# Netflix Bussiness Case Study

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
netflix_data = pd.read_csv("netflix.csv")
netflix_data.head()
```

| | show_id | type | title | director | cast | country | date_added | release_year | rat |
|---|---|---|---|---|---|---|---|---|---|
| 0 | s1 | Movie | Dick Johnson Is Dead | Kirsten Johnson | NaN | United States | September 25, 2021 | 2020 | PG |
| 1 | s2 | TV Show | Blood & Water | NaN | Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban... | South Africa | September 24, 2021 | 2021 | TV- |
| 2 | s3 | TV Show | Ganglands | Julien Leclercq | Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi | NaN | September 24, 2021 | 2021 | TV- |

Next steps: Generate code with netflix_data View recommended plots

## Defining Problem Statement and Analyzing basic metrics

Problem: Analyze the data to generate insights helping Netflix decide what type of shows/movies to produce and grow the business in different countries.

## Observations on the shape of data, data types, conversion of categorical attributes to 'category', missing value detection, statistical summary

```
# Observations on the shape of data
print("Shape of the data:", netflix_data.shape)
# Data types of all the attributes
print("Data types of attributes:\n", netflix_data.dtypes)
# Convert 'type' and 'rating' columns to 'category'
netflix_data['type'] = netflix_data['type'].astype('category')
netflix_data['rating'] = netflix_data['rating'].astype('category')
netflix_data['country'] = netflix_data['country'].astype('category')

# Missing value detection
print("Missing values count:\n", netflix_data.isnull().sum())
# Statistical summary
print("Statistical summary:\n", netflix_data.describe())
```

```
Shape of the data: (8807, 12)
Data types of attributes:
 show_id         object
type            object
title           object
director        object
cast            object
country         object
```

```
date_added      object
release_year     int64
rating          object
duration        object
listed_in       object
description     object
dtype: object
Missing values count:
 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
count   8807.000000
mean    2014.180198
std        8.819312
min     1925.000000
25%     2013.000000
50%     2017.000000
75%     2019.000000
max     2021.000000
```

**Relese year has a wide range of 1925 to 2022 albeit most of the movies and tv shows in the dataframe are from 2018 onwards.**

## Non-Graphical Analysis: Value counts and unique attributes

```
print("value counts for each attribute:")
for col in netflix_data.columns:
    if netflix_data[col].dtype == 'int64':
        print(col, ":", netflix_data[col].value_counts())

print("Unique values for each attribute:")
for col in netflix_data.columns:
    print(col, ":", netflix_data[col].nunique())
```

```
value counts for each attribute:
release_year : 2018    1147
2017    1032
2019    1030
2020     953
2016     902
        ...
1959       1
1925       1
1961       1
1947       1
1966       1
Name: release_year, Length: 74, dtype: int64
Unique values for each attribute:
show_id : 8807
type : 2
title : 8807
director : 4528
cast : 7692
country : 748
date_added : 1767
release_year : 74
rating : 17
duration : 220
listed_in : 514
description : 8775
```

## Visual Analysis - Univariate, Bivariate after pre-processing of the data

*Pre-processing involves unnesting of the data in columns like Actor, Director, Country*

```
# spliting the string to list elements
netflix_data['actor'] = netflix_data['cast'].str.split(', ')
netflix_data['director'] = netflix_data['director'].str.split(', ')
netflix_data['country'] = netflix_data['country'].str.split(', ')
netflix_data['genre'] = netflix_data['listed_in'].str.split(', ')

# Unnesting the each column
netflix_data = netflix_data.explode('actor')
netflix_data = netflix_data.explode('director')
netflix_data = netflix_data.explode('country')
netflix_data = netflix_data.explode('genre')

netflix_data.drop(['cast', 'listed_in'], axis=1, inplace=True)
netflix_data.reset_index()

netflix_data.head()
```

| | show_id | type | title | director | country | date_added | release_year | rating | duration |
|---|---|---|---|---|---|---|---|---|---|
| 0 | s1 | Movie | Dick Johnson Is Dead | Kirsten Johnson | United States | September 25, 2021 | 2020 | PG-13 | 90 min |
| 1 | s2 | TV Show | Blood & Water | NaN | South Africa | September 24, 2021 | 2021 | TV-MA | 2 Seasons |
| 1 | s2 | TV | Blood & | NaN | South | September | 2021 | TV-MA | 2 |

```
# Convert date columns to datetime format, handling errors
netflix_data['date_added'] = pd.to_datetime(netflix_data['date_added'], errors='coerce')

# Extract relevant information from date columns
netflix_data['year_added'] = netflix_data['date_added'].dt.year
netflix_data['month_added'] = netflix_data['date_added'].dt.month

netflix_data.head()
```

| | show_id | type | title | director | country | date_added | release_year | rating | duration |
|---|---|---|---|---|---|---|---|---|---|
| 0 | s1 | Movie | Dick Johnson Is Dead | Kirsten Johnson | United States | 2021-09-25 | 2020 | PG-13 | 90 min |
| 1 | s2 | TV Show | Blood & Water | NaN | South Africa | 2021-09-24 | 2021 | TV-MA | 2 Seasons |
| 1 | s2 | TV Show | Blood & Water | NaN | South Africa | 2021-09-24 | 2021 | TV-MA | 2 Seasons |

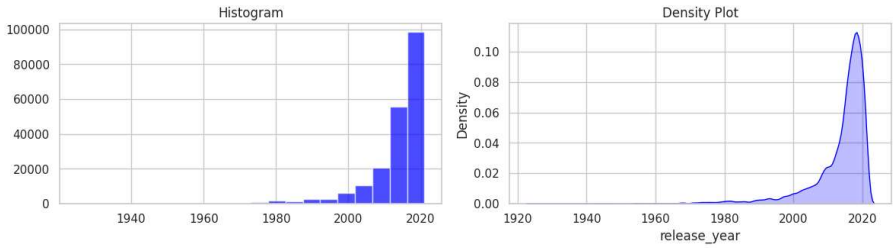
```
# Split 'duration' into separate columns for movies' length and TV shows' number of seasons
netflix_data['duration_min'] = netflix_data['duration'].str.extract(r'(\d+)', expand=False)
netflix_data['seasons'] = netflix_data['duration'].str.extract(r'(\d+) Seasons', expand=False)

# Convert 'duration_min' and 'seasons' columns to numeric type
netflix_data['duration_min'] = pd.to_numeric(netflix_data['duration_min'], errors='coerce')
netflix_data['seasons'] = pd.to_numeric(netflix_data['seasons'], errors='coerce')

# Drop the original 'duration' column
netflix_data.drop(columns=['duration'], inplace=True)

# some necessary cleaning for rating column
netflix_data = netflix_data[~netflix_data['rating'].isin(['66 min', '74 min', '84 min'])]

netflix_data.head()
```

| | show_id | type | title | director | country | date_added | release_year | rating | descript |
|---|---|---|---|---|---|---|---|---|---|
| **0** | s1 | Movie | Dick Johnson Is Dead | Kirsten Johnson | United States | 2021-09-25 | 2020 | PG-13 | As her fa nears end o life, filn |
| **1** | s2 | TV Show | Blood & Water | NaN | South Africa | 2021-09-24 | 2021 | TV-MA | cros paths party, a C Tow |
| **1** | s2 | TV Show | Blood & Water | NaN | South Africa | 2021-09-24 | 2021 | TV-MA | cros paths party, a C Tow |
| **1** | s2 | TV Show | Blood & Water | NaN | South Africa | 2021-09-24 | 2021 | TV-MA | cros paths party, a C Tow |
| **1** | s2 | TV Show | Blood & Water | NaN | South Africa | 2021-09-24 | 2021 | TV-MA | cros paths party, a C Tow |

## Univariate Analysis for continuous/categorical variable(s)

```
# Setting up the plotting style
sns.set(style="whitegrid")

# Define a function to create the required univariate plots for continuous variables
def plot_continuous_variable(data, variable):
    plt.figure(figsize=(12, 6))

    # Histogram
    plt.subplot(2, 2, 1)
    plt.hist(data[variable].dropna(), bins=20, color='blue', alpha=0.7)
    plt.title('Histogram')

    # Density plot
    plt.subplot(2, 2, 2)
    sns.kdeplot(data[variable].dropna(), color='blue', shade=True)
    plt.title('Density Plot')

    plt.tight_layout()
    plt.show()

plot_continuous_variable(netflix_data, 'release_year')
```

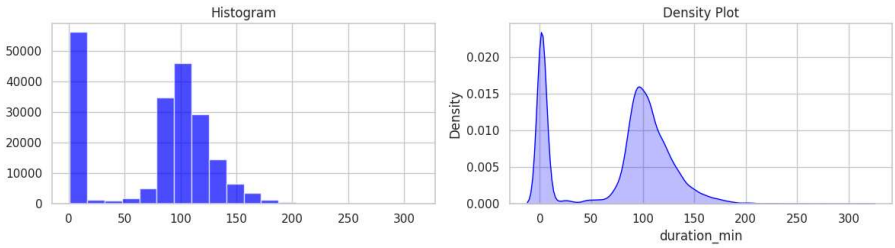
```
<ipython-input-9-8290f6d56524>:15: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data[variable].dropna(), color='blue', shade=True)
```




**From the above graph we can observe that most of the available content is from 21st century, more precisely from 2019 and onwards.**

```
# Plotting for 'duration_min'
plot_continuous_variable(netflix_data, 'duration_min')
```

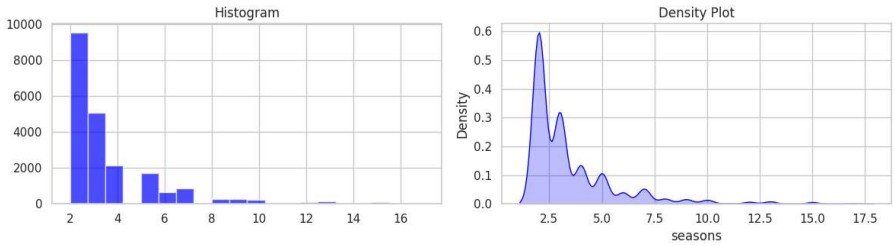
```
<ipython-input-9-8290f6d56524>:15: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data[variable].dropna(), color='blue', shade=True)
```




**Majority of the movies are of either in very short format, i.e, 1-2 minutes or regular features lengths of duration 95 mins.**

```
# Plotting for 'duration_min'
plot_continuous_variable(netflix_data, 'seasons')
```

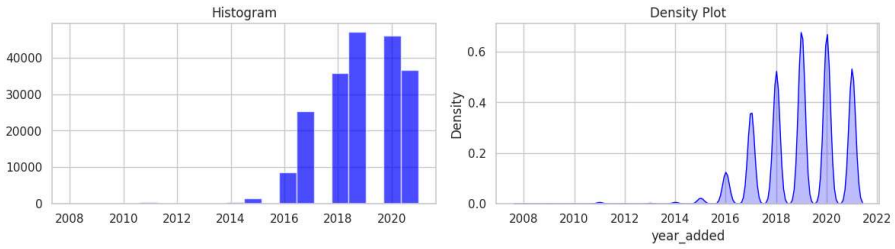
```
<ipython-input-9-8290f6d56524>:15: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data[variable].dropna(), color='blue', shade=True)
```




**Most of the TV shows last only between 2 to 4 seasons with very few outliers in the range of 10+ seasons.**

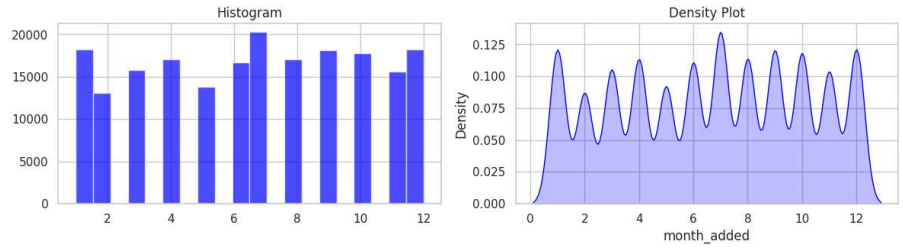
```
# Plotting for 'duration_min'
plot_continuous_variable(netflix_data, 'year_added')
```

```
<ipython-input-9-8290f6d56524>:15: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data[variable].dropna(), color='blue', shade=True)
```




```
plot_continuous_variable(netflix_data, 'month_added')
```

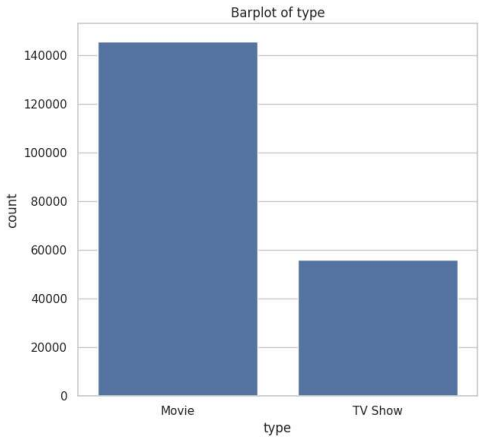
```
<ipython-input-9-8290f6d56524>:15: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data[variable].dropna(), color='blue', shade=True)
```

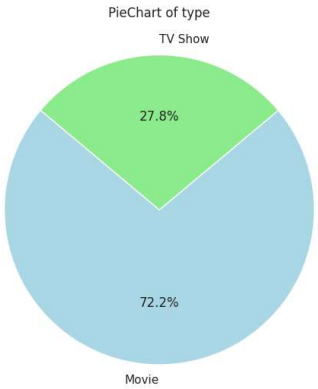



**From the above graph we can deduct that while there isn't much disparity in the content addition month by month, peak time to add content are**

- start of the year - January
- Halfway of the year - July
- End of the year - December

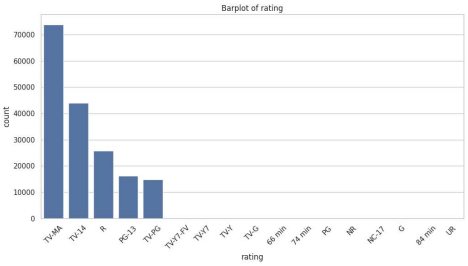
```
# Define a function to create the required univariate plots for categorical variables
def plot_categorical_variable(data, category, figsize=(12, 6), rotation=0):
    plt.figure(figsize=figsize)

    # Baxplot
    if len(data[category].value_counts()) > 5:
        top_5_categories = data[category].value_counts().nlargest(5).index
    # Filter the data to include only the top 5 categories
        filter_data = data[data[category].isin(top_5_categories)]
        plt.subplot(1, 2, 1)
        sns.countplot(data=filter_data, x=category, order=filter_data[category].value_counts().index)
        plt.xticks(rotation=rotation)
    else:
        plt.subplot(1, 2, 1)
        sns.countplot(data=data, x=category)
    plt.title(f'Barplot of {category}')
    plt.xlabel(category)

    # piechart
    data_counts = data[category].value_counts()
    if len(data_counts) >= 5:
        data_counts = data_counts[:5]
    plt.subplot(1, 2, 2)
    plt.pie(data_counts, labels=data_counts.index, autopct='%1.1f%%', startangle=140, colors=['lightblue', 'lightgreen'])
    plt.title(f'PieChart of {category}')
    plt.axis('equal')

    plt.tight_layout()
    plt.show()

plot_categorical_variable(netflix_data, 'type')
```

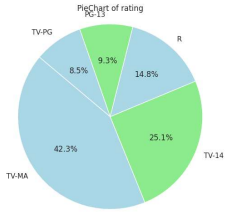



**Movies constitute 3 times more of the content than TV shows.**

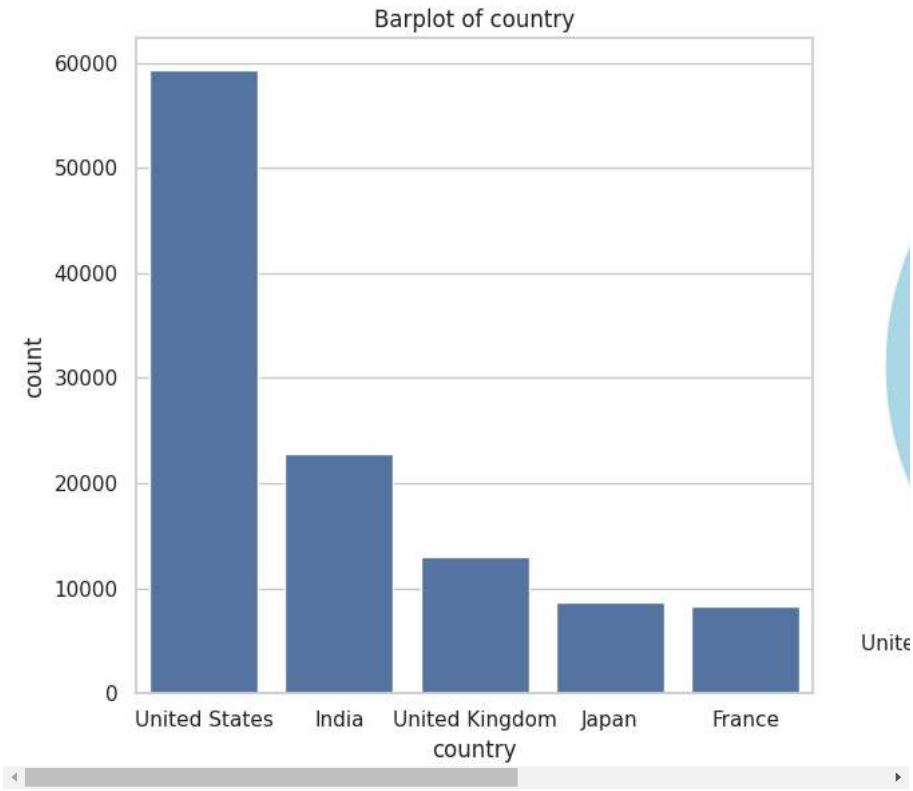
```
plot_categorical_variable(netflix_data, 'rating', (20, 6), rotation=45)
```



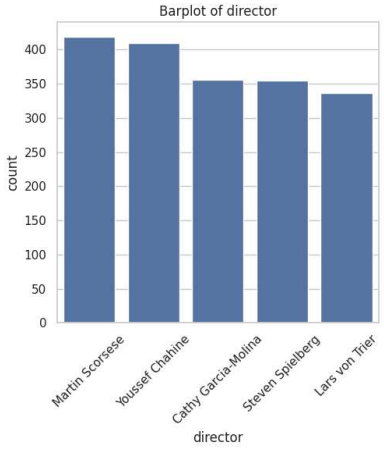

**42.3% of the content is rated TV-MA and 25.1% is of TV-14 rating.**

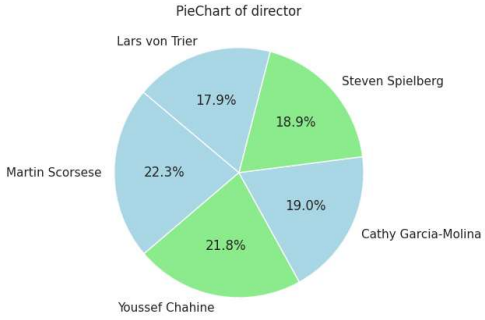
```
plot_categorical_variable(netflix_data, 'country')
```



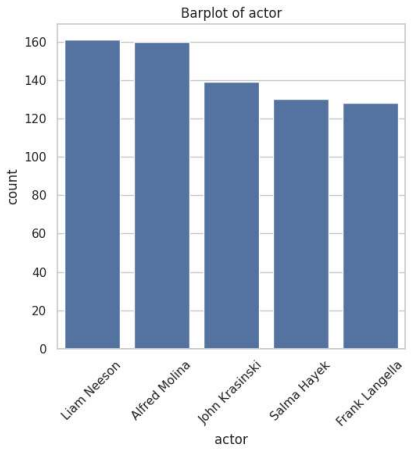

**India and Japan are the biggest non-english speaking content producers and should be prioritized for next bussiness ventures.**

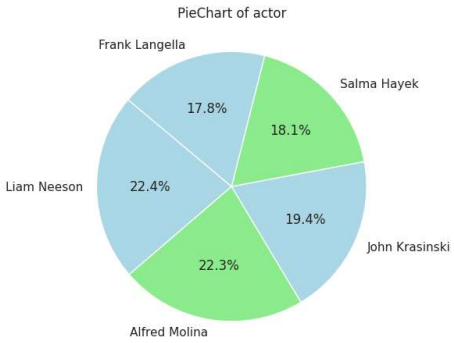
```
plot_categorical_variable(netflix_data, 'director', rotation=45)
```



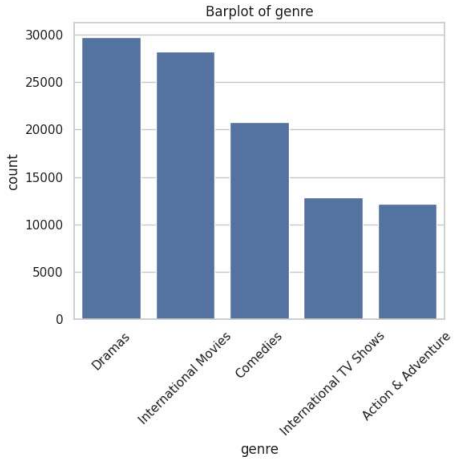



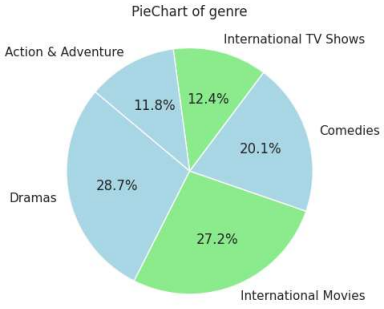

```
plot_categorical_variable(netflix_data, 'actor', rotation=45)
```




```
plot_categorical_variable(netflix_data, 'genre', rotation=45)
```



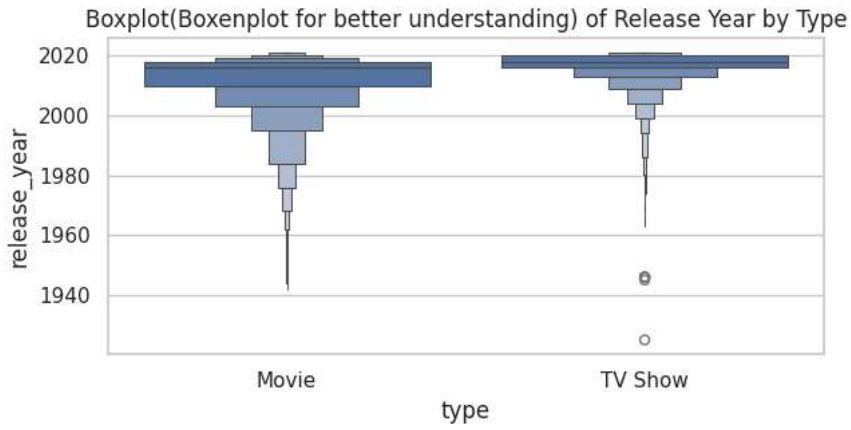




**As International Movies and TV shows are gaining popularity, should be more invested in developing next in these genres.**

## 4.2 Bivariate Analysis for continuous/categorical variable(s)

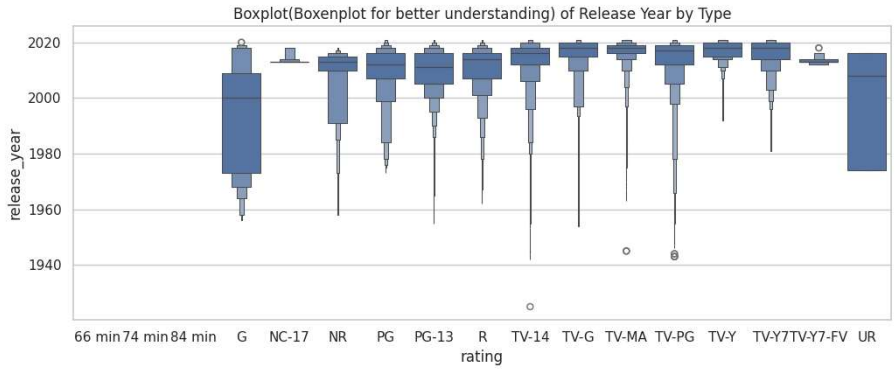
Double-click (or enter) to edit

```
# Define a function to create the required bivariate plots for continuous-categorical variables
def plot_bivariate_plot_NC(data, category, variable, figsize=(12, 6)):
    plt.figure(figsize=figsize)

    # Boxplot
    plt.subplot(2, 2, 1)
    sns.boxenplot(x=category, y=variable, data=data)
    plt.title('Boxplot(Boxenplot for better understanding) of Release Year by Type')
    plt.xlabel(category)
    plt.ylabel(variable)


    plt.tight_layout()
    plt.show()

plot_bivariate_plot_NC(netflix_data, 'type', 'release_year')
```




- **TV shows have very high number of outliers but are mostly skewed to 2019 and onwards.**
- **While Movies are a bit more even data is still has majority from 2015 and onwards.**

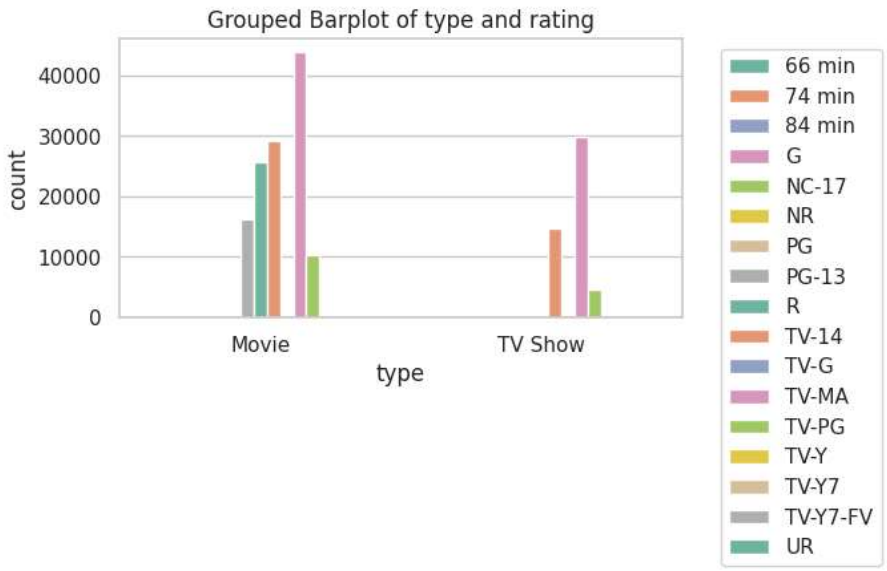
```
plot_bivariate_plot_NC(netflix_data.reset_index(drop=True), 'rating', 'release_year', (20, 8))
```




**Shows that could get ratings of NC-17, TV-Y-FV, shouldn't be developed further due to the lack of audience.**

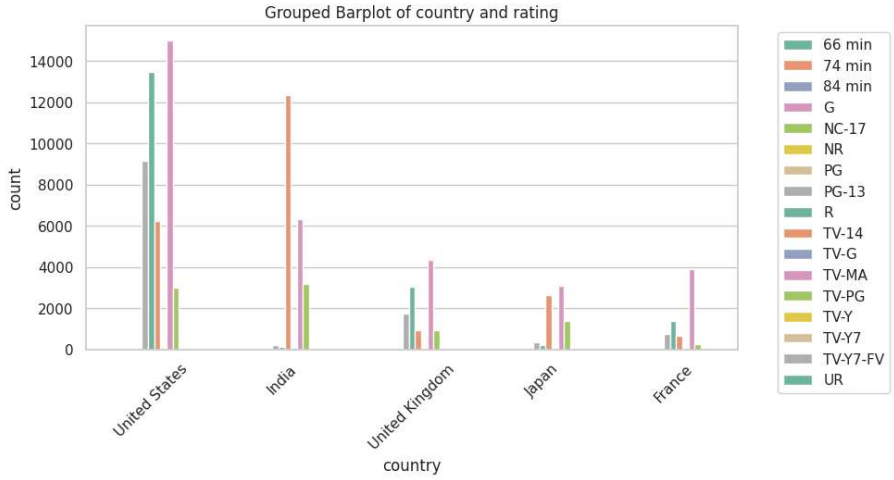
```
# Define a function to create the required bivariate plots for categorical-categorical variables
def plot_bivariate_plot_CC(data, category_1, category_2, figsize=(12, 6), rotation=0):
    plt.figure(figsize=figsize)
    # Grouped Baxplot
    if len(data[category_1].value_counts()) > 5:
        top_5_categories = data[category_1].value_counts().nlargest(5).index
    # Filter the data to include only the top 5 categories
        filter_data = data[data[category_1].isin(top_5_categories)]
```

```
    if len(data[category_2].value_counts()) > 5:
        top_5_categories = data[category_2].value_counts().nlargest(5).index
    # Filter the data to include only the top 5 categories
        filter_data = data[data[category_2].isin(top_5_categories)]

    plt.subplot(2, 2, 1)
    sns.countplot(data=filter_data, x=category_1, hue=category_2, palette='Set2', order=data[category_1].value_counts().nlargest(5).index)
    plt.xticks(rotation=rotation)
    plt.title(f'Grouped Barplot of {category_1} and {category_2}')
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plot_bivariate_plot_CC(netflix_data, 'type', 'rating')
```



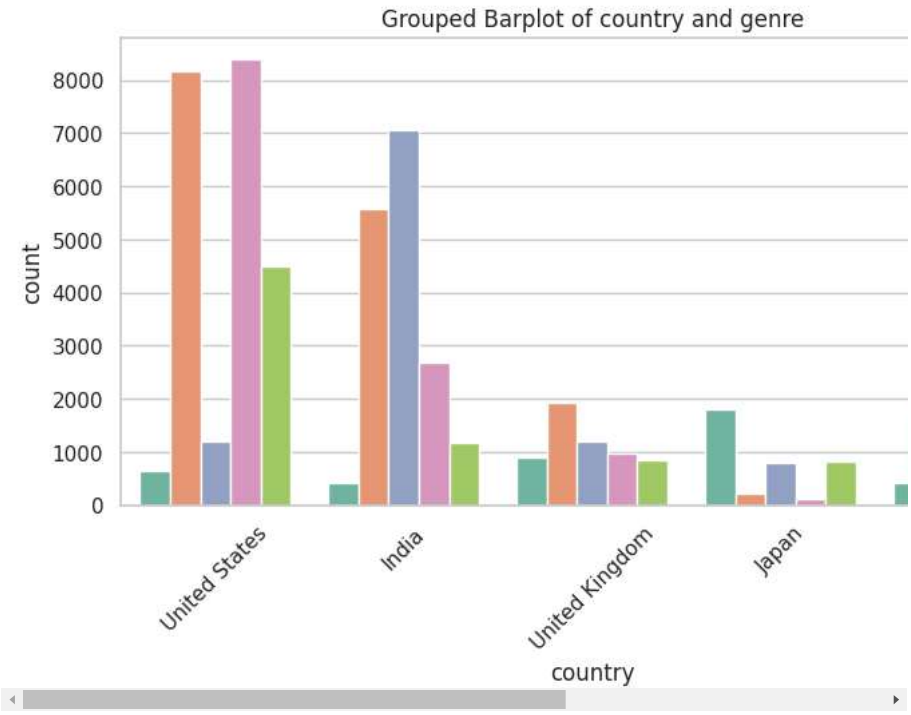

**Most popular items in TV show and Movies are among TV-MA rated content.**

```
plot_bivariate_plot_CC(netflix_data.reset_index(drop=True), 'country', 'rating', figsize=(20, 10), rotation=45)
```



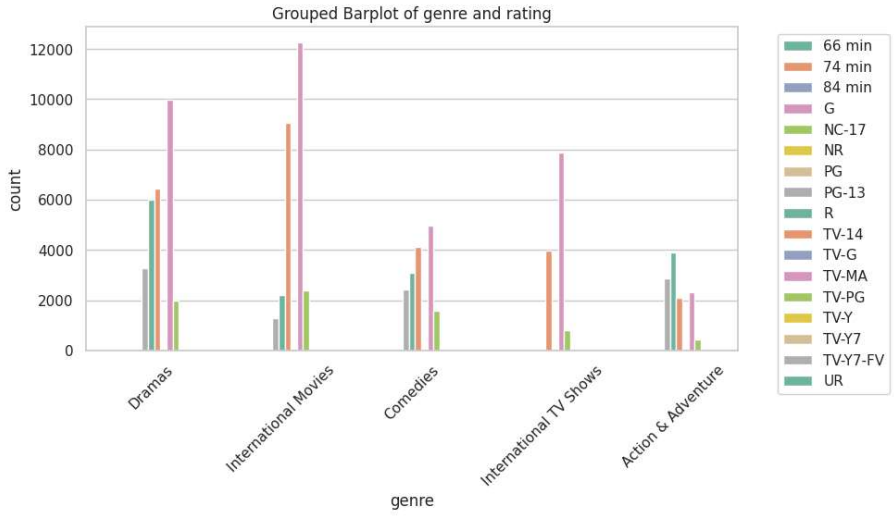

**India and Japan have way higher percentage of TV-14 rated content as compared to US and UK.**

```
plot_bivariate_plot_CC(netflix_data.reset_index(drop=True), 'country', 'genre', figsize=(20, 10), rotation=45)
```



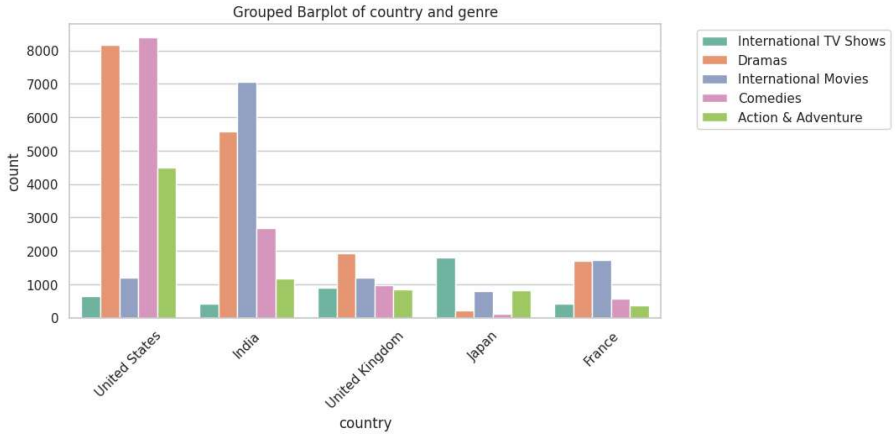

- **India produce more Movies than TV shows**
- **Japan produce more TV-Dramas than movies**

```
plot_bivariate_plot_CC(netflix_data.reset_index(drop=True), 'genre', 'rating', figsize=(20, 10), rotation=45)
```



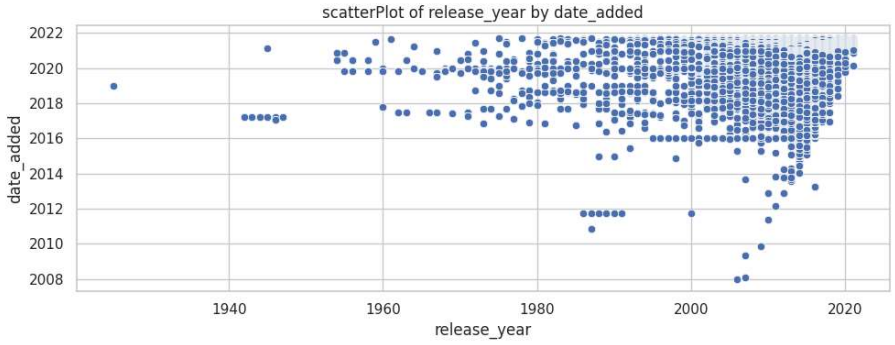

**A better precent of International movies and TV shows are rated TV-14 as TV-MA as is in normal TV Dramas.**

```
plot_bivariate_plot_CC(netflix_data.reset_index(drop=True), 'country', 'genre', figsize=(20, 10), rotation=45)
```



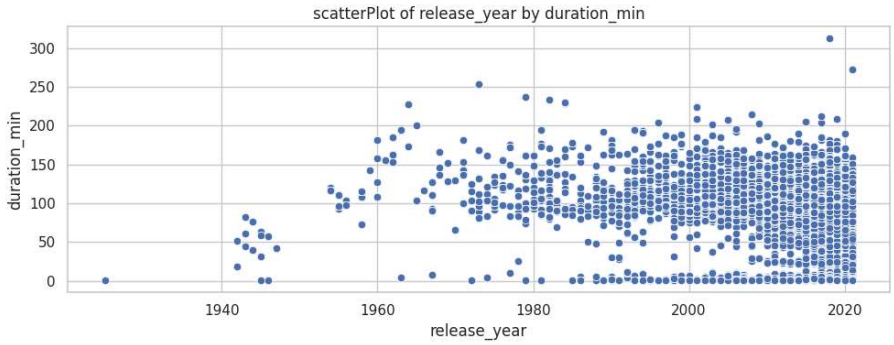

- **A Massive portion of content from India are Dramas and Comedies.**
- **A good chunk of content from Japan is in Action/Adventure genre.**

```
# Define a function to create the required bivariate plots for continuous-continuous variables
def plot_bivariate_plot_NN(data, variable_1, variable_2, figsize=(10, 4)):
    plt.figure(figsize=figsize)
    # Scatterplot
    sns.scatterplot(x=variable_1, y=variable_2, data=data.reset_index(drop=True))
    plt.title(f'scatterPlot of {variable_1} by {variable_2}')
    plt.xlabel(variable_1)
    plt.ylabel(variable_2)

    plt.tight_layout()
    plt.show()

plot_bivariate_plot_NN(netflix_data, 'release_year', 'date_added')
```



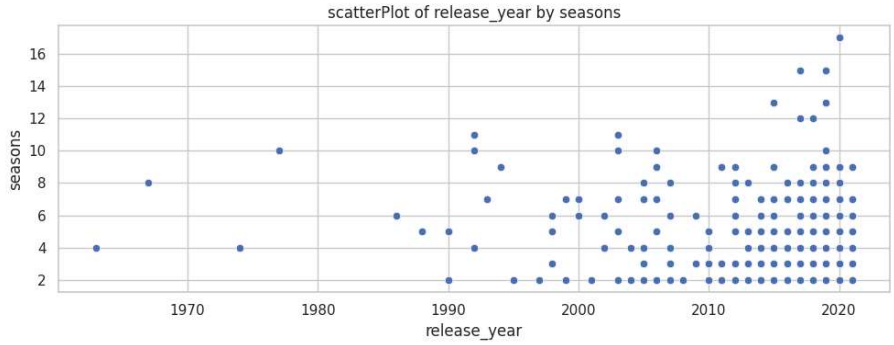

**A good portion of older releases have been added in 2017, 2019**

```
plot_bivariate_plot_NN(netflix_data, 'release_year', 'duration_min')
```




**Majority of the latest content is 95-120 min range.**

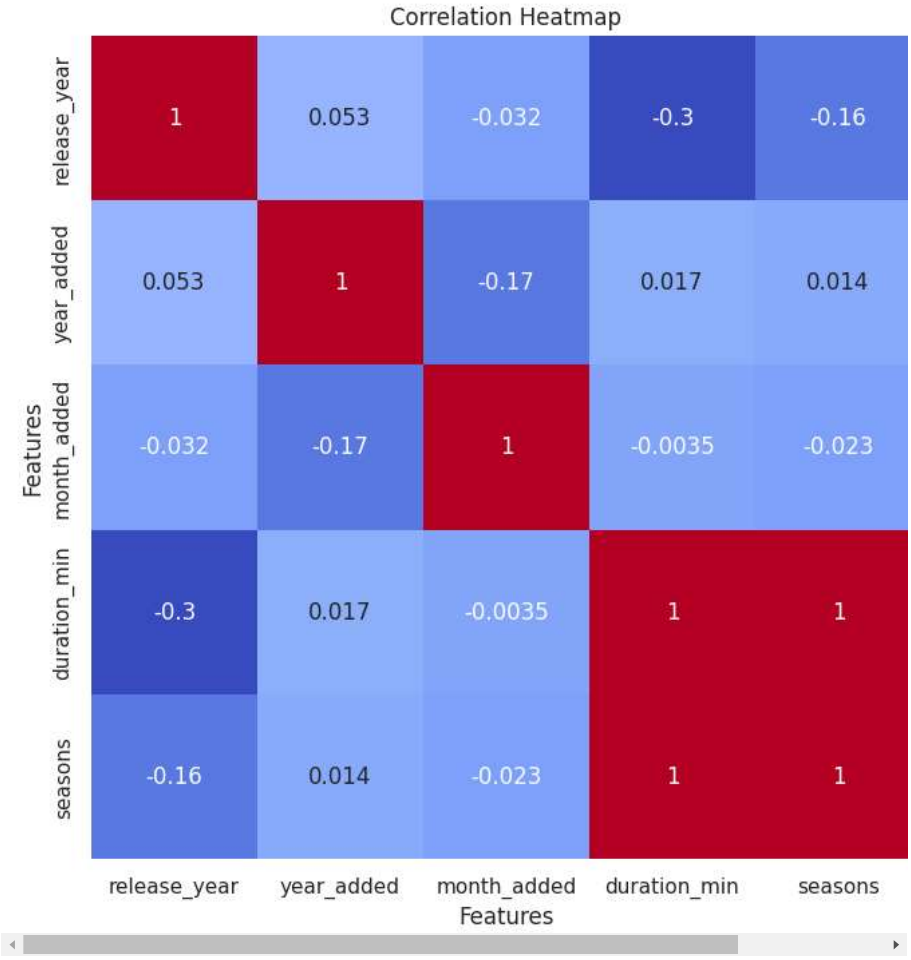
```
plot_bivariate_plot_NN(netflix_data, 'release_year', 'seasons')
```




**TV shows in general lasted only 2-5 seasons with very few and singular exceptions lasting more than 10 seasons.**

## 4.3 Correlation Analysis

```
# Exclude non-numeric columns from the DataFrame
numeric_data = netflix_data.select_dtypes(include=['number'])

# Calculate the correlation matrix
correlation_matrix = numeric_data.corr()

# Create a heatmap of correlations with additional details
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')

# Add x and y axis labels
plt.xlabel('Features')
plt.ylabel('Features')

plt.show()
```

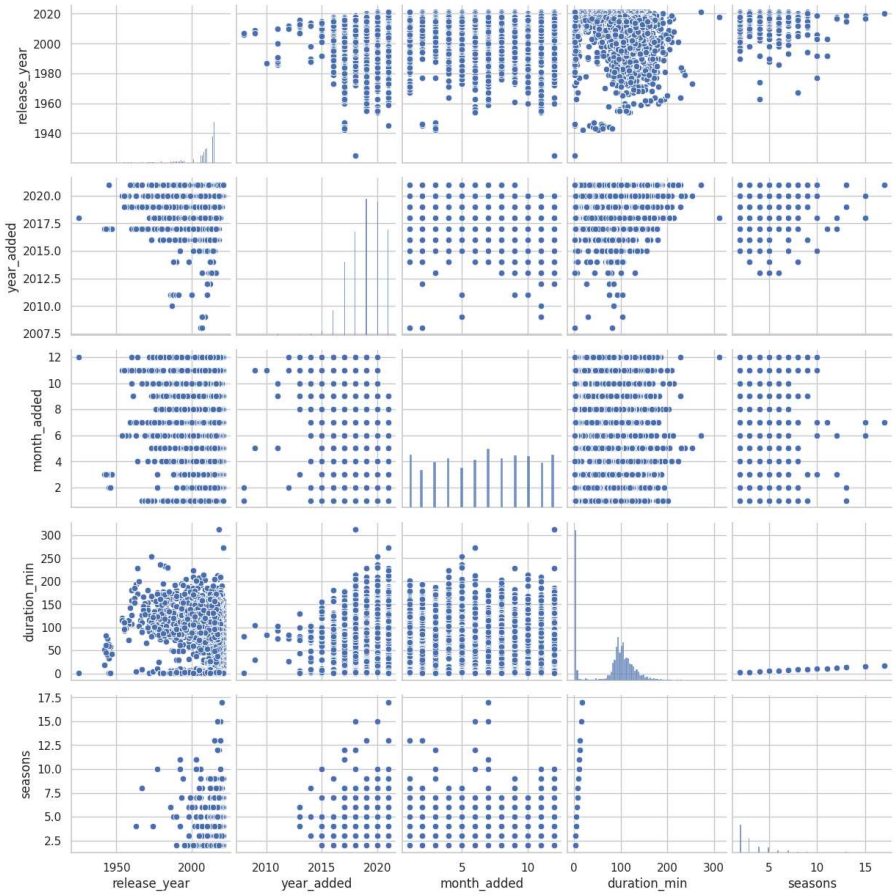



Based on the above correlation matrix:

- **The duration of the content has very little to no correlation to release year or year_added.**
- **Most of the variable have likely negetive correlation, i.e, moves in opposite directions**

```
# Reset the index of the DataFrame
#pairplot_data = netflix_data.select_dtypes(include=['number']).reset_index(drop=True)

# Create pair plots
plt.figure(figsize=(10, 10))
sns.pairplot(netflix_data.reset_index(drop=True))
plt.show()
```

```
<Figure size 1000x1000 with 0 Axes>
```




**Same scenario as heatmap.**

## Missing Value & Outlier check

```
print("Total number of rows:\n", len(netflix_data))
# Check for missing values
print("\nMissing values count:\n", netflix_data.isnull().sum())

# Since there are no missing values in the 'title' and 'release_year' columns, we can drop rows with missing values in other columns.
cleaned_data = netflix_data.dropna(subset=['director', 'actor', 'country', 'date_added', 'rating'])

print("Missing values count:\n", cleaned_data.isnull().sum())
print("\nTotal number of rows after droppping missing values:\n", len(cleaned_data))
```

```
Total number of rows:
 201988

Missing values count:
 show_id              0
type                  0
title                 0
director          50643
country           11897
date_added          158
release_year          0
rating               67
description           0
actor              2146
genre                 0
year_added          158
month_added         158
duration_min          0
seasons          180875
dtype: int64
Missing values count:
 show_id              0
type                  0
title                 0
director              0
country               0
date_added            0
```