In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data=pd.read_csv(r'C:\Users\Vallabh Kade\Downloads\amazon_prime_titles.csv\amazon_p
```

In [3]:
```
data.head()
```

Out[3]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	du
0	s1	Movie	The Grand Seduction	Don McKellar	Brendan Gleeson, Taylor Kitsch, Gordon Pinsent	Canada	March 30, 2021	2014	NaN	1
1	s2	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar	India	March 30, 2021	2018	13+	1
2	s3	Movie	Secrets of Deception	Josh Webber	Tom Sizemore, Lorenzo Lamas, Robert LaSardo, R...	United States	March 30, 2021	2017	NaN	
3	s4	Movie	Pink: Staying True	Sonia Anderson	Interviews with: Pink, Adele, Beyoncé, Britney...	United States	March 30, 2021	2014	NaN	
4	s5	Movie	Monster Maker	Giles Foster	Harry Dean Stanton, Kieran O'Brien, George Cos...	United Kingdom	March 30, 2021	1989	NaN	

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

Out[4]:
```
(9668, 12)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9668 entries, 0 to 9667
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       9668 non-null   object
 1   type          9668 non-null   object
 2   title         9668 non-null   object
 3   director      7586 non-null   object
 4   cast          8435 non-null   object
 5   country       672 non-null    object
 6   date_added    155 non-null    object
 7   release_year  9668 non-null   int64 
 8   rating        9331 non-null   object
 9   duration      9668 non-null   object
 10  listed_in     9668 non-null   object
 11  description   9668 non-null   object
dtypes: int64(1), object(11)
memory usage: 906.5+ KB
```

In [6]:
```
#Check Missing Values
data.isna().sum()
```

Out[6]:
```
show_id            0
type               0
title              0
director        2082
cast            1233
country         8996
date_added      9513
release_year       0
rating           337
duration           0
listed_in          0
description        0
dtype: int64
```

In [7]:
```
data.duplicated().sum()
```

Out[7]:
```
0
```

In [8]:
```
#Missing Values Handling
# show percentage of missing values
missing_values = data.isnull().sum()
total_values = data.shape[0]

# percentage of missing values
percentage_missing = round((missing_values / total_values) * 100, 2)

print(percentage_missing)
```

```
show_id          0.00
type             0.00
title            0.00
director        21.53
cast            12.75
country         93.05
date_added      98.40
release_year     0.00
rating           3.49
duration         0.00
listed_in        0.00
description      0.00
dtype: float64
```

In [9]:
```
# dropping 'cast', 'country', and 'date_added' fields
data.drop(['cast','country', 'date_added'], axis=1, inplace=True)
```

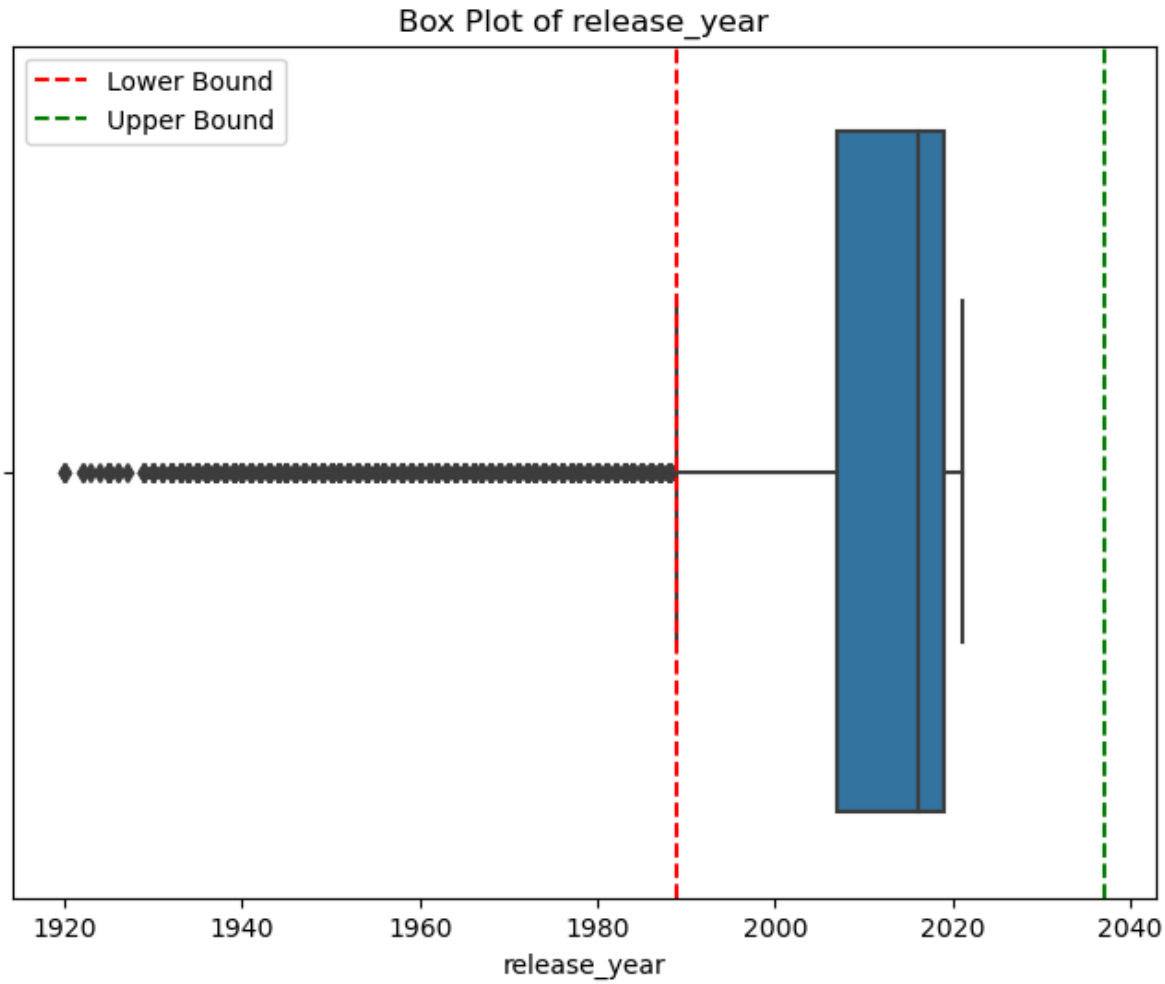
In [10]:
```
# fill the missing value in the 'director' column with 'Unknown'
data['director'].fillna('Unknown', inplace=True)
```

In [11]:
```
# fill the missing value in the 'director' column with 'NR' (Not Rated)
data['rating'].fillna('NR', inplace=True)
```

In [12]:
```
#Check Data After Handling Missing Values
data.isna().sum()
```

Out[12]:
```
show_id         0
type            0
title           0
director        0
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64
```

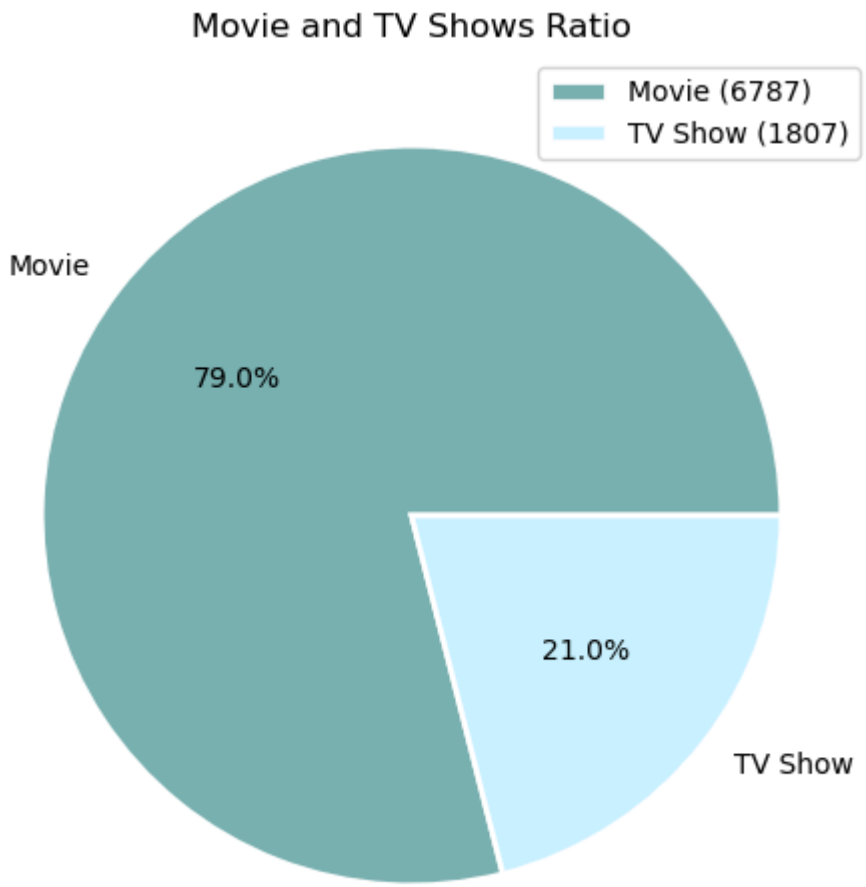
In [13]:
```
#Detecting & Handling Outliers

# detecting outliers using bloxplot method
for col in data.columns:
    if data[col].dtype != 'object':
        plt.figure(figsize=(8, 6))
        sns.boxplot(x=data[col])

        # calculate the first quartile (Q1) and third quartile (Q3)
        Q1 = data[col].quantile(0.25)
        Q3 = data[col].quantile(0.75)

        # calculate the interquartile range (IQR)
        IQR = Q3 - Q1

        # calculate the lower bound and upper bound
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Show lower and upper bounds on the plot
        plt.axvline(x=lower_bound, color='r', linestyle='--', label='Lower Bound')
        plt.axvline(x=upper_bound, color='g', linestyle='--', label='Upper Bound')

        plt.title(f'Box Plot of {col}')
        plt.legend()
        plt.show()

print(f'Lower Bound: {lower_bound}')
print(f'Upper Bound: {upper_bound}')
```



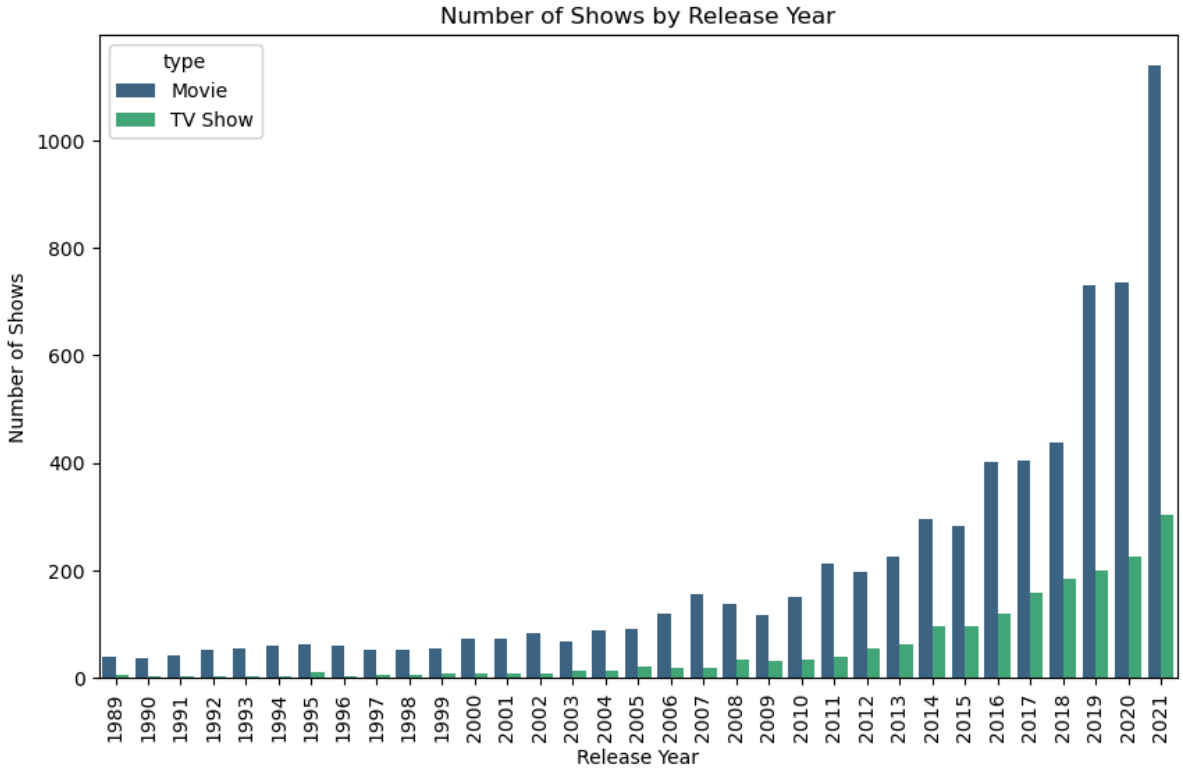

```
Lower Bound: 1989.0
Upper Bound: 2037.0
```

In [14]:
```
# Handling Outlier
filtered_data = data[(data['release_year'] >= lower_bound) & (data['release_year']
data_cleaned = filtered_data.copy()
```

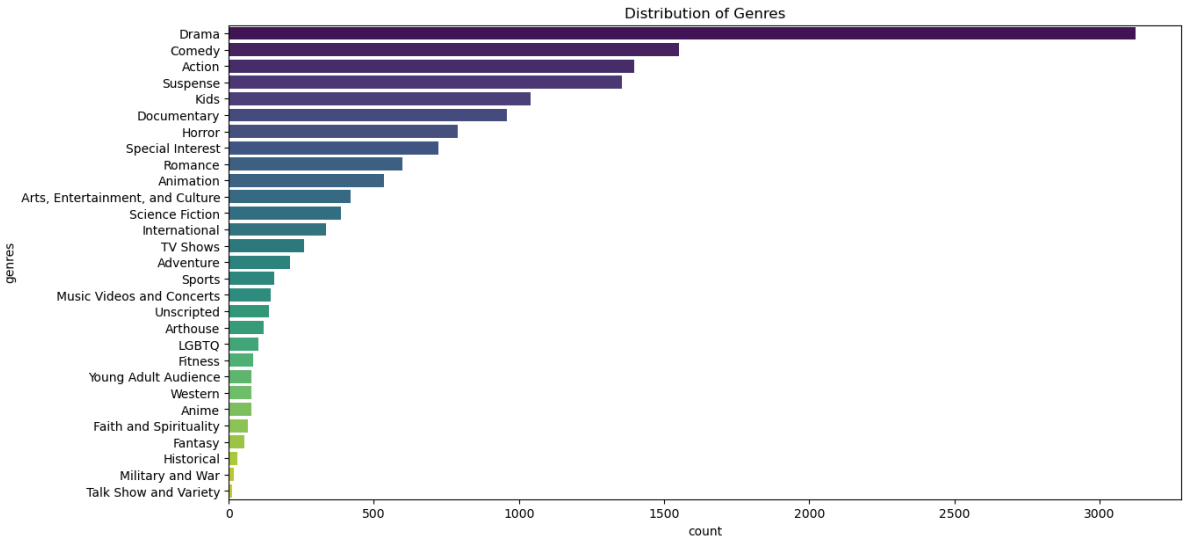
In [15]:
```
#Exploratory Data Analysis (EDA)

# counts the number of occurrences of each value in the 'type' column
type_counts = data_cleaned['type'].value_counts()

# plot pie chart
plt.figure(figsize = (6,6))
plt.pie(
    x= type_counts.values,labels = type_counts.index,autopct = '%.1f%%',
    wedgeprops = {'linewidth': 2.0,'edgecolor': 'white'},
    colors = ['#7AB2B2', '#CAF4FF']
    )
plt.title('Movie and TV Shows Ratio')
plt.legend(labels = [f"{label} ({count})" for label, count in zip(type_counts.index
plt.show()
```



In [16]:
```
# Count the number of shows by release year
release_year_counts = data_cleaned['release_year'].value_counts().sort_index()

# Ploting
plt.figure(figsize = (10, 6))
sns.countplot(x ='release_year', data = data_cleaned, hue = 'type', order = release
plt.xticks(rotation = 90)
plt.title('Number of Shows by Release Year')
plt.xlabel('Release Year')
plt.ylabel('Number of Shows')
plt.show()
```

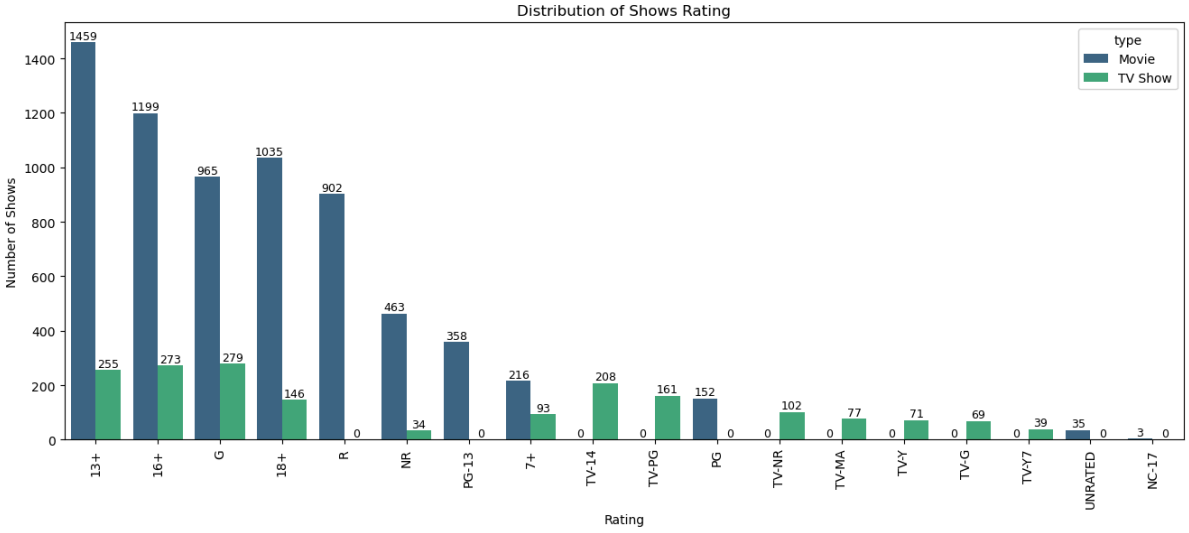


In [17]:
```
# Distribution of genres
# Splitting the genres
def extract_genres(listed_in):
    if 'Arts, Entertainment, and Culture' in listed_in:
        return ['Arts, Entertainment, and Culture']
    else:
        return [genre.strip() for genre in listed_in.split(',')]

data_cleaned['genres'] = data_cleaned['listed_in'].apply(extract_genres)

# Exploding the genres into separate rows
genres_exploded = data_cleaned.explode('genres')

# Plotting the distribution of genres
plt.figure(figsize = (14, 7))
sns.countplot(data = genres_exploded, y = 'genres', order = genres_exploded['genres
plt.title('Distribution of Genres')
plt.show()
```



In [18]:
```
data_cleaned['rating'] = data_cleaned['rating'].replace({
    'ALL': 'G',
```

```
    'AGES_18_': '18+',
    'AGES_16_': '16+',
    '16' : '16+',
    'ALL_AGES': 'G',
    'NOT_RATE': 'UNRATED'
})

# Plotting the distribution of Shows Rating
plt.figure(figsize = (16,6))
sns.countplot(x='rating', data = data_cleaned, hue = 'type', order = data_cleaned['
plt.xticks(rotation = 90)
plt.title('Distribution of Shows Rating')
plt.xlabel('Rating')
plt.ylabel('Number of Shows')
ax = plt.gca()
for p in ax.patches:
    ax.annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_hei
                ha='center', va='center', fontsize=9, color='black', xytext=(0, 5),
                textcoords='offset points')
plt.show()
```



In [ ]:
```
#Thanks for visiting my profile - Vallabh Kade
```