Capstone Project

Github repository: https://github.com/MalihehGaroosiha/Capstone-project.git

In this project, I am applying content-based recommender systems method to the Netflix dataset

In [232]:

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
```

In [233]:

```
import plotly.express as px
```

In [234]:

```
import os
os.getcwd()
```

Out[234]:

```
'/content'
```

Uploading Dataset¶

In [235]:

```
from google.colab import files
uploaded=files.upload()
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving netflix_titles.csv to netflix_titles (3).csv
```

In [236]:

```
df=pd.read_csv("netflix_titles.csv")
```

In [238]:

```
df_orig = df.copy()
```

In [237]:

```
df.head(2)
```

Out[237]:

	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	25-Sep-21	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	24-Sep-21	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...

In [239]:

```
df.tail(5)
```

Out[239]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert	United States	20-Nov-19	2007	R	158 min	Cult Movies, Dramas, Thrillers	A political cartoonist, a crime reporter

					Downey J...							and a...
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	1-Jul-19	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	1-Nov-19	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	11-Jan-20	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	2-Mar-19	2015	TV-14	111 min	Dramas, International Movies, Music & Musicals	A scrappy but poor boy worms his way into a ty...

Finding data type¶

In [240]:

```
df.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
```

Descriptive statistics for numeric columns¶

In [241]:

```
print(df.describe())
```

```
       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
```

number of rows and columns¶

In [242]:

```
df.shape
```

Out[242]:

```
(8807, 12)
```

Exploring Dataset¶

1.Finding columns with missing values:director,cast, country,date_added,rating, duration

In [243]:

```
df.count()
```

Out[243]:

```
show_id         8807
type            8807
title           8807
director        6173
cast            7982
country         7976
date_added      8797
release_year    8807
rating          8803
duration        8804
listed_in       8807
description     8807
dtype: int64
```

List of columns name¶

In [244]:

```
df.columns
```

Out[244]:

```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country',
'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

Unique variables values with plots¶

In [245]:

```
df.sample()
```

Out[245]:

	sho	ty	title	dire	ca	cou	date_a	release	rat	dura	listed_i	descri

	w_id	pe		ctor	st	ntry	dded	_year	ing	tion	n	ption
4841	s4842	TV Show	November 13: Attack on Paris	NaN	NaN	France	1-Jun-18	2018	TV-MA	1 Season	Crime TV Shows, Docuseries, International TV S...	Survivors and first responders share personal ...

In [246]:

```
df["release_year"].unique()
```

Out[246]:

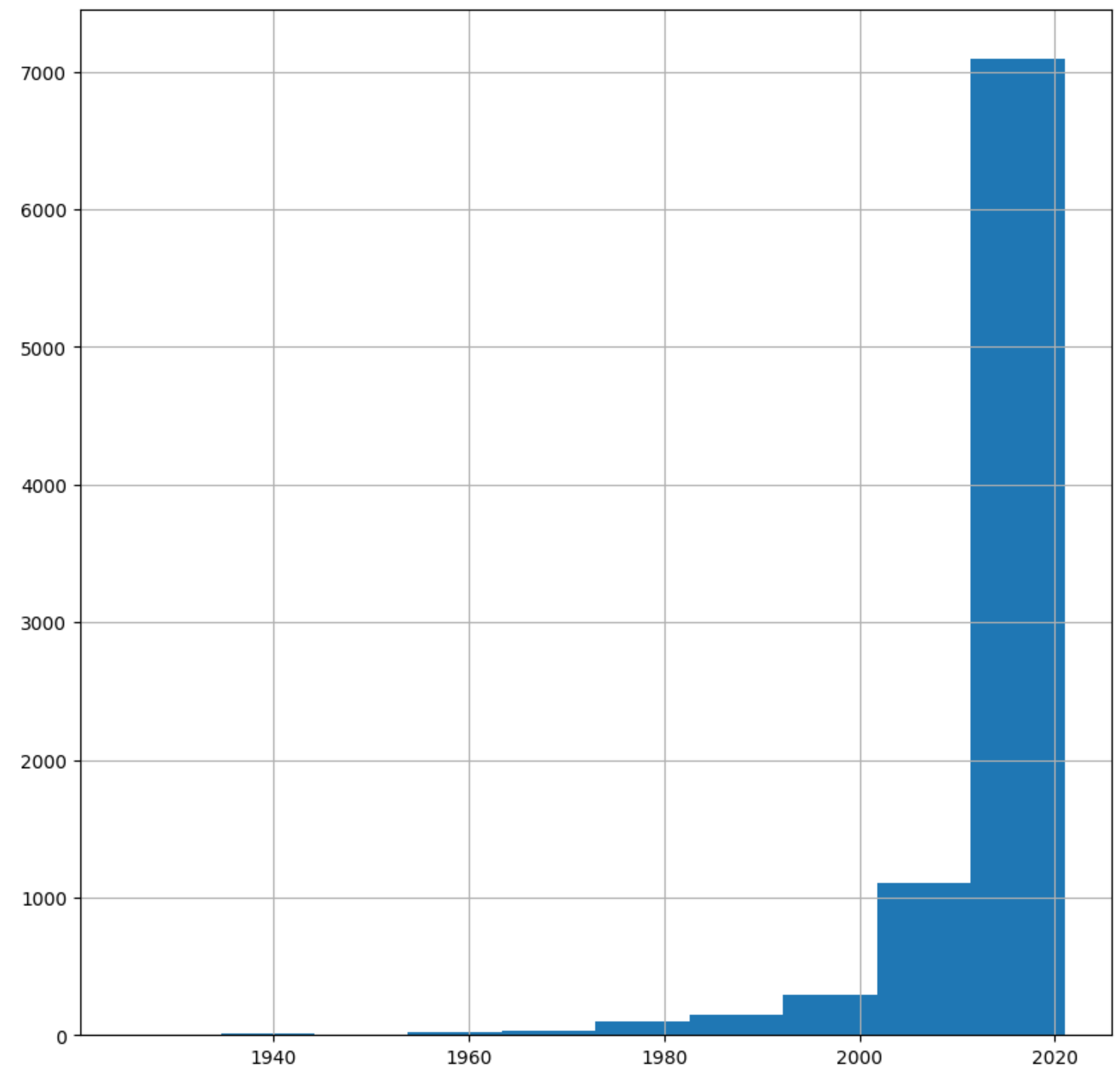
```
array([2020, 2021, 1993, 2018, 1996, 1998, 1997, 2010, 2013, 2017, 1975,
       1978, 1983, 1987, 2012, 2001, 2014, 2002, 2003, 2004, 2011, 2008,
       2009, 2007, 2005, 2006, 1994, 2015, 2019, 2016, 1982, 1989, 1990,
       1991, 1999, 1986, 1992, 1984, 1980, 1961, 2000, 1995, 1985, 1976,
       1959, 1988, 1981, 1972, 1964, 1945, 1954, 1979, 1958, 1956, 1963,
       1970, 1973, 1925, 1974, 1960, 1966, 1971, 1962, 1969, 1977, 1967,
       1968, 1965, 1946, 1942, 1955, 1944, 1947, 1943])
```

In [247]:

```
df["release_year"].hist()
```

Out[247]:

```
<Axes: >
```



for the country bar chart, the primary country selected¶

In [ ]:

```
df["country"].unique()
```

In [249]:

```
df["country"].nunique()
```

Out[249]:

```
748
```

In [250]:

```
import numpy as np
```

```
# Define a function to split the country values
def split_country(x):
    if isinstance(x, str):  # Check if x is a string
        return x.split(',')[0]
    elif isinstance(x, float) and np.isnan(x):  # Check if x is NaN
        return np.nan
    else:
        return x  # Return the original value if it's not a string or NaN

# Apply the split_country function to the 'country' column
df_orig ['country'] = df_orig ['country'].apply(split_country)
```

In [251]:

```
df_orig ["country"].nunique()
```

Out[251]:

```
86
```

In [252]:

```
orginal_country_count=df_orig ["country"].value_counts()
orginal_country_count
```

Out[252]:

```
United States     3211
India             1008
United Kingdom     628
Canada             271
Japan              259
                  ... 
Namibia              1
Senegal              1
Luxembourg           1
Syria                1
Somalia              1
Name: country, Length: 86, dtype: int64
```

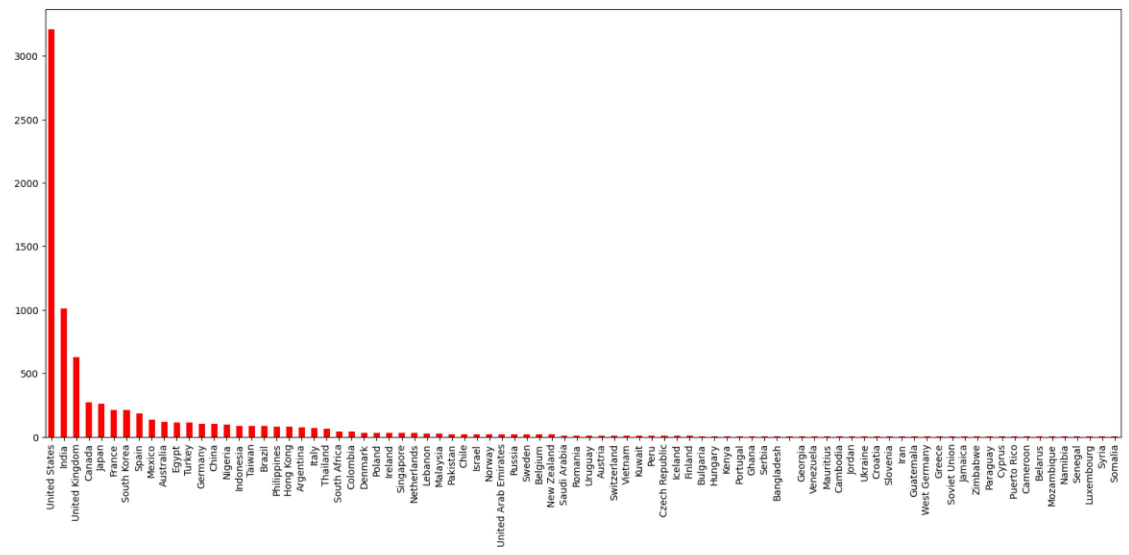
In [253]:

```
plt.figure(figsize=(20, 8))

orginal_country_count.plot(kind='bar', color='red')
```

Out[253]:

```
<Axes: >
```



Cheking rating from https://help.netflix.com/en/node/2064/ca.

In [254]:

```
df["rating"].unique()
```

Out[254]:

```
array(['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7', 'R',
       'TV-G', 'G', 'NC-17', '74 min', '84 min', '66 min', 'NR', nan,
       'TV-Y7-FV', 'UR'], dtype=object)
```

In [255]:

```
df["rating"].nunique()
```

Out[255]:

```
17
```

In [256]:

```
rating_count=df["rating"].value_counts()
rating_count
```

Out[256]:

```
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
```

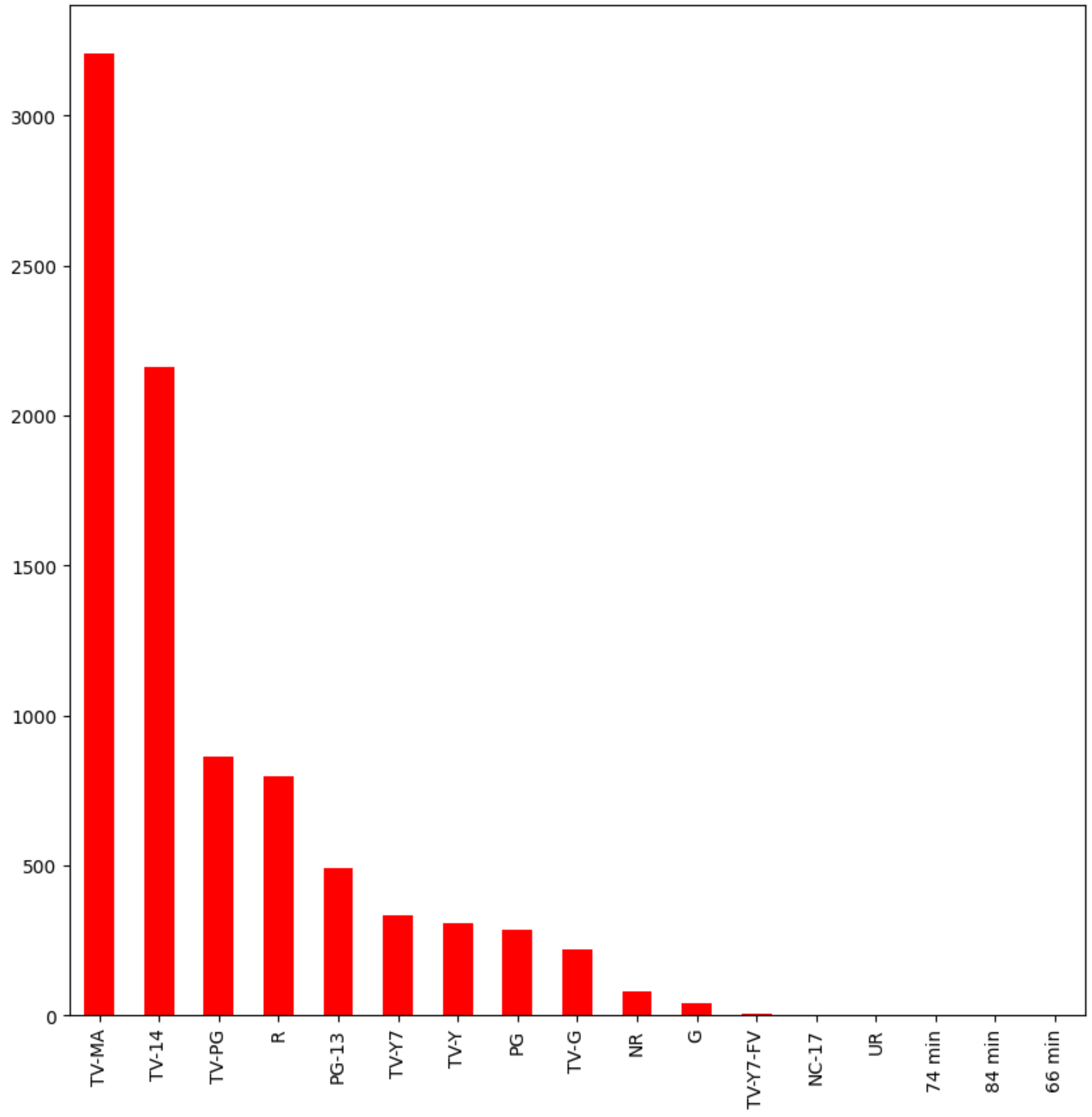
```
NR            80
G             41
TV-Y7-FV       6
NC-17          3
UR             3
74 min         1
84 min         1
66 min         1
Name: rating, dtype: int64
```

In [257]:

```
rating_count.plot(kind='bar', color='red')
```

Out[257]:

```
<Axes: >
```



In [258]:

```
df["type"].unique()
```

Out[258]:

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

In [259]:

```
type_count= df["type"].value_counts()
```

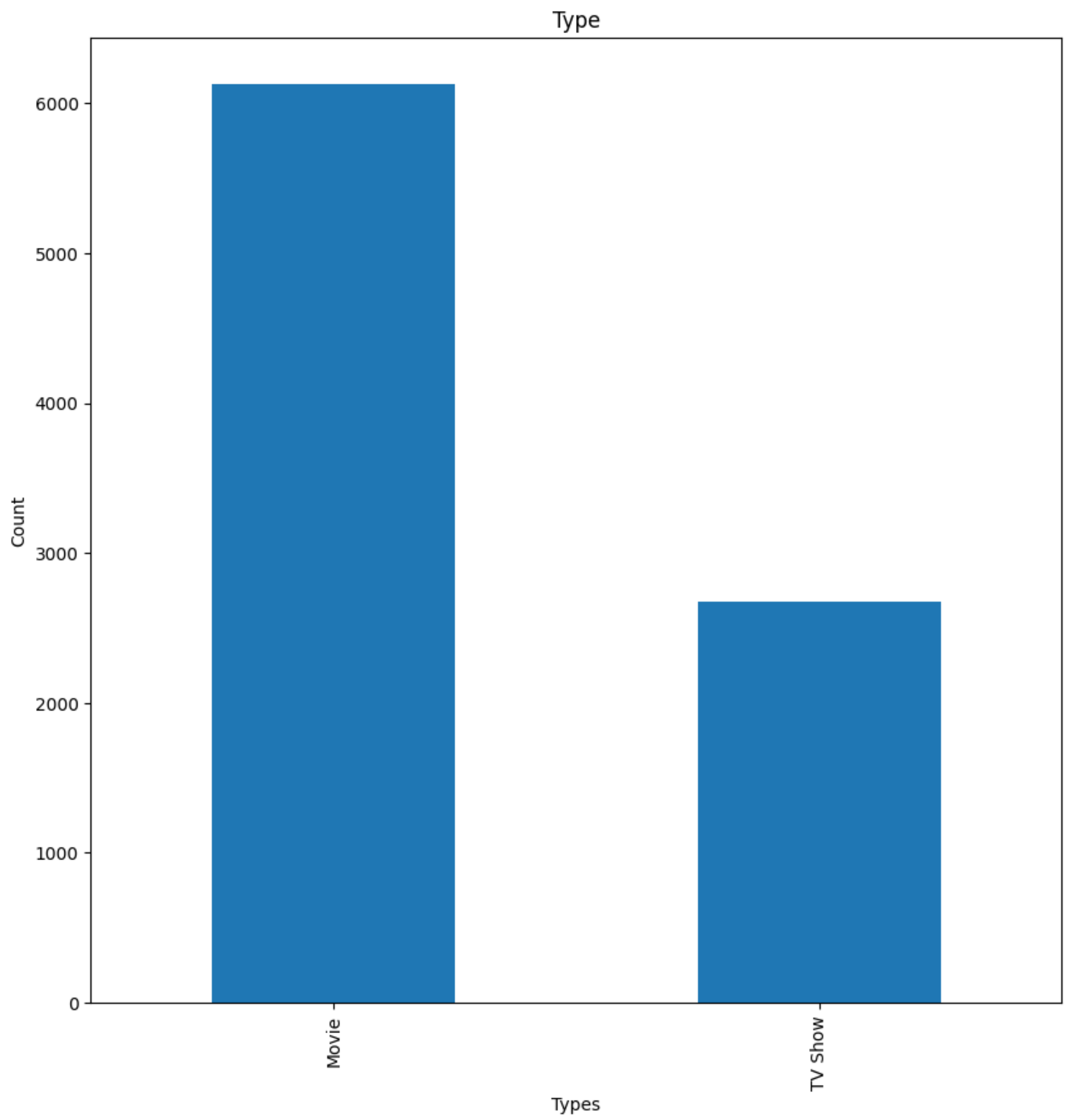
In [260]:

```
type_count.plot(kind='bar')
plt.xlabel('Types')
plt.ylabel('Count')
plt.title('Bar Plot of Type Counts')

plt.title('Type')
```

Out[260]:

```
Text(0.5, 1.0, 'Type')
```

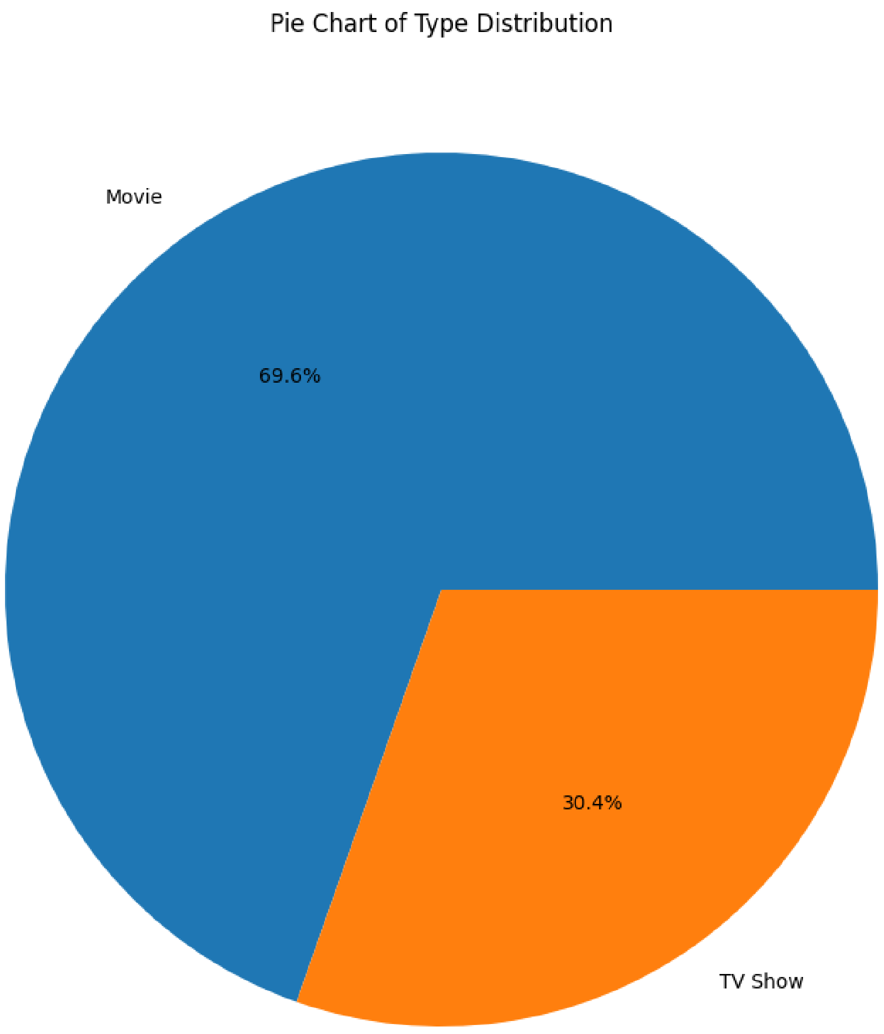


In [261]:

```
plt.pie(type_count, labels=type_count.index, autopct='%1.1f%%')
plt.title('Pie Chart of Type Distribution')
```

Out[261]:

```
Text(0.5, 1.0, 'Pie Chart of Type Distribution')
```

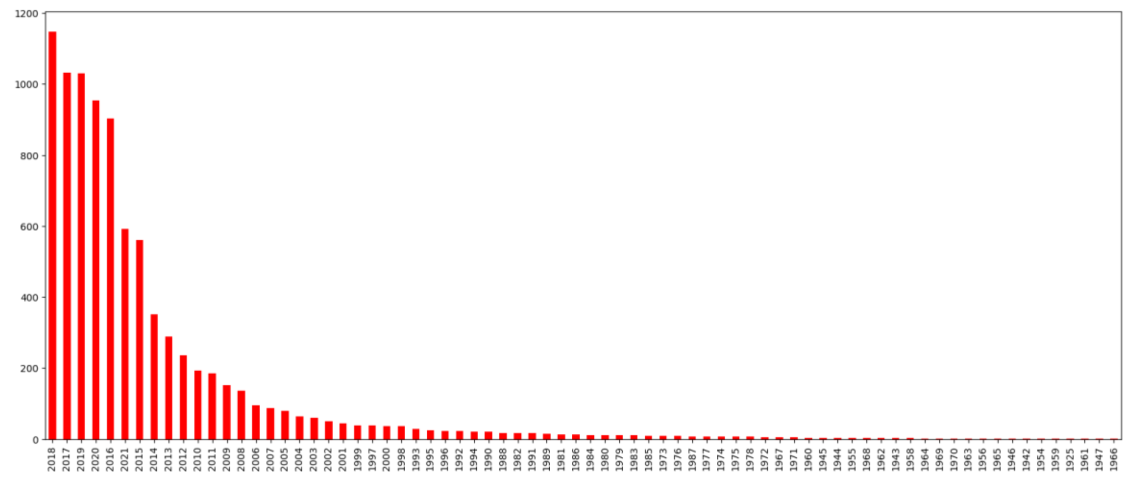


In [262]:

```
type_counts = df['release_year'].value_counts()
plt.figure(figsize=(20, 8))
type_counts.plot(kind='bar', color='red')
```

Out[262]:

```
<Axes: >
```



In [263]:

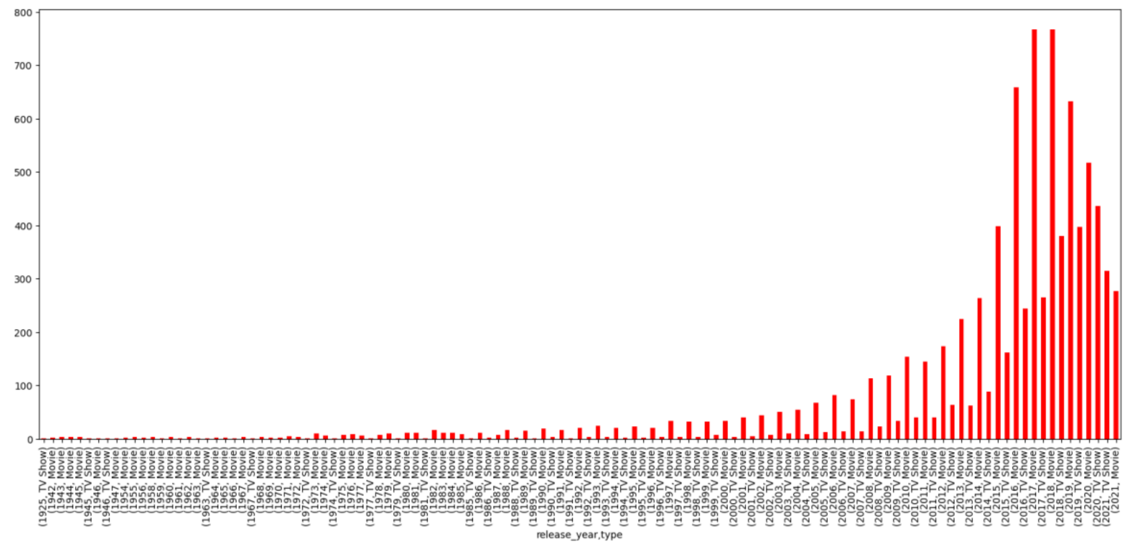
```
type_counts_by_year = df.groupby('release_year')['type'].value_counts()
```

In [264]:

```
plt.figure(figsize=(20, 8))
type_counts_by_year.plot(kind='bar', color='red')
```

Out[264]:

```
<Axes: xlabel='release_year,type'>
```



In [265]:

```
import numpy as np

# Define a function to split the country values
def split_list_in(x):
    if isinstance(x, str):  # Check if x is a string
```

```
        return x.split(',')[0]
    elif isinstance(x, float) and np.isnan(x):  # Check if x is NaN
        return np.nan
    else:
        return x  # Return the original value if it's not a string or NaN

# Apply the split_country function to the 'country' column
df_orig ['listed_in'] = df_orig ['listed_in'].apply(split_list_in)
```

In [266]:

```
orginal_listed_in_count=df_orig ["listed_in"].value_counts()
orginal_listed_in_count
```

Out[266]:

```
Dramas                            1600
Comedies                          1210
Action & Adventure                 859
Documentaries                      829
International TV Shows             774
Children & Family Movies           605
Crime TV Shows                     399
Kids' TV                           388
Stand-Up Comedy                    334
Horror Movies                      275
British TV Shows                   253
Docuseries                         221
Anime Series                       176
International Movies               128
TV Comedies                        120
Reality TV                         120
Classic Movies                      80
TV Dramas                           67
Thrillers                           65
Movies                              57
TV Action & Adventure               40
Stand-Up Comedy & Talk Shows        34
Romantic TV Shows                   32
Classic & Cult TV                   22
Anime Features                      21
Independent Movies                  20
Music & Musicals                    18
TV Shows                            16
Sci-Fi & Fantasy                    13
Cult Movies                         12
TV Horror                           11
Romantic Movies                      3
Spanish-Language TV Shows            2
LGBTQ Movies                         1
TV Sci-Fi & Fantasy                  1
```

```
Sports Movies                      1
Name: listed_in, dtype: int64
```

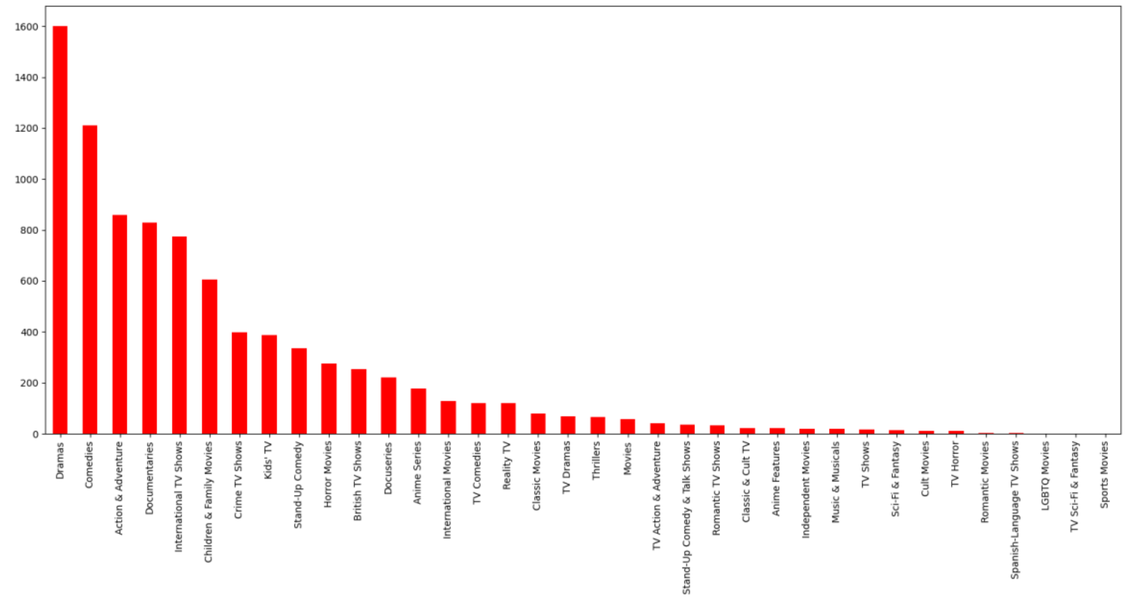
In [267]:

```
plt.figure(figsize=(20, 8))

orginal_listed_in_count.plot(kind='bar', color='red')
```

Out[267]:

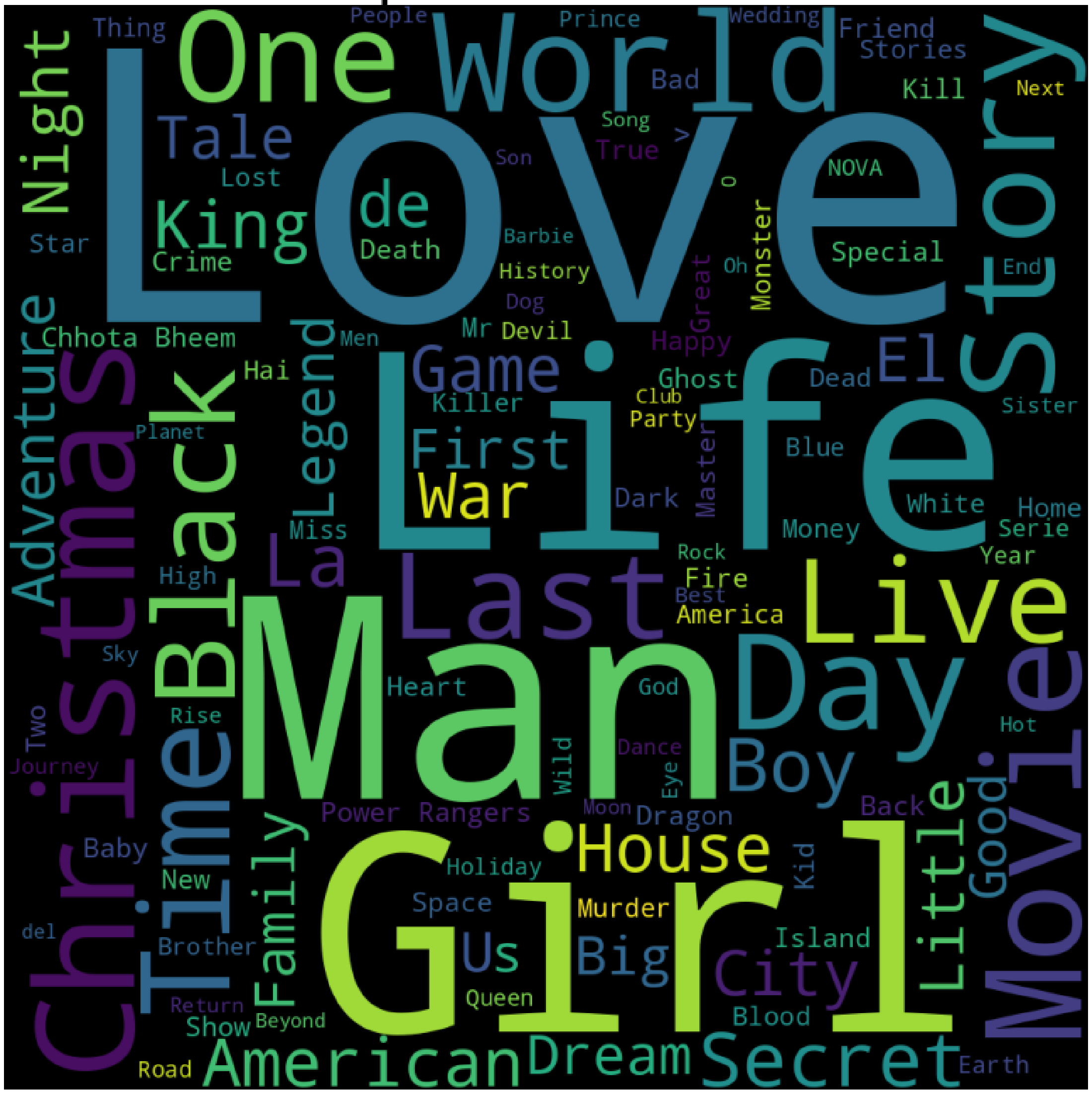
```
<Axes: >
```



In [268]:

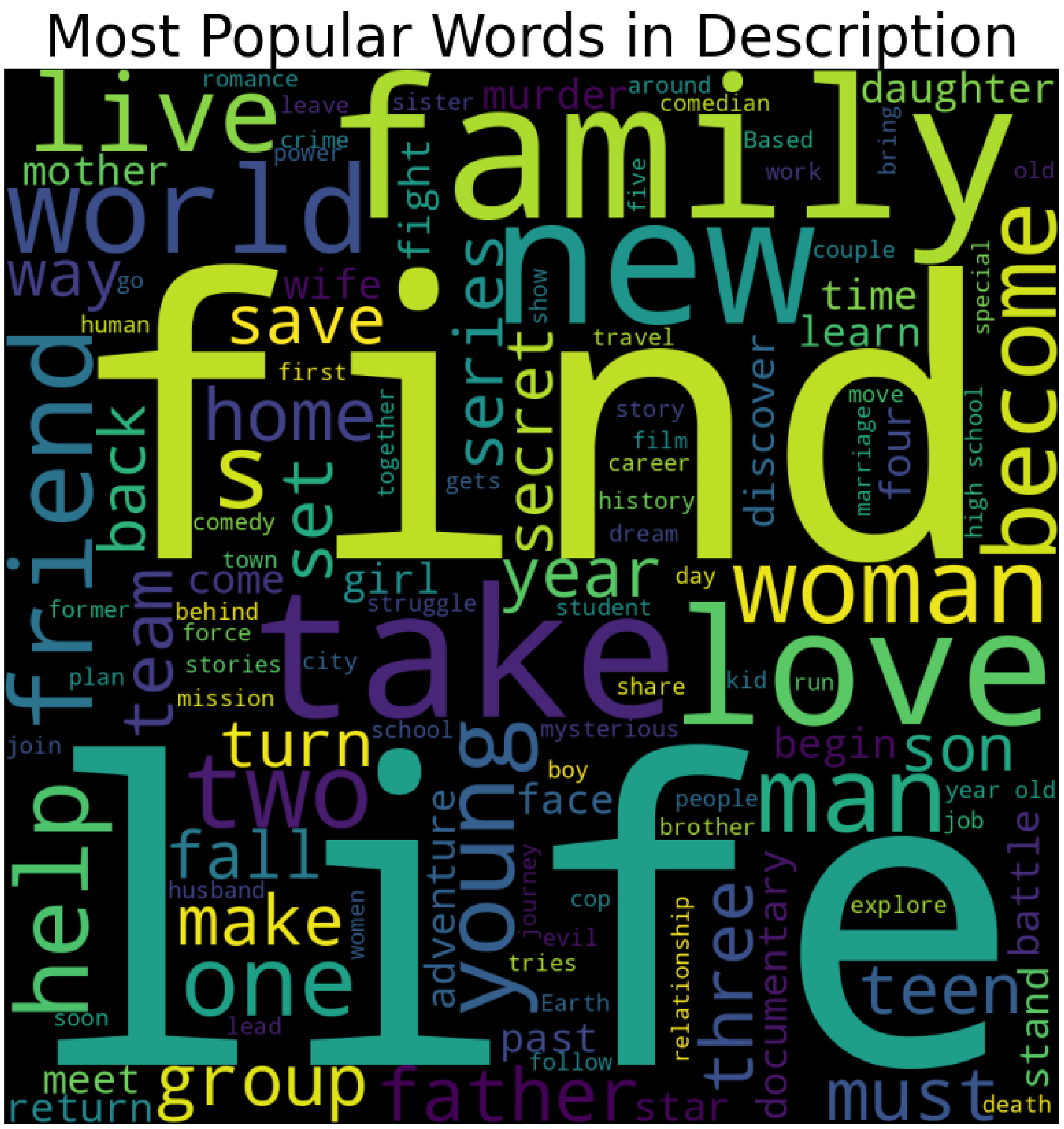
```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
plt.rcParams['figure.figsize'] = (10, 10)
wordcloud = WordCloud(stopwords=STOPWORDS,background_color = 'black', width =
1000,  height = 1000, max_words = 121).generate(' '.join(df['title']))
plt.imshow(wordcloud)
plt.axis('off')
plt.title('Most Popular Words in Title',fontsize = 30)
plt.show()
```

Most Popular Words in Title



In [269]:

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
plt.rcParams['figure.figsize'] = (10, 10)
wordcloud = WordCloud(stopwords=STOPWORDS,background_color = 'black', width =
1000,  height = 1000, max_words = 121).generate(' '.join(df['description']))
plt.imshow(wordcloud)
plt.axis('off')
plt.title('Most Popular Words in Description',fontsize = 30)
plt.show()
```



Cleaning Data¶

1.Handling missing value

In [270]:

```
df.isnull().sum()
```

Out[270]:

```
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 [271]:

```
# handling missing values
df['director'] = df['director'].fillna('')
df['country'] = df['country'].fillna('')
df['cast'] = df['cast'].fillna('')
df['date_added'] = df['date_added'].fillna(df['date_added'].mode()[0])
#df['rating'] = df['rating'].fillna(df['rating'].mode()[0])
df['rating'] = df['rating'].fillna(df['rating'].mode()[0])
df = df.dropna(subset=["duration"])
print('count of values')
print(df.isna().sum())
```

```
count of values
show_id         0
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64
```

2.Changing "date_added" column type from object to datetime64[

In [272]:

```
df['date_added'] = pd.to_datetime(df['date_added'])
```

3.Deleting Duplicated rows

In [273]:

```
df.duplicated().sum()
```

Out[273]:

```
0
```

Handling outlier¶

rating column has outlier('84 min') & ('66 min') & ('74 min')& ('NR')& ("UR") &'TV-Y7-FV'
I have filtered data according to deleting outlier.

In [274]:

```
filtered_df = df[(df["rating"] != '84 min') & (df["rating"] != '66 min') &
(df["rating"] != '74 min')& (df["rating"] != 'NR')& (df["rating"] != "UR")]

df = filtered_df
df.loc[df['rating'] == 'TV-Y7-FV', 'rating'] ='TV-Y7'
```

In [275]:

```
df["rating"].unique()
```

Out[275]:

```
array(['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7', 'R',
       'TV-G', 'G', 'NC-17'], dtype=object)
```

In [276]:

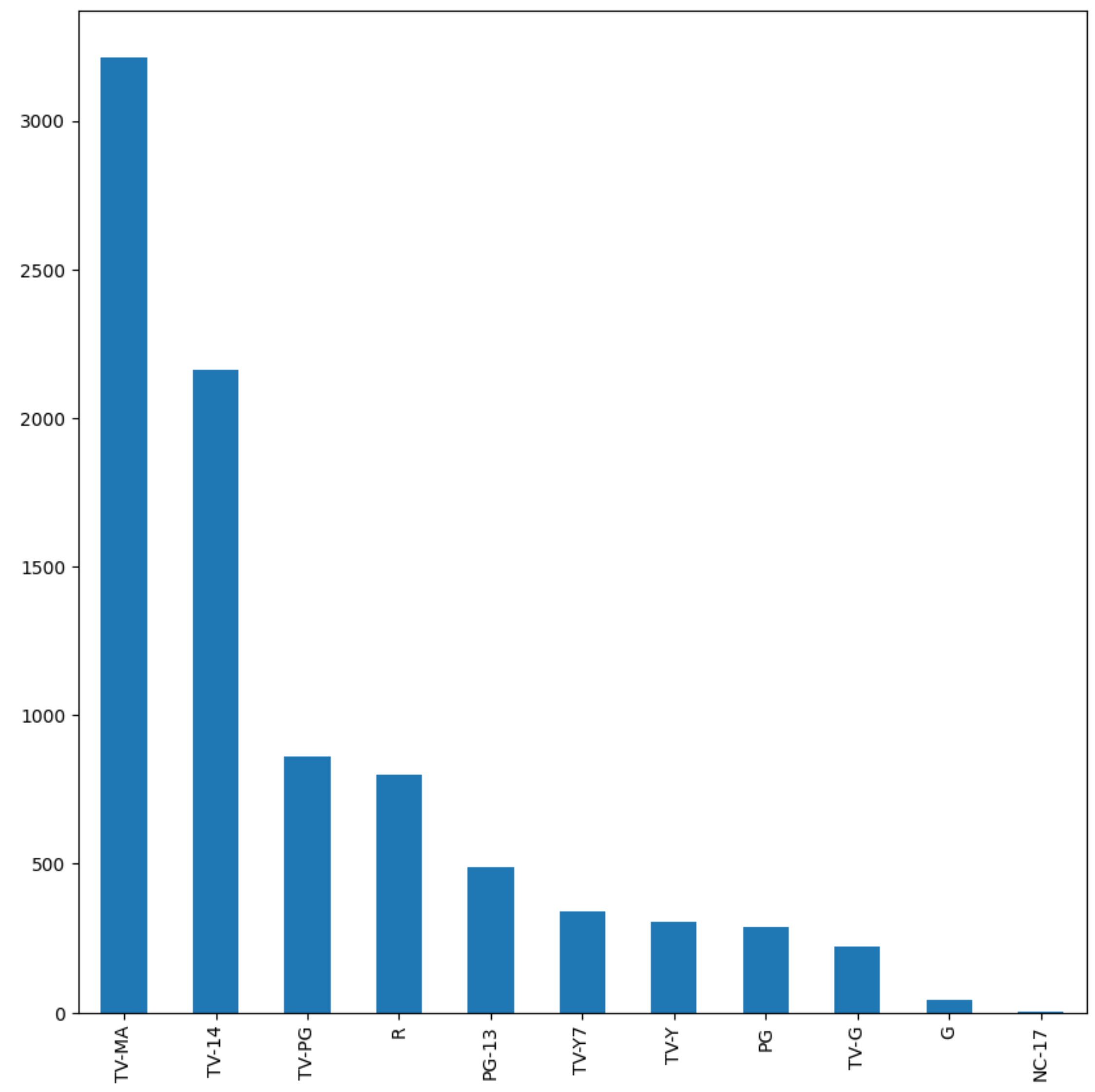
```
country_count=df["rating"].value_counts()
```

In [277]:

```
country_count.plot(kind="bar")
```

Out[277]:

```
<Axes: >
```



In [278]:

```
df.head()
```

Out[278]:

	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		United States	2021-09-25	2020	PG-13	90 min	Documentaries	As her father nears the end of his life,

												filmm...
1	s2	TV Show	Blood & Water		Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	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...		2021-09-24	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				2021-09-24	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory		Mayur More, Jitendra Kumar, Ranjan Raj, Alam	India	2021-09-24	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...

In [279]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8721 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8721 non-null   object
 1   type          8721 non-null   object
 2   title         8721 non-null   object
 3   director      8721 non-null   object
 4   cast          8721 non-null   object
 5   country       8721 non-null   object
 6   date_added    8721 non-null   datetime64[ns]
 7   release_year  8721 non-null   int64
 8   rating        8721 non-null   object
 9   duration      8721 non-null   object
 10  listed_in     8721 non-null   object
 11  description   8721 non-null   object
dtypes: datetime64[ns](1), int64(1), object(10)
memory usage: 885.7+ KB
```

Recommender Systems Content Base with variable "description"¶

In [280]:

```
df["description"].head()
```

Out[280]:

```
0    As her father nears the end of his life, filmm...
1    After crossing paths at a party, a Cape Town t...
2    To protect his family from a powerful drug lor...
3    Feuds, flirtations and toilet talk go down amo...
4    In a city of coaching centers known to train I...
Name: description, dtype: object
```

In [281]:

```
df["description"] = df["description"].str.lower()
```

In [ ]:

Constructing the required TF-IDF matrix by fitting and transforming the data TF-IDF matrix has 8721 rows (each row corresponds to a movie or TV show) and 18791 columns (each column represents a unique word from the text data).

In [282]:

```
#Here, you import the TfidfVectorizer class from the
sklearn.feature_extraction.text module. This class is used to convert a
collection of raw documents into a matrix of TF-IDF features.
from sklearn.feature_extraction.text import TfidfVectorizer
#vector space model
tfidf = TfidfVectorizer(stop_words='english')

#Constructing the required TF-IDF matrix by fitting and transforming the data
tfidf_matrix = tfidf.fit_transform(df['description'])

#Output the shape of tfidf_matrix
tfidf_matrix.shape
```

Out[282]:

```
(8721, 18791)
```

Importing linear_kernel: In this step, we import the linear_kernel function from the sklearn.metrics.pairwise module. The linear_kernel function is used to compute the cosine similarity between vectors.The cosine similarity measures the cosine of the angle between two vectors and is commonly used in text similarity tasks.it means that they have more similar if the cosine angle is 0.The linear_kernel function is applied to the TF-IDF matrix tfidf_matrix twice. This computes the dot product (inner product) between each pair of document vectors in tfidf_matrix, effectively calculating the cosine similarity between all pairs of documents. The resulting cosine_sim matrix is a symmetric matrix where cosine_sim[i][j] represents the cosine similarity between document i and document j. $\langle a,b\rangle=\|a\|\cdot\|b\|\cdot\cos(\theta)$

In [283]:

```
#using the linear_kernel function from sklearn.metrics.pairwise to calculate
cosine
from sklearn.metrics.pairwise import linear_kernel

#compute the cosine similarity matrix
cosine_sim = linear_kernel(tfidf_matrix,tfidf_matrix)
```

this code creates a Series indices where each unique value in the 'title' column of DataFrame df is mapped to its corresponding index value from df.index. This mapping allows you to quickly look up the index of a row in df based on its 'title'.

In [284]:

```
indices = pd.Series(df.index, index = df['title']).drop_duplicates()
```

This Python function recommendations takes a movie title as input and returns the top 10 movies that are most similar to the input movie based on cosine similarity scores

In [285]:

```
def recommendations(title, cosine_sim=cosine_sim):
    # Get the index of the movie that matches the title
    idx = indices[title]

    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    return df['title'].iloc[movie_indices]
```

Testing Recommender system¶

In [286]:

```
df["title"].head()
```

Out[286]:

```
0     Dick Johnson Is Dead
1            Blood & Water
2                Ganglands
3    Jailbirds New Orleans
4             Kota Factory
Name: title, dtype: object
```

In [287]:

```
recommendations("Dick Johnson Is Dead")
```

Out[287]:

```
4877                                    End Game
1066                                    The Soul
7506                                        Moon
5047                      The Cloverfield Paradox
5233    The Death and Life of Marsha P. Johnson
```

```
5494                                   Kazoops!
2674                                      Alelí
4241                  Secrets in the Hot Spring
4735                      Tere Naal Love Ho Gaya
2760              Kannum Kannum Kollaiyadithaal
Name: title, dtype: object
```

As seen here, the first suggestion is listed in documentaries too, but their ratings are different. The best suggestion should pay attention to the rating. In the next model, I will try to incorporate the rating into the model

In [288]:

```
df[(df["title"] == "Dick Johnson Is Dead")|(df["title"] == "End Game") |
(df["title"] == "The Soul")]
```

Out[288]:

	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		United States	2021-09-25	2020	PG-13	90 min	Documentaries	as her father nears the end of his life, filmm...
1066	s1067	Movie	The Soul	Cheng Wei-hao	Chang Chen, Janine Chang, Christopher Lee, Ank...	China, Taiwan	2021-04-14	2021	TV-MA	130 min	Dramas, International Movies, Thrillers	while investigating the death of a businessman...
4877	s4878	Movie	End Game	Rob Epstein, Jeffrey Friedman		United States	2018-05-04	2018	TV-PG	40 min	Documentaries	facing an inevitable outcome, terminally ill p...

Next Step is wite another program with more variables not just description. I made recommend system with this attributes "title","director","cast","listed_in","description". the steps of writting the algorithm is the same as first algorithm.

In [289]:

```
def data_clean(x):
      return str.lower(x)
```

In [290]:

```
filter_data=df[["title","director","cast","listed_in","description"]]
```

In [291]:

```
filter_data.head()
```

Out[291]:

	title	director	cast	listed_in	description
0	Dick Johnson Is Dead	Kirsten Johnson		Documentaries	as her father nears the end of his life, filmm...
1	Blood & Water		Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	International TV Shows, TV Dramas, TV Mysteries	after crossing paths at a party, a cape town t...
2	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	Crime TV Shows, International TV Shows, TV Act...	to protect his family from a powerful drug lor...
3	Jailbirds New Orleans			Docuseries, Reality TV	feuds, flirtations and toilet talk go down amo...
4	Kota Factory		Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	International TV Shows, Romantic TV Shows, TV ...	in a city of coaching centers known to train i...

In [292]:

```
features= ["title","director","cast","listed_in","description"]
```

In [293]:

```
for feature in features:
   filter_data[feature]= filter_data[feature].apply(data_clean)
```

```
<ipython-input-293-da892f5c0307>:2: 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
  filter_data[feature]= filter_data[feature].apply(data_clean)
```

In [294]:

```
filter_data.head()
```

Out[294]:

	title	director	cast	listed_in	description
0	dick johnson is dead	kirsten johnson		documentaries	as her father nears the end of his life, filmm...
1	blood & water		ama qamata, khosi ngema, gail mabalane, thaban...	international tv shows, tv dramas, tv mysteries	after crossing paths at a party, a cape town t...
2	ganglands	julien leclercq	sami bouajila, tracy gotoas, samuel jouy, nabi...	crime tv shows, international tv shows, tv act...	to protect his family from a powerful drug lor...
3	jailbirds new orleans			docuseries, reality tv	feuds, flirtations and toilet talk go down amo...
4	kota factory		mayur more, jitendra kumar, ranjan raj, alam k...	international tv shows, romantic tv shows, tv ...	in a city of coaching centers known to train i...

Changing to the columns to one column to compute cosine_similarity

In [295]:

```
def create_soup(x):
    return x['title']+ ' ' + x['director'] + ' ' + x['cast'] + ' '
+x['listed_in']+' '+ x['description']
     #return  x['director'] + ' ' + x['cast'] + ' ' +x['listed_in']+' '+
x['description']
filter_data['soup'] = filter_data.apply(create_soup, axis=1)
```

```
<ipython-input-295-636f80b039af>:4: 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
  filter_data['soup'] = filter_data.apply(create_soup, axis=1)
```

In [296]:

```
# Import CountVectorizer and create the count matrix
from sklearn.feature_extraction.text import CountVectorizer

count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(filter_data['soup'])



# Compute the Cosine Similarity matrix based on the count_matrix
from sklearn.metrics.pairwise import cosine_similarity

cosine_sim2 = cosine_similarity(count_matrix, count_matrix)



# Reset index of our main DataFrame and construct reverse mapping as before
filter_data=filter_data.reset_index()
indices = pd.Series(filter_data.index, index=filter_data['title'])
```

In [297]:

```
def get_recommendations_new(title, cosine_sim=cosine_sim):
   # title=title.replace(' ','').lower()
    title=title.lower()
    idx = indices[title]

    sim_scores = list(enumerate(cosine_sim[idx]))

    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    sim_scores = sim_scores[1:11]

    movie_indices = [i[0] for i in sim_scores]

    return df['title'].iloc[movie_indices]
```

In [298]:

```
get_recommendations_new('Welcome', cosine_sim2)
```

Out[298]:

```
4736                  Thank You
5183                  Mubarakan
6296                Bhagam Bhag
6107                    Aitraaz
8161             Tees Maar Khan
8171                       Tezz
6289                    Bewafaa
7023    Humko Deewana Kar Gaye
7590                   No Entry
7837                      Ready
Name: title, dtype: object
```

In [299]:

```
get_recommendations_new("Dick Johnson Is Dead", cosine_sim2)
```

Out[299]:

```
5233    The Death and Life of Marsha P. Johnson
7015                          How to Be a Player
5894                  Anjelah Johnson: Not Fancy
4877                                    End Game
5797                                    Extremis
3927                                    New Girl
3717                               Triple Threat
129                           An Unfinished Life
7622                                 Nowhere Boy
5540                                  Win It All
Name: title, dtype: object
```

In [300]:

```
df[(df["title"] == "Dick Johnson Is Dead")|(df["title"] == "The Death and
Life of Marsha P. Johnson") | (df["title"] == "How to Be a Player")|
(df["title"] == "Anjelah Johnson: Not Fancy")| (df["title"] == "Extremis")]
```

Out[300]:

	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		United States	2021-09-25	2020	PG-13	90 min	Documentaries	as her father nears the end of his life, filmm ...

5233	s5234	Movie	The Death and Life of Marsha P. Johnson	David France		Greece, United States	2017-10-06	2017	TV-MA	106 min	Documentaries, LGBTQ Movies	as she fights the tide of violence against tra...
5797	s5798	Movie	Extremis	Dan Krauss		United States	2016-09-13	2016	TV-PG	25 min	Documentaries	witness the wrenching emotions that accompany ...
5894	s5895	Movie	Anjelah Johnson: Not Fancy	Jay Karas	Anjelah Johnson-Reyes	United States	2015-10-02	2015	TV-14	64 min	Stand-Up Comedy	the actress, comedian and youtube sensation ri...
7015	s7016	Movie	How to Be a Player	Lionel C. Martin	Bill Bellamy, Natalie Desselle, Lark Voorhies,...	United States	2019-11-01	1997	R	94 min	Comedies	dray lives life one woman at a time and is the...

In [ ]:

```
df[(df["title"] == "Dick Johnson Is Dead")|(df["title"] == "How to Be a
Player") | (df["title"] == " Extremis")| (df["title"] == "End Game")]
```

Out[ ]:

	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		United States	2021-09-25	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
4877	s4878	Movie	End Game	Rob Epstein, Jeffrey Friedman		United States	2018-05-04	2018	TV-PG	40 min	Documentaries	Facing an inevitable outcome, terminally ill p...
7015	s7016	Movie	How to Be a Player	Lionel C. Martin	Bill Bellamy, Natalie Desselle, Lark Voorhies,...	United States	2019-11-01	1997	R	94 min	Comedies	Dray lives life one woman at a time and is the...

In [ ]:

```
get_recommendations_new("How to Be a Player", cosine_sim2)
```

Out[ ]:

```
0                                     Dick Johnson Is Dead
7365                          Mac & Devin Go to High School
```

```
149                                     I Got the Hook Up
144                                           House Party
6060                    A Thin Line Between Love & Hate
3908                                           About Time
5233                The Death and Life of Marsha P. Johnson
4851    Steve Martin and Martin Short: An Evening You ...
3927                                             New Girl
67                                      Saved by the Bell
Name: title, dtype: object
```

In [301]:

```
!pip install pandoc
```

```
Collecting pandoc
  Downloading pandoc-2.3.tar.gz (33 kB)
  Preparing metadata (setup.py) ... done
Collecting plumbum (from pandoc)
  Downloading plumbum-1.8.2-py3-none-any.whl (127 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.0/127.0 kB 4.0 MB/s eta
0:00:00
Collecting ply (from pandoc)
  Downloading ply-3.11-py2.py3-none-any.whl (49 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.6/49.6 kB 5.1 MB/s eta
0:00:00
Building wheels for collected packages: pandoc
  Building wheel for pandoc (setup.py) ... done
  Created wheel for pandoc: filename=pandoc-2.3-py3-none-any.whl size=33263
sha256=3cba48457d7831fa78fea3b0d026648b24ec8f567eea55f1215b857201a6e9c5
  Stored in directory:
/root/.cache/pip/wheels/76/27/c2/c26175310aadcb8741b77657a1bb49c50cc7d4cdbf9e
ee0005
Successfully built pandoc
Installing collected packages: ply, plumbum, pandoc
Successfully installed pandoc-2.3 plumbum-1.8.2 ply-3.11
```