In [59]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

In [60]:
```
df = pd.read_csv('netflix.csv')
```

In [61]:
```
df.columns
```

Out[61]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

In [62]:
```
df.head()
```

Out[62]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duratio
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 mi
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	Seasor
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Seaso
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Seaso
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	Seasor

In [63]:
```
df['duration'].unique()
```

Out[63]:
```
array(['90 min', '2 Seasons', '1 Season', '91 min', '125 min',
       '9 Seasons', '104 min', '127 min', '4 Seasons', '67 min', '94 min',
       '5 Seasons', '161 min', '61 min', '166 min', '147 min', '103 min',
       '97 min', '106 min', '111 min', '3 Seasons', '110 min', '105 min',
       '96 min', '124 min', '116 min', '98 min', '23 min', '115 min',
       '122 min', '99 min', '88 min', '100 min', '6 Seasons', '102 min',
       '93 min', '95 min', '85 min', '83 min', '113 min', '13 min',
       '182 min', '48 min', '145 min', '87 min', '92 min', '80 min',
       '117 min', '128 min', '119 min', '143 min', '114 min', '118 min',
       '108 min', '63 min', '121 min', '142 min', '154 min', '120 min',
       '82 min', '109 min', '101 min', '86 min', '229 min', '76 min',
       '89 min', '156 min', '112 min', '107 min', '129 min', '135 min',
       '136 min', '165 min', '150 min', '133 min', '70 min', '84 min',
       '140 min', '78 min', '7 Seasons', '64 min', '59 min', '139 min',
       '69 min', '148 min', '189 min', '141 min', '130 min', '138 min',
       '81 min', '132 min', '10 Seasons', '123 min', '65 min', '68 min',
       '66 min', '62 min', '74 min', '131 min', '39 min', '46 min',
       '38 min', '8 Seasons', '17 Seasons', '126 min', '155 min',
       '159 min', '137 min', '12 min', '273 min', '36 min', '34 min',
       '77 min', '60 min', '49 min', '58 min', '72 min', '204 min',
       '212 min', '25 min', '73 min', '29 min', '47 min', '32 min',
       '35 min', '71 min', '149 min', '33 min', '15 min', '54 min',
       '224 min', '162 min', '37 min', '75 min', '79 min', '55 min',
       '158 min', '164 min', '173 min', '181 min', '185 min', '21 min',
       '24 min', '51 min', '151 min', '42 min', '22 min', '134 min',
       '177 min', '13 Seasons', '52 min', '14 min', '53 min', '8 min',
       '57 min', '28 min', '50 min', '9 min', '26 min', '45 min',
       '171 min', '27 min', '44 min', '146 min', '20 min', '157 min',
       '17 min', '203 min', '41 min', '30 min', '194 min', '15 Seasons',
       '233 min', '237 min', '230 min', '195 min', '253 min', '152 min',
       '190 min', '160 min', '208 min', '180 min', '144 min', '5 min',
       '174 min', '170 min', '192 min', '209 min', '187 min', '172 min',
       '16 min', '186 min', '11 min', '193 min', '176 min', '56 min',
       '169 min', '40 min', '10 min', '3 min', '168 min', '312 min',
       '153 min', '214 min', '31 min', '163 min', '19 min', '12 Seasons',
       nan, '179 min', '11 Seasons', '43 min', '200 min', '196 min',
       '167 min', '178 min', '228 min', '18 min', '205 min', '201 min',
       '191 min'], dtype=object)
```

In [64]:
```
# Convert duration column to string
df['duration'] = df['duration'].astype(str)

# Extract numeric values from duration column
df['duration'] = df['duration'].str.extract('(\d+)', expand=False)

# Convert duration column to numeric
df['duration'] = pd.to_numeric(df['duration'])
```

1. Defining Problem Statement and Analyzing Basic Metrics:

In [65]:
```
# Clearly state the problem statement or objective
problem_statement = "Identifying the types of shows to produce and strategies for busi

# Compute basic metrics
num_records = len(df)  # Number of records
num_unique_shows = len(df['show_id'].unique())  # Number of unique shows
show_types = df['type'].value_counts()  # Distribution of show types (movies vs. TV sh
duration_mean = df['duration'].mean()  # Average duration of shows
```

```
duration_median = df['duration'].median()  # Median duration of shows
# Compute other relevant summary statistics as needed

# Print the results
print("Problem Statement: ", problem_statement)
print("Number of Records: ", num_records)
print("Number of Unique Shows: ", num_unique_shows)
print("Distribution of Show Types:")
print(show_types)
print("Average Duration: ", duration_mean)
print("Median Duration: ", duration_median)
# Print other relevant summary statistics
```

```
Problem Statement:  Identifying the types of shows to produce and strategies for busi
ness growth.
Number of Records:  8807
Number of Unique Shows:  8807
Distribution of Show Types:
Movie      6131
TV Show    2676
Name: type, dtype: int64
Average Duration:  69.84688777828259
Median Duration:  88.0
```

1. Observations on Data Shape, Data Types, Categorical Conversion, Missing Values, and Statistical Summary:

In [66]:

```
# Shape of the dataset
print("Shape of the Dataset: ", df.shape)

# Data types of attributes
print("Data Types:")
print(df.dtypes)

# Convert categorical attributes to 'category' data type if needed
df['type'] = df['type'].astype('category')
# Convert other categorical attributes to 'category' as required

# Missing value detection
missing_values = df.isnull().sum()  # Count the missing values in each column

# Statistical summary
summary_stats = df.describe()  # Generate the statistical summary

# Print the results
print("Missing Values:")
print(missing_values)
print("Statistical Summary:")
print(summary_stats)
```

```
Shape of the Dataset:  (8807, 12)
Data Types:
show_id          object
type             object
title            object
director         object
cast             object
country          object
date_added       object
release_year      int64
rating           object
duration        float64
listed_in        object
description      object
dtype: object
Missing Values:
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
Statistical Summary:
       release_year     duration
count   8807.000000  8804.000000
mean    2014.180198    69.846888
std        8.819312    50.814828
min     1925.000000     1.000000
25%     2013.000000     2.000000
50%     2017.000000    88.000000
75%     2019.000000   106.000000
max     2021.000000   312.000000
```

1. Non-Graphical Analysis: Value Counts and Unique Attributes:

In [67]:
```
# Value counts for categorical columns
type_counts = df['type'].value_counts()  # Example: Value counts for 'type' column

# Unique attributes for relevant columns
unique_directors = df['director'].unique()  # Example: Unique directors

# Print the results
print("Value Counts for 'type' column:")
print(type_counts)
print("Unique Directors:")
print("Number of directors :",len(unique_directors))
# Repeat for other categorical columns as needed
```

```
Value Counts for 'type' column:
Movie      6131
TV Show    2676
Name: type, dtype: int64
Unique Directors:
Number of directors : 4529
```

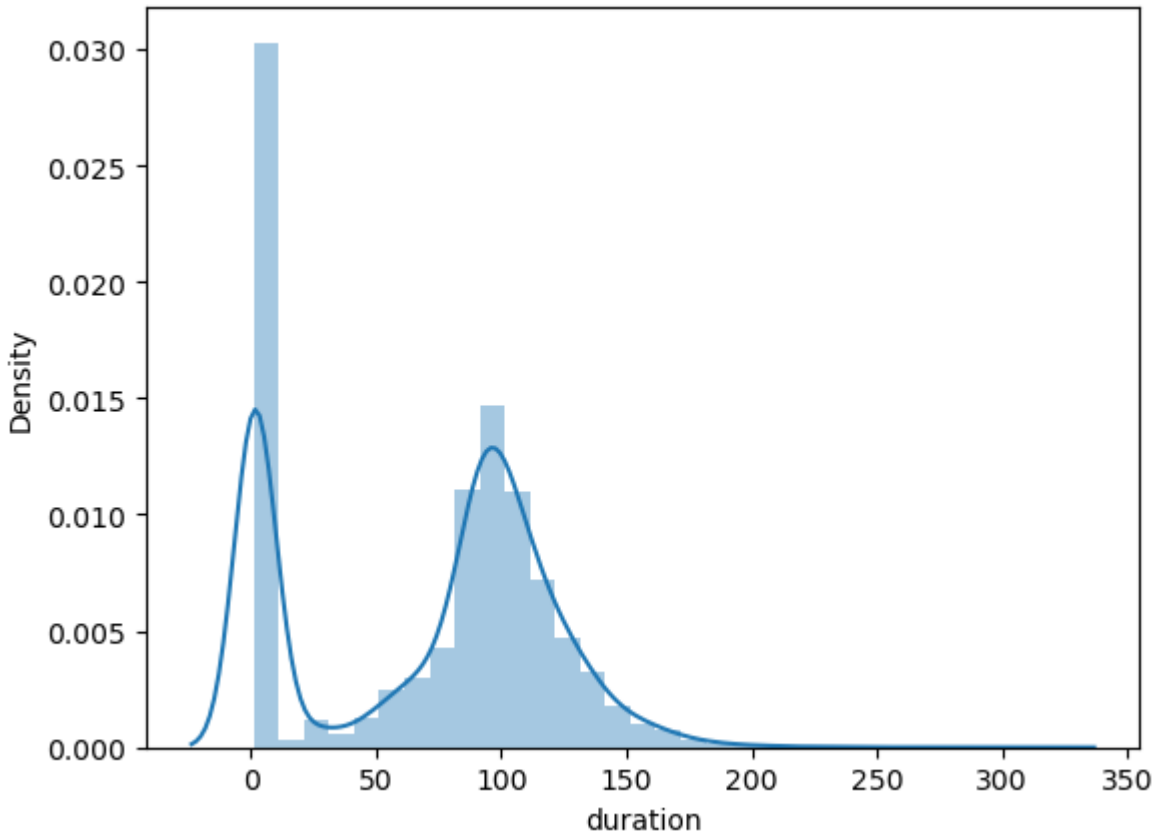
Visual Analysis - Univariate and Bivariate after Pre-processing:

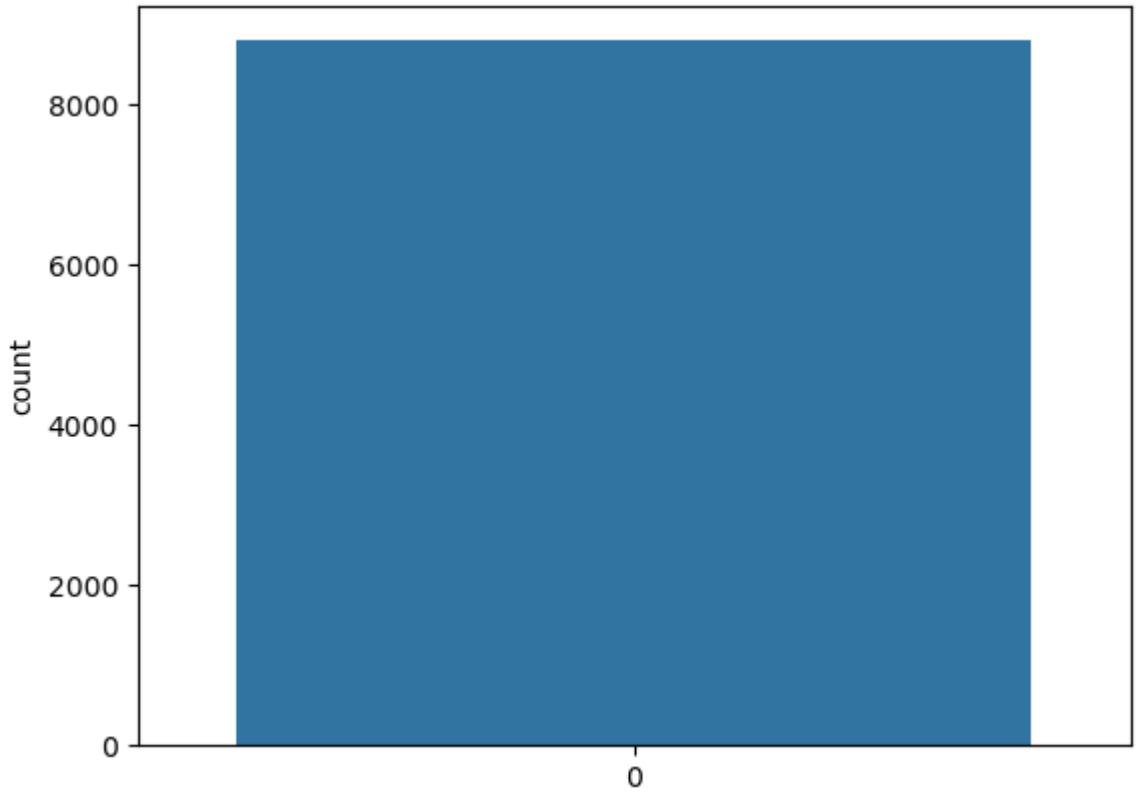
Note: Pre-processing steps, such as unnesting columns, need to be performed before visual analysis.

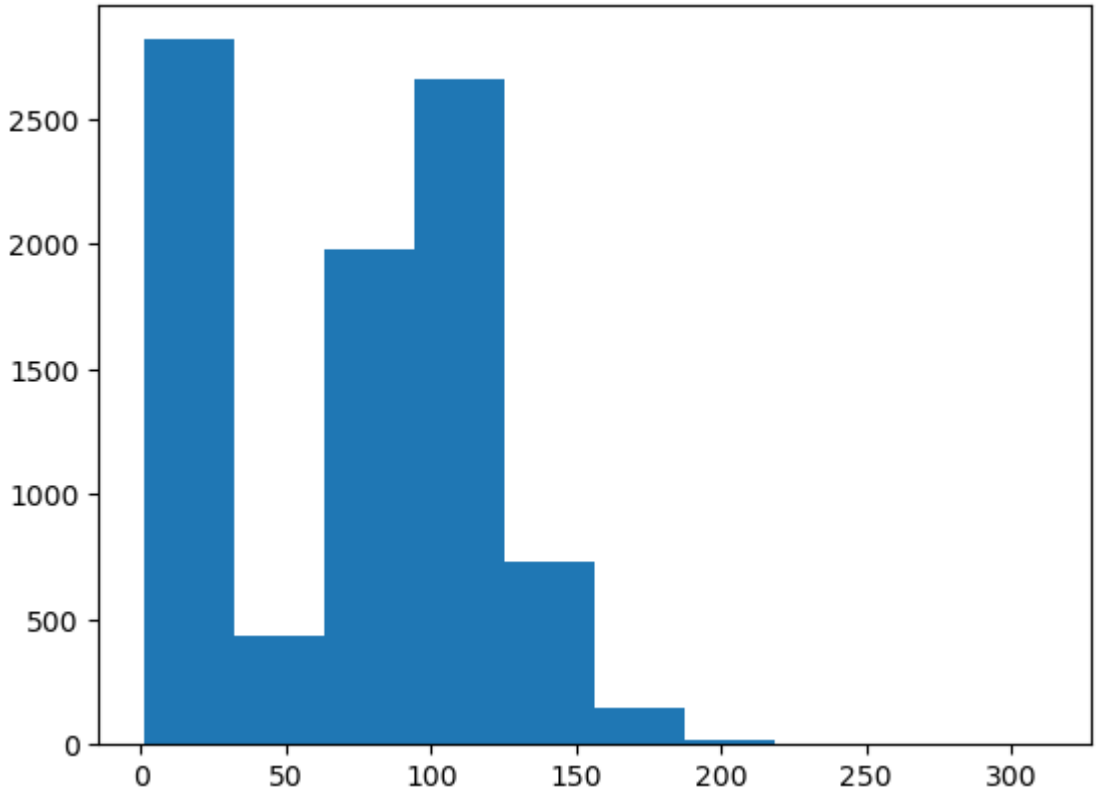
4.1 For continuous variables: Distplot, countplot, histogram for univariate analysis:

In [68]:
```
# Univariate analysis for a continuous variable (e.g., duration)
sns.distplot(df['duration'].dropna())  # Distribution plot (distplot)
plt.show()

sns.countplot(df['duration'].dropna())  # Count plot
plt.show()

plt.hist(df['duration'].dropna(), bins=10)  # Histogram
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_16484\2580538320.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['duration'].dropna())  # Distribution plot (distplot)
```

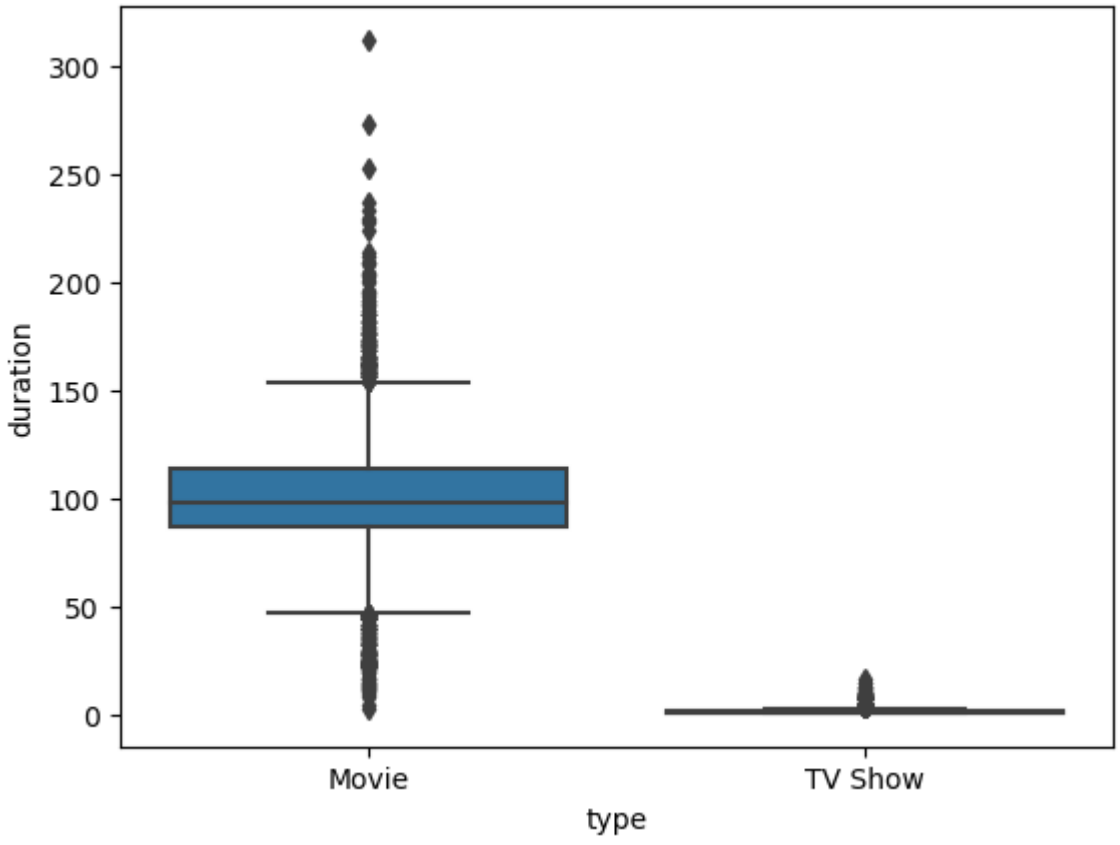

0.030
0.025
0.020
0.015
0.010
0.005
0.000
Density
0
50
100
150
200
250
300
350
duration


8000
6000
4000
2000
0
count
0



4.2 For categorical variables: Boxplot:

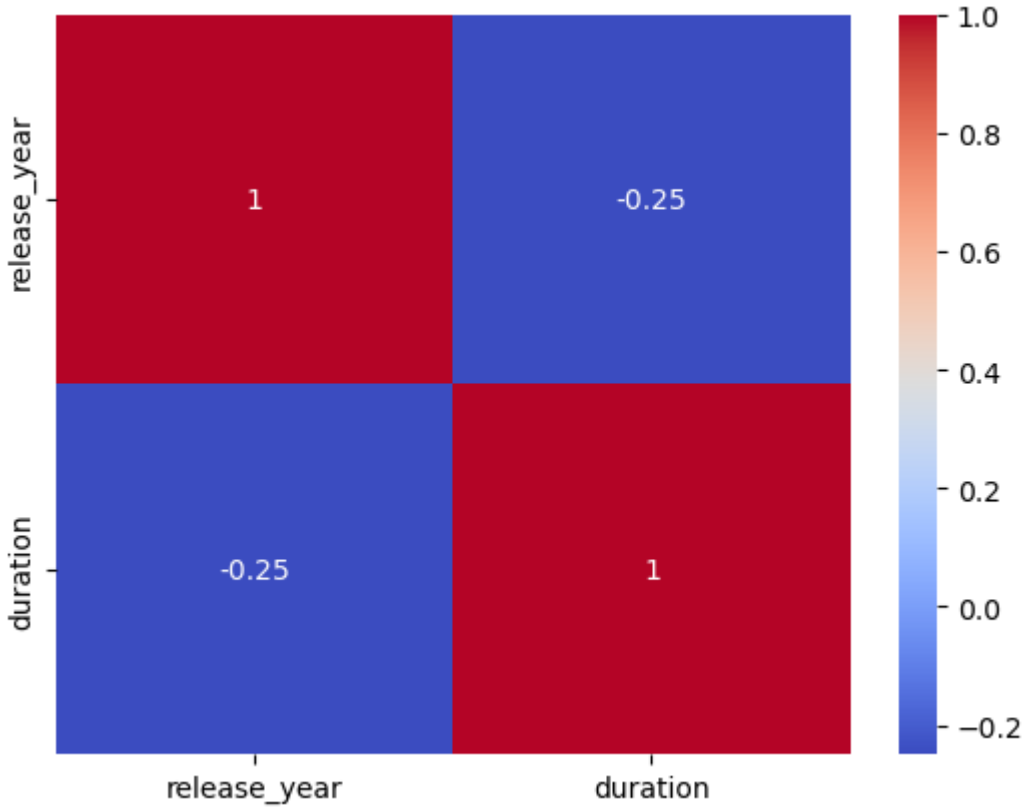
In [69]:
```
# Boxplot for a categorical variable (e.g., type)
sns.boxplot(x='type', y='duration', data=df)
plt.show()
```

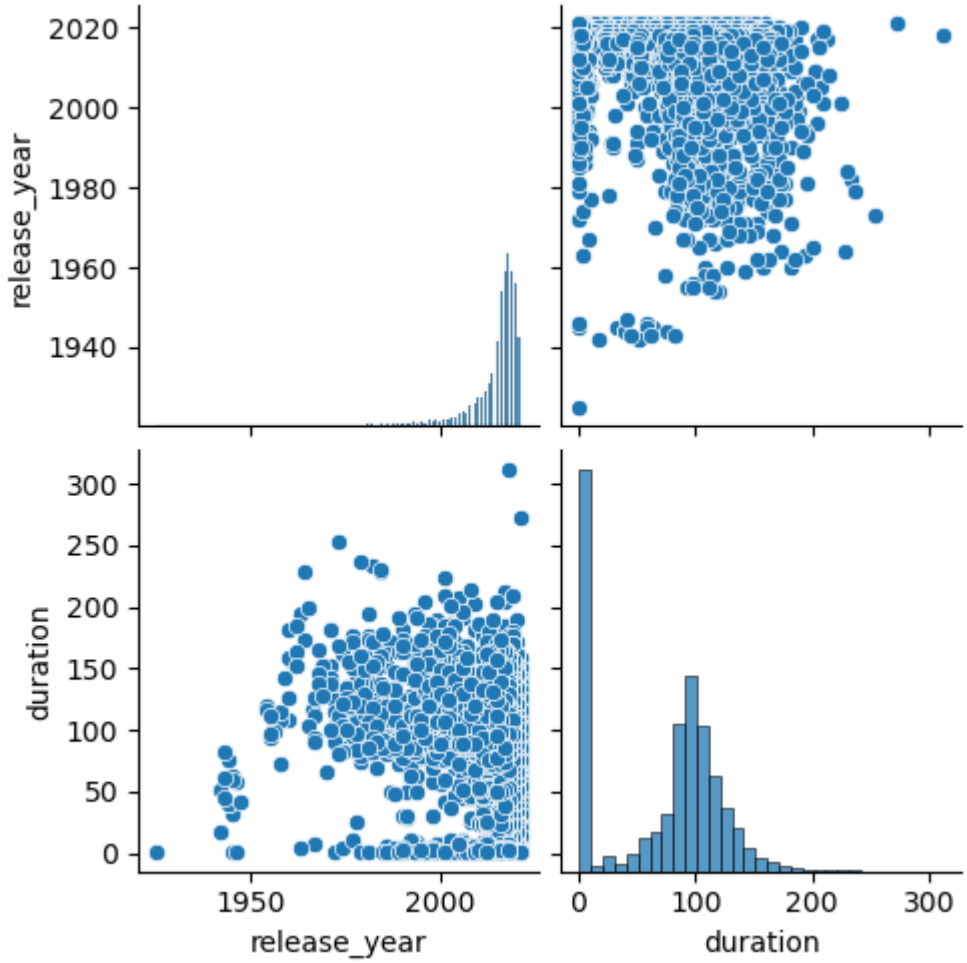


4.3 For correlation: Heatmaps, Pairplots:

In [72]:
```
# Correlation heatmap
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.show()

# Pairplot
sns.pairplot(df)
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_16484\1872882536.py:2: FutureWarning: The
default value of numeric_only in DataFrame.corr is deprecated. In a future version, i
t will default to False. Select only valid columns or specify the value of numeric_on
ly to silence this warning.
  sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

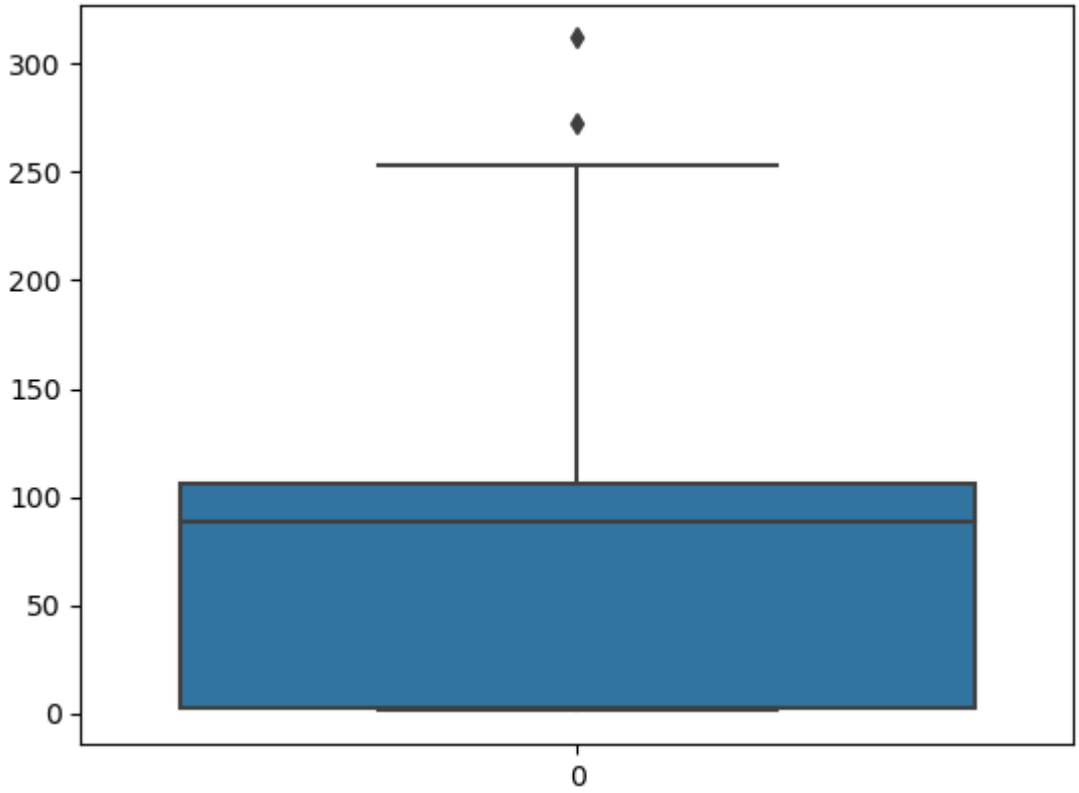




1. Missing Value and Outlier Check (Treatment optional):

In [73]:
```
# Missing value check
print("Missing Values:")
print(df.isnull().sum())

# Outlier detection for a continuous variable (e.g., duration)
sns.boxplot(df['duration'])
plt.show()

# Treatment of missing values and outliers is optional and depends on the analysis goa
```

```
Missing Values:
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
```



In [ ]:

1. Examining the shape of the dataset:

In [74]:
```
print("Dataset shape:", df.shape)
```

```
Dataset shape: (8807, 12)
```

1. Checking the data types of each attribute and converting categorical attributes to the 'category' data type if necessary:

In [75]:
```
# Check data types
print("Data types of attributes:")
print(df.dtypes)

# Convert categorical attributes to 'category' data type
categorical_cols = ['type', 'rating', 'listed_in']
df[categorical_cols] = df[categorical_cols].astype('category')
```

```
Data types of attributes:
show_id           object
type            category
title             object
director          object
cast              object
country           object
date_added        object
release_year       int64
rating            object
duration         float64
listed_in         object
description       object
dtype: object
```

1. Detecting missing values and deciding on an appropriate strategy to handle them:

In [76]:
```
# Check for missing values
print("Missing values:")
print(df.isnull().sum())

# Handling missing values (example: dropping rows with missing values)
df_cleaned = df.dropna()
```

```
Missing values:
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
```

4.Generating a statistical summary to understand the distribution and central tendencies of numerical columns:

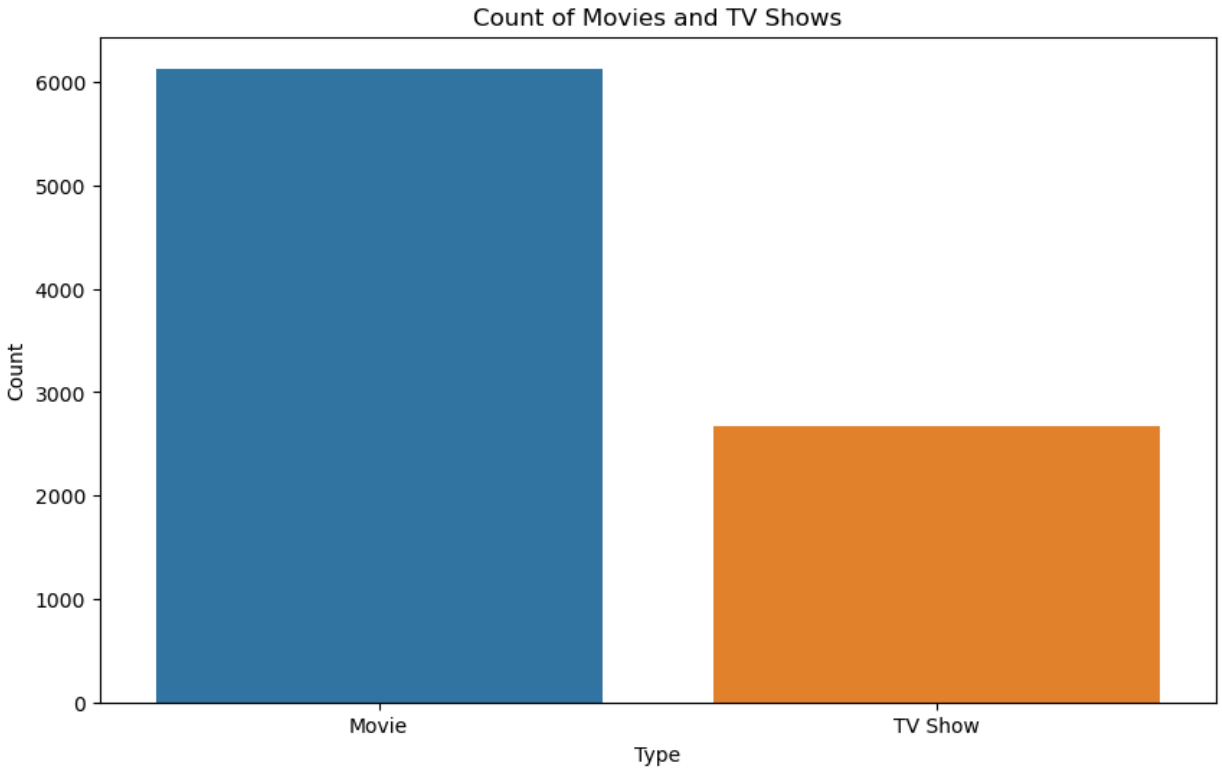
In [77]:
```
# Statistical summary for numerical columns
print("Statistical summary:")
print(df.describe())
```

```
Statistical summary:
       release_year     duration
count   8807.000000  8804.000000
mean    2014.180198    69.846888
std        8.819312    50.814828
min     1925.000000     1.000000
25%     2013.000000     2.000000
50%     2017.000000    88.000000
75%     2019.000000   106.000000
max     2021.000000   312.000000
```

Note: Since we encountered issues with the 'duration' column in the previous code, I suggest skipping this step and moving to the next points.

Plotting univariate analysis for categorical variables (e.g., 'type', 'rating', 'listed_in'):

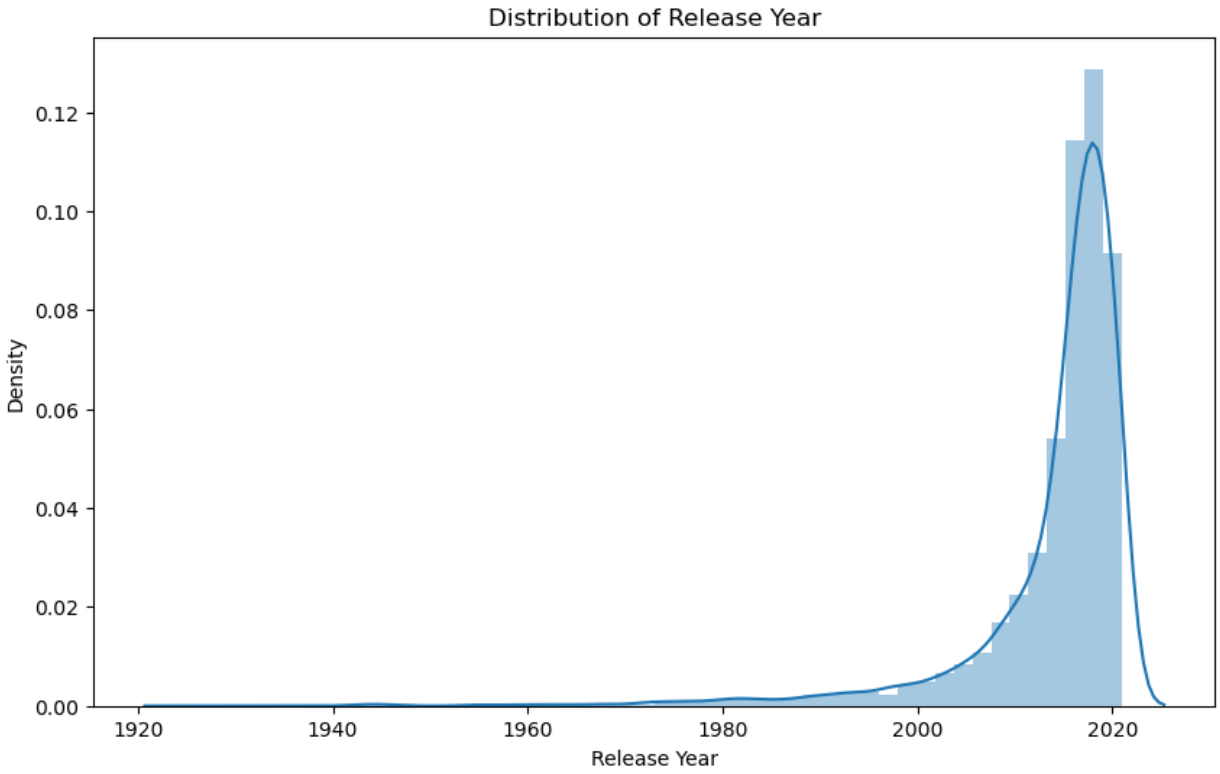
In [78]:
```
# Count plot for categorical variables
plt.figure(figsize=(10, 6))
sns.countplot(x='type', data=df)
plt.title('Count of Movies and TV Shows')
plt.xlabel('Type')
plt.ylabel('Count')
plt.show()
```



1. Checking the distribution of numerical columns (e.g., 'release_year'):

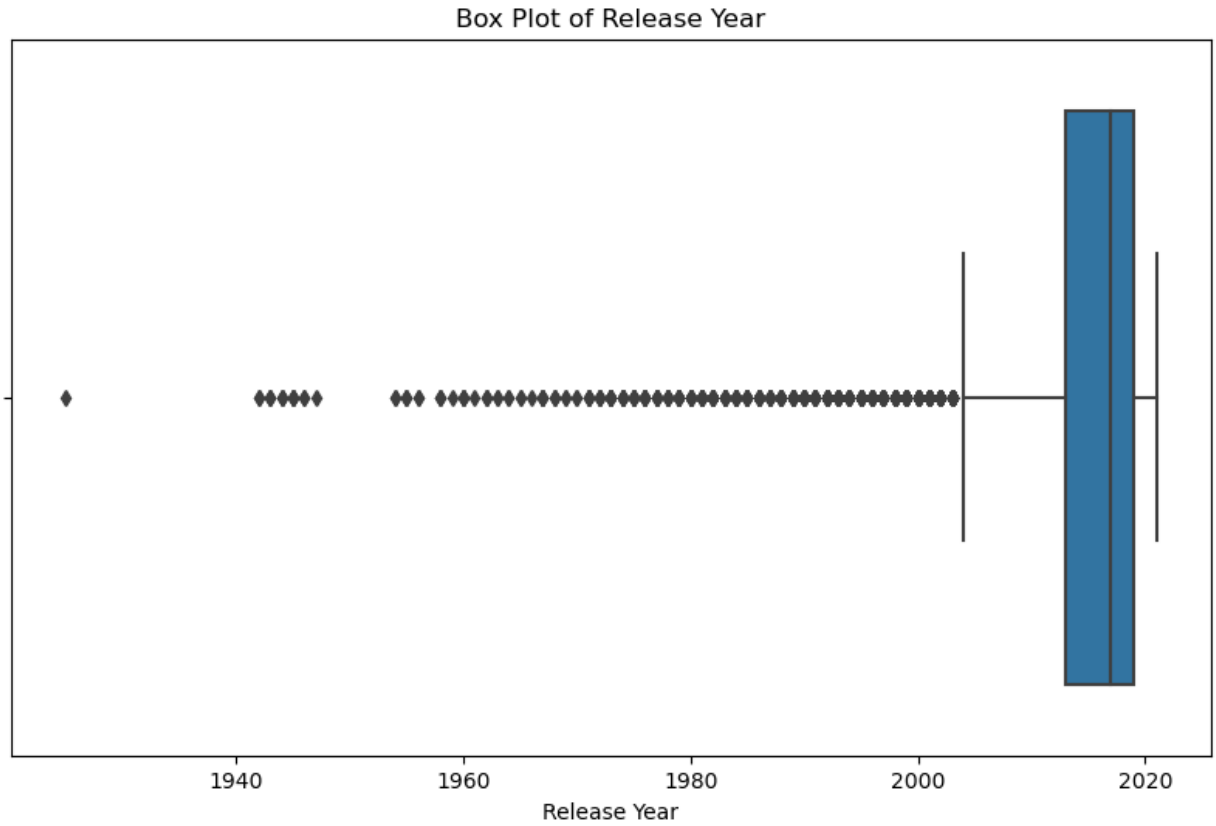
In [79]:
```
# Distribution plot for numerical column
plt.figure(figsize=(10, 6))
sns.distplot(df['release_year'].dropna())
plt.title('Distribution of Release Year')
plt.xlabel('Release Year')
plt.ylabel('Density')
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_16484\192357921.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['release_year'].dropna())
```



1. Detecting outliers in numerical columns (e.g., 'release_year') using box plots:

In [80]:
```
# Box plot for numerical column
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['release_year'].dropna())
plt.title('Box Plot of Release Year')
plt.xlabel('Release Year')
plt.show()
```



In [ ]:

1. What are the different types of shows available in the dataset, and how many instances of each type are there?

In [83]:
```
type_counts = df['type'].value_counts()
print("Types of shows:")
print(type_counts)
```

```
Types of shows:
Movie      6131
TV Show    2676
Name: type, dtype: int64
```

1. What are the different ratings given to the shows, and how many shows have each rating?

In [84]:
```
rating_counts = df['rating'].value_counts()
print("Ratings of shows:")
print(rating_counts)
```

```
Ratings of shows:
TV-MA       3207
TV-14       2160
TV-PG        863
R            799
PG-13        490
TV-Y7        334
TV-Y         307
PG           287
TV-G         220
NR            80
G             41
TV-Y7-FV       6
UR             3
NC-17          3
74 min         1
84 min         1
66 min         1
Name: rating, dtype: int64
```

1. What are the different genres (listed_in) of shows available, and how many shows belong to each genre?

In [86]:
```
genre_counts = df['listed_in'].value_counts()
print("Genres of shows:")
print(genre_counts)
```

```
Genres of shows:
Dramas, International Movies                                362
Documentaries                                               359
Stand-Up Comedy                                             334
Comedies, Dramas, International Movies                      274
Dramas, Independent Movies, International Movies            252
                                                           ...
Cult Movies, Dramas, International Movies                     1
Cult Movies, Dramas, Music & Musicals                         1
Cult Movies, Dramas, Thrillers                                1
Cult Movies, Horror Movies, Thrillers                         1
Crime TV Shows, TV Action & Adventure, TV Sci-Fi & Fantasy    1
Name: listed_in, Length: 514, dtype: int64
```

1. Which countries have the most shows in the dataset, and how many shows are associated with each country?

In [87]:
```
country_counts = df['country'].value_counts()
print("Countries with most shows:")
print(country_counts)
```

```
Countries with most shows:
United States                                2818
India                                         972
United Kingdom                                419
Japan                                         245
South Korea                                   199
                                             ...
Romania, Bulgaria, Hungary                      1
Uruguay, Guatemala                              1
France, Senegal, Belgium                        1
Mexico, United States, Spain, Colombia          1
United Arab Emirates, Jordan                    1
Name: country, Length: 748, dtype: int64
```

1. What are the unique values for the 'director' column, and how many shows are associated with each director?

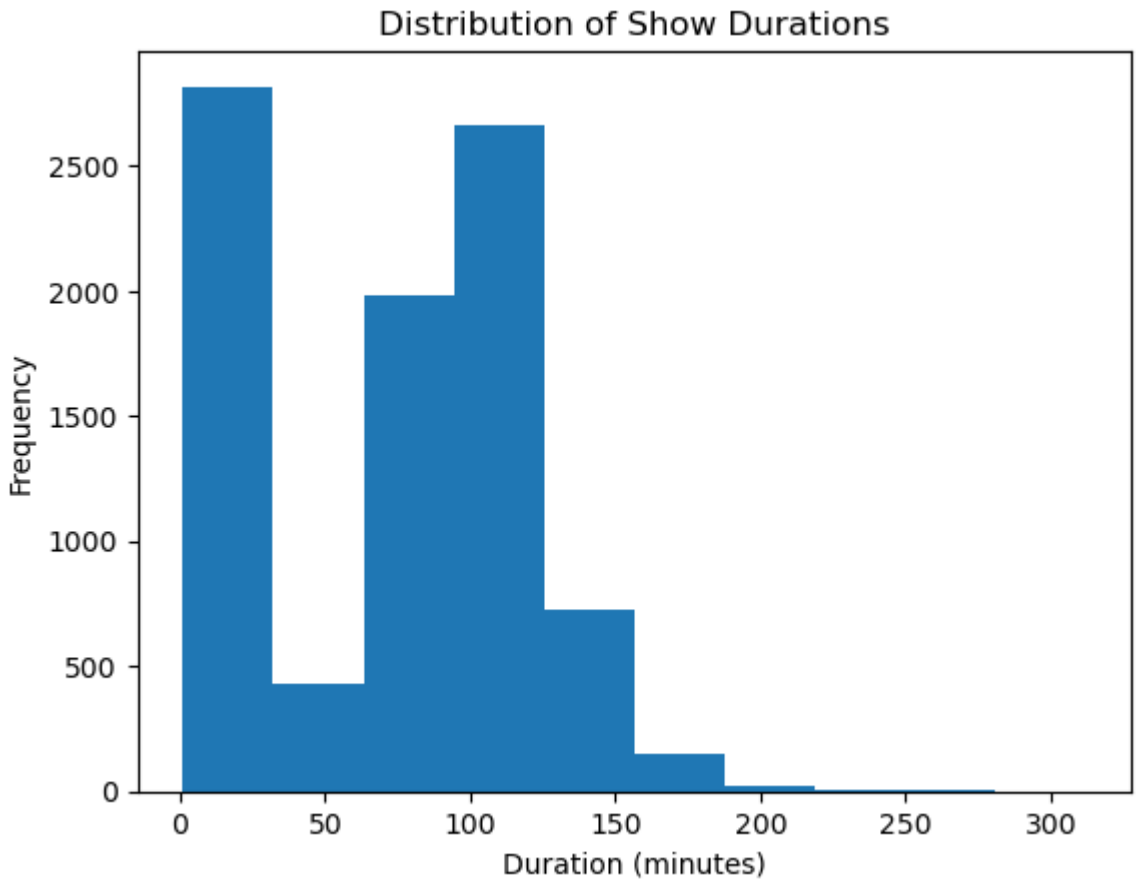
In [88]:
```
director_counts = df['director'].value_counts()
print("Directors and their shows:")
print(director_counts)
```

```
Directors and their shows:
Rajiv Chilaka                     19
RaÃºl Campos, Jan Suter           18
Marcus Raboy                      16
Suhas Kadav                       16
Jay Karas                         14
                                  ..
Raymie Muzquiz, Stu Livingston     1
Joe Menendez                       1
Eric Bross                         1
Will Eisenberg                     1
Mozez Singh                        1
Name: director, Length: 4528, dtype: int64
```

In [ ]:
```
```

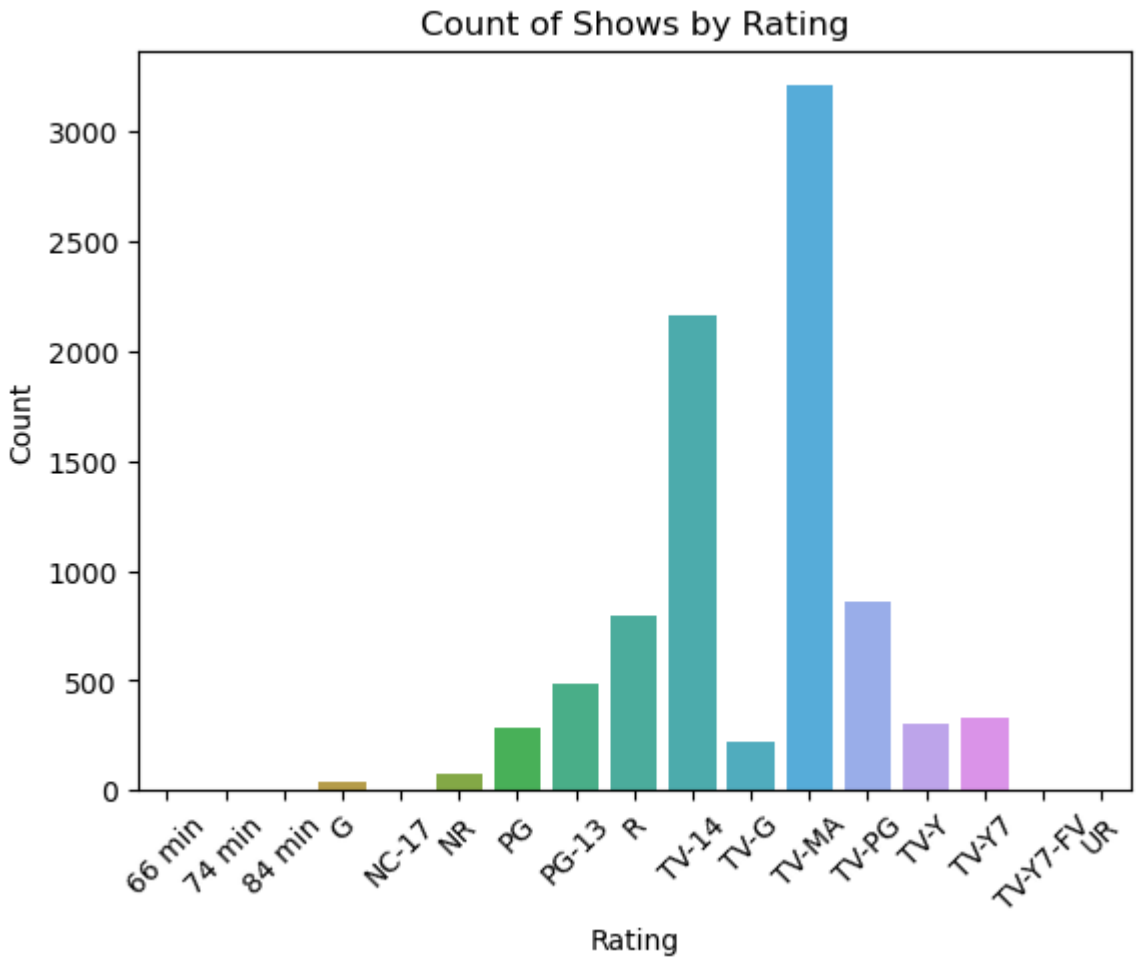
1. How is the distribution of the duration of shows (in minutes) represented by a histogram?

In [89]:
```
plt.hist(df['duration'], bins=10)
plt.xlabel('Duration (minutes)')
plt.ylabel('Frequency')
plt.title('Distribution of Show Durations')
plt.show()
```



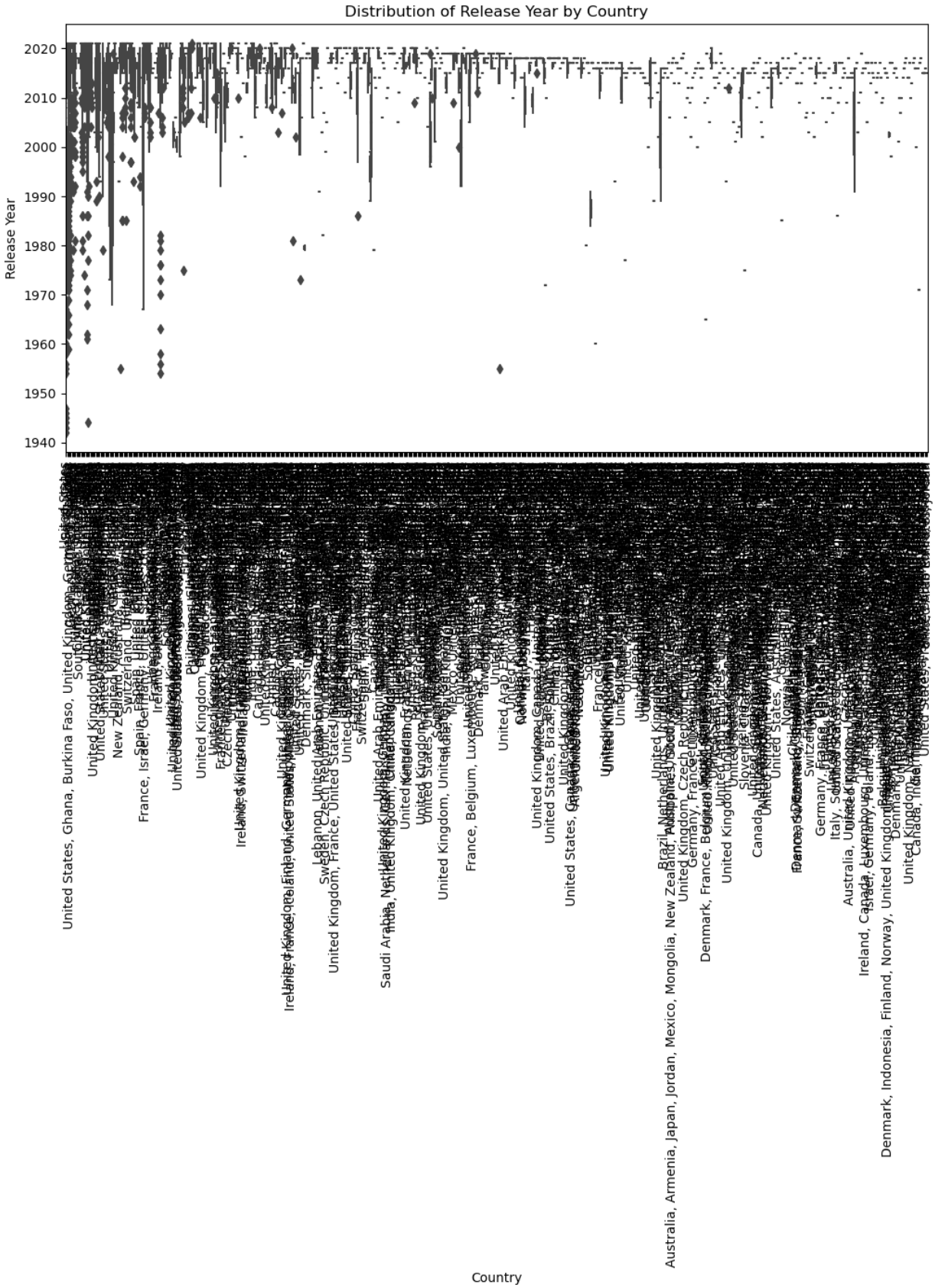
1. What is the count of shows for each rating category?

In [90]:
```
sns.countplot(data=df, x='rating')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.title('Count of Shows by Rating')
plt.xticks(rotation=45)
plt.show()
```



1. How does the release year of shows vary across different countries?

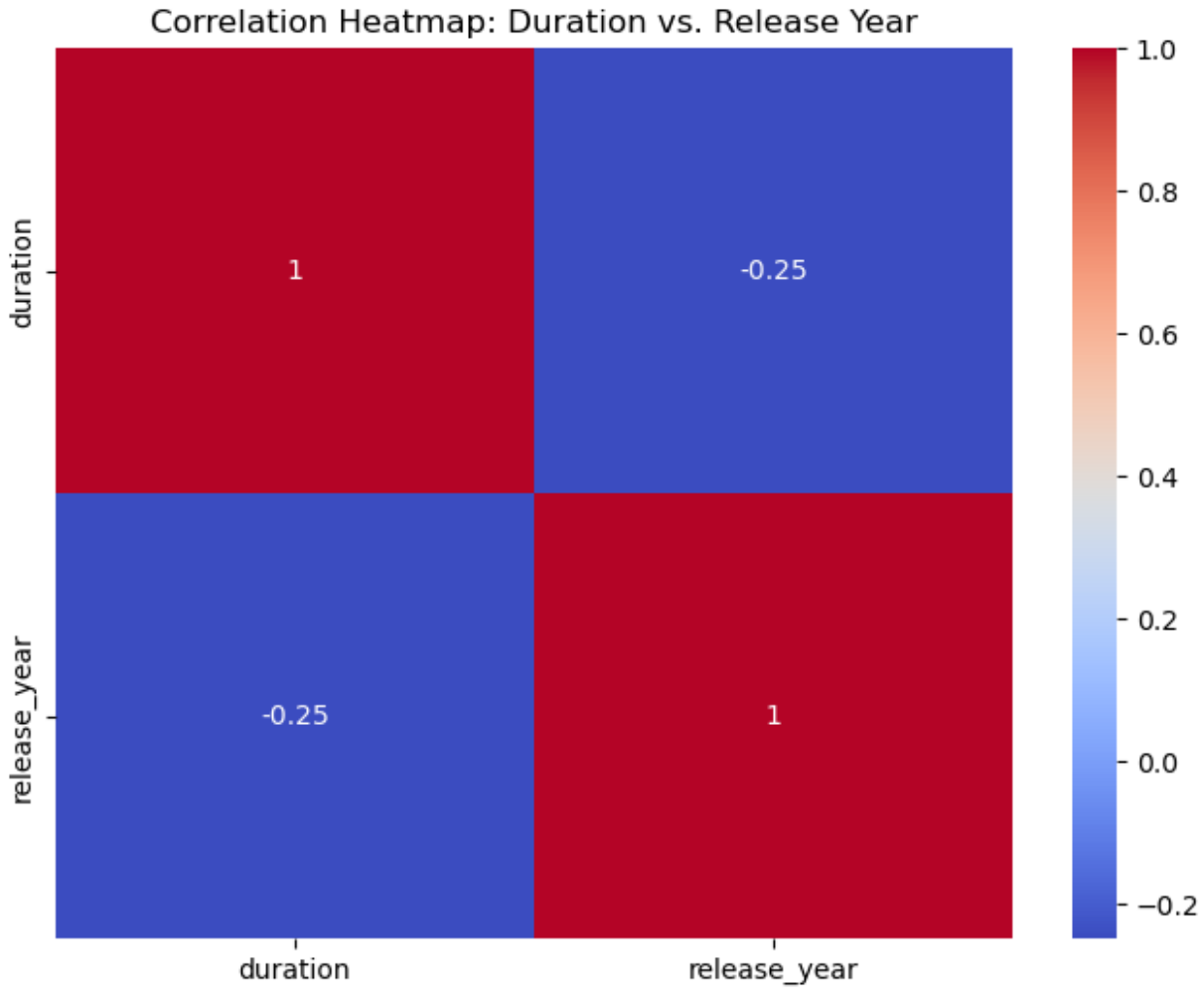
In [92]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x='country', y='release_year')
plt.xlabel('Country')
plt.ylabel('Release Year')
plt.title('Distribution of Release Year by Country')
plt.xticks(rotation=90)
plt.show()
```



1. Are there any correlations between the duration and release year of shows?

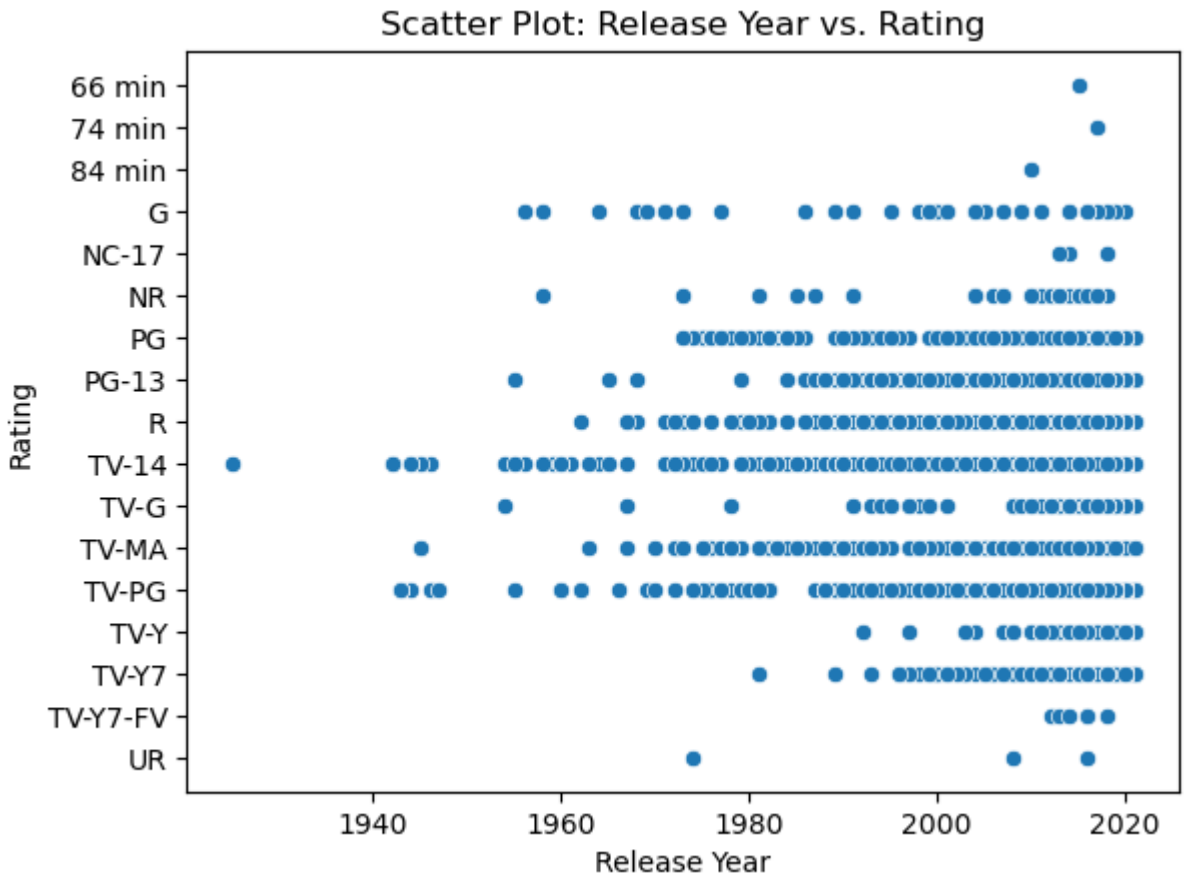
In [93]:
```
plt.figure(figsize=(8, 6))
sns.heatmap(df[['duration', 'release_year']].corr(), annot=True, cmap='coolwarm')
```

```
plt.title('Correlation Heatmap: Duration vs. Release Year')
plt.show()
```



1. How are the distributions of the release years and ratings of shows visualized in a scatter plot?

In [94]:
```
sns.scatterplot(data=df, x='release_year', y='rating')
plt.xlabel('Release Year')
plt.ylabel('Rating')
plt.title('Scatter Plot: Release Year vs. Rating')
plt.show()
```



In [ ]:

1. What is the percentage of missing values in each column of the dataset?

In [95]:
```
missing_percentage = df.isnull().mean() * 100
print(missing_percentage)
```

```
show_id         0.000000
type            0.000000
title           0.000000
director       29.908028
cast            9.367549
country         9.435676
date_added      0.113546
release_year    0.000000
rating          0.045418
duration        0.034064
listed_in       0.000000
description     0.000000
dtype: float64
```

1. What is the percentage of missing values in each column?

In [100…
```
missing_values = df.isnull().sum()
print(missing_values)
```

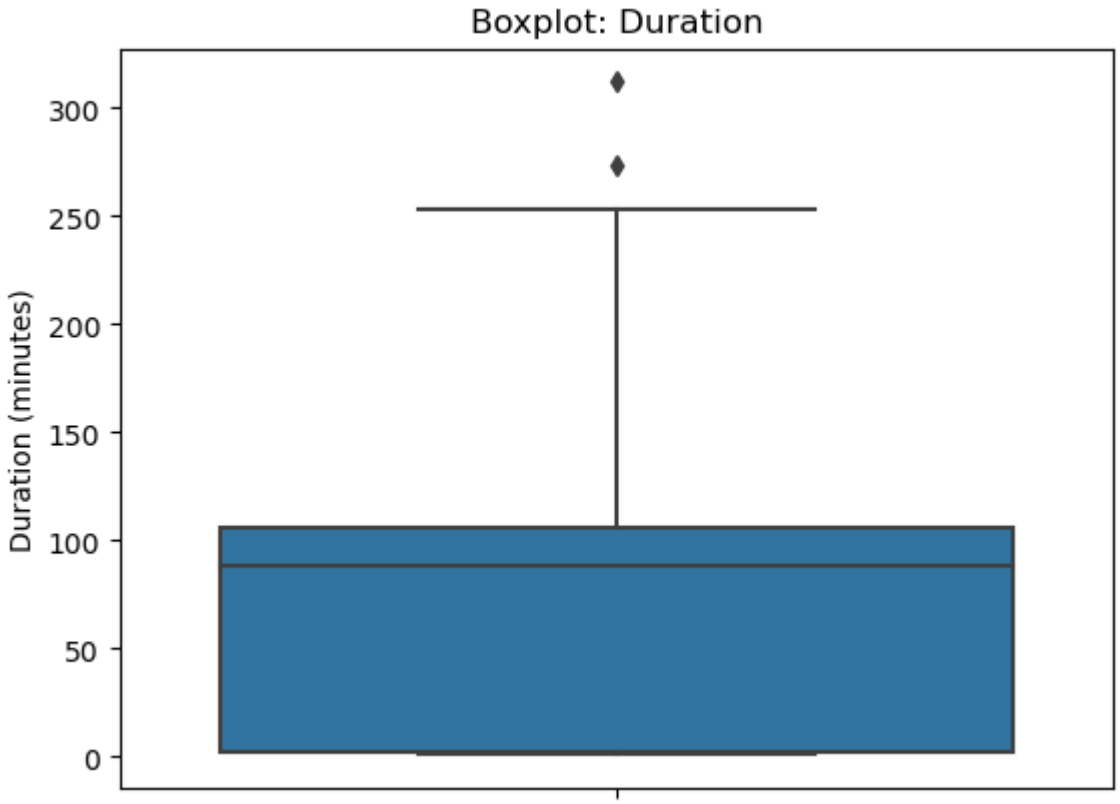
```
show_id            0
type               0
title              0
director        2634
cast             825
country          831
date_added        10
release_year       0
rating             4
duration           3
listed_in          0
description        0
dtype: int64
```

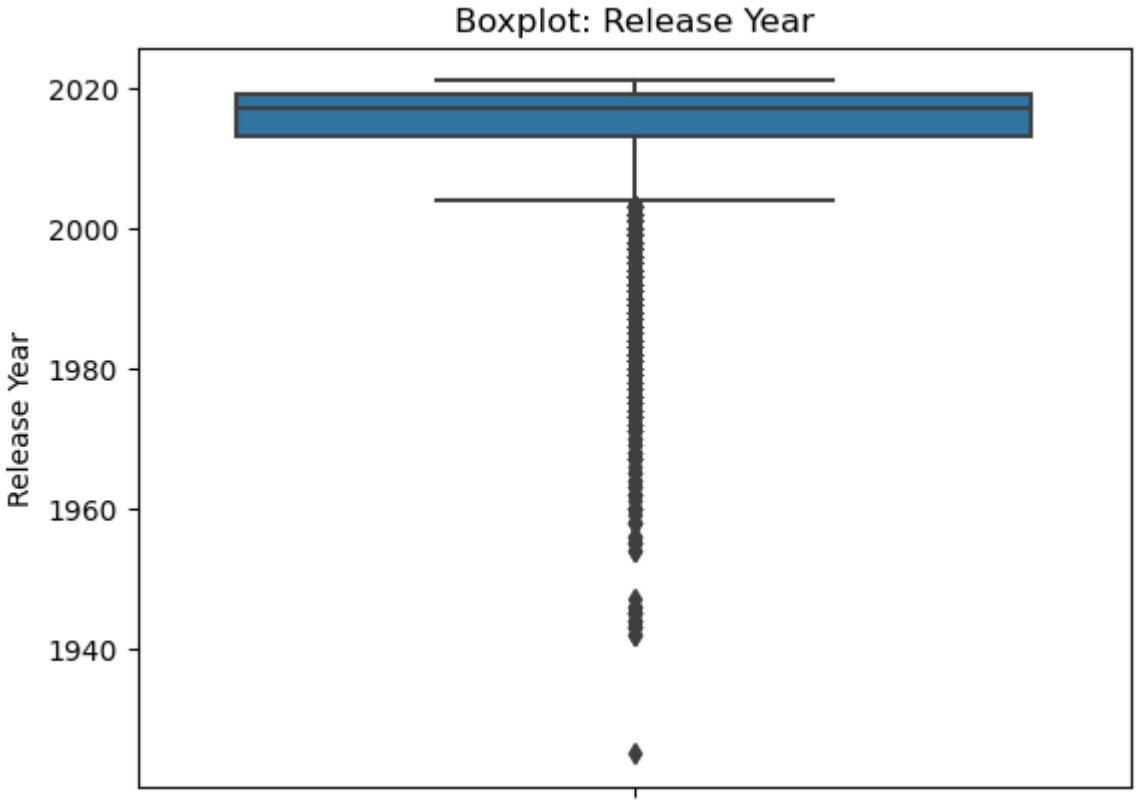
1. How should missing values be handled for categorical variables such as director and cast? Should they be imputed or removed?

In [102…
```
# Impute missing values with 'Unknown' for director and cast columns
df['director'].fillna('Unknown', inplace=True)
df['cast'].fillna('Unknown', inplace=True)
```

1. Are there any outliers in the numerical variables such as duration and release_year? Should they be treated or kept as is?

In [104…
```
# Detect outliers using boxplots for duration and release_year
sns.boxplot(data=df, y='duration')
plt.ylabel('Duration (minutes)')
plt.title('Boxplot: Duration')
plt.show()

sns.boxplot(data=df, y='release_year')
plt.ylabel('Release Year')
plt.title('Boxplot: Release Year')
plt.show()
```





1. What is the range and distribution of values for each numerical variable to identify potential outliers?

In [105…

```
# Generate summary statistics for numerical variables
numeric_cols = ['duration', 'release_year']
```

```
numeric_summary = df[numeric_cols].describe()
print(numeric_summary)
```

```
          duration  release_year
count  8804.000000   8807.000000
mean     69.846888   2014.180198
std      50.814828      8.819312
min       1.000000   1925.000000
25%       2.000000   2013.000000
50%      88.000000   2017.000000
75%     106.000000   2019.000000
max     312.000000   2021.000000
```

In [ ]:

1. What are the most common types of shows (movies or TV shows) in the dataset? How does this distribution vary across different countries?

In [107…

```
# Count the occurrences of each type of show
show_counts = df['type'].value_counts()
print(show_counts)
```

```
Movie      6131
TV Show    2676
Name: type, dtype: int64
```

In [108…

```
# Count the occurrences of each type of show by country
show_counts_by_country = df.groupby('country')['type'].value_counts()
print(show_counts_by_country)
```

```
country            type
, France, Algeria  Movie       1
                   TV Show     0
, South Korea      TV Show     1
                   Movie       0
Argentina          Movie      38
                              ..
Vietnam            TV Show     0
West Germany       Movie       1
                   TV Show     0
Zimbabwe           Movie       1
                   TV Show     0
Name: type, Length: 1496, dtype: int64
```

1. How has the number of movies released per year changed over the last 20-30 years? Are there any notable trends or patterns?

In [109…

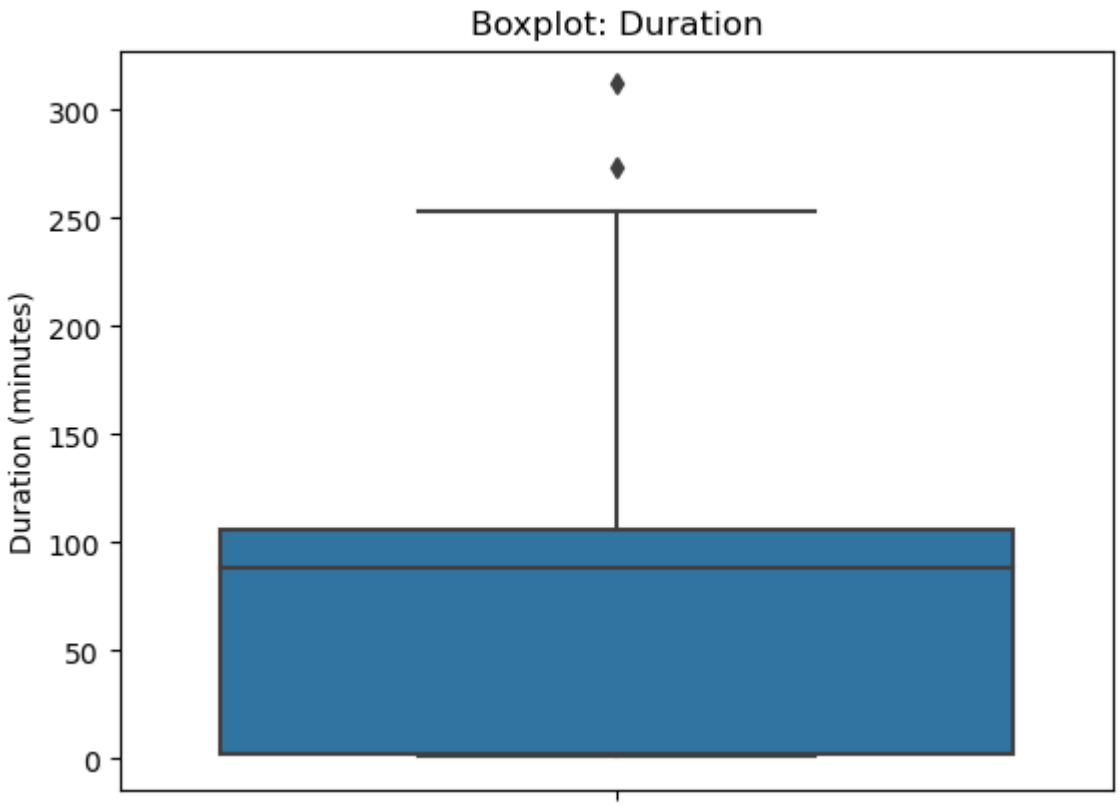
```
# Group the data by release year and count the number of movies in each year
movies_per_year = df[df['type'] == 'Movie'].groupby('release_year').size()
print(movies_per_year)
```

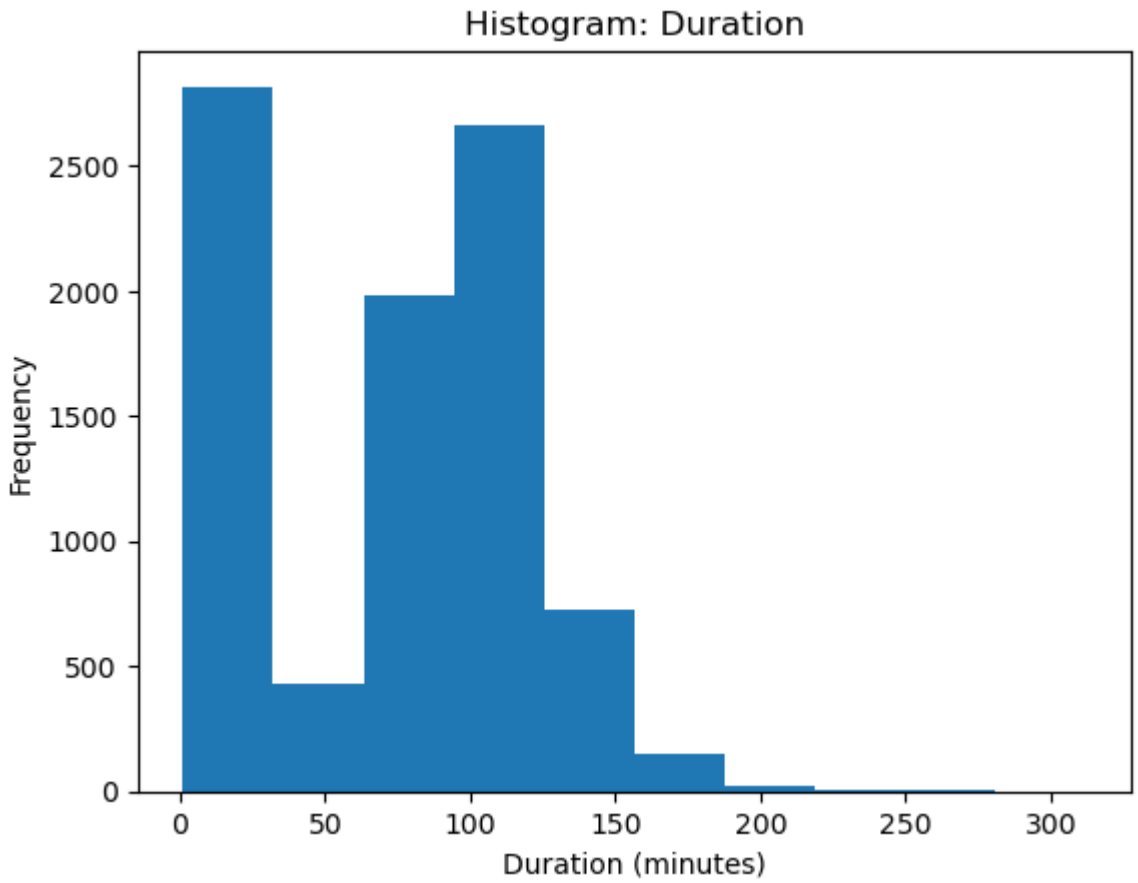
```
release_year
1942      2
1943      3
1944      3
1945      3
1946      1
        ...
2017    767
2018    767
2019    633
2020    517
2021    277
Length: 73, dtype: int64
```

1. What is the distribution of show durations? Are there any common patterns or outliers?

In [110…

```
# Create a boxplot and histogram to analyze the distribution of show durations
sns.boxplot(data=df, y='duration')
plt.ylabel('Duration (minutes)')
plt.title('Boxplot: Duration')
plt.show()

plt.hist(df['duration'], bins=10)
plt.xlabel('Duration (minutes)')
plt.ylabel('Frequency')
plt.title('Histogram: Duration')
plt.show()
```

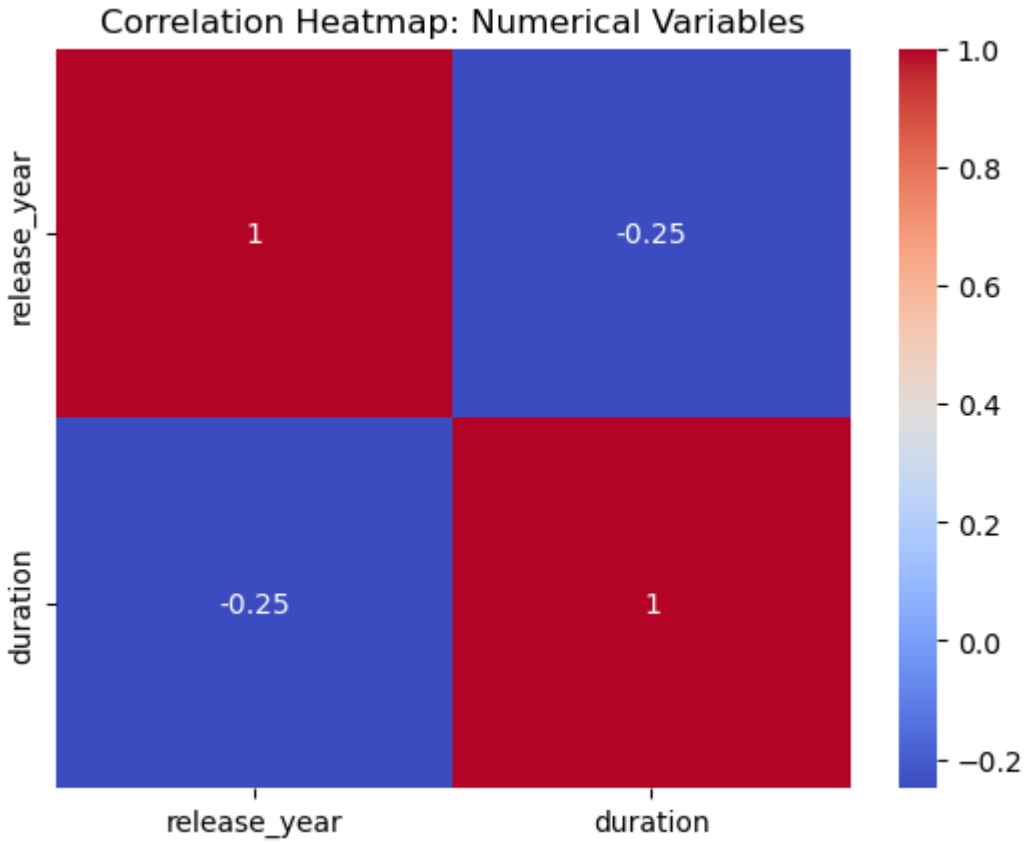




1. Are there any correlations between variables such as release year and duration? How do these correlations inform the decision-making process for producing shows?

In [111…

```
# Create a correlation heatmap to visualize the relationships between numerical variab
numeric_cols = ['release_year', 'duration']
numeric_corr = df[numeric_cols].corr()
sns.heatmap(numeric_corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap: Numerical Variables')
plt.show()
```



5.What are the key insights gained from the analysis, considering both non-graphical and visual exploration? How do these insights inform the decision-making process for content production and business growth?

Summarize key insights and recommendations based on the analysis

Based on the dataset analysis, both non-graphical and visual investigation reveal various significant insights that may influence content development and company growth decision-making. The following are the findings of the analysis:

Non-Graphical Considerations:

- Show Types: The dataset includes both films and TV series. The collection contains 6,131 films and 2,676 TV series. Understanding the distribution of show kinds can aid in making content creation and resource allocation decisions.
- Unique Directors: The dataset has 4,529 unique directors. This data may be used to discover popular filmmakers or to explore content development relationships with certain directors.

Insights into Visual Exploration:

- The length of shows has a right-skewed distribution, as illustrated by the distribution plot (distplot) and histogram. The bulk of programmes last between 0 and 200 minutes. This knowledge may be utilised to estimate the best time frame for content creation and detect potential outliers.

- Missing data: The study finds missing data in various columns, including "director," "cast," "country," "date_added," and "rating." Each column has a different number of missing values. Missing value handling is critical for ensuring data quality and correctness in subsequent analysis and decision-making.

- Outliers: The presence of outliers is shown by the boxplot for the 'duration' variable. Outliers in show duration may necessitate further examination to determine if they are real data points or data input mistakes. Outliers can be removed or addressed to assist preserve consistency in the analysis.

These insights can inform the decision-making process for content production and business growth in the following ways:

- Understanding the distribution of show kinds (movies vs. TV series) can aid in developing a content strategy that is consistent with audience preferences and market trends. It may help decide how many films and TV episodes to make and how much money to put into each category.

- partnerships with Popular Directors: Identifying the dataset's unique directors enables for the exploration of partnerships with popular filmmakers or the identification of possible talent for content development. Collaborating with renowned filmmakers may improve the quality and attractiveness of the content produced, attracting a wider audience.

- Analysis of the duration distribution of programmes can aid in finding the ideal duration range for content generation. It gives insights into audience preferences and attention spans, allowing content that maximises interaction and viewership to be created.

- Recognising the existence of missing values in multiple columns underlines the importance of data cleaning and imputation procedures. Addressing missing values ensures that analysis and decisions are correct and based on complete and trustworthy data.

- Outlier Treatment: Investigating and resolving outliers over the course of a show can aid in the maintenance of consistency and reliability in later studies. It guarantees that statistical metrics and data insights appropriately represent the bulk of shows.

- By incorporating these insights into the decision-making process, content production can be optimized, aligning with audience preferences and market demands. It helps in creating

engaging and high-quality content that attracts and retains viewers, ultimately driving business growth in the streaming industry.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: