BABU BANARASI DAS UNIVERSITY

LUCKNOW

Session : 2024-2025

SCHOOL OF COMPUTER APPLICATION

ASSIGNMENT

ON

Artificial Intelligence (MCADSN13202)

Submittd By:

Mohd Yousuf Arif

MCADS21-3rd Semester

Roll No:1240259031

Submitted To:

Mr Ankit Soni

Netflix Data Analysis Project

Installing Pandas

!pip install pandas

```
Requirement already satisfied: pandas in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (2.3.3)
Requirement already satisfied: numpy>=1.26.0 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from pandas) (2.3.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from pandas) (2.9.
0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from python-dateutil>=2.8.2->pand
as) (1.17.0)
```

import pandas as pd

print(pd.__version__)

2.3.3

Loading The Data

df= pd.read_csv(r"C:\Users\Lenovo\Downloads\netflix.csv")

df.head()

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...

```
df.drop_duplicates(inplace=True)
```

Cleaning the Data

```
df.drop_duplicates(inplace=True)

# Check missing values

df.isnull().sum()


# Fill or drop based on column importance

df['director'].fillna('Unknown', inplace=True)

df['cast'].fillna('Unknown', inplace=True)

df['country'].fillna('Unknown', inplace=True)

df['rating'].fillna('Not Rated', inplace=True)

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

df['duration'].fillna('Unknown', inplace=True)
```

Installing Matplotlib

```
pip install matplotlib ipympl
```

```
Requirement already satisfied: matplotlib in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (3.10.7)
Requirement already satisfied: ipympl in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (0.9.8)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from matplotlib) (4.60.
1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from matplotlib) (2.3.3)
Requirement already satisfied: packaging>=20.0 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=3 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from matplotlib) (2.
9.0.post0)
Requirement already satisfied: ipython<10 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipympl) (9.6.0)
Requirement already satisfied: ipywidgets<9,>=7.6.0 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipympl) (8.1.7)
Requirement already satisfied: traitlets<6 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipympl) (5.14.3)
Requirement already satisfied: colorama in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipython<10->ipympl) (0.4.6)
Requirement already satisfied: decorator in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipython<10->ipympl) (5.2.1)
Requirement already satisfied: ipython-pygments-lexers in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipython<10->i
pympl) (1.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipython<10->ipympl) (0.19.
2)
Requirement already satisfied: matplotlib-inline in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipython<10->ipympl)
(0.1.7)
Requirement already satisfied: prompt_toolkit<3.1.0,>=3.0.41 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipython
<10->ipympl) (3.0.52)
Requirement already satisfied: pygments>=2.4.0 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipython<10->ipympl)
(2.19.2)
Requirement already satisfied: stack_data in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipython<10->ipympl) (0.6.
3)
Requirement already satisfied: comm>=0.1.3 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipywidgets<9,>=7.6.0->ipy
mpl) (0.2.3)
Requirement already satisfied: widgetsnbextension~=4.0.14 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipywidgets
<9,>=7.6.0->ipympl) (4.0.14)
Requirement already satisfied: jupyterlab_widgets~=3.0.15 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from ipywidgets
<9,>=7.6.0->ipympl) (3.0.15)
Requirement already satisfied: wcwidth in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from prompt_toolkit<3.1.0,>=3.0.41
->ipython<10->ipympl) (0.2.14)
Requirement already satisfied: parso<0.9.0,>=0.8.4 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from jedi>=0.16->ipyth
on<10->ipympl) (0.8.5)
Requirement already satisfied: six>=1.5 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from python-dateutil>=2.7->matplo
tlib) (1.17.0)
Requirement already satisfied: executing>=1.2.0 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from stack_data->ipython<
10->ipympl) (2.2.1)
Requirement already satisfied: asttokens>=2.1.0 in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from stack_data->ipython<
10->ipympl) (3.0.0)
Requirement already satisfied: pure-eval in c:\users\lenovo\appdata\local\programs\python\python314\lib\site-packages (from stack_data->ipython<10->ipy
mpl) (0.2.3)
Note: you may need to restart the kernel to use updated packages.
```

Importing Matplotlib

import matplotlib.pyplot as plt

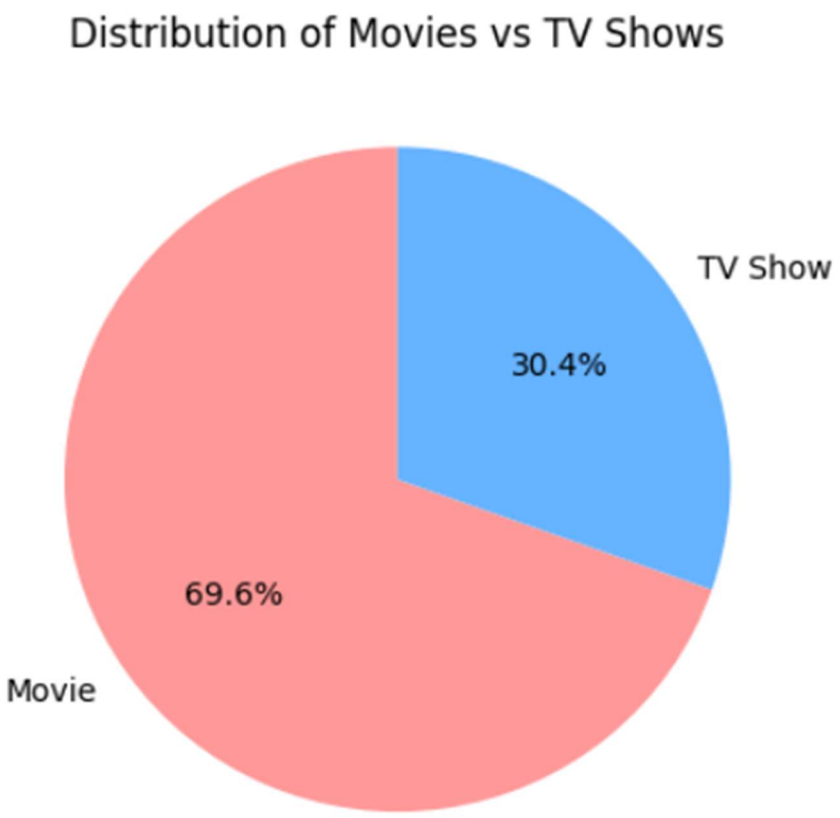
1.Content Strategy

Q1. What is the ratio of movies vs TV shows on Netflix?

Insight: Movies make up approximately 69.6% of Netflix's catalog, while TV Shows account for 30.4%.

Recommendation: Expand TV show offerings to boost binge-watching engagement and diversify content formats.

```
content_type = df['type'].value_counts()

content_type.plot(kind='pie', autopct='%1.1f%%', colors=['#ff9999','#66b3ff'], startangle=90)

plt.title('Distribution of Movies vs TV Shows')

plt.ylabel('')

plt.show()
```



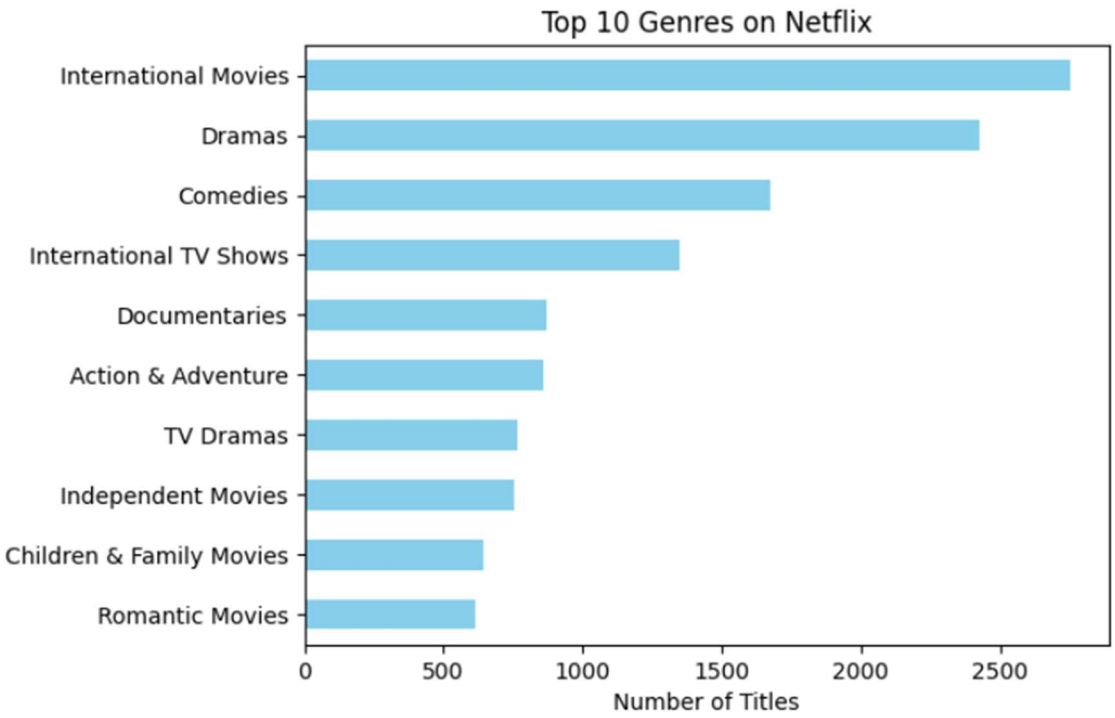
Insight: This shows Netflix's investment focus—whether it's more into movies or series.

Q2. Which genres are most popular on Netflix globally?

Insight: International Movies, Dramas, Comedies, and Documentaries are the most frequent genres.

Recommendation: Prioritize these genres in future content investments to maximize global appeal.

```
from collections import Counter

genres = df['listed_in'].dropna().str.split(', ')

genre_list = [genre for sublist in genres for genre in sublist]

top_genres =

pd.Series(Counter(genre_list)).sort_values(ascending=False).head(10)


top_genres.plot(kind='barh', color='skyblue')

plt.title('Top 10 Genres on Netflix')

plt.xlabel('Number of Titles')

plt.gca().invert_yaxis()

plt.show()
```



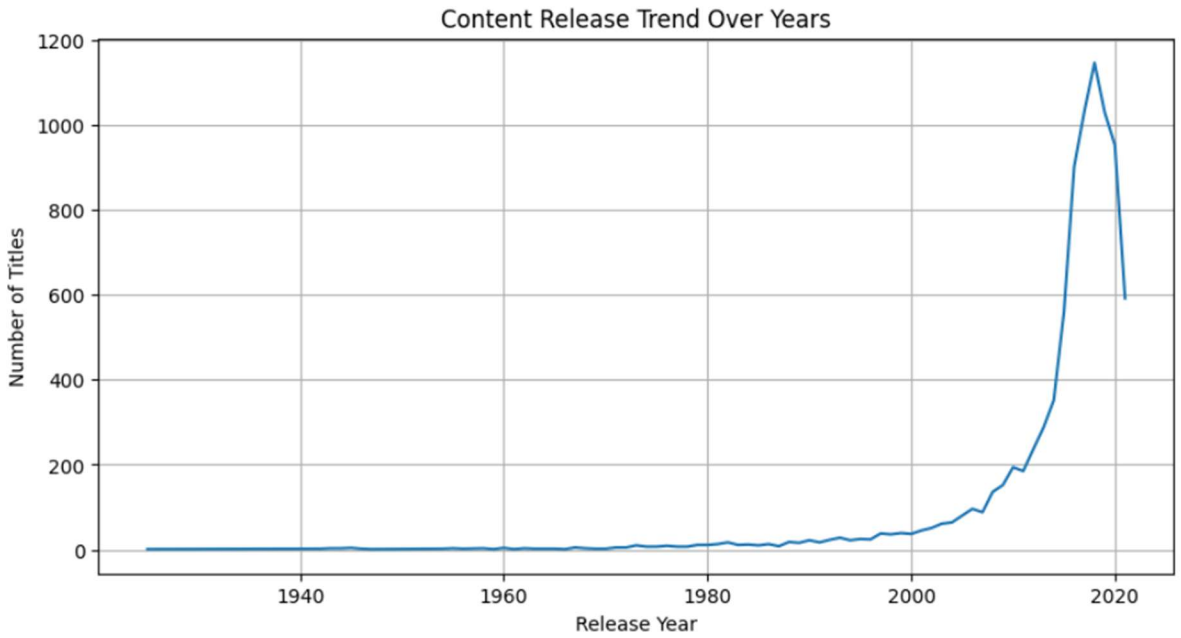
Insight: Helps Netflix prioritize genre acquisition and production.

Q3. Which years saw the highest release of content on Netflix?

Insight: Content releases peaked between 2018 and 2021, showing Netflix's aggressive growth phase.

Recommendation: Analyze successful releases from peak years to guide future production strategies.

```
df['release_year'].value_counts().sort_index().plot(kind='line', figsize=(10,5))

plt.title('Content Release Trend Over Years')

plt.xlabel('Release Year')

plt.ylabel('Number of Titles')

plt.grid(True)

plt.show()
```



Insight: Reveals aggressive content addition years.

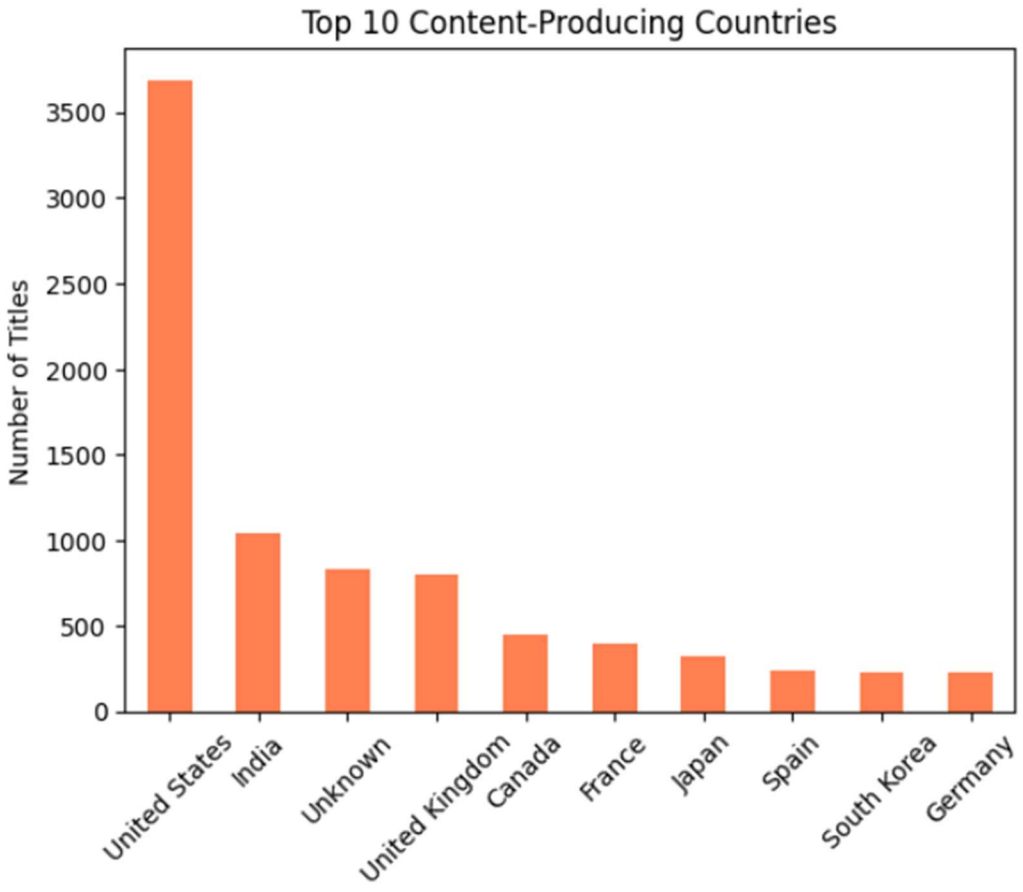
Q4. Which countries produce the most Netflix content?

Insight: The United States leads, followed by India, UK, Canada, and France. These countries are major content contributors.

Recommendation: Strengthen partnerships in these regions and explore emerging markets to diversify content sources.

```
country_series = df['country'].dropna().str.split(', ')

country_list = [c for sublist in country_series for c in sublist]

top_countries =
pd.Series(Counter(country_list)).sort_values(ascending=False).head(10)


top_countries.plot(kind='bar', color='coral')
```

```
plt.title('Top 10 Content-Producing Countries')
plt.ylabel('Number of Titles')
plt.xticks(rotation=45)
plt.show()
```



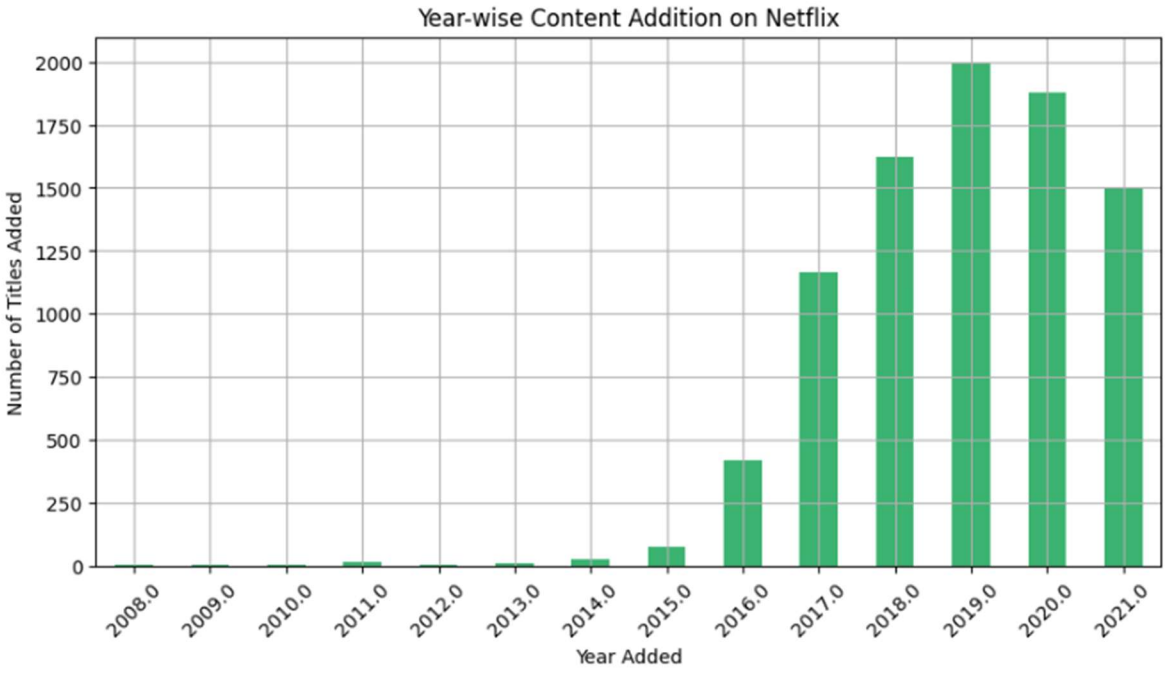
05.How has the trend of adding new content evolved year by year?

Insight : Netflix's content addition accelerated sharply post-2015, peaking in 2019–2021.

Recommendation: Analyze what genres and countries dominated during peak years to replicate success.

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

```
df['year_added'] = df['date_added'].dt.year


yearly_additions = df['year_added'].value_counts().sort_index()

yearly_additions.plot(kind='bar', figsize=(10,5),
color='mediumseagreen')


plt.title('Year-wise Content Addition on Netflix')

plt.xlabel('Year Added')

plt.ylabel('Number of Titles Added')

plt.xticks(rotation=45)

plt.grid(True)

plt.show()
```



2.User Demographics & Targeting

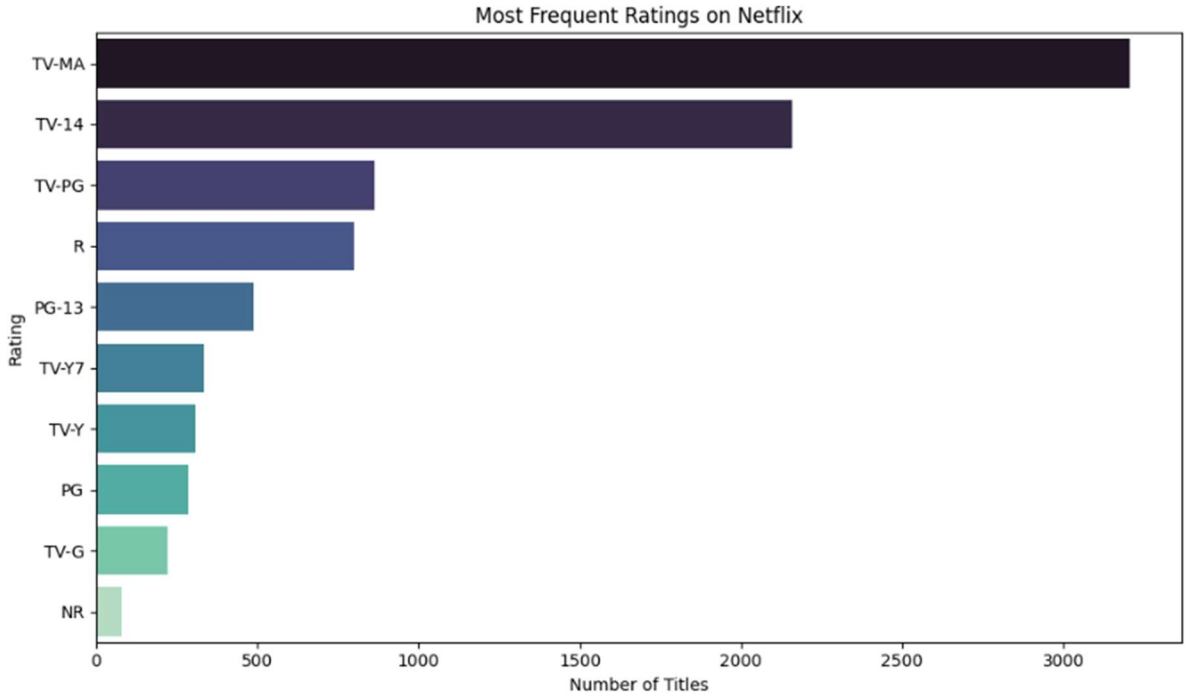
6.Which ratings (e.g., TV-MA, PG, etc.) are most frequent on Neflix?

Insight: TV-MA is the most frequent rating, followed by TV-14 and PG.

Recommendation: Continue investing in mature content for adult audiences.

```
rating_counts = df['rating'].value_counts().head(10)


plt.figure(figsize=(10,6))
sns.barplot(x=rating_counts.values, y=rating_counts.index, palette="mako")
plt.title("Most Frequent Ratings on Netflix")
plt.xlabel("Number of Titles")
plt.ylabel("Rating")
plt.tight_layout()
```

plt.show()



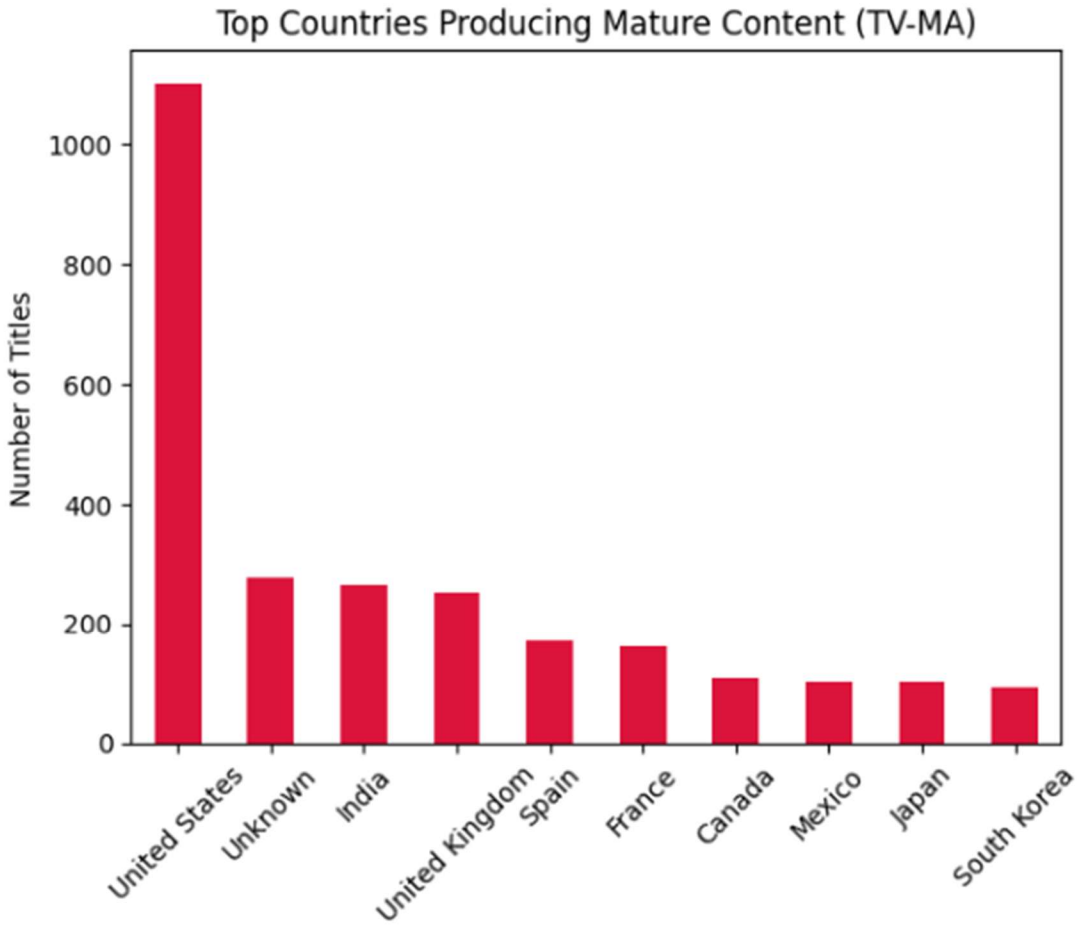
Q7. Do some countries tend to produce more mature content (TV-MA)?

Insight: The U.S., UK, and India show a high concentration of TV-MA content.

Recommendation: Tailor marketing and recommendations based on regional maturity preferences.

```
# Filter for mature content

mature_df = df[df['rating'] == 'TV-MA'].copy()


# Expand country list

mature_df['country_list'] = mature_df['country'].dropna().apply(lambda x: [i.strip() for i in x.split(',')])
```

```
# Flatten and count
from collections import Counter
mature_countries = [country for sublist in mature_df['country_list'] for country in sublist]
top_mature = pd.Series(Counter(mature_countries)).sort_values(ascending=False).head(10)

# Plot
top_mature.plot(kind='bar', color='crimson')
plt.title('Top Countries Producing Mature Content (TV-MA)')
plt.ylabel('Number of Titles')
plt.xticks(rotation=45)
plt.show()
```



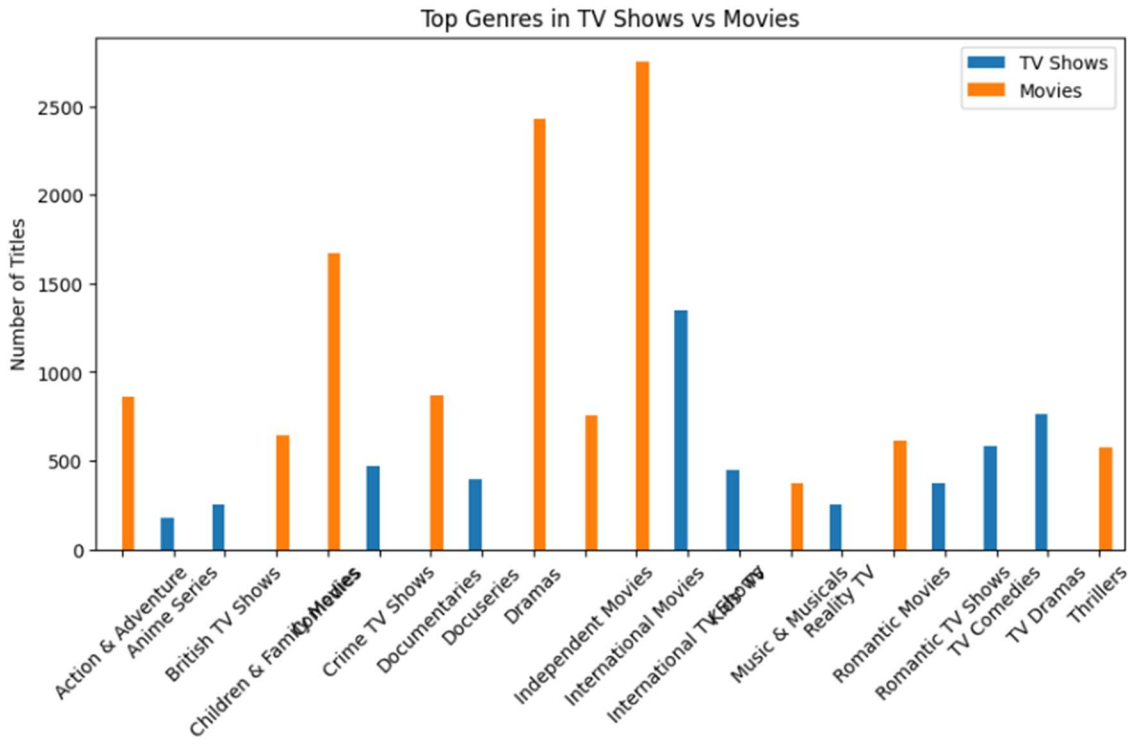
Q8. Which genres are more associated with TV shows?

Insight: TV Shows favor genres like Action & Adventure, Anime Series, and Docuseries, while Movies lean toward Dramas and Romantic genres.

Recommendation: Use genre-type associations to personalize recommendations and improve user satisfaction.

```
# Create genre lists

df['genre_list'] = df['listed_in'].dropna().apply(lambda x: [i.strip() for i in x.split(',')])
```

```
# Separate by type
tv_genres = df[df['type'] == 'TV Show']['genre_list']
movie_genres = df[df['type'] == 'Movie']['genre_list']

# Flatten and count
tv_genre_counts = pd.Series([g for sublist in tv_genres for g in sublist]).value_counts().head(10)
movie_genre_counts = pd.Series([g for sublist in movie_genres for g in sublist]).value_counts().head(10)

# Combine into DataFrame
genre_compare = pd.DataFrame({'TV Shows': tv_genre_counts, 'Movies': movie_genre_counts}).fillna(0)

# Plot
genre_compare.plot(kind='bar', figsize=(10,5))
plt.title('Top Genres in TV Shows vs Movies')
plt.ylabel('Number of Titles')
plt.xticks(rotation=45)
plt.show()
```



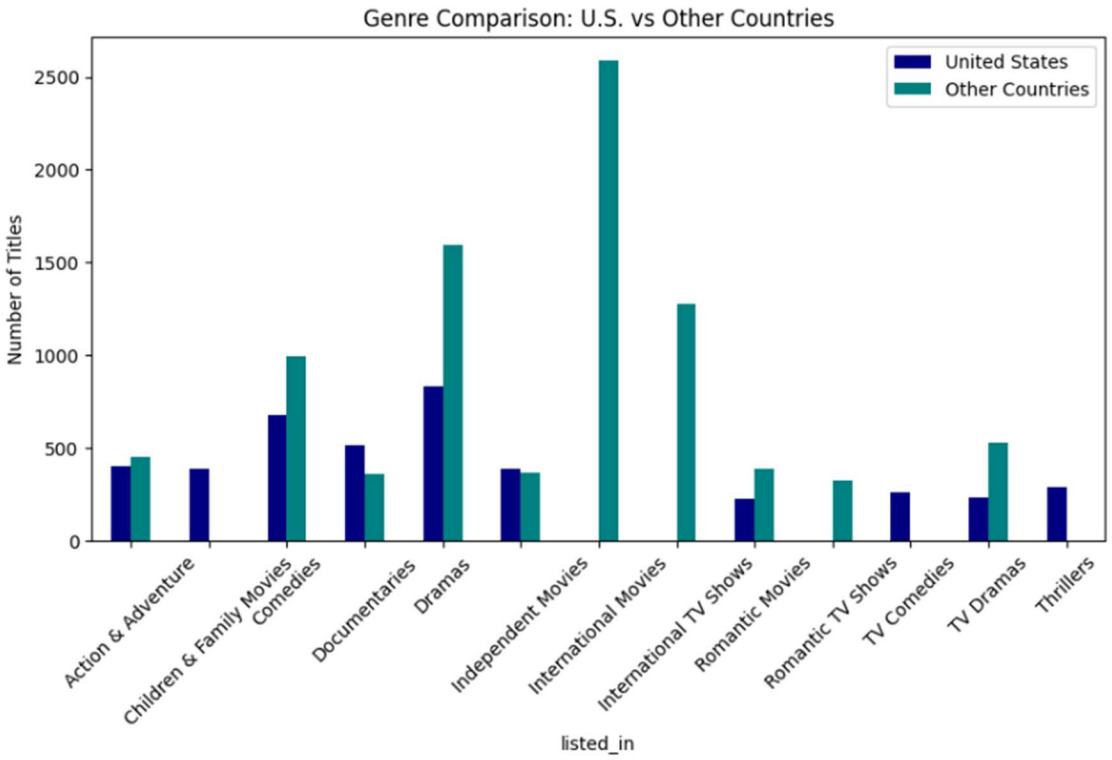
Q9. Which genres dominate the U.S. vs other countries?

Insight: U.S. content emphasizes Comedies and Documentaries, while other countries focus on International and Romantic genres.

Recommendation: Localize content strategies based on regional genre preferences to boost engagement.

```
# Filter U.S. and non-U.S. content
us_df = df[df['country'].str.contains('United States', na=False)].copy()
non_us_df = df[~df['country'].str.contains('United States', na=False)].copy()


# Extract genres
us_genres = us_df['listed_in'].dropna().str.split(', ').explode()
non_us_genres = non_us_df['listed_in'].dropna().str.split(', ').explode()
```

```
# Count top genres
us_top = us_genres.value_counts().head(10)
non_us_top = non_us_genres.value_counts().head(10)

# Combine
genre_geo = pd.DataFrame({'United States': us_top, 'Other Countries':
non_us_top}).fillna(0)

# Plot
genre_geo.plot(kind='bar', figsize=(10,5), color=['navy', 'teal'])
plt.title('Genre Comparison: U.S. vs Other Countries')
plt.ylabel('Number of Titles')
plt.xticks(rotation=45)
plt.show()
```



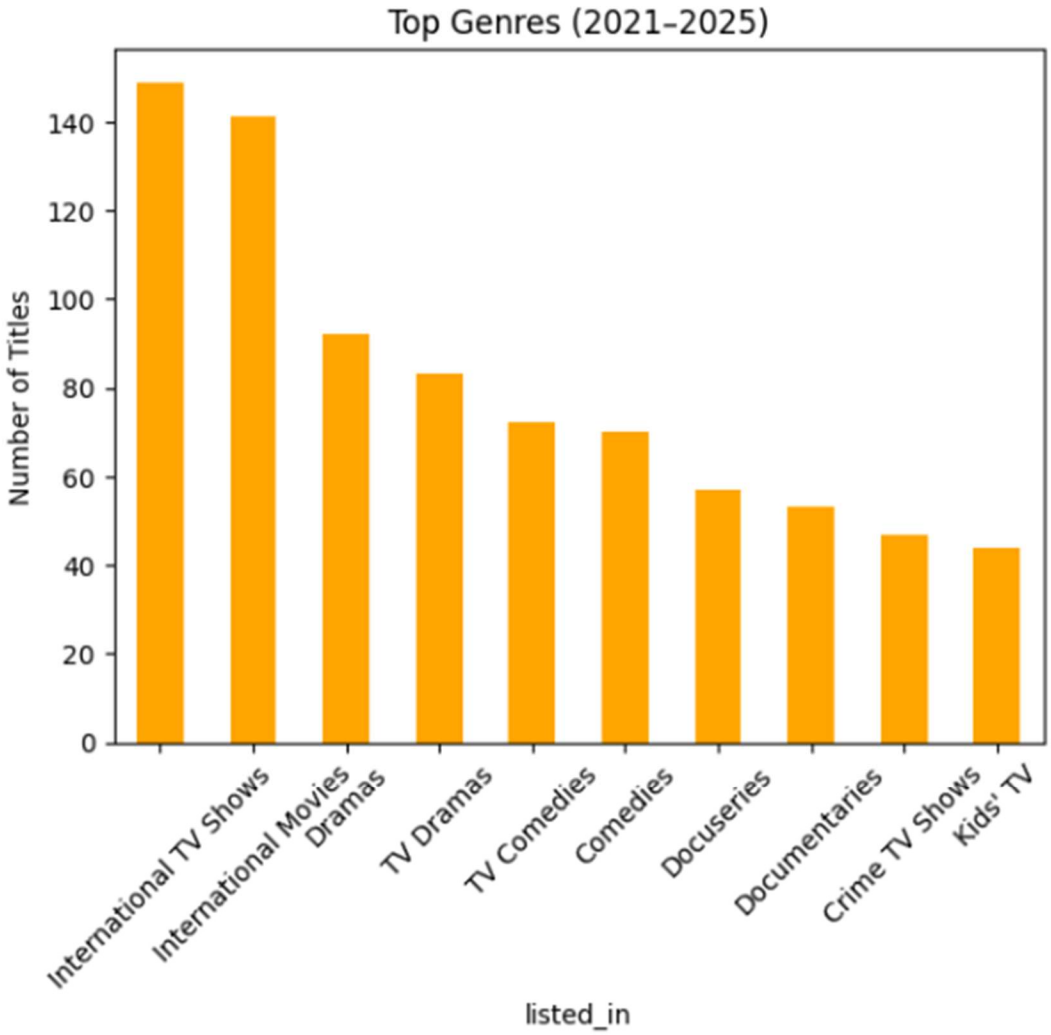
Q10. What genres are most popular in the last 3 years?

Insight: International TV Shows, Dramas, Docuseries, and Crime TV Shows have surged in popularity since 2021.

Recommendation: Invest in trending genres to stay relevant and meet evolving viewer demands.

```
# Filter for recent content

recent_years = df[df['release_year'] >= 2021].copy()


# Extract genres

recent_genres = recent_years['listed_in'].dropna().str.split(', ').explode()


# Count and plot
```

```
recent_genres.value_counts().head(10).plot(kind='bar', color='orange')
plt.title('Top Genres (2021–2025)')
plt.ylabel('Number of Titles')
plt.xticks(rotation=45)
plt.show()
```



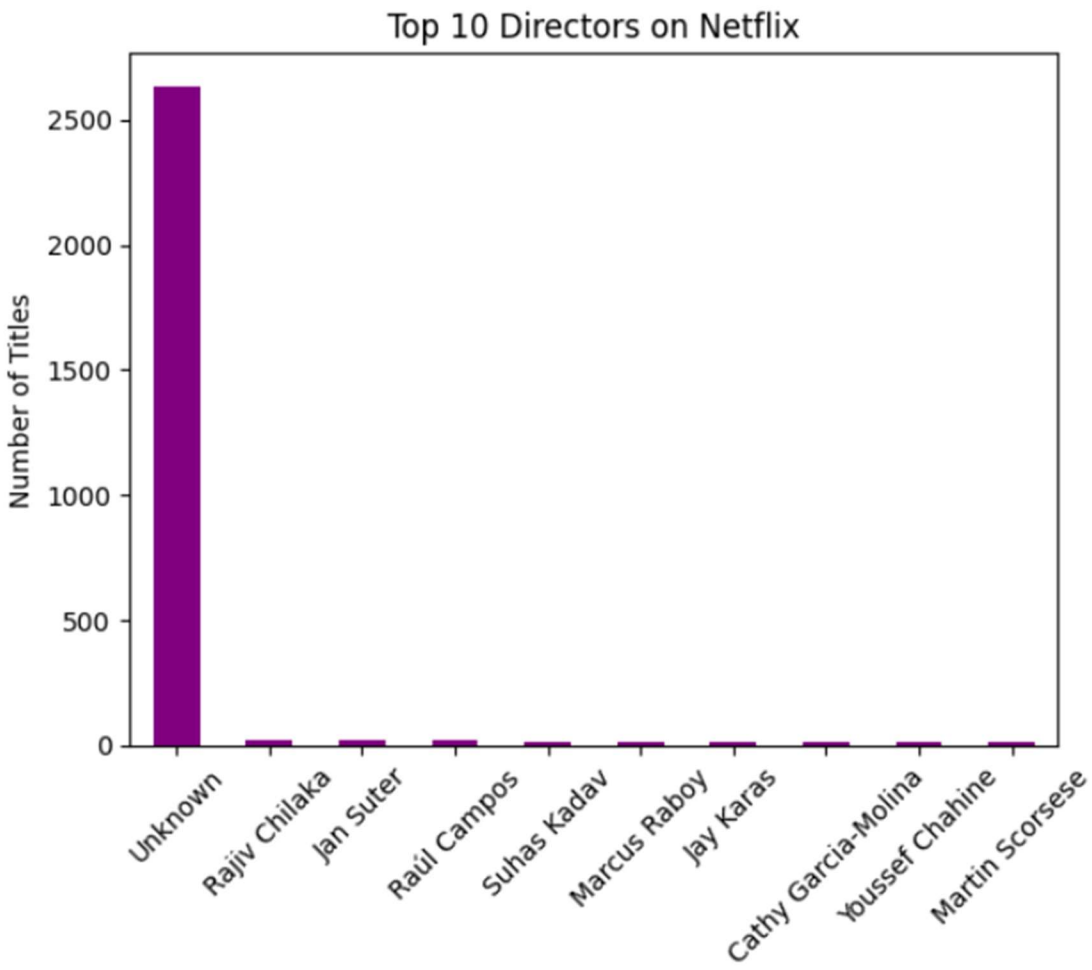
3.Talent Acquisition & Partnership

Q11. Who are the top 10 directors with the most Netflix content?

Insight: Rajiv Chilaka, Jan Suter, Raúl Campos, and Martin Scorsese are among the most prolific directors.

Recommendation: Strengthen collaborations with these directors and promote their work to loyal audiences.

```
top_directors = df['director'].dropna().str.split(', ')

director_list = [d for sublist in top_directors for d in sublist]

pd.Series(Counter(director_list)).sort_values(ascending=False).head(10).plot(kind
='bar', color='purple')

plt.title('Top 10 Directors on Netflix')

plt.ylabel('Number of Titles')

plt.xticks(rotation=45)

plt.show()
```



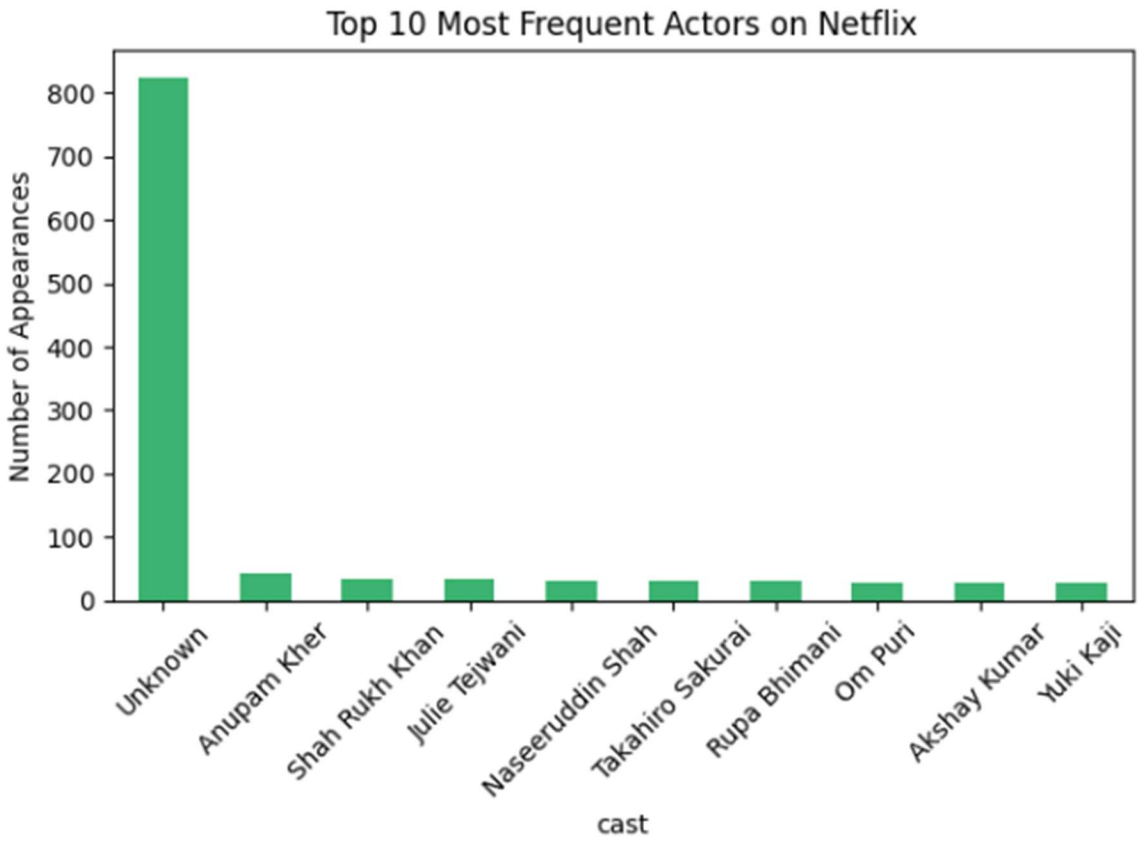
Q12. Which actors appear most frequently in Netflix shows?

Insight: Indian actors like Anupam Kher, Shah Rukh Khan, and Om Puri dominate the list.

Recommendation: Leverage star power in regional marketing campaigns and content promotion.

```
# Drop missing cast entries

df['cast'] = df['cast'].fillna('Unknown')



# Split and flatten actor names
```

```
actor_series = df['cast'].str.split(', ').explode()

# Count top actors
top_actors = actor_series.value_counts().head(10)

# Plot
top_actors.plot(kind='bar', color='mediumseagreen')
plt.title('Top 10 Most Frequent Actors on Netflix')
plt.ylabel('Number of Appearances')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



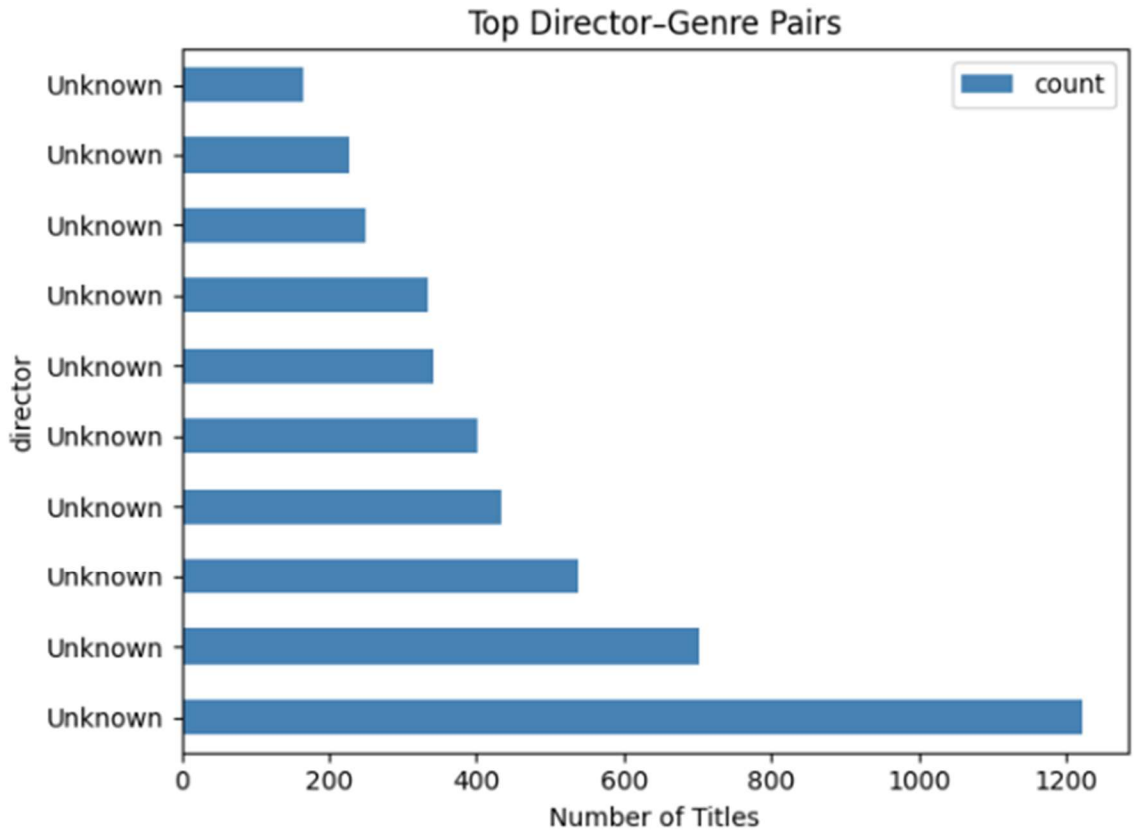
Q13. Which director-genre pairs are most frequent?

Insight: “Unknown” dominates due to missing metadata, but some directors show strong genre specialization.

Recommendation: Improve metadata quality to enable better talent-genre mapping and strategic hiring.

```
df['director'] = df['director'].fillna('Unknown')

df['listed_in'] = df['listed_in'].fillna('Unknown')

df['genre_list'] = df['listed_in'].apply(lambda x: [i.strip() for i in x.split(',')])


# Explode genre list
```

```
exploded_df = df.explode('genre_list')


# Group and count
pair_counts = exploded_df.groupby(['director',
'genre_list']).size().reset_index(name='count')
top_pairs = pair_counts.sort_values(by='count', ascending=False).head(10)


# Plot
import matplotlib.pyplot as plt
top_pairs.plot(kind='barh', x='director', y='count', color='steelblue')
plt.title('Top Director–Genre Pairs')
plt.xlabel('Number of Titles')
plt.tight_layout()
plt.show()
```



Q14. How many titles have unknown directors or cast?

Insight: 2,634 titles lack director info; 825 lack cast info — a significant metadata gap.

Recommendation: Prioritize metadata enrichment to improve searchability and recommendation accuracy.

```
# Count missing or 'Unknown' entries

unknown_directors = df['director'].isin(['Unknown']).sum()

unknown_cast = df['cast'].isin(['Unknown']).sum()


print(f"Titles with unknown directors: {unknown_directors}")
```

print(f"Titles with unknown cast: {unknown_cast}")

```
Titles with unknown directors: 2634
Titles with unknown cast: 825
```

4.Duration and Engagement

Q15. What is the average duration of Movies on Netflix?

Insight: Average movie duration is ~99.6 minutes, aligning with standard feature-length expectations.

Recommendation: Maintain this duration range for optimal viewer retention and satisfaction.

```
movie_df = df[df['type'] == 'Movie']

movie_df['duration_mins'] = movie_df['duration'].str.extract(r'(\d+)').astype(float)

movie_df['duration_mins'].mean()
```

```
[41]: np.float64(99.57718668407311)
```

Insight: Helps define optimal movie length

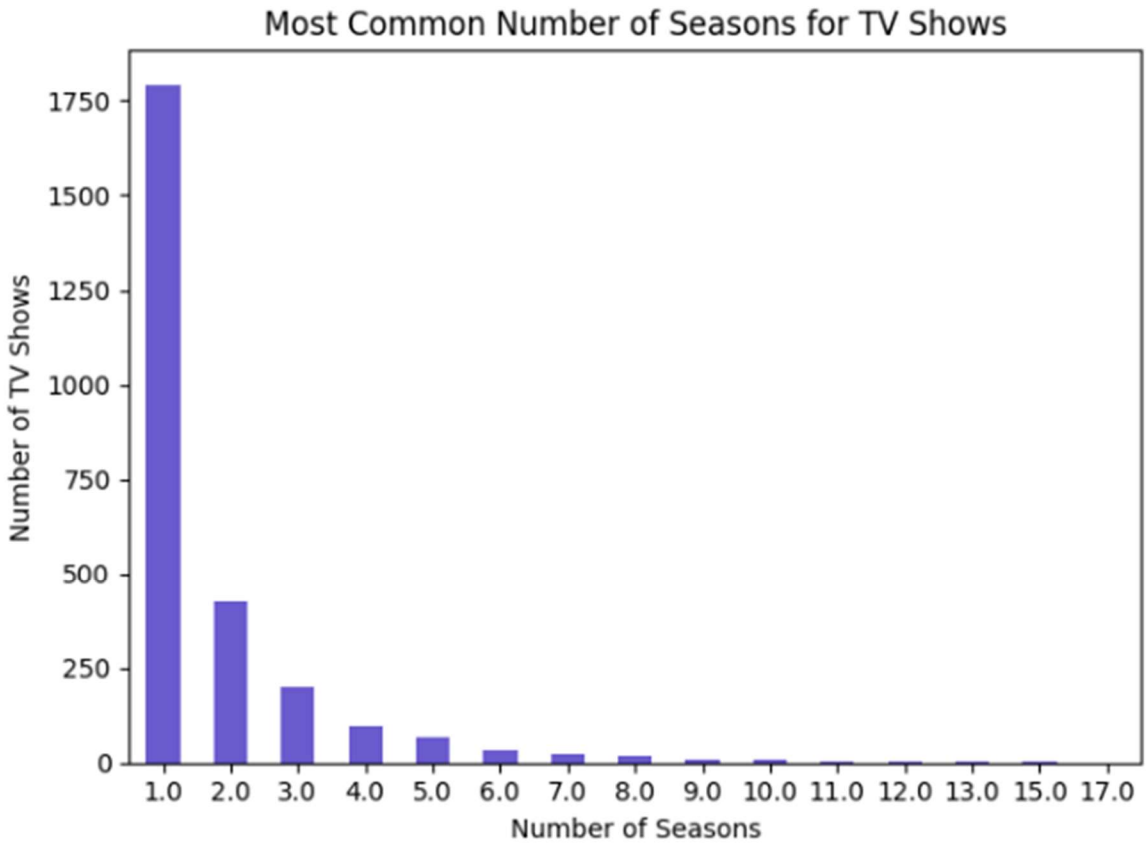
Q16. What's the most common number of seasons for TV shows?

Insight: 1–2 seasons are most common, suggesting Netflix favors short-run series.

Recommendation: Use short series as pilots to test audience interest before committing to longer formats.

```
# Filter TV Shows
```

```
tv_df = df[df['type'] == 'TV Show'].copy()

# Extract number of seasons
tv_df['seasons'] = tv_df['duration'].str.extract(r'(\d+)').astype(float)

# Count most common season counts
season_counts = tv_df['seasons'].value_counts().sort_index()

# Plot
season_counts.plot(kind='bar', color='slateblue')
plt.title('Most Common Number of Seasons for TV Shows')
plt.xlabel('Number of Seasons')
plt.ylabel('Number of TV Shows')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



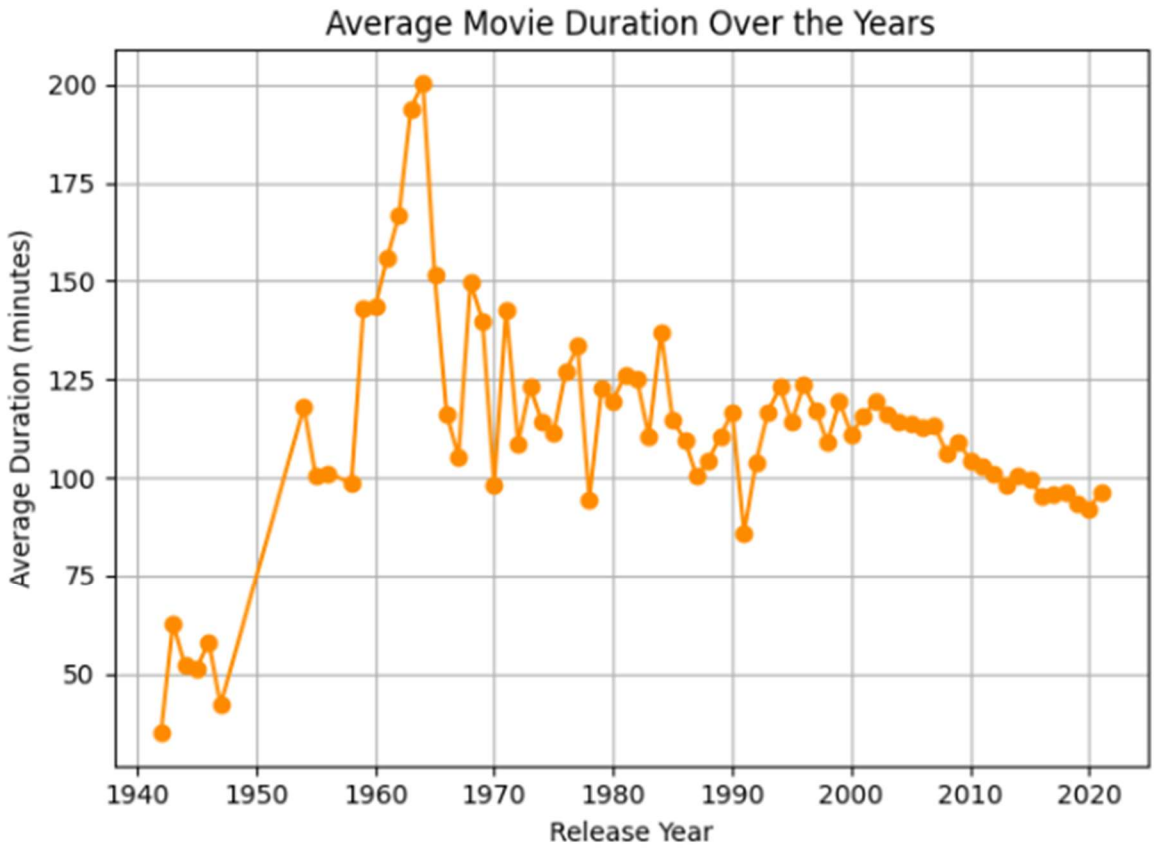
Q17. Is there a trend in movie durations over the years?

Insight: Movie durations have gradually increased over time, reflecting deeper storytelling.

Recommendation: Balance long-form and short-form content to cater to varied viewer preferences.

```
# Filter Movies

movie_df = df[df['type'] == 'Movie'].copy()


# Extract duration in minutes

movie_df['duration_mins'] = movie_df['duration'].str.extract(r'(\d+)').astype(float)
```

```
# Group by release year
duration_trend = movie_df.groupby('release_year')['duration_mins'].mean().dropna()


# Plot
duration_trend.plot(kind='line', marker='o', color='darkorange')
plt.title('Average Movie Duration Over the Years')
plt.xlabel('Release Year')
plt.ylabel('Average Duration (minutes)')
plt.grid(True)
plt.tight_layout()
plt.show()
```



5.Content Launch Strategy

Q18. In which months does Netflix add the most content?

Insight: July, August, and October see the highest content additions.
Recommendation: Align marketing campaigns and new releases with these peak months for maximum impact.

```
df['date_added'] = pd.to_datetime(df['date_added'])

df['month_added'] = df['date_added'].dt.month_name()

df['month_added'].value_counts().loc[
```

['January','February','March','April','May','June','July','August','September','October','November','December']

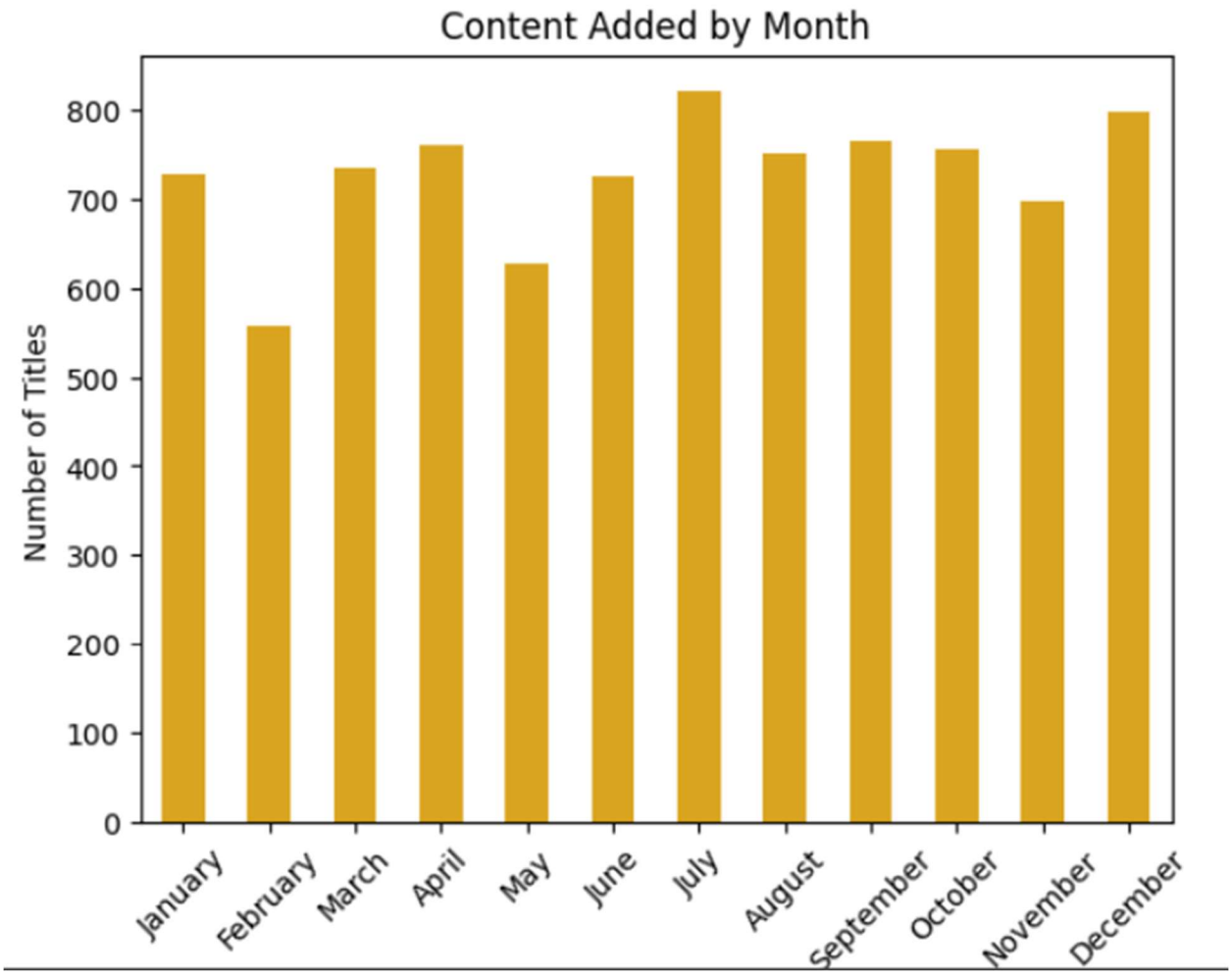
].plot(kind='bar', color='goldenrod')

plt.title('Content Added by Month')

plt.ylabel('Number of Titles')

plt.xticks(rotation=45)

plt.show()

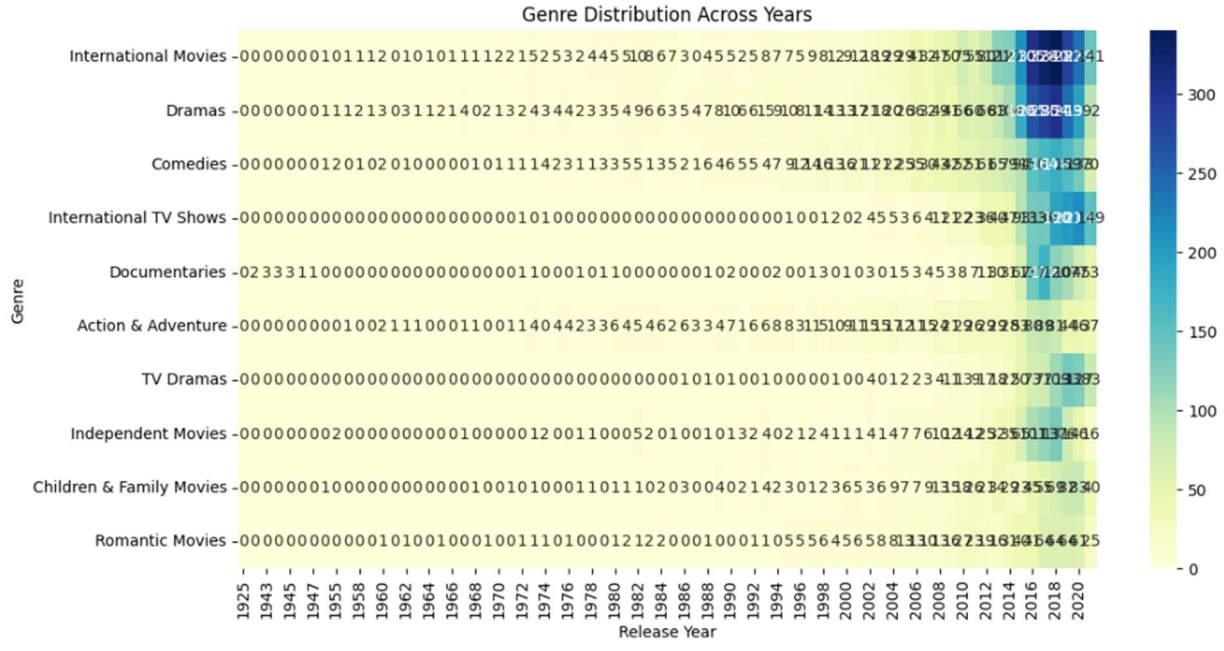


Q19. How does genre distribution vary across years?

Insight: Genres like Dramas and Documentaries remain consistently popular across years.
Recommendation: Use heatmap insights to guide annual content planning and genre diversification.

```
# Clean and prepare genre data

df['listed_in'] = df['listed_in'].fillna('Unknown')

df['genre_list'] = df['listed_in'].apply(lambda x: [i.strip() for i in x.split(',')])


# Explode genres and group by release year

genre_year_df = df.explode('genre_list')[['release_year', 'genre_list']]


# Count top genres per year

top_genres_by_year = genre_year_df.groupby(['release_year',
'genre_list']).size().reset_index(name='count')


# Pivot for heatmap

genre_pivot = top_genres_by_year.pivot(index='genre_list',
columns='release_year', values='count').fillna(0)


# Filter top genres for readability

top_genres = genre_pivot.sum(axis=1).sort_values(ascending=False).head(10)

filtered_pivot = genre_pivot.loc[top_genres.index]


# Plot heatmap
```

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
sns.heatmap(filtered_pivot, cmap='YlGnBu', annot=True, fmt='.0f')
plt.title('Genre Distribution Across Years')
plt.xlabel('Release Year')
plt.ylabel('Genre')
plt.tight_layout()
plt.show()
```



Q20. Which countries produce the most content in each genre?

Insight: U.S. leads across most genres; India excels in Romantic and Family genres.

Recommendation: Tailor genre acquisition strategies based on country strengths to optimize content sourcing.

```
# Clean country and genre data

df['country'] = df['country'].fillna('Unknown')

df['listed_in'] = df['listed_in'].fillna('Unknown')

df['genre_list'] = df['listed_in'].apply(lambda x: [i.strip() for i in x.split(',')])

df['country_list'] = df['country'].apply(lambda x: [i.strip() for i in x.split(',')])


# Explode both lists

genre_country_df = df.explode('genre_list').explode('country_list')[['genre_list',
'country_list']]


# Group and count

genre_country_counts = genre_country_df.groupby(['genre_list',
'country_list']).size().reset_index(name='count')


# Pivot for heatmap

pivot_gc = genre_country_counts.pivot(index='genre_list', columns='country_list',
values='count').fillna(0)


# Filter top genres and countries

top_genres = pivot_gc.sum(axis=1).sort_values(ascending=False).head(10)

top_countries = pivot_gc.sum().sort_values(ascending=False).head(10)

filtered_gc = pivot_gc.loc[top_genres.index, top_countries.index]
```

```
# Plot heatmap
plt.figure(figsize=(12, 6))
sns.heatmap(filtered_gc, cmap='OrRd', annot=True, fmt='.0f')
plt.title('Top Countries Producing Each Genre')
plt.xlabel('Country')
plt.ylabel('Genre')
plt.tight_layout()
plt.show()
```