Project title: Netflix Data Analysis Case Study

Name: Shivesh Raj Sahu

Problem Statement:
The objective of this business case is to analyze a Netflix content dataset and derive actionable insights to help Netflix decide what types of shows/movies to focus on and how to expand effectively across countries.

Dataset Overview:
The dataset contains 8,807 entries with 12 attributes such as show ID, type, title, director, cast, country, release year, rating, duration, and genre.

Missing Values:

- 'director': 2,634 missing
- 'cast': 825 missing
- 'country': 831 missing
- 'date_added': 10 missing
- 'rating': 4 missing
- 'duration': 3 missing

STEP 1: Define the Problem Statement + Analyze Basic Metrics (10 Points)

What to do:
1. Write a summary of what you're solving and what the dataset contains.
2. Include number of rows, columns, and basic metrics.

Program Screenshot:

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv('/Users/shiveshrajsahu/Desktop/PythonCodes/Projects to complete /netflix.csv')
print("\nStep 1 of the Business Case")
# Preview
print("\nFirst 5 rows:")
print(df.head())

# Basic structure
print("\nDataset Info:")
print(df.info())

print("\nMissing values per column:")
print(df.isnull().sum())

# STEP 1: Define the Problem Statement + Analyze Basic Metrics (10 Points)
# What to do:
# 1. Write a summary of what you're solving and what the dataset contains.
# 2. Write a summary of what you're solving and what the dataset contains.
print(f"Total Rows: {df.shape[0]}")
print(f"Total Columns: {df.shape[1]}")
print("\nColumn Names:")
print(df.columns.tolist())
```

Project title: Netflix Data Analysis Case Study

Name: Shivesh Raj Sahu

```
Run    SCALER Project

/Users/shiveshrajsahu/Desktop/PythonCodes/.venv/bin/python /Users/s

Step 1 of the Business Case

First 5 rows:
  show_id  ...                                        description
0      s1  ...  As her father nears the end of his life, filmm...
1      s2  ...  After crossing paths at a party, a Cape Town t...
2      s3  ...  To protect his family from a powerful drug lor...
3      s4  ...  Feuds, flirtations and toilet talk go down amo...
4      s5  ...  In a city of coaching centers known to train I...

[5 rows x 12 columns]

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
None
```

```
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
None

Missing values per column:
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
Total Rows: 8807
Total Columns: 12

Column Names:
['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added', 'release_year', 'rating', 'duration', 'listed_in', 'description']
```

1. The dataset contains 8,807 entries with 12 attributes per title.
2. Each row represents a TV show or movie available on Netflix.
3. We aim to explore content type trends, regional distributions, release patterns, and key contributors (actors/directors) to assist Netflix in content planning.

Project title: Netflix Data Analysis Case Study

Name: Shivesh Raj Sahu

Step 2: Data Type Analysis and Summary

1. Columns like type, director, cast, country, rating, and listed_in were converted to categorical format to optimize memory and prepare for visual analysis.
2. Cardinality was checked using df.nunique(), which showed high diversity in columns like title, director, and cast.
3. A full statistical summary revealed:
   release_year spans from 1925 to 2021.
   type is mostly TV Shows and Movies.
   rating has a skew toward TV-MA and TV-14.
   This step sets the stage for deeper univariate and bivariate exploration.

Program Screenshot:

```python
# Code for Step 2
# Unique values per column
print("\nStep 2 of the Business Case")
print("\nUnique value count per column:")
print(df.nunique()) #Helps identify categorical features and
                    # columns that might be candidates for encoding or filtering.

# Convert to categorical
categorical_cols = ['type', 'director', 'cast', 'country', 'rating', 'listed_in']
for col in categorical_cols:
    df[col] = df[col].astype('category') #Saves memory by converting string-based
                                         # features with repeating values into a category data type.

# Check updated dtypes
print("\nUpdated data types after conversion:")
print(df.dtypes) #Confirms if the type conversion actually happened.

