In [72]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

pd.set_option('display.max_colwidth', 200)
```

In [21]:
```
df = pd.read_csv('../data/netflix1.csv')

# Show first 5 rows
df.head()
```

Out[21]:

	show_id	type	title	director	country	date_added	release_year	rating	du
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	United States	9/25/2021	2020	PG-13	
1	s3	TV Show	Ganglands	Julien Leclercq	France	9/24/2021	2021	TV-MA	1
2	s6	TV Show	Midnight Mass	Mike Flanagan	United States	9/24/2021	2021	TV-MA	1
3	s14	Movie	Confessions of an Invisible Girl	Bruno Garotti	Brazil	9/22/2021	2021	TV-PG	
4	s8	Movie	Sankofa	Haile Gerima	United States	9/24/2021	1993	TV-MA	1

In [15]:
```
# Check shape (rows, columns)
df.shape

# Check column info
df.info()

# Count missing values
df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8790 entries, 0 to 8789
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8790 non-null   object
 1   type          8790 non-null   object
 2   title         8790 non-null   object
 3   director      8790 non-null   object
 4   country       8790 non-null   object
 5   date_added    8790 non-null   object
 6   release_year  8790 non-null   int64 
 7   rating        8790 non-null   object
 8   duration      8790 non-null   object
 9   listed_in     8790 non-null   object
dtypes: int64(1), object(9)
memory usage: 686.8+ KB
```

Out[15]:
```
show_id         0
type            0
title           0
director        0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
dtype: int64
```

In [ ]:

In [ ]:

In [31]:
```
data = pd.read_csv(r"C:\Users\Surendranath TV\Desktop\netflix project\data\netfl
```

In [ ]:

In [35]:
```
# Step 1: Remove duplicates
print("Duplicates before:", data.duplicated().sum())
data.drop_duplicates(inplace=True)
print("Duplicates after:", data.duplicated().sum())

# Step 2: Handle missing values safely
print("\nMissing values before cleaning:")
print(data.isnull().sum())

# Only drop columns if they exist
cols_to_clean = ['director', 'cast', 'country']
existing_cols = [col for col in cols_to_clean if col in data.columns]

if existing_cols:
    data.dropna(subset=existing_cols, inplace=True)
else:
    print("⚠️ None of the target columns found for cleaning.")

print("\nMissing values after cleaning:")
print(data.isnull().sum())
```

```
# Step 3: Convert 'date_added' to datetime if present
if 'date_added' in data.columns:
    data['date_added'] = pd.to_datetime(data['date_added'], errors='coerce')
    print("\nData types after converting:")
    print(data.dtypes)
else:
    print("⚠️ 'date_added' column not found.")
```

```
Duplicates before: 0
Duplicates after: 0

Missing values before cleaning:
show_id         0
type            0
title           0
director        0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
dtype: int64

Missing values after cleaning:
show_id         0
type            0
title           0
director        0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
dtype: int64

Data types after converting:
show_id                 object
type                    object
title                   object
director                object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
dtype: object
```

In [37]:

```
# Step 1: Treat Nulls
# Fill missing values in director, cast, country with "Not Given"
for col in ['director', 'cast', 'country']:
    if col in data.columns:
        data[col].fillna("Not Given", inplace=True)

print("Missing values after filling:")
print(data.isnull().sum())

# Step 2: Treat Duplicates
```

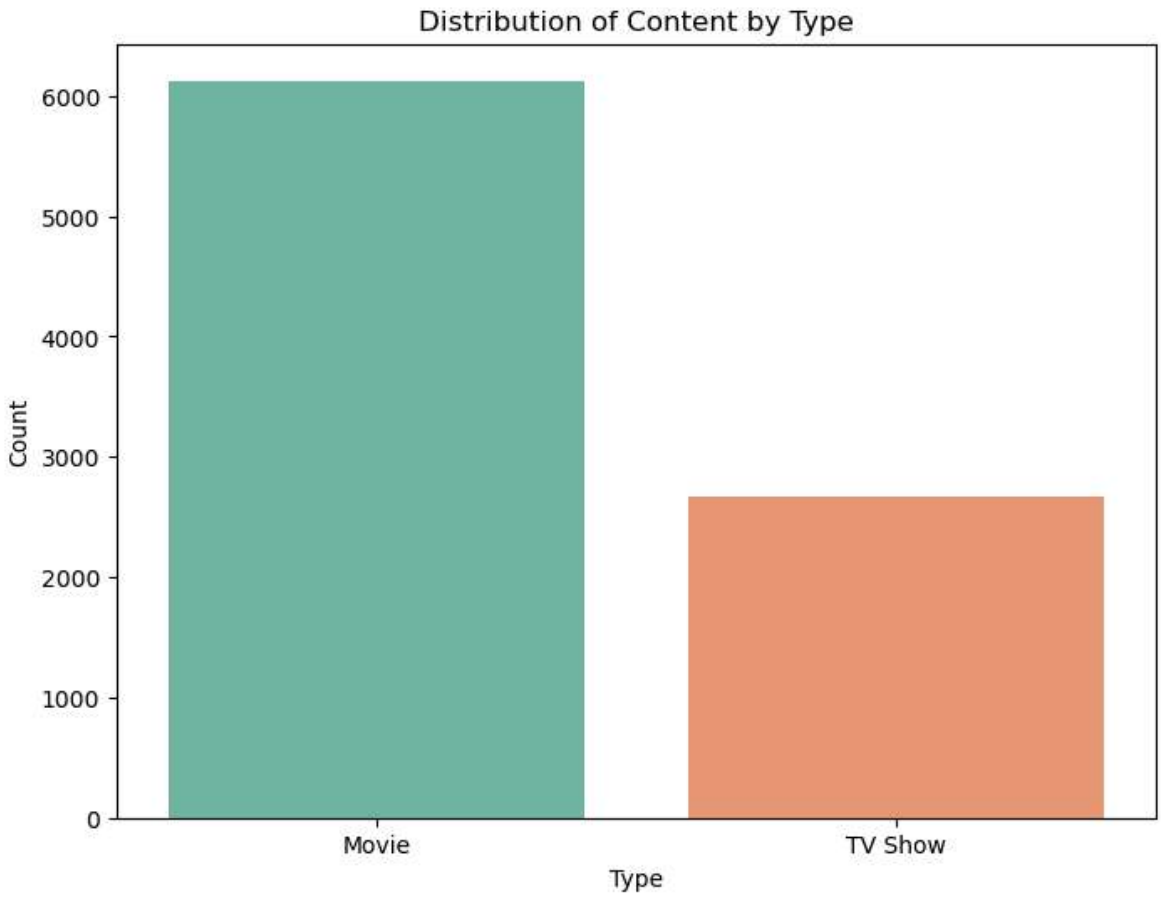
```
print("\nDuplicates before:", data.duplicated().sum())
data.drop_duplicates(inplace=True)
print("Duplicates after:", data.duplicated().sum())

# Step 3: Populate Missing Rows
# (Handled above by filling "Not Given" instead of dropping)

# Step 4: Drop Unneeded Columns
# Drop show_id (not useful for analysis)
if 'show_id' in data.columns:
    data.drop(columns=['show_id'], inplace=True)
    print("\nDropped column: show_id")

# Step 5: Split Columns
# Split genres from 'listed_in' into a list
if 'listed_in' in data.columns:
    data['genres'] = data['listed_in'].apply(lambda x: x.split(','))
    print("\nGenres column created!")

# Extra: Convert date_added to datetime (if it exists)
if 'date_added' in data.columns:
    data['date_added'] = pd.to_datetime(data['date_added'], errors='coerce')

# Final check
print("\nData types after cleaning:")
print(data.dtypes)

print("\nFinal shape:", data.shape)
```

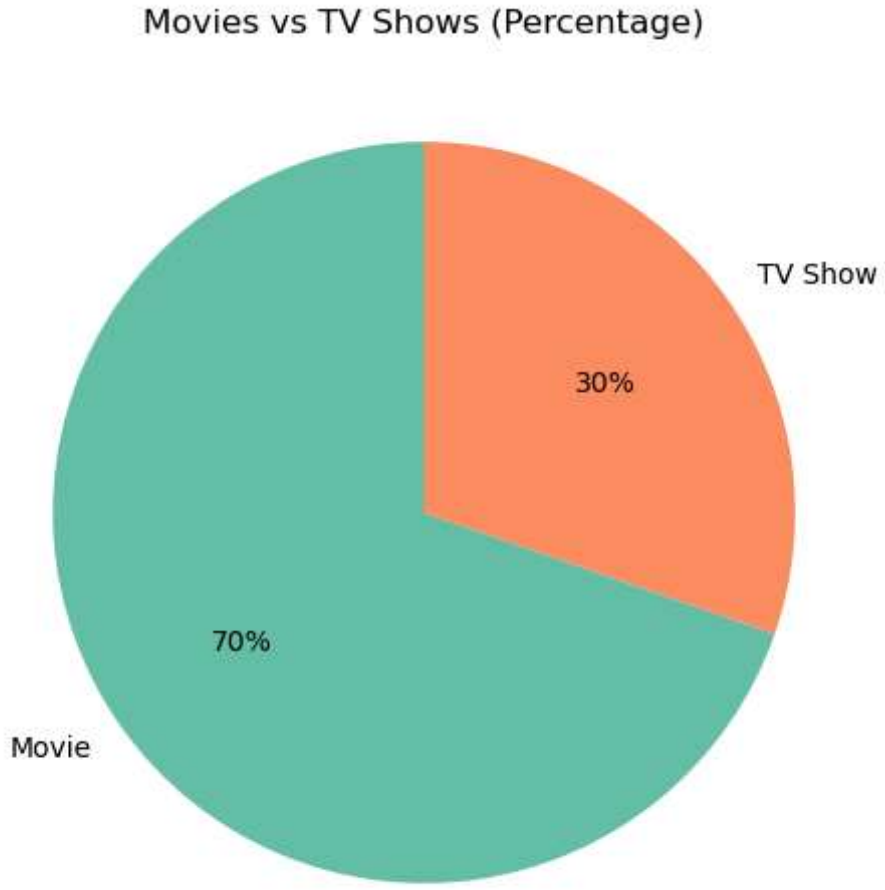
```
C:\Users\Surendranath TV\AppData\Local\Temp\ipykernel_9032\3072471028.py:5: Futur
eWarning: A value is trying to be set on a copy of a DataFrame or Series through
chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  data[col].fillna("Not Given", inplace=True)
```

```
Missing values after filling:
show_id         0
type            0
title           0
director        0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
dtype: int64

Duplicates before: 0
Duplicates after: 0

Dropped column: show_id

Genres column created!

Data types after cleaning:
type                     object
title                    object
director                 object
country                  object
date_added       datetime64[ns]
release_year              int64
rating                   object
duration                 object
listed_in                object
genres                   object
dtype: object

Final shape: (8790, 10)
```

In [39]:
```
# Count the number of Movies and TV Shows
type_counts = data['type'].value_counts()

# Plot the distribution (Bar chart)
plt.figure(figsize=(8, 6))
sns.barplot(x=type_counts.index, y=type_counts.values, palette='Set2')
plt.title('Distribution of Content by Type')
plt.xlabel('Type')
plt.ylabel('Count')
plt.show()

# Plot the distribution (Pie chart)
plt.figure(figsize=(6, 6))
plt.pie(type_counts.values, labels=type_counts.index, autopct='%.0f%%', startang
plt.title('Movies vs TV Shows (Percentage)')
plt.show()
```

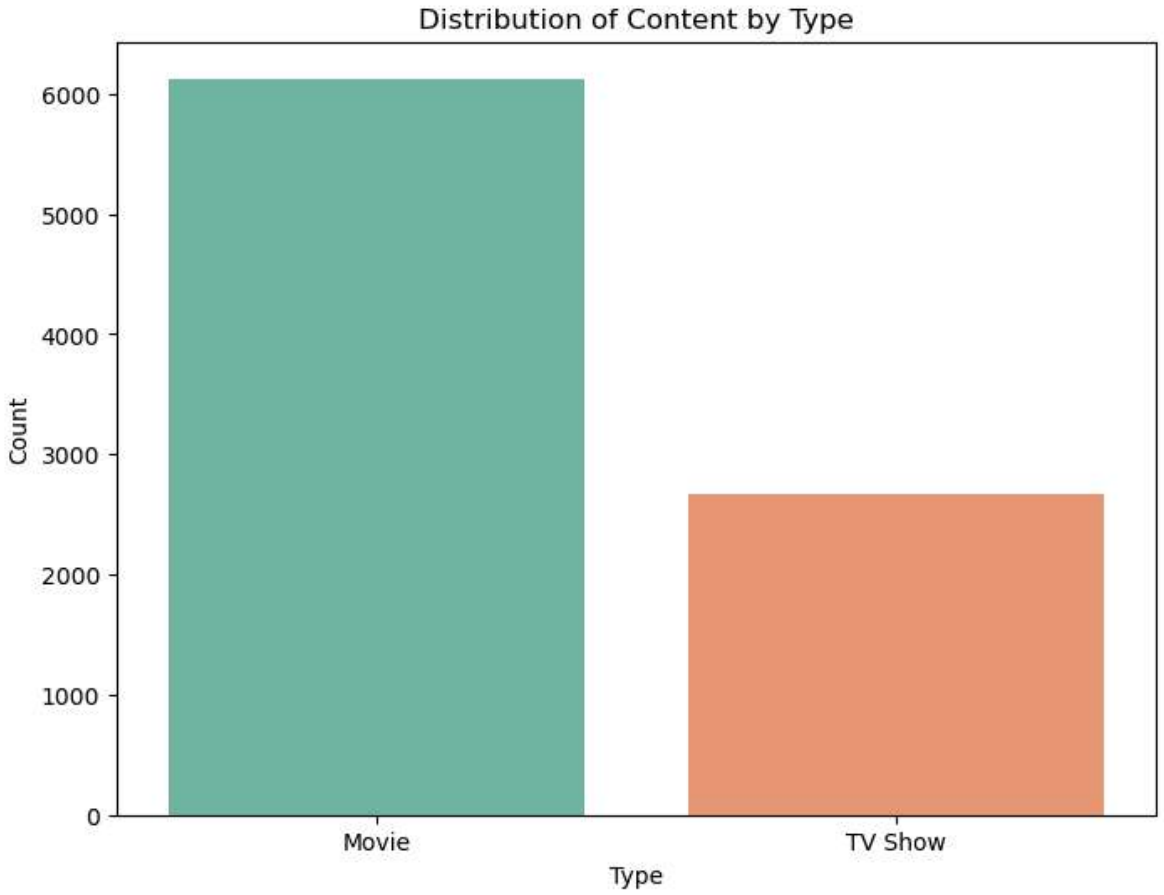
```
C:\Users\Surendranath TV\AppData\Local\Temp\ipykernel_9032\855350275.py:6: Future
Warning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x=type_counts.index, y=type_counts.values, palette='Set2')
```





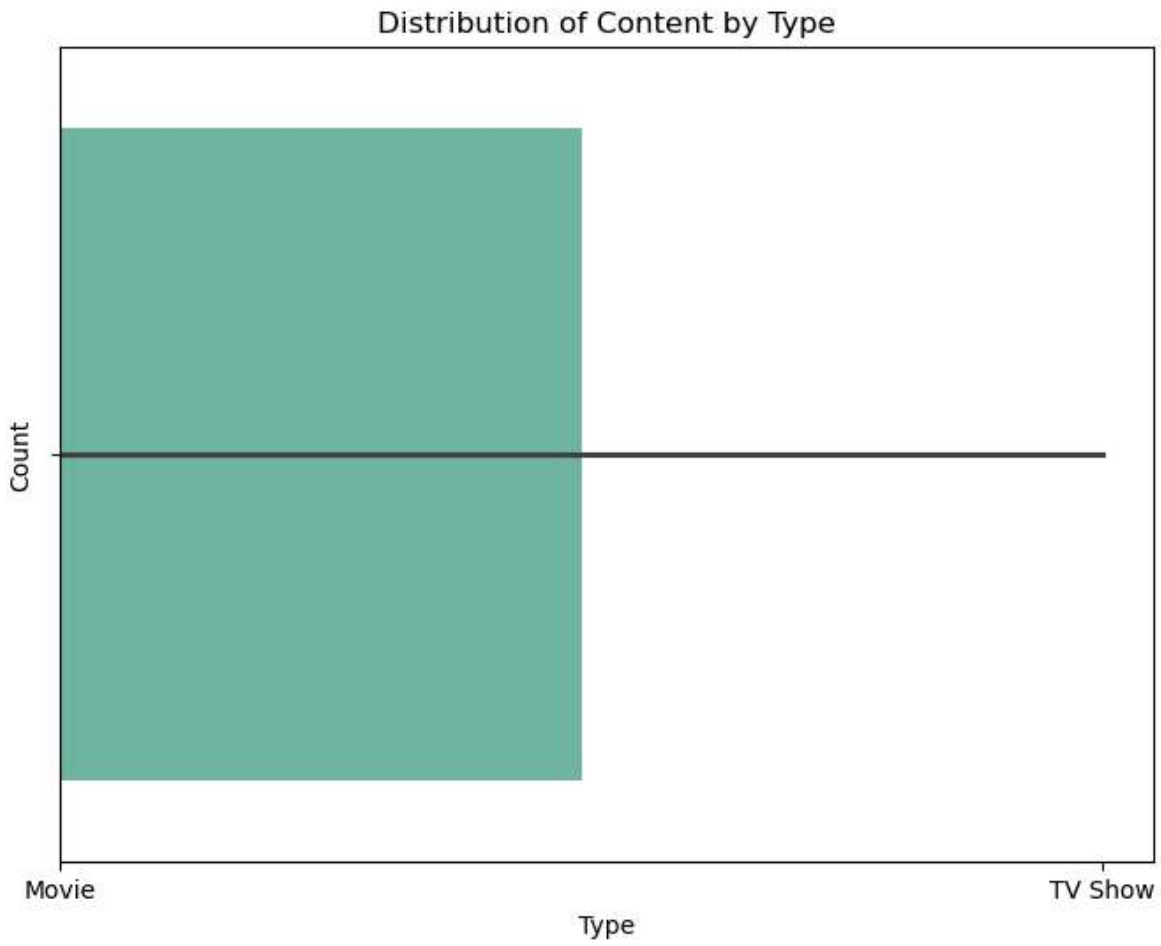
In [41]:
```
plt.figure(figsize=(8, 6))
sns.barplot(x=type_counts.index, y=type_counts.values, hue=type_counts.index, pa
```

