```python
import pandas as pd
import matplotlib.pyplot as plt
```

```python
df=pd.read_csv('Amazon Prime RT.csv')
```

```python
df.columns
```

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

```python
df.head(10)
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	The Grand Seduction	Don McKellar	Brendan Gleeson, Taylor Kitsch, Gordon Pinsent	Canada	March 30, 2021	2014	NaN	113 min	Comedy, Drama	A small fishing village must procure a local d...
1	s2	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar	India	March 30, 2021	2018	13+	110 min	Drama, International	A Metro Family decides to fight a Cyber Crimin...
2	s3	Movie	Secrets of Deception	Josh Webber	Tom Sizemore, Lorenzo Lamas, Robert LaSardo, R...	United States	March 30, 2021	2017	NaN	74 min	Action, Drama, Suspense	After a man discovers his wife is cheating on ...
3	s4	Movie	Pink: Staying True	Sonia Anderson	Interviews with: Pink, Adele, Beyoncé, Britney...	United States	March 30, 2021	2014	NaN	69 min	Documentary	Pink breaks the mold once again, bringing her ...
					Harry							

```python
df.shape
```

```
(9668, 12)
```

```python
df.isnull().any()
```

	0
show_id	False
type	False
title	False
director	True
cast	True
country	True
date_added	True
release_year	False
rating	True
duration	False
listed_in	False
description	False

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

	0
show_id	0
type	0
title	0
director	2083
cast	1233
country	8996
date_added	9513
release_year	0
rating	337
duration	0
listed_in	0
description	0

```
df.drop_duplicates
```

```
pandas.core.frame.DataFrame.drop_duplicates
def drop_duplicates(subset: Hashable | Sequence[Hashable] | None=None, *, keep: DropKeep='first',
inplace: bool=False, ignore_index: bool=False) -> DataFrame | None

Return DataFrame with duplicate rows removed.

Considering certain columns is optional. Indexes, including time indexes
are ignored.

Parameters
```

```
df['director'].fillna('Unknown',inplace=True)
df['cast'].fillna('Unkown',inplace=True)
df['country'].fillna('Unkown',inplace=True)
df['date_added'].fillna('Unkown',inplace=True)
```

```
<ipython-input-10-d7aebea3dda6>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[co


  df['director'].fillna('Unknown',inplace=True)
<ipython-input-10-d7aebea3dda6>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[co


  df['cast'].fillna('Unkown',inplace=True)
<ipython-input-10-d7aebea3dda6>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[co


  df['country'].fillna('Unkown',inplace=True)
<ipython-input-10-d7aebea3dda6>:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[co


  df['date_added'].fillna('Unkown',inplace=True)
```

```
#use mode for categorical value otherwise use mean
mode_value = df['rating'].mode()[0]
df['rating'].fillna(mode_value)
```

```
      rating
   0     13+
   1     13+
   2     13+
   3     13+
   4     13+
 ...     ...
9663      7+
9664     13+
9665       R
9666   TV-MA
9667       R
9668 rows × 1 columns

dtype: object
```

```
1 df
```

```
   show_id  type   title                 director        cast                                                          country          date_added      release_year  rating  duration  listed_in                   description
0  s1       Movie  The Grand Seduction   Don McKellar    Brendan Gleeson, Taylor Kitsch, Gordon Pinsent                Canada           March 30, 2021  2014          NaN     113 min   Comedy, Drama               A small fishing village must procure a local d...
1  s2       Movie  Take Care Good Night  Girish Joshi    Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar              India            March 30, 2021  2018          13+     110 min   Drama, International        A Metro Family decides to fight a Cyber Crimin...
2  s3       Movie  Secrets of Deception  Josh Webber     Tom Sizemore, Lorenzo Lamas, Robert LaSardo, R...             United States    March 30, 2021  2017          NaN     74 min    Action, Drama, Suspense     After a man discovers his wife is cheating on ...
3  s4       Movie  Pink: Staying True    Sonia Anderson  Interviews with: Pink, Adele, Beyoncé, Britney...             United States    March 30, 2021  2014          NaN     69 min    Documentary                 Pink breaks the mold once again, bringing her ...
4  s5       Movie  Monster Maker         Giles Foster    Harry Dean Stanton, Kieran O'Brien, George                    United Kingdom   March 30, 2021  1989          NaN     45 min    Drama, Fantasy              Teenage Matt Banting wants to work with a famo...
```

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9668 entries, 0 to 9667
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       9668 non-null   object
 1   type          9668 non-null   object
 2   title         9668 non-null   object
 3   director      9668 non-null   object
 4   cast          9668 non-null   object
 5   country       9668 non-null   object
 6   date_added    9668 non-null   object
 7   release_year  9668 non-null   int64 
 8   rating        9331 non-null   object
 9   duration      9668 non-null   object
 10  listed_in     9668 non-null   object
 11  description   9668 non-null   object
dtypes: int64(1), object(11)
memory usage: 906.5+ KB
```

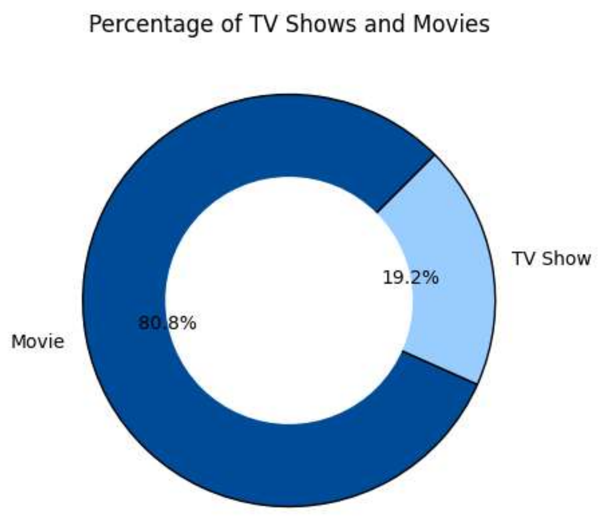
```
total_shows= df['show_id'].count()
total_shows #This also the total number of titles
```

```
np.int64(9668)
```

```
show_types=df['type'].value_counts()
show_types
```

	count
type	
Movie	7814
TV Show	1854

```
plt.figure(figsize=(5,5))
plt.pie(
    show_types,
    labels=show_types.index,
    colors=['#004C99','#99CCFF'],
    autopct='%1.1f%%',
    startangle=45,
    wedgeprops={'edgecolor': 'black', 'linewidth': 1}
)
# Add a white circle at the center to create the donut effect
centre_circle = plt.Circle((0, 0), 0.60, fc='white')
plt.gca().add_artist(centre_circle)
plt.title('Percentage of TV Shows and Movies')
plt.show()
```



```
#total number of countries
total_countries=df['country'].nunique()
total_countries
```

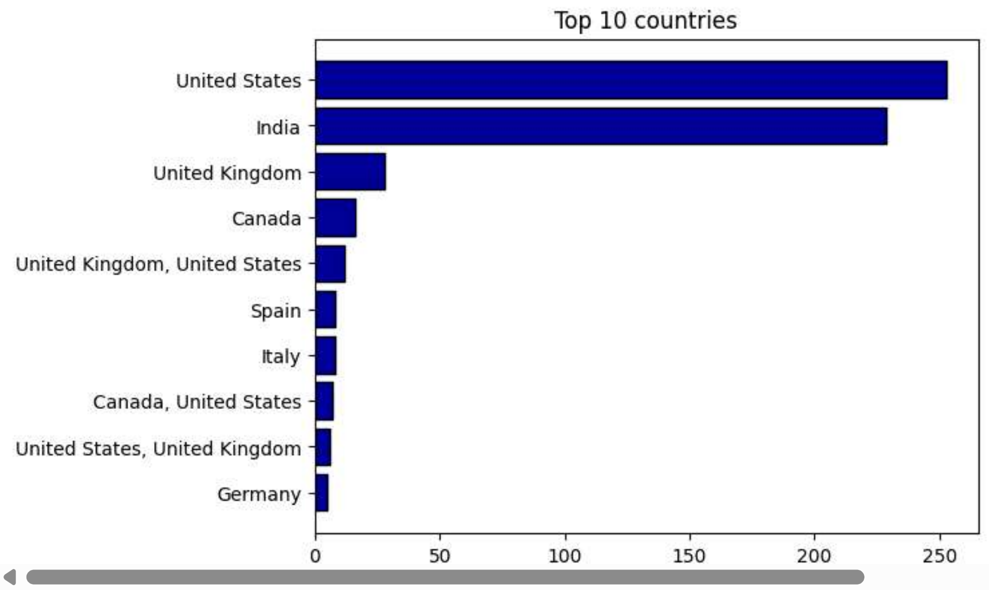
```
87
```

```
df['country'].value_counts().head(5)
```

country	count
Unkown	8996
United States	253
India	229
United Kingdom	28
Canada	16

```
#Top 10 countries except Unknown
x=df['country'].value_counts().keys()[1:11]
y=df['country'].value_counts()[1:11]
#plot
plt.barh(x[::-1],y[::-1],
color=['#000099'],
edgecolor='black')
plt.title('Top 10 countries')
plt.show
```

```
matplotlib.pyplot.show
def show(*args, **kwargs) -> None

Display all open figures.

Parameters
----------
block : bool, optional
    Whether to wait for all figures to be closed before returning.
```

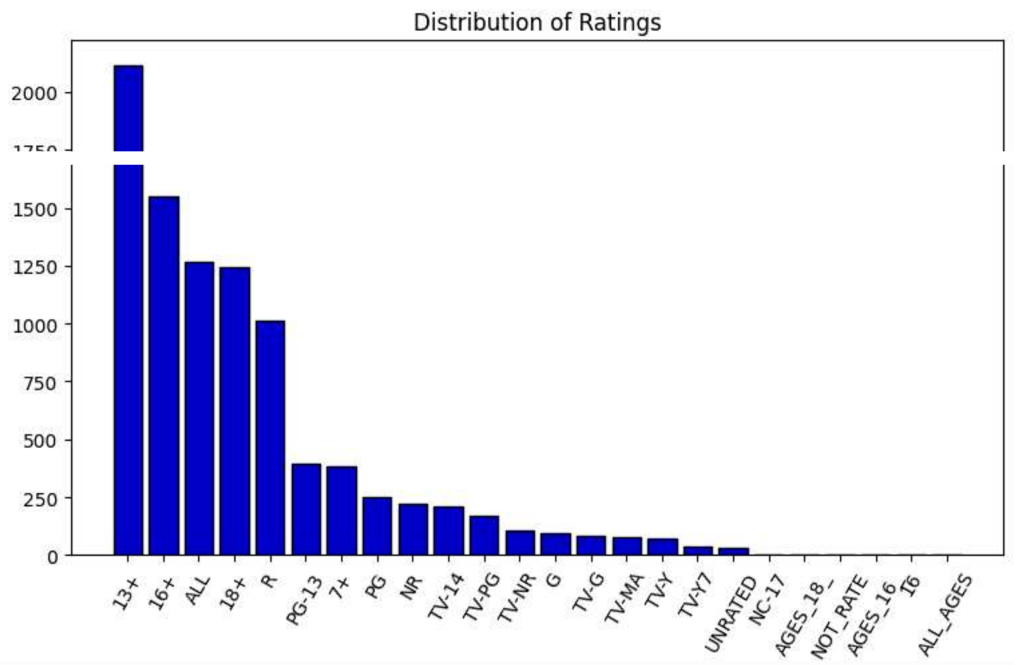


```
#Total types of rating
df['rating'].nunique()
```

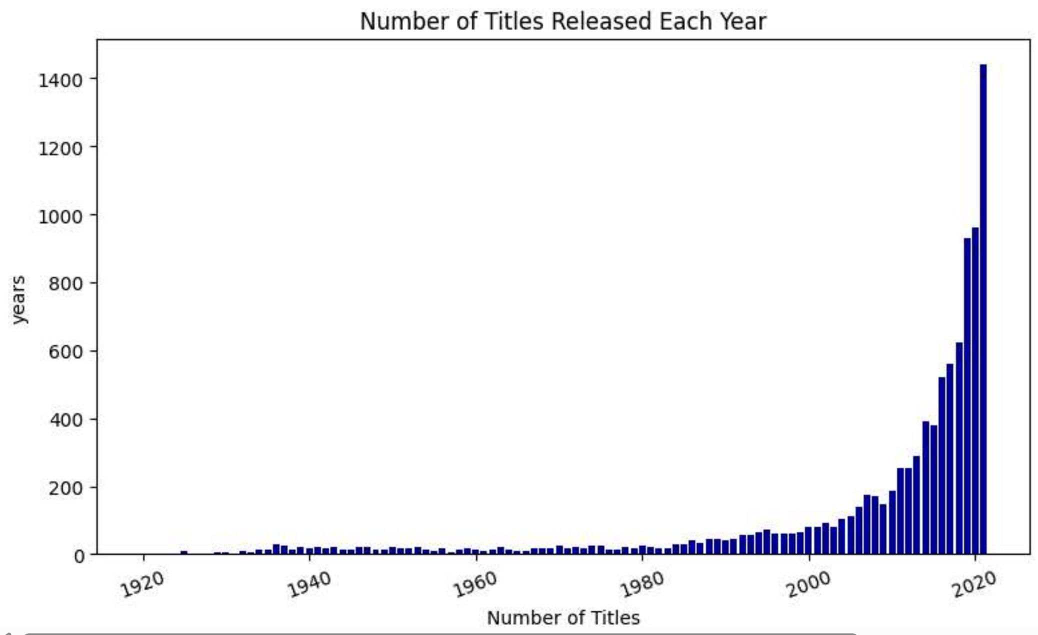
```
24
```

```
#Bar chart for Distribution of Ratings
plt.figure(figsize=(9,5))
bars=plt.bar(df['rating'].value_counts().keys(),df['rating'].value_counts(),color='#0000CC',edgecolor
plt.title('Distribution of Ratings')
plt.xticks(rotation=60)
```

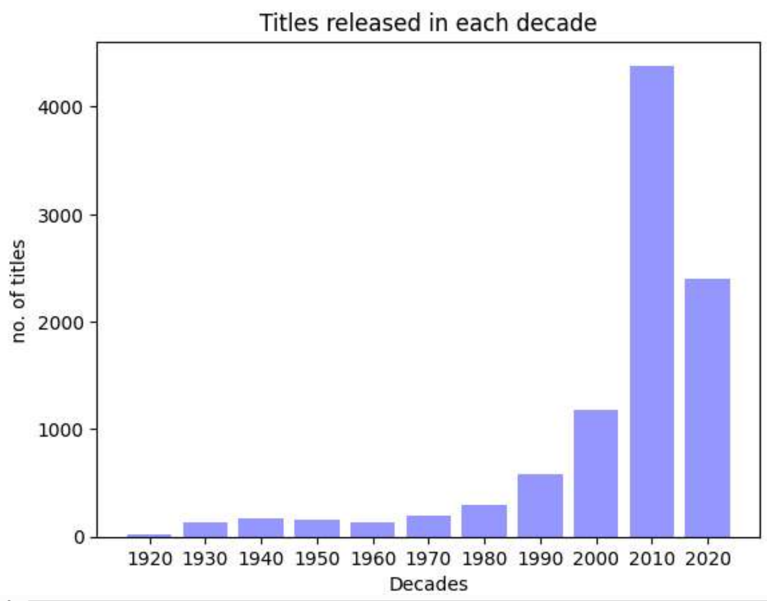
```
#for bar in bars:
# yval = bar.get_height()
#plt.scatter(bar.get_x() + bar.get_width() / 2, yval, color='black', s=20, zorder=5)

plt.show()
```



```
#Bar chart of titles released over the years
plt.figure(figsize=(9,5))
plt.bar(df['release_year'].value_counts().index,df['release_year'].value_counts(),color='#000099')
plt.title('Number of Titles Released Each Year')
plt.xlabel('Number of Titles')
plt.ylabel('years')
plt.xticks(rotation=20)
plt.show()
```



```
#Titles released each decade
#2005 // 10 computes to 200 and *10 makes it 2000 . Hence grouping it to 2000s
#Appends the character 's' after year like 2000s 2010s
df['decade']=(df['release_year'] // 10 * 10).astype(str)
decade_counts=df['decade'].value_counts().sort_index()
plt.bar(decade_counts.index,decade_counts.values,color='#9999FF')
plt.xlabel('Decades')
plt.ylabel(' no. of titles ')
plt.title('Titles released in each decade')
plt.show()
```



```
df
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	The Grand Seduction	Don McKellar	Brendan Gleeson, Taylor Kitsch, Gordon Pinsent	Canada	March 30, 2021	2014	NaN	113 min	Comedy, Drama	A small fishing village must procure a local d...
1	s2	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar	India	March 30, 2021	2018	13+	110 min	Drama, International	A Metro Family decides to fight a Cyber Crimin...
2	s3	Movie	Secrets of Deception	Josh Webber	Tom Sizemore, Lorenzo Lamas, Robert LaSardo, R...	United States	March 30, 2021	2017	NaN	74 min	Action, Drama, Suspense	After a man discovers his wife is cheating on ...
3	s4	Movie	Pink: Staying True	Sonia Anderson	Interviews with: Pink, Adele, Beyoncé, Britney...	United States	March 30, 2021	2014	NaN	69 min	Documentary	Pink breaks the mold once again, bringing her ...
4	s5	Movie	Monster Maker	Giles Foster	Harry Dean Stanton, Kieran O'Brien, George Cos...	United Kingdom	March 30, 2021	1989	NaN	45 min	Drama, Fantasy	Teenage Matt Banting wants to work with a famo...

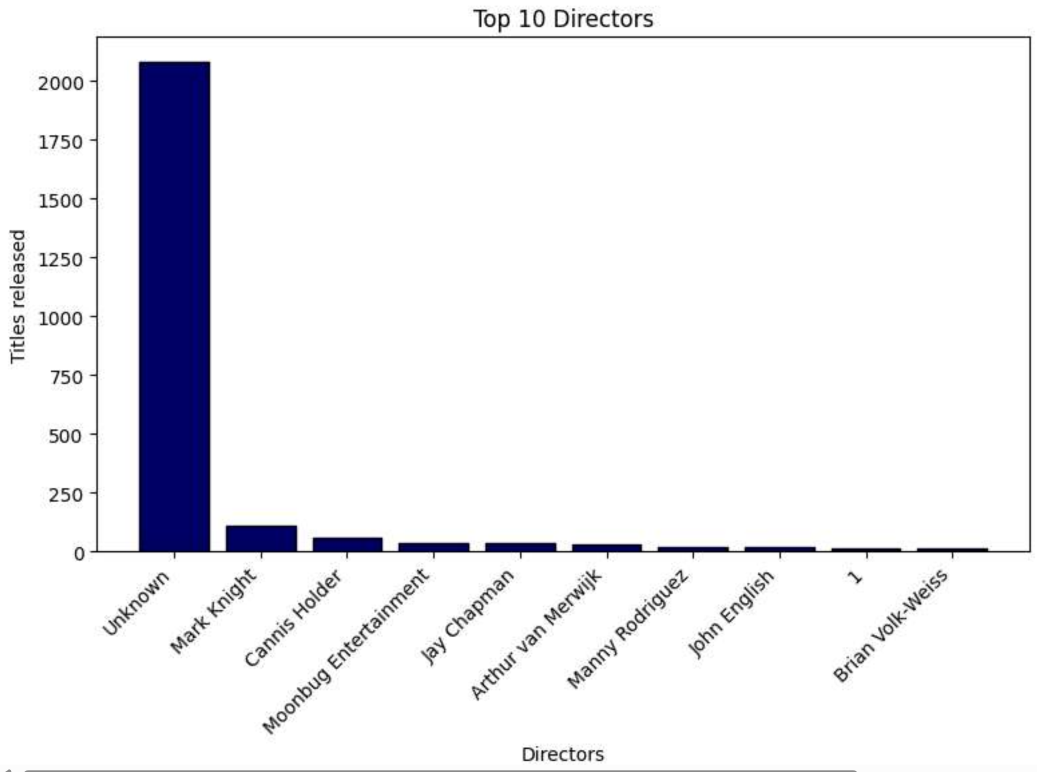
```
df['director'].value_counts().head(10)
```

director	count
Unknown	2083
Mark Knight	113
Cannis Holder	61
Moonbug Entertainment	37
Jay Chapman	34
Arthur van Merwijk	30
Manny Rodriguez	22
John English	20
1	16
Brian Volk-Weiss	15

```
#show .nunique() to find no. of unique values instead of unique which shows all unique values
df['director'].nunique()
```

```
5774
```

```
#Top 10 Directors
plt.figure(figsize=(9,5))
plt.bar(df['director'].value_counts().keys()[:10],df['director'].value_counts()[:10],color='#000066',
plt.title('Top 10 Directors')
plt.xlabel('Directors')
plt.ylabel('Titles released')
plt.xticks(rotation=45,ha='right')
plt.show()
```



```
#Top 10 geners after splitting data considering','
#Use lambda to remove spaces before and afer the values
#Use .explode() to create
#all_genres = df['listed_in'].str.split(',').apply(lambda x:[genre.strip() for genre in x]).explode()
```

```python
df['listed_in'].nunique()
```

```
518
```

```python
#Start year in the dataset
df['release_year'].min()
```

```
1920
```

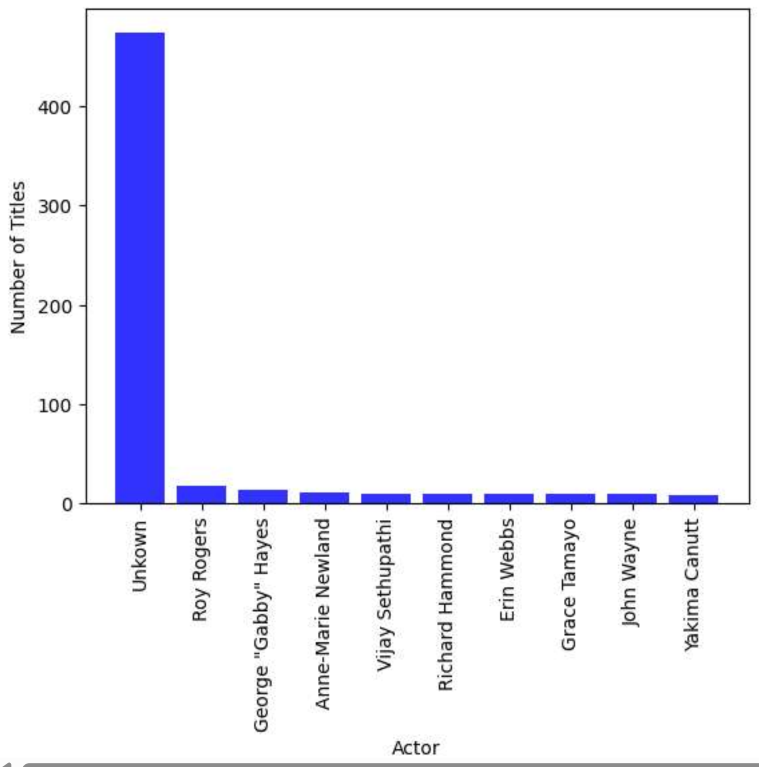
```python
#End year in the dataset
df['release_year'].max()
```

```
2021
```

```python
df['duration'].value_counts(5)
```

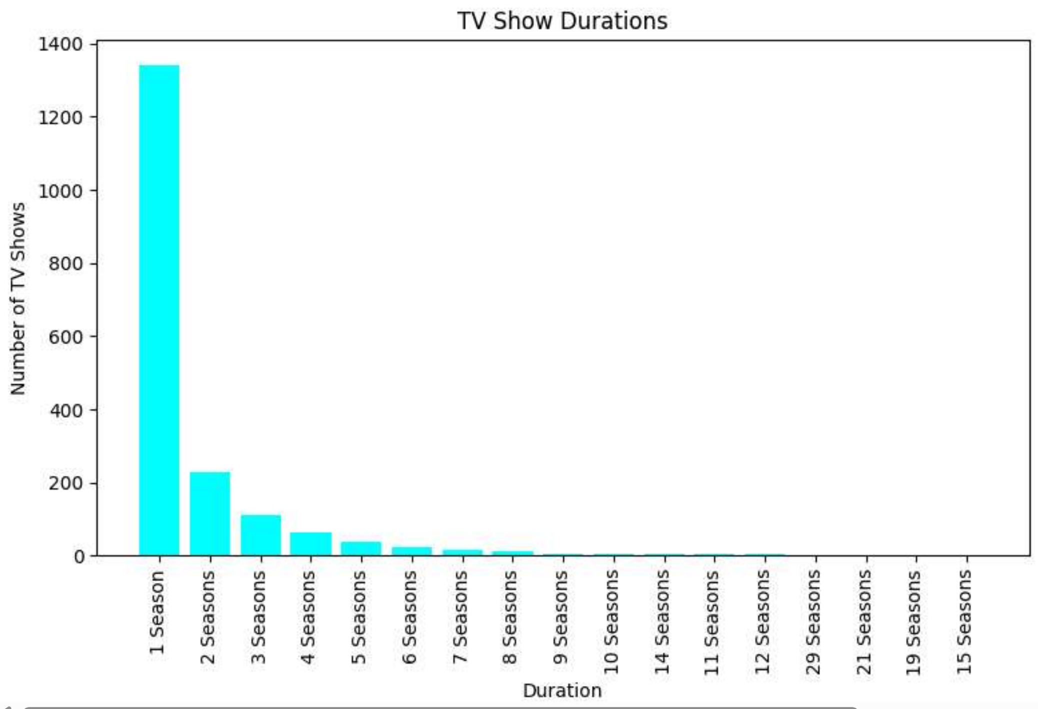
```
            proportion
duration
1 Season      0.138808
2 Seasons     0.023686
90 min        0.023066
91 min        0.022755
92 min        0.020997
...                ...
192 min       0.000103
207 min       0.000103
269 min       0.000103
15 Seasons    0.000103
191 min       0.000103

219 rows × 1 columns

dtype: float64
```

```python
#Reset the index to ensure unique index values. drop=True to discard old index
# ignore_index=True will prevent the error by creating a new index for the exploded column
df = df.reset_index(drop=True)
df['actors']=df['cast'].str.split(',').apply(lambda x: [actor.strip() for actor in x]).explode(ignore_


plt.bar(df['actors'].value_counts().keys()[:10],df['actors'].value_counts()[:10],color='#3333FF')
plt.ylabel('Number of Titles')
plt.xlabel('Actor')
plt.xticks(rotation=90)
plt.show()
```



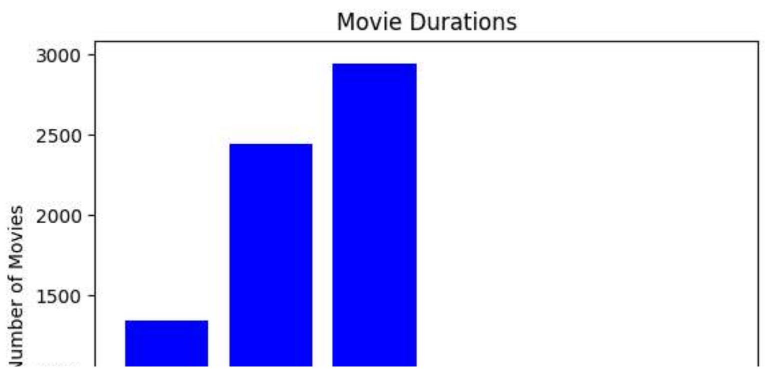
```
#TV shows duration
# Filter for TV Shows and remove rows with NaN values in the 'duration' column
tv_shows = df[df['type'] == 'TV Show'].dropna(subset=['duration'])
duration_counts = tv_shows['duration'].value_counts()
```

```
# Plot the distribution of TV Show durations
plt.figure(figsize=(9, 5))
plt.bar(duration_counts.index, duration_counts.values, color='#00FFFF')
plt.title('TV Show Durations')
plt.xlabel('Duration')
plt.ylabel('Number of TV Shows')
plt.xticks(rotation=90)
plt.show()

```



```
#Filter for movies and remove the NaN values in the 'Duration' column
movie= df[df['type']=='Movie'].dropna(subset=['duration'])

# Extract numeric duration in minutes from the 'duration' column
movie['duration_minutes']=movie['duration'].str.extract('(\d+)').astype(int)

# Define bins and labels for movie durations
bins = [0, 60, 90, 120, 150, 180, 300]
labels = ['0-60 min', '61-90 min', '91-120 min', '121-150 min', '151-180 min', '180+ min']

# Categorize movies into duration groups
movie['duration_group'] = pd.cut(movie['duration_minutes'], bins=bins, labels=labels)

duration_group_counts = movie['duration_group'].value_counts().sort_index()
```

```
plt.bar(duration_group_counts.index, duration_group_counts.values, color='blue')
plt.title('Movie Durations')
plt.xlabel('Duration Group')
plt.ylabel('Number of Movies')
plt.xticks(rotation=90)
plt.show()
```


Movie Durations
Number of Movies