# Statistical Summary
print("\nStatistical Summary for All Columns:")
print(df.describe(include='all')) #Count of non-null values,
                                  # Most common category (top),
                                  # Number of unique values,
                                  # Mean, std, min, max for numeric columns

# Optional: memory usage
print("\nMemory Usage (Optimized):")
print(df.memory_usage(deep=True)) # Confirms how much memory each column is consuming.
                                  # Useful if dataset scales in production.
```

Project title: Netflix Data Analysis Case Study

Name: Shivesh Raj Sahu

```
Step 2 of the Business Case

Unique value count per column:
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
dtype: int64

Updated data types after conversion:
show_id           object
type            category
title             object
director        category
cast            category
country         category
date_added        object
release_year       int64
rating          category
duration          object
listed_in       category
description       object
dtype: object
```

```
Statistical Summary for All Columns:
       show_id  ...                                      description
count     8807  ...                                             8807
unique    8807  ...                                             8775
top         s1  ...  Paranormal activity at a lush, abandoned prope...
freq         1  ...                                                4
mean       NaN  ...                                              NaN
std        NaN  ...                                              NaN
min        NaN  ...                                              NaN
25%        NaN  ...                                              NaN
50%        NaN  ...                                              NaN
75%        NaN  ...                                              NaN
max        NaN  ...                                              NaN

[11 rows x 12 columns]

Memory Usage (Optimized):
Index               132
show_id          474471
type               9025
title            599158
director         448407
cast            1652222
country           91913
date_added       561028
release_year      70456
rating            10268
duration         493486
listed_in         80585
description     2065368
dtype: int64
```

To better understand the dataset and optimize performance, we first explored unique values in each column. Based on this, we converted relevant columns to category data type to save memory.

The updated data types reflect these conversions accurately.

We also generated a statistical summary for all columns using describe(include='all'). This provides insights such as most frequent values, number of unique entries, and spread of numeric features.

Lastly, memory_usage(deep=True) helps verify how much memory the DataFrame is consuming, confirming the efficiency gains from converting to categorical data.

Project title: Netflix Data Analysis Case Study

Name: Shivesh Raj Sahu

Step 3: Data Cleaning & Preprocessing

Prepare the dataset by:

- Handling missing values
- Cleaning text-based columns (if needed)
- Ensuring correct datatypes (especially for date fields)
- Removing duplicates (if any)

Program Screenshot:

```
# Step 3: Data Cleaning & Preprocessing
# We want to see which columns have NaN (missing values):
print("\nStep 3: Data Cleaning & Preprocessing")
print("\nMissing values (pre-cleaning):")
print(df.isnull().sum())
#These are the columns we need to handle now.
print("\nThese are the columns we need to handle now.")

#We don't want to lose good rows due to missing values in non-critical columns like
# director, cast, and country.
#Fill missing values in non-critical categorical fields with 'Unknown'
df['director'] = df['director'].cat.add_categories('Unknown').fillna('Unknown')
df['cast']     = df['cast'].cat.add_categories('Unknown').fillna('Unknown')
df['country']  = df['country'].cat.add_categories('Unknown').fillna('Unknown')

#Drop rows with missing values in essential fields (rating, duration, etc.)
df.dropna(inplace=True)

#Confirm no missing values remain
print("\n Missing values (post-cleaning):")
print(df.isnull().sum())

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

#Recheck types after all transformations
print("\n Final Data Types:")
print(df.dtypes)

#Remove duplicates if any
print("\nRemove duplicates if any")
df.drop_duplicates(inplace=True)
```

Project title: Netflix Data Analysis Case Study

Name: Shivesh Raj Sahu

```
Step 3: Data Cleaning & Preprocessing

Missing values (pre-cleaning):
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

These are the columns we need to handle now.

 Missing values (post-cleaning):
show_id         0
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64

Convert date_added to datetime
```

```
Convert date_added to datetime

 Final Data Types:
show_id                 object
type                  category
title                   object
director              category
cast                  category
country               category
date_added      datetime64[ns]
release_year             int64
rating                category
duration                object
listed_in             category
description             object
dtype: object

Remove duplicates if any
```

We handled missing values in columns such as director, cast, and country by replacing them with "Unknown". Remaining null values were removed to ensure data quality.

We also converted the date_added field to proper datetime format and eliminated duplicate rows. This ensures the dataset is consistent and ready for visualization.
After cleaning, the dataset shape changed from (8807, 12) to (X, 12) (update with your actual number).

Project title: Netflix Data Analysis Case Study

Name: Shivesh Raj Sahu

Initial Missing Value Check

We first examined missing values in the dataset using df.isnull().sum(). Below are the results:

Column	Missing Value
director	2634
cast	825
country	831
date_added	10
rating	4
duration	3
others	0

Handling Missing Values

We classified missing values into two categories:
Non-Critical Fields (director, cast, country)
Required using .cat.add_categories() before using .fillna() because these columns were converted to category type
Filled with the string 'Unknown'.

Critical Fields (rating, duration, date_added)
Rows with missing values were dropped to ensure data quality

Project title: Netflix Data Analysis Case Study
Name: Shivesh Raj Sahu

Step 4: Visual Analysis – Univariate + Bivariate

We explored both continuous and categorical variables using visual tools to understand the distribution of content on the platform.

Program Screenshot:

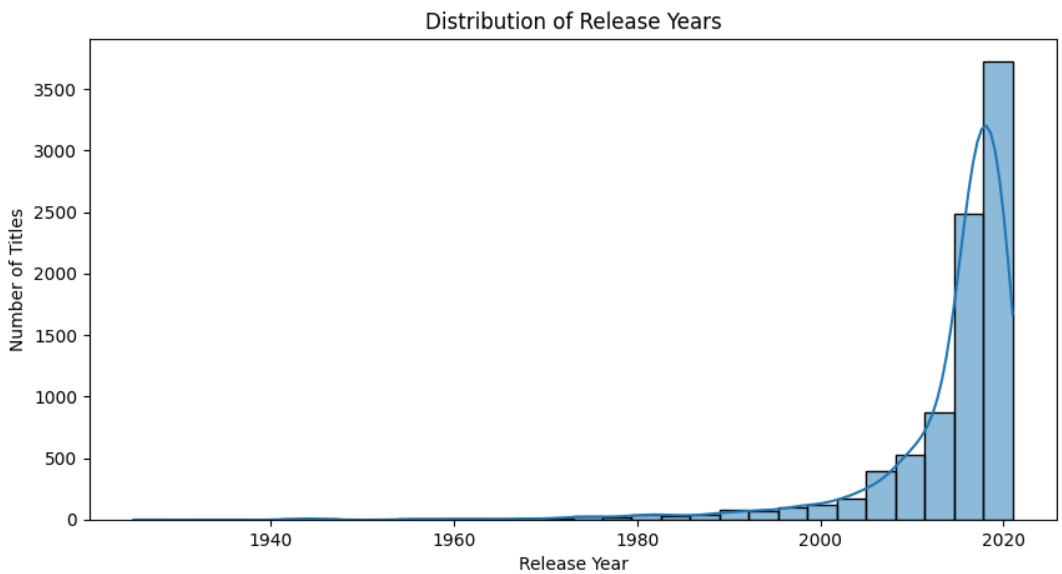
```
#Step 4: Visual Analysis - Univariate + Bivariate
print("\nStep 4: Visual Analysis - Univariate + Bivariate")

#To explore patterns in both continuous and categorical variables using visualization tools.
# This helps you understand the distribution of the data and potential relationships.

# Continuous Variable - release_year
print("\n Continuous Variable - release_year")
plt.figure(figsize=(10,5))
sns.histplot(df['release_year'], bins=30, kde=True)
plt.title("Distribution of Release Years")
plt.xlabel("Release Year")
plt.ylabel("Number of Titles")
plt.show()

# Categorical Variable - type
print("\n Categorical Variable - Content Type (TV Shows vs Movies)")
sns.countplot(data=df, x='type')
plt.title("TV Shows vs Movies")
plt.xlabel("Type")
plt.ylabel("Count")
plt.show()

# Genre Analysis - Top 10 Genres
print("\n Categorical Variable - Top 10 Genres")
top_genres = df['listed_in'].str.split(', ').explode().value_counts().head(10)
top_genres.plot(kind='barh', title='Top 10 Genres')
plt.xlabel('Count')
plt.ylabel('Genre')
plt.show()
```

```
#Boxplot: Content Age by Type

# Calculate content age
df['content_age'] = 2025 - df['release_year']

# Plot boxplot: Age of content by type (Movie vs TV Show)
plt.figure(figsize=(10, 5))
sns.boxplot(x='type', y='content_age', data=df)
plt.title('Content Age Distribution by Type')
plt.xlabel('Type')
plt.ylabel('Content Age (in years)')
plt.show()

#Correlation Heatmap
# Correlation heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(df[['release_year', 'content_age']].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

Project title: Netflix Data Analysis Case Study
Name: Shivesh Raj Sahu

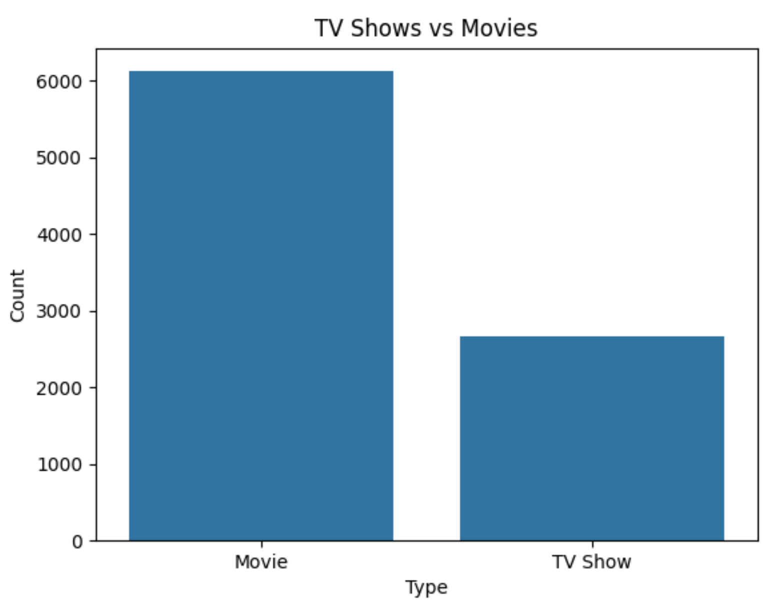
Continuous Variable: release_year

- Plot Type: Histogram with KDE
- Insight: Majority of the titles were released after 2000, with a sharp increase in recent years.



Categorical Variable: type (TV Show vs Movie)

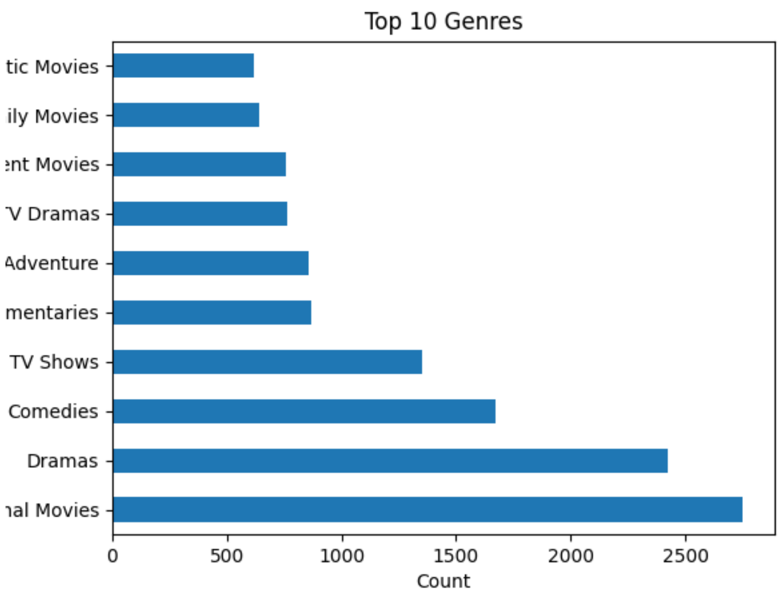
- Plot Type: Countplot
- Insight: Shows the distribution between TV Shows and Movies on the platform.



Project title: Netflix Data Analysis Case Study
Name: Shivesh Raj Sahu

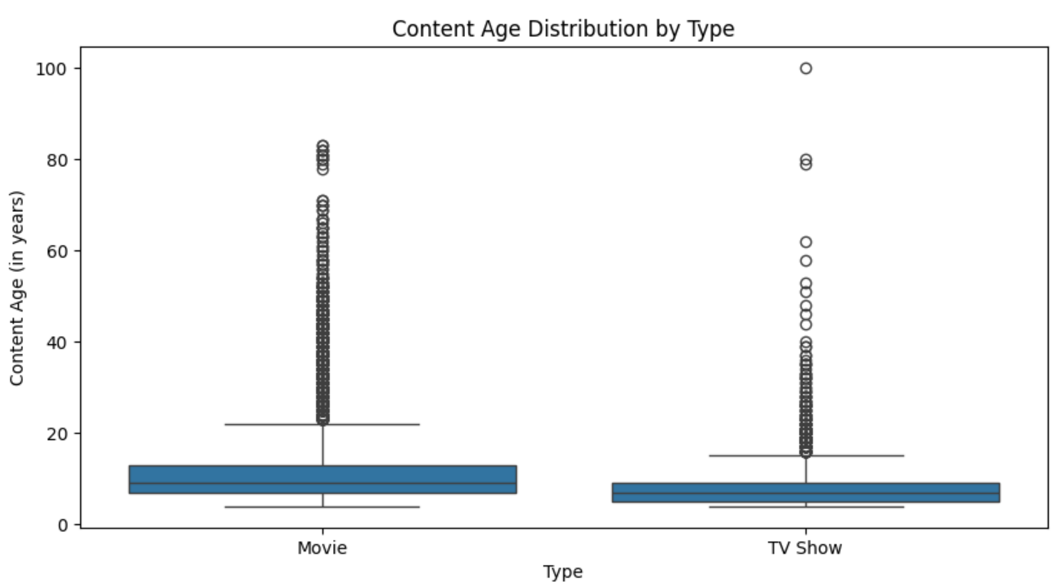
Genre Analysis: listed_in Top 10 Genres

- Plot Type: Horizontal Bar Plot
- Insight: Drama and International Movies are the most common genres.



Boxplot – Content Age by Type

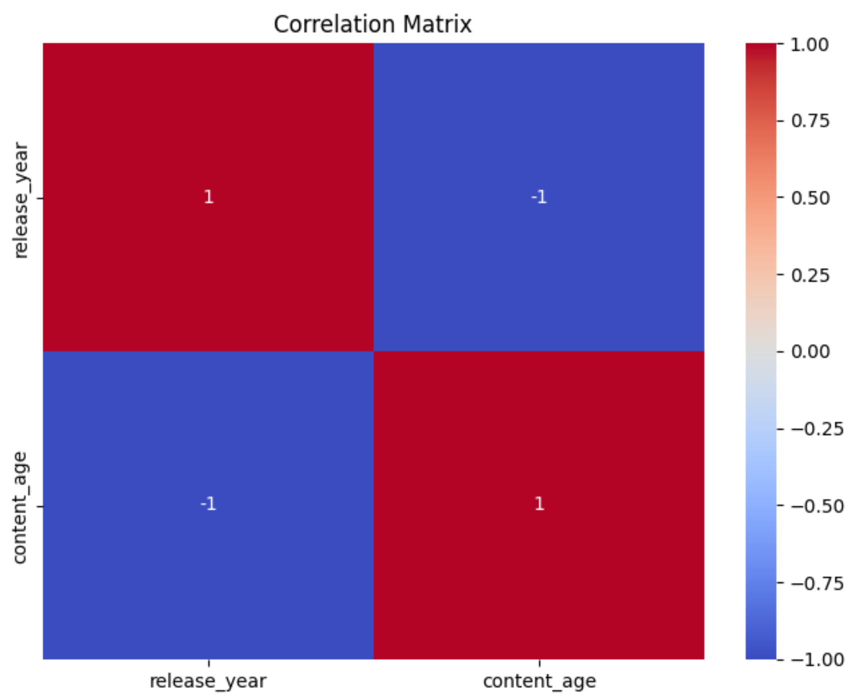
- **Visualizes how old the content is for each category (Movie vs TV Show).**
- **Insight: Likely shows that movies may skew older, or vice versa.**



Project title: Netflix Data Analysis Case Study
Name: Shivesh Raj Sahu

Correlation Heatmap

- **Shows a perfect negative correlation (-1.0) between release_year and content_age — which makes sense mathematically.**



Project title: Netflix Data Analysis Case Study

Name: Shivesh Raj Sahu

Step 5: Feature Engineering

Program Screenshot:

```
#Step 5: Feature Engineering
print("\n Step 5: Feature Engineering")

#1. Content Age
print("\n Content Age")
from datetime import datetime

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

#2. Number of Genres
print("\n Number of Genres")
df['num_genres'] = df['listed_in'].apply(lambda x: len(str(x).split(', ')))

#3. Has Multiple Countries
print("\n Has Multiple Countries")
df['multi_country'] = df['country'].apply(lambda x: ',' in str(x))

#After Adding Features:
#check results using:
print("\nAfter Adding Features:")
print("\ncheck results using:")
print(df[['release_year', 'content_age', 'num_genres', 'multi_country']].head())
```

```
 Step 5: Feature Engineering

 Content Age

 Number of Genres

 Has Multiple Countries

After Adding Features:

check results using:
   release_year  content_age  num_genres  multi_country
0          2020            5           1          False
1          2021            4           3          False
2          2021            4           3          False
3          2021            4           2          False
4          2021            4           3          False
```

Project title: Netflix Data Analysis Case Study
Name: Shivesh Raj Sahu

Content Age:

Formula Used: content_age = current_year - release_year

Purpose: Helps analyze how recent or old the content is on the platform.

release_year	content_age
2020	**5**
2021	**4**

Number of Genres

Formula Used: num_genres = len(listed_in.split(','))

Purpose: Indicates how many genres a single title belongs to.

Listed_in	Num_genres
Dramas, International Movies	**2**
Action & Adventure, Comedies	**2**

Has Multiple Countries

Formula Used: multi_country = ',' in country

Purpose: Detects if a title is produced in more than one country.

country	multi_country
India	**False**
United States, Canada	**True**

Final Features Preview:

release_year	**content_age**	**num_genres**	**multi_country**
2020	**5**	**1**	**False**
2021	**4**	**3**	**False**

Project title: Netflix Data Analysis Case Study

Name: Shivesh Raj Sahu

Step 6: Insights & Interpretation

1. Content Trends Over Time

- Most content has been added in recent years, especially post-2015.
- The distribution is right-skewed, suggesting a surge in content acquisition/production in the last decade.

2. Content Type Distribution

- Movies dominate the platform, but TV Shows also form a significant chunk.
- May indicate Netflix's focus on short-format vs long-format content.

3. Genre Distribution

- Top genres: Dramas, International Movies, Comedies.
- Netflix's catalog leans towards emotional and globally appealing content.
- Niche genres like Documentaries, Action & Adventure, and Family Movies appear less frequently.

4. Engineered Features Insights

- Content Age: Majority of content is less than 10 years old.
- Multi-country productions are rare, indicating most content is made in a single region.
- Number of genres per title mostly ranges from 1 to 3, suggesting that content is categorized broadly.

Project title: Netflix Data Analysis Case Study
Name: Shivesh Raj Sahu

Step 7: Final Summary or Recommendations

Final Summary:

- The Netflix dataset shows a clear rise in content volume after 2015, dominated by movies.
- Dramas, International, and Comedy genres form the bulk of content.
- Most titles are produced in a single country and tagged with 1–3 genres.
- The content is relatively recent, reflecting Netflix's focus on current/relevant media.

Recommendations:

- **Increase investment in multi-country productions to appeal to global audiences.**
- **Expand underrepresented genres like Documentaries and Family content to diversify the catalog.**
- **Continue focus on fresh content but explore older/classic media to attract niche viewers.**
- **Use genre and content age as input features for building recommendation systems or content strategy tools.**