Project : Analysis of OTT Platforms

Domain : Sales

Organisation : Vigor Council

Intern Name : Abhilasha Chatterjee

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

In [2]:
```
#importing csv file
df=pd.read_csv('C:/Users/User/Desktop/INTERNSHIP/PROJECT 1/ottdataset.csv')
df.head()
```

Out[2]:

	Unnamed: 0	show_id	OTT Platform	type	title	director	cast	country	date_added	release_year	r
0	1	S0002	Amazon	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar	India	March 30, 2021	2018	
1	9670	S9671	Disney	Movie	Ice Age: A Mammoth Christmas	Karen Disher	Raymond Albert Romano, John Leguizamo, Denis L...	United States	November 26, 2021	2011	
2	9673	S9674	Disney	Movie	Becoming Cousteau	Liz Garbus	Jacques Yves Cousteau, Vincent Cassel	United States	November 24, 2021	2021	
3	9677	S9678	Disney	Movie	A Muppets Christmas: Letters To Santa	Kirk R. Thatcher	Steve Whitmire, Dave Goelz, Bill Barretta, Eri...	United States	November 19, 2021	2008	
4	9680	S9681	Disney	Movie	The Pixar Story	Leslie Iwerks	Stacy Keach, John Lasseter, Brad Bird, John Mu...	United States	November 19, 2021	2007	

In [3]:
```
df.columns
```

Out[3]:
```
Index(['Unnamed: 0', 'show_id', 'OTT Platform', 'type', 'title', 'director',
       'cast', 'country', 'date_added', 'release_year', 'rating', 'duration',
       'listed_in', 'description'],
      dtype='object')
```

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

Out[4]:
```
(6151, 14)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6151 entries, 0 to 6150
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Unnamed: 0    6151 non-null   int64 
 1   show_id       6151 non-null   object
 2   OTT Platform  6151 non-null   object
 3   type          6151 non-null   object
 4   title         6151 non-null   object
 5   director      6151 non-null   object
 6   cast          6151 non-null   object
 7   country       6151 non-null   object
 8   date_added    6151 non-null   object
 9   release_year  6151 non-null   int64 
 10  rating        6151 non-null   object
 11  duration      6151 non-null   object
 12  listed_in     6151 non-null   object
 13  description   6151 non-null   object
dtypes: int64(2), object(12)
memory usage: 672.9+ KB
```

In [6]:
```
df.drop(['Unnamed: 0'],axis=1,inplace=True)
```

In [7]:
```
df['OTT Platform'].value_counts()
```

Out[7]:
```
OTT Platform
Netflix    5332
Disney      818
Amazon        1
Name: count, dtype: int64
```

In [8]:
```
df['type'].value_counts()
```

Out[8]:
```
type
Movie      6004
TV Show     147
Name: count, dtype: int64
```

In [9]:
```
df.head()
```

Out[9]:

	show_id	OTT Platform	type	title	director	cast	country	date_added	release_year	rating	durat
0	S0002	Amazon	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar	India	March 30, 2021	2018	13+	110
1	S9671	Disney	Movie	Ice Age: A Mammoth Christmas	Karen Disher	Raymond Albert Romano, John Leguizamo, Denis L...	United States	November 26, 2021	2011	TV-G	23
2	S9674	Disney	Movie	Becoming Cousteau	Liz Garbus	Jacques Yves Cousteau,	United States	November 24, 2021	2021	PG-13	94

						Vincent Cassel					
3	S9678	Disney	Movie	A Muppets Christmas: Letters To Santa	Kirk R. Thatcher	Steve Whitmire, Dave Goelz, Bill Barretta, Eri...	United States	November 19, 2021	2008	G	45
4	S9681	Disney	Movie	The Pixar Story	Leslie Iwerks	Stacy Keach, John Lasseter, Brad Bird, John Mu...	United States	November 19, 2021	2007	G	91

In [10]:
```
def duration_to_min(duration):
    if 'Season' in duration:
        return int(duration.split()[0]) * 22* 30
    else:
        return int(duration.split()[0])

df['duration_min'] = df['duration'].map(duration_to_min)
df.drop(['duration'],axis=1,inplace=True)
df.head()
```

Out[10]:

	show_id	OTT Platform	type	title	director	cast	country	date_added	release_year	rating	li
0	S0002	Amazon	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar	India	March 30, 2021	2018	13+	Inter
1	S9671	Disney	Movie	Ice Age: A Mammoth Christmas	Karen Disher	Raymond Albert Romano, John Leguizamo, Denis L...	United States	November 26, 2021	2011	TV-G	An
2	S9674	Disney	Movie	Becoming Cousteau	Liz Garbus	Jacques Yves Cousteau, Vincent Cassel	United States	November 24, 2021	2021	PG-13	Biogr Docu
3	S9678	Disney	Movie	A Muppets Christmas: Letters To Santa	Kirk R. Thatcher	Steve Whitmire, Dave Goelz, Bill Barretta, Eri...	United States	November 19, 2021	2008	G	C
4	S9681	Disney	Movie	The Pixar Story	Leslie Iwerks	Stacy Keach, John Lasseter, Brad Bird, John Mu...	United States	November 19, 2021	2007	G	Docu

In [11]:
```
# Data type change for all columns
datatype_map = {
    'show_id' : 'str',
    'OTT Platform' : 'category',
    'type' : 'category',
    'title' : 'str',
```

```
    'director' : 'category',
    'cast' : 'category',
    'country' : 'category',
    'rating' : 'category',
    'duration_min' : 'int',
    'date_added' : 'datetime64',
    'listed_in' : 'category',
    'description' : 'str'
}
# apply mapped category
df = df.astype(datatype_map,errors='ignore')
```

In [12]:
```
df.head()
```

Out[12]:

	show_id	OTT Platform	type	title	director	cast	country	date_added	release_year	rating	li
0	S0002	Amazon	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar	India	March 30, 2021	2018	13+	Inter
1	S9671	Disney	Movie	Ice Age: A Mammoth Christmas	Karen Disher	Raymond Albert Romano, John Leguizamo, Denis L...	United States	November 26, 2021	2011	TV-G	An
2	S9674	Disney	Movie	Becoming Cousteau	Liz Garbus	Jacques Yves Cousteau, Vincent Cassel	United States	November 24, 2021	2021	PG-13	Biogr Docu
3	S9678	Disney	Movie	A Muppets Christmas: Letters To Santa	Kirk R. Thatcher	Steve Whitmire, Dave Goelz, Bill Barretta, Eri...	United States	November 19, 2021	2008	G	C
4	S9681	Disney	Movie	The Pixar Story	Leslie Iwerks	Stacy Keach, John Lasseter, Brad Bird, John Mu...	United States	November 19, 2021	2007	G	Docu

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6151 entries, 0 to 6150
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype   
---  ------        --------------  -----   
 0   show_id       6151 non-null   object  
 1   OTT Platform  6151 non-null   category
 2   type          6151 non-null   category
 3   title         6151 non-null   object  
 4   director      6151 non-null   category
 5   cast          6151 non-null   category
 6   country       6151 non-null   category
 7   date_added    6151 non-null   object  
 8   release_year  6151 non-null   int64   
 9   rating        6151 non-null   category
 10  listed_in     6151 non-null   category
```

```
 11  description   6151 non-null   object
 12  duration_min  6151 non-null   int32
dtypes: category(7), int32(1), int64(1), object(4)
memory usage: 711.8+ KB
```

In [14]:
```
ratings=df['rating'].unique()
ratings
```

Out[14]:
```
['13+', 'TV-G', 'PG-13', 'G', 'PG', ..., 'TV-MA', 'R', 'NC-17', 'NR', 'UR']
Length: 15
Categories (15, object): ['13+', 'G', 'NC-17', 'NR', ..., 'TV-Y', 'TV-Y7', 'TV-Y7-FV',
'UR']
```

In [15]:
```
ratings_order = ['TV-Y', 'TV-Y7', 'TV-Y7-FV', 'G', 'PG', 'TV-G', 'TV-PG', 'PG-13', 'TV-1
```

In [16]:
```
# Reorder the 'rating' column
# df['rating'] = pd.Categorical(df['rating'], categories=ratings_order, ordered=True)
df['rating'].value_counts()
```

Out[16]:
```
rating
TV-MA       1822
TV-14       1226
R            778
PG-13        536
TV-PG        535
PG           498
G            271
TV-G         241
TV-Y7         97
TV-Y          77
NR            58
TV-Y7-FV       6
UR             3
NC-17          2
13+            1
Name: count, dtype: int64
```

In [17]:
```
print(df['date_added'])
```

```
0          March 30, 2021
1       November 26, 2021
2       November 24, 2021
3       November 19, 2021
4       November 19, 2021
              ...
6146       March 9, 2016
6147    November 20, 2019
6148     November 1, 2019
6149     January 11, 2020
6150       March 2, 2019
Name: date_added, Length: 6151, dtype: object
```

In [18]:
```
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')
```

In [19]:
```
df['date_added'] = pd.to_datetime(df['date_added'], format='%Y-%m-%d', errors='coerce')
```

In [21]:
```
df.head()
```

Out[21]:

	show_id	OTT Platform	type	title	director	cast	country	date_added	release_year	rating	li
0	S0002	Amazon	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan,	India	2021-03-30	2018	13+	Inter

						Sachin Khedekar					
1	S9671	Disney	Movie	Ice Age: A Mammoth Christmas	Karen Disher	Raymond Albert Romano, John Leguizamo, Denis L...	United States	2021-11-26	2011	TV-G	An
2	S9674	Disney	Movie	Becoming Cousteau	Liz Garbus	Jacques Yves Cousteau, Vincent Cassel	United States	2021-11-24	2021	PG-13	Biogr Docu
3	S9678	Disney	Movie	A Muppets Christmas: Letters To Santa	Kirk R. Thatcher	Steve Whitmire, Dave Goelz, Bill Barretta, Eri...	United States	2021-11-19	2008	G	
4	S9681	Disney	Movie	The Pixar Story	Leslie Iwerks	Stacy Keach, John Lasseter, Brad Bird, John Mu...	United States	2021-11-19	2007	G	Docu

In [22]:
```
# Generating new features : `delay_in_release` In years (int) we can take the difference
df['delay_in_release'] = df['date_added'].dt.year - df['release_year']
df.head()
```

Out[22]:

	show_id	OTT Platform	type	title	director	cast	country	date_added	release_year	rating	li
0	S0002	Amazon	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar	India	2021-03-30	2018	13+	Inter
1	S9671	Disney	Movie	Ice Age: A Mammoth Christmas	Karen Disher	Raymond Albert Romano, John Leguizamo, Denis L...	United States	2021-11-26	2011	TV-G	An
2	S9674	Disney	Movie	Becoming Cousteau	Liz Garbus	Jacques Yves Cousteau, Vincent Cassel	United States	2021-11-24	2021	PG-13	Biogr Docu
3	S9678	Disney	Movie	A Muppets Christmas: Letters To Santa	Kirk R. Thatcher	Steve Whitmire, Dave Goelz, Bill Barretta, Eri...	United States	2021-11-19	2008	G	
4	S9681	Disney	Movie	The Pixar Story	Leslie Iwerks	Stacy Keach, John Lasseter, Brad Bird, John Mu...	United States	2021-11-19	2007	G	Docu

In [23]:
```
director_counts = df['director'].str.split(', ').explode().value_counts().head()
```

