In [1]:

```
import os
from pathlib import Path
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from wordcloud import WordCloud

%matplotlib inline
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[1], line 6
      4 import numpy as np
      5 import matplotlib.pyplot as plt
----> 6 from wordcloud import WordCloud
      8 get_ipython().run_line_magic('matplotlib', 'inline')

ModuleNotFoundError: No module named 'wordcloud'
```

In [2]:

```
!pip install wordcloud
```

```
Access is denied.
```

In [4]:

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

sns.set(style="whitegrid")
%matplotlib inline
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[4], line 5
      3 import matplotlib.pyplot as plt
      4 import seaborn as sns
----> 5 from wordcloud import WordCloud
      7 sns.set(style="whitegrid")
      8 get_ipython().run_line_magic('matplotlib', 'inline')

ModuleNotFoundError: No module named 'wordcloud'
```

In [5]:

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

sns.set(style="whitegrid")
%matplotlib inline
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[5], line 5
      3 import matplotlib.pyplot as plt
      4 import seaborn as sns
----> 5 from wordcloud import WordCloud
      7 sns.set(style="whitegrid")
      8 get_ipython().run_line_magic('matplotlib', 'inline')

ModuleNotFoundError: No module named 'wordcloud'
```

In [6]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
import re

sns.set(style="whitegrid")
%matplotlib inline
```

In [7]:
```python
data = pd.read_csv("netflix1.csv")

data.head()
```

Out[7]:

	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 [8]:
```python
data.info()

print("Number of duplicate rows:", data.duplicated().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
Number of duplicate rows: 0
```

In [11]:
```
data = data.drop_duplicates()

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

required_columns = ['director', 'country']
existing_columns = [col for col in required_columns if col in data.columns]
data.dropna(subset=existing_columns, inplace=True)

if 'rating' in data.columns:
    data['rating'].fillna('Not Rated', inplace=True)

data.info()
```

```
<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   datetime64[ns]
 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: datetime64[ns](1), int64(1), object(8)
memory usage: 686.8+ KB
```

```
C:\Users\krish\AppData\Local\Temp\ipykernel_16000\1361052712.py:10: FutureWarnin
g: 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['rating'].fillna('Not Rated', inplace=True)
```

In [12]:
```
if 'rating' in data.columns:
    data['rating'] = data['rating'].fillna('Not Rated')
```

In [13]:
```
data = data.drop_duplicates()

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

required_columns = ['director', 'country']
existing_columns = [col for col in required_columns if col in data.columns]
data.dropna(subset=existing_columns, inplace=True)

if 'rating' in data.columns:
    data['rating'].fillna('Not Rated', inplace=True)

data.info()
```

```
<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   datetime64[ns]
 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: datetime64[ns](1), int64(1), object(8)
memory usage: 686.8+ KB
```

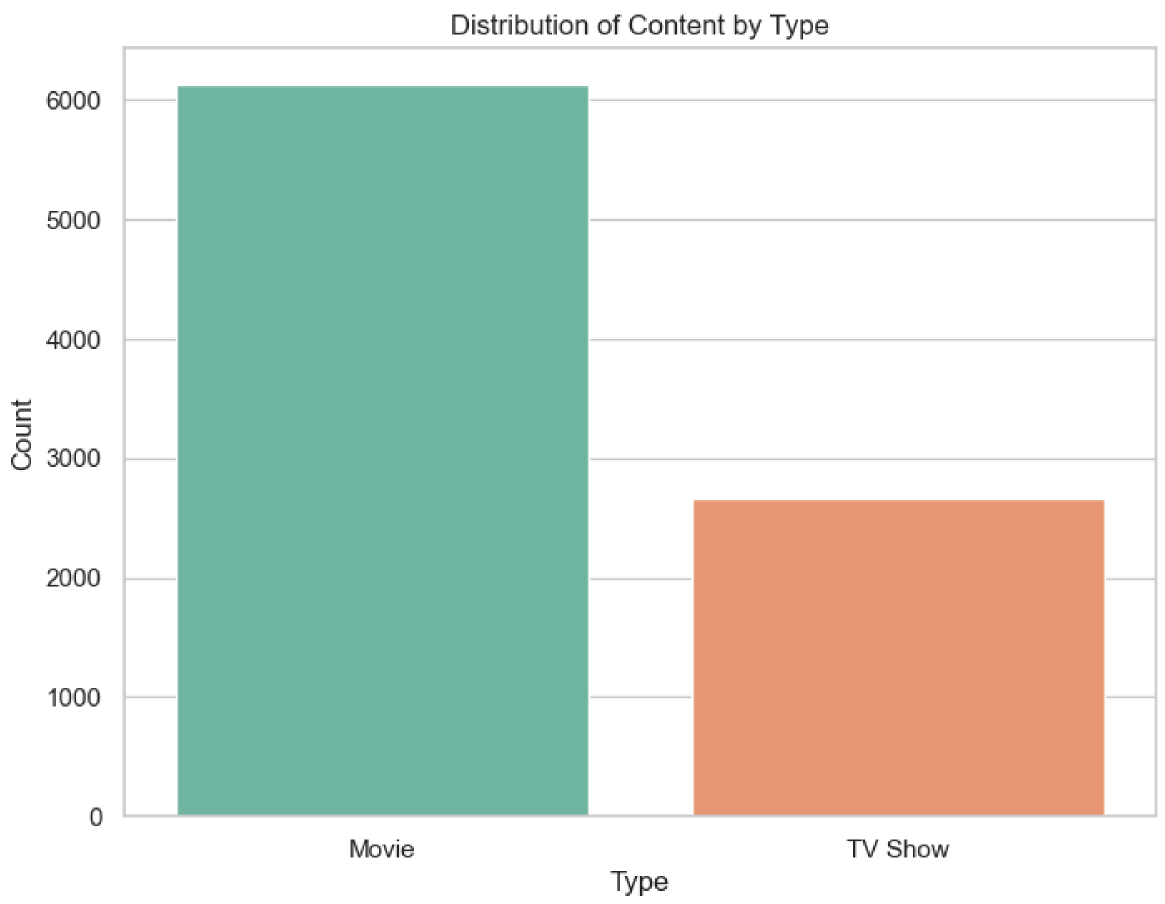
```
C:\Users\krish\AppData\Local\Temp\ipykernel_16000\3197550.py:10: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assi
gnment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate 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['rating'].fillna('Not Rated', inplace=True)
```

In [14]:
```
data = data.drop_duplicates()

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

required_columns = ['director', 'country']
existing_columns = [col for col in required_columns if col in data.columns]
data.dropna(subset=existing_columns, inplace=True)

if 'rating' in data.columns:
    data['rating'] = data['rating'].fillna('Not Rated')

data.info()
```

```
<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   datetime64[ns]
 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: datetime64[ns](1), int64(1), object(8)
memory usage: 686.8+ KB
```

In [17]:
```
plt.figure(figsize=(8,6))
sns.countplot(x='type', data=data, hue=None, palette='Set2')
plt.title("Distribution of Content by Type")
plt.xlabel("Type")
plt.ylabel("Count")
plt.show()
```

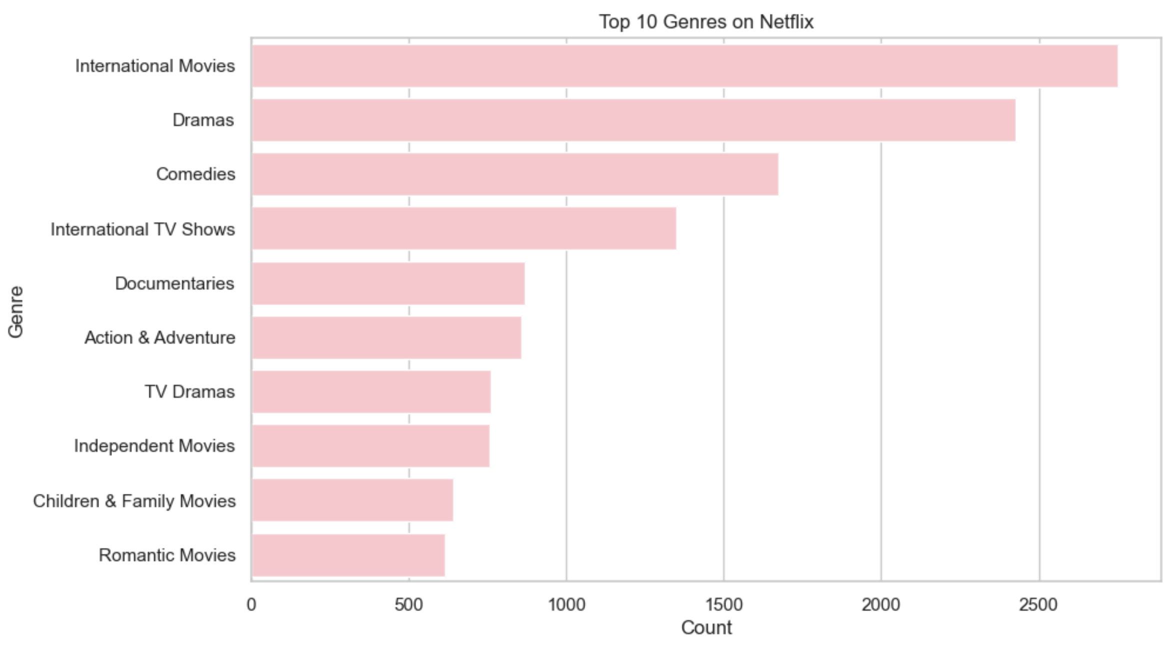
```
C:\Users\krish\AppData\Local\Temp\ipykernel_16000\1060851790.py:2: FutureWarning:

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, hue=None, palette='Set2')
```



In [23]:
```
plt.figure(figsize=(10,6))
sns.barplot(x=genre_counts.values, y=genre_counts.index, color='skyblue')  # sin
plt.title("Top 10 Genres on Netflix")
plt.xlabel("Count")
plt.ylabel("Genre")
plt.show()
```

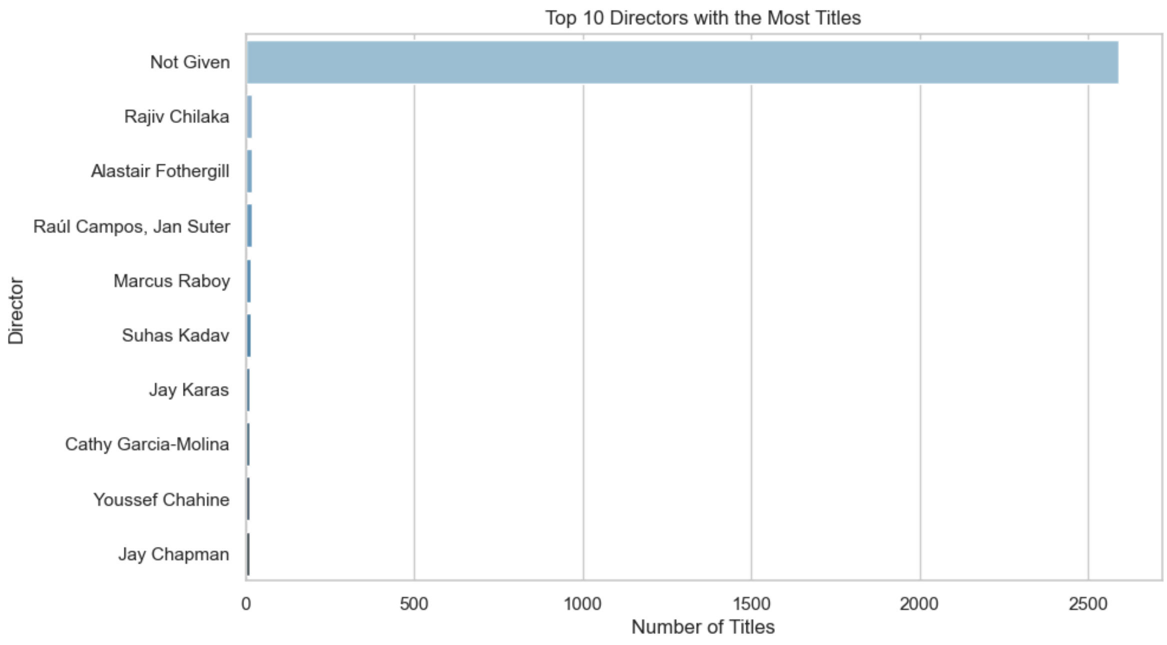
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[23], line 2
      1 plt.figure(figsize=(10,6))
----> 2 sns.barplot(x=genre_counts.values, y=genre_counts.index, color='skyblue')
# single color
      3 plt.title("Top 10 Genres on Netflix")
      4 plt.xlabel("Count")

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

In [25]:
```
data['genres'] = data['listed_in'].apply(lambda x: x.split(", "))
all_genres = sum(data['genres'], [])
genre_counts = pd.Series(all_genres).value_counts().head(10)

plt.figure(figsize=(10,6))
sns.barplot(x=genre_counts.values, y=genre_counts.index, color='pink')
plt.title("Top 10 Genres on Netflix")
plt.xlabel("Count")
plt.ylabel("Genre")
plt.show()
```



```
In [26]: top_directors = data['director'].value_counts().head(10)

         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\krish\AppData\Local\Temp\ipykernel_16000\3845443546.py:4: FutureWarning:

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')
```



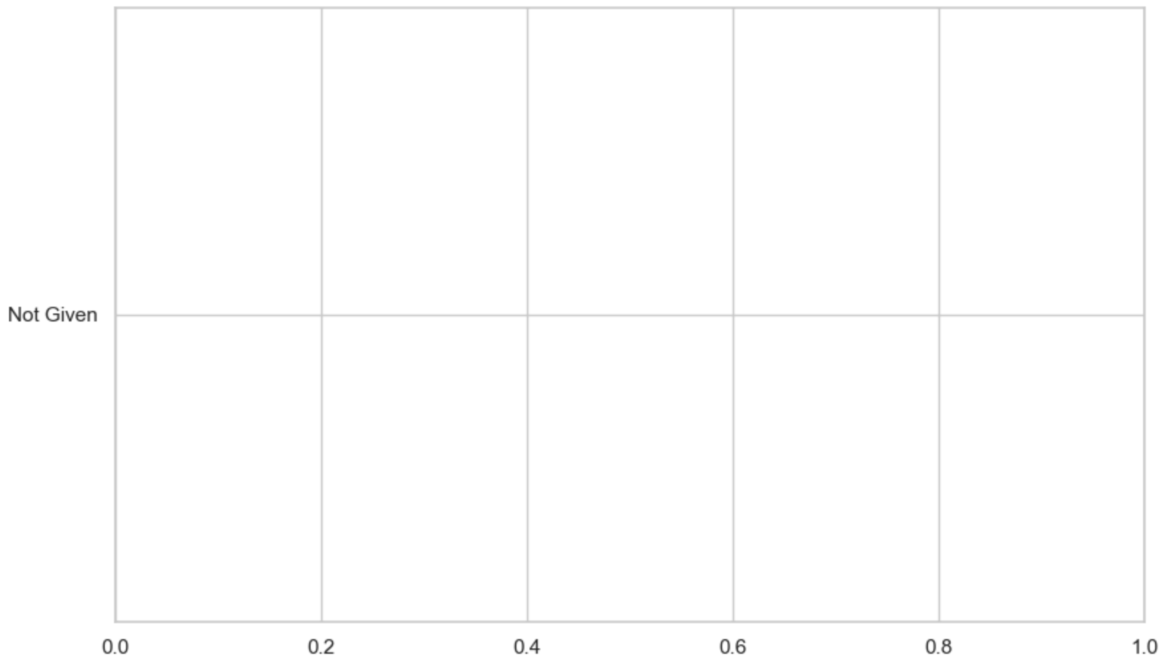
```
In [27]:
```

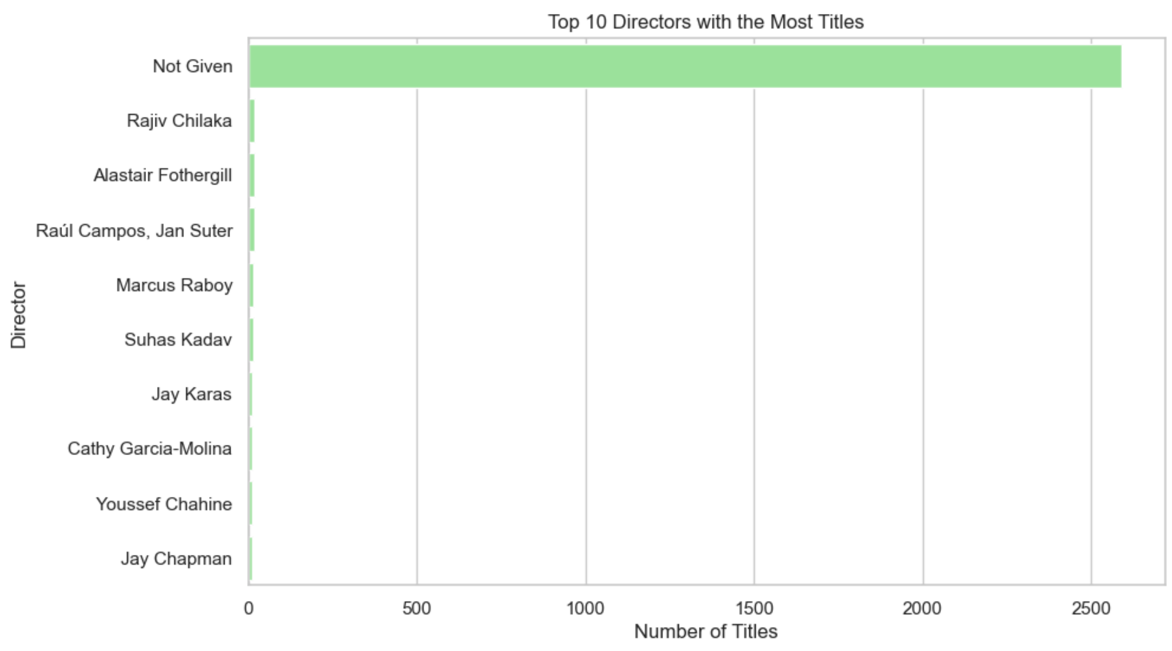
```
C:\Users\krish\AppData\Local\Temp\ipykernel_16000\2517931203.py:4: FutureWarning:

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='lightgree
n')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\palettes.py:235, in color
_palette(palette, n_colors, desat, as_cmap)
    233 try:
    234     # Perhaps a named matplotlib colormap?
--> 235     palette = mpl_palette(palette, n_colors, as_cmap=as_cmap)
    236 except (ValueError, KeyError):  # Error class changed in mpl36

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\palettes.py:406, in mpl_p
alette(name, n_colors, as_cmap)
    405 else:
--> 406     cmap = get_colormap(name)
    408 if name in MPL_QUAL_PALS:

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_compat.py:62, in get_col
ormap(name)
     61 try:
---> 62     return mpl.colormaps[name]
     63 except AttributeError:

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\cm.py:98, in ColormapR
egistry.__getitem__(self, item)
     97 except KeyError:
---> 98     raise KeyError(f"{item!r} is not a known colormap name") from None

KeyError: "'lightgreen' is not a known colormap name"

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
Cell In[27], line 4
      1 top_directors = data['director'].value_counts().head(10)
      3 plt.figure(figsize=(10,6))
----> 4 sns.barplot(x=top_directors.values, y=top_directors.index, palette='light
green')
      5 plt.title("Top 10 Directors with the Most Titles")
      6 plt.xlabel("Number of Titles")

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\categorical.py:2370, in b
arplot(data, x, y, hue, order, hue_order, estimator, errorbar, n_boot, seed, unit
s, weights, orient, color, palette, saturation, fill, hue_norm, width, dodge, ga
p, log_scale, native_scale, formatter, legend, capsize, err_kws, ci, errcolor, er
rwidth, ax, **kwargs)
   2367 palette, hue_order = p._hue_backcompat(color, palette, hue_order)
   2369 saturation = saturation if fill else 1
-> 2370 p.map_hue(palette=palette, order=hue_order, norm=hue_norm, saturation=sat
uration)
   2371 color = _default_color(ax.bar, hue, color, kwargs, saturation=saturation)
   2373 agg_cls = WeightedAggregator if "weight" in p.plot_data else EstimateAggr
egator

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_base.py:838, in VectorPl
otter.map_hue(self, palette, order, norm, saturation)
    837 def map_hue(self, palette=None, order=None, norm=None, saturation=1):
--> 838     mapping = HueMapping(self, palette, order, norm, saturation)
    839     self._hue_map = mapping

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_base.py:150, in HueMappi
ng.__init__(self, plotter, palette, order, norm, saturation)
```

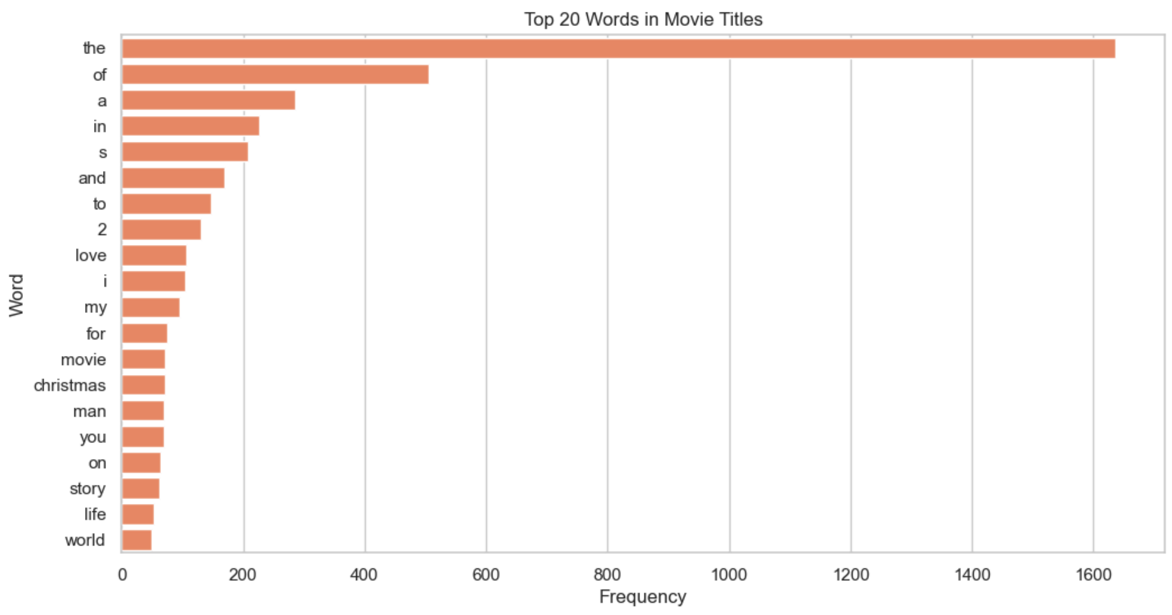
```
    147 elif map_type == "categorical":
    149     cmap = norm = None
--> 150     levels, lookup_table = self.categorical_mapping(
    151         data, palette, order,
    152     )
    154 # --- Option 3: datetime mapping
    155
    156 else:
    157     # TODO this needs actual implementation
    158     cmap = norm = None

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_base.py:248, in HueMappi
ng.categorical_mapping(self, data, palette, order)
    246         colors = self._check_list_length(levels, palette, "palette")
    247     else:
--> 248         colors = color_palette(palette, n_colors)
    250     lookup_table = dict(zip(levels, colors))
    252 return levels, lookup_table

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\palettes.py:237, in color
_palette(palette, n_colors, desat, as_cmap)
    235             palette = mpl_palette(palette, n_colors, as_cmap=as_cmap)
    236         except (ValueError, KeyError):  # Error class changed in mpl36
--> 237             raise ValueError(f"{palette!r} is not a valid palette name")
    239 if desat is not None:
    240     palette = [desaturate(c, desat) for c in palette]

ValueError: 'lightgreen' is not a valid palette name
```

In [28]:
```python
top_directors = data['director'].value_counts().head(10)

plt.figure(figsize=(10,6))
sns.barplot(x=top_directors.values, y=top_directors.index, color='lightgreen')
plt.title("Top 10 Directors with the Most Titles")
plt.xlabel("Number of Titles")
plt.ylabel("Director")
plt.show()
```

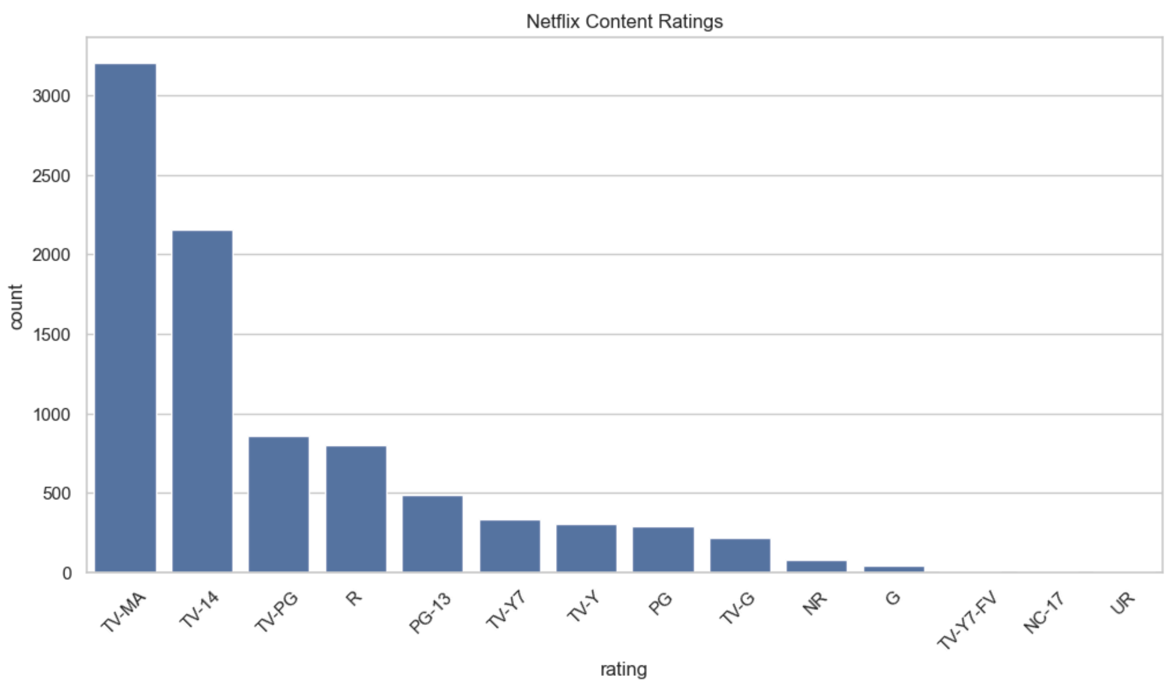




```
In [31]: titles = " ".join(data[data['type']=='Movie']['title']).lower()
         words = re.findall(r'\b\w+\b', titles)
         top_words = Counter(words).most_common(20)
         words_list, counts = zip(*top_words)

         plt.figure(figsize=(12,6))
         sns.barplot(x=list(counts), y=list(words_list), color='coral')
         plt.title("Top 20 Words in Movie Titles")
         plt.xlabel("Frequency")
         plt.ylabel("Word")
         plt.show()
```

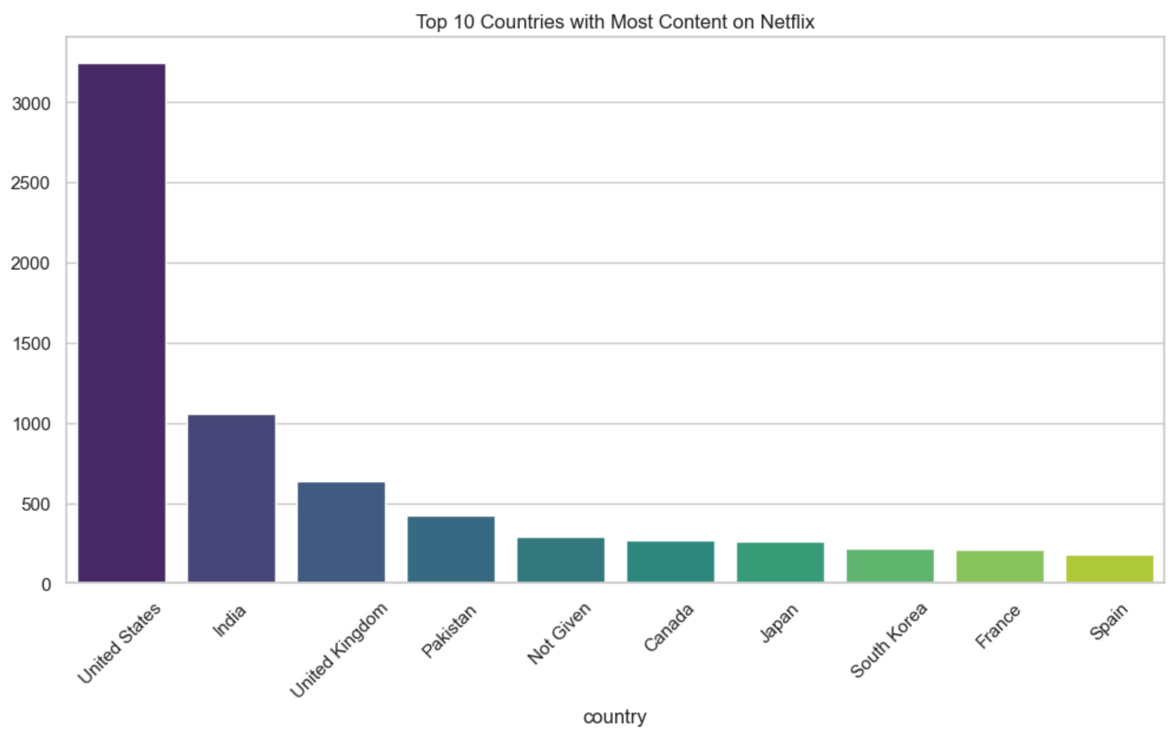


```
In [32]: ratings = data['rating'].value_counts().reset_index()
         ratings.columns = ['rating','count']

         plt.figure(figsize=(12,6))
         sns.barplot(x='rating', y='count', data=ratings)
         plt.xticks(rotation=45)
         plt.title("Netflix Content Ratings")
         plt.show()
```



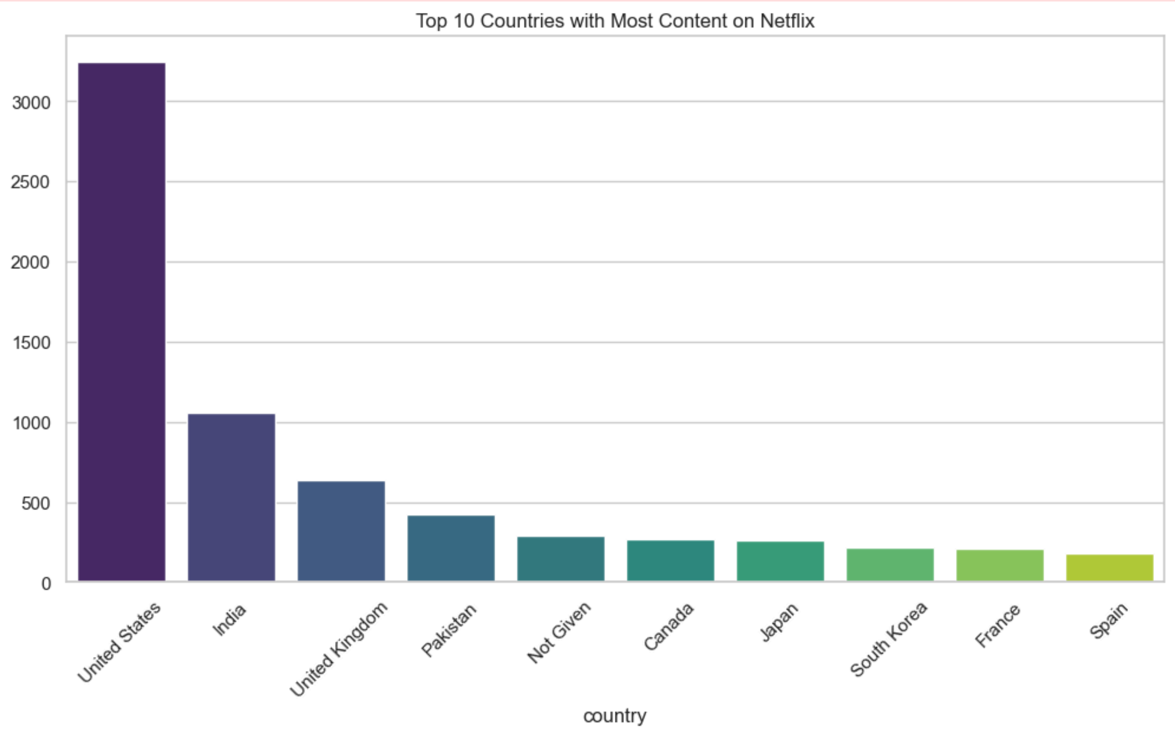
```
In [35]: top_countries = data['country'].value_counts().head(10)

         plt.figure(figsize=(12,6))
         sns.barplot(x=top_countries.index, y=top_countries.values, palette='viridis')
         plt.xticks(rotation=45)
         plt.title("Top 10 Countries with Most Content on Netflix")
         plt.show()
```

```
C:\Users\krish\AppData\Local\Temp\ipykernel_16000\2444156998.py:4: FutureWarning:

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=top_countries.index, y=top_countries.values, palette='viridis')
```

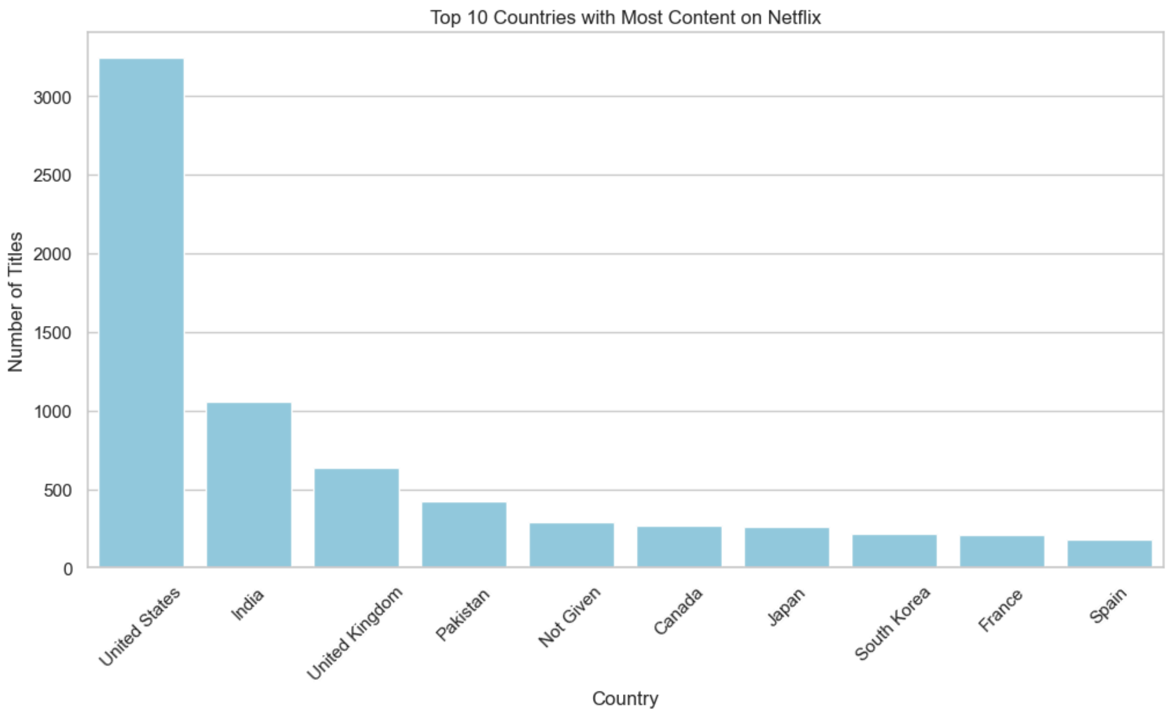


```
In [36]: top_countries = data['country'].value_counts().head(10)

         plt.figure(figsize=(12,6))
         sns.barplot(x=top_countries.index, y=top_countries.values, palette='viridis')
         plt.xticks(rotation=45)
         plt.title("Top 10 Countries with Most Content on Netflix")
         plt.show()
```

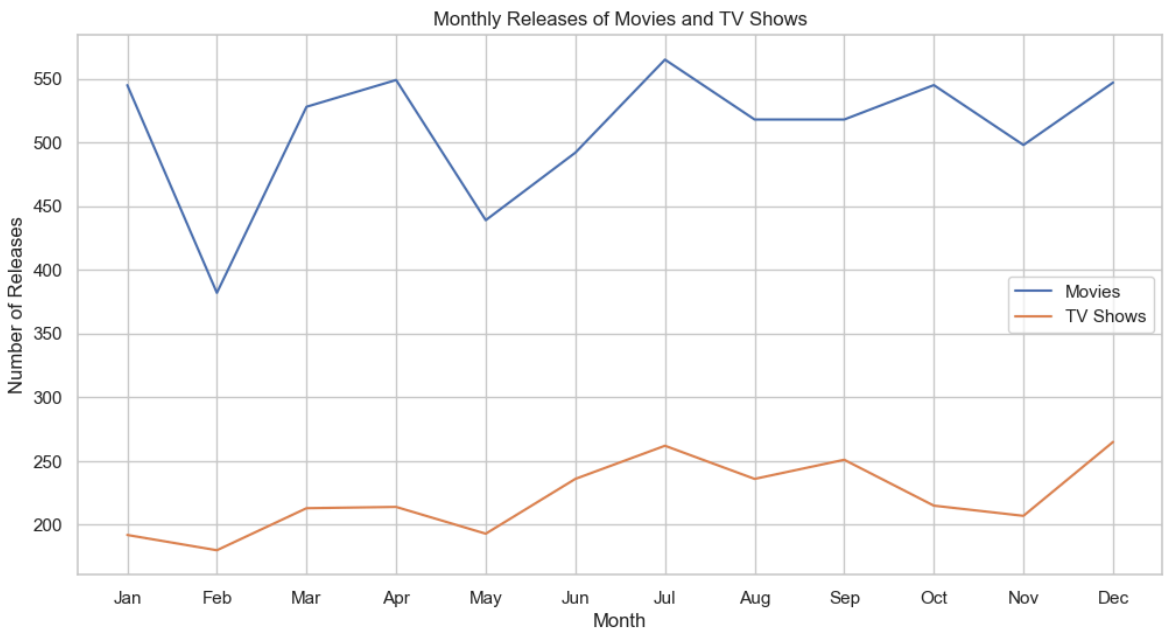
```
C:\Users\krish\AppData\Local\Temp\ipykernel_16000\2444156998.py:4: FutureWarning:

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=top_countries.index, y=top_countries.values, palette='viridis')
```

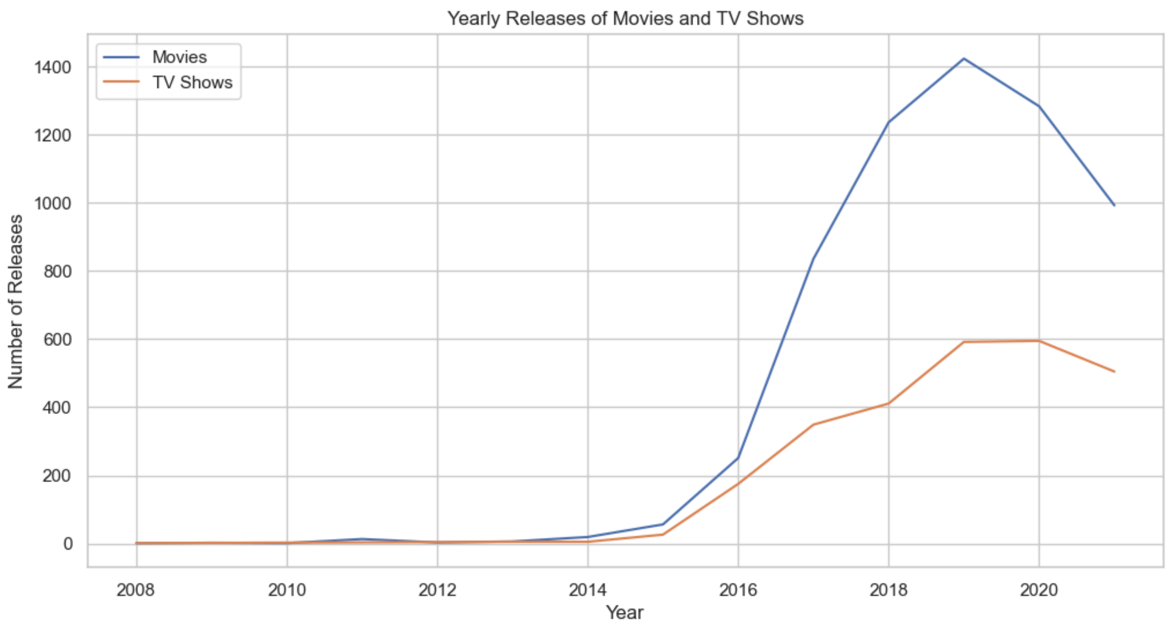


In [37]:
```
top_countries = data['country'].value_counts().head(10)

plt.figure(figsize=(12,6))
sns.barplot(x=top_countries.index, y=top_countries.values, color='skyblue')
plt.xticks(rotation=45)
plt.title("Top 10 Countries with Most Content on Netflix")
plt.xlabel("Country")
plt.ylabel("Number of Titles")
plt.show()
```

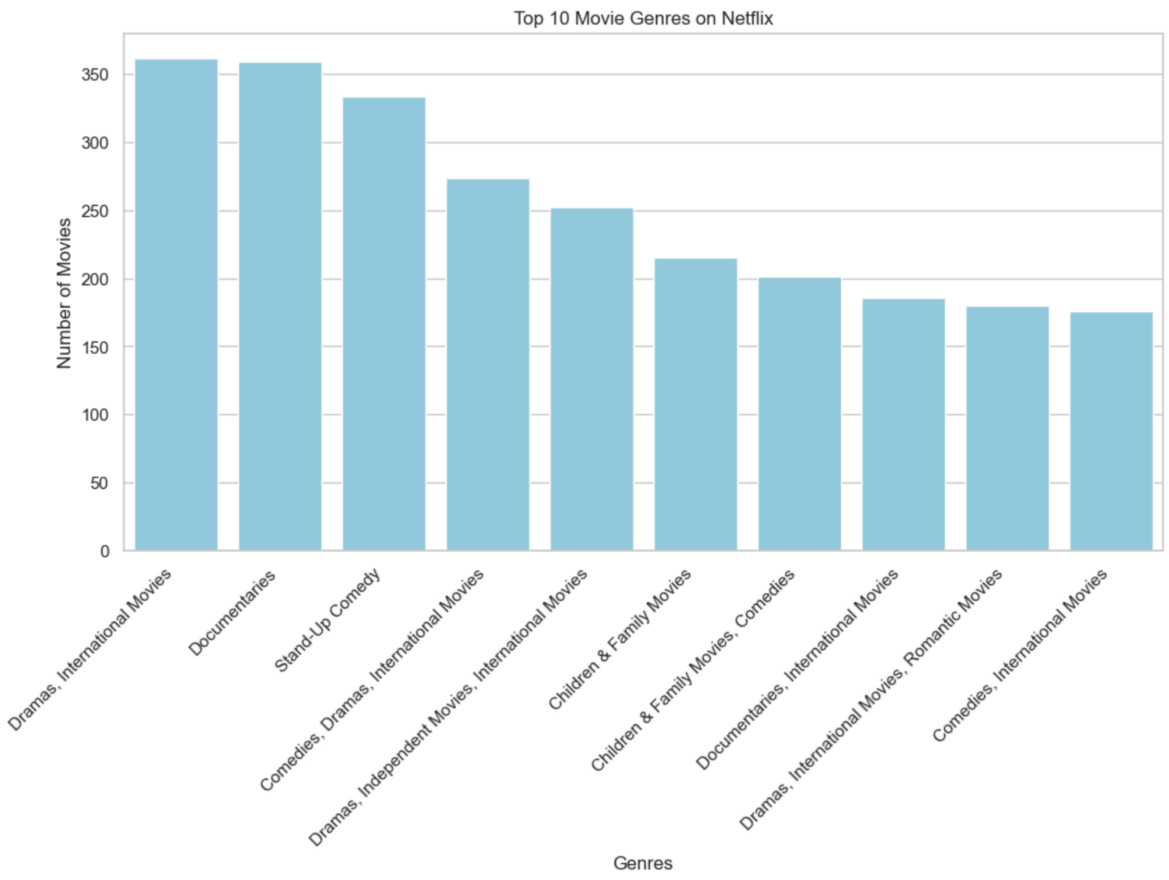


In [38]:
```
data['year_added'] = data['date_added'].dt.year
data['month_added'] = data['date_added'].dt.month

monthly_movies = data[data['type']=='Movie']['month_added'].value_counts().sort_
monthly_tv = data[data['type']=='TV Show']['month_added'].value_counts().sort_in

plt.figure(figsize=(12,6))
plt.plot(monthly_movies.index, monthly_movies.values, label='Movies')
plt.plot(monthly_tv.index, monthly_tv.values, label='TV Shows')
plt.xticks(range(1,13), ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','
plt.xlabel("Month")
plt.ylabel("Number of Releases")
plt.title("Monthly Releases of Movies and TV Shows")
plt.legend()
plt.grid(True)
plt.show()

yearly_movies = data[data['type']=='Movie']['year_added'].value_counts().sort_in
yearly_tv = data[data['type']=='TV Show']['year_added'].value_counts().sort_inde

plt.figure(figsize=(12,6))
plt.plot(yearly_movies.index, yearly_movies.values, label='Movies')
plt.plot(yearly_tv.index, yearly_tv.values, label='TV Shows')
plt.xlabel("Year")
plt.ylabel("Number of Releases")
plt.title("Yearly Releases of Movies and TV Shows")
plt.legend()
plt.grid(True)
plt.show()
```

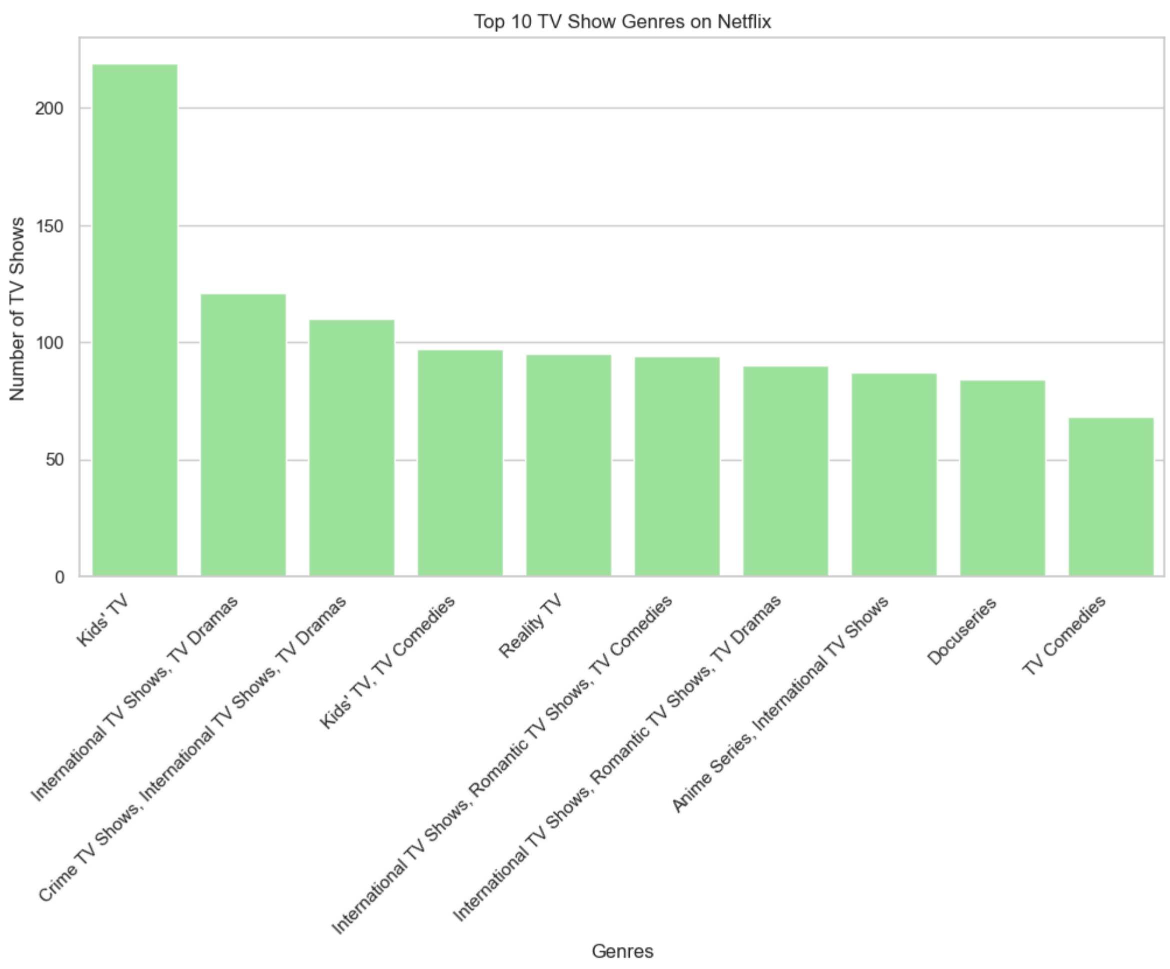




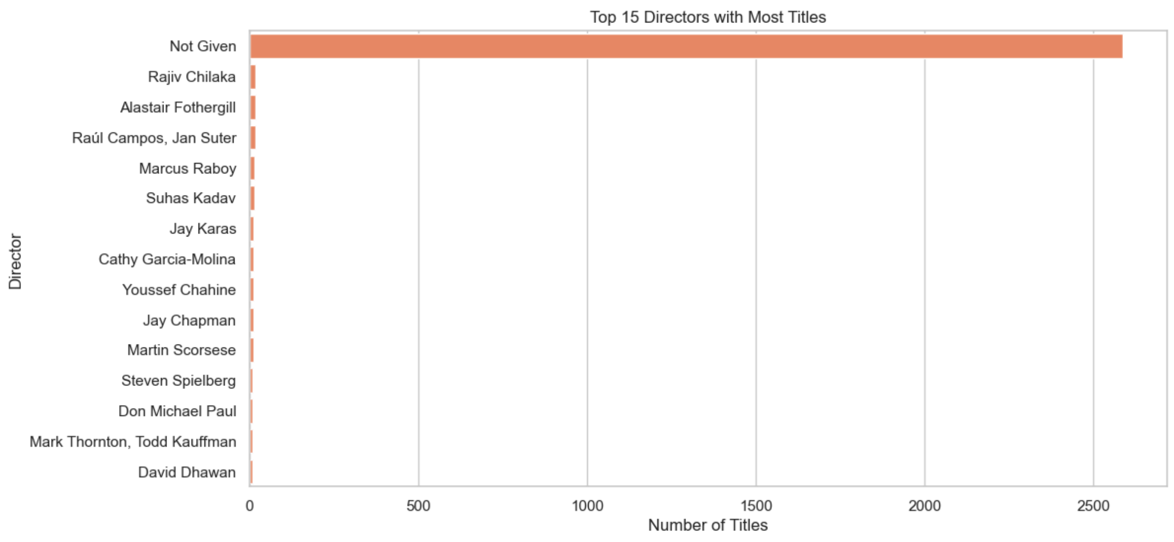
In [39]:
```
popular_movie_genre = data[data['type']=='Movie'].groupby("listed_in").size().sc

plt.figure(figsize=(12,6))
sns.barplot(x=popular_movie_genre.index, y=popular_movie_genre.values, color='sk
plt.xticks(rotation=45, ha='right')
plt.xlabel("Genres")
plt.ylabel("Number of Movies")
plt.title("Top 10 Movie Genres on Netflix")
plt.show()
```



In [40]:
```
popular_tv_genre = data[data['type']=='TV Show'].groupby("listed_in").size().sor

plt.figure(figsize=(12,6))
sns.barplot(x=popular_tv_genre.index, y=popular_tv_genre.values, color='lightgre
plt.xticks(rotation=45, ha='right')
plt.xlabel("Genres")
plt.ylabel("Number of TV Shows")
plt.title("Top 10 TV Show Genres on Netflix")
plt.show()
```



```
In [41]: top_directors = data['director'].value_counts().head(15)

         plt.figure(figsize=(12,6))
         sns.barplot(x=top_directors.values, y=top_directors.index, color='coral')
         plt.xlabel("Number of Titles")
         plt.ylabel("Director")
         plt.title("Top 15 Directors with Most Titles")
         plt.show()
```



```
In [ ]:
```