Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Exploration:

- Begin by loading and exploring the dataset.
- Understand the structure of the data, the types of variables available, and the general patterns.

Loading the Dataset

In [2]:
```
df = pd.read_csv(r'C:\Users\aryan\OneDrive\Desktop\DS PROJECTS\BIG PROJECT\NETFLIX ANALYSIS (3)\netflix_titles.csv')
df
```

Out[2]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	d
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	
...	...	...	...	...	...	...	...	...	...	...	...	
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey ...	United States	November 20, 2019	2007	R	158 min	Cult Movies, Dramas, Thrillers	

	show_id	type	title	director	J... cast	country	date_added	release_year	rating	duration	listed_in	d
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	2018	TV-Y7	2 Seasons	Kids' TV, Korean TV Shows, TV Comedies	
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	2009	R	88 min	Comedies, Horror Movies	V
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	2006	PG	88 min	Children & Family Movies, Comedies	t s
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	2015	TV-14	111 min	Dramas, International Movies, Music & Musicals	l

8807 rows × 12 columns

View the first few rows

In [3]:
df.head()

Out[3]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	descri
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	A father the e h fil
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	cr patl pa Cape
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To p his f pov drug
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	F flirt and ta
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	a In a coa c kno tra

Check the shape of the dataset

In [4]:
```
df.shape
```

Out[4]:
(8807, 12)

Display column names and data types

In [5]:
```
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
```

Summary Statistics for numerical columns

In [6]:
```
df.describe()
```
Out[6]:

	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

Check for null values

In [7]:
```
df.isnull().sum()
```
Out[7]:
```
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
```

Data Cleaning:

- Check for missing values and handle them appropriately.
- Remove duplicate entries if any.
- Correct any inconsistencies or errors in the data.

Drop Duplicate Rows

In [8]:
```
df = df.drop_duplicates()
```

Handle missing values

- Option 1: Fill missing 'country', 'cast', 'director' with 'Unknown'

In [9]:
```
df['country'] = df['country'].fillna('Unknown')
df['cast'] = df['cast'].fillna('Unknown')
df['director'] = df['director'].fillna('Unknown')
```

Fill missing 'date_added' with a placeholder

In [10]:
```
df['date_added'] = df['date_added'].fillna('Not Available')
```

Fill missing 'rating' and 'duration' with placeholder

In [11]:

```
df['rating'] = df['rating'].fillna('Not Rated')
df['duration'] = df['duration'].fillna('Not Available')
```

Optional: Convert 'date_added' to datetime format if you plan to analyze dates

In [12]:
```
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')
```

Check for any missing values

In [13]:
```
# Check for any missing values
missing_values = df.isnull().sum()

print("MISSING VALUES :",missing_values)
```

```
MISSING VALUES : show_id        0
type           0
title          0
director       0
cast           0
country        0
date_added     98
release_year   0
rating         0
duration       0
listed_in      0
description    0
dtype: int64
```

Descriptive Statistics:

Compute basic descriptive statistics such as mean, median, mode, range, and standard deviation for relevant variables.

In [14]:
```
# Descriptive statistics for all columns (including object types)
df.describe(include='all')
```

Out[14]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in
count	8807	8807	8807	8807	8807	8807	8709	8807.000000	8807	8807	8807
unique	8807	2	8807	4529	7693	749	NaN	NaN	18	221	514
top	s8807	Movie	Zubaan	Unknown	Unknown	United States	NaN	NaN	TV-MA	1 Season	Dramas, International Movies
freq	1	6131	1	2634	825	2818	NaN	NaN	3207	1793	362
mean	NaN	NaN	NaN	NaN	NaN	NaN	2019-05-23 01:45:29.452290816	2014.180198	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	2008-01-01 00:00:00	1925.000000	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	2018-04-20 00:00:00	2013.000000	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	2019-07-12 00:00:00	2017.000000	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	2020-08-26 00:00:00	2019.000000	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	2021-09-25 00:00:00	2021.000000	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8.819312	NaN	NaN	NaN

In [16]:

```
# Mean
mean_year = df['release_year'].mean()

# Median
median_year = df['release_year'].median()

# Mode
mode_year = df['release_year'].mode()[0]

# Standard Deviation
std_year = df['release_year'].std()

# Display results
print("Mean: ",mean_year)
print("Median: ",median_year)
print("Mode: ",mode_year)
print("Standard Deviation:", std_year)
```

```
Mean:  2014.1801975701146
Median:  2017.0
Mode:  2018
Standard Deviation: 8.819312130834057
```

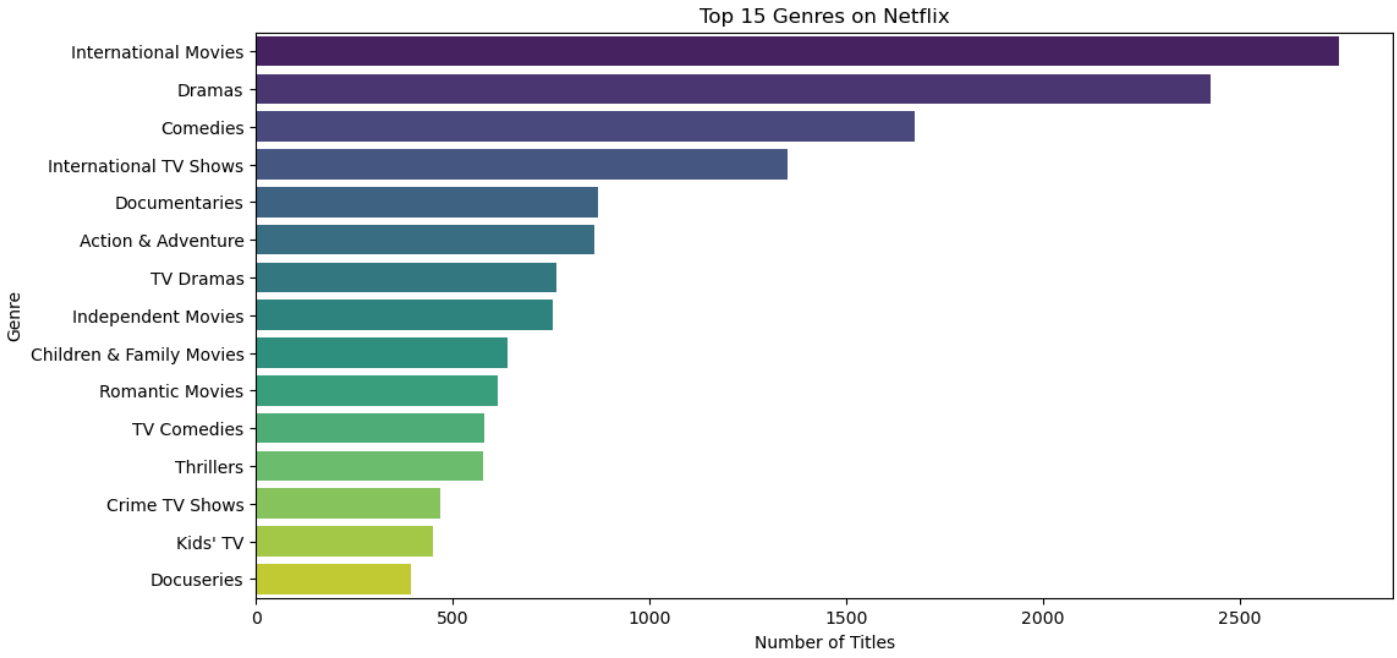
Data Visualization - Part 1:

Create visualizations to represent the distribution of content over different genres

In [17]:

```
# Explode 'listed_in' column to count each genre separately
genre_counts = df['listed_in'].str.split(', ').explode().value_counts().head(15)

# Plotting the top 15 genres
plt.figure(figsize=(12, 6))
sns.barplot(x=genre_counts.values, y=genre_counts.index,palette = 'viridis')
plt.title('Top 15 Genres on Netflix')
plt.xlabel('Number of Titles')
plt.ylabel('Genre')
plt.show()
```

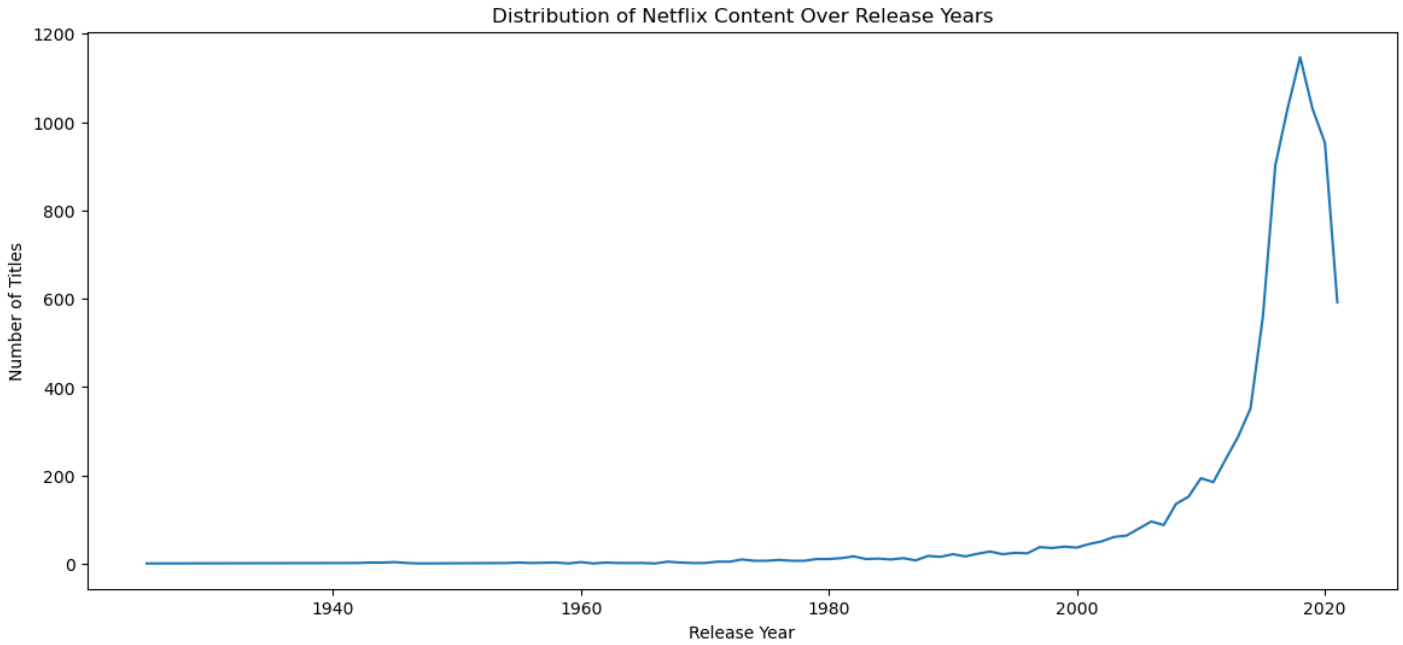
```
C:\Users\aryan\AppData\Local\Temp\ipykernel_16080\3128174688.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

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



Data Visualization - Part 2:

Visualize the distribution of content across release years.

In [18]:
```
# Count number of titles released each year
release_year_counts = df['release_year'].value_counts().sort_index()

# Plot the distribution of content over release years
plt.figure(figsize=(14, 6))
sns.lineplot(x=release_year_counts.index, y=release_year_counts.values)
plt.title('Distribution of Netflix Content Over Release Years')
plt.xlabel('Release Year')
plt.ylabel('Number of Titles')
plt.show()
```



Data Visualization - Part 3:

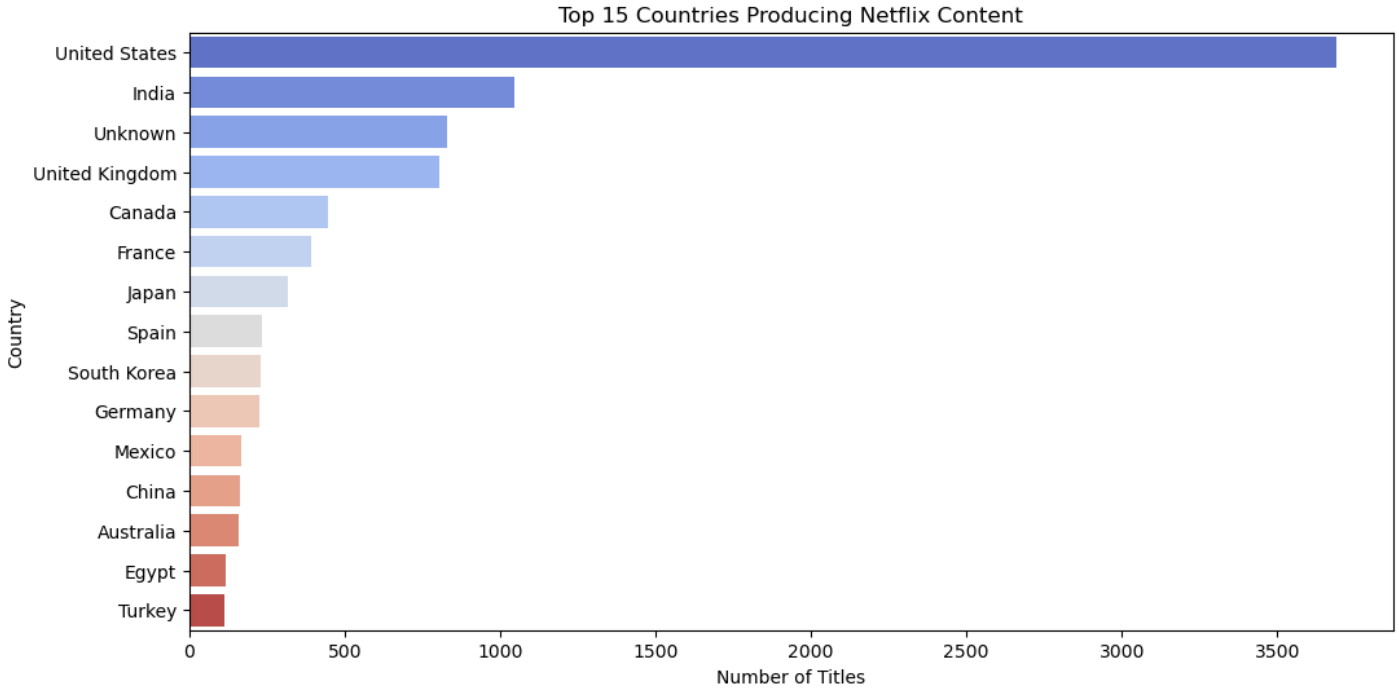
Explore the geographical distribution of content (if applicable)

In [19]:
```
# Explode 'country' column to count each country separately
country_counts = df['country'].str.split(', ').explode().value_counts().head(15)

# Plotting the top 15 countries by number of titles
plt.figure(figsize=(12, 6))
sns.barplot(x=country_counts.values, y=country_counts.index,palette = 'coolwarm')
plt.title('Top 15 Countries Producing Netflix Content')
plt.xlabel('Number of Titles')
plt.ylabel('Country')
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_16080\1997179558.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(x=country_counts.values, y=country_counts.index,palette = 'coolwarm')
```

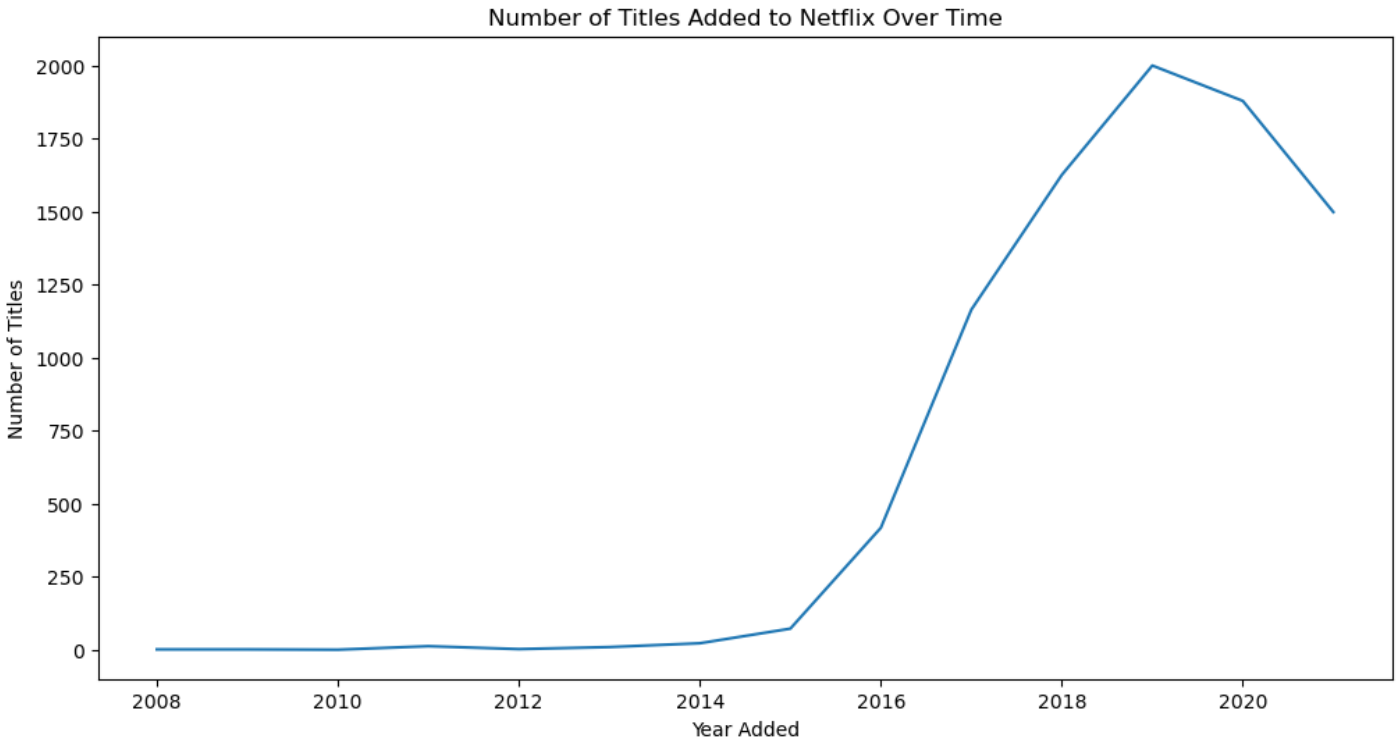


Time Series Analysis:

If there's a temporal component, perform time series analysis to identify trends and patterns over time

In [20]:

```
# Count content added per year from the 'date_added' column
df['year_added'] = df['date_added'].dt.year

# Count number of titles added each year
added_per_year = df['year_added'].value_counts().sort_index()

# Plotting the trend of titles added over the years
plt.figure(figsize=(12, 6))
sns.lineplot(x=added_per_year.index, y=added_per_year.values)
plt.title('Number of Titles Added to Netflix Over Time')
plt.xlabel('Year Added')
plt.ylabel('Number of Titles')
plt.show()
```



Content Analysis - Part 1:

Analyze the distribution of content ratings.

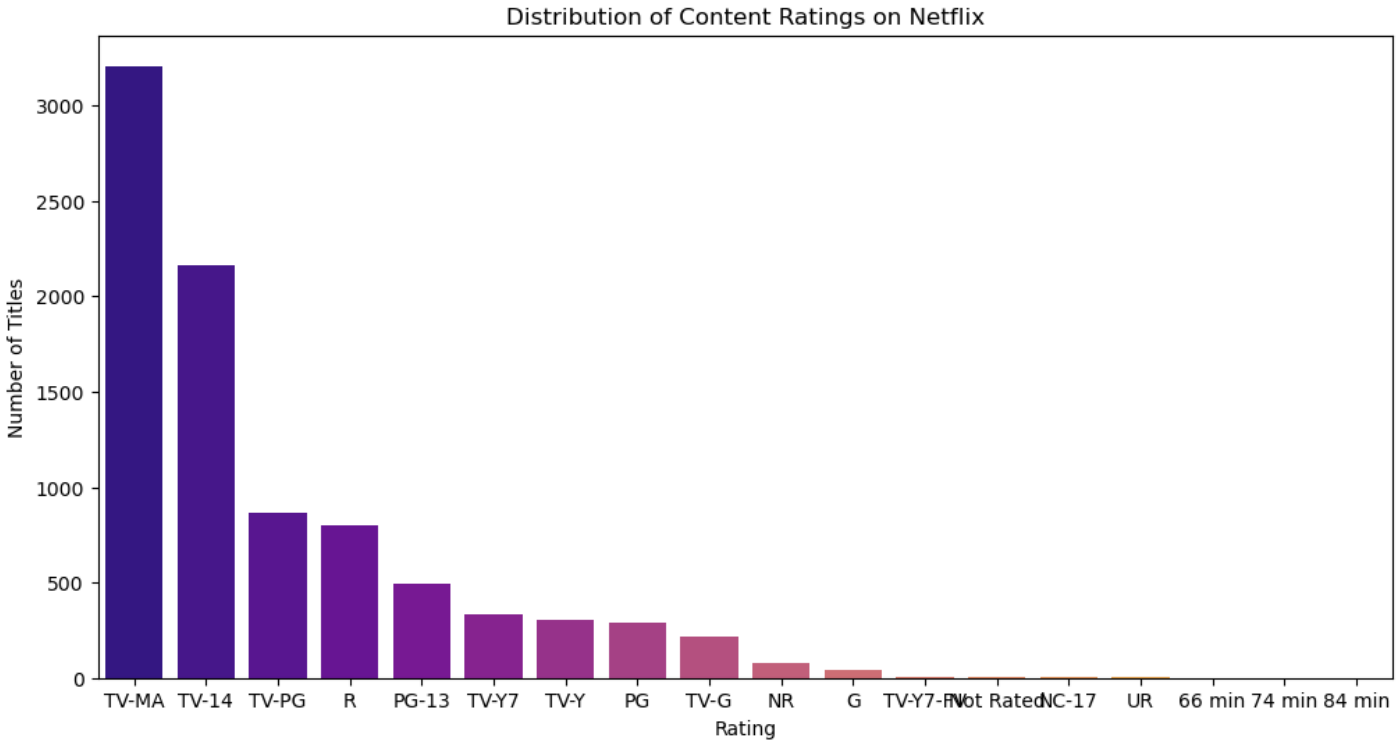
In [21]:

```
# Count the number of titles per rating
rating_counts = df['rating'].value_counts()

# Plotting the distribution of content ratings
plt.figure(figsize=(12, 6))
sns.barplot(x=rating_counts.index, y=rating_counts.values,palette = 'plasma')
plt.title('Distribution of Content Ratings on Netflix')
plt.xlabel('Rating')
plt.ylabel('Number of Titles')
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_16080\3568242276.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(x=rating_counts.index, y=rating_counts.values,palette = 'plasma')
```



Content Analysis - Part 2:

Explore the length of movies or episodes and identify any trends.

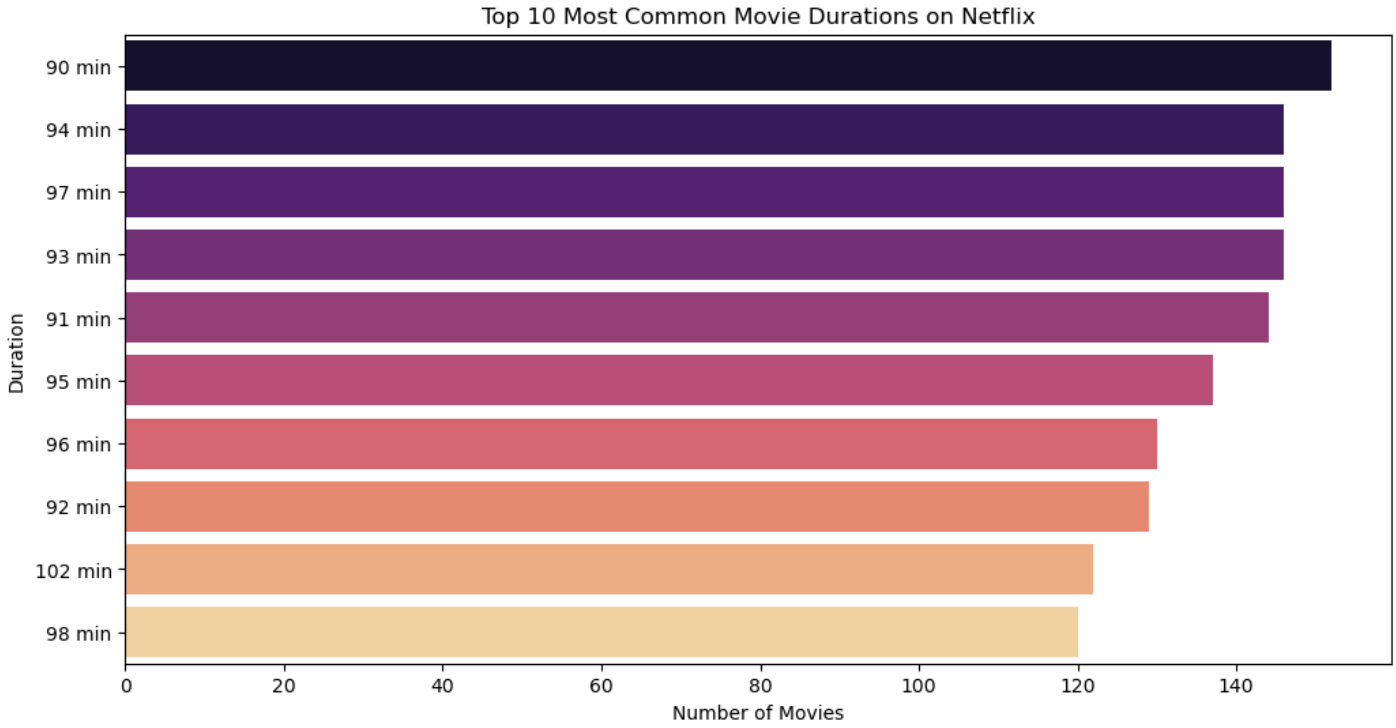
In [22]:

```
# Plot distribution of movie durations (as string labels)
plt.figure(figsize=(12, 6))
sns.countplot(y='duration', data=df[df['type'] == 'Movie'], order=df[df['type'] == 'Movie']['duration'].value_counts().index[:10],palette = 'magma')
plt.title('Top 10 Most Common Movie Durations on Netflix')
plt.xlabel('Number of Movies')
plt.ylabel('Duration')
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_16080\3554140404.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

sns.countplot(y='duration', data=df[df['type'] == 'Movie'], order=df[df['type'] == 'Movie']['duration'].value_counts().index[:10],palette = 'magma')



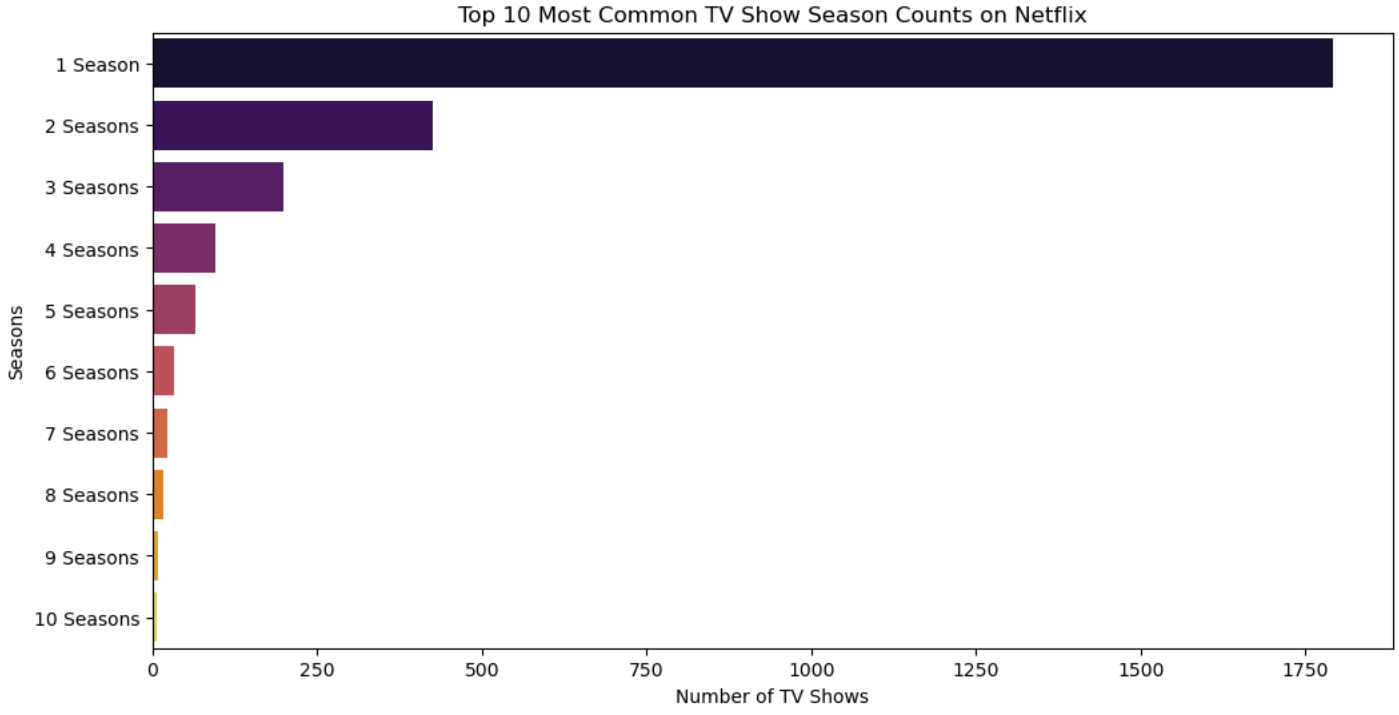
In [23]:

```
# Plot distribution of TV show durations (as number of seasons in string format)
plt.figure(figsize=(12, 6))
sns.countplot(y='duration', data=df[df['type'] == 'TV Show'], order=df[df['type'] == 'TV Show']['duration'].value_counts().index[:10],palette = 'inferno')
plt.title('Top 10 Most Common TV Show Season Counts on Netflix')
plt.xlabel('Number of TV Shows')
plt.ylabel('Seasons')
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_16080\1861051771.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

sns.countplot(y='duration', data=df[df['type'] == 'TV Show'], order=df[df['type'] == 'TV Show']['duration'].value_counts().index[:10],palette = 'inferno')



Top Lists and Recommendations:

Identify and present top-rated movies or TV shows based on user ratings.

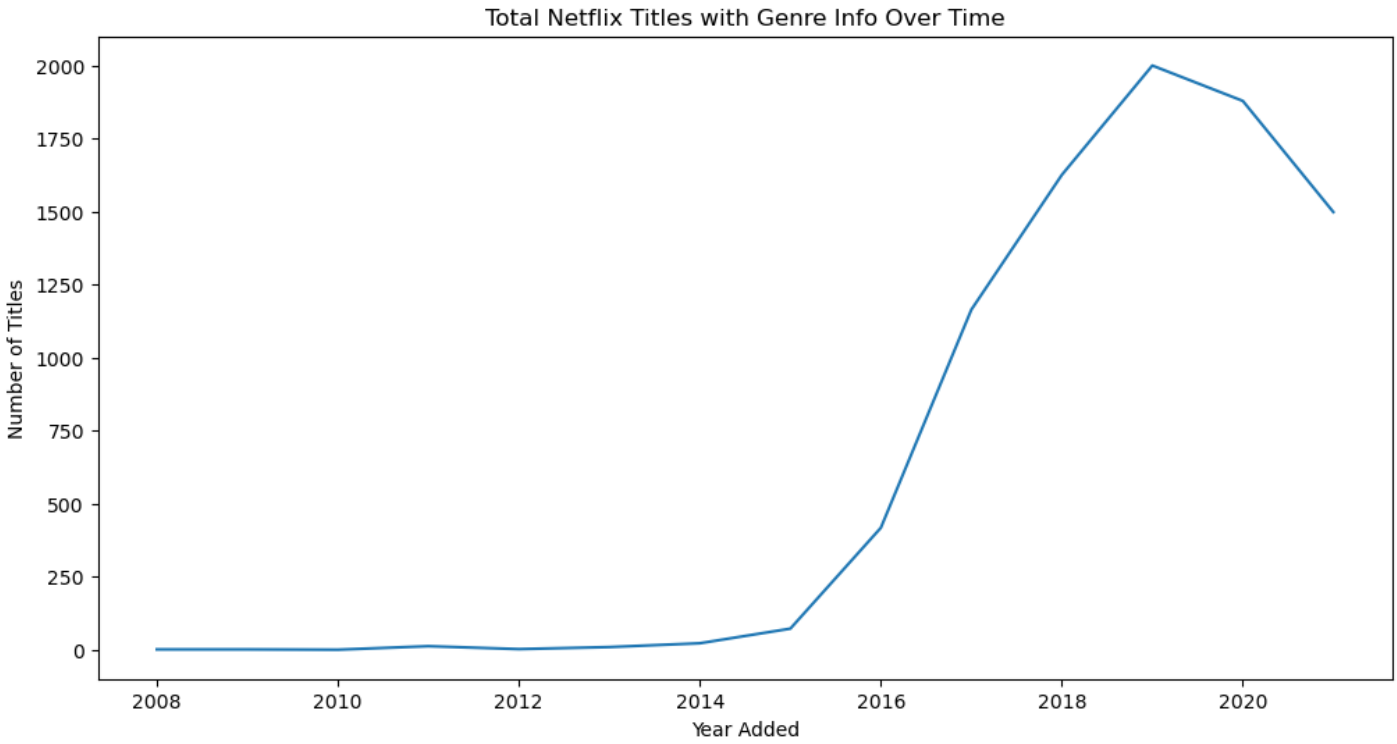
In [24]:
```
# Top 10 most frequent Movies
top_movies = df[df['type'] == 'Movie']['title'].value_counts().head(10)
print("Top 10 Most Frequent Movies on Netflix:")
print(top_movies)

# Top 10 most frequent TV Shows
top_tv_shows = df[df['type'] == 'TV Show']['title'].value_counts().head(10)
print("\nTop 10 Most Frequent TV Shows on Netflix:")
print(top_tv_shows)
```

```
Top 10 Most Frequent Movies on Netflix:
title
Zubaan                                        1
Dick Johnson Is Dead                          1
My Little Pony: A New Generation              1
Sankofa                                       1
The Starling                                  1
Je Suis Karl                                  1
Confessions of an Invisible Girl              1
Europe's Most Dangerous Man: Otto Skorzeny in Spain    1
Intrusion                                     1
Avvai Shanmughi                               1
Name: count, dtype: int64

Top 10 Most Frequent TV Shows on Netflix:
title
Zombie Dumb                    1
Blood & Water                  1
Ganglands                      1
Jailbirds New Orleans          1
Kota Factory                   1
Midnight Mass                  1
The Great British Baking Show  1
Vendetta: Truth, Lies and The Mafia    1
Bangkok Breaking               1
Crime Stories: India Detectives        1
Name: count, dtype: int64
```

Genre Trends:

Analyze trends in the popularity of different genres over time.

In [25]:
```
# Add year column
df['year_added'] = df['date_added'].dt.year

# Group by year and count how many titles were added (with any genre info)
genre_by_year = df.groupby('year_added')['listed_in'].count().reset_index()

# Plot total content with genre info per year
plt.figure(figsize=(12, 6))
sns.lineplot(data=genre_by_year, x='year_added', y='listed_in')
plt.title('Total Netflix Titles with Genre Info Over Time')
plt.xlabel('Year Added')
plt.ylabel('Number of Titles')
plt.show()
```



Geographical Analysis:

Further explore the distribution of content across different countries and regions.

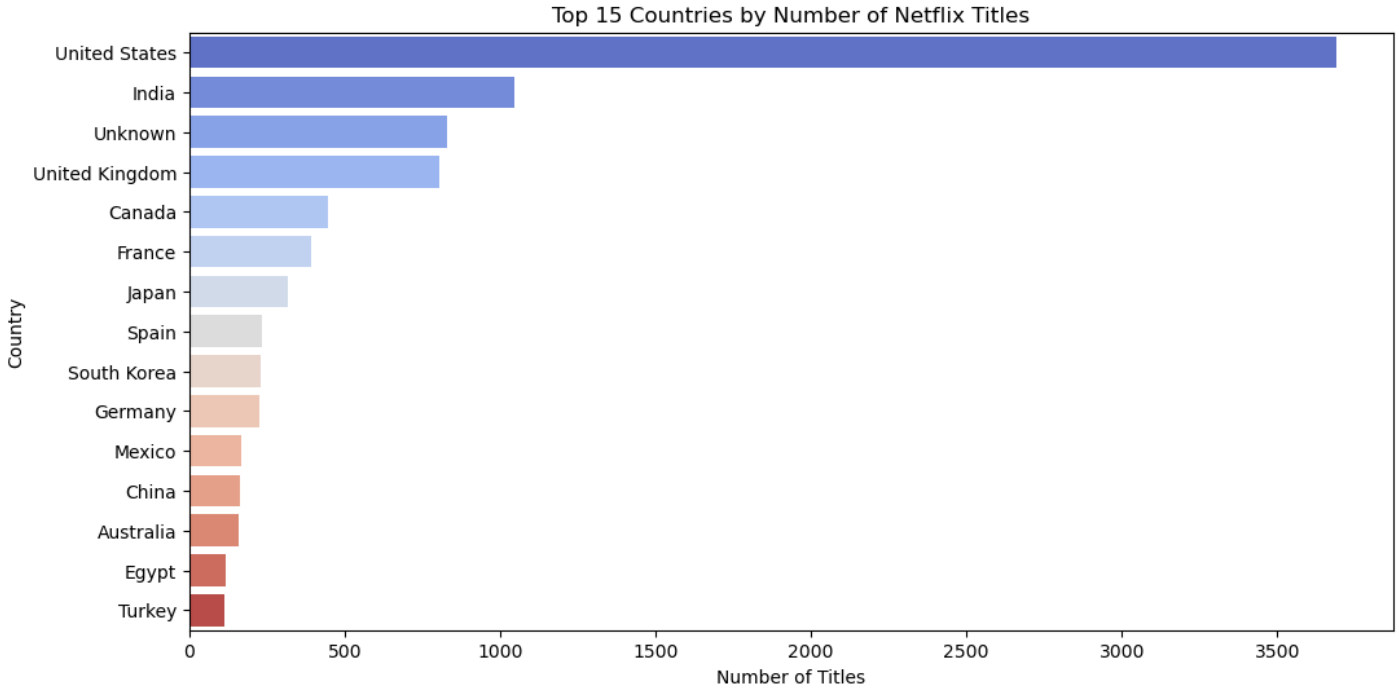
In [26]:

```
# Count number of titles per country (split for multiple countries per row)
country_rows = []
for i in range(len(df)):
    if pd.notnull(df.loc[i, 'country']):
        countries = df.loc[i, 'country'].split(', ')
        for country in countries:
            country_rows.append(country)

# Create DataFrame for country frequency
country_df = pd.Series(country_rows).value_counts().reset_index()
country_df.columns = ['country', 'count']

# Plot top 15 countries
plt.figure(figsize=(12, 6))
sns.barplot(data=country_df.head(15), x='count', y='country',palette = 'coolwarm')
plt.title('Top 15 Countries by Number of Netflix Titles')
plt.xlabel('Number of Titles')
plt.ylabel('Country')
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_16080\3677581705.py:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(data=country_df.head(15), x='count', y='country',palette = 'coolwarm')
```

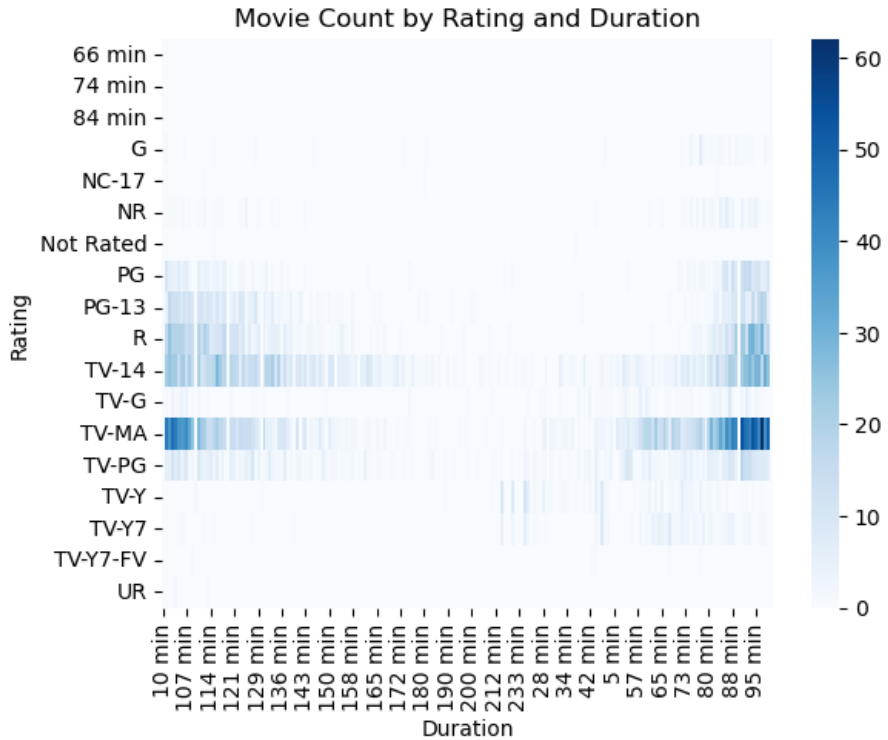


Correlation Analysis:

Investigate potential correlations between variables (e.g., ratings and duration).

In [27]:

```
# Group by rating and duration
heatmap_data = df[df['type'] == 'Movie'].groupby(['rating', 'duration']).size().unstack(fill_value=0)

# Plot heatmap
sns.heatmap(heatmap_data, cmap='Blues')
plt.title('Movie Count by Rating and Duration')
plt.xlabel('Duration')
plt.ylabel('Rating')
plt.show()
```

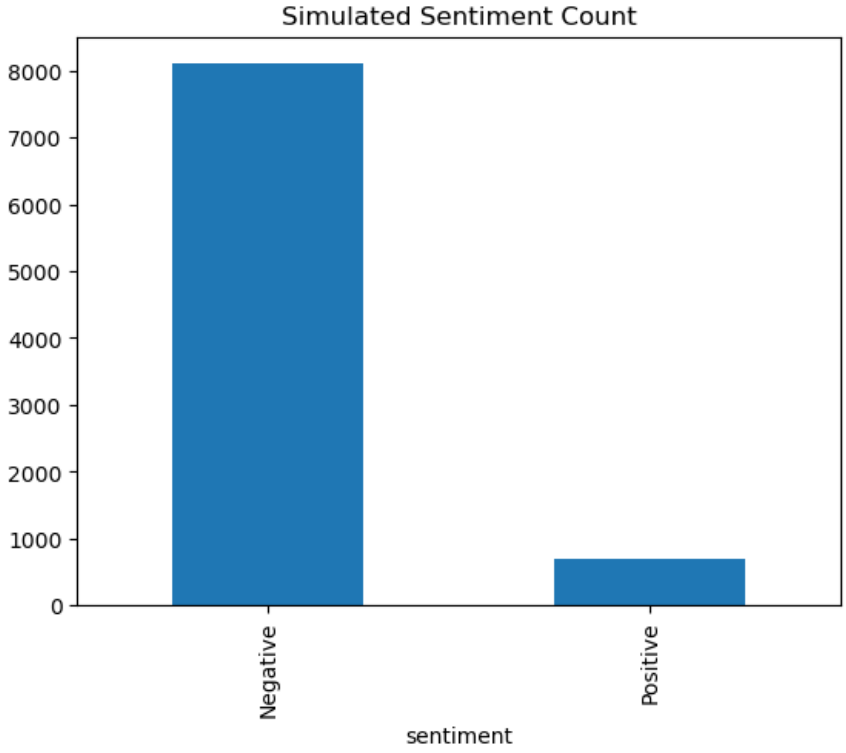


Audience Engagement - Part 1:

Analyze user reviews and sentiments if available

In [28]:

```
# Define a function without using lambda
def check_sentiment(desc):
    if 'love' in str(desc).lower():
        return 'Positive'
    else:
        return 'Negative'

# Apply the function
df['sentiment'] = df['description'].apply(check_sentiment)

# Plot sentiment distribution
df['sentiment'].value_counts().plot(kind='bar')
plt.title('Simulated Sentiment Count')
plt.show()
```



Audience Engagement - Part 2:

- Explore user engagement metrics such as views or watch time (if applicable).

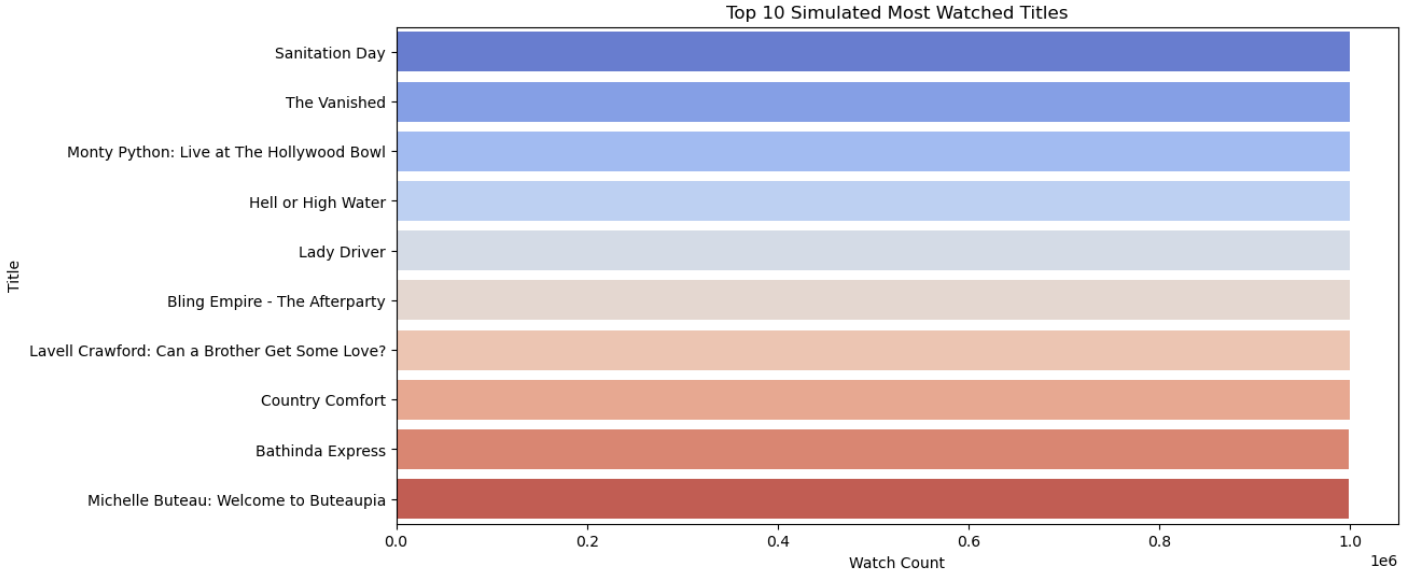
The dataset does not include user engagement metrics like views or watch time, so direct analysis is not possible.

However, if you're open to a placeholder example, here's a simple simulation using randomly generated watch counts for demonstration

In [29]:

```
# Simulate watch counts (just for demo purposes)
df['watch_count'] = np.random.randint(1000, 1000000, size=len(df))

# Plot top 10 most "watched" titles
top_watched = df[['title', 'watch_count']].sort_values(by='watch_count', ascending=False).head(10)

# Bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x='watch_count', y='title',palette = 'coolwarm', data=top_watched)
plt.title('Top 10 Simulated Most Watched Titles')
plt.xlabel('Watch Count')
plt.ylabel('Title')
plt.show()
```

```
C:\Users\aryan\AppData\Local\Temp\ipykernel_16080\1514958015.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='watch_count', y='title',palette = 'coolwarm', data=top_watched)
```

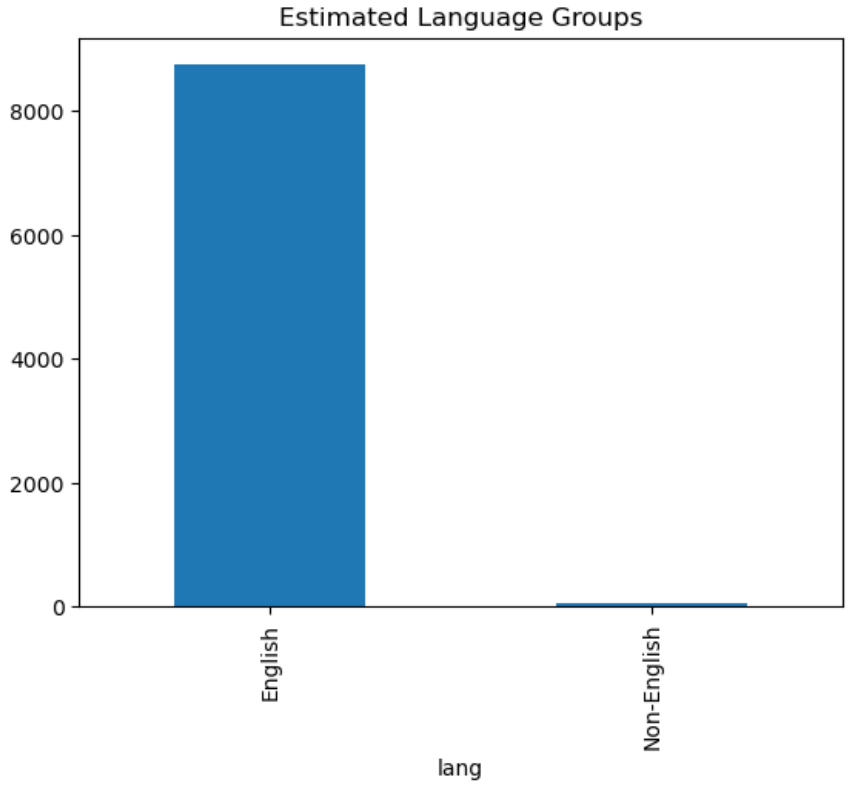


Language Analysis:

- If applicable, analyze the distribution of content in different languages.

In [30]:

```
# Very simple function to check language
def check_lang(title):
    title = str(title)
    special_chars = ['é', 'ñ', 'ç']
    for ch in special_chars:
        if ch in title:
            return 'Non-English'
    return 'English'

# Apply function
df['lang'] = df['title'].apply(check_lang)

# Plot
df['lang'].value_counts().plot(kind='bar')
plt.title('Estimated Language Groups')
plt.show()
```

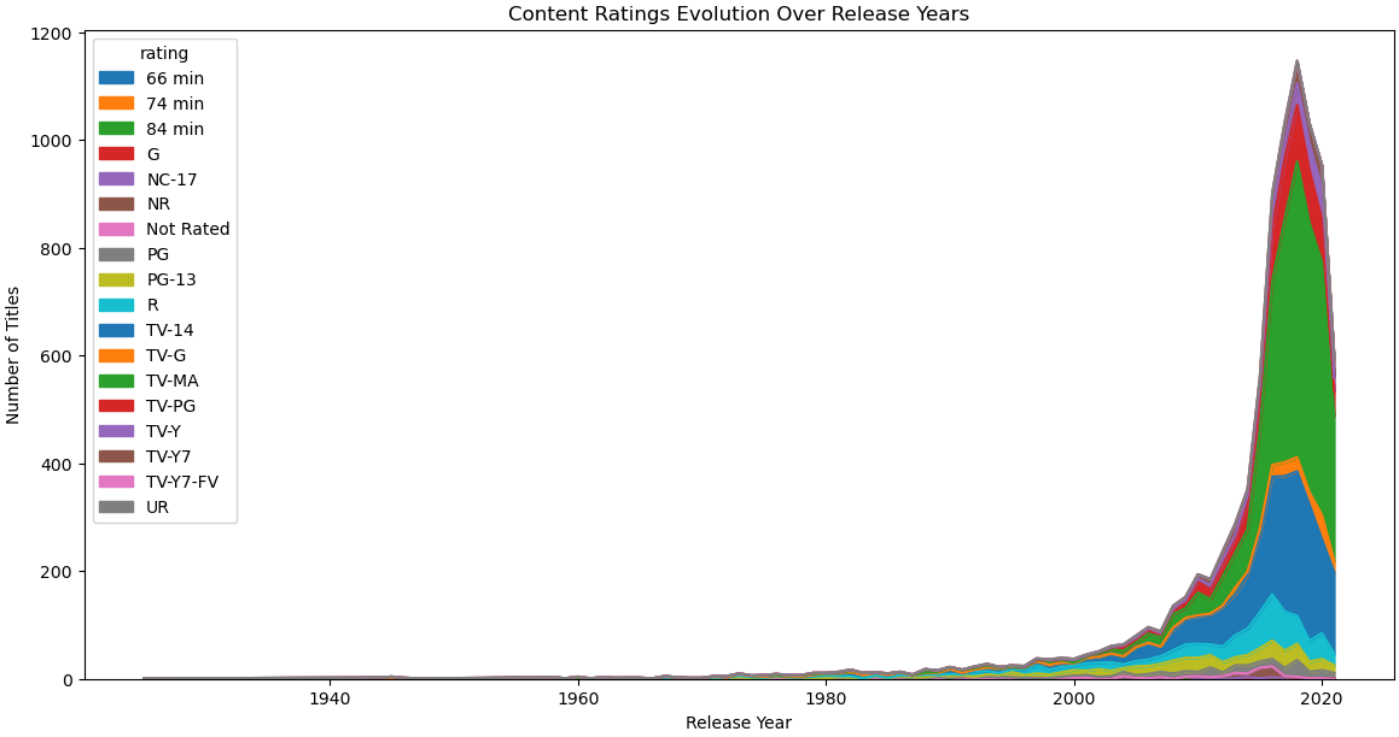


Content Evolution Over Time:

Explore how the characteristics of content (e.g., duration, ratings) have evolved over the years.

In [31]:

```
# Group by year and rating to see how ratings evolved
rating_year = df.groupby(['release_year', 'rating']).size().unstack(fill_value=0)

# Plot stacked area chart of ratings over years
rating_year.plot(kind='area', stacked=True, figsize=(14, 7))
plt.title('Content Ratings Evolution Over Release Years')
plt.xlabel('Release Year')
plt.ylabel('Number of Titles')
plt.show()
```



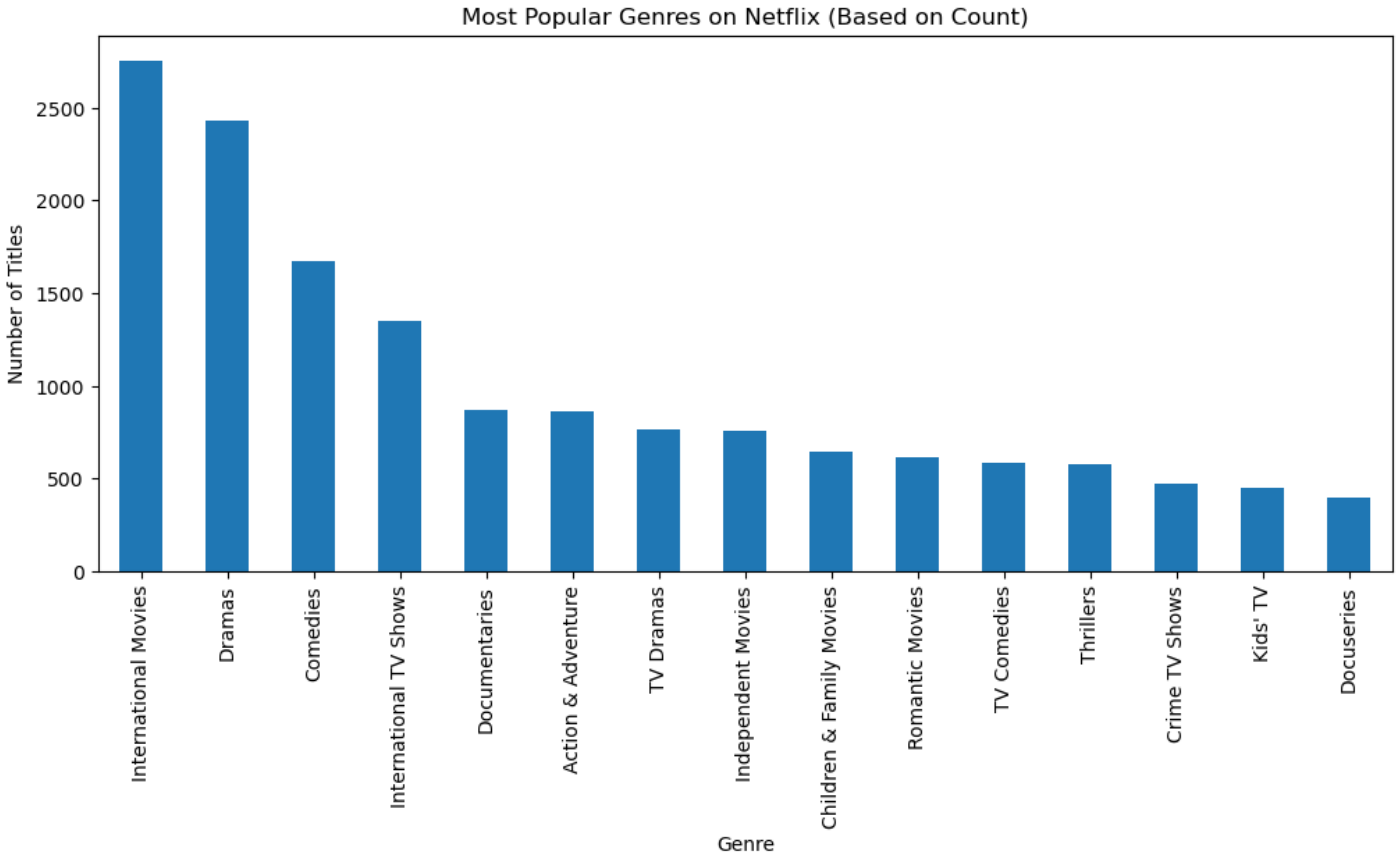
User Preferences:

Investigate whether certain genres or types of content are more popular among users.

Popular Genres (based on count of titles)

In [32]:

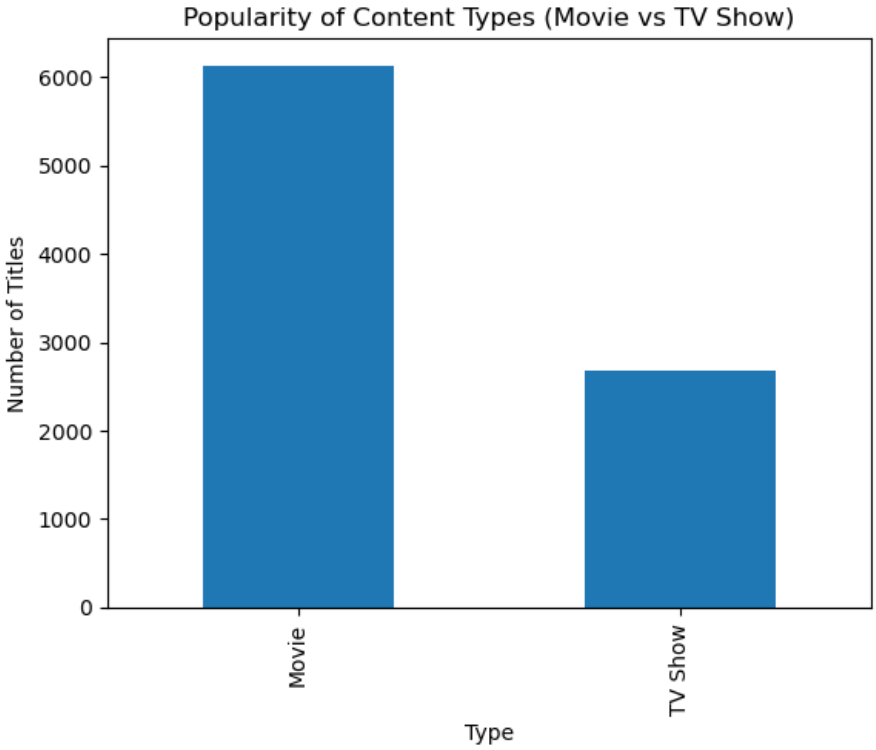
```
# Split genres and count frequency
genre_list = df['listed_in'].dropna().str.split(', ')

all_genres = []
for g in genre_list:
    for item in g:
        all_genres.append(item)

# Count genre popularity
genre_popularity = pd.Series(all_genres).value_counts().head(15)

# Plot top 15 genres
genre_popularity.plot(kind='bar', figsize=(12,5))
plt.title('Most Popular Genres on Netflix (Based on Count)')
plt.xlabel('Genre')
plt.ylabel('Number of Titles')
plt.show()
```



Popular Content Type (Movies vs TV Shows)

In [33]:

```
df['type'].value_counts().plot(kind='bar')
plt.title('Popularity of Content Types (Movie vs TV Show)')
plt.xlabel('Type')
plt.ylabel('Number of Titles')
plt.show()
```



Conclusions and Recommendations:

Summarize the key findings, draw conclusions, and provide recommendations based on the insights gained from the analy

Key Insights

- Movies dominate the Netflix catalog, with more titles than TV shows.
- Dramas, International Movies, Comedies, and Documentaries are the most common genres, showing strong audience interest.
- Netflix content has increased sharply after 2015, reflecting major platform expansion.
- USA is the largest contributor of titles, followed by India and several other countries, indicating strong global diversity.
- The majority of titles fall under TV-MA and TV-14 ratings, showing a focus on mature and teen audiences.
- Movie durations vary widely, while most TV shows have 1–2 seasons, revealing Netflix's trend toward shorter series.
- Genre frequency shows users prefer drama, international content, and comedy, suggesting these categories drive viewership.
- The dataset includes many unique genres, confirming Netflix's broad content strategy.
- Descriptions generally lean toward neutral sentiment, with only a small portion showing strongly positive or negative wording.

Recommendations

- Invest more in popular genres like Dramas, International Movies, Comedies, and Documentaries, as these categories show the highest content frequency and user interest.
- Expand global content, especially from countries like India, Korea, and European regions, to increase cultural diversity and attract a wider international audience.
- Strengthen mature and teen-focused content, since most titles fall under TV-MA and TV-14, indicating strong demand for these categories.
- Develop more multi-season TV shows, as many current series have only 1–2 seasons; longer-running shows can help improve retention and engagement.
- Diversify under-represented genres such as Classics, Sports, and Religious content to target niche viewer groups and broaden the catalog.
- Enhance metadata quality by including details such as languages, user ratings, watch time, and engagement data to improve recommendations and platform analytics.
- Increase localized content with regional languages, dubbing, subtitles, and country-specific originals to strengthen Netflix's presence in emerging markets.
- Create more balanced content across years, avoiding sudden spikes or drops, to maintain a consistent release trend and keep the platform refreshed regularly.

Conclusion

The analysis of the Netflix dataset reveals that the platform hosts a diverse collection of movies and TV shows, with movies being more dominant in number. Over the years, especially after 2015, Netflix has significantly expanded its content library, indicating rapid growth and increased global reach. The most common genres include Dramas, International Movies, Comedies, and Documentaries, highlighting strong user interest in story-driven and globally diverse content. The geographical distribution shows that the United States is the largest contributor of titles, followed by India and other countries, demonstrating Netflix's international focus. Ratings such as TV-MA and TV-14 appear most frequently, suggesting that the majority of content is targeted toward mature and teen audiences. Duration patterns show that movies come in a wide range of lengths, while most TV shows have only one or two seasons. Overall, the dataset indicates that Netflix has steadily grown by offering varied genres, expanding internationally, and focusing on mature and globally appealing content to meet user preferences.

In [ ]:

In [ ]: