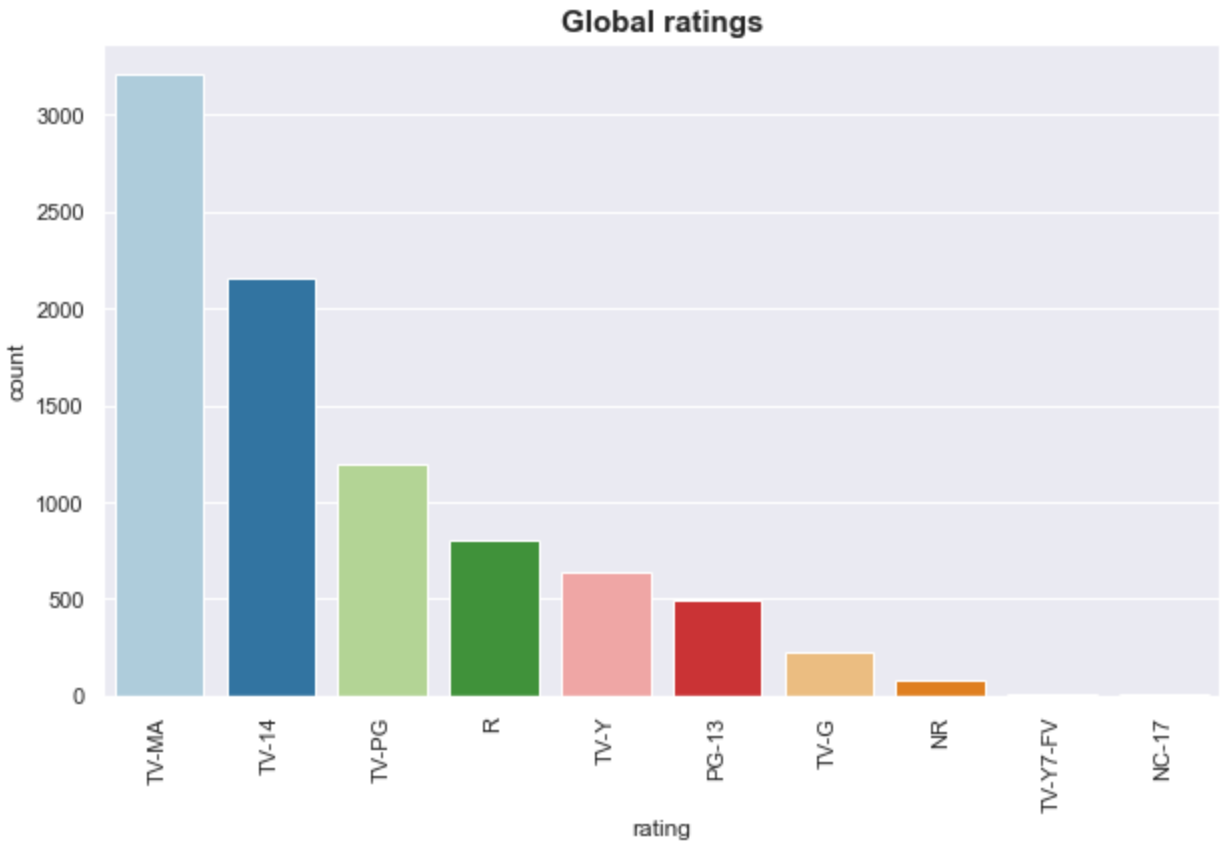
> The **upper 25 %** of our data has more than **2 seasons** and a **maximum** of **3 seasons**, with **outliers** that ranges from **4 seasons to 17 seasons**
> There is **a lot of variation in the duration of TV shows as compared to Movies**, it is **highly skewed to right**

- **Movies and TV shows ratings**

In [80]:
```
data["rating"].value_counts()
```

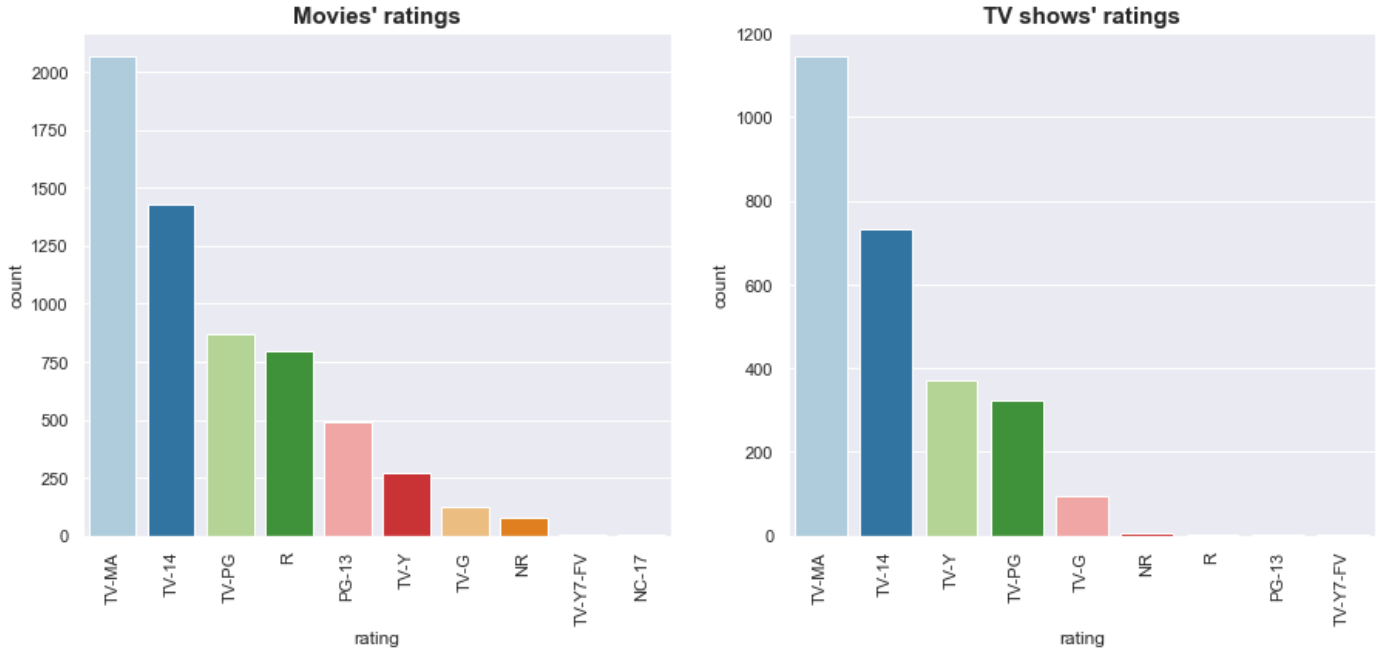
Out[80]:
```
TV-MA       3210
TV-14       2160
TV-PG       1195
R            799
TV-Y         639
PG-13        492
TV-G         220
NR            83
TV-Y7-FV       6
NC-17          3
Name: rating, dtype: int64
```

In [81]:
```
palette = sns.color_palette ("Paired")
sns.set_palette(palette)

plt.figure(figsize=(10,6))
sns.countplot(data=data, x = "rating",order= data["rating"].value_counts().index )
plt.title("Global ratings", fontsize=15, fontweight = "bold")
plt.xticks(rotation = 90)
plt.show()
```



In [82]:
```
palette = sns.color_palette ("Paired")
sns.set_palette(palette)

plt.figure(figsize=(15,6))
```

```
plt.subplot(121)
sns.countplot(data=data_movie, x = "rating",order = data_movie["rating"].value_counts().
plt.title("Movies' ratings", fontsize=15, fontweight = "bold")
plt.xticks(rotation = 90)

plt.subplot(122)
sns.countplot(data=data_show, x = "rating",order = data_show["rating"].value_counts().in
plt.title("TV shows' ratings", fontsize=15, fontweight= "bold")
plt.xticks(rotation = 90)
plt.show()
```



- **What can we say?**

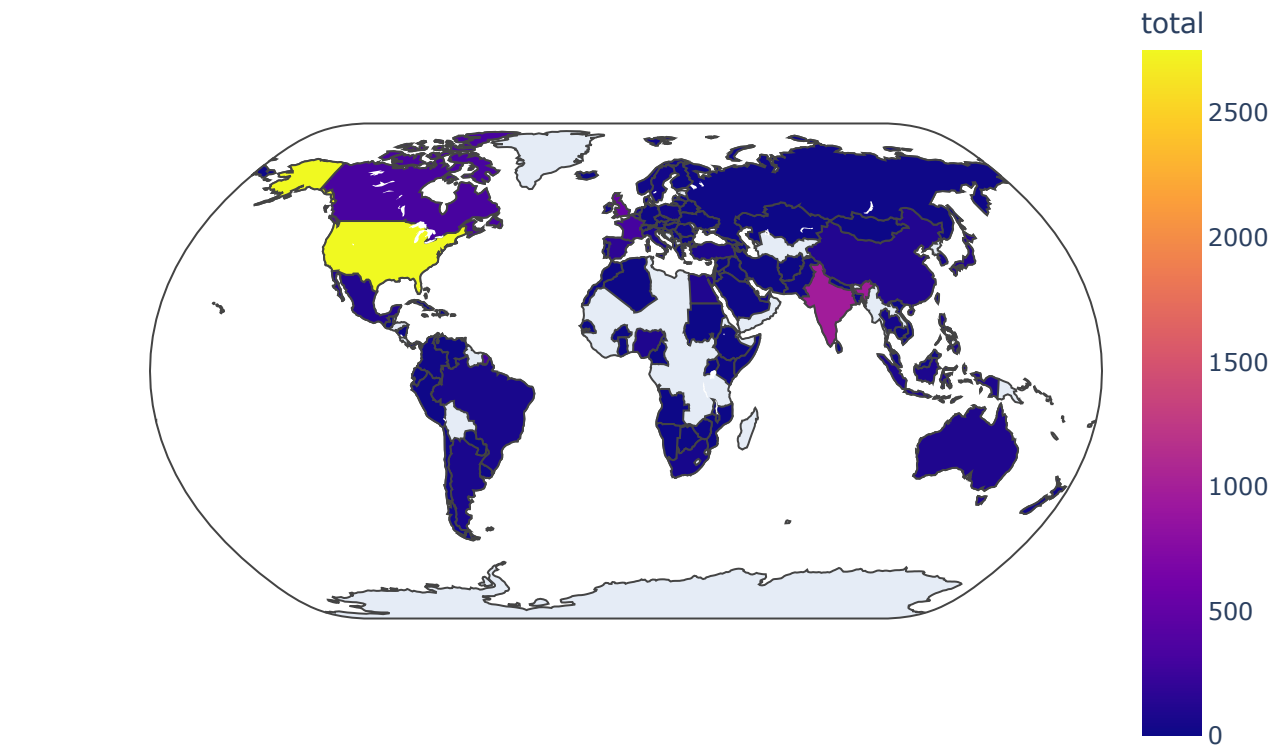
> Generally, we have **more videos**(movies & tv-shows) in the ratings **TV-MA**, **TV-14**, **TV-PG**, **R** which are for **mature**, **adult audiences** and may be **unsuitable for children under 17**
> For a **young audience** we have: **PG-13 more** in **movies** than **tv-shows**(very few), **TV-Y** and **TV-G** are **more** in **tv-shows** than **movies**
> We have **few movies** and **tv-shows** for the ratings **V-Y7-FV**, **NR**,**NC-17** and **PG-13**

- **We look at countries where films and television programs have been produced**

> Let us keep in mind that some films(tv shows and movies) as seen previously, are produced in several countries, so we donot have a necessarily mutually exclusive event for a film
> The grand total of films here is not eqaul to the total number of movies (6131) or tv-shows (2676)
> Let us not forget that we had 831 records with no country values

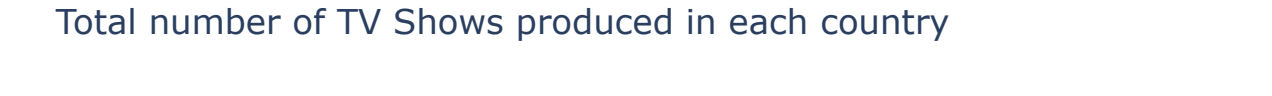
In [83]:
```
import plotly.express as px
import plotly.offline as pyo
pyo.init_notebook_mode()
fig = px.choropleth(data_choro[data_choro["type"]== "Movie"],
                    locations="country_iso_code", # colunm containing ISO 3166 country c
                    color="total", # column by which to color-code
                    hover_name="index", # column to display in hover information
```

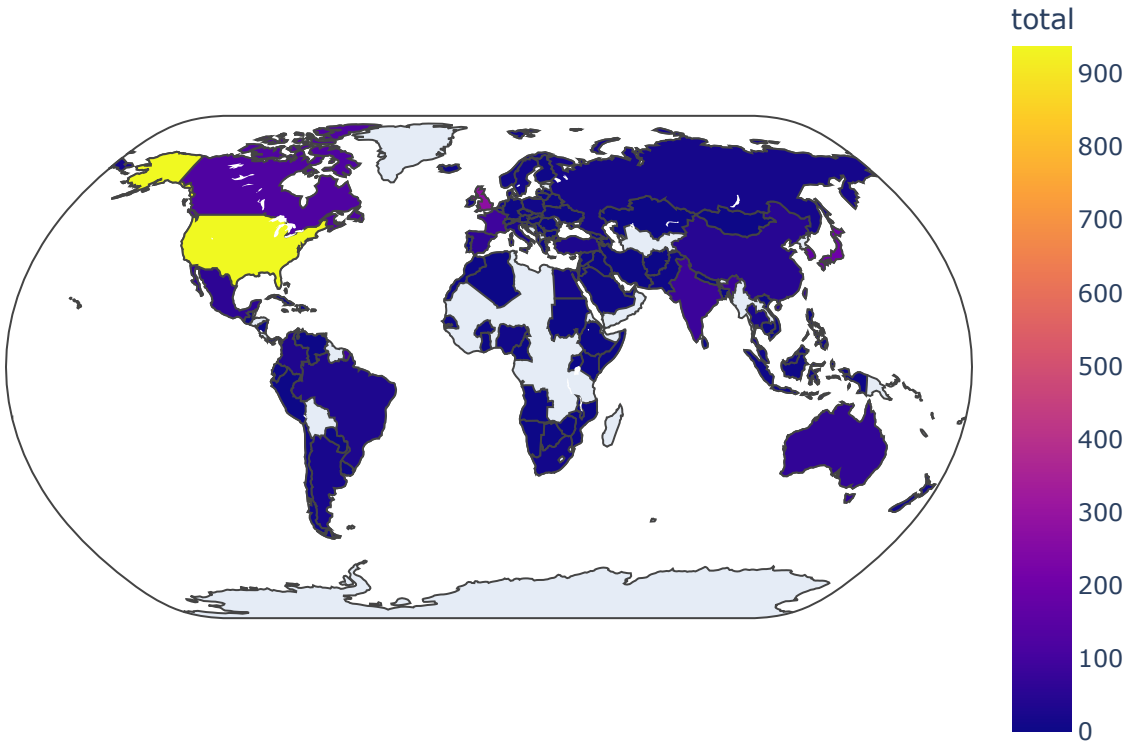
```
                    color_continuous_scale=px.colors.sequential.Plasma)

fig.update_layout(
    # add a title text for the plot
    title_text = 'Total number of movies produced in each country',
    #geo_scope = 'africa', # can be set to north america | south america | africa | asia
    geo = dict(projection={'type':'natural earth'}) # by default, projection type is set
)
fig.show()
```



In [84]:

```
import plotly.express as px
import plotly.offline as pyo
pyo.init_notebook_mode()
fig = px.choropleth(data_choro[data_choro["type"]== "TV Show"],
                    locations="country_iso_code", # colunm containing ISO 3166 country c
                    color="total", # column by which to color-code
                    hover_name="index", # column to display in hover information
                   color_continuous_scale=px.colors.sequential.Plasma)

fig.update_layout(
    # add a title text for the plot
    title_text = 'Total number of TV Shows produced in each country',
    # geo_scope = 'north america', # can be set to north america | south america | afric
    geo = dict(projection={'type':'natural earth'}) # by default, projection type is set
)
fig.show()
```





- **What can we say?**

> We have **more movies** produced in the **United states**, followed by **India**, **Canada**, etc. **More tv-shows** in the **United States**, followed by **United Kingdom**, **Canada**, **Japan**, **India**, etc.