```
plt.title('Distribution of Content by Type')
plt.xlabel('Type')
plt.ylabel('Count')
plt.show()
```



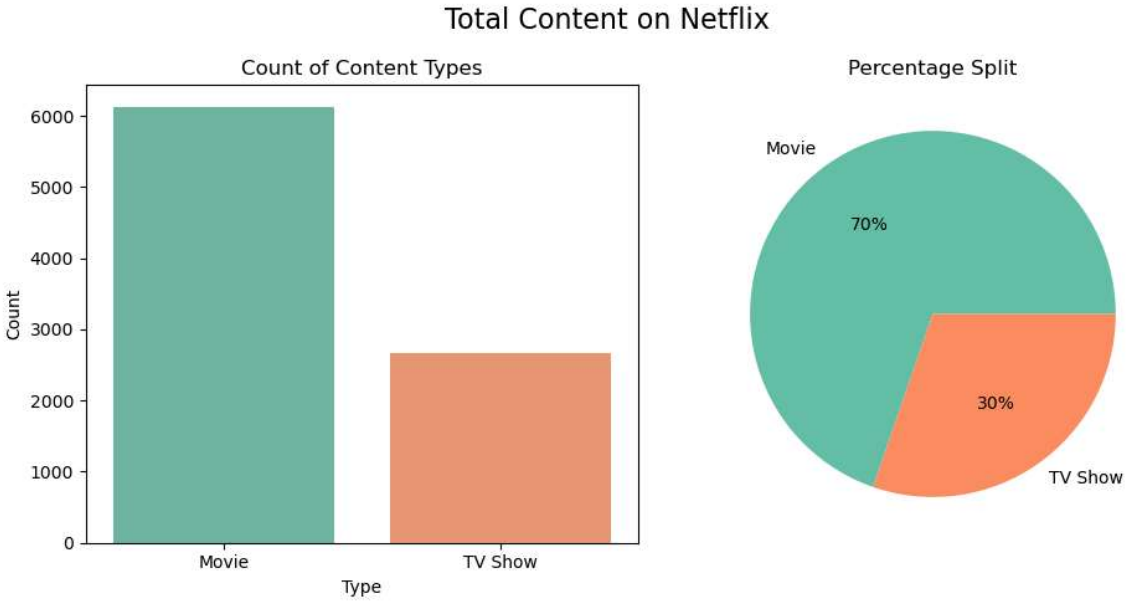
In [43]:
```
type_counts = data['type'].value_counts()
# Plot the distribution
plt.figure(figsize=(8, 6))
sns.barplot(x=type_counts.index,
palette='Set2')
y=type_counts.values,
plt.title('Distribution of Content by Type')
plt.xlabel('Type')
plt.ylabel('Count')
plt.show()
```

```
C:\Users\Surendranath TV\AppData\Local\Temp\ipykernel_9032\4157910420.py:4: Futur
eWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x=type_counts.index,
```



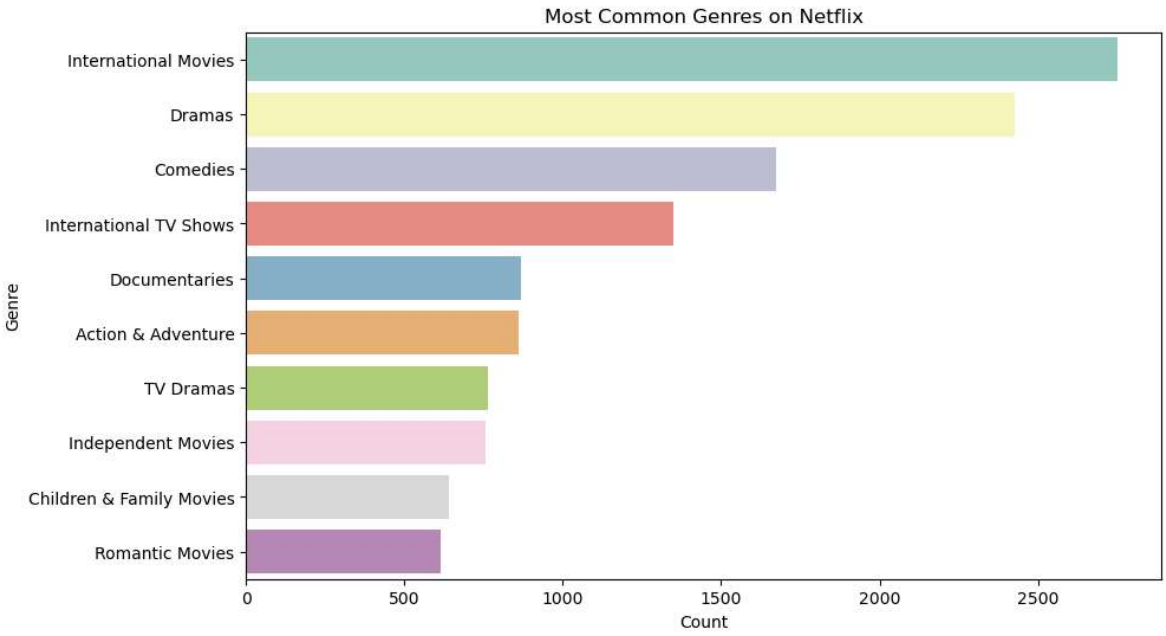
In [45]:
```
# Count frequency of Movies vs TV Shows
freq = data['type'].value_counts()

# Create a figure with 2 subplots
fig, axes = plt.subplots(1, 2, figsize=(10, 5))

# Left: countplot
sns.countplot(x='type', data=data, ax=axes[0], palette='Set2')
axes[0].set_title("Count of Content Types")
axes[0].set_xlabel("Type")
axes[0].set_ylabel("Count")

# Right: pie chart
axes[1].pie(freq.values, labels=freq.index, autopct='%.0f%%', colors=sns.color_p
axes[1].set_title("Percentage Split")

# Overall title
plt.suptitle("Total Content on Netflix", fontsize=16)
plt.tight_layout()
plt.show()
```

```
C:\Users\Surendranath TV\AppData\Local\Temp\ipykernel_9032\1281326807.py:8: Futur
eWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(x='type', data=data, ax=axes[0], palette='Set2')
```



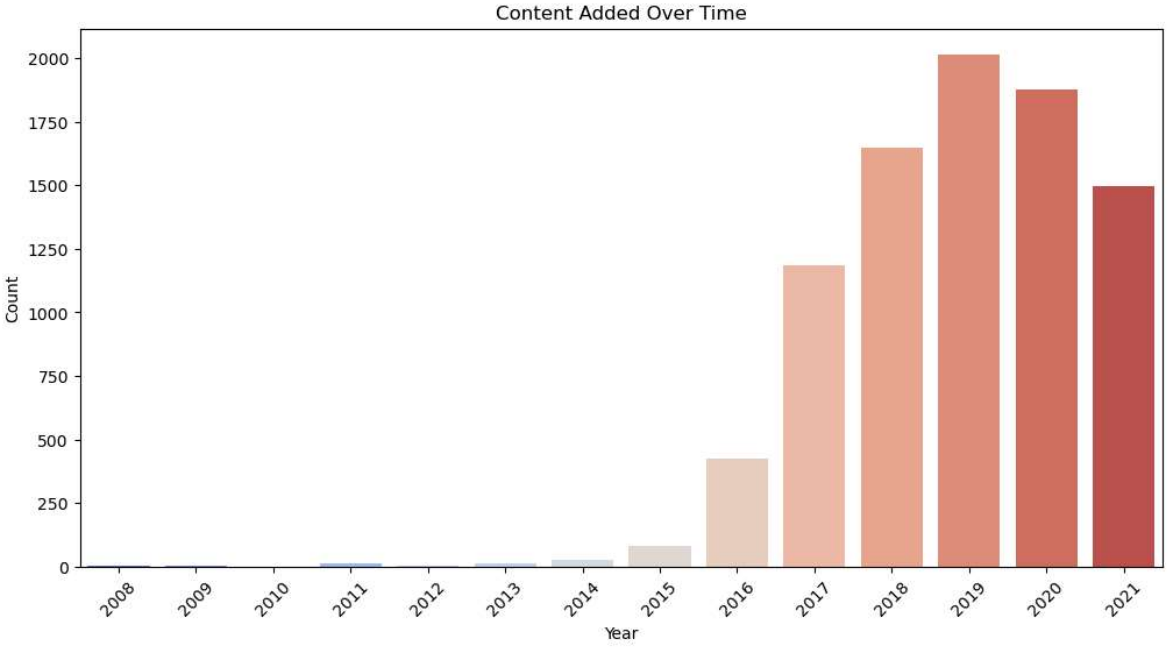
In [47]:
```
# Split the 'listed_in' column into genres
data['genres'] = data['listed_in'].apply(lambda x: x.split(', '))

# Flatten the list of genres
all_genres = sum(data['genres'], [])

# Count top 10 genres
genre_counts = pd.Series(all_genres).value_counts().head(10)

# Plot the most common genres
plt.figure(figsize=(10, 6))
sns.barplot(x=genre_counts.values, y=genre_counts.index, palette='Set3')
plt.title('Most Common Genres on Netflix')
plt.xlabel('Count')
plt.ylabel('Genre')
plt.show()
```

```
C:\Users\Surendranath TV\AppData\Local\Temp\ipykernel_9032\3606265878.py:12: Futu
reWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x=genre_counts.values, y=genre_counts.index, palette='Set3')
```



In [51]:
```
#Extract year and month from 'date_added'
y=genre_counts.index,
data['year_added'] = data['date_added'].dt.year
data['month_added'] = data['date_added'].dt.month
# Plot content added over the years
plt.figure(figsize=(12, 6))
sns.countplot(x='year_added', data=data, palette='coolwarm')
plt.title('Content Added Over Time')
plt.xlabel('Year')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

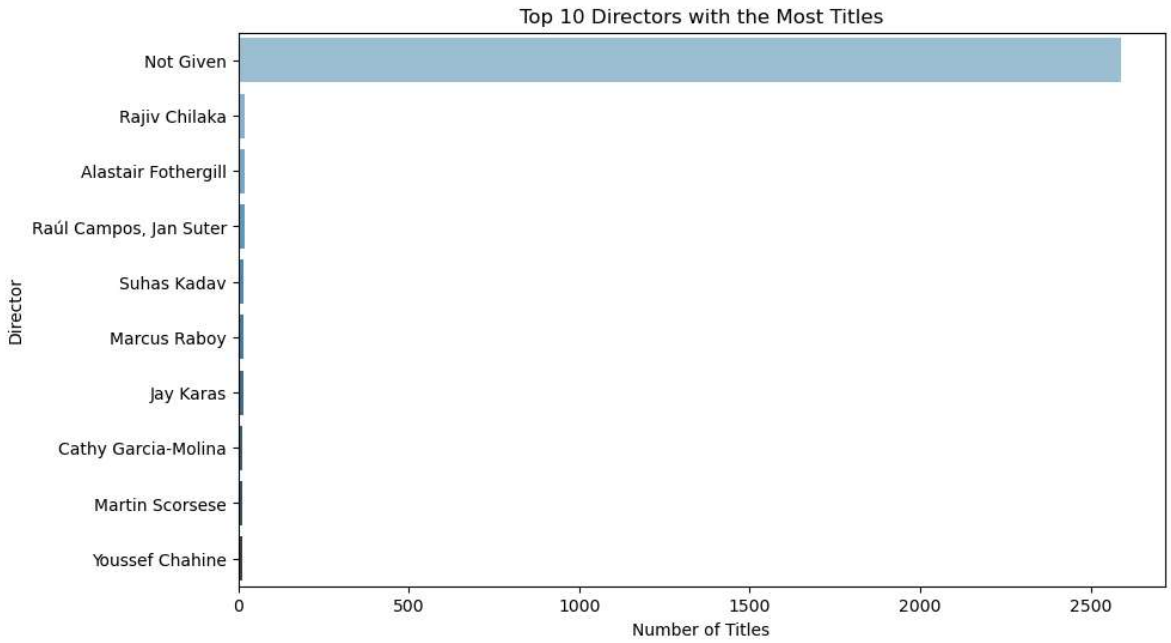
```
C:\Users\Surendranath TV\AppData\Local\Temp\ipykernel_9032\979967686.py:7: Future
Warning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(x='year_added', data=data, palette='coolwarm')
```



In [ ]:

In [55]:
```
# Count titles by director
top_directors = data['director'].value_counts().head(10)

# Plot top directors
plt.figure(figsize=(10, 6))
sns.barplot(x=top_directors.values, y=top_directors.index, palette='Blues_d')
plt.title('Top 10 Directors with the Most Titles')
plt.xlabel('Number of Titles')
plt.ylabel('Director')
plt.show()
```

```
C:\Users\Surendranath TV\AppData\Local\Temp\ipykernel_9032\1405477472.py:6: Futur
eWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x=top_directors.values, y=top_directors.index, palette='Blues_d')
```



In [59]:
```
plt.pie(ratings['count'][:8], labels=ratings['rating'][:8], autopct='%.0f%%')
plt.suptitle('Rating on Netflix', fontsize=20)
plt.show()
```

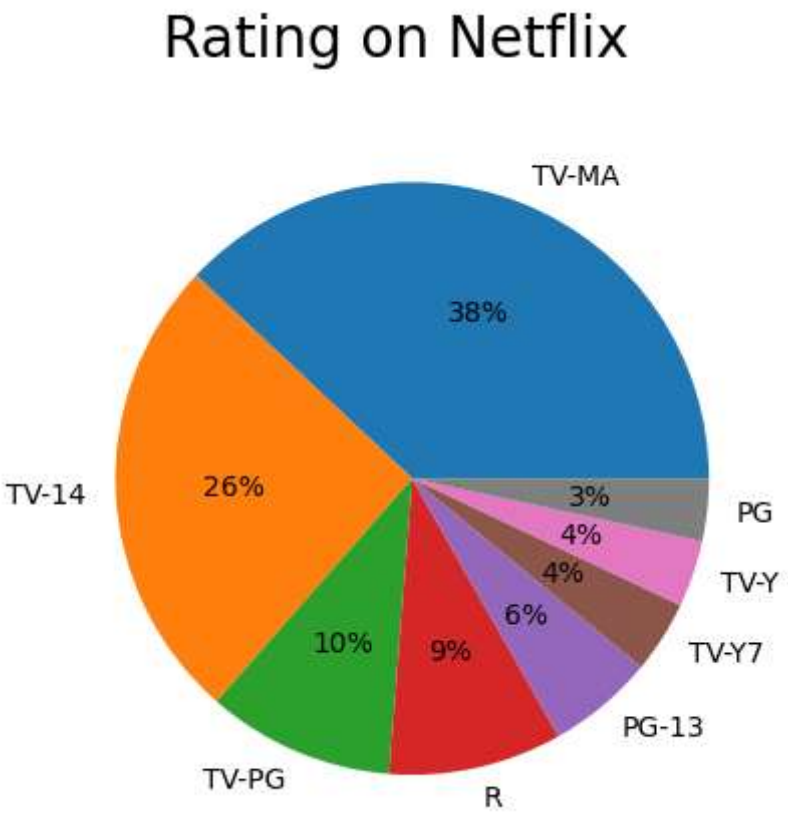
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[59], line 1
----> 1 plt.pie(rating['count'][:8], labels=rating['rating'][:8], autopct='%.0f%
%')
      2 plt.suptitle('Rating on Netflix', fontsize=20)
      3 plt.show()

NameError: name 'rating' is not defined
```

In [4]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
data = pd.read_csv("C:\Users\Surendranath TV\Downloads\Netflix Data_ Cleaning, A
print(data.shape)

# Count rating frequencies
ratings = data['rating'].value_counts().reset_index()
ratings.columns = ['rating', 'count']
ratings = ratings.sort_values(by='count', ascending=False)

# Debug check
print(ratings.head())

# Pie chart for top 8 ratings
plt.pie(ratings['count'][:8], labels=ratings['rating'][:8], autopct='%.0f%%')
plt.suptitle('Rating on Netflix', fontsize=20)
plt.show()
```

```
  Cell In[4], line 6
    data = pd.read_csv("C:\Users\Surendranath TV\Downloads\Netflix Data_ Cleanin
g, Analysis and Visualization_ _ ML _ FA _ DA projects.pdf")  # Make sure the pat
h is correct
                       ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position
2-3: truncated \UXXXXXXXX escape
```

In [ ]:

In [70]:
```
s# create the `ratings` DataFrame (top 8)
ratings = data['rating'].value_counts().nlargest(8).reset_index()
ratings.columns = ['rating', 'count']

# then plot
plt.pie(ratings['count'], labels=ratings['rating'], autopct='%.0f%%')
plt.suptitle('Rating on Netflix', fontsize=20)
plt.show()
```



In [74]:
```
 # lets convert column date_added to datetime.
 data['date_added']=pd.to_datetime(data['date_added'])
```

In [76]:
```
data.describe()
```

Out[76]:

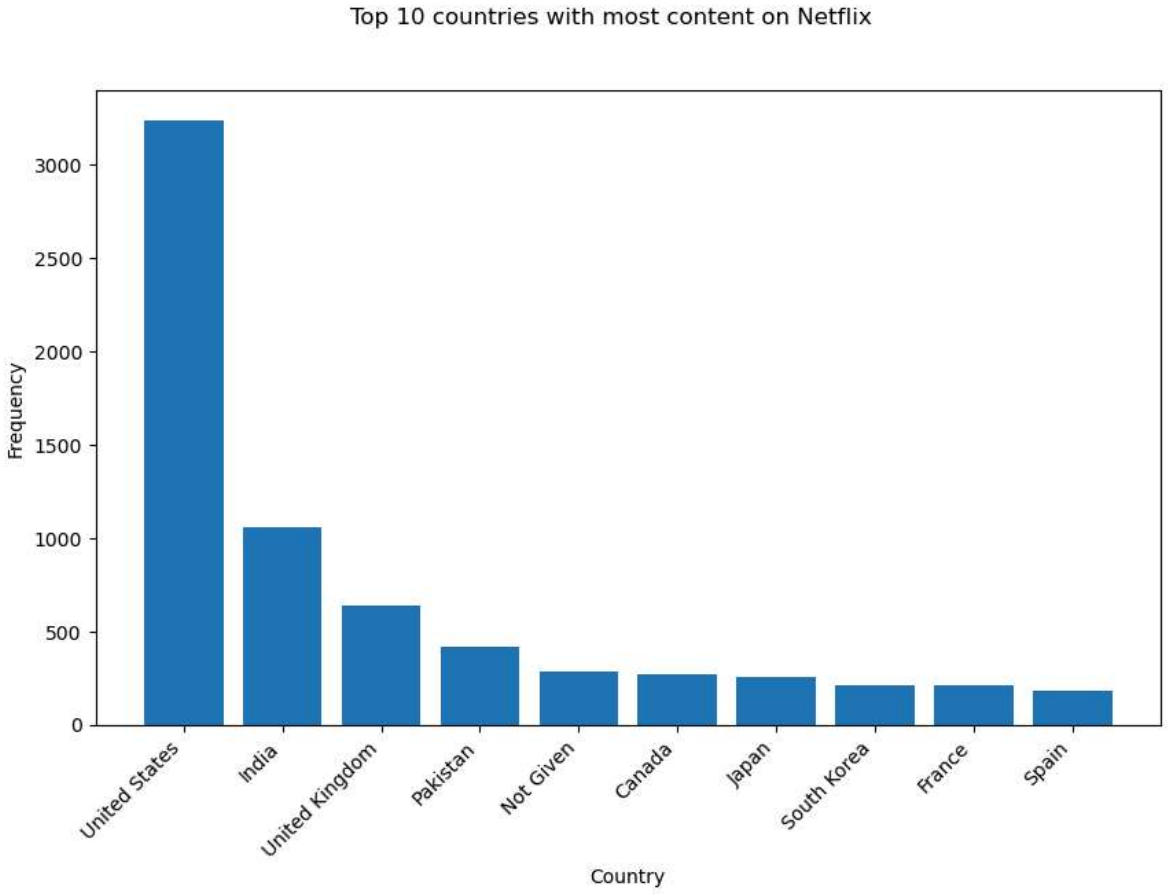
	date_added	release_year	year_added	month_added
count	8790	8790.000000	8790.000000	8790.000000
mean	2019-05-17 21:44:01.638225408	2014.183163	2018.873606	6.655859
min	2008-01-01 00:00:00	1925.000000	2008.000000	1.000000
25%	2018-04-06 00:00:00	2013.000000	2018.000000	4.000000
50%	2019-07-03 00:00:00	2017.000000	2019.000000	7.000000
75%	2020-08-19 18:00:00	2019.000000	2020.000000	10.000000
max	2021-09-25 00:00:00	2021.000000	2021.000000	12.000000
std	NaN	8.825466	1.573568	3.436103

In [78]:
```
data['country'].value_counts()
```

Out[78]:
```
country
United States     3240
India             1057
United Kingdom     638
Pakistan           421
Not Given          287
                  ...
Iran                 1
West Germany         1
Greece               1
Zimbabwe             1
Soviet Union         1
Name: count, Length: 86, dtype: int64
```

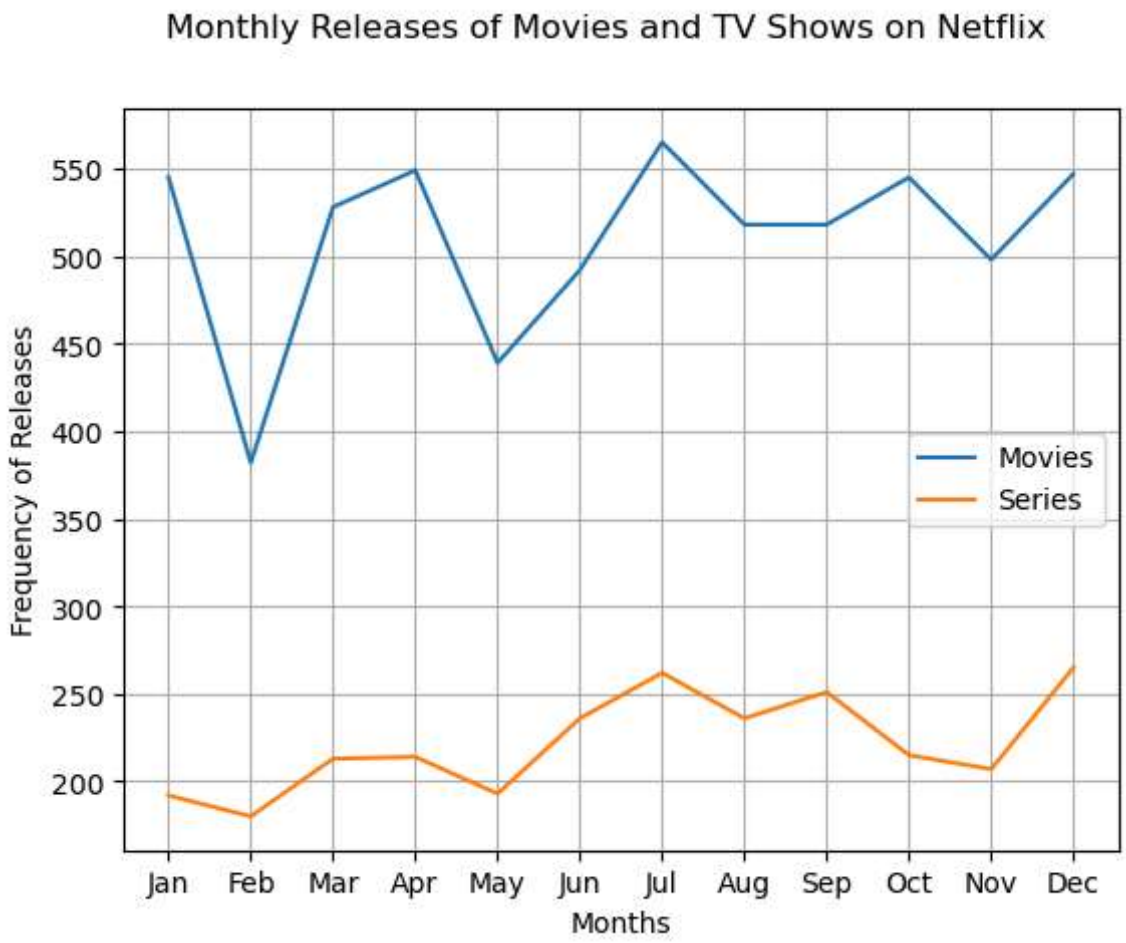
In [ ]:
```

```

In [86]:
```
top_ten_countries=data['country'].value_counts().reset_index().sort_values(by='
plt.figure(figsize=(10,6))
plt.bar(top_ten_countries['country'],
top_ten_countries['count'])
plt.xticks(rotation=45,ha='right')
plt.xlabel("Country")
plt.ylabel("Frequency")
plt.suptitle("Top 10 countries with most content on Netflix")
plt.show()
```



In [88]:
```
data['year']=data['date_added'].dt.year
data['month']=data['date_added'].dt.month
data['day']=data['date_added'].dt.day
```

In [90]:
```
# Monthly movie releases
monthly_movie_release = data[data['type'] == 'Movie']['month'].value_counts().so

# Monthly TV show releases
monthly_series_release = data[data['type'] == 'TV Show']['month'].value_counts()

# Plotting
plt.plot(monthly_movie_release.index, monthly_movie_release.values, label='Movie
plt.plot(monthly_series_release.index, monthly_series_release.values, label='Ser

plt.xlabel("Months")
plt.ylabel("Frequency of Releases")
plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May',
                          'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
plt.legend()
plt.grid(True)
plt.suptitle("Monthly Releases of Movies and TV Shows on Netflix")
plt.show()
```

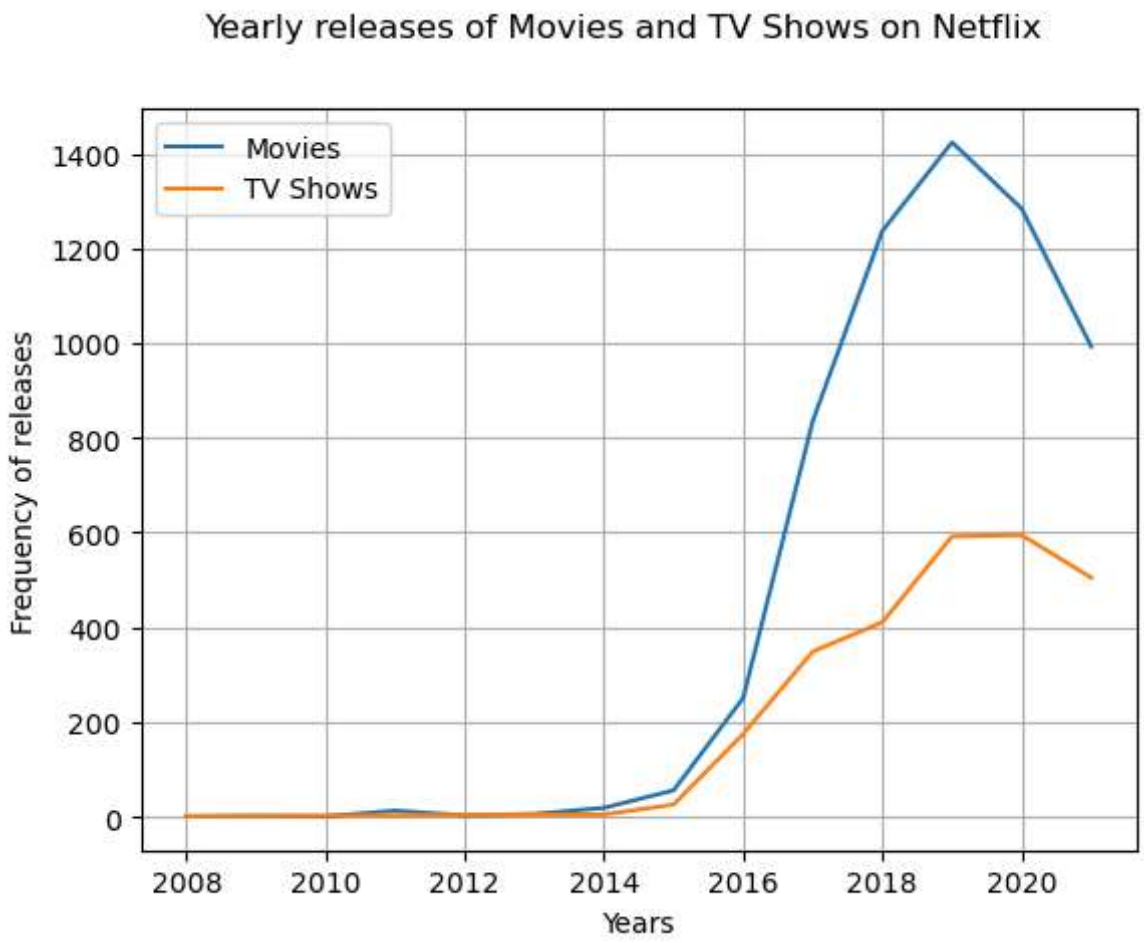


In [100…

```
yearly_movie_releases=data[data['type']=='Movie']['year'].value_counts().sort_i
yearly_series_releases=data[data['type']=='TV Show']['year'].value_counts().sor
plt.plot(yearly_movie_releases.index,
yearly_movie_releases.values, label='Movies')
plt.plot(yearly_series_releases.index,
yearly_series_releases.values, label='TV Shows')
plt.xlabel("Years")
plt.ylabel("Frequency of releases")
plt.grid(True)
plt.suptitle("Yearly releases of Movies and TV Shows on Netflix")
plt.legend()
```

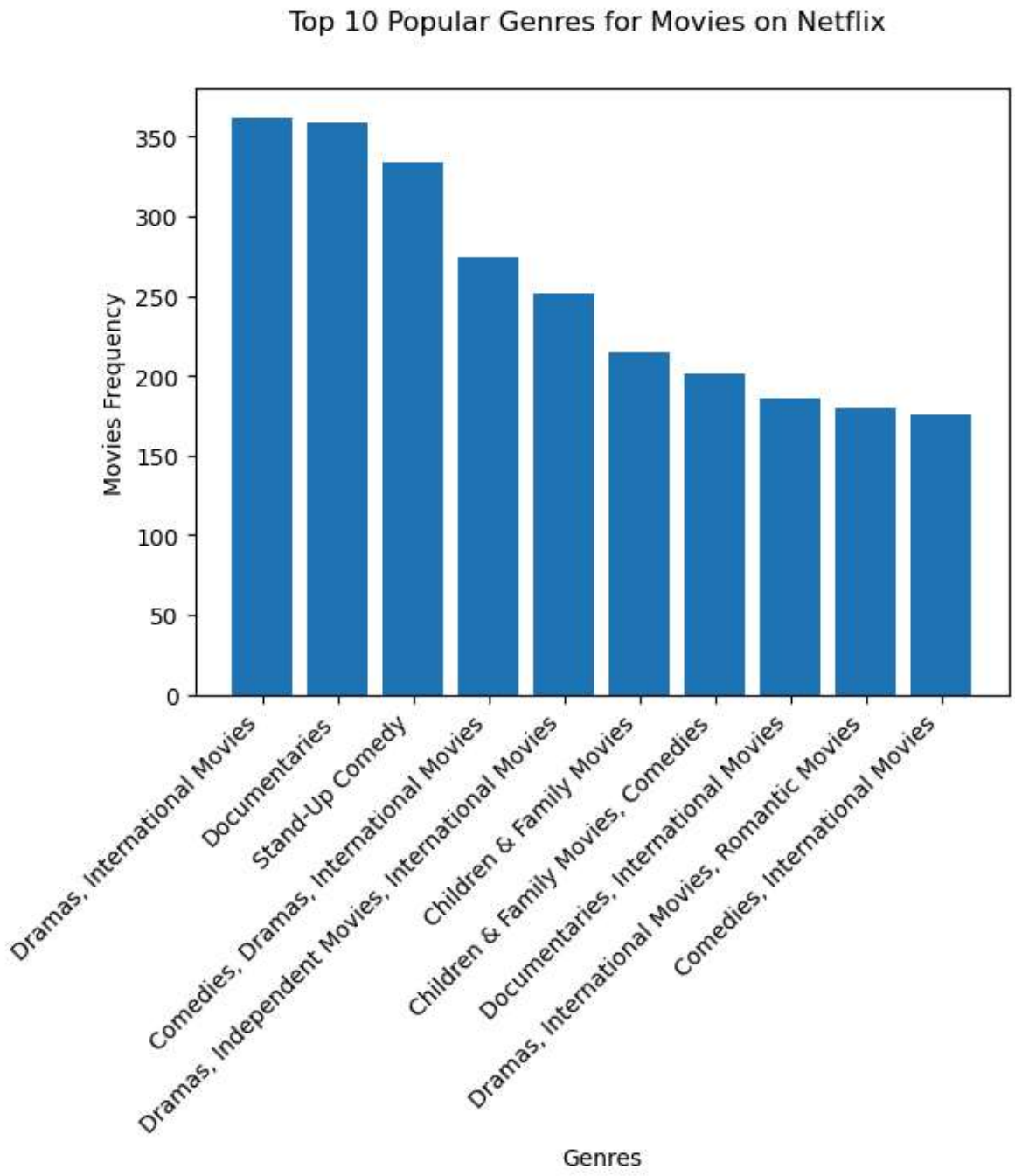
Out[100…

```
<matplotlib.legend.Legend at 0x15c520e86b0>
```



