**TITLE PAGE**:

• **Project Title**: “Netflix Data: Cleaning, Analysis and Visualization”

• **Project Info**: Excel, python, Jupiter Notebook

• **Domain**: Data Analytics

• **Data Set Source**: **NETFLIX DATA SET**

**PROJECT OVERVIEW**:

• To clean messy data, explore the patterns, visualize insights.

• **Scope**:

Cover content types, release years, genres, countries etc.

• Size of the data set using rows and columns.

**DATA CLEANING PROCESS**:

• Handling missing values

• Removing duplicates

• Standardizing formats

• Splitting /Merging columns

• Correcting inconsistent data

**VISUALIZATIONS: Using charts and graphs**

• Bar chart

• Line chart

• Pie chart

• Word cloud, Geographic Map

**NETFLIX DATA CLEANING**

The Netflix Data Cleaning and Analysis project involves preprocessing a dataset by handling missing values, standardizing formats, and extracting key features. It then analyses trends in content type, genre, country, ratings, and yearly growth using Python and visualization tools.

**CREATING THE DATA :**

**Step1: Importing the library function and accessing the data**

import pandas as pd

import numPy as np

import matplotlib.pyplot as plt *#accesing the library function*

df=pd.read\_csv(r"C:\Users\user\Downloads\netflix\_titles.csv\netflix\_titles.csv") *#importing csv file*

df



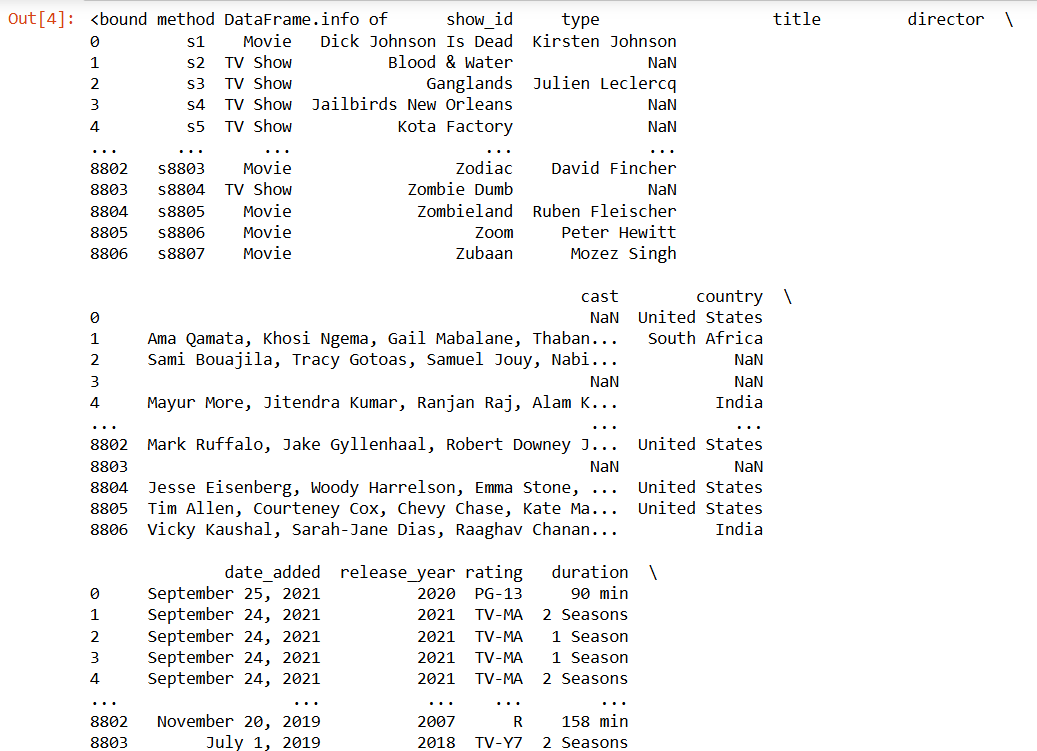
**CLEANING PROCESS :**

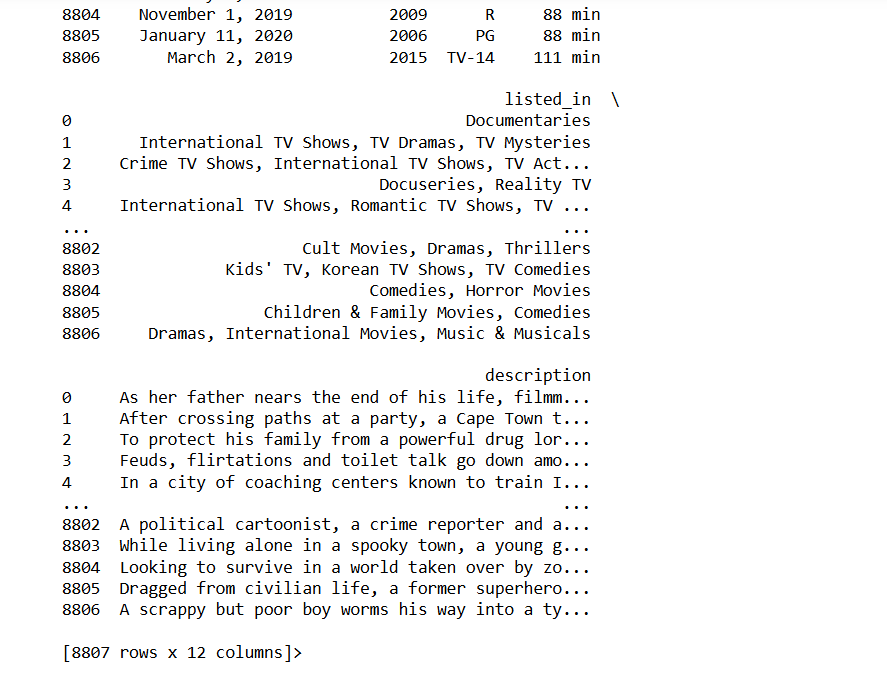
**Step2: Data cleaning**

df.shape *# To know size of the data*

**output:(**8807, 12)

df.info *# shows the range of data*

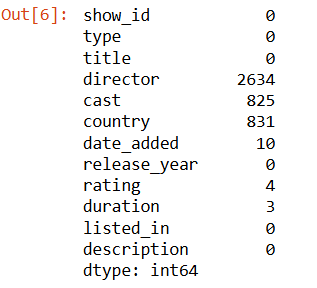




df.duplicated().sum() *# to check the no of duplicates*

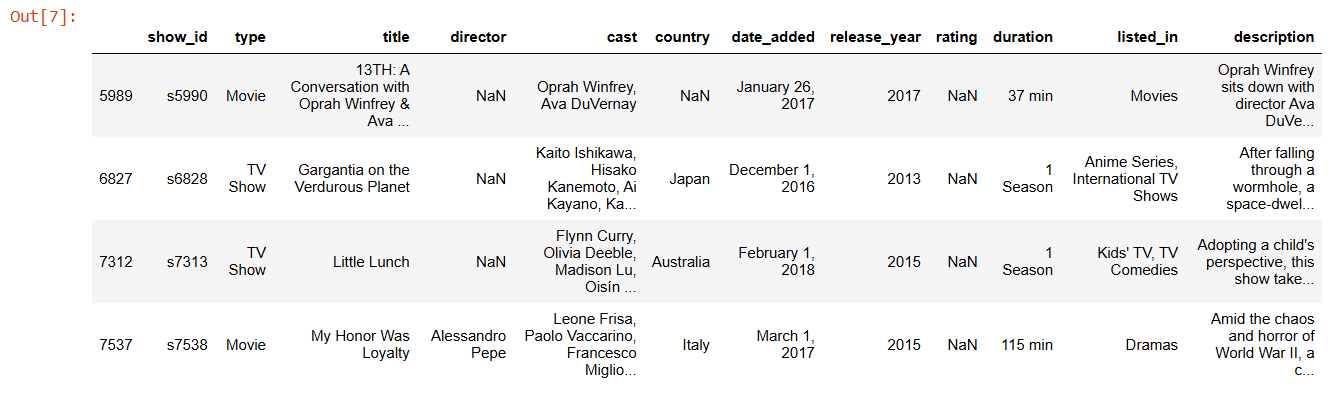
**output :** 0

df.isna().sum() *# to check the null values in each row*



**Step 3: modifying respect following rows and column using functions**

df[df['rating'].isna()] *#to specified column*



*# accessing the specific rating to each rows*

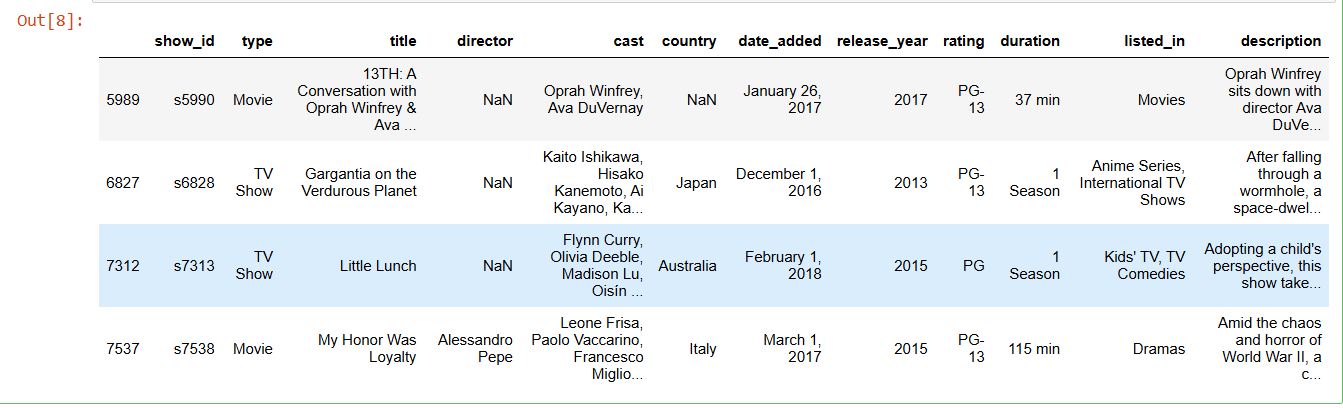
df.loc[df['show\_id'] == 's5990', 'rating'] = 'PG-13'

df.loc[df['show\_id'] == 's6828', 'rating'] = 'PG-13'

df.loc[df['show\_id'] == 's7313', 'rating'] = 'PG'

df.loc[df['show\_id'] == 's7538', 'rating'] = 'PG-13'

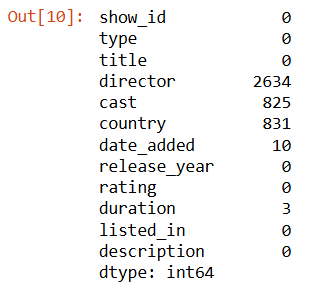
df[df['show\_id'].isin(['s5990','s6828','s7313','s7538'])]\



df[df['rating'].isna()]



df.isna().sum()



*#fill missing values in the column of your DataFrame with the most frequent (mode) value found in that column*

df['director'] = df['director'].fillna('Unspecified')

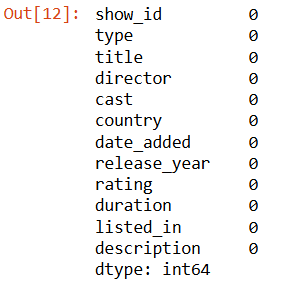
df['cast'] = df['cast'].fillna('Unknown')

df['country'] = df['country'].fillna(df['country'].mode()[0])

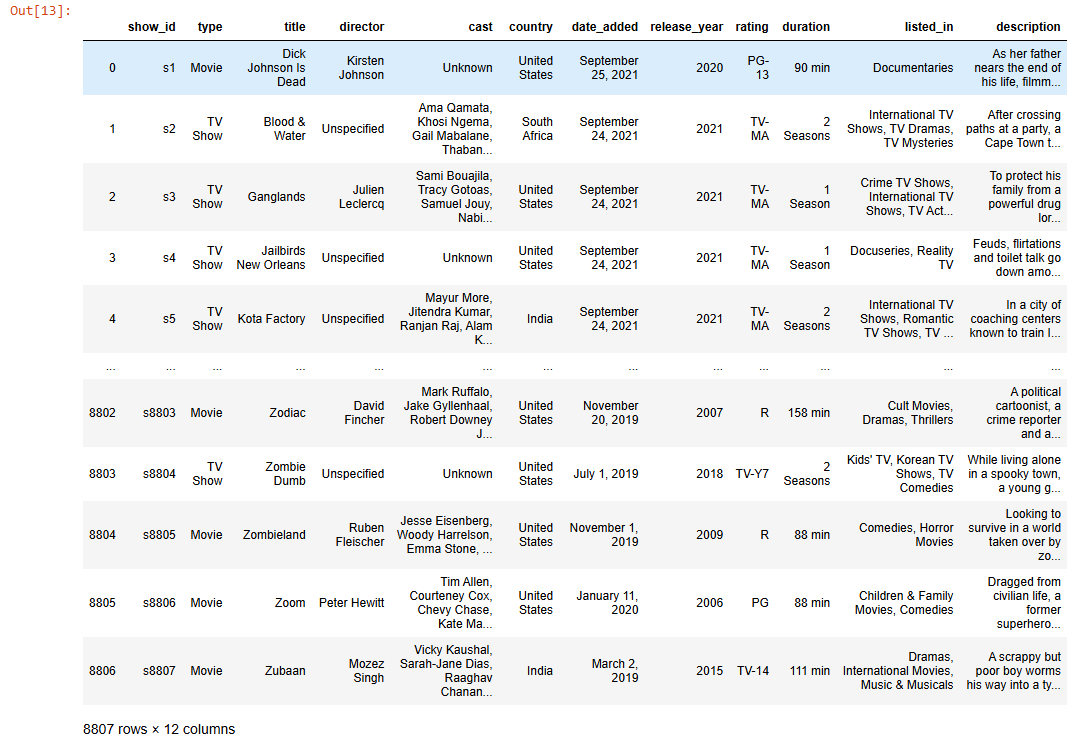
df['date\_added'] = df['date\_added'].fillna(df['date\_added'].mode()[0])

df['duration']= df['duration'].fillna(df['duration'].mode()[0])

df.isna().sum()



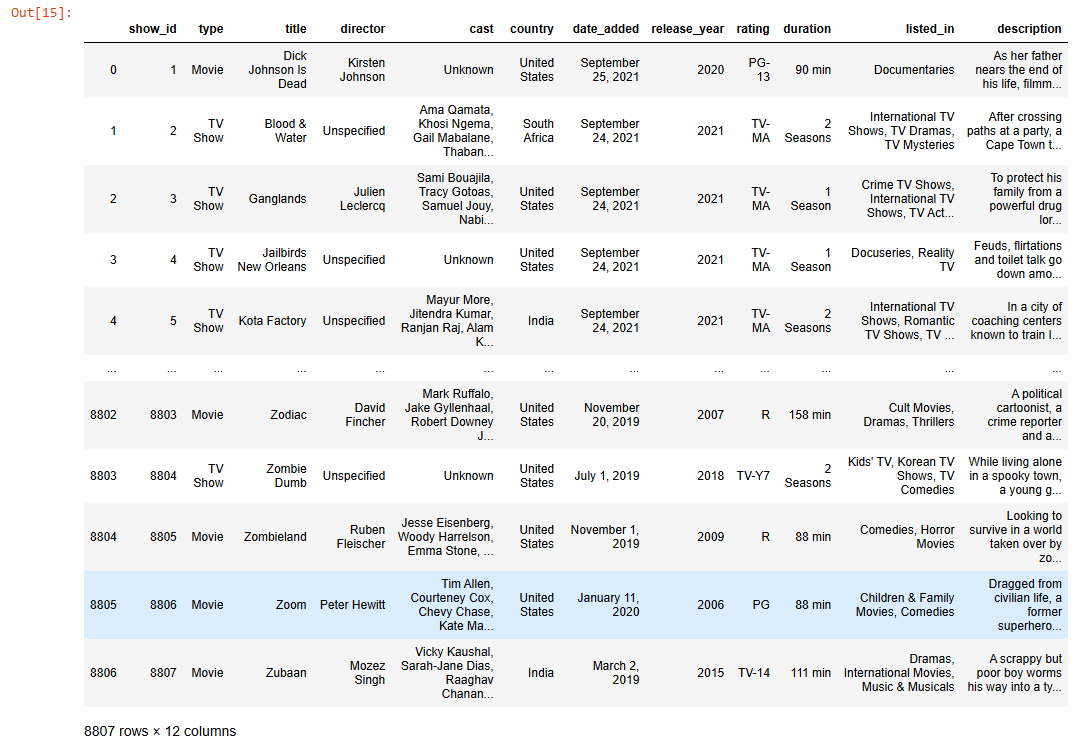
df



*# replacing the specific column*

df['show\_id'] = df['show\_id'].str.replace('s','')

df



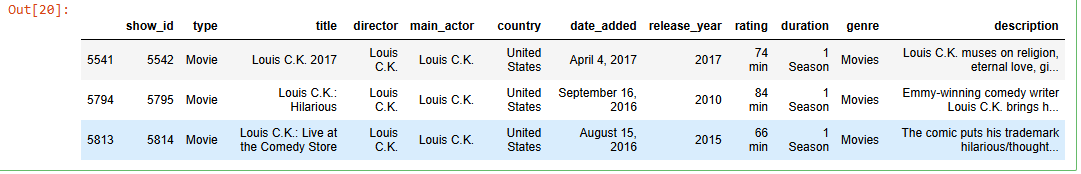
df['cast'] = df['cast'].str.split(',').str[0] # to seperate into words

df.rename(columns = {"cast":"main\_actor"}, inplace = True) #to replace the new string

df.rename(columns={"listed\_in":"genre"}, inplace = True)

df['genre'] = df['genre'].str.split(',').str[0]

df[df['show\_id'].isin(['5542','5795','5814'])]# to check



df[df['director'].str.contains(",", case=False, na=False)]

filtered\_rows = df[df['director'].str.contains(",", na=False)]

df.loc[filtered\_rows.index, 'director'] = filtered\_rows['director'].str.split(",", n=1).str[0]

df



**ANALYSIS AND VISUALIZATION :**

*#* ***The total no of titles in the dataset***

total titles= df['show\_id'].count()

print('total number of ' + total\_titles. astype(str) + ' titles')

output: total number of 8807 titles

***# accessing no of movie’s / tv shows***

df\_type = pd.DataFrame(df.groupby('type')['show\_id'].count()).reset\_index()

df\_type.columns = ['Type','Number of titles']

vals = pd.Series(df\_type['Number of titles'].values)

colors=['blue', 'crimson']

plt.bar(df\_type['Type'],height = df\_type['Number of titles'], color = colors)

plt.xlabel('Type')

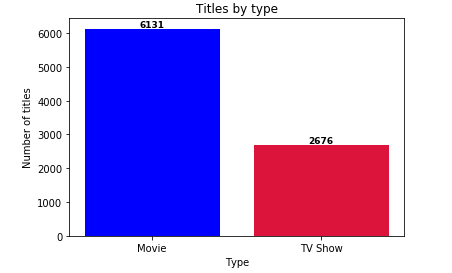
plt.ylabel('Number of titles')

plt.title('Titles by type')

for index,value in enumerate(vals):

plt.text(index, value, str(value), ha='centre', va='bottom', fontsize= 9, fontweight='semibold')

plt.show()



*# Percentage of movie and tv shows from the total number of titles*

movies\_percentage = ((df[df['type'] == 'Movie']['show\_id'].count() / total\_titles) \* 100).round(0).astype(str)

tv\_shows\_percentage = ((df[df['type'] == 'TV Show']['show\_id'].count() / total\_titles) \* 100).round(0).astype(str)

colors=['blue', 'crimson']

slices = np.array([movies\_percentage,tv\_shows\_percentage])

pie\_labels=["Movie","TV Show"]

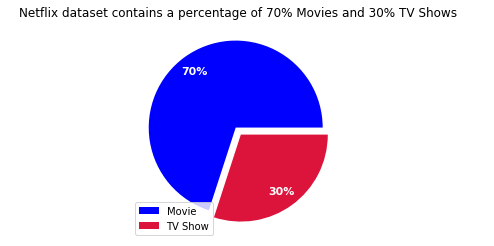
explode = (0.05, 0.05)

plt.pie(slices, labels=pie\_labels , autopct='%1.0f%%', colors= colors, explode=explode, pctdistance=0.80, textprops={'color': 'white', 'weight': 'bold', 'fontsize': 11})

plt.legend(labels=pie\_labels, loc='lower left')

plt.title('Netflix dataset contains a percentage of ' + '%d' %movies\_percentage.astype(float) + '% Movies and ' + '%d' %tv\_shows\_percentage.astype(float) + '% TV Shows')

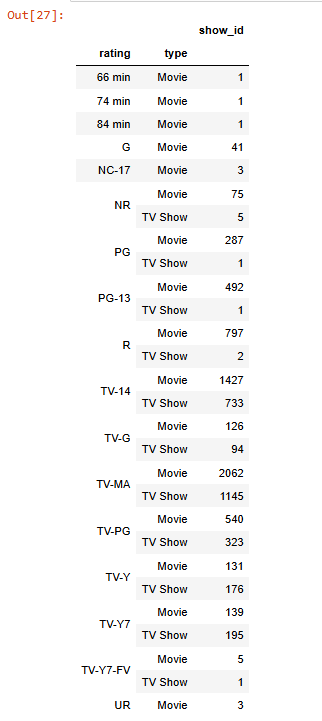
plt.show()



*# Ratings by type ( no of movies/tv shows per rating type)*

df\_titles\_by\_rating\_and\_type = pd.DataFrame(df.groupby(['rating','type'])['show\_id'].count()) #I'm bringing it to data cleaning for verification

df\_titles\_by\_rating\_and\_type



*# Titles by genre (top 5)*

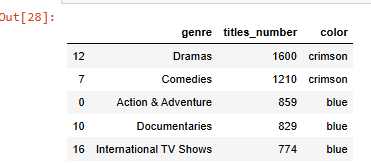
df\_movies\_by\_genre = pd.DataFrame(df.groupby('genre')['show\_id'].count()).reset\_index()

df\_movies\_by\_genre.columns = ['genre','titles\_number']

df\_movies\_by\_genre = df\_movies\_by\_genre.sort\_values('titles\_number',ascending=False).head(5)

df\_movies\_by\_genre['color'] = df\_movies\_by\_genre['titles\_number'].apply(lambda x:'blue' if x < 1000 else 'crimson')

df\_movies\_by\_genre



plt.bar(df\_movies\_by\_genre['genre'],df\_movies\_by\_genre['titles\_number'], color = df\_movies\_by\_genre['color'])

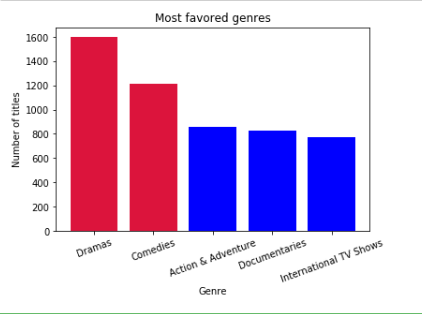
plt.xlabel('Genre')

plt.ylabel('Number of titles')

plt.title('Most favored genres')

plt.xticks(rotation = 20)

plt.show()



*# Titles by country*

df\_titles\_by\_country = df.groupby('country')['show\_id'].count().reset\_index()

df\_titles\_by\_country.columns = ['country', 'title no']

*#keep the first value before the comma*

*#replace the comma*

*#Extract the first value before the first comma*

df\_titles\_by\_country['country'] = df\_titles\_by\_country['country'].str.split(',').str[-1]

df\_titles\_by\_country = df\_titles\_by\_country.sort\_values(by='title no', ascending= False).head(10)

top\_10\_countries\_desc = df\_titles\_by\_country.sort\_values(by='title no', ascending=True)

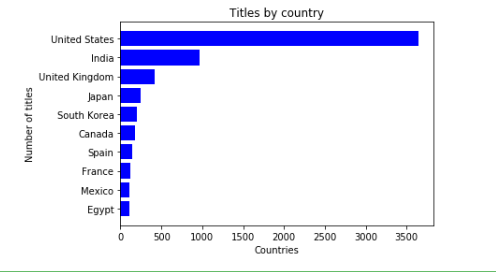
plt.barh(top\_10\_countries\_desc['country'], top\_10\_countries\_desc['title no'], color='blue')

plt.xlabel('Countries')

plt.ylabel('Number of titles')

plt.title('Titles by country')

plt.show()



*# Titles per duration (title per duration and type)*

df\_titles\_by\_duration = pd.DataFrame(df.groupby(['type','duration'])['show\_id'].count()).reset\_index()

df\_titles\_by\_duration.columns = ['type','duration','number of titles']

df\_titles\_by\_duration

movies\_data = df\_titles\_by\_duration[df\_titles\_by\_duration['type'] == 'Movie']

movies\_data\_sorted = movies\_data.sort\_values(by='number of titles', ascending= False).head(10)# de revenit cu sortarea

top\_10\_movies\_desc = movies\_data\_sorted.sort\_values(by='number of titles', ascending=True)

tv\_shows\_data = df\_titles\_by\_duration[df\_titles\_by\_duration['type']=='TV Show']

tv\_show\_data\_sorted = tv\_shows\_data.sort\_values(by='number of titles', ascending= False).head(10)

top\_10\_tv\_show\_desc = tv\_show\_data\_sorted.sort\_values(by='number of titles', ascending=True)

colors\_movies = top\_10\_movies\_desc['number of titles'].apply(lambda y: 'blue' if y < 140 else 'crimson')

colors\_tv\_shows = top\_10\_tv\_show\_desc['number of titles'].apply(lambda y: 'blue' if y < 50 else 'crimson')

plt.figure(figsize=(12, 7))

plt.subplot(1, 2, 1) # row 1, col 2 index 1

plt.barh(top\_10\_movies\_desc['duration'], top\_10\_movies\_desc['number of titles'], color=colors\_movies)

plt.title("Top 10 Movies Length")

plt.xlabel('X-axis ')

plt.ylabel('Y-axis ')

plt.subplot(1, 2, 2) # index 2

plt.barh(top\_10\_tv\_show\_desc['duration'], top\_10\_tv\_show\_desc['number of titles'], color=colors\_tv\_shows)

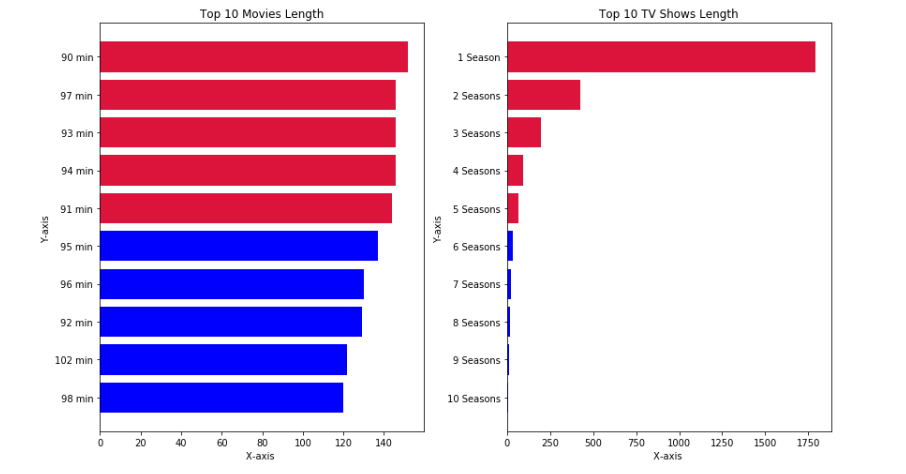
plt.title("Top 10 TV Shows Length")

plt.xlabel('X-axis ')

plt.ylabel('Y-axis ')

plt.tight\_layout()

plt.show()



*# A timeline on how many titles Netflix gathered from 2010 till now*

df\_titles\_per\_year = df.groupby('release\_year')['show\_id'].count().reset\_index()

df\_titles\_per\_year.columns = ['Year','Number of titles']

df\_titles\_after\_1997 = df\_titles\_per\_year[(df\_titles\_per\_year['Year'] > 1997)]

plt.plot(df\_titles\_after\_1997['Year'], df\_titles\_after\_1997['Number of titles'], color='black',linewidth=3)

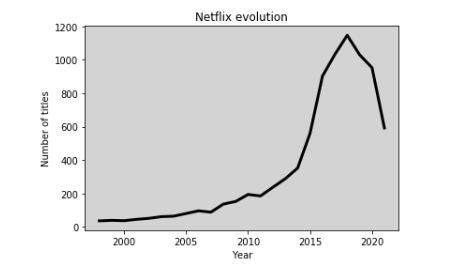
plt.xlabel('Year')

plt.ylabel('Number of titles')

plt.title('Netflix evolution')

plt.gca().set\_facecolor('lightgray')

plt.show()



titles\_per\_director=

df.groupby('director')['show\_id'].count().astype(int).reset\_index()

titles\_per\_director.columns = ['Director','Titles No']

director\_titles = titles\_per\_director.sort\_values(['Titles No','Director'], ascending=False).iloc[1:6]

director\_titles = director\_titles.sort\_values('Titles No', ascending = True)

plt.barh(director\_titles['Director'], director\_titles['Titles No'], color = 'blue')

plt.title('Top 5 directors by number of Titles')

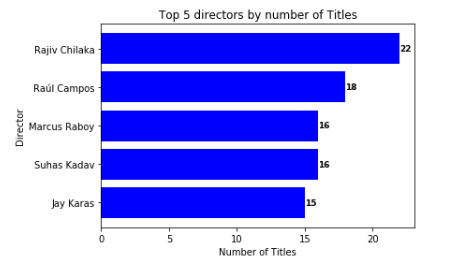
plt.ylabel('Director')

plt.xlabel('Number of Titles')

for index,value in enumerate(director\_titles['Titles No']):

plt.text(value, index, str(value), ha='left', va='centre', fontsize= 9, fontweight='semibold')

plt.show()



**CONCLUSION:**

* **Summary of process +Key insights+Final refletion+Future scope**