```
# Print the most prolific directors
print("Most Prolific Directors:")
print(director_counts)
```

```
Most Prolific Directors:
director
Jan Suter       21
Raúl Campos     19
Jay Karas       16
Paul Hoen       16
Marcus Raboy    15
Name: count, dtype: int64
```

In [24]:
```
director_counts.index
```

Out[24]:
```
Index(['Jan Suter', 'Raúl Campos', 'Jay Karas', 'Paul Hoen', 'Marcus Raboy'], dtype='object', name='director')
```

In [25]:
```
director_counts.values
```

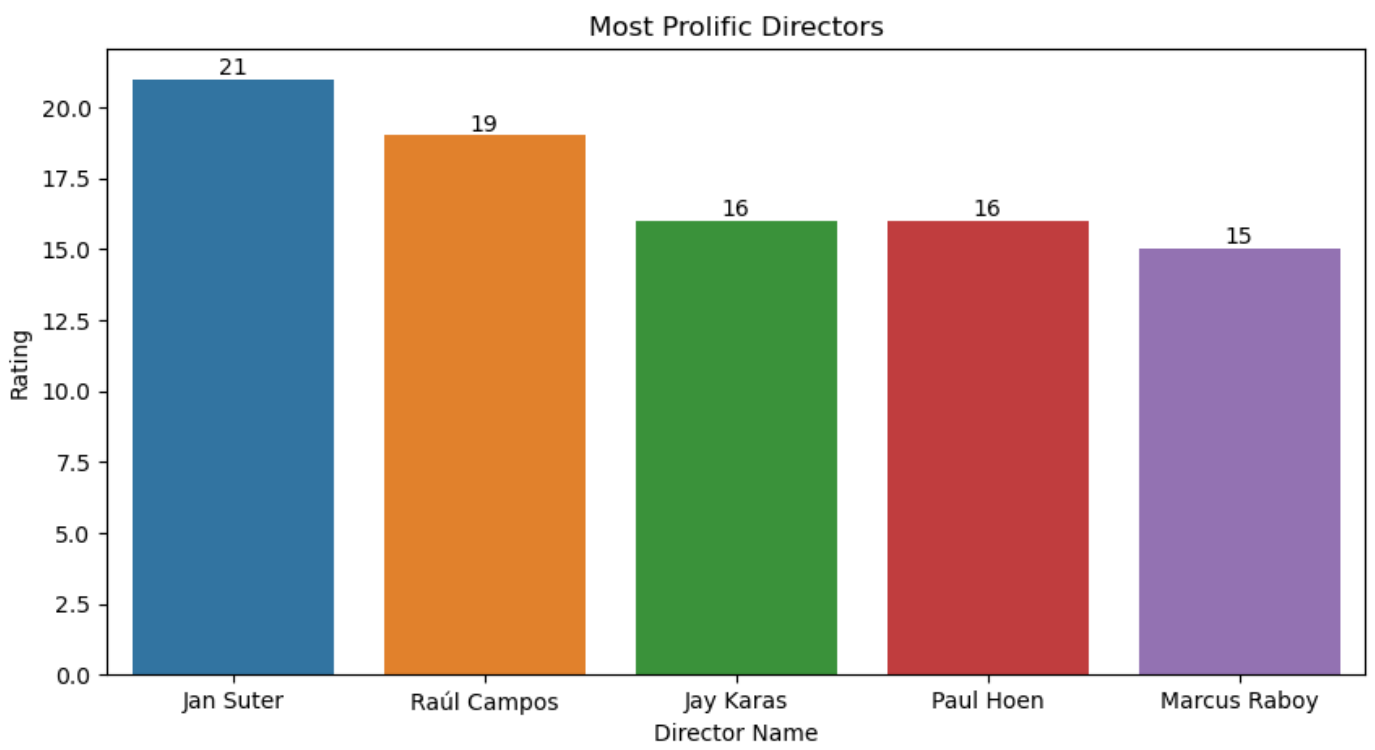
Out[25]:
```
array([21, 19, 16, 16, 15], dtype=int64)
```

In [26]:
```
# Converting series to DataFrame
data={
    'Director Name': director_counts.index,
    'Rating' : director_counts.values
}
director_df=pd.DataFrame(data)
director_df
```

Out[26]:

	Director Name	Rating
0	Jan Suter	21
1	Raúl Campos	19
2	Jay Karas	16
3	Paul Hoen	16
4	Marcus Raboy	15

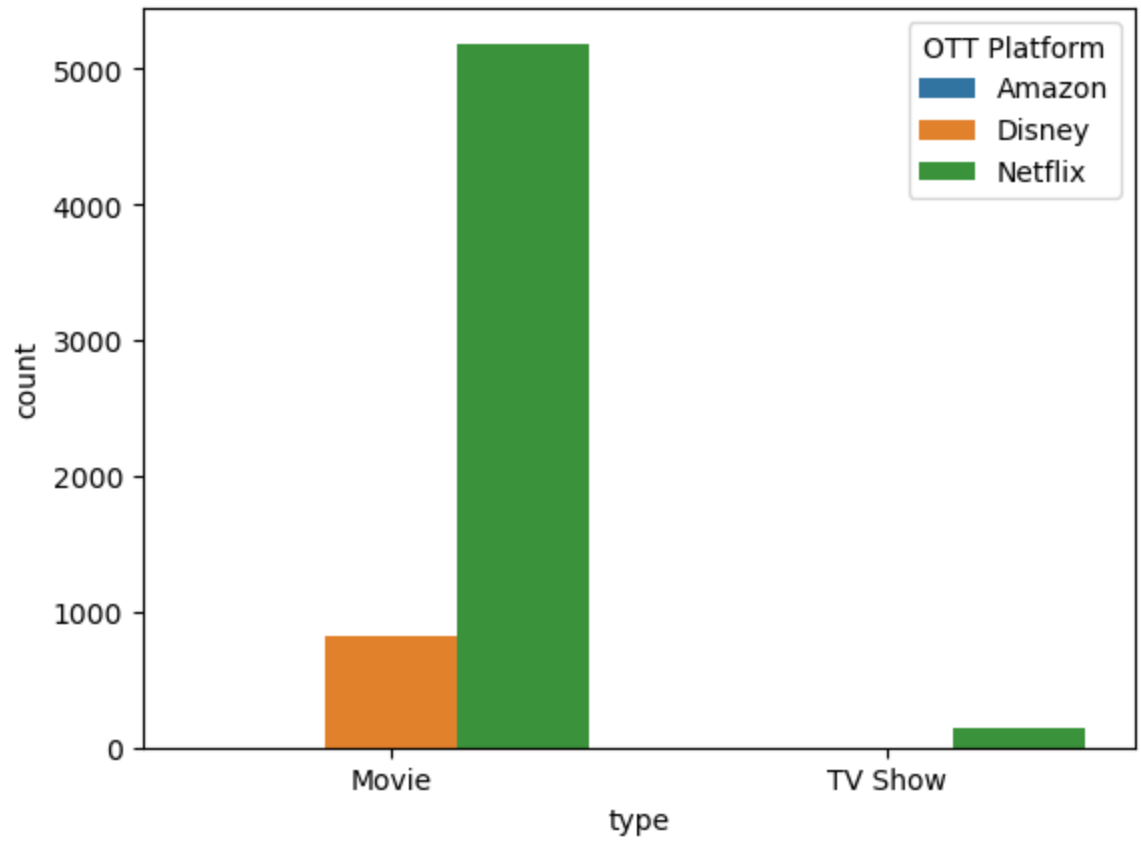
In [27]:
```
plt.figure(figsize=(10,5))
plt.title('Most Prolific Directors')
chart=sns.barplot(x='Director Name',y='Rating',data=director_df)
for v in chart.containers: chart.bar_label(v)
```



In [28]:
```
sns.countplot(x='type',data=df,hue='OTT Platform')
```

```
C:\Users\User\anaconda3\Lib\site-packages\seaborn\categorical.py:641: FutureWarning: The
default of observed=False is deprecated and will be changed to True in a future version
of pandas. Pass observed=False to retain current behavior or observed=True to adopt the
future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
```

Out[28]:
```
<Axes: xlabel='type', ylabel='count'>
```

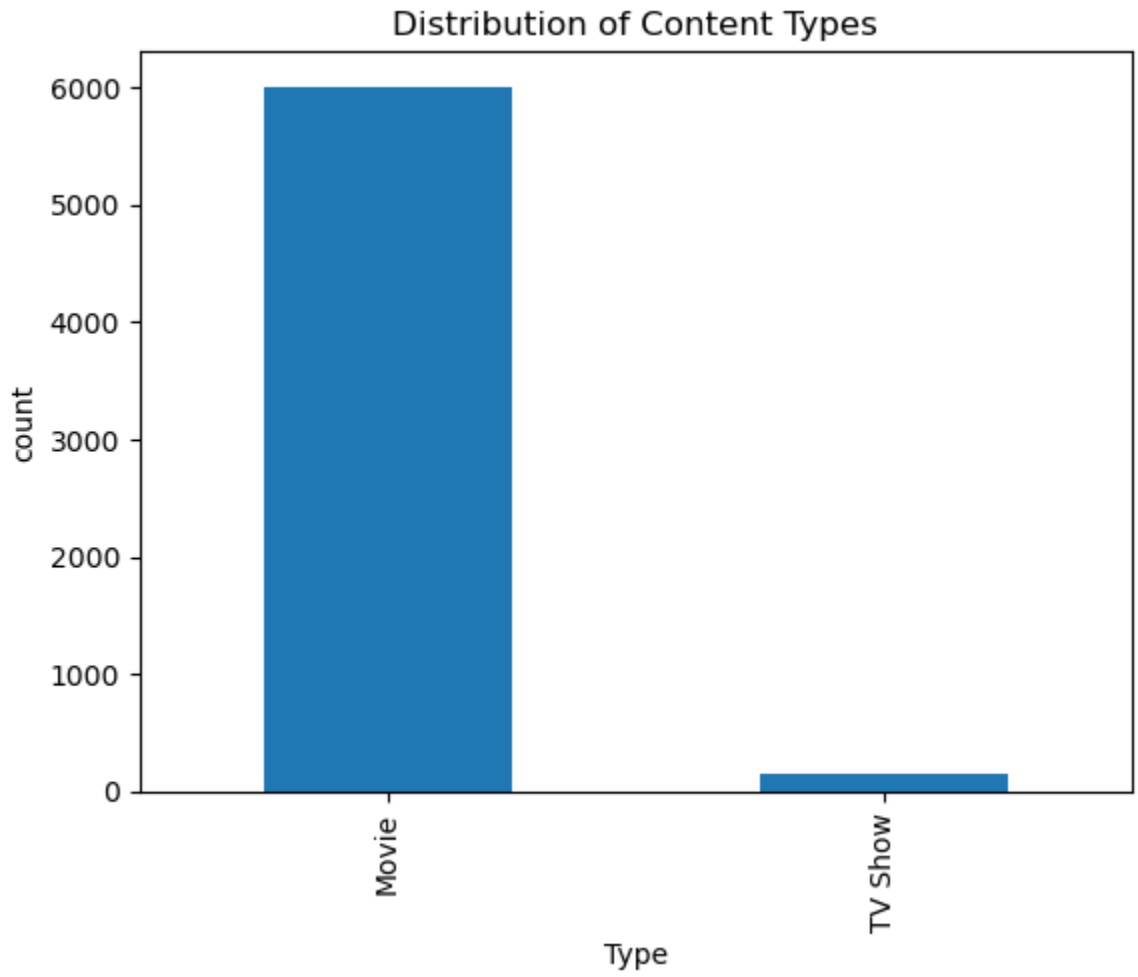


1. Distribution of Content Types [Movies V/s TV

Shows]

In [30]:
```
# This shows how many entries of movies and tv shows are there
distribution= df['type'].value_counts()
distribution.plot(kind='bar',title='Distribution of Content Types', xlabel='Type' , ylab
```

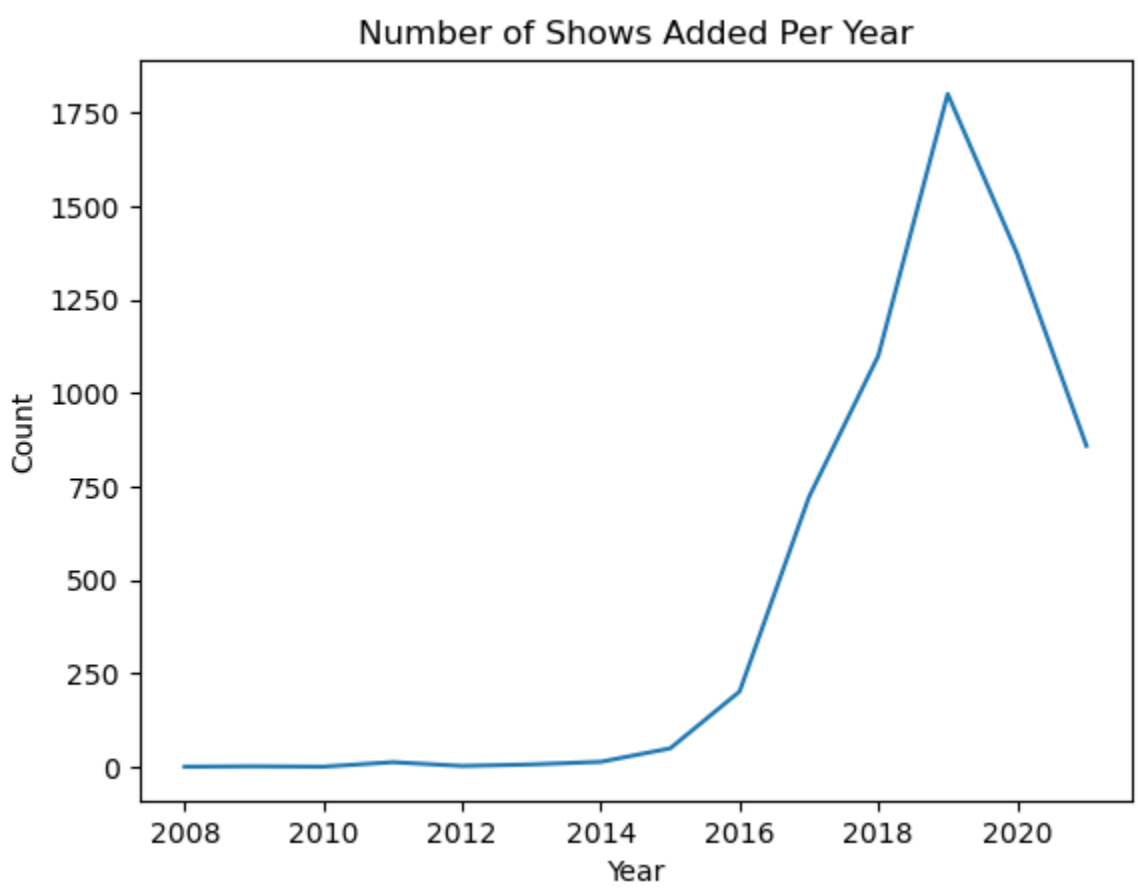
Out[30]:
```
<Axes: title={'center': 'Distribution of Content Types'}, xlabel='Type', ylabel='count'>
```



2. Number of Shows Added Per year

In [31]:
```
df['date_added'] = pd.to_datetime(df['date_added'], format='%B %d, %Y', errors='coerce')
shows_per_year=df['date_added'].dt.year.value_counts().sort_index()
shows_per_year.plot(kind='line', title='Number of Shows Added Per Year', xlabel= 'Year'
```

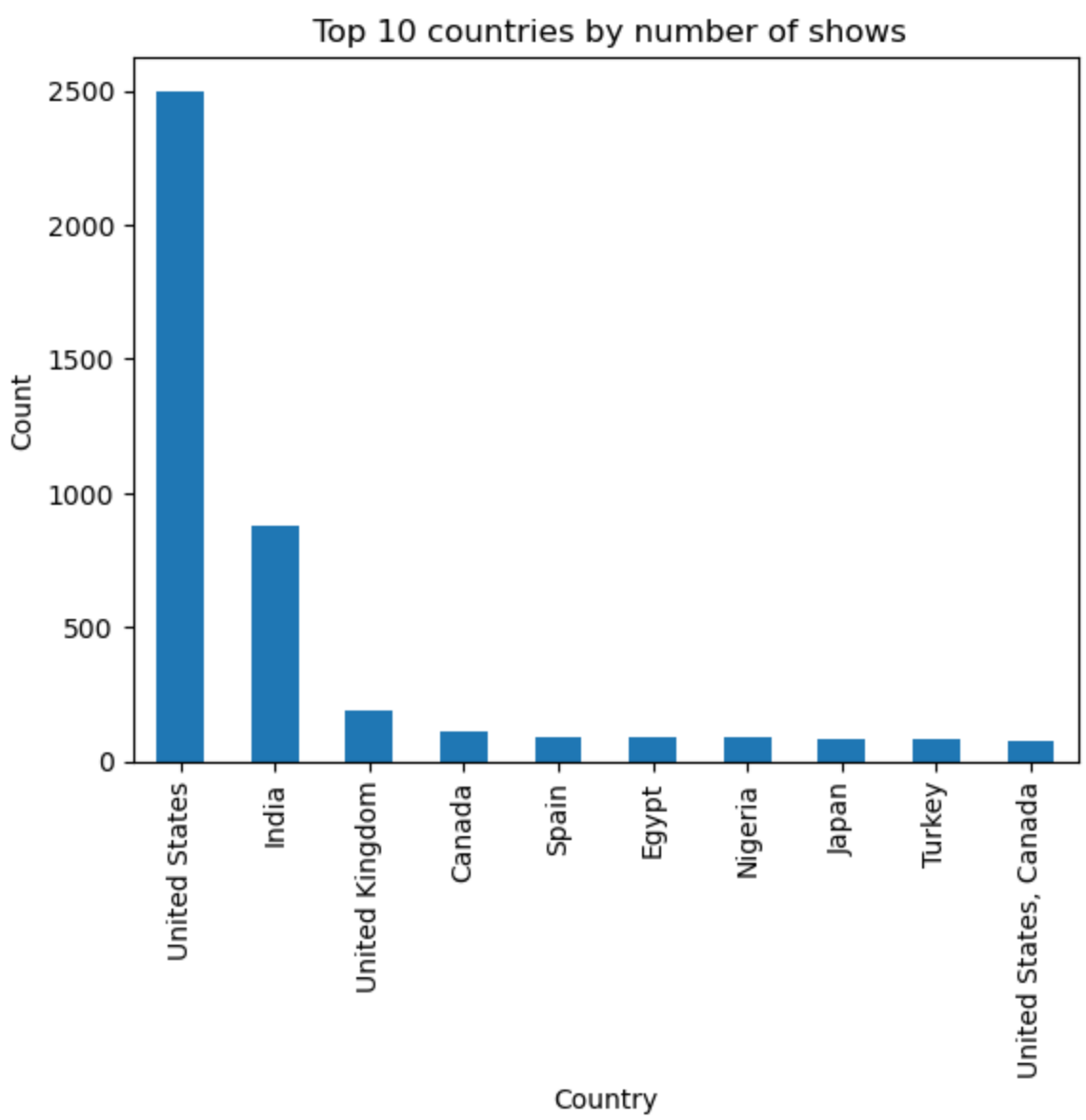
Out[31]:
```
<Axes: title={'center': 'Number of Shows Added Per Year'}, xlabel='Year', ylabel='Coun
t'>
```



3. Top 10 countries by number of shows

In [32]:
```
top_countries=df['country'].value_counts().head(10)
#value_counts().head(10): Counts the unique values in the 'country' column and selects t
top_countries.plot(kind='bar', title=' Top 10 countries by number of shows', xlabel= 'Co
```

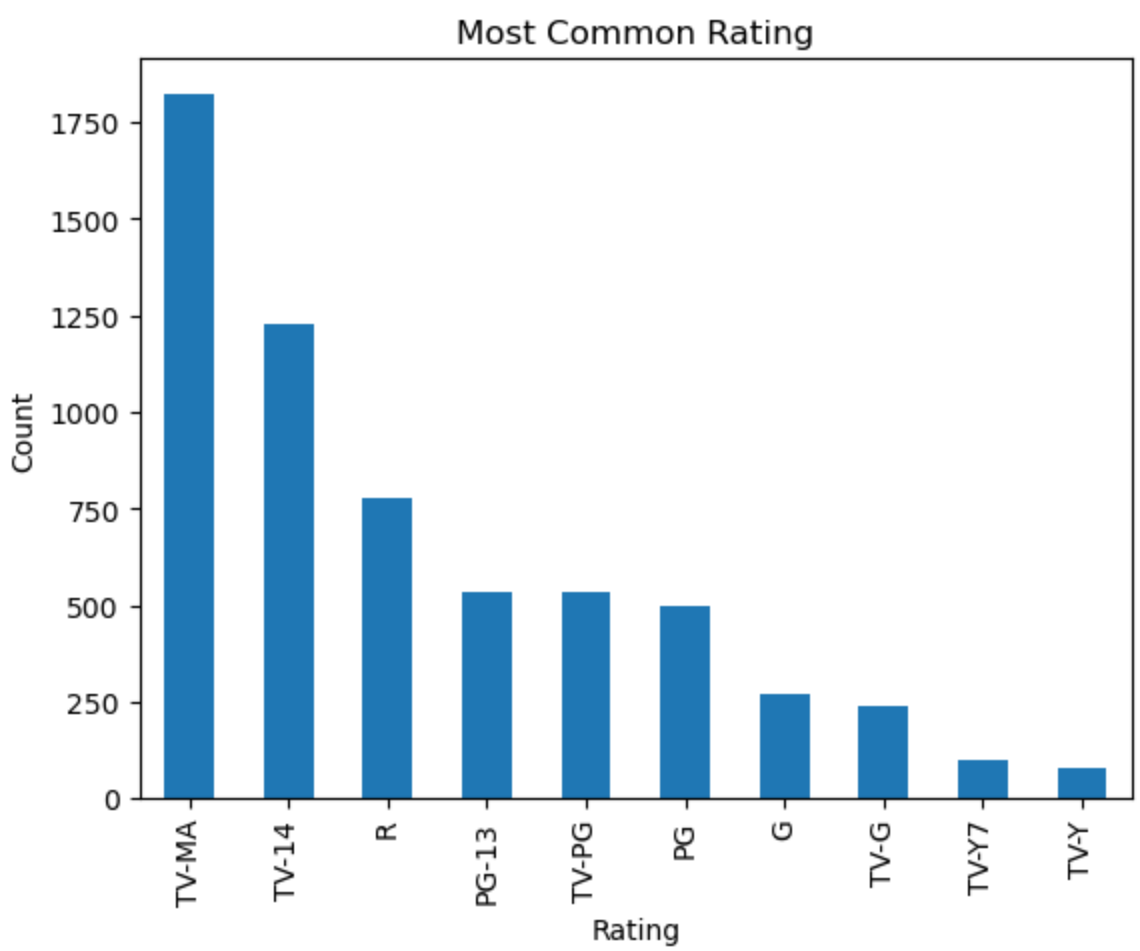
Out[32]:
```
<Axes: title={'center': ' Top 10 countries by number of shows'}, xlabel='Country', ylabe
l='Count'>
```



4. Most Common Content Ratings

In [33]:
```
count_ratings= df['rating'].value_counts().head(10)
count_ratings.plot(kind='bar', title='Most Common Rating',xlabel='Rating',ylabel='Count'
```

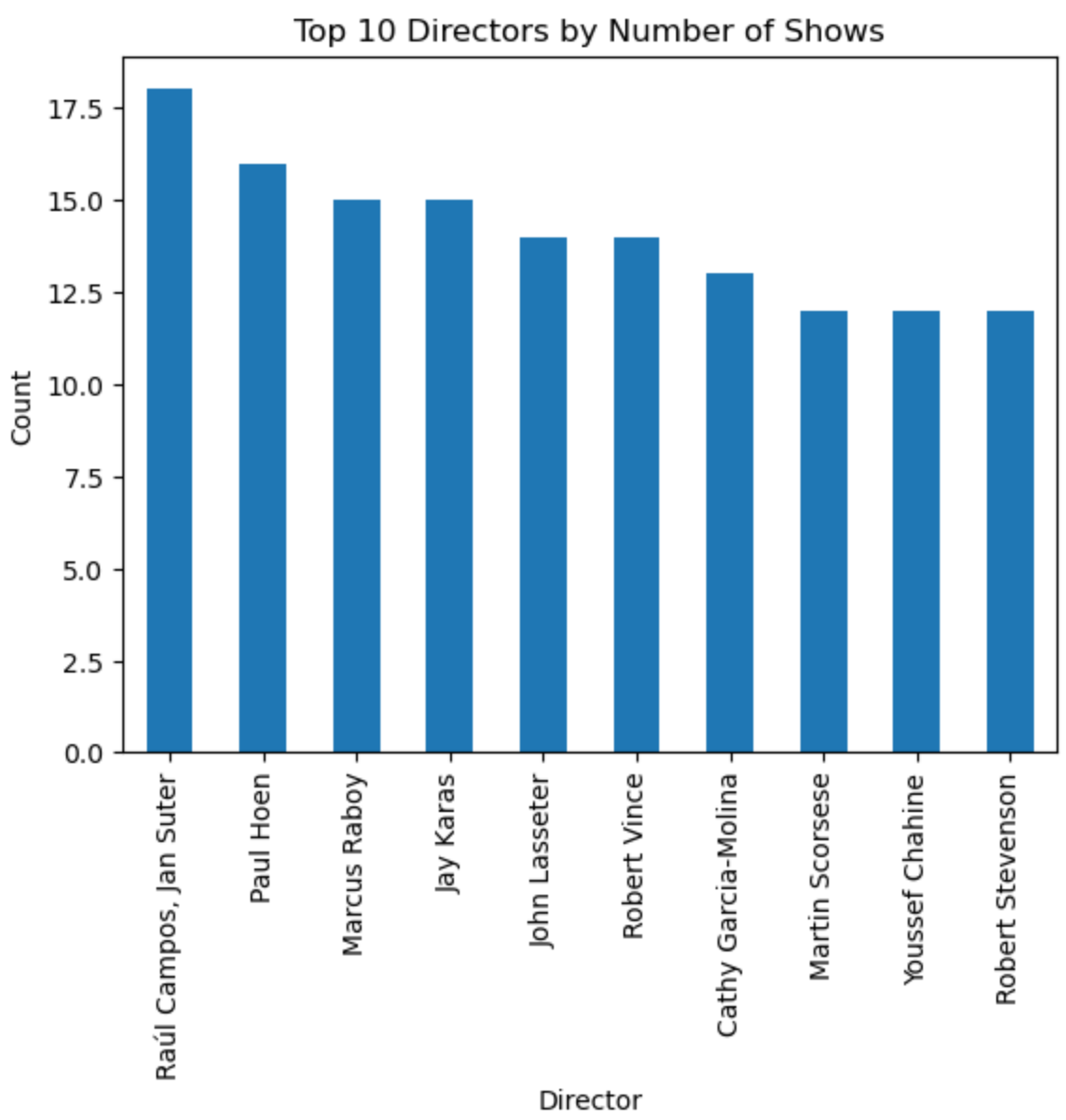
Out[33]:
```
<Axes: title={'center': 'Most Common Rating'}, xlabel='Rating', ylabel='Count'>
```



5. Top 10 directors by number of shows

In [34]:
```
top_directors = df['director'].value_counts().head(10)
top_directors.plot(kind='bar', title='Top 10 Directors by Number of Shows', xlabel='Dire
```

Out[34]:
```
<Axes: title={'center': 'Top 10 Directors by Number of Shows'}, xlabel='Director', ylabe
l='Count'>
```



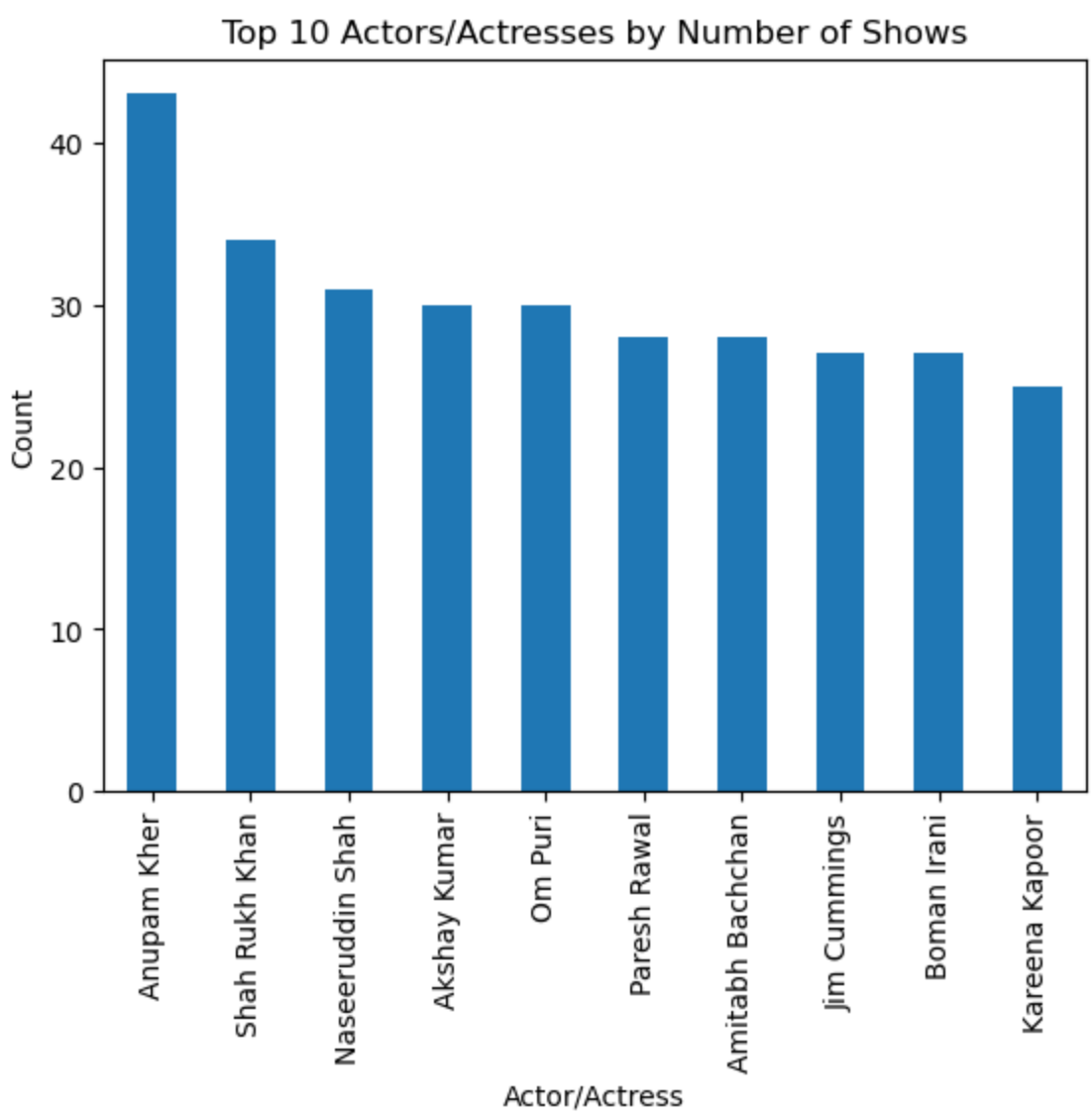
6. Top 10 actors/actresses by number of shows

For calculating the actors/actresses we have to first split the 'Cast' column into a list of actors.

Then we have to count the occurences of each actor by 'counter(flat_cast)

In [36]:
```
from collections import Counter

cast_members = df['cast'].str.split(', ')
flat_cast = [item for sublist in cast_members.dropna() for item in sublist]
top_cast = pd.Series(Counter(flat_cast)).sort_values(ascending=False).head(10)
top_cast.plot(kind='bar', title='Top 10 Actors/Actresses by Number of Shows', xlabel='Ac
```

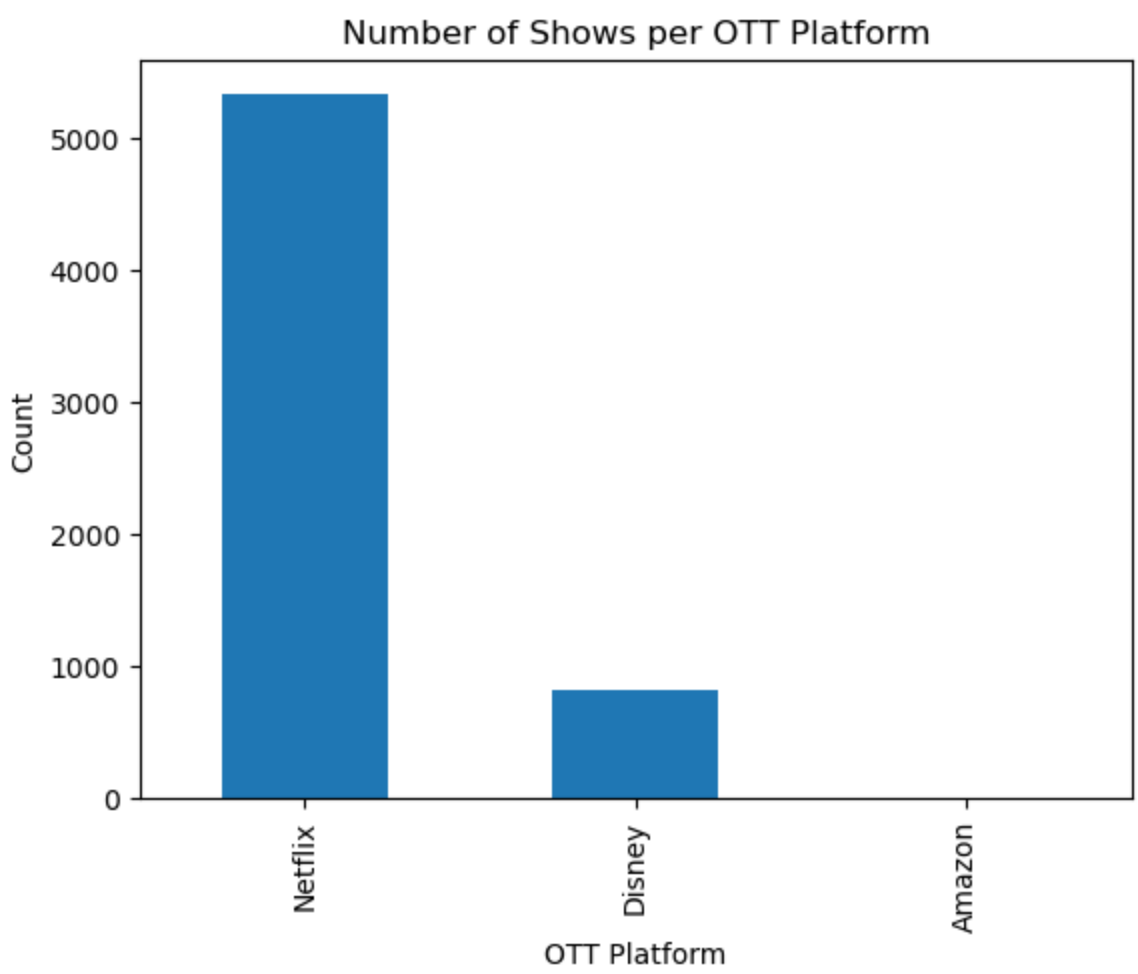
Out[36]:
```
<Axes: title={'center': 'Top 10 Actors/Actresses by Number of Shows'}, xlabel='Actor/Act
ress', ylabel='Count'>
```



7. Number of shows per OTT platform

In [37]:
```
shows_per_platform = df['OTT Platform'].value_counts()
shows_per_platform.plot(kind='bar', title='Number of Shows per OTT Platform', xlabel='OT
```

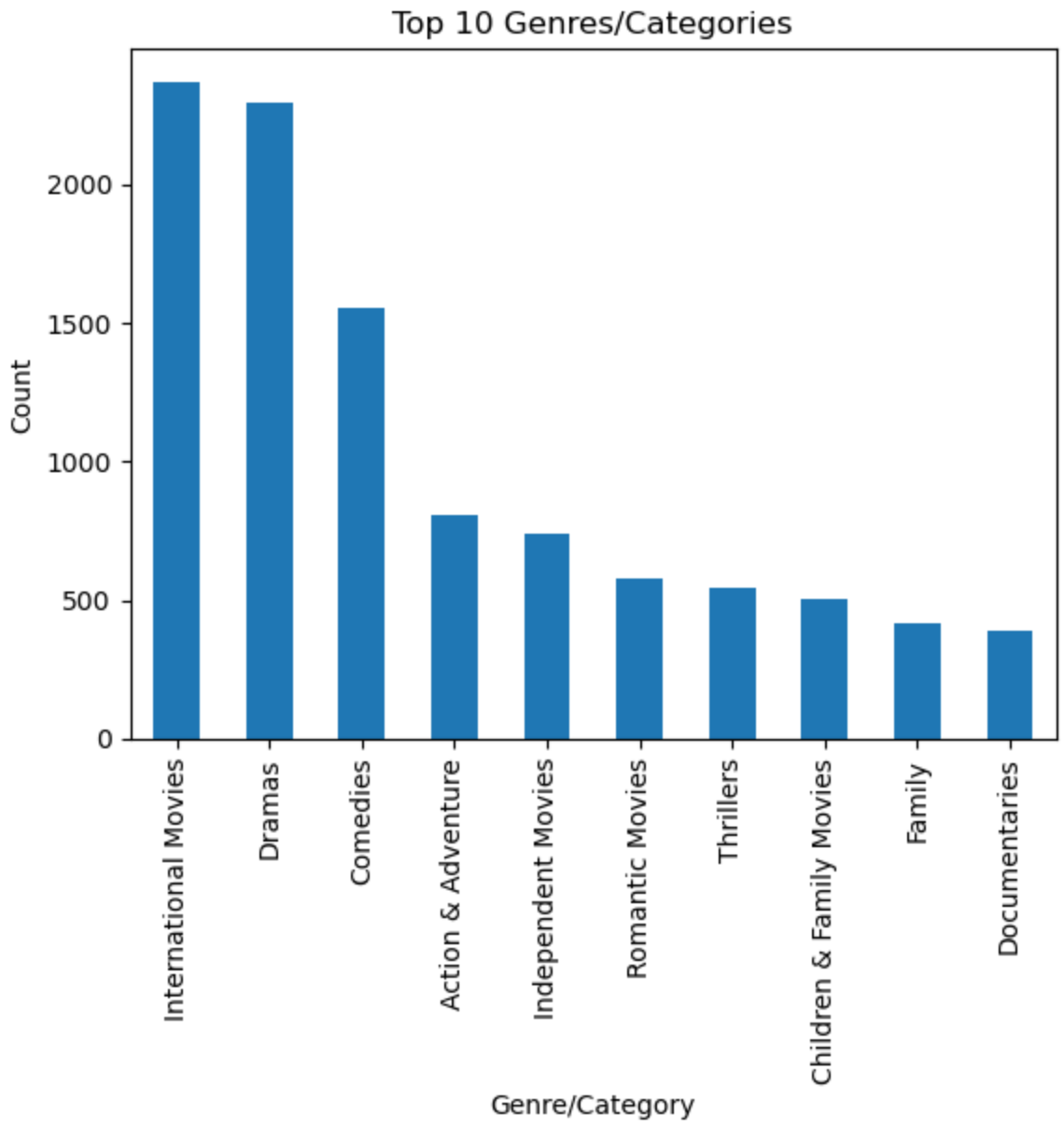
Out[37]:
```
<Axes: title={'center': 'Number of Shows per OTT Platform'}, xlabel='OTT Platform', ylab
el='Count'>
```



8. Distribution of Genres/Categories

In [38]:
```
df['listed_in'] = df['listed_in'].astype(str).fillna('')

