1. **Data Loading & Overview.**

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

Choose Files netflix_titles.csv

netflix_titles.csv(text/csv) - 3399670 bytes, last modified: n/a - 100% done

```
Saving netflix_titles.csv to netflix_titles.csv
```

```
import pandas as pd

# Replace 'netflix_titles.csv' with the exact filename if it's different
df = pd.read_csv('netflix_titles.csv')

# Display shape and preview
print("Shape:", df.shape)
display(df.head())
display(df.tail())
```

```
Shape: (8807, 12)
```

	show_id	type	title	director	cast	country	date_added	release_ye
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	20
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	20
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	20
3	s4	TV Show	Jailbirds New	NaN	NaN	NaN	September	20

```
                Show     Orleans                                              24, 2021

                                           Mayur
                                           More,
                                         Jitendra
                  TV       Kota             Kumar,              September
4       s5      Show    Factory    NaN    Ranjan      India     24, 2021     20
                                         Raj, Alam
                                              K...
```

	show_id	type	title	director	cast	country	date_added	releas
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	

2. **Data Cleaning.**

```
# Checking for missing values in the dataset
missing_values = df.isnull().sum()
missing_values_percentage = (missing_values / len(df)) * 100

missing_values[missing_values > 0], missing_values_percentage[missing_values > 0]
```

```
(director      2634
 cast           825
 country        831
 date_added      10
 rating           4
 duration         3
 dtype: int64,
 director      29.908028
 cast           9.367549
 country        9.435676
 date_added     0.113546
 rating         0.045418
 duration       0.034064
 dtype: float64)
```

```
# Fill missing values with appropriate placeholders
df['director'] = df['director'].fillna('Unknown')
df['cast'].fillna('Unknown', inplace=True)
df['country'].fillna('Unknown', inplace=True)
df['date_added'].fillna('Unknown', inplace=True)
df['rating'].fillna('NR', inplace=True)  # 'NR' for Not Rated
df['duration'].fillna('Unknown', inplace=True)

# Verify that there are no missing values left
df.isnull().sum()
```

```
<ipython-input-12-79b00edfff5f>:3: FutureWarning: A value is trying to be set
The behavior will change in pandas 3.0. This inplace method will never work be

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.m


  df['cast'].fillna('Unknown', inplace=True)
<ipython-input-12-79b00edfff5f>:4: FutureWarning: A value is trying to be set
The behavior will change in pandas 3.0. This inplace method will never work be

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.m


  df['country'].fillna('Unknown', inplace=True)
<ipython-input-12-79b00edfff5f>:5: FutureWarning: A value is trying to be set
```

```
The behavior will change in pandas 3.0. This inplace method will never work be

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.m


  df['date_added'].fillna('Unknown', inplace=True)
<ipython-input-12-79b00edfff5f>:6: FutureWarning: A value is trying to be set 
The behavior will change in pandas 3.0. This inplace method will never work be

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.m


  df['rating'].fillna('NR', inplace=True)  # 'NR' for Not Rated
<ipython-input-12-79b00edfff5f>:7: FutureWarning: A value is trying to be set 
The behavior will change in pandas 3.0. This inplace method will never work be

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.m


  df['duration'].fillna('Unknown', inplace=True)
```

	0
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

3. **Data Types & Conversion.**

```
# Check current data types
print("Before conversion:\n", df.dtypes)

# Convert 'date_added' to datetime format
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')

# Check updated data types
print("\nAfter conversion:\n", df.dtypes)
```

```
Before conversion:
 show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
rating          object
duration        object
listed_in       object
description     object
dtype: object

After conversion:
 show_id                object
type                   object
title                  object
director               object
cast                   object
country                object
date_added      datetime64[ns]
release_year            int64
rating                 object
duration               object
listed_in              object
description            object
dtype: object
```

```
# Check data types of all columns
data_types_before = df.dtypes

# Convert 'date_added' to datetime, errors='coerce' will turn unparseable dates i
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')

# Check updated data types
data_types_after = df.dtypes

data_types_before, data_types_after
```

```
(show_id                 object
 type                    object
 title                   object
 director                object
 cast                    object
 country                 object
 date_added      datetime64[ns]
 release_year             int64
 rating                  object
 duration                object
 listed_in               object
 description             object
 dtype: object,
 show_id                 object
 type                    object
 title                   object
 director                object
 cast                    object
 country                 object
 date_added      datetime64[ns]
 release_year             int64
 rating                  object
 duration                object
 listed_in               object
 description             object
 dtype: object)
```

4. **Univariate Analysis.**

1. How many Movies vs. TV Shows are there?

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

	count
type	
Movie	6131
TV Show	2676

dtype: int64

2. What are the top 5 most common ratings?

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

	count
rating	
TV-MA	3207
TV-14	2160
TV-PG	863
R	799
PG-13	490

dtype: int64

3. Which release year appears most frequently?

```
df['release_year'].value_counts().head(1)
```

	count
release_year	
2018	1147

dtype: int64

5. **Duration & Seasons.**

```
# Create two new columns
df['duration_minutes'] = df['duration'].str.extract(r'(\d+)').astype(float
df['num_seasons'] = df['duration_minutes'].where(df['type'] == 'TV Show',

# Keep only movie durations in minutes
df['duration_minutes'] = df['duration_minutes'].where(df['type'] == 'Movie


# Average duration of Movies
avg_movie_length = df['duration_minutes'].dropna().mean()

# Average number of seasons for TV Shows
avg_tv_seasons = df['num_seasons'].dropna().mean()

print(f"Average Movie Length: {avg_movie_length:.2f} minutes")
print(f"Average TV Show Seasons: {avg_tv_seasons:.2f} seasons")
```

```
Average Movie Length: 99.58 minutes
Average TV Show Seasons: 1.76 seasons
```

6. **Genre Analysis.**

```
# Create a new DataFrame with one genre per row
df_genres = df[['title', 'release_year', 'listed_in']].copy()
df_genres['genre'] = df_genres['listed_in'].str.split(', ')
df_genres = df_genres.explode('genre')
```

```
import pandas as pd

# Replace 'netflix_titles.csv' with the exact filename if it's different
df = pd.read_csv('netflix_titles.csv')
```

```
# Group by genre and calculate average release year
avg_year_by_genre = df_genres.groupby('genre')['release_year'].mean().sort

# Display the top 10 genres with most recent average release year
avg_year_by_genre.head(10)
```

genre	release_year
TV Mysteries	2018.346939
TV Horror	2018.200000
Reality TV	2017.894118
Stand-Up Comedy & Talk Shows	2017.857143
TV Thrillers	2017.736842
Crime TV Shows	2017.687234
Spanish-Language TV Shows	2017.477011
TV Action & Adventure	2017.404762
Docuseries	2017.232911
TV Dramas	2017.190039

dtype: float64

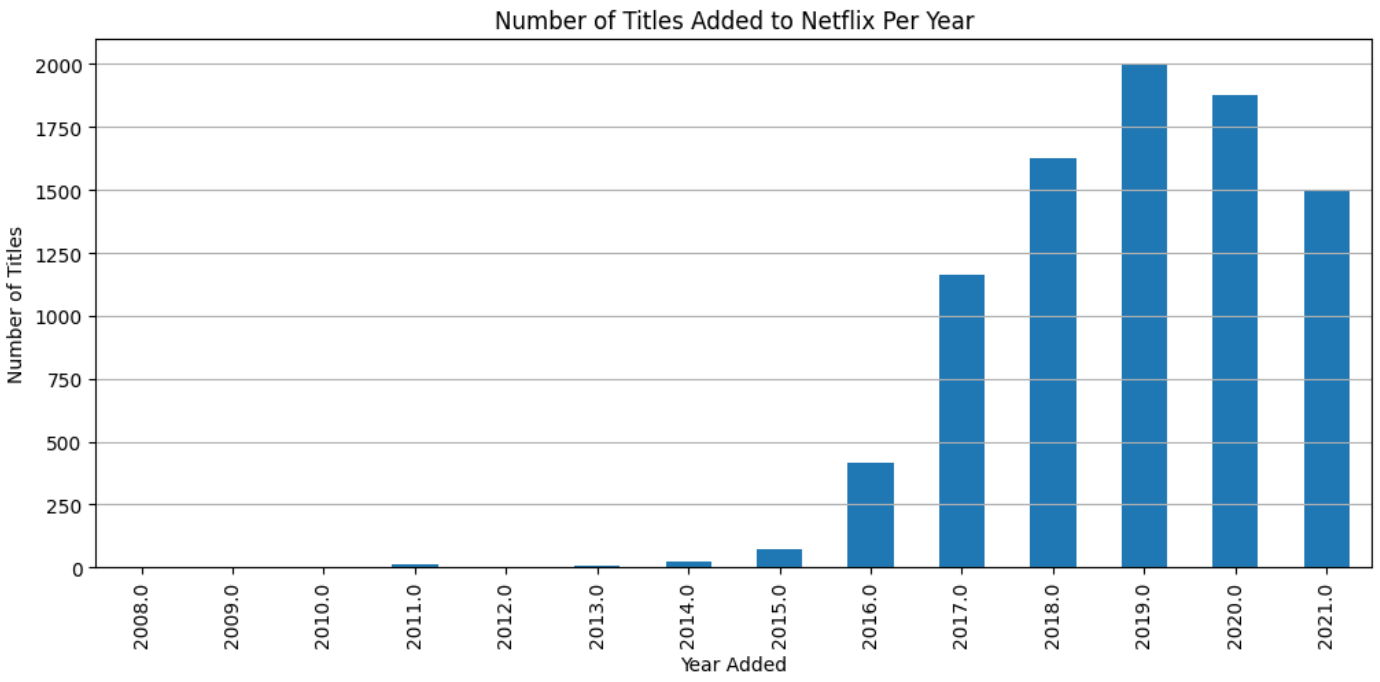
7. **Temporal Trends.**

```
# Extract year from date_added
df['year_added'] = df['date_added'].dt.year


# Calculate the difference
df['years_to_add'] = df['year_added'] - df['release_year']
```

```
# Drop NaNs and calculate the average delay
df['years_to_add'].dropna().describe()
```

```
       years_to_add
count   8709.000000
mean       4.690894
std        8.792208
min       -3.000000
25%        0.000000
50%        1.000000
75%        5.000000
max       93.000000

dtype: float64
```

```
import matplotlib.pyplot as plt

# Count of titles added per year
df['year_added'].value_counts().sort_index().plot(kind='bar', figsize=(12, 5))
plt.title("Number of Titles Added to Netflix Per Year")
plt.xlabel("Year Added")
plt.ylabel("Number of Titles")
plt.grid(axis='y')
plt.show()
```



8. **Rating vs. Type.**

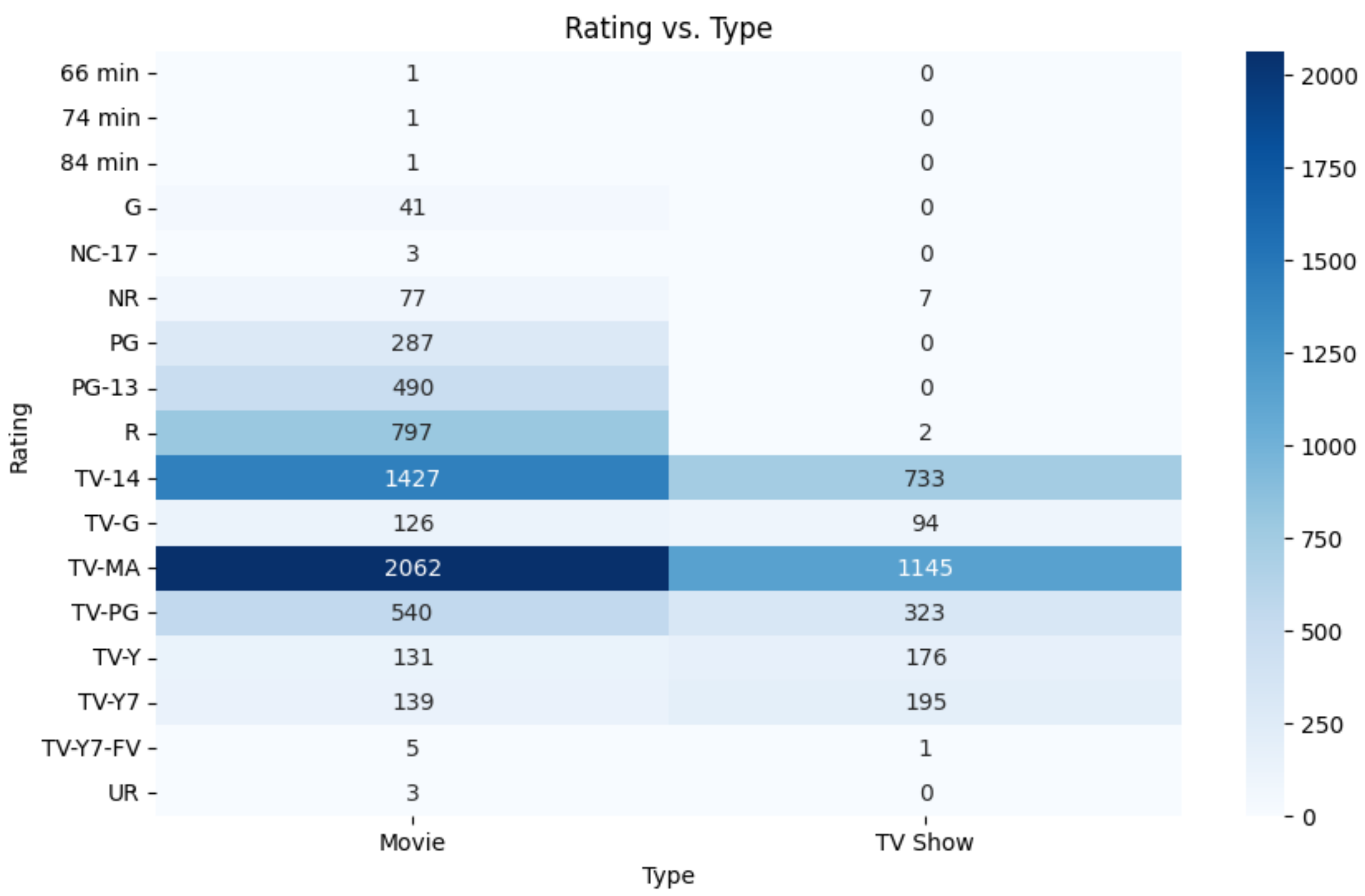
```
rating_type_crosstab = pd.crosstab(df['rating'], df['type'])
print(rating_type_crosstab)
```

```
type      Movie  TV Show
rating                  
66 min        1        0
74 min        1        0
84 min        1        0
G            41        0
NC-17         3        0
NR           77        7
PG          287        0
PG-13       490        0
R           797        2
TV-14      1427      733
TV-G        126       94
TV-MA      2062     1145
TV-PG       540      323
TV-Y        131      176
TV-Y7       139      195
TV-Y7-FV      5        1
UR            3        0
```

```
# Get proportions of each rating within content type
rating_type_proportion = pd.crosstab(df['rating'], df['type'], normalize='columns
print(rating_type_proportion)
```

```
type         Movie   TV Show
rating                      
66 min    0.000163  0.000000
74 min    0.000163  0.000000
84 min    0.000163  0.000000
G         0.006687  0.000000
NC-17     0.000489  0.000000
NR        0.012559  0.002616
PG        0.046811  0.000000
PG-13     0.079922  0.000000
R         0.129995  0.000747
TV-14     0.232752  0.273916
TV-G      0.020551  0.035127
TV-MA     0.336324  0.427877
TV-PG     0.088077  0.120703
TV-Y      0.021367  0.065770
TV-Y7     0.022672  0.072870
TV-Y7-FV  0.000816  0.000374
UR        0.000489  0.000000
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Plot the cross-tab as a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(rating_type_crosstab, annot=True, fmt='d', cmap='Blues')
plt.title("Rating vs. Type")
plt.ylabel("Rating")
plt.xlabel("Type")
plt.show()
```



9. **Filtering & Querying.**

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

Choose Files netflix_titles.csv

netflix_titles.csv(text/csv) - 3399670 bytes, last modified: n/a - 100% done

```
Saving netflix_titles.csv to netflix_titles (1).csv
```

```python
import pandas as pd

# Adjust the filename if needed
df = pd.read_csv('netflix_titles.csv')


# Convert date_added to datetime format
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')

# Filter R-rated content added after 2020
filtered_r_after_2020 = df[
    (df['rating'] == 'R') &
    (df['date_added'].dt.year > 2020)
]

# Output the result count
print(f"Number of R-rated titles added after 2020: {len(filtered_r_after_2020)}")

# Preview some results
filtered_r_after_2020[['title', 'date_added', 'release_year']].head()
```

```
Number of R-rated titles added after 2020: 190
```

	title	date_added	release_year
46	Safe House	2021-09-16	2012
48	Training Day	2021-09-16	2001
81	Kate	2021-09-10	2021
122	In the Cut	2021-09-02	2003
131	Blade Runner: The Final Cut	2021-09-01	1982

10. **Aggregations & GroupBy.**

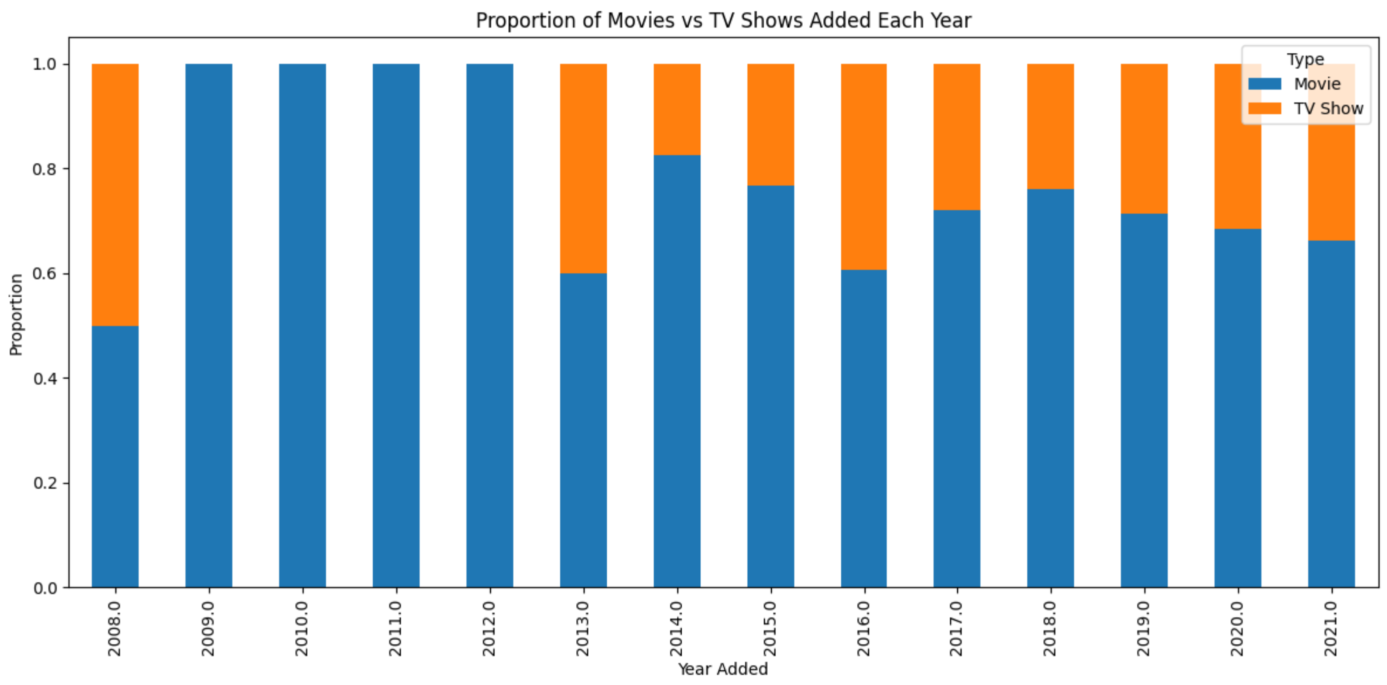
```
# Replace 'Unknown' with NaN temporarily to avoid skewed results
df_country_year = df.replace("Unknown", pd.NA)

# Group by country and calculate average release year
avg_release_year_by_country = df_country_year.groupby('country')['release_year'].

# Display the top 10 countries with the most recent average release year
avg_release_year_by_country.head(10)
```

	release_year
country	
Italy, United Kingdom	2021.0
Mauritius	2021.0
Belgium, United Kingdom	2021.0
United States, Singapore	2021.0
Mexico, France, Colombia	2021.0
Italy, Brazil, Greece	2021.0
Mexico, Brazil	2021.0
United States, Brazil, Japan, Spain, India	2021.0
Canada, United States, Ireland	2021.0
Canada, United States, Cayman Islands	2021.0

dtype: float64

```
# Extract the year from 'date_added' into a new column
df['year_added'] = pd.to_datetime(df['date_added'], errors='coerce').dt.year


# Count how many Movies and TV Shows were added per year
type_per_year = df.groupby(['year_added', 'type']).size().unstack()

# Convert to proportions
type_proportion_per_year = type_per_year.div(type_per_year.sum(axis=1), axis=0)

# Plot the proportions
import matplotlib.pyplot as plt
```

```
type_proportion_per_year.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.title("Proportion of Movies vs TV Shows Added Each Year")
plt.xlabel("Year Added")
plt.ylabel("Proportion")
plt.legend(title='Type')
plt.tight_layout()
plt.show()
```



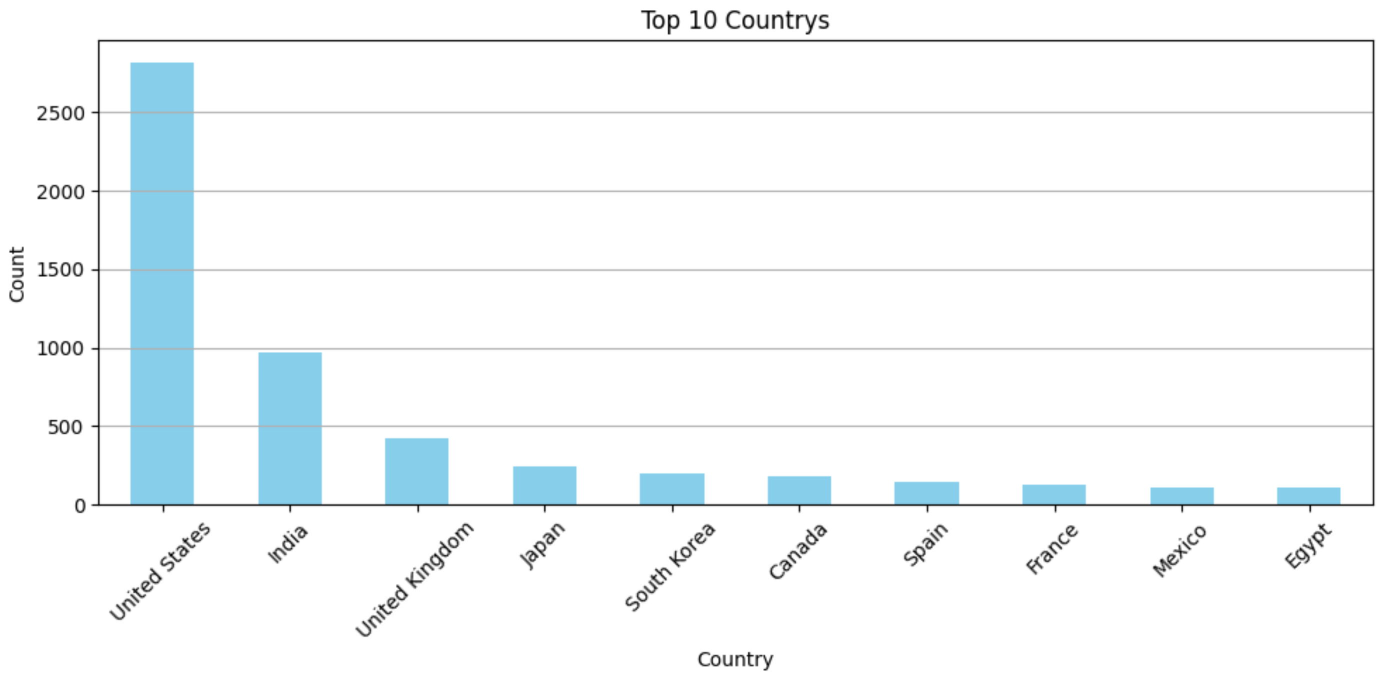
11. **Applying Functions.**

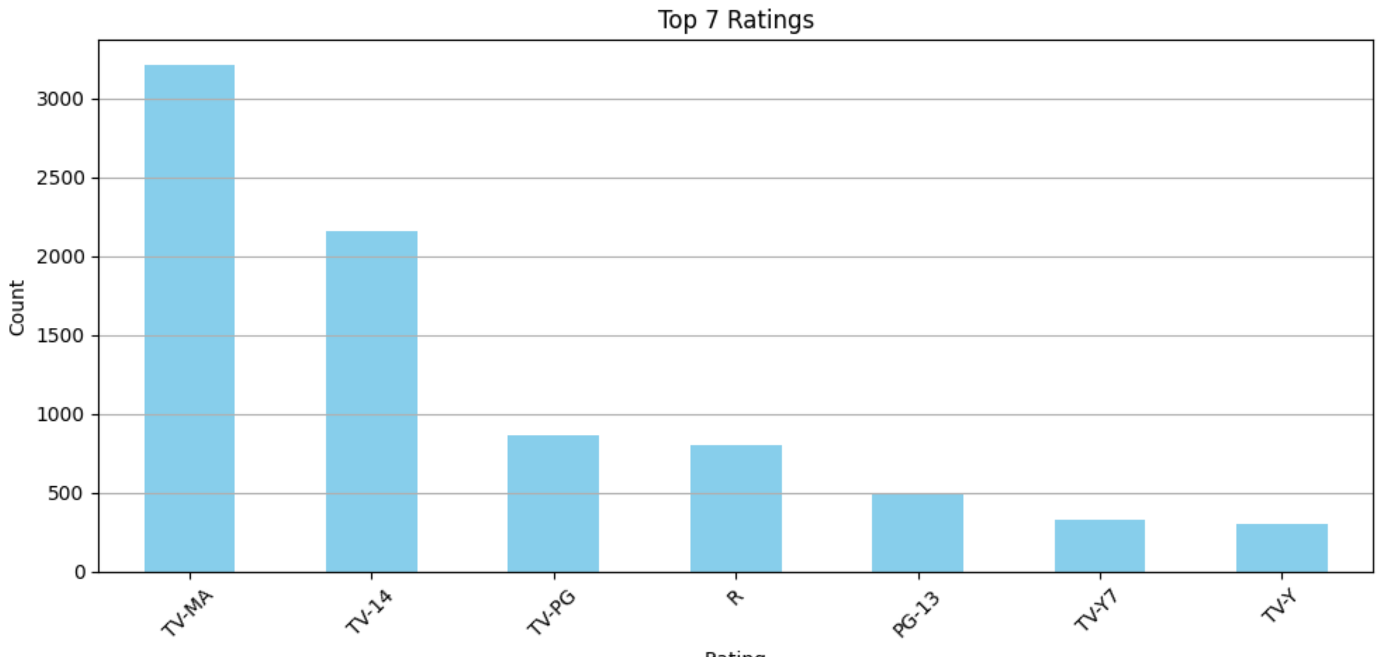
```
def get_director_titles(name):
    # Filter for the director's titles
    return df[df['director'].str.contains(name, case=False, na=False)]\
            .sort_values(by='release_year')[['title', 'release_year', 'type', 'r

# Example usage:
get_director_titles("Martin Scorsese")
```

	title	release_year	type	rating
8735	Who's That Knocking at My Door?	1967	Movie	R
7431	Mean Streets	1973	Movie	R
6111	Alice Doesn't Live Here Anymore	1974	Movie	PG
7820	Raging Bull	1980	Movie	R
6880	GoodFellas	1990	Movie	R
6826	Gangs of New York	2002	Movie	R
2632	No Direction Home: Bob Dylan	2005	Movie	TV-MA
8272	The Departed	2006	Movie	R
1358	Shutter Island	2010	Movie	R
2860	Hugo	2011	Movie	PG
3759	Rolling Thunder Revue: A Bob Dylan Story by Ma...	2019	Movie	TV-MA
3227	The Irishman	2019	Movie	R

```
import seaborn as sns
import matplotlib.pyplot as plt

def plot_top_categories(column_name, top_n=10):
    plt.figure(figsize=(10, 5))
    df[column_name].value_counts().head(top_n).plot(kind='bar', color='skyblue')
    plt.title(f"Top {top_n} {column_name.capitalize()}s")
    plt.ylabel("Count")
    plt.xlabel(column_name.capitalize())
    plt.xticks(rotation=45)
    plt.grid(axis='y')
    plt.tight_layout()
    plt.show()
```

```
# Example usage:
plot_top_categories('country', top_n=10)
plot_top_categories('rating', top_n=7)
```


Top 10 Countrys
Count
United States
India
United Kingdom
Japan
South Korea
Canada
Spain
France
Mexico
Egypt
Country


Top 7 Ratings
Count
TV-MA
TV-14
TV-PG
R
PG-13
TV-Y7
TV-Y