In [2]:

```python
# 1. Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
```

In [3]:

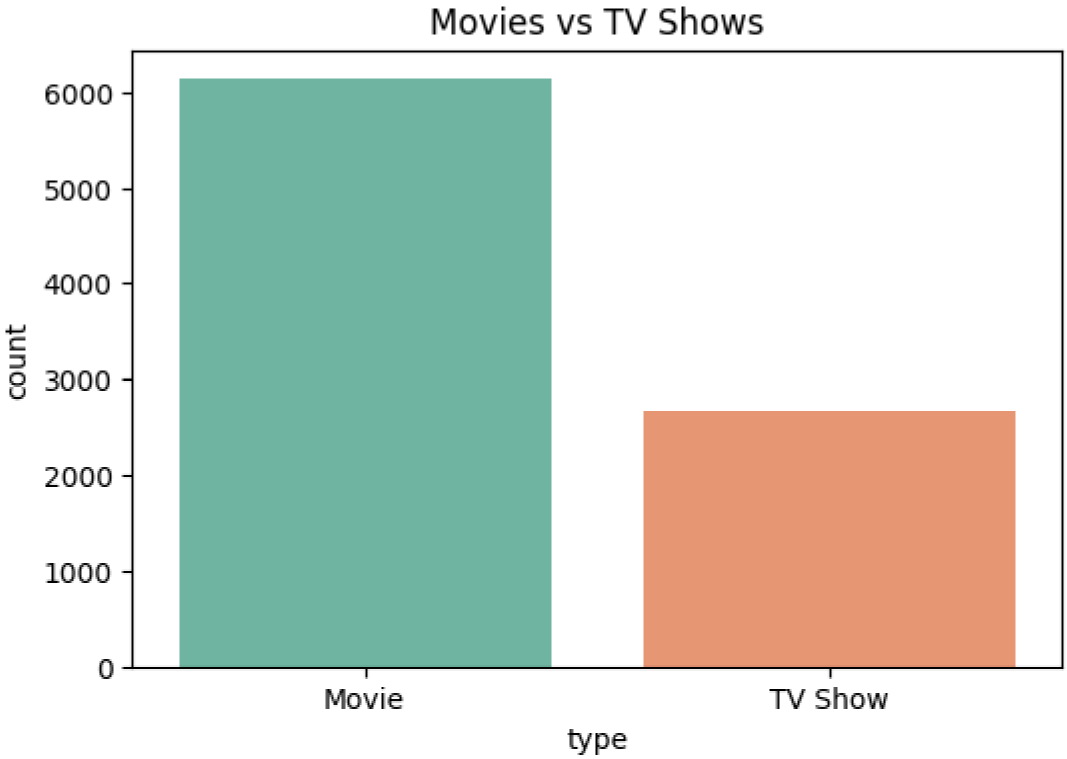
```python
# 2. Load dataset
df = pd.read_csv("netflix_titles.csv.zip")
print("Shape of dataset:", df.shape)
print(df.head())
```

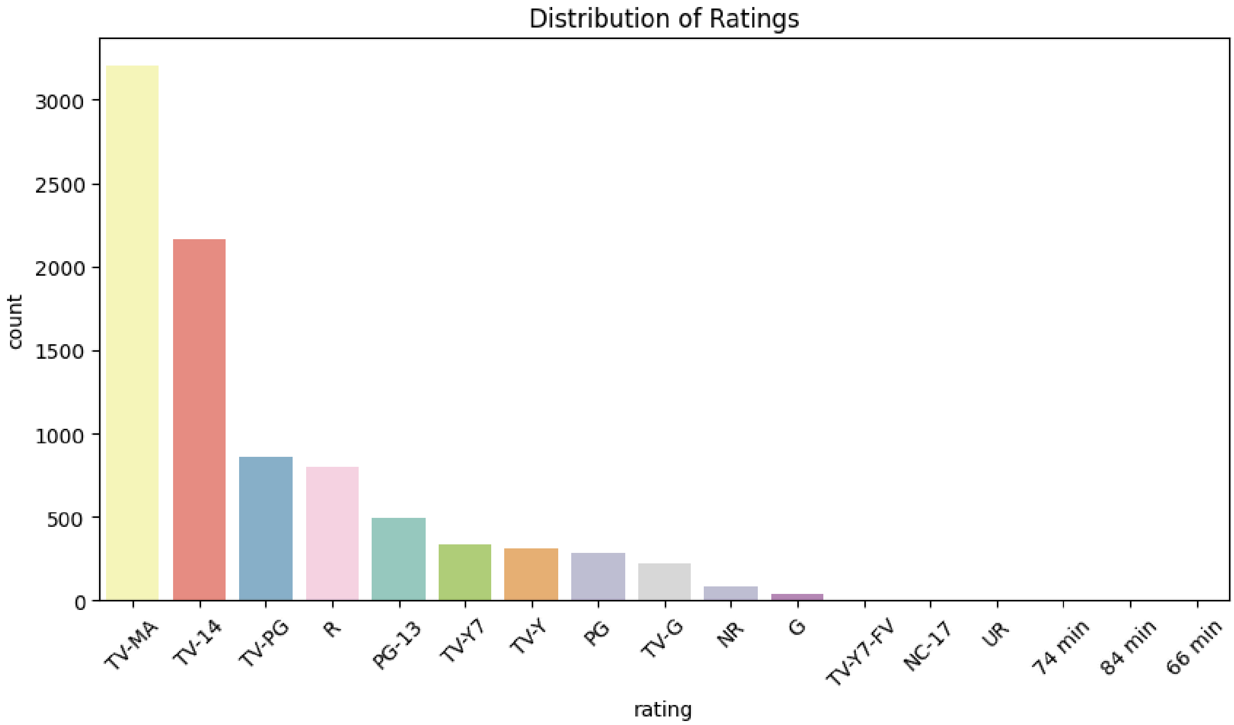
```
Shape of dataset: (8807, 12)
  show_id     type                  title         director  \
0      s1    Movie   Dick Johnson Is Dead  Kirsten Johnson
1      s2  TV Show          Blood & Water              NaN
2      s3  TV Show              Ganglands  Julien Leclercq
3      s4  TV Show  Jailbirds New Orleans              NaN
4      s5  TV Show           Kota Factory              NaN

                                                cast        country  \
0                                                NaN  United States
1  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...   South Africa
2  Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...            NaN
3                                                NaN            NaN
4  Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...          India

           date_added  release_year rating   duration  \
0  September 25, 2021          2020  PG-13     90 min
1  September 24, 2021          2021  TV-MA  2 Seasons
2  September 24, 2021          2021  TV-MA   1 Season
3  September 24, 2021          2021  TV-MA   1 Season
4  September 24, 2021          2021  TV-MA  2 Seasons

                                           listed_in  \
0                                      Documentaries
1    International TV Shows, TV Dramas, TV Mysteries
2  Crime TV Shows, International TV Shows, TV Act...
3                             Docuseries, Reality TV
4  International TV Shows, Romantic TV Shows, TV ...

                                         description
0  As her father nears the end of his life, filmm...
1  After crossing paths at a party, a Cape Town t...
2  To protect his family from a powerful drug lor...
3  Feuds, flirtations and toilet talk go down amo...
4  In a city of coaching centers known to train I...
```

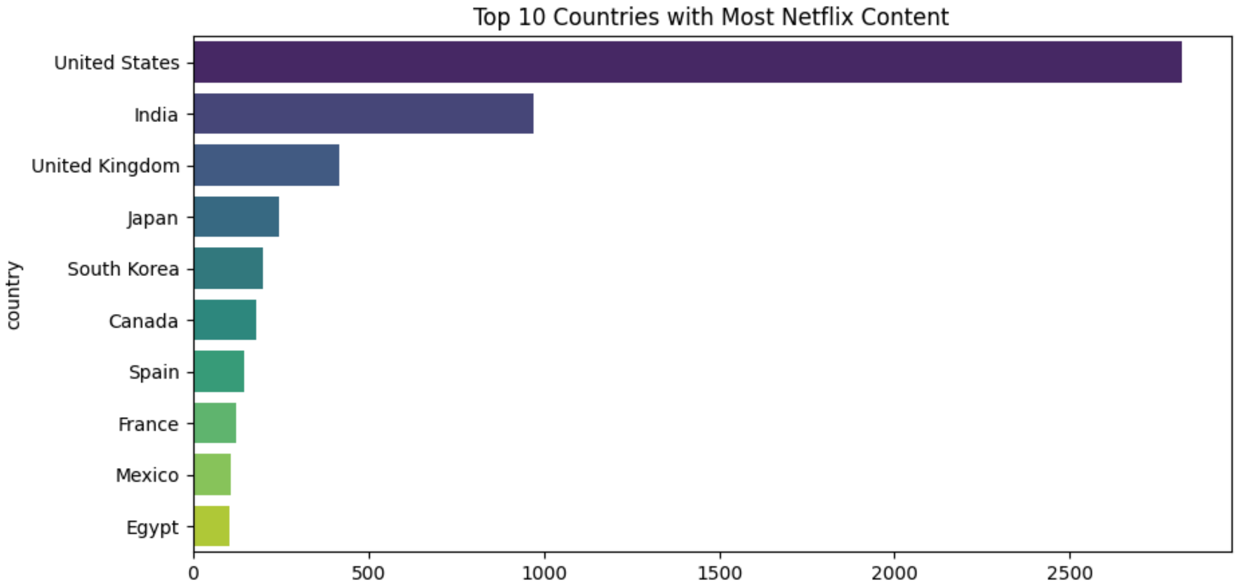
In [4]:

```python
# 3. Content Distribution Analysis
# a. Movies vs TV Shows
plt.figure(figsize=(6,4))
sns.countplot(data=df, x="type", hue="type", palette="Set2", legend=False)
plt.title("Movies vs TV Shows")
plt.show()
```

```python
# b. Ratings distribution
plt.figure(figsize=(10,5))
sns.countplot(data=df, x="rating", hue="rating",
              order=df['rating'].value_counts().index, palette="Set3", legend=
plt.title("Distribution of Ratings")
plt.xticks(rotation=45)
plt.show()

# c. Countries with most Netflix content
top_countries = df['country'].value_counts().head(10)
plt.figure(figsize=(10,5))
sns.barplot(x=top_countries.values, y=top_countries.index, hue=top_countries.i
            palette="viridis", legend=False)
plt.title("Top 10 Countries with Most Netflix Content")
plt.show()
```



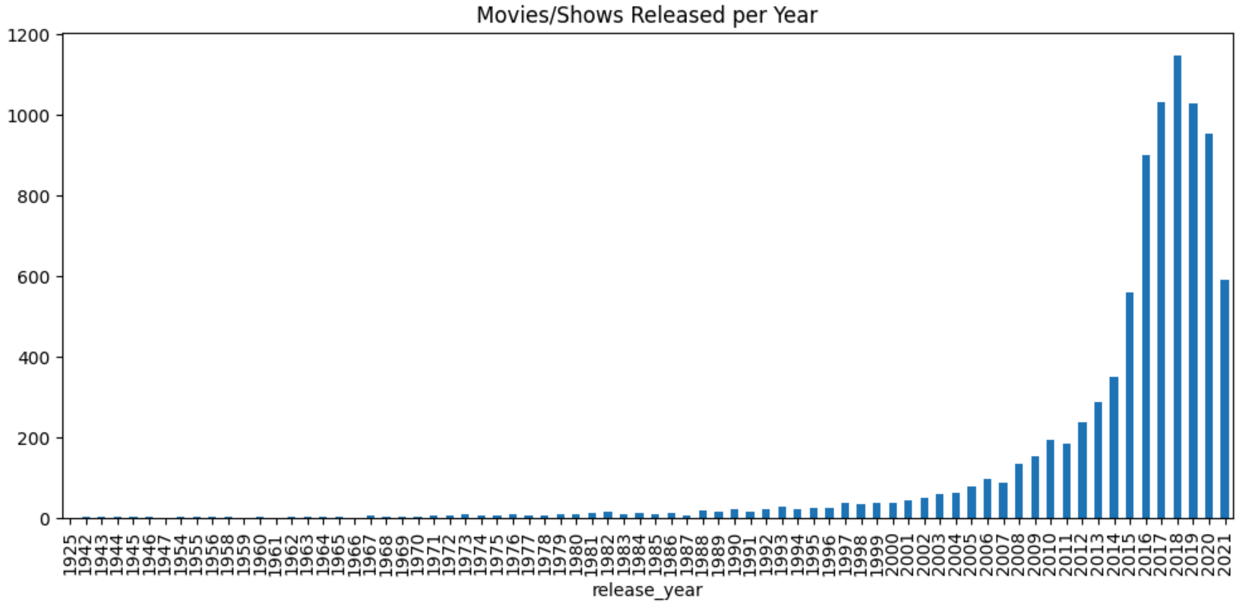


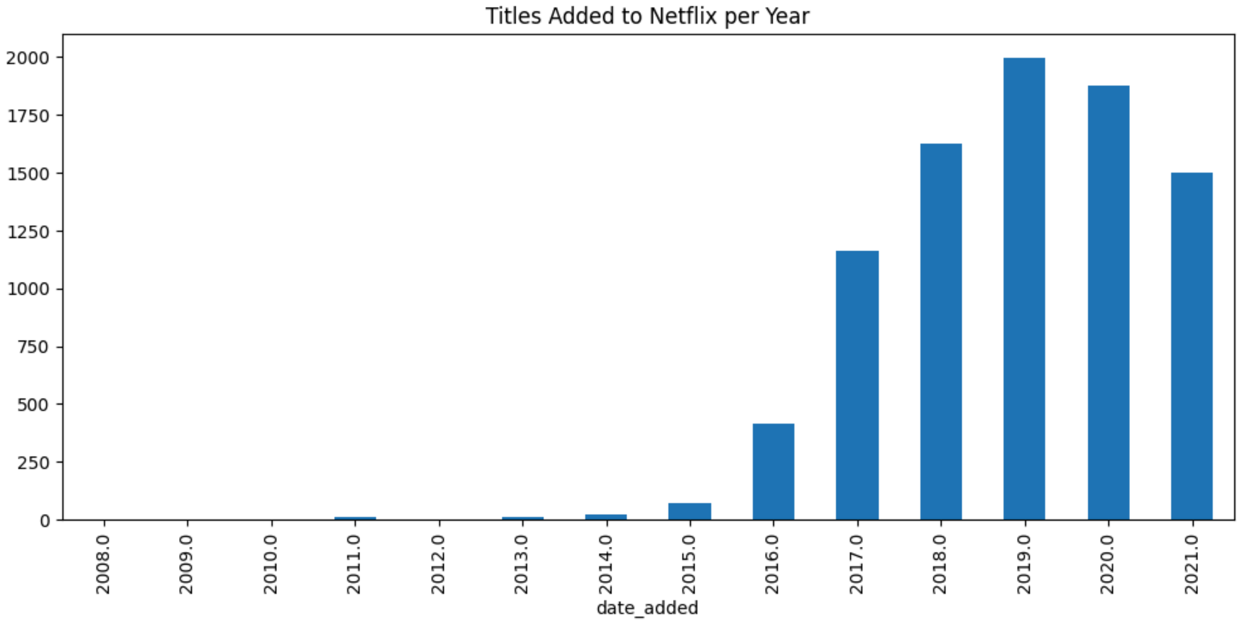


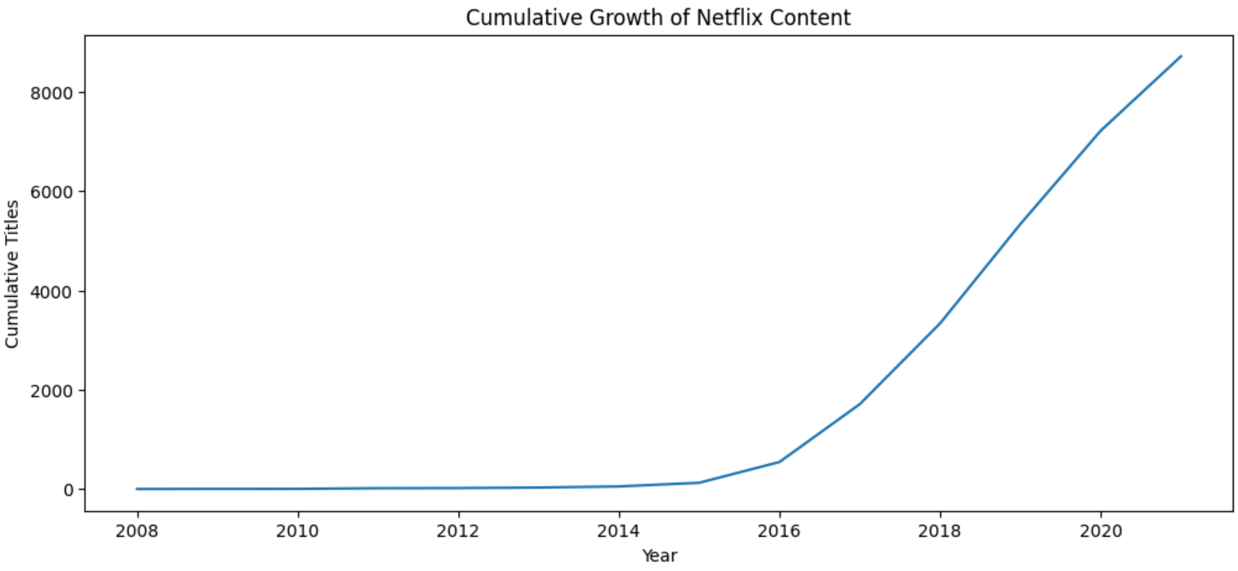
In [5]:

```
# 4. Time-Based Trend Analysis (Fixed)

df['release_year'] = pd.to_numeric(df['release_year'], errors='coerce')
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')

# a. Year-wise releases
plt.figure(figsize=(12,5))
df['release_year'].value_counts().sort_index().plot(kind='bar')
plt.title("Movies/Shows Released per Year")
plt.show()

# b. Year-wise additions to Netflix
plt.figure(figsize=(12,5))
```

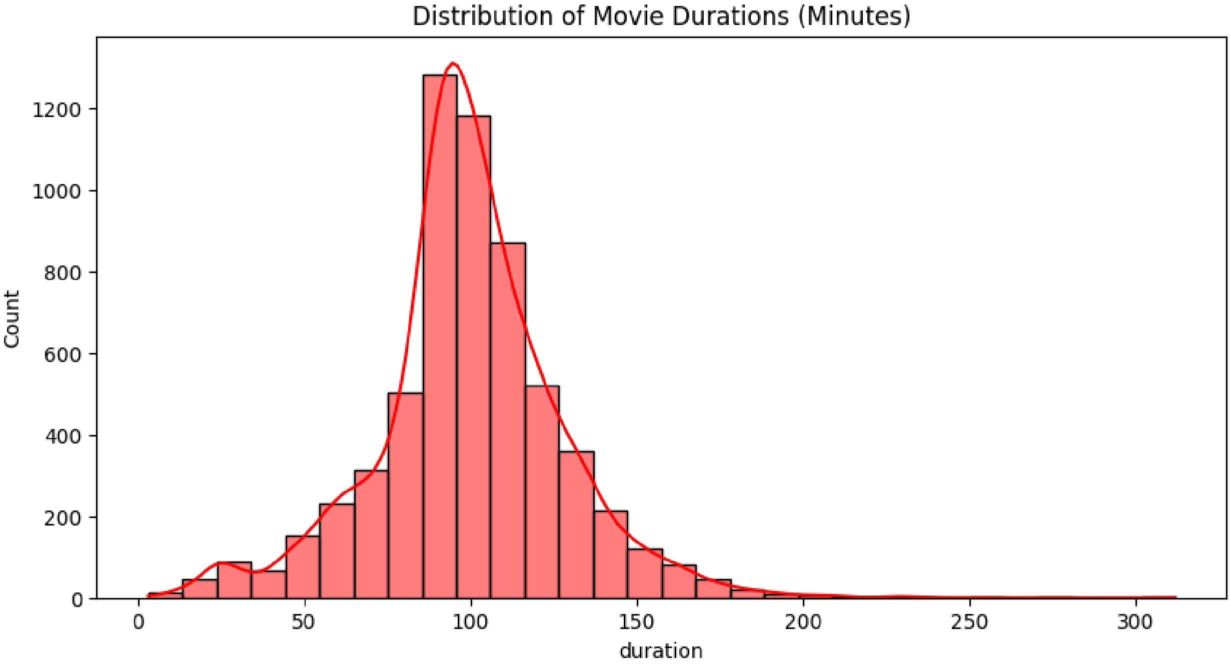
```python
df['date_added'].dt.year.value_counts().sort_index().plot(kind='bar')
plt.title("Titles Added to Netflix per Year")
plt.show()

# c. Growth trend
plt.figure(figsize=(12,5))
df['date_added'].dt.year.value_counts().sort_index().cumsum().plot()
plt.title("Cumulative Growth of Netflix Content")
plt.xlabel("Year")
plt.ylabel("Cumulative Titles")
plt.show()
```

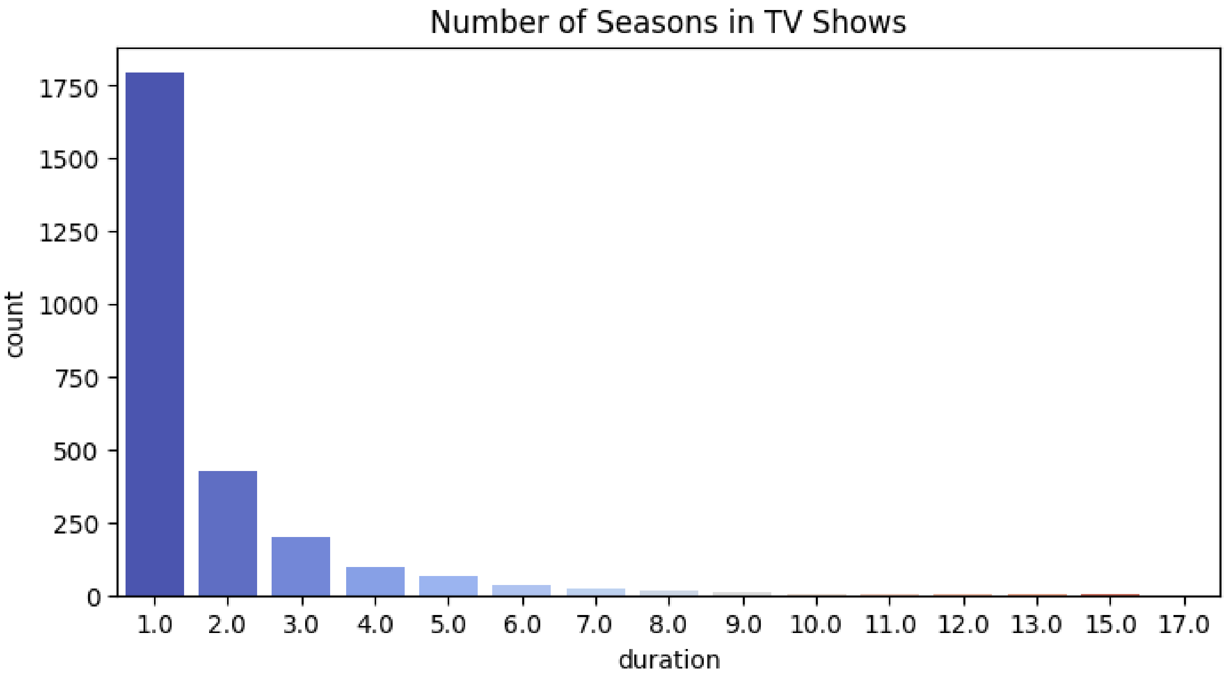






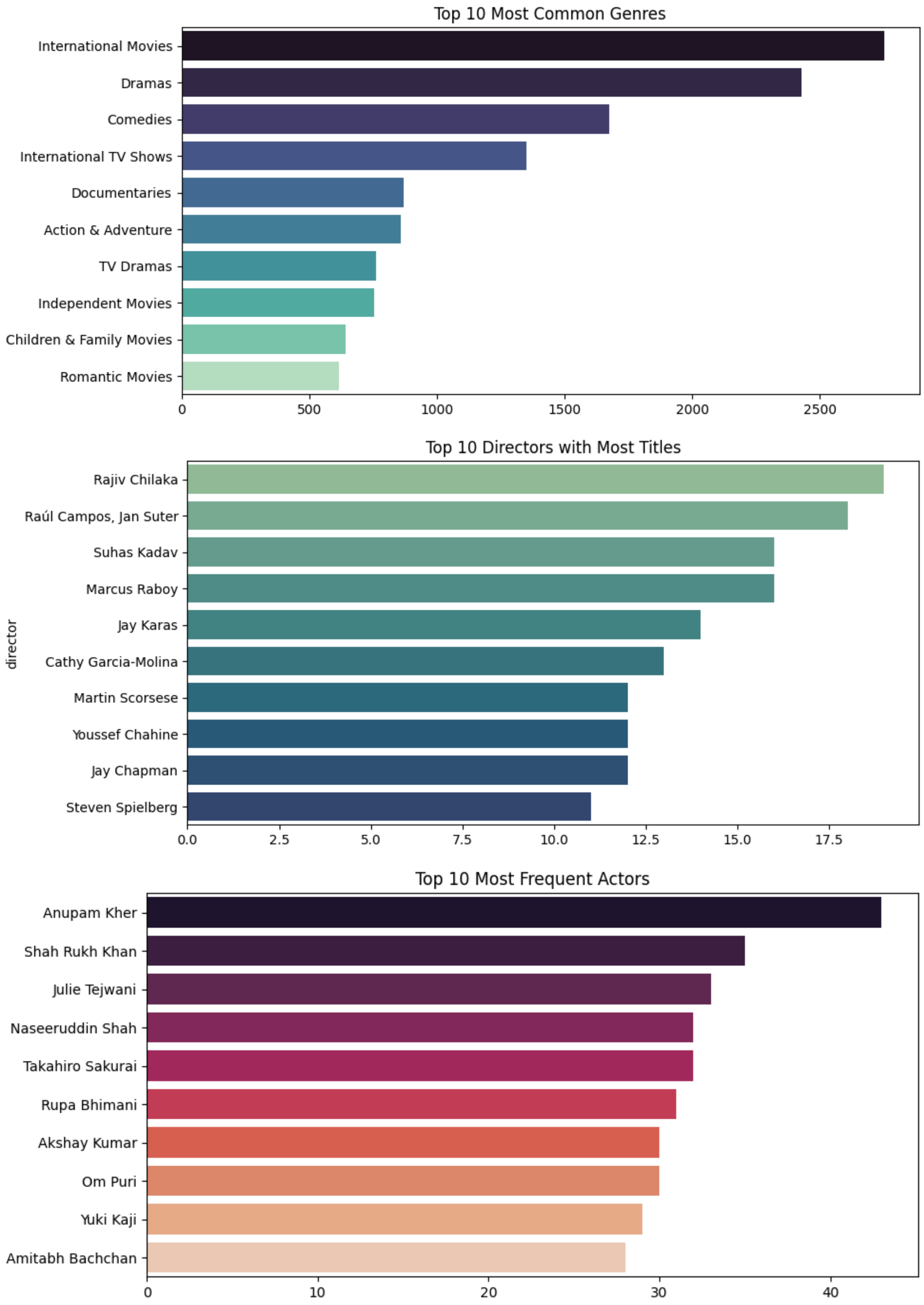
```
In [6]: # 5. Duration & Content Type Analysis
        # a. Movie durations
        movie_durations = df[df['type']=='Movie']['duration'].str.replace(" min","").c
        plt.figure(figsize=(10,5))
        sns.histplot(movie_durations, bins=30, kde=True, color="red")
        plt.title("Distribution of Movie Durations (Minutes)")
        plt.show()

        # b. TV show seasons
        tv_seasons = df[df['type']=='TV Show']['duration'].str.replace(" Season","").s
        plt.figure(figsize=(8,4))
        sns.countplot(x=tv_seasons, hue=tv_seasons, palette="coolwarm", legend=False)
        plt.title("Number of Seasons in TV Shows")
        plt.show()
```





In [7]:

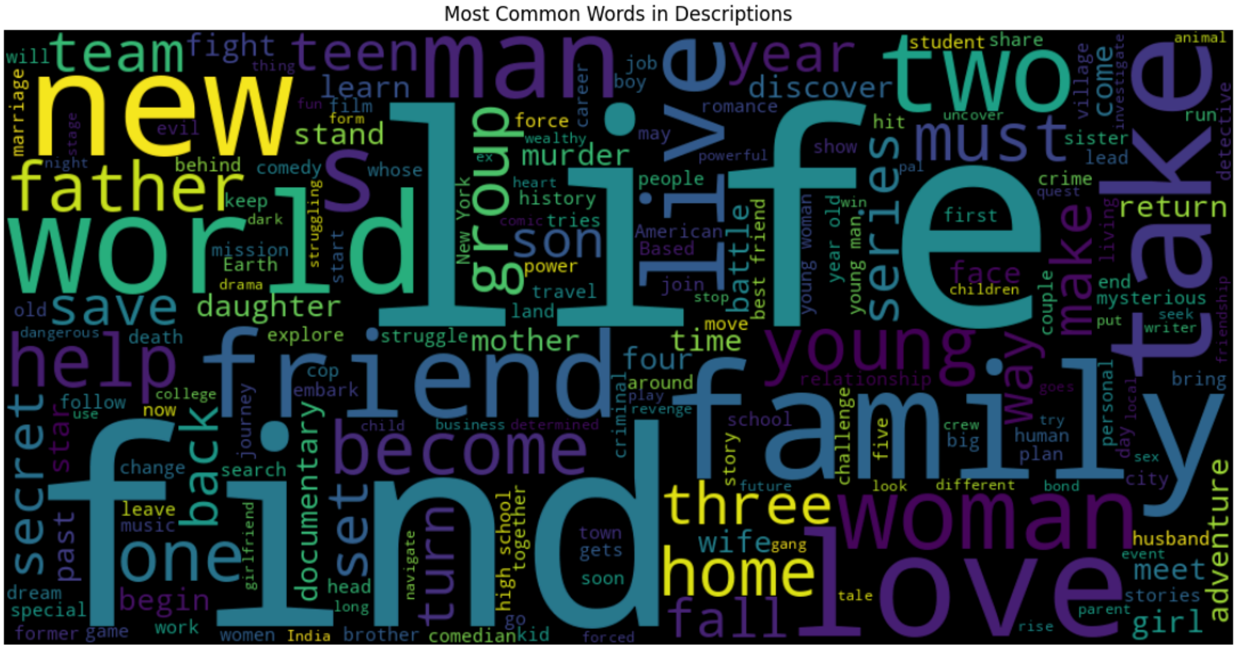
```
from collections import Counter

# 6. Genre & Categories Analysis
# Split listed_in column (genres/categories)
genres = df['listed_in'].dropna().str.split(', ')
all_genres = [g for sub in genres for g in sub]
top10_genres = Counter(all_genres).most_common(10)

plt.figure(figsize=(10,5))
sns.barplot(x=[g[1] for g in top10_genres], y=[g[0] for g in top10_genres],
            hue=[g[0] for g in top10_genres], palette="mako", legend=False)
plt.title("Top 10 Most Common Genres")
plt.show()

# Directors with most titles
top_directors = df['director'].dropna().value_counts().head(10)
plt.figure(figsize=(10,5))
sns.barplot(x=top_directors.values, y=top_directors.index,
            hue=top_directors.index, palette="crest", legend=False)
plt.title("Top 10 Directors with Most Titles")
plt.show()

# Actors with most appearances
cast = df['cast'].dropna().str.split(', ')
all_cast = [c.strip() for sub in cast for c in sub]
top_actors = Counter(all_cast).most_common(10)

plt.figure(figsize=(10,5))
sns.barplot(x=[a[1] for a in top_actors], y=[a[0] for a in top_actors],
            hue=[a[0] for a in top_actors], palette="rocket", legend=False)
plt.title("Top 10 Most Frequent Actors")
plt.show()
```



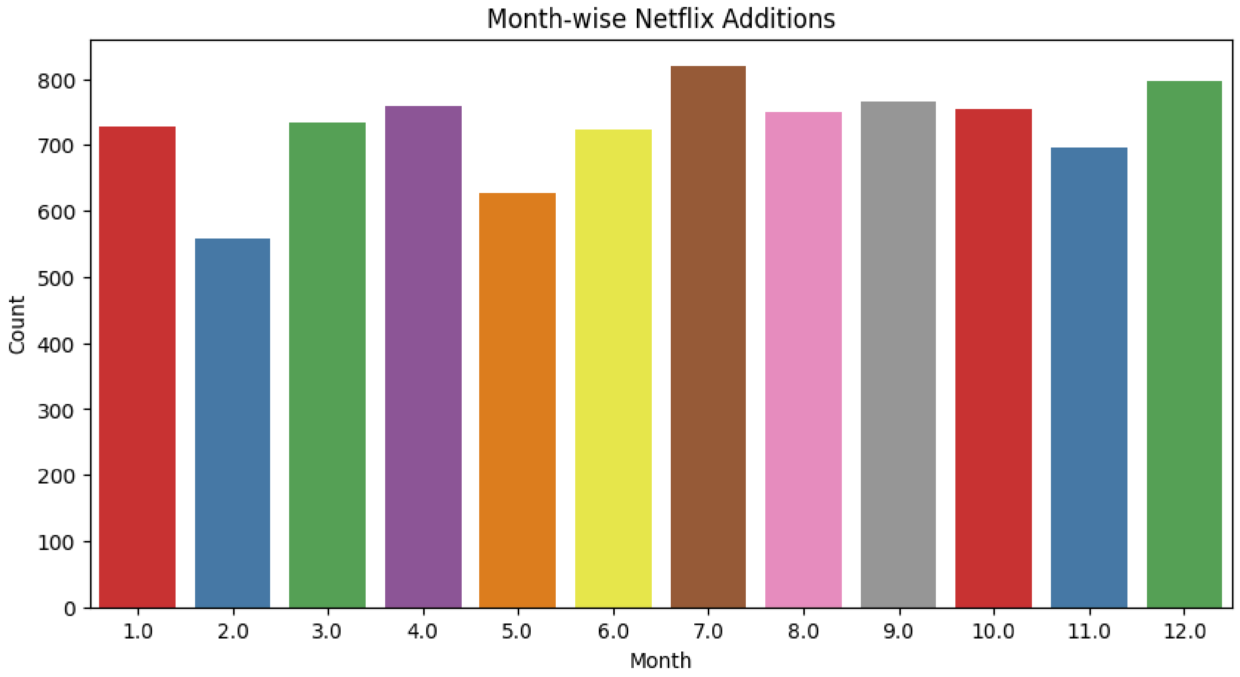
In [ ]:

```
text = " ".join(desc for desc in df['description'].dropna())
```

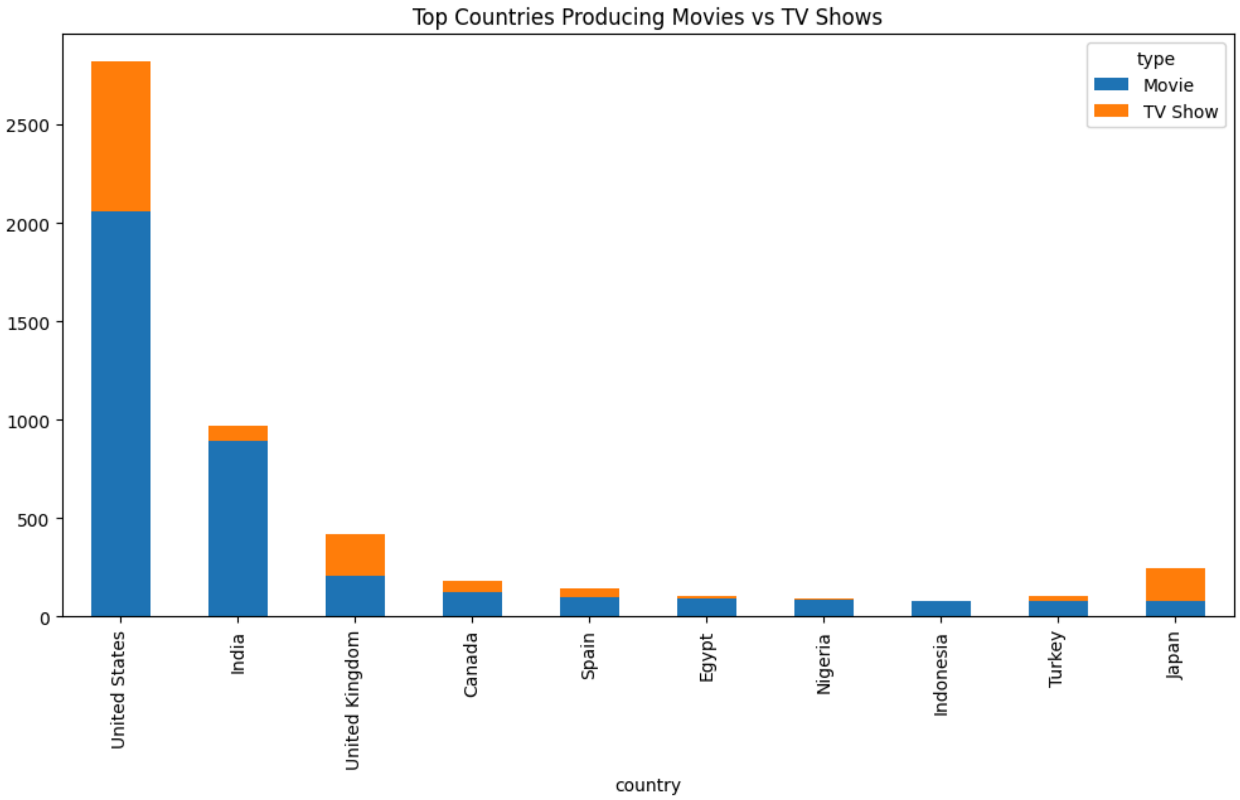
```
wordcloud = WordCloud(width=1000, height=500, background_color="black").genera
plt.figure(figsize=(15,7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("Most Common Words in Descriptions")
plt.show()
```



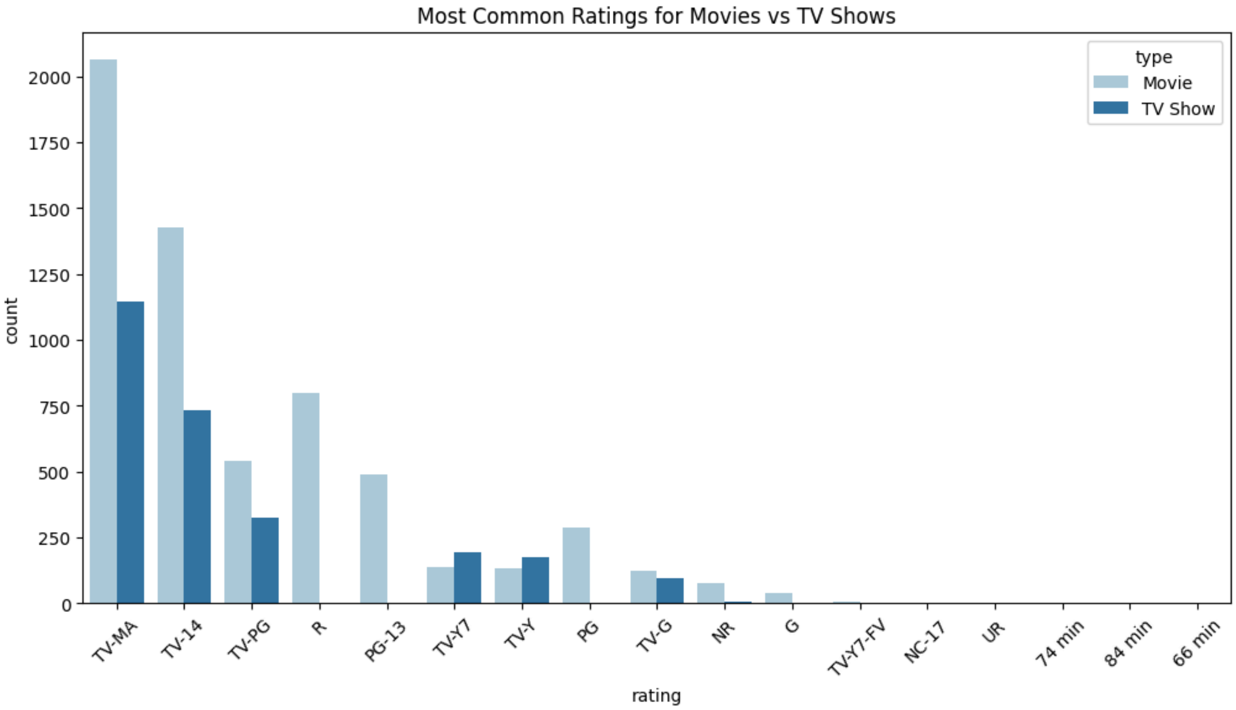
```
In [8]: #8. Month-wise Additions (Not available, since no date_added)
        df['month_added'] = df['date_added'].dt.month
        plt.figure(figsize=(10,5))
        sns.countplot(x=df['month_added'], hue=df['month_added'], palette="Set1", lege
        plt.title("Month-wise Netflix Additions")
        plt.xlabel("Month")
        plt.ylabel("Count")
        plt.show()
```



In [ ]:
```
# 9. Countries producing more Movies vs TV Shows

country_type = df.groupby(['country','type']).size().unstack().fillna(0).sort_
country_type.plot(kind="bar", stacked=True, figsize=(12,6))
plt.title("Top Countries Producing Movies vs TV Shows")
plt.show()
```



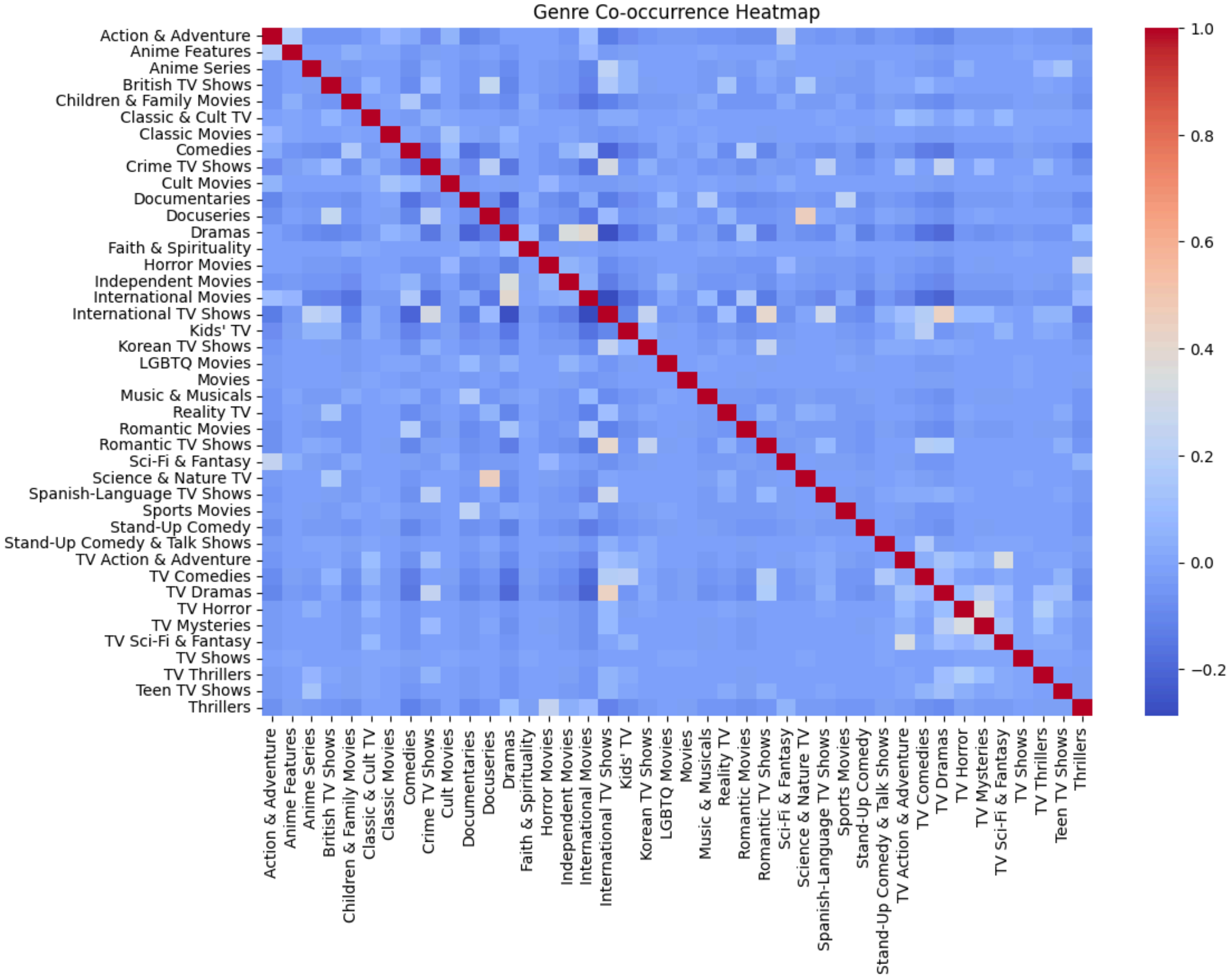
```
In [ ]: # 10. Most Common Ratings (Age certifications) for Movies vs TV Shows

        plt.figure(figsize=(12,6))
        sns.countplot(data=df, x="rating", hue="type", order=df['rating'].value_counts
        plt.title("Most Common Ratings for Movies vs TV Shows")
        plt.xticks(rotation=45)
        plt.show()
```



```
In [ ]: # 11. Oldest & Newest Titles
        print("Oldest Title:\n", df[df['release_year']==df['release_year'].min()][['ti
        print("\nNewest Title:\n", df[df['release_year']==df['release_year'].max()][['
```

```
Oldest Title:
                                   title  release_year
4250  Pioneers: First Women Filmmakers*          1925

Newest Title:
                           title  release_year
1                  Blood & Water          2021
2                      Ganglands          2021
3          Jailbirds New Orleans          2021
4                   Kota Factory          2021
5                  Midnight Mass          2021
...                          ...           ...
1468  What Happened to Mr. Cha?          2021
1551                       Hilda          2021
1696                Polly Pocket          2021
2920                Love Is Blind          2021
8437      The Netflix Afterparty          2021

[592 rows x 2 columns]
```

```
In [ ]: # 12. Genre Correlation Heatmap
        genre_dummies = df['listed_in'].str.get_dummies(sep=', ')
        plt.figure(figsize=(12,8))
        sns.heatmap(genre_dummies.corr(), cmap="coolwarm")
        plt.title("Genre Co-occurrence Heatmap")
        plt.show()
```



In [1]:

```
# Step 1: Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error, mean_absolute_error
import math
```

In [2]:

```
# Step 2: Load Dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/airline-pas
df = pd.read_csv(url, parse_dates=["Month"], index_col="Month")

print("First 5 rows of dataset:")
print(df.head())
```

```
First 5 rows of dataset:
            Passengers
Month
1949-01-01         112
1949-02-01         118
1949-03-01         132
1949-04-01         129
1949-05-01         121
```

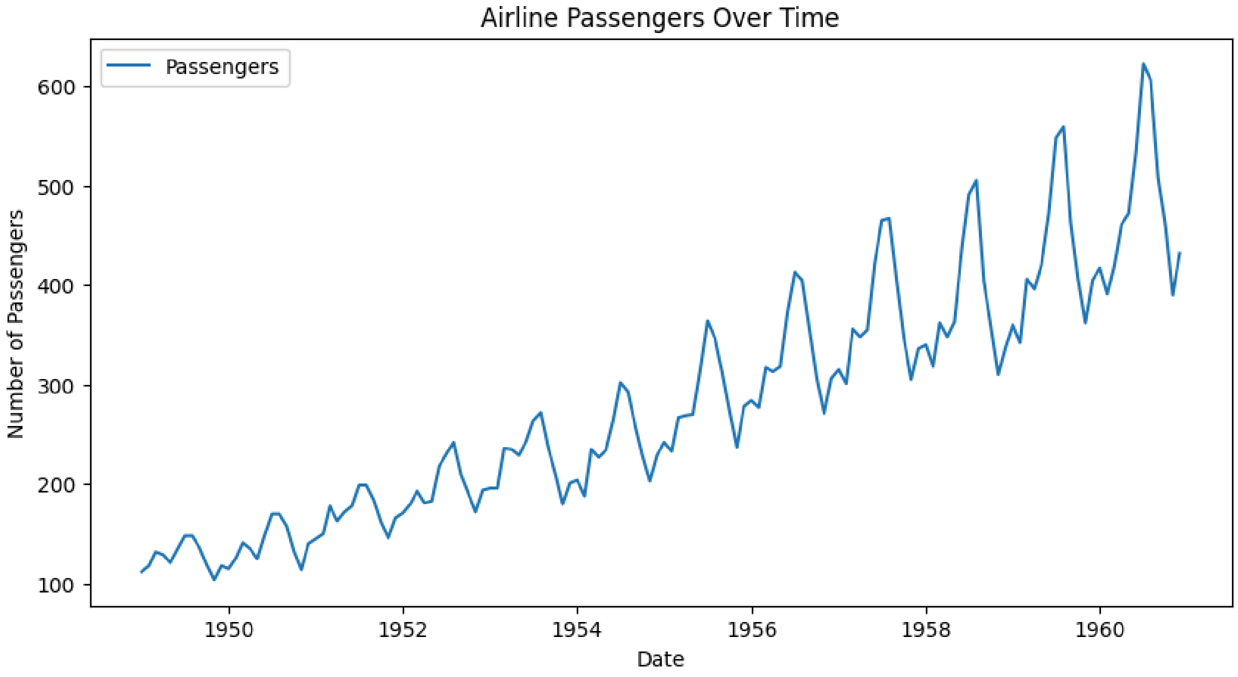
In [3]:

```
# Step 3: Explore Dataset
print("\nDataset Info:")
print(df.info())

print("\nSummary Statistics:")
print(df.describe())
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 144 entries, 1949-01-01 to 1960-12-01
Data columns (total 1 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Passengers  144 non-null    int64
dtypes: int64(1)
memory usage: 2.2 KB
None

Summary Statistics:
       Passengers
count  144.000000
mean   280.298611
std    119.966317
min    104.000000
25%    180.000000
50%    265.500000
75%    360.500000
max    622.000000
```

In [4]:

```python
# Step 4: Plot series to inspect trend/seasonality
plt.figure(figsize=(10,5))
plt.plot(df, label="Passengers")
plt.title("Airline Passengers Over Time")
plt.xlabel("Date")
plt.ylabel("Number of Passengers")
plt.legend()
plt.show()
```



In [5]:

```python
# Step 5: Stationarity Check - Augmented Dickey Fuller Test
```

```
def adf_test(series):
    result = adfuller(series)
    print("ADF Statistic: ", result[0])
    print("p-value: ", result[1])
    print("Critical Values:")
    for key, value in result[4].items():
        print(f"\t{key}: {value}")
    if result[1] <= 0.05:
        print("Series is Stationary")
    else:
        print("Series is Non-Stationary")

print("\nADF Test on Original Series:")
adf_test(df["Passengers"])
```

```
ADF Test on Original Series:
ADF Statistic:  0.8153688792060498
p-value:  0.991880243437641
Critical Values:
	1%: -3.4816817173418295
	5%: -2.8840418343195267
	10%: -2.578770059171598
Series is Non-Stationary
```

In [6]:

```
# Step 6 & 7: Differencing if Non-Stationary
df_diff = df["Passengers"].diff().dropna()

print("\nADF Test after Differencing (d=1):")
adf_test(df_diff)

# Step 8: If still non-stationary, you can apply second differencing
# (here usually d=1 is enough for this dataset)
```

```
ADF Test after Differencing (d=1):
ADF Statistic:  -2.8292668241700047
p-value:  0.05421329028382478
Critical Values:
	1%: -3.4816817173418295
	5%: -2.8840418343195267
	10%: -2.578770059171598
Series is Non-Stationary
```

In [7]:

```
# Step 9: Split dataset into train and test
train_size = int(len(df) * 0.8)
train, test = df.iloc[:train_size], df.iloc[train_size:]

print(f"\nTrain size: {len(train)}, Test size: {len(test)}")
```

```
Train size: 115, Test size: 29
```

In [8]:

```
# Step 10: Fit ARIMA Model
# Here we use (p,d,q) = (2,1,2) as an example, you can tune
model = ARIMA(train, order=(2,1,2))
model_fit = model.fit()
```

```
print(model_fit.summary())
```

```
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency MS w
ill be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency MS w
ill be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency MS w
ill be used.
  self._init_dates(dates, freq)
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:             Passengers   No. Observations:                  115
Model:                 ARIMA(2, 1, 2)   Log Likelihood                -523.758
Date:                Tue, 23 Sep 2025   AIC                           1057.516
Time:                        09:38:14   BIC                           1071.197
Sample:                    01-01-1949   HQIC                          1063.069
                         - 07-01-1958
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.3280      0.145      2.268      0.023       0.045       0.611
ar.L2          0.2521      0.165      1.528      0.126      -0.071       0.575
ma.L1         -0.0125      0.109     -0.114      0.909      -0.227       0.202
ma.L2         -0.7544      0.130     -5.812      0.000      -1.009      -0.500
sigma2       568.4920    103.877      5.473      0.000     364.897     772.087
================================================================================
====
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):
3.39
Prob(Q):                              0.90   Prob(JB):
0.18
Heteroskedasticity (H):               5.24   Skew:
0.11
Prob(H) (two-sided):                  0.00   Kurtosis:
2.19
================================================================================
====

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```

In [9]:

```
# Step 11: Forecast on test dataset
forecast = model_fit.forecast(steps=len(test))
forecast = pd.Series(forecast, index=test.index)
```

In [10]:

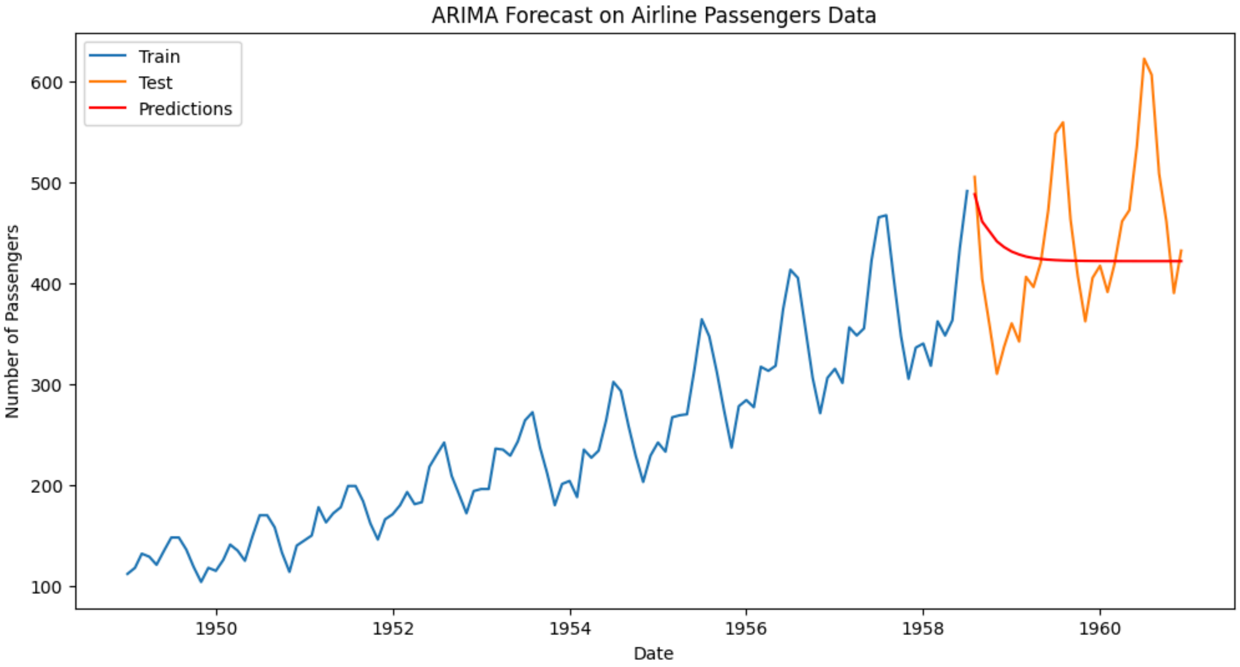
```
# Step 12: Compute metrics
```

```
mse = mean_squared_error(test, forecast)
mae = mean_absolute_error(test, forecast)
rmse = math.sqrt(mse)

print("\nEvaluation Metrics:")
print(f"MSE: {mse}")
print(f"MAE: {mae}")
print(f"RMSE: {rmse}")
```

```
Evaluation Metrics:
MSE: 6808.397034418323
MAE: 63.54531127532635
RMSE: 82.51301130378361
```

In [11]:

```
# Step 13: Plot Train, Test, and Predictions
plt.figure(figsize=(12,6))
plt.plot(train, label="Train")
plt.plot(test, label="Test")
plt.plot(forecast, label="Predictions", color="red")
plt.title("ARIMA Forecast on Airline Passengers Data")
plt.xlabel("Date")
plt.ylabel("Number of Passengers")
plt.legend()
plt.show()
```


ARIMA Forecast on Airline Passengers Data
Train
Test
Predictions
Number of Passengers
Date