# Split the 'listed_in' column and explode it into separate rows
#Split and Explode: Split the 'listed_in' column on ', ' and use the explode method to t
df_exploded = df.assign(listed_in=df['listed_in'].str.split(', ')).explode('listed_in')

# Get the top 10 genres/categories
category_distribution = df_exploded['listed_in'].value_counts().head(10)

# Plot the distribution
category_distribution.plot(kind='bar', title='Top 10 Genres/Categories', xlabel='Genre/C
```

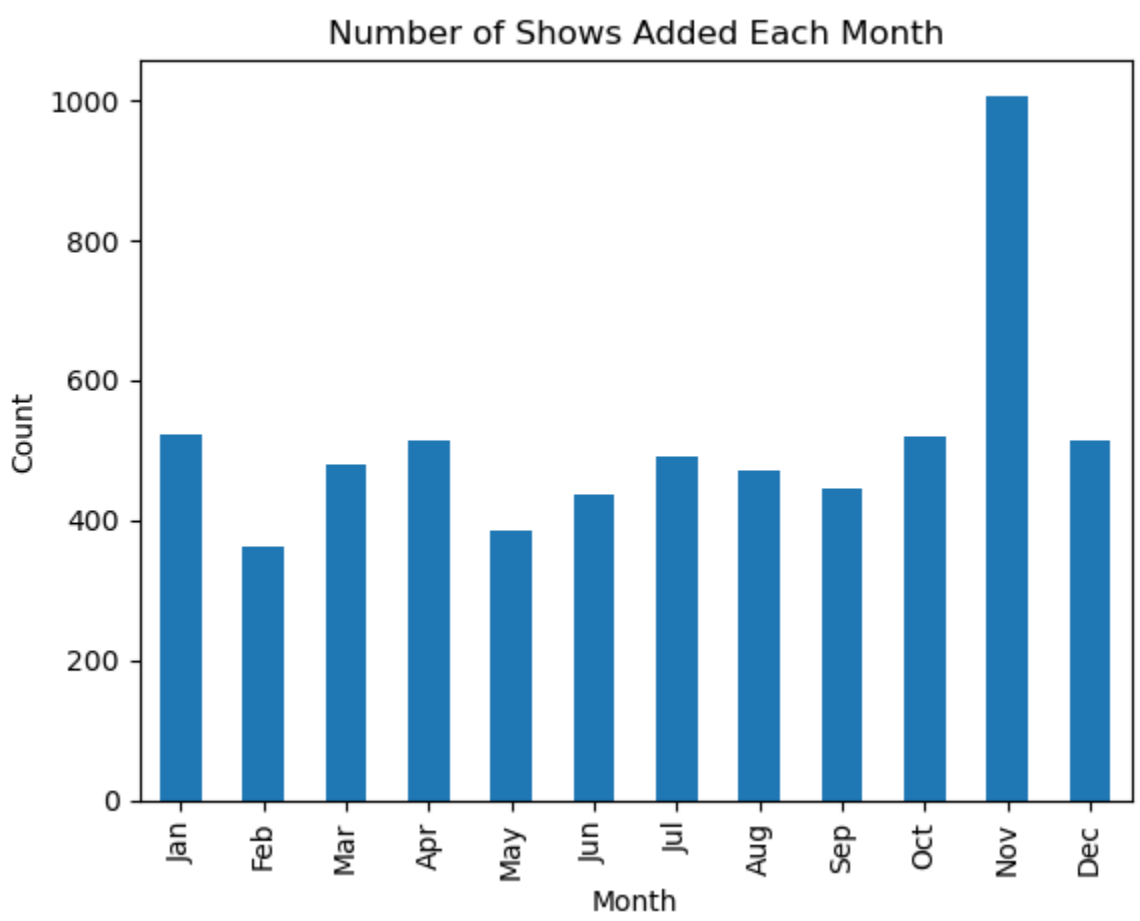
Out[38]:
```
<Axes: title={'center': 'Top 10 Genres/Categories'}, xlabel='Genre/Category', ylabel='Co
unt'>
```



9. Number of Shows added each month

In [39]:
```
shows_per_month = df['date_added'].dt.month.value_counts().sort_index()
shows_per_month.index = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',
shows_per_month.plot(kind='bar', title='Number of Shows Added Each Month', xlabel='Month
```

Out[39]:
```
<Axes: title={'center': 'Number of Shows Added Each Month'}, xlabel='Month', ylabel='Cou
nt'>
```

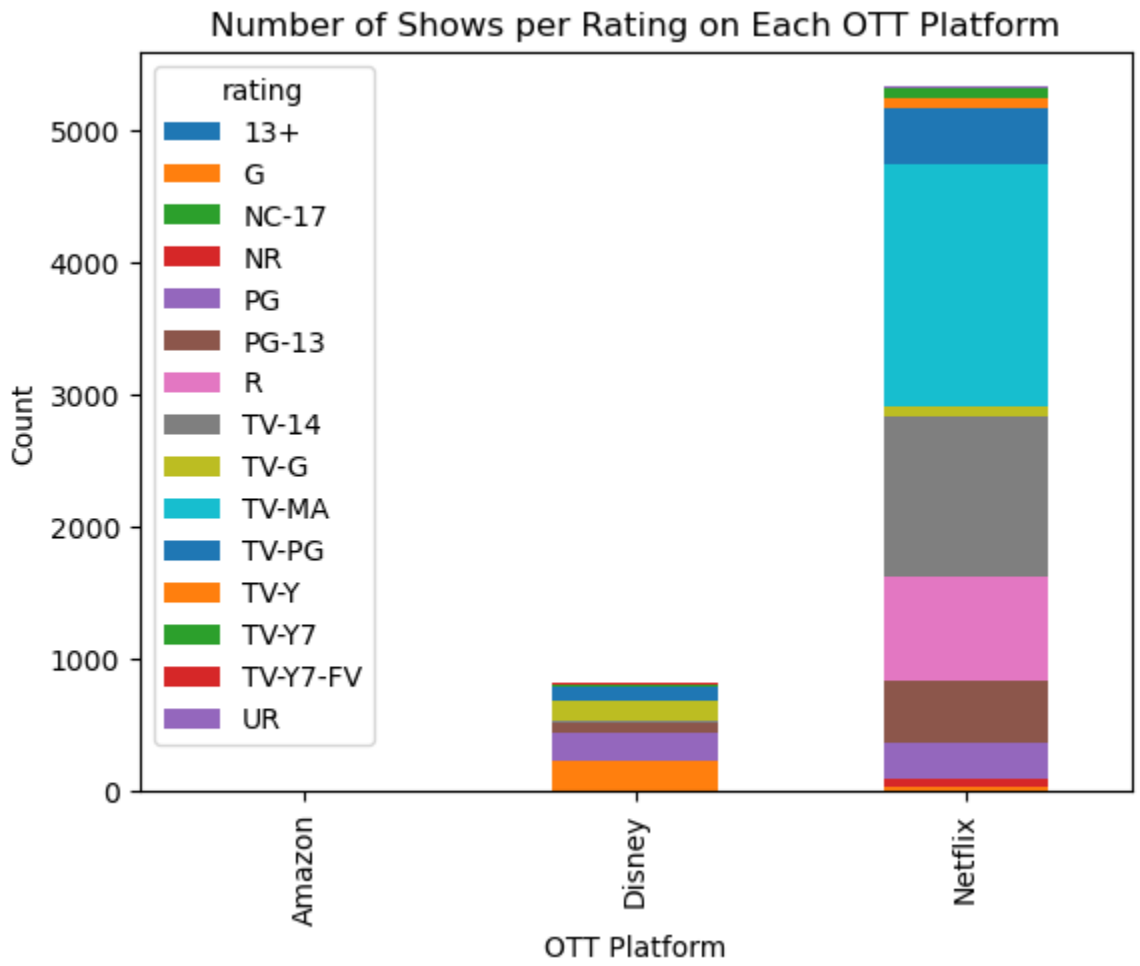


10. Number of shows per rating on each OTT platform

In [40]:
```
platform_rating_distribution = df.groupby(['OTT Platform', 'rating']).size().unstack().f
platform_rating_distribution.plot(kind='bar', stacked=True, title='Number of Shows per R
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_15308\898644889.py:1: FutureWarning: The defa
ult of observed=False is deprecated and will be changed to True in a future version of p
andas. Pass observed=False to retain current behavior or observed=True to adopt the futu
re default and silence this warning.
  platform_rating_distribution = df.groupby(['OTT Platform', 'rating']).size().unstack
().fillna(0)
```

Out[40]:
```
<Axes: title={'center': 'Number of Shows per Rating on Each OTT Platform'}, xlabel='OTT
Platform', ylabel='Count'>
```

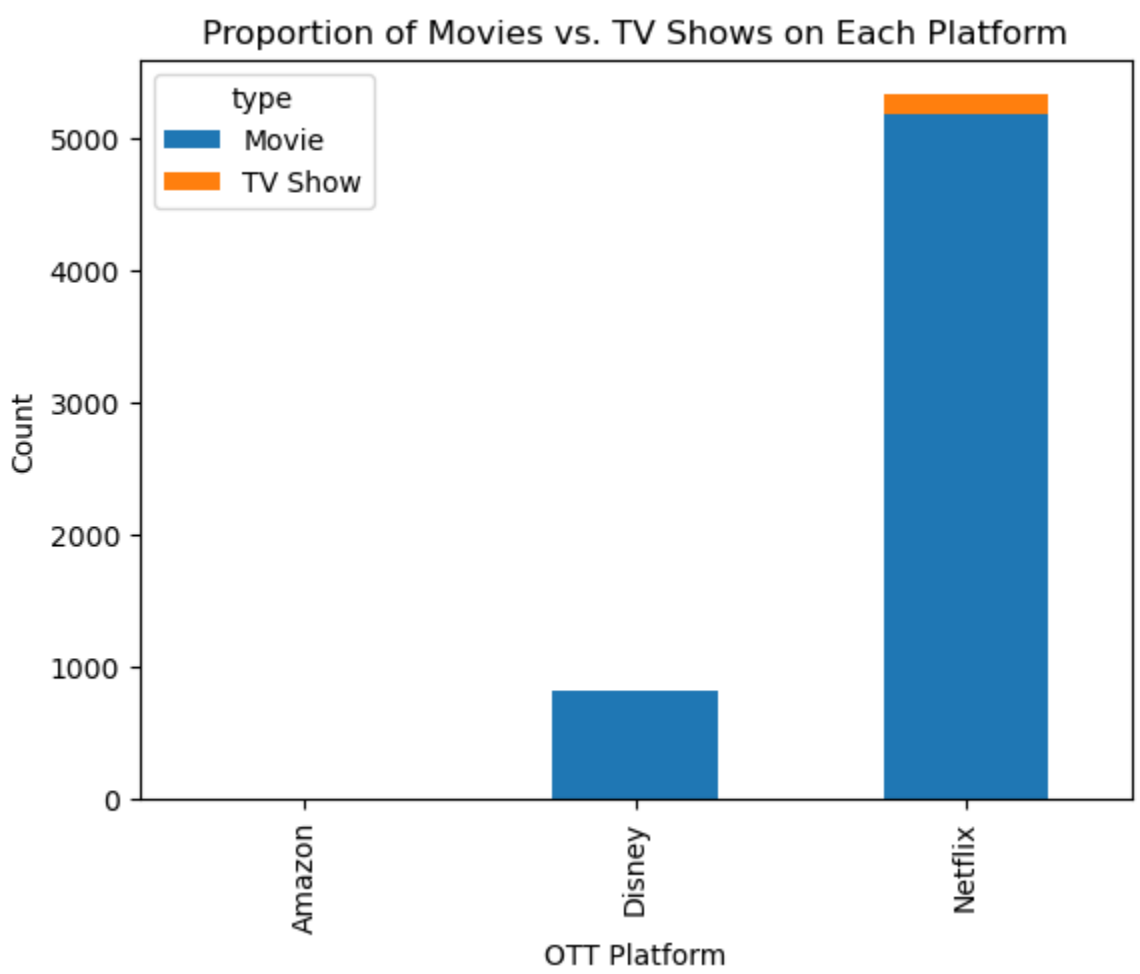


11. Proportion of Movies V/S TV shows on each platform

In [41]:
```
platform_type_distribution = df.groupby(['OTT Platform', 'type']).size().unstack().filln
platform_type_distribution.plot(kind='bar', stacked=True, title='Proportion of Movies vs
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_15308\1209155936.py:1: FutureWarning: The def
ault of observed=False is deprecated and will be changed to True in a future version of
pandas. Pass observed=False to retain current behavior or observed=True to adopt the fut
ure default and silence this warning.
  platform_type_distribution = df.groupby(['OTT Platform', 'type']).size().unstack().fil
lna(0)
```

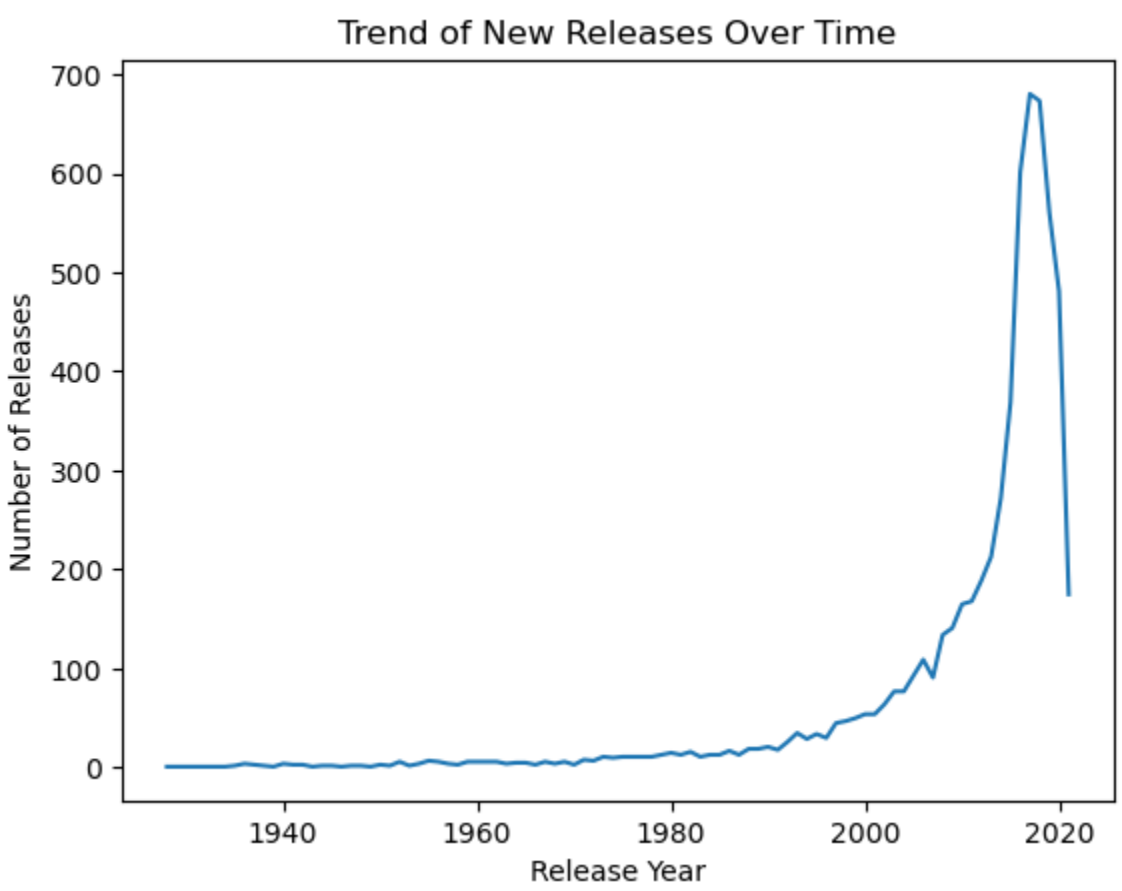
Out[41]:
```
<Axes: title={'center': 'Proportion of Movies vs. TV Shows on Each Platform'}, xlabel='O
TT Platform', ylabel='Count'>
```



12. Trend of new releases over time

In [42]:
```
new_releases_trend = df['release_year'].value_counts().sort_index()
new_releases_trend.plot(kind='line', title='Trend of New Releases Over Time', xlabel='Re
```

Out[42]:
```
<Axes: title={'center': 'Trend of New Releases Over Time'}, xlabel='Release Year', ylabe
l='Number of Releases'>
```

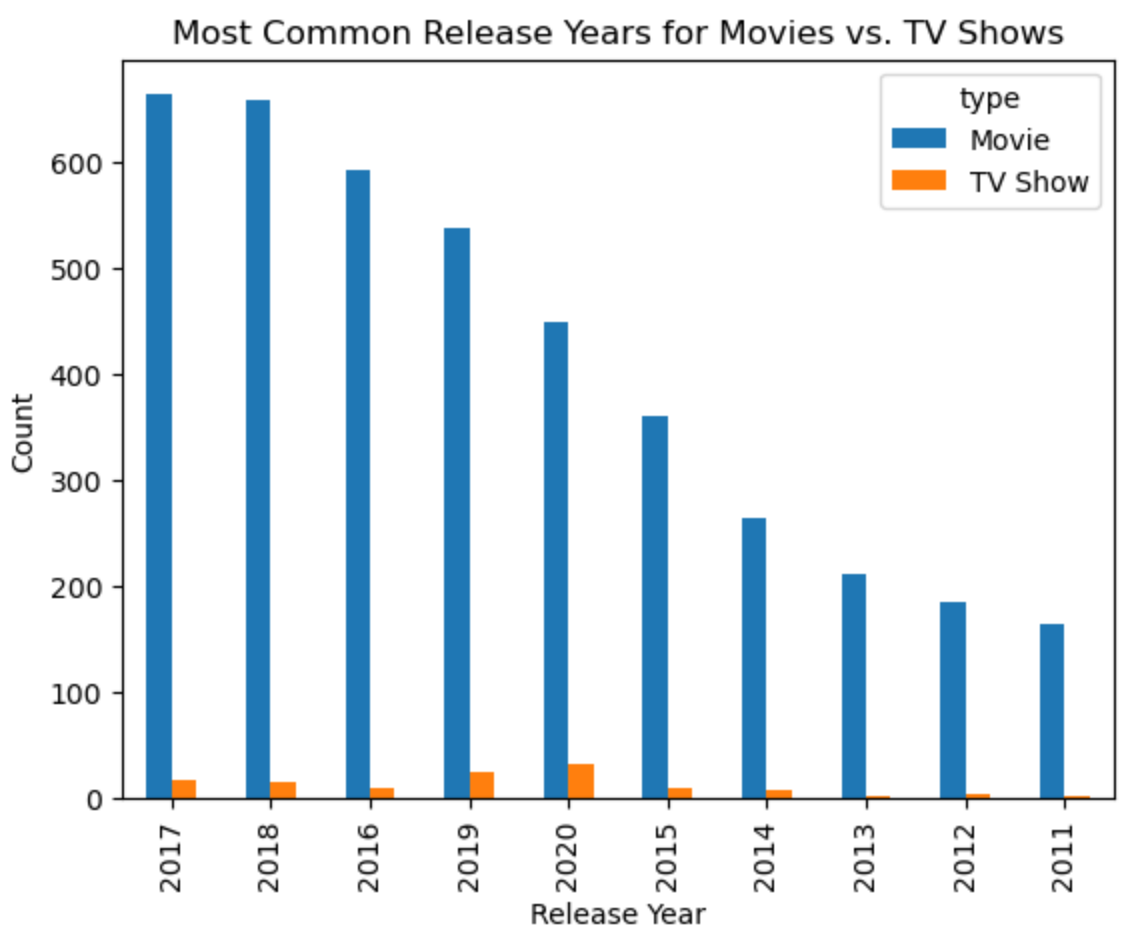


13. Most common release years for Movies V/S TV shows

In [44]:
```
common_release_years = df.groupby(['release_year', 'type']).size().unstack().fillna(0).s
common_release_years.plot(kind='bar', title='Most Common Release Years for Movies vs. TV
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_15308\2477785207.py:1: FutureWarning: The def
ault of observed=False is deprecated and will be changed to True in a future version of
pandas. Pass observed=False to retain current behavior or observed=True to adopt the fut
ure default and silence this warning.
  common_release_years = df.groupby(['release_year', 'type']).size().unstack().fillna
(0).sort_values(by='Movie', ascending=False).head(10)
```

Out[44]:
```
<Axes: title={'center': 'Most Common Release Years for Movies vs. TV Shows'}, xlabel='Re
lease Year', ylabel='Count'>
```

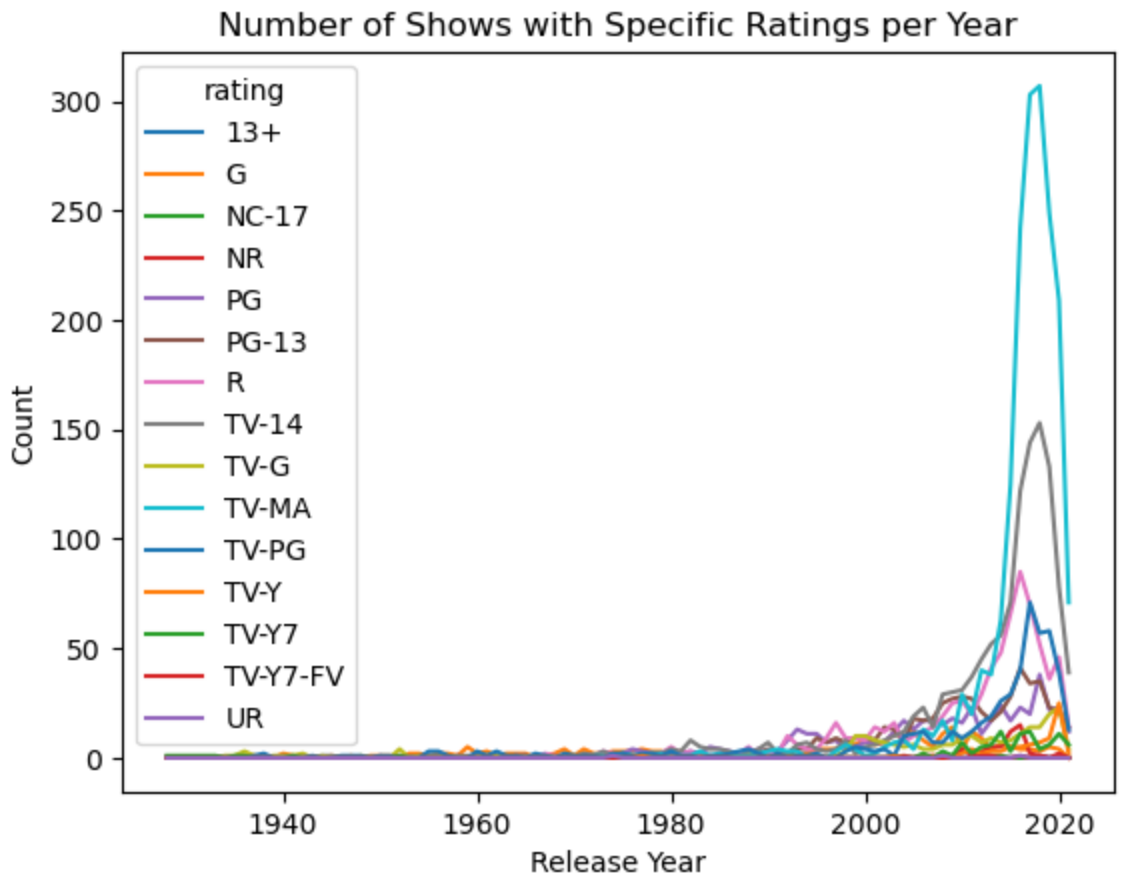


14. Number of shows with specific ratings per year

In [45]:
```
ratings_per_year = df.groupby(['release_year', 'rating']).size().unstack().fillna(0)
ratings_per_year.plot(kind='line', title='Number of Shows with Specific Ratings per Year
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_15308\215686873.py:1: FutureWarning: The defa
ult of observed=False is deprecated and will be changed to True in a future version of p
andas. Pass observed=False to retain current behavior or observed=True to adopt the futu
re default and silence this warning.
  ratings_per_year = df.groupby(['release_year', 'rating']).size().unstack().fillna(0)
```

Out[45]:
```
<Axes: title={'center': 'Number of Shows with Specific Ratings per Year'}, xlabel='Relea
se Year', ylabel='Count'>
```

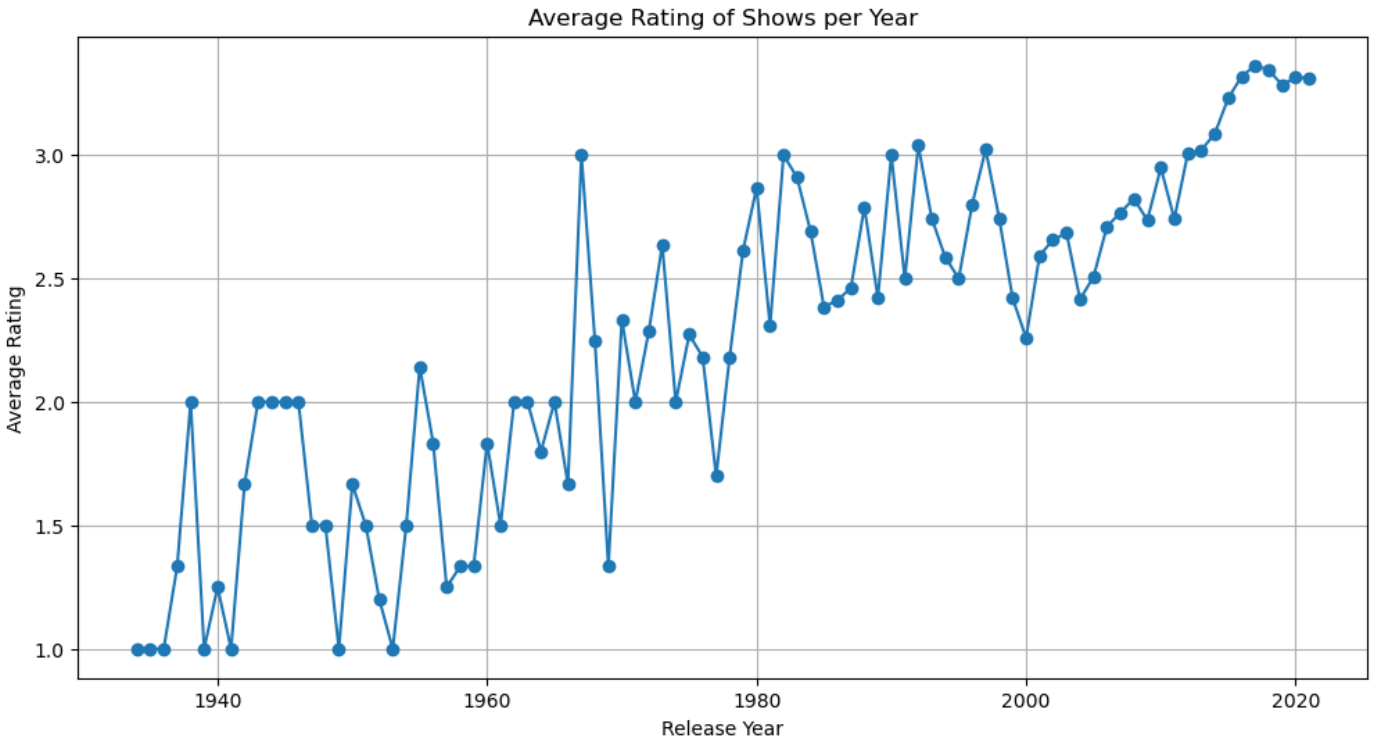


15. Average rating of shows per year

In [46]:

```
# Define a mapping for the ratings (example mapping)
rating_map = {
    'G': 1,
    'TV-G': 1,
    'PG': 2,
    'TV-PG': 2,
    'PG-13': 3,
    'TV-14': 3,
    'R': 4,
    'TV-MA': 4,
    'NC-17': 5,
    'NR'  : 0,
    'UR' : 0,

}

# Convert 'rating' using the mapping
df['numeric_rating'] = df['rating'].map(rating_map)

# Drop rows with NaN values in 'numeric_rating' or 'release_year'
df_clean = df.dropna(subset=['numeric_rating', 'release_year'])

# Calculate the average rating per year manually
average_rating_per_year = df_clean.groupby('release_year')['numeric_rating'].mean().rese

# Plot the results using matplotlib
plt.figure(figsize=(12, 6))
plt.plot(average_rating_per_year['release_year'], average_rating_per_year['numeric_ratin
plt.title('Average Rating of Shows per Year')
plt.xlabel('Release Year')
plt.ylabel('Average Rating')
```

```
plt.grid(True)
plt.show()
```



Conclusion

1. People mostly prefer watching movies than TV shows
2. More shows were added between 2018-2020 especially during lockdown
3. The people of United States have been wathcing more movies compared to other countries
4. Directors like Raul Campos and Jan Suter have been more influencial with their work than others
5. Netflix has the most no. of shows
6. International movies is the most watched category
7. Most no. of shows were added in November
8. All the platforms majorly have movies more than TV shows
9. There were more new releases in the beginning of 2020
10. A lot of movies were released in 2017
11. The movie ratings have certainly increased in the past two decades