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

```
data = pd.read_csv('/content/netflix.csv')
data.head()
```

	show_id	type	title	director	cast	country	date_added	release_y
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
print("Shape:", data.shape)
data.info()
```

```
Shape: (8807, 16)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 16 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       8807 non-null   object
 4   cast           8807 non-null   object
 5   country        8807 non-null   object
 6   date_added     8709 non-null   datetime64[ns]
 7   release_year   8807 non-null   int64
 8   rating         8807 non-null   object
 9   listed_in      8807 non-null   object
 10  description    8807 non-null   object
 11  movie_minutes  8807 non-null   float64
 12  show_seasons   8807 non-null   float64
 13  year_added     8709 non-null   float64
 14  month_added    8709 non-null   float64
 15  week_added     8709 non-null   UInt32
dtypes: UInt32(1), datetime64[ns](1), float64(4), int64(1), object(9)
memory usage: 1.1+ MB
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
```

NameError NameError Nai •••

Please explain the error:

```
KeyError: "['movie_minutes', 'show_seasons',
```

Hypothesis

The error `KeyError: "['movie_minutes', 'show_seasons', 'days_to_add'] not in index"` is raised because the columns `'movie_minutes'`, `'show_seasons'`, and `'days_to_add'` are not found in your DataFrame `data` when trying to select them using `data[numeric_cols]`. This likely happened because these columns were created or modified in a step that did not persist the changes to the `data` DataFrame. You might have accidentally overwritten the DataFrame or performed operations on a copy instead of the original.

Suggested Changes

```
numeric_cols = ['release_year', 'movie_minute
# Ensure these columns are actually present i
for col in numeric_cols:
    if col not in data.columns:
        print(f"Column '{col}' is missing fro
        # If missing, you might need to re-cr
        # For example, to re-create 'movie_mi
        # data.loc[data["type"]=="Movie","mov
        # ... (similarly for other missing co
data_numeric = data[numeric_cols].dropna()
# ... (rest of your code)
```

Use code with caution

▶ Sources

Rate this answer

```
 0   show_id        8807 non-null   object
 1   type           8807 non-null   object
 2   title          8807 non-null   object
 3   director       8807 non-null   object
 4   cast           8807 non-null   object
 5   country        8807 non-null   object
 6   date_added     8709 non-null   datetime64[ns]
 7   release_year   8807 non-null   int64
 8   rating         8807 non-null   object
 9   listed_in      8807 non-null   object
 10  description    8807 non-null   object
 11  movie_minutes  8807 non-null   float64
 12  show_seasons   8807 non-null   float64
 13  year_added     8709 non-null   float64
 14  month_added    8709 non-null   float64
 15  week_added     8709 non-null   UInt32
dtypes: UInt32(1), datetime64[ns](1), float64(4), int64(1), object(9)
memory usage: 1.1+ MB
```

```
data.duplicated().sum()
```

```
np.int64(0)
```

Start coding or generate with AI.

Duration column

```
data.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

```
data.loc[data["type"]=="Movie","movie_minutes"] = data.loc[data["type"]== "Movie","(
data.loc[data["type"]=="TV show","show_seasons"]= data.loc[data["type"]== "TV Show",
```

```
data.drop(columns="duration",inplace = True)
```

Null values

```
data.fillna({
    "director" : "Unknown",
    "cast" : "Unknown",
    "country" : "Unknown",
    "rating" : "Not available",
    "date_added" : "Unkonwn"
},inplace = True)
```

```
data.fillna({
    "movie_minutes" : 0,
    "show_seasons" : 0
},inplace= True)
```

```
from sre_constants import error
data["date_added"] = pd.to_datetime(data["date_added"],errors= "coerce")
```

```
<ipython-input-30-9c0cd943d5c2>:1: DeprecationWarning: module 'sre_constants' is
  from sre_constants import error
```

Un-nesting the values in "director,"cast","country" and "listed_in" columns

```
director = ( pd.DataFrame(data["director"].apply(lambda x: str(x).split(",")).tolist
.stack()
```

```
.reset_index(level = 1, drop = True)
.reset_index()
.rename(columns = {0:"Director"})
)


cast =  pd.DataFrame(data["cast"].apply(lambda x: str(x).split(",")).tolist(), index


country = ( pd.DataFrame(data["country"].apply(lambda x: str(x).split(",")).tolist(
.stack()
.reset_index(level = 1, drop = True)
.reset_index()
.rename(columns = {0:"country"})
)



listed_in = ( pd.DataFrame(data["listed_in"].apply(lambda x: str(x).split(",")).tol
.stack()
.reset_index(level = 1, drop = True)
.reset_index()
.rename(columns = {0:"listed_in"})
)




def unnest_column(data, column):
    return data.drop(column, axis=1).join(
        data[column].str.split(',', expand=True).stack().reset_index(level=1, drop=
    )


data['cast'] = data['cast'].apply(lambda x: x.split(',') if isinstance(x, str) else [
data['country'] = data['country'].apply(lambda x: x.split(',') if isinstance(x, str)
data['director'] = data['director'].apply(lambda x: x.split(',') if isinstance(x, str
data['listed_in'] = data['listed_in'].apply(lambda x: x.split(',') if isinstance(x, s

data = data.explode('cast')
data = data.explode('country')
data = data.explode('director')
data = data.explode('listed_in')


for col in ['cast', 'country', 'director', 'listed_in']:
    data[col] = data[col].str.strip()

data = data.drop_duplicates().reset_index(drop=True)

print("Final row count after unnesting:", data.shape[0])
```

```
Final row count after unnesting: 202010
```

Convert 'date_added' to datetime and extract time info

```
data['date_added'] = pd.to_datetime(data['date_added'], errors='coerce')
data['year_added'] = data['date_added'].dt.year
data['month_added'] = data['date_added'].dt.month
data['week_added'] = data['date_added'].dt.isocalendar().week


data.head()
```

	show_id	type	title	director	cast	country	date_added	release_y
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	2
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	2021-09-24	2
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	2021-09-24	2
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	2021-09-24	2

Next steps: Generate code with data | View recommended plots | New interactive sheet

Merging Dataframes

```
director_data = data.merge(director,on = "title",how = "left")

cast_data = data.merge(cast,on = "title",how = "left")

country_data = data.merge(country,on = "title",how = "left")

listed_in_data = data.merge(listed_in,on = "title",how = "left")
```

Double-click (or enter) to edit

```
data = data.drop_duplicates()

data = data.reset_index(drop=True)

num_movies = data[data['type'] == 'Movie'].shape[0]
print("Number of movies:", num_movies)
```

```
Number of movies: 6131
```

```
num_tv_show = data[data['type'] == 'TV show'].shape[0]
print("Number of TV show:", num_tv_show)
```

```
Number of TV show: 0
```

Data visulization non graphical visulization

```
categorical_columns = data.select_dtypes(include=['object', 'category']).columns.tol
print("Categorical columns:", categorical_columns)

for col in categorical_columns:
    print(f"\n--- Value counts for '{col}' ---")
    print(data[col].value_counts(dropna=False))
```

```
ca United States Japan 1
```

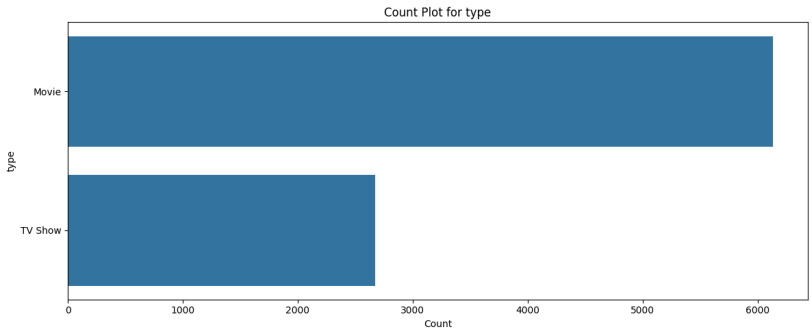
```
ica, United States, Japan                                  1
nt, Length: 749, dtype: int64

 counts for 'rating' ---

        3207
        2160
         863
         799
         490
         334
         307
         287
         220
          80
          41
           6
able       4
           3
           3
           1
           1
           1
nt, dtype: int64

 counts for 'listed_in' ---

nternational Movies                        362
ries                                       359
Comedy                                     334
 Dramas, International Movies              274
ndependent Movies, International Movies    252
                                          ...
dventure, Cult Movies                        1
dventure, Comedies, Music & Musicals         1
ovies, Horror Movies, Thrillers              1
 Family Movies, Classic Movies, Dramas       1
es, Dramas, Thrillers                        1
nt, Length: 514, dtype: int64

 counts for 'description' ---
on
 activity at a lush, abandoned property alarms a group eager to redevelop the
eptuagenarian gets another chance at her 20s after having her photo snapped at
omen report their husbands as missing but when it appears they are looking fo
 to compose 100 songs before he can marry the girl he loves, a tortured but p
 matriarch plots to cut off her disabled stepson and his wife from the family

ter the assassination of African American leader Malcolm X, an activist embar
rs after a disease that turns the infected into carnivorous insects emerged,
 governments fail to stop the slaughter of elephants for ivory, an 80-year-ol
C Comics' Green Arrow, an affluent playboy becomes a vengeful superhero, savi
sfit with a fondness for crafts, horses and supernatural crime shows finds he
nt, Length: 8775, dtype: int64
```

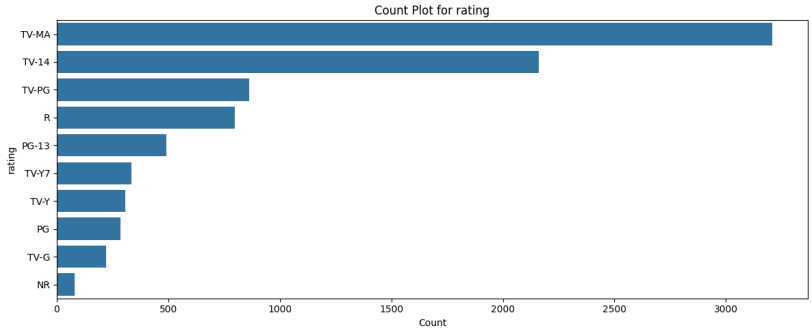
```
data['type'].value_counts()
data['country'].value_counts().head(10)
data['rating'].value_counts()
data['release_year'].nunique()
```

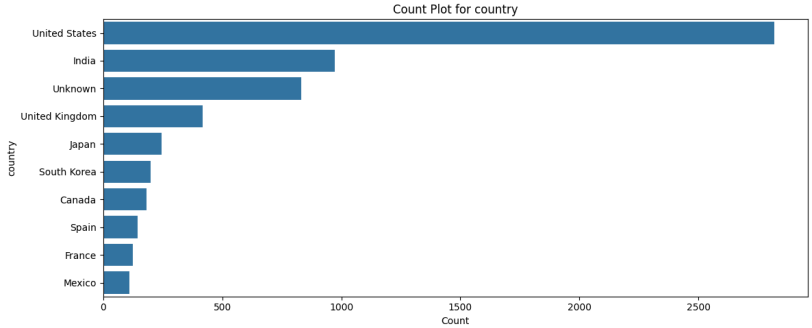
```
74
```

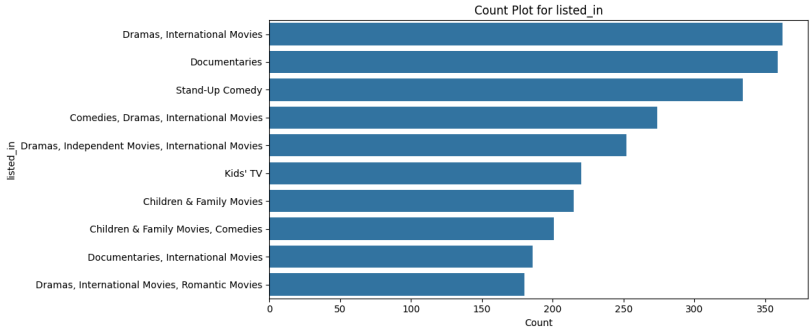
```
categorical_columns = ['type', 'rating', 'country', 'listed_in']
```

```
for col in categorical_columns:
    plt.figure(figsize=(12, 5))
    sns.countplot(data=data, y=col, order=data[col].value_counts().index[:10])
    plt.title(f'Count Plot for {col}')
    plt.xlabel('Count')
    plt.ylabel(col)
    plt.tight_layout()
    plt.show()
```


Count Plot for type
Movie
TV Show
type
0
1000
2000
3000
4000
5000
6000
Count


Count Plot for rating
TV-MA
TV-14
TV-PG
R
PG-13
TV-Y7
TV-Y
PG
TV-G
NR
rating
0
500
1000
1500
2000
2500
3000
Count

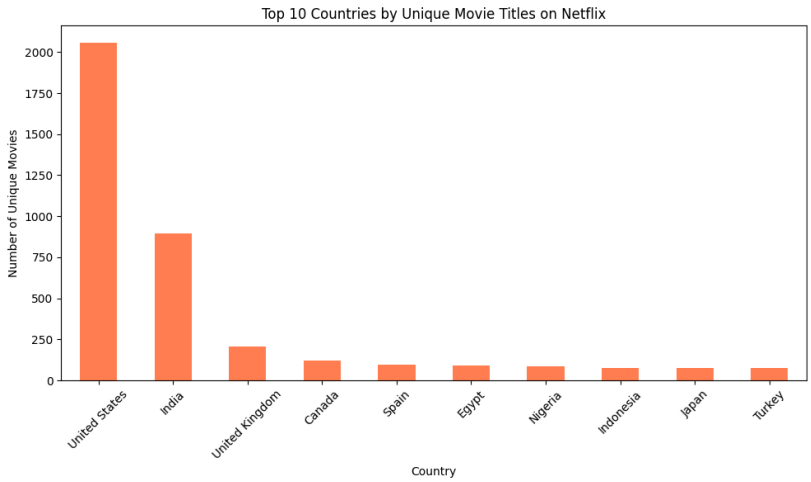

Count Plot for country
United States
India
Unknown
United Kingdom
Japan
South Korea
Canada
Spain
France
Mexico
country
0
500
1000
1500
2000
2500
Count


Count Plot for listed_in
Dramas, International Movies
Documentaries
Stand-Up Comedy
Comedies, Dramas, International Movies
Dramas, Independent Movies, International Movies
Kids' TV
Children & Family Movies
Children & Family Movies, Comedies
Documentaries, International Movies
Dramas, International Movies, Romantic Movies
listed_in
0
50
100
150
200
250
300
350
Count

```
movies_data = data[data['type'] == 'Movie']

top10_movie_countries = (
    movies_data.groupby('country')['title']
    .nunique()
    .sort_values(ascending=False)
    .head(10)
)
print("Top 10 countries by number of unique movies:")
print(top10_movie_countries)
```

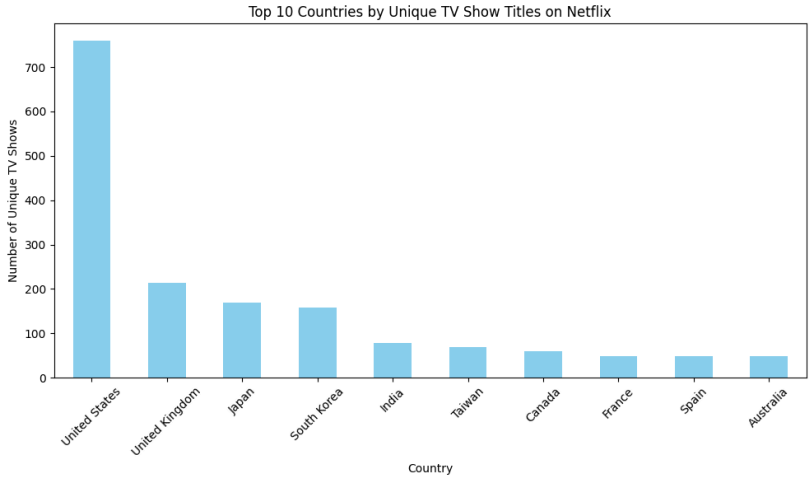
```
Top 10 countries by number of unique movies:
country
United States     2752
India              962
United Kingdom     534
Canada             319
France             303
Germany            182
Spain              171
Japan              119
China              114
Mexico             111
Name: title, dtype: int64
```

```
top10_movie_countries.plot(kind='bar', figsize=(10, 6), color='coral')
plt.title("Top 10 Countries by Unique Movie Titles on Netflix")
plt.ylabel("Number of Unique Movies")
plt.xlabel("Country")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



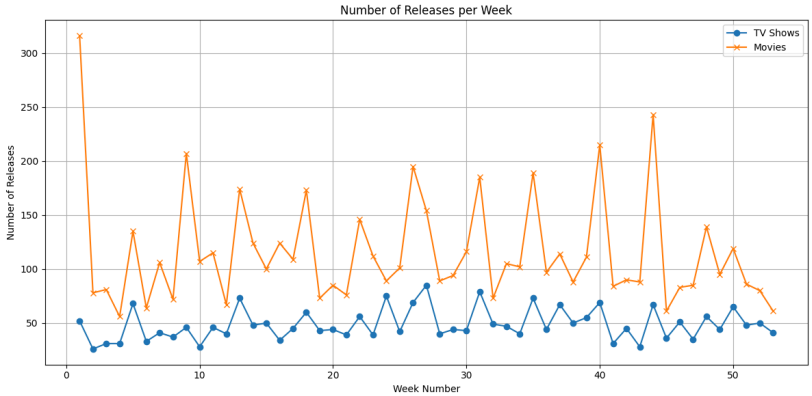

```
tv_data = data[data['type'] == 'TV Show']
tv_counts = tv_data.groupby('country')['title'].nunique()
top10_tv_countries = tv_counts.sort_values(ascending=False).head(10)
print("Top 10 countries by unique TV Show titles:")
print(top10_tv_countries)

top10_tv_countries.plot(kind='bar', figsize=(10, 6), color='skyblue')
plt.title("Top 10 Countries by Unique TV Show Titles on Netflix")
plt.ylabel("Number of Unique TV Shows")
plt.xlabel("Country")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Top 10 countries by unique TV Show titles:
country
United States     760
United Kingdom    213
Japan             169
South Korea       158
India              79
Taiwan             68
Canada             59
France             49
Spain              48
Australia          48
Name: title, dtype: int64
```



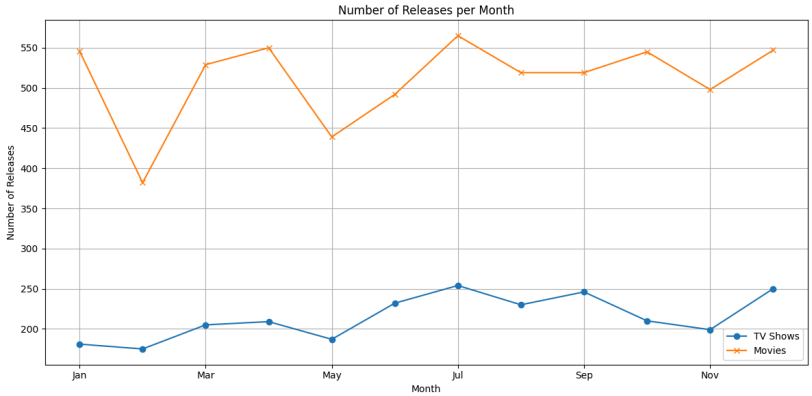

```
data['date_added'] = pd.to_datetime(data['date_added'], errors='coerce')

data['week_added'] = data['date_added'].dt.isocalendar().week

tv_shows = data[data['type'] == 'TV Show']
movies = data[data['type'] == 'Movie']
tv_weekly = tv_shows.groupby('week_added')['title'].count()
movie_weekly = movies.groupby('week_added')['title'].count()
best_tv_week = tv_weekly.idxmax()
best_movie_week = movie_weekly.idxmax()

print(f"Best week to release TV Shows: Week {best_tv_week} with {tv_weekly.max()} re
print(f"Best week to release Movies: Week {best_movie_week} with {movie_weekly.max(

plt.figure(figsize=(12,6))
tv_weekly.plot(label='TV Shows', marker='o')
movie_weekly.plot(label='Movies', marker='x')
plt.title("Number of Releases per Week")
plt.xlabel("Week Number")
plt.ylabel("Number of Releases")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Best week to release TV Shows: Week 27 with 85 releases
Best week to release Movies: Week 1 with 316 releases
```




```
data['date_added'] = pd.to_datetime(data['date_added'], errors='coerce')

data['month_added'] = data['date_added'].dt.month

tv_shows = data[data['type'] == 'TV Show']
movies = data[data['type'] == 'Movie']

tv_monthly = tv_shows.groupby('month_added')['title'].count()
movie_monthly = movies.groupby('month_added')['title'].count()

best_tv_month = tv_monthly.idxmax()
best_movie_month = movie_monthly.idxmax()

print(f"📺 Best month to release TV Shows: Month {best_tv_month} with {tv_monthly.m
print(f"🎬 Best month to release Movies: Month {best_movie_month} with {movie_month

month_labels = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
                'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

# Plot
plt.figure(figsize=(12,6))

tv_monthly.index = tv_monthly.index.astype(int).map(lambda x: month_labels[x-1])
movie_monthly.index = movie_monthly.index.astype(int).map(lambda x: month_labels[x-1

tv_monthly.plot(label='TV Shows', marker='o')
movie_monthly.plot(label='Movies', marker='x')
plt.title("Number of Releases per Month")
plt.xlabel("Month")
plt.ylabel("Number of Releases")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Best month to release TV Shows: Month 7.0 with 254 releases
Best month to release Movies: Month 7.0 with 565 releases
```




```
top_tv_actors = data[data['type'] == 'TV Show'].groupby('cast')['title'].nunique().s
top_tv_actors.columns = ['Actor', 'Unique TV Shows']

top_movie_actors = data[data['type'] == 'Movie'].groupby('cast')['title'].nunique().
top_movie_actors.columns = ['Actor', 'Unique Movies']

# Display
print("Top 10 Actors in TV Shows:")
display(top_tv_actors)

top_tv_actors = data[data['type'] == 'TV Show'].groupby('cast')['title'].nunique().s
top_tv_actors.columns = ['Actor', 'Unique TV Shows']

# Actors in Movies
top_movie_actors = data[data['type'] == 'Movie'].groupby('cast')['title'].nunique().
top_movie_actors.columns = ['Actor', 'Unique Movies']

# Display
print("Top 10 Actors in TV Shows:")
display(top_tv_actors)

print("Top 10 Actors in Movies:")
display(top_movie_actors)
# Step 2: Top 10 Directors in TV Shows and Movies (Separate)

# Directors in TV Shows
top_tv_directors = data[data['type'] == 'TV Show'].groupby('director')['title'].nuni
top_tv_directors.columns = ['Director', 'Unique TV Shows']

# Directors in Movies
top_movie_directors = data[data['type'] == 'Movie'].groupby('director')['title'].nur
top_movie_directors.columns = ['Director', 'Unique Movies']

# Display
print("Top 10 Directors in TV Shows:")
display(top_tv_directors)

print("Top 10 Directors in Movies:")
display(top_movie_directors)
#Edit
# Directors in TV Shows
top_tv_directors = data[data['type'] == 'TV Show'].groupby('director')['title'].nuni
top_tv_directors.columns = ['Director', 'Unique TV Shows']

# Directors in Movies
top_movie_directors = data[data['type'] == 'Movie'].groupby('director')['title'].nur
top_movie_directors.columns = ['Director', 'Unique Movies']

# Display
print("Top 10 Directors in TV Shows:")
display(top_tv_directors)
```

```
print("Top 10 Directors in Movies:")
display(top_movie_directors)
```

```
Top 10 Actors in TV Shows:
```

	Actor	Unique TV Shows
0	David Attenborough	14
1	Michela Luci, Jamie Watson, Anna Claire Bartla...	4
2	Dave Chappelle	3
3	Marie Kondo	2
4	Chris Packham	2
5	Erik Thompson	2
6	Bettany Hughes	2
7	Monty Don	2
8	You, Reina Triendl, Yoshimi Tokui, Azusa Babaz...	2
9	Morgan Freeman	2

```
Top 10 Actors in TV Shows:
```

	Actor	Unique TV Shows
0	David Attenborough	14
1	Michela Luci, Jamie Watson, Anna Claire Bartla...	4
2	Dave Chappelle	3
3	Marie Kondo	2
4	Chris Packham	2
5	Erik Thompson	2
6	Bettany Hughes	2
7	Monty Don	2
8	You, Reina Triendl, Yoshimi Tokui, Azusa Babaz...	2
9	Morgan Freeman	2

```
Top 10 Actors in Movies:
```

	Actor	Unique Movies
0	Vatsal Dubey, Julie Tejwani, Rupa Bhimani, Jig...	13
1	Samuel West	10
2	Jeff Dunham	7
3	Craig Sechler	6
4	Kevin Hart	6
5	Jim Gaffigan	5
6	Bill Burr	5
7	Iliza Shlesinger	5
8	Michela Luci, Jamie Watson, Eric Peterson, Ann...	5
9	David Spade, London Hughes, Fortune Feimster	5

```
Top 10 Directors in TV Shows:
```

	Director	Unique TV Shows
0	Alastair Fothergill	3
1	Hsu Fu-chun	2
2	Iginio Straffi	2
3	Rob Seidenglanz	2
4	Stan Lathan	2
5	Ken Burns	2
6	Shin Won-ho	2
7	Alejandro Lozano	1
8	Alessandro Angulo	1
9	Alex Gibney	1

```
Top 10 Directors in Movies:
```

	Director	Unique Movies
0	Rajiv Chilaka	19
1	Raúl Campos, Jan Suter	18

2	Suhas Kadav	16
3	Marcus Raboy	15
4	Jay Karas	14
5	Cathy Garcia-Molina	13
6	Jay Chapman	12
7	Martin Scorsese	12
8	Youssef Chahine	12
9	Steven Spielberg	11

```
Top 10 Directors in TV Shows:
```

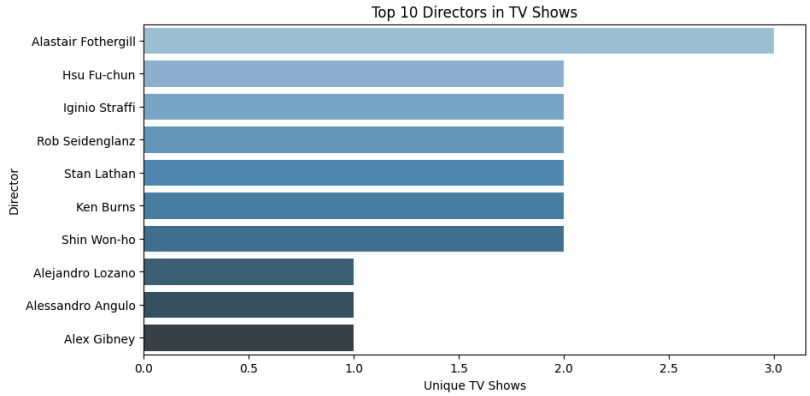
	Director	Unique TV Shows
0	Alastair Fothergill	3
1	Hsu Fu-chun	2
2	Iginio Straffi	2
3	Rob Seidenglanz	2
4	Stan Lathan	2
5	Ken Burns	2
6	Shin Won-ho	2
7	Alejandro Lozano	1
8	Alessandro Angulo	1
9	Alex Gibney	1

```
Top 10 Directors in Movies:
```

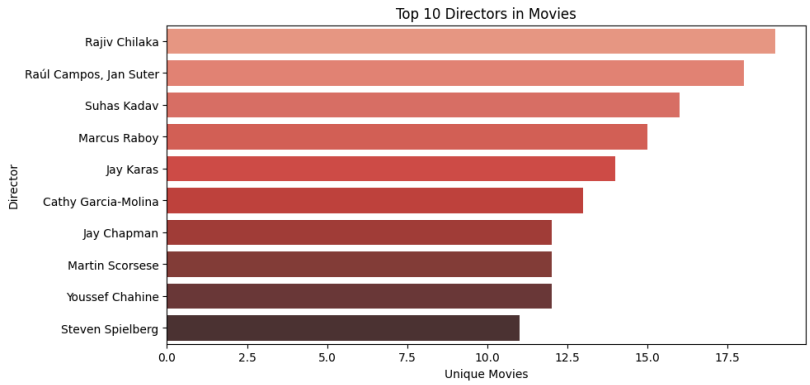
	Director	Unique Movies
0	Rajiv Chilaka	19
1	Raúl Campos, Jan Suter	18
2	Suhas Kadav	16
3	Marcus Raboy	15
4	Jay Karas	14
5	Cathy Garcia-Molina	13
6	Jay Chapman	12
7	Martin Scorsese	12
8	Youssef Chahine	12
9	Steven Spielberg	11

```
plt.figure(figsize=(10, 5))
sns.barplot(data=top_tv_directors, x='Unique TV Shows', y='Director', palette='Blues
plt.title('Top 10 Directors in TV Shows')
plt.show()

plt.figure(figsize=(10, 5))
sns.barplot(data=top_movie_directors, x='Unique Movies', y='Director', palette='Reds
plt.title('Top 10 Directors in Movies')
plt.show()
```

```
<ipython-input-21-14143bd8d639>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v

  sns.barplot(data=top_tv_directors, x='Unique TV Shows', y='Director', palette=
```




```
<ipython-input-21-14143bd8d639>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v

  sns.barplot(data=top_movie_directors, x='Unique Movies', y='Director', palette
```




```
!pip install wordcloud
from wordcloud import WordCloud

genre_text = ' '.join(data['listed_in'].dropna().astype(str))

# Generate the word cloud
wordcloud = WordCloud(width=1000, height=500, background_color='white', colormap='ta

# Display the word cloud
plt.figure(figsize=(12, 6))
```

```
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Most Frequent Genres on Netflix', fontsize=16)
plt.show()
```

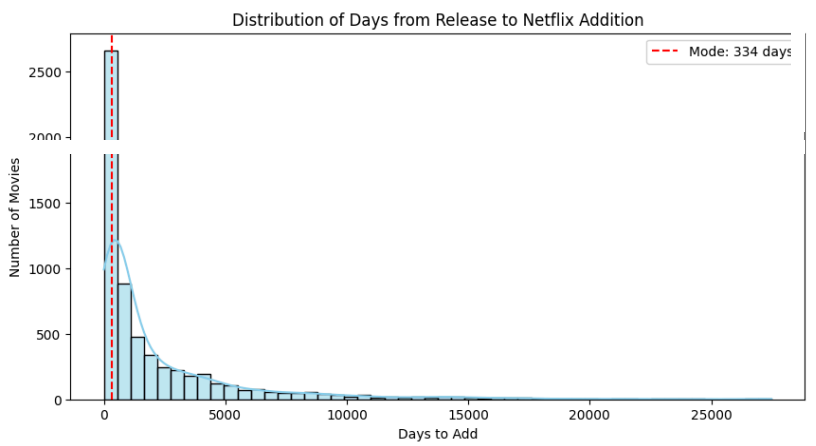
```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packag
```

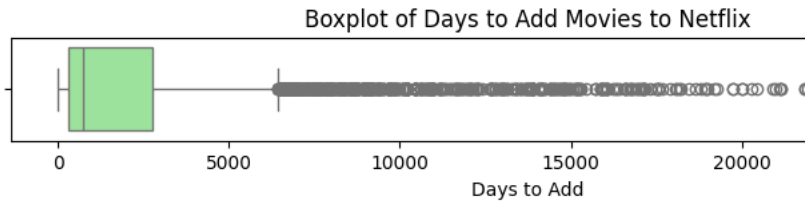

```
data['date_added'] = pd.to_datetime(data['date_added'], errors='coerce')

movies = data[(data['type'] == 'Movie') & (data['date_added'].notna()) & (data['rele

movies['release_date'] = pd.to_datetime(movies['release_year'].astype(str), format='

movies['days_to_add'] = (movies['date_added'] - movies['release_date']).dt.days

movies = movies[movies['days_to_add'] >= 0]

most_common_delay = movies['days_to_add'].mode()[0]

print("Most common number of days after release that a movie is added to Netflix:",
```

```
Most common number of days after release that a movie is added to Netflix: 334
<ipython-input-25-3a1ea2192dcb>:7: 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/stal
  movies['release_date'] = pd.to_datetime(movies['release_year'].astype(str), fo
<ipython-input-25-3a1ea2192dcb>:10: 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/stal
  movies['days_to_add'] = (movies['date_added'] - movies['release_date']).dt.day
```

```
plt.figure(figsize=(10, 5))
sns.histplot(movies['days_to_add'], bins=50, kde=True, color='skyblue')
plt.axvline(movies['days_to_add'].mode()[0], color='red', linestyle='--', label=f"
plt.title('Distribution of Days from Release to Netflix Addition')
plt.xlabel('Days to Add')
plt.ylabel('Number of Movies')
```

```
plt.legend()
plt.show()

plt.figure(figsize=(10, 1))
sns.boxplot(x=movies['days_to_add'], color='lightgreen')
plt.title('Boxplot of Days to Add Movies to Netflix')
plt.xlabel('Days to Add')
plt.show()
```





Double-click (or enter) to edit

```
data['date_added'] = pd.to_datetime(data['date_added'], errors='coerce')
data['year_added'] = data['date_added'].dt.year
yearly_counts = data.groupby(['year_added', 'type'])['title'].count().unstack(fill_v
print("TV Shows vs Movies (Recent Years):")
print(yearly_counts[yearly_counts.index >= 2017])
```

```
TV Shows vs Movies (Recent Years):
type        Movie  TV Show
year_added
2017.0        839      325
2018.0       1237      388
2019.0       1424      575
2020.0       1284      594
2021.0        993      505
```

```
yearly_counts[yearly_counts.index >= 2010].plot(kind='bar', stacked=False, figsize=(
plt.title('Number of Movies vs TV Shows Added to Netflix (per Year)')
plt.xlabel('Year Added')
plt.ylabel('Number of Titles')
plt.xticks(rotation=45)
plt.legend(title='Type')
plt.tight_layout()
plt.show()
```