Netflix Data: Cleaning, Analysis, and Visualization

◇ Project Objective

This project focuses on cleaning and analyzing Netflix's catalog data to uncover meaningful patterns and trends in the platform's content.
We'll explore content types, genres, countries of origin, ratings, and temporal trends to gain insights into Netflix's global library.

◇ Tools & Technologies

- **Python (Pandas, NumPy, Matplotlib, Seaborn, Plotly)** — Data cleaning, analysis, and visualization
- **SQL (optional)** — Data querying (can use PostgreSQL / SQLite)
- **Tableau (optional)** — Interactive dashboards
- **Google Colab** — Development environment

◇ Dataset

The dataset contains Netflix titles (movies & TV shows) added to the platform from 2008 to 2021.
It includes fields like:

- `show_id`
- `type`
- `title`
- `director`
- `cast`
- `country`
- `date_added`
- `release_year`
- `rating`
- `duration`
- `listed_in` (genres)

❓ Key Questions

- What is the distribution of content types (Movies vs TV Shows)?
- How have content additions evolved over time?
- What are the most common genres and countries represented?
- Who are the most frequent directors?
- How does the duration of content vary?
- Are there clusters of similar content?

❓ Advanced Goals

- Feature engineering (e.g., duration in minutes, genre counts)
- Clustering of content using machine learning
- Build interactive charts using Plotly

Step 2: Import Required Libraries

In this step, we will import essential Python libraries for data handling, analysis, and visualization.
We will use:

- **Pandas / NumPy** for data manipulation
- **Matplotlib / Seaborn / WordCloud / Plotly** for visualizations

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
import plotly.express as px

# Set default style
sns.set(style="whitegrid")
plt.rcParams["figure.figsize"] = (10, 6)
```

Step 2: Upload Dataset

We will now upload the Netflix dataset CSV file into our Colab environment using `google.colab.files.upload()`.

In [ ]:
```
from google.colab import files
```

```
# Open file picker
uploaded = files.upload()

# Read uploaded CSV
import io
df = pd.read_csv(io.BytesIO(next(iter(uploaded.values()))))

# Display first few rows
df.head()
```

Browse... Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving netflix1 (5).csv to netflix1 (5) (1).csv
```

Out[ ]:

	show_id	type	title	director	country	date_added	release_year	rati
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	United States	9/25/2021	2020	PG-
1	s3	TV Show	Ganglands	Julien Leclercq	France	9/24/2021	2021	TV-I
2	s6	TV Show	Midnight Mass	Mike Flanagan	United States	9/24/2021	2021	TV-I
3	s14	Movie	Confessions of an Invisible Girl	Bruno Garotti	Brazil	9/22/2021	2021	TV-
4	s8	Movie	Sankofa	Haile Gerima	United States	9/24/2021	1993	TV-I

Step 2: Inspect Dataset Shape and Information

We will check the dataset's size and the data types of each column to understand its structure.

In [ ]:

```
# Show dataset shape
print(f"Dataset Shape: {df.shape}")

# Show column info and data types
df.info()
```

```
Dataset Shape: (8790, 10)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8790 entries, 0 to 8789
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8790 non-null   object
 1   type          8790 non-null   object
 2   title         8790 non-null   object
 3   director      8790 non-null   object
 4   country       8790 non-null   object
 5   date_added    8790 non-null   object
 6   release_year  8790 non-null   int64 
 7   rating        8790 non-null   object
 8   duration      8790 non-null   object
 9   listed_in     8790 non-null   object
dtypes: int64(1), object(9)
memory usage: 686.8+ KB
```

Step 3: Data Inspection and Cleaning

In this step, we will:

- Check for missing values
- Drop duplicates
- Clean up columns
- Convert data types
- Engineer new useful features

In [ ]:
```
# Check missing values
missing_values = df.isnull().sum()
print("Missing values per column:")
print(missing_values[missing_values > 0])
```

```
Missing values per column:
Series([], dtype: int64)
```

Remove Duplicates

We will remove duplicate rows to ensure data integrity.

In [ ]:
```
# Remove duplicates
initial_shape = df.shape
df.drop_duplicates(inplace=True)
print(f"Removed {initial_shape[0] - df.shape[0]} duplicate rows.")
```

```
Removed 0 duplicate rows.
```

Handle Missing Data

We will:

- Fill missing `director` and `cast` with `Unknown`
- Fill missing `country` with `Not Given`
- Fill missing `rating` with `Not Rated`
- Convert `date_added` to datetime

In [ ]:
```python
# Fill missing values (clean syntax, no warnings)
df = df.fillna({
    'director': 'Unknown',
    'country': 'Not Given',
    'rating': 'Not Rated'
})

# Convert date_added to datetime
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')

# Final check
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8790 entries, 0 to 8789
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8790 non-null   object
 1   type          8790 non-null   object
 2   title         8790 non-null   object
 3   director      8790 non-null   object
 4   country       8790 non-null   object
 5   date_added    8790 non-null   datetime64[ns]
 6   release_year  8790 non-null   int64
 7   rating        8790 non-null   object
 8   duration      8790 non-null   object
 9   listed_in     8790 non-null   object
dtypes: datetime64[ns](1), int64(1), object(8)
memory usage: 686.8+ KB
```

Cleaning Summary

◈ Removed duplicate records
◈ Filled missing values in `director`, `country`, `rating`
◈ Converted `date_added` to datetime

Step 4: Feature Engineering and EDA Preparation

In this step:

- Extract duration value and type
- Create a content age column
- Count number of genres per title
- Prepare data for advanced visualizations

In [ ]:
```
# Extract numeric duration and type (e.g., 90 min, 1 Season)
df[['duration_int', 'duration_type']] = df['duration'].str.extract(r'(\d+)\s*(

# Convert duration_int to numeric
df['duration_int'] = pd.to_numeric(df['duration_int'], errors='coerce')

# Check results
df[['duration', 'duration_int', 'duration_type']].head()
```

Out[ ]:

	duration	duration_int	duration_type
0	90 min	90	min
1	1 Season	1	Season
2	1 Season	1	Season
3	91 min	91	min
4	125 min	125	min

Content Age

We will compute the age of the content based on the current year.

In [ ]:
```
from datetime import datetime

current_year = datetime.now().year
df['content_age'] = current_year - df['release_year']

# Preview
df[['title', 'release_year', 'content_age']].head()
```

Out[ ]:

	title	release_year	content_age
0	Dick Johnson Is Dead	2020	5
1	Ganglands	2021	4
2	Midnight Mass	2021	4
3	Confessions of an Invisible Girl	2021	4
4	Sankofa	1993	32

Genre Count

We will calculate how many genres each title belongs to by counting commas in `listed_in`.

In [ ]:

```
df['num_genres'] = df['listed_in'].apply(lambda x: len(str(x).split(',')))

# Preview
df[['title', 'listed_in', 'num_genres']].head()
```

Out[ ]:

	title	listed_in	num_genres
0	Dick Johnson Is Dead	Documentaries	1
1	Ganglands	Crime TV Shows, International TV Shows, TV Act...	3
2	Midnight Mass	TV Dramas, TV Horror, TV Mysteries	3
3	Confessions of an Invisible Girl	Children & Family Movies, Comedies	2
4	Sankofa	Dramas, Independent Movies, International Movies	3

Feature Engineering Summary

◈ Extracted duration value and type
◈ Created `content_age`
◈ Counted number of genres

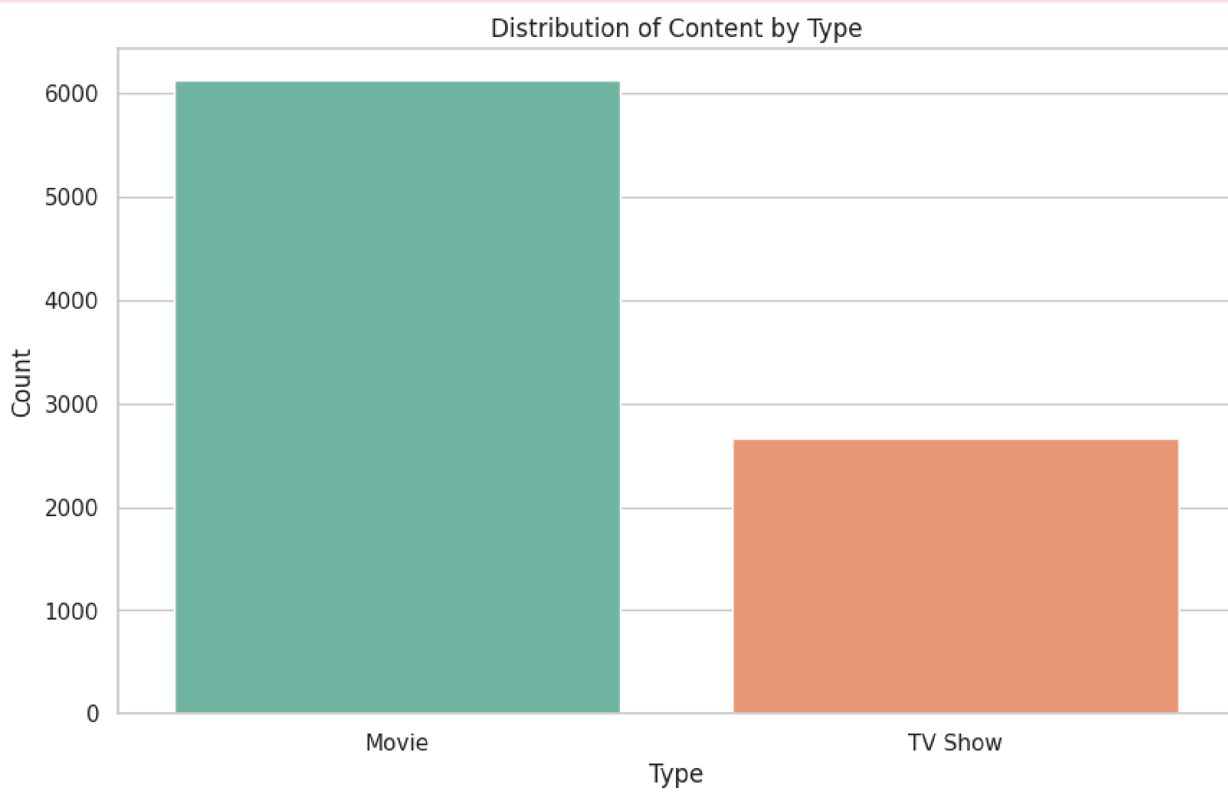
Step 5: Exploratory Data Analysis (EDA)

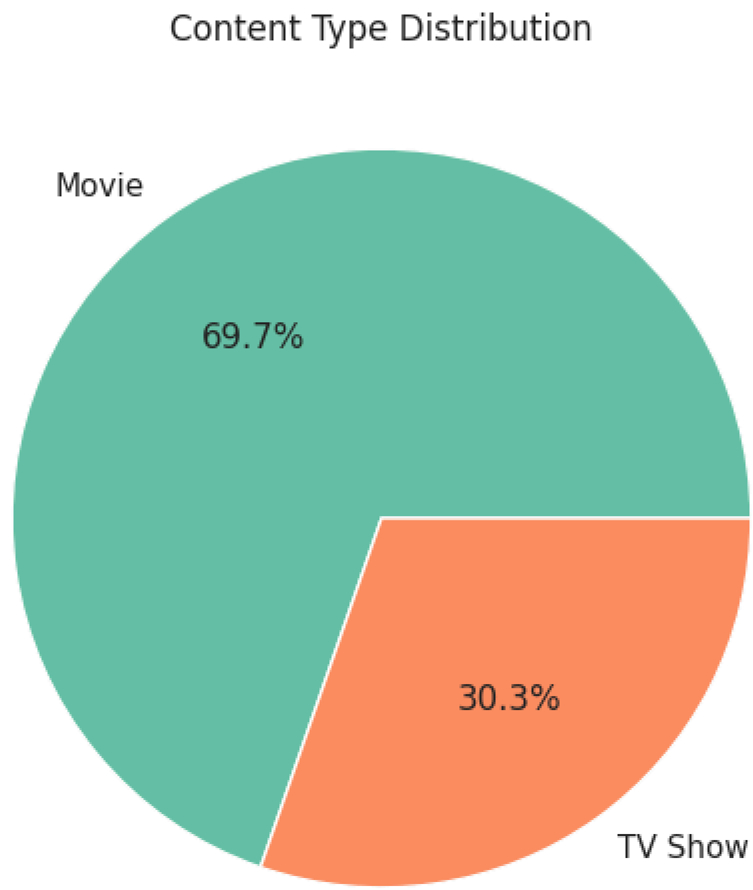
Content Type Distribution

We'll see how many Movies vs TV Shows are in the dataset.

In [ ]:
```
type_counts = df['type'].value_counts()

# Bar plot
sns.barplot(x=type_counts.index, y=type_counts.values, palette='Set2')
plt.title('Distribution of Content by Type')
plt.xlabel('Type')
plt.ylabel('Count')
plt.show()

# Pie chart
plt.pie(type_counts.values, labels=type_counts.index, autopct='%.1f%%', colors
plt.title('Content Type Distribution')
plt.show()
```

```
/tmp/ipython-input-39-2778042718.py:4: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same e
ffect.
```



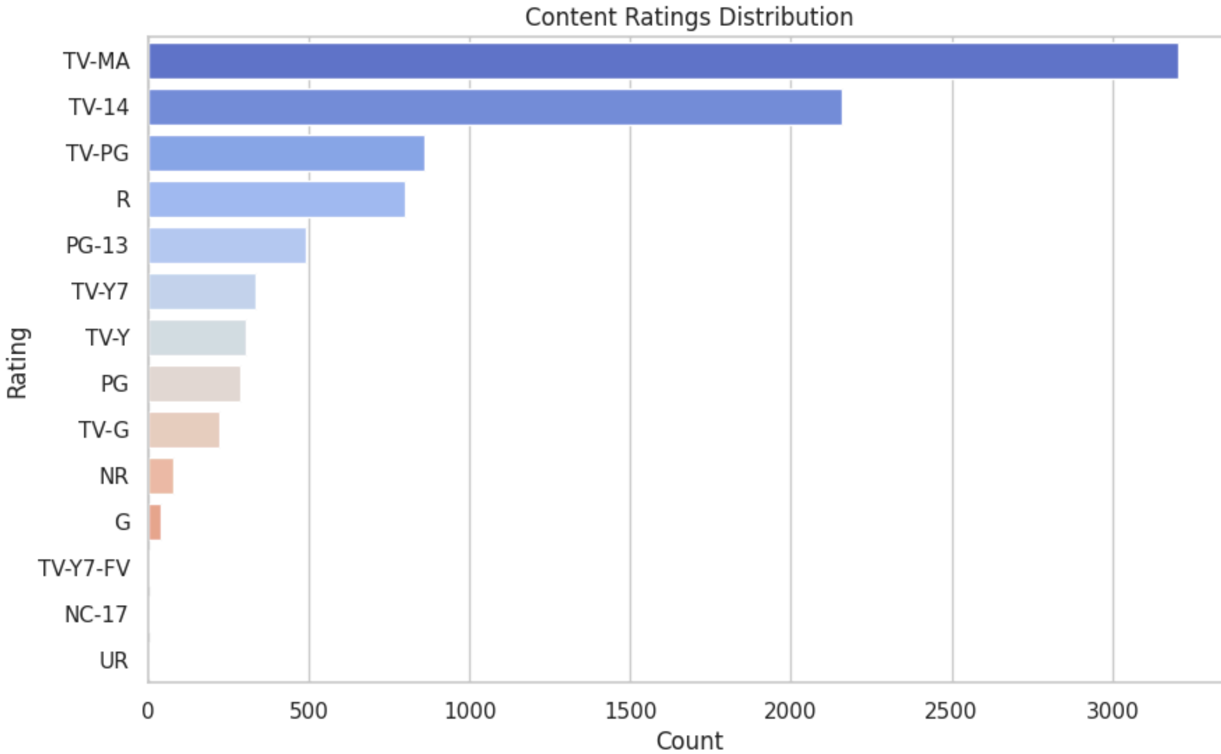


Ratings Breakdown

Let’s look at the frequency of different content ratings.

In [ ]:
```
ratings = df['rating'].value_counts()

sns.barplot(y=ratings.index, x=ratings.values, palette='coolwarm')
plt.title('Content Ratings Distribution')
plt.xlabel('Count')
plt.ylabel('Rating')
plt.show()
```

```
/tmp/ipython-input-40-3010740316.py:3: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.
```

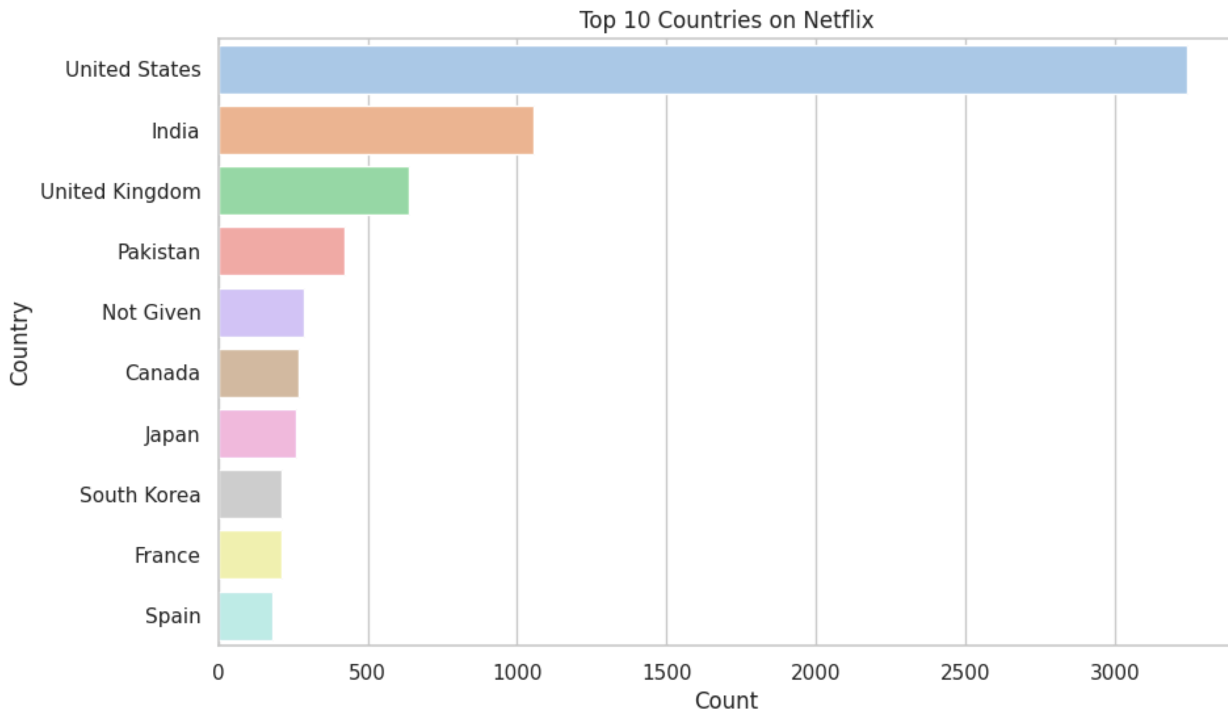


Top Countries

Which countries contribute the most content?

In [ ]:
```
top_countries = df['country'].value_counts().head(10)

sns.barplot(y=top_countries.index, x=top_countries.values, palette='pastel')
plt.title('Top 10 Countries on Netflix')
plt.xlabel('Count')
plt.ylabel('Country')
plt.show()
```

```
/tmp/ipython-input-41-654798387.py:3: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.
```

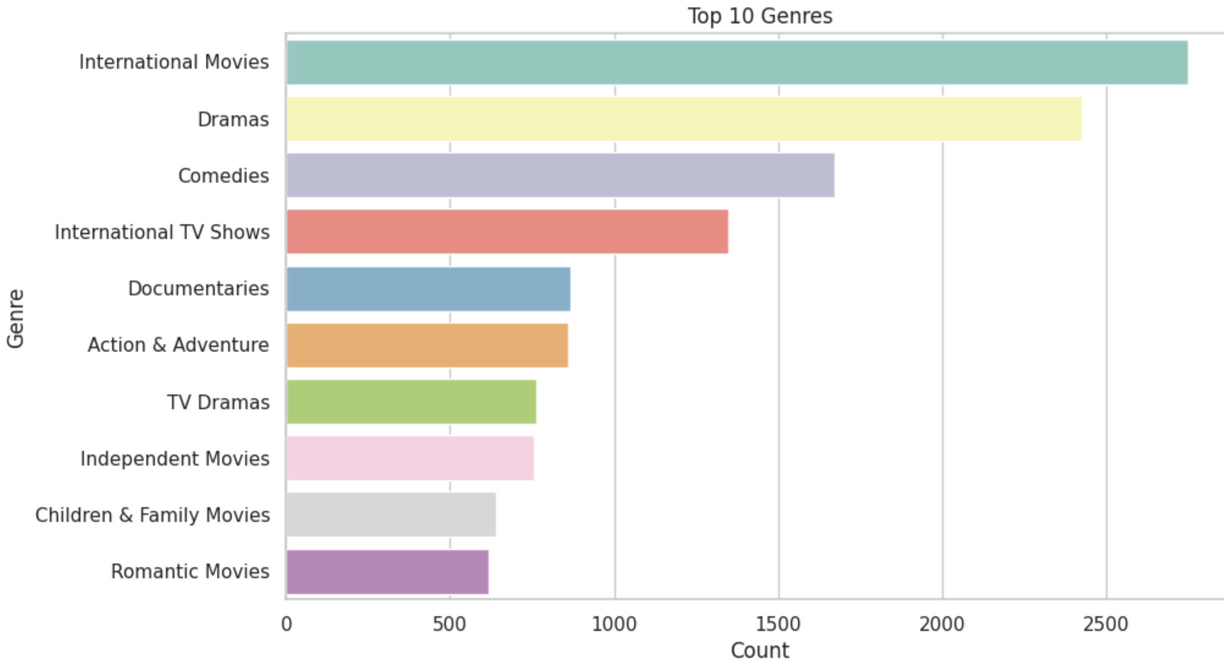


Top Genres

What are the most common genres on Netflix?

In [ ]:
```
# Split and flatten genre list
all_genres = df['listed_in'].str.split(',').explode().str.strip()
top_genres = all_genres.value_counts().head(10)

sns.barplot(y=top_genres.index, x=top_genres.values, palette='Set3')
plt.title('Top 10 Genres')
plt.xlabel('Count')
plt.ylabel('Genre')
plt.show()
```

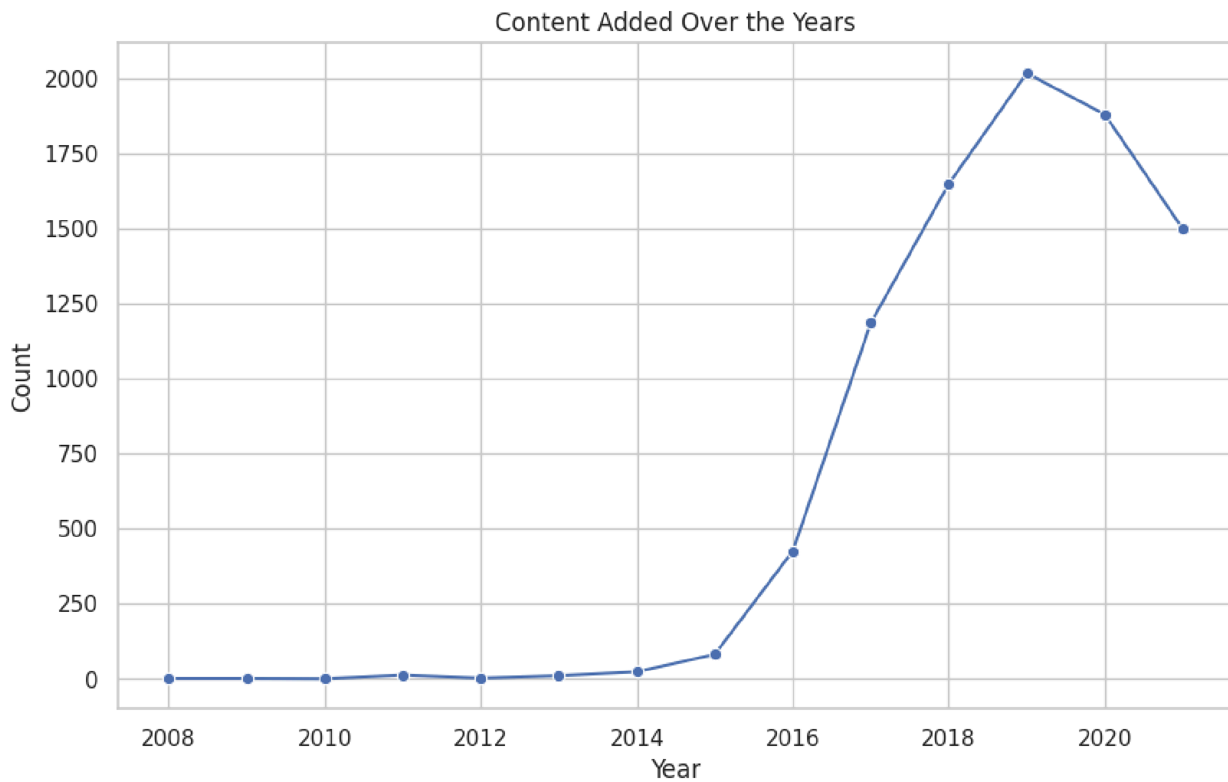
```
/tmp/ipython-input-42-1221969359.py:5: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.
```



Content Added Over Time

We'll explore how content additions have changed over years.

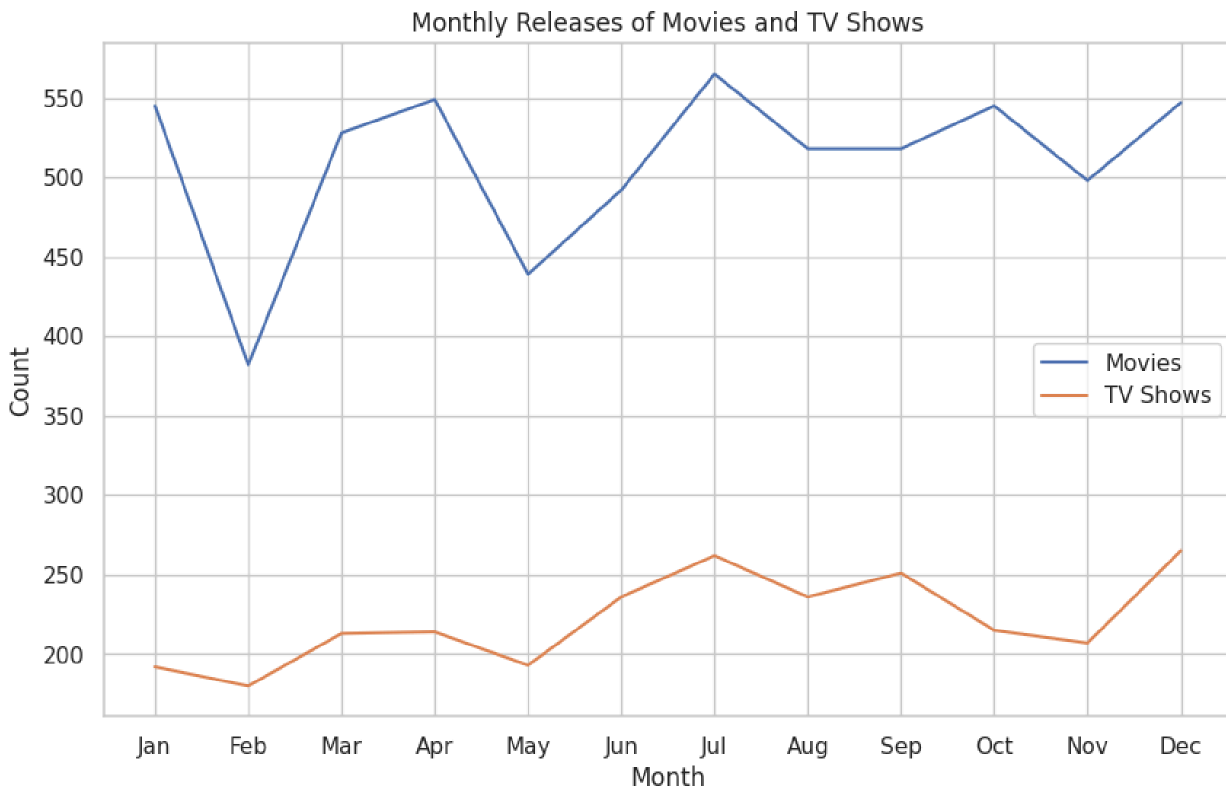
In [ ]:
```
# Extract year added
df['year_added'] = df['date_added'].dt.year

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

sns.lineplot(x=yearly_counts.index, y=yearly_counts.values, marker='o')
plt.title('Content Added Over the Years')
plt.xlabel('Year')
plt.ylabel('Count')
plt.show()
```



Monthly Content Releases

Compare release trends for Movies vs TV Shows by month.

In [ ]:
```
df['month_added'] = df['date_added'].dt.month

monthly_movie = df[df['type'] == 'Movie']['month_added'].value_counts().sort_i
monthly_tv = df[df['type'] == 'TV Show']['month_added'].value_counts().sort_in

plt.plot(monthly_movie.index, monthly_movie.values, label='Movies')
plt.plot(monthly_tv.index, monthly_tv.values, label='TV Shows')
plt.xlabel('Month')
plt.ylabel('Count')
plt.title('Monthly Releases of Movies and TV Shows')
plt.xticks(range(1, 13), ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep
plt.legend()
plt.grid(True)
plt.show()
```

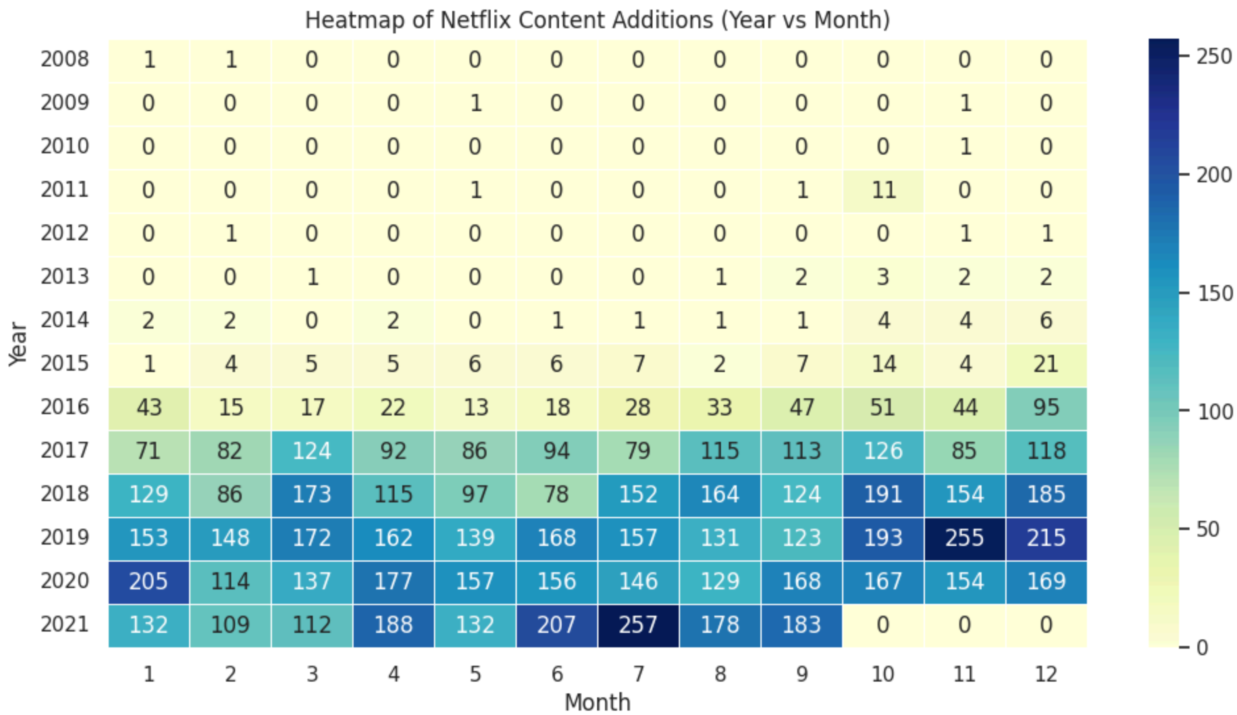


Step 6: Advanced Visualizations

Heatmap of Content Additions

Shows how many titles were added per month-year combo.

In [ ]:
```
# Group by year and month
heatmap_data = df.groupby(['year_added', 'month_added']).size().unstack(fill_v

plt.figure(figsize=(12, 6))
sns.heatmap(heatmap_data, cmap='YlGnBu', linewidths=0.5, annot=True, fmt='d')
plt.title('Heatmap of Netflix Content Additions (Year vs Month)')
plt.xlabel('Month')
plt.ylabel('Year')
plt.show()
```



Interactive Genre Distribution

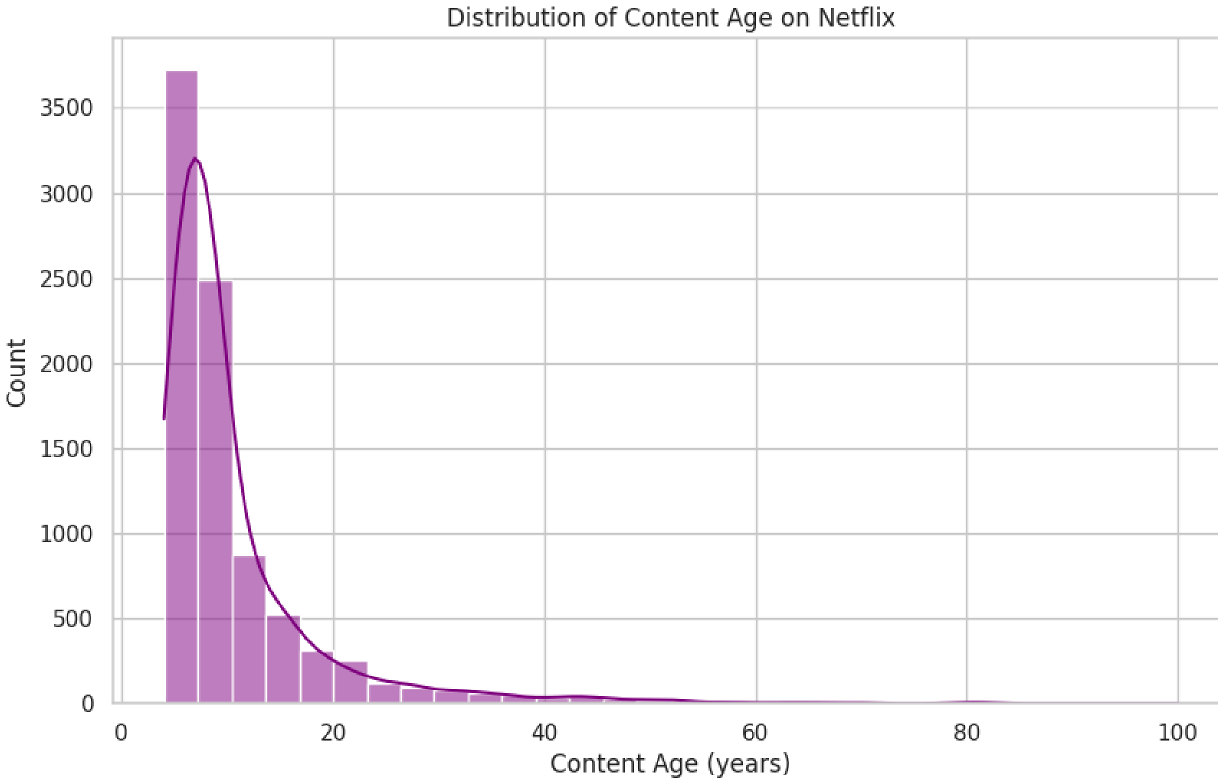
Use Plotly for an interactive bar plot of top genres.

In [ ]:
```
fig = px.bar(top_genres.reset_index(), x='listed_in', y='count',
             labels={'listed_in': 'Genre', 'count': 'Count'}, # Update labels
             title='Top 10 Genres (Interactive)')
fig.show()
```

Content Age Distribution

See how old Netflix's catalog content is.

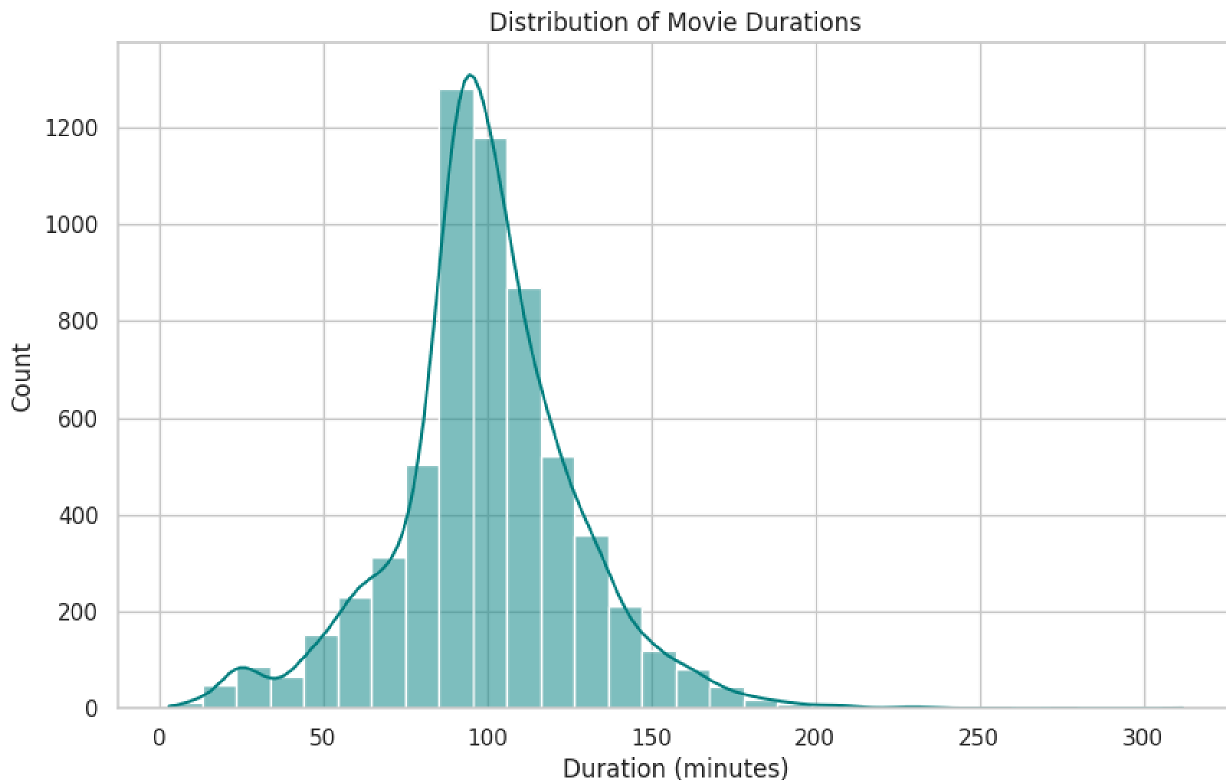
```
In [ ]: sns.histplot(df['content_age'], bins=30, kde=True, color='purple')
        plt.title('Distribution of Content Age on Netflix')
        plt.xlabel('Content Age (years)')
        plt.ylabel('Count')
        plt.show()
```



Movie Duration Distribution

We’ll examine the spread of movie durations.

In [ ]:
```
sns.histplot(df[df['type'] == 'Movie']['duration_int'].dropna(), bins=30, kde=
plt.title('Distribution of Movie Durations')
plt.xlabel('Duration (minutes)')
plt.ylabel('Count')
plt.show()
```



Clustering Prep

We'll prepare features for clustering: duration, age, genre count.

In [ ]:
```
# Select features
clustering_df = df[['duration_int', 'content_age', 'num_genres']].dropna()

# Show summary
clustering_df.describe()
```

Out[ ]:

	duration_int	content_age	num_genres
count	8790.000000	8790.000000	8790.000000
mean	69.934471	10.816837	2.194994
std	50.794433	8.825466	0.784114
min	1.000000	4.000000	1.000000
25%	2.000000	6.000000	2.000000
50%	88.500000	8.000000	2.000000
75%	106.000000	12.000000	3.000000
max	312.000000	100.000000	3.000000

Step 7: Clustering Netflix Content

We will apply **KMeans clustering** to group Netflix titles based on:

- Duration (minutes or seasons)
- Content age
- Number of genres

This helps identify natural groupings (e.g., short modern movies, long classic TV shows).

In [ ]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Standardize features
scaler = StandardScaler()
clustering_scaled = scaler.fit_transform(clustering_df)

# Check scaled data
pd.DataFrame(clustering_scaled, columns=clustering_df.columns).head()
```

Out[ ]:

	duration_int	content_age	num_genres
0	0.395056	-0.659134	-1.524093
1	-1.357204	-0.772449	1.026702
2	-1.357204	-0.772449	1.026702
3	0.414745	-0.772449	-0.248695
4	1.084148	2.400368	1.026702

Apply KMeans Clustering

We'll start with 3 clusters (you can tune this later).

In [ ]:
```
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
clusters = kmeans.fit_predict(clustering_scaled)

# Add to DataFrame
clustering_df['cluster'] = clusters

clustering_df.head()
```

Out[ ]:

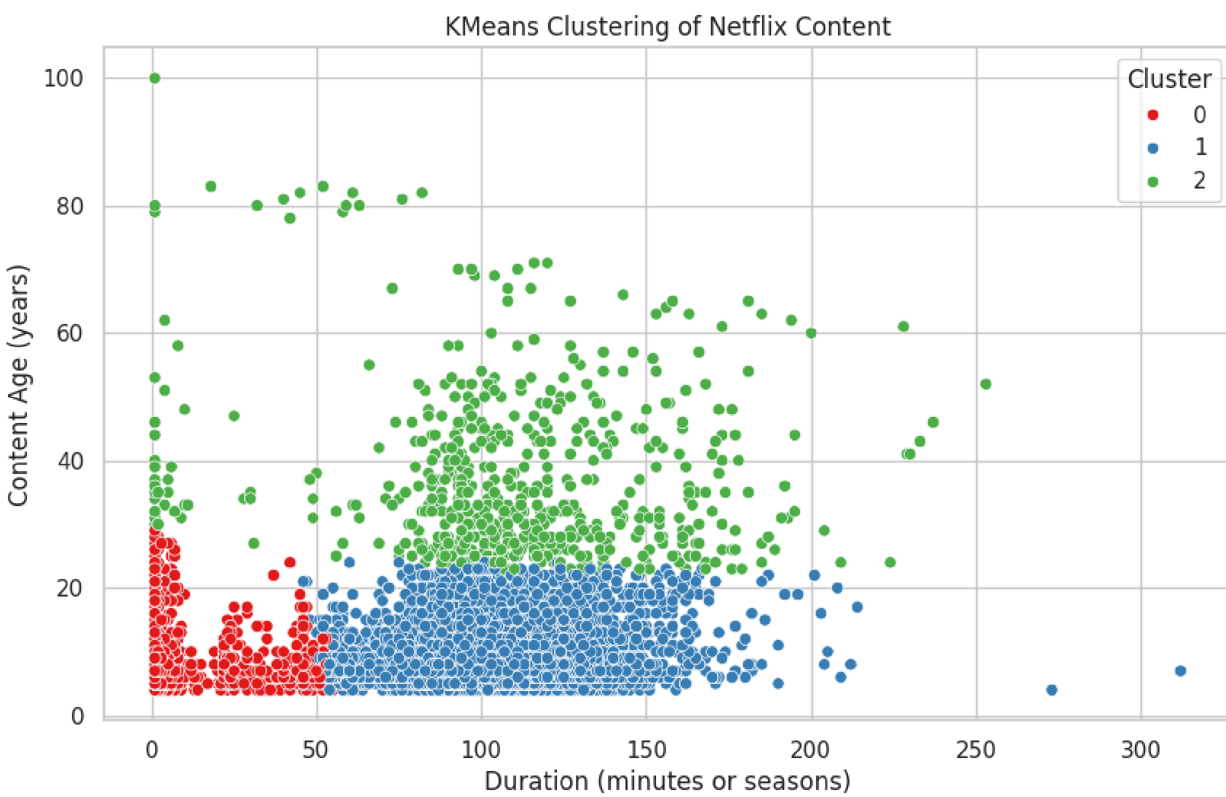
	duration_int	content_age	num_genres	cluster
0	90	5	1	1
1	1	4	3	0
2	1	4	3	0
3	91	4	2	1
4	125	32	3	2

Visualize Clusters

We'll plot content age vs duration, colored by cluster.

In [ ]:

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=clustering_df, x='duration_int', y='content_age', hue='cl
plt.title('KMeans Clustering of Netflix Content')
plt.xlabel('Duration (minutes or seasons)')
plt.ylabel('Content Age (years)')
plt.legend(title='Cluster')
plt.show()
```



Cluster Summary

We'll review the average values of features per cluster.

In [ ]:
```python
clustering_df.groupby('cluster').mean()
```

Out[ ]:

cluster	duration_int	content_age	num_genres
0	4.994184	7.939788	2.220322
1	101.446547	9.477956	2.168969
2	109.932990	37.424399	2.304124

Step 8: Conclusion and Key Insights

◈ We cleaned and enriched the Netflix dataset

- Filled missing values in `director`, `country`, `rating`
- Converted `date_added` to proper datetime
- Engineered features like `content_age`, `duration_int`, `num_genres`

◈ We explored Netflix's catalog

- Movies dominate over TV Shows
- `TV-MA` and `TV-14` are the most common ratings
- The US, India, and UK contribute the most content
- Drama and comedy are the most frequent genres
- Content additions peaked between 2018-2020

◈ We applied clustering

- Grouped content into clusters based on duration, age, and genre diversity
- Visualized clusters to see patterns (e.g., modern short movies, older long TV shows)