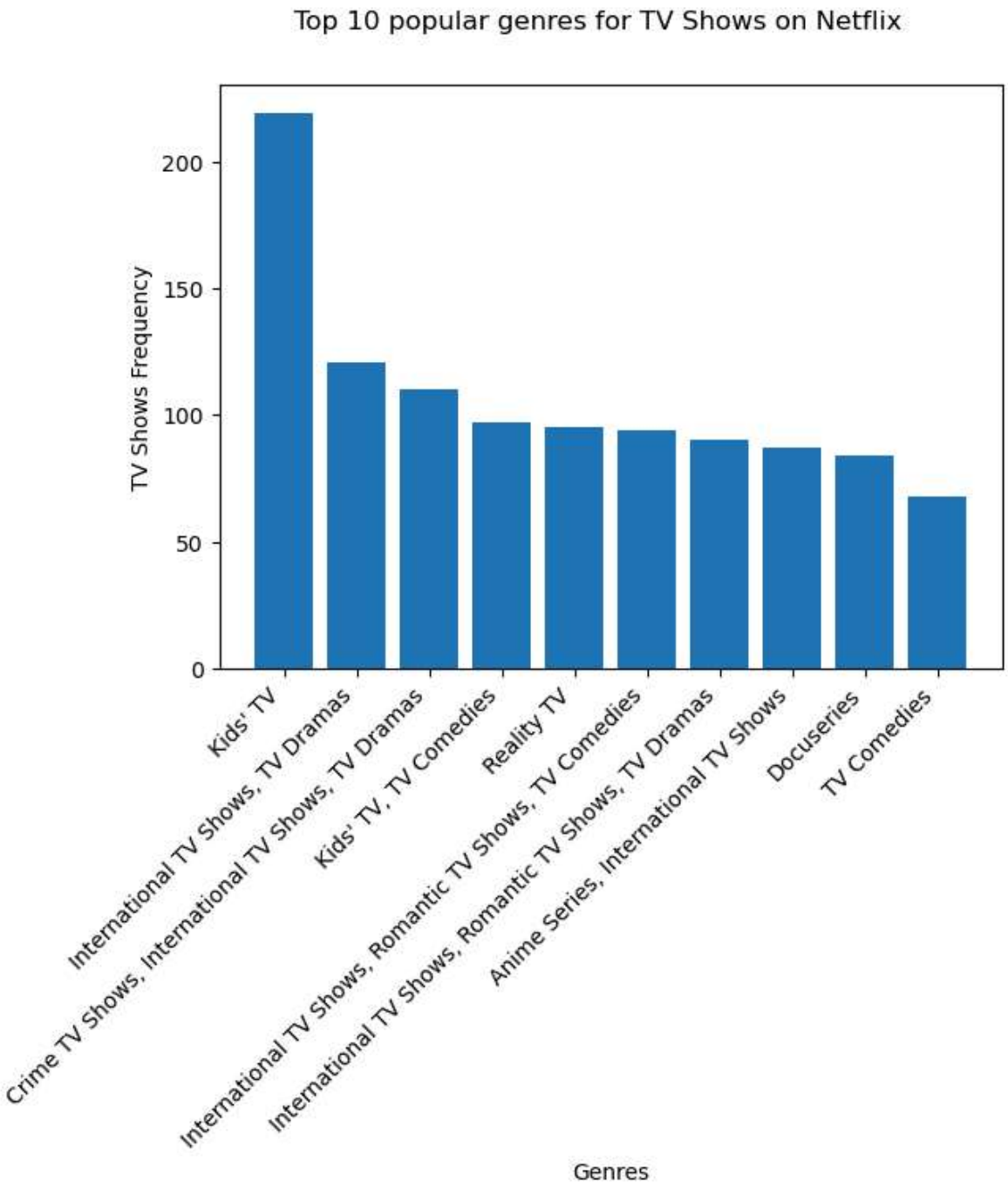
In [102…

```
# Top 10 popular movie genres
popular_movie_genre = (
    data[data['type'] == 'Movie']
    .groupby("listed_in")
    .size()
    .sort_values(ascending=False)[:10]
)

# Top 10 popular TV show genres
popular_series_genre = (
    data[data['type'] == 'TV Show']
    .groupby("listed_in")
    .size()
    .sort_values(ascending=False)[:10]
)

# Plot for movies
plt.bar(popular_movie_genre.index, popular_movie_genre.values)
plt.xticks(rotation=45, ha='right')
plt.xlabel("Genres")
plt.ylabel("Movies Frequency")
plt.suptitle("Top 10 Popular Genres for Movies on Netflix")
plt.show()
```



In [104…

```
plt.bar(popular_series_genre.index,popular_series_genre.values)
plt.xticks(rotation=45, ha='right')
plt.xlabel("Genres")
plt.ylabel("TV Shows Frequency")
plt.suptitle("Top 10 popular genres for TV Shows on Netflix")
plt.show()
```

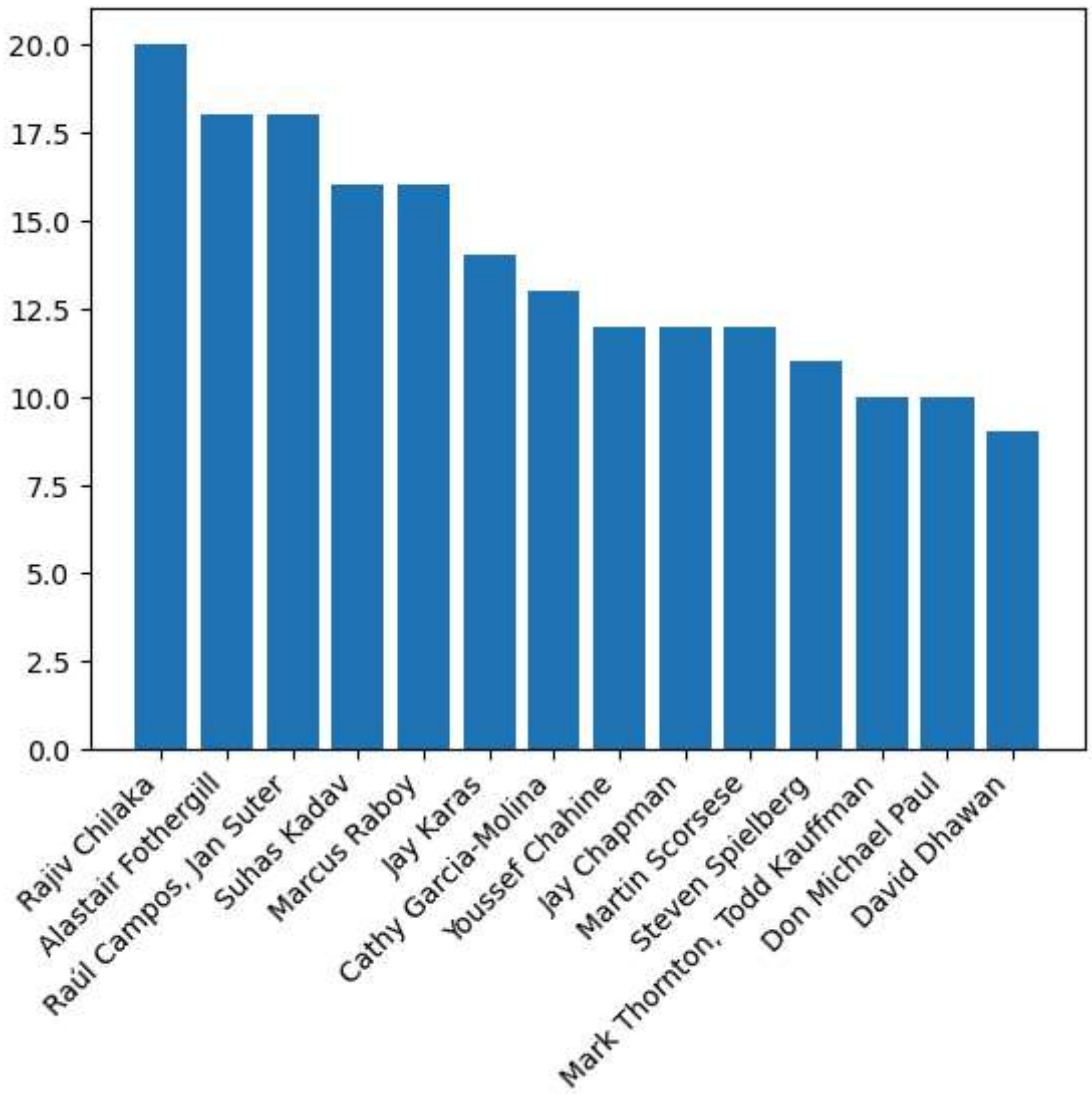


In [106…

```
directors=data['director'].value_counts().reset_index().sort_values(by='count',
plt.bar(directors['director'], directors['count'])
plt.xticks(rotation=45, ha='right')
```

Out[106…

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13],
 [Text(0, 0, 'Rajiv Chilaka'),
  Text(1, 0, 'Alastair Fothergill'),
  Text(2, 0, 'Raúl Campos, Jan Suter'),
  Text(3, 0, 'Suhas Kadav'),
  Text(4, 0, 'Marcus Raboy'),
  Text(5, 0, 'Jay Karas'),
  Text(6, 0, 'Cathy Garcia-Molina'),
  Text(7, 0, 'Youssef Chahine'),
  Text(8, 0, 'Jay Chapman'),
  Text(9, 0, 'Martin Scorsese'),
  Text(10, 0, 'Steven Spielberg'),
  Text(11, 0, 'Mark Thornton, Todd Kauffman'),
  Text(12, 0, 'Don Michael Paul'),
  Text(13, 0, 'David Dhawan')])
```



In [ ]:

In [ ]:

In [ ]: