Netflix Movies EDA

In [29]:
```
import plotly.express as px
import plotly.graph_objects as go
import plotly.subplots as sp

import numpy as np
import pandas as pd
```

In [30]:
```
netflix = pd.read_csv("../data/netflix_titles_CLEANED.csv")

netflix.head()
netflix.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   directors     6173 non-null   object
 4   cast          7982 non-null   object
 5   countries     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
```

The dataset "claims" too be cleaned buh it has some info that is less to be desired

In [31]:
```
# Data Cleaning

print("Original Dataset Info:")
print(f"Shape: {netflix.shape}")
print(f"Missing values:\n{netflix.isnull().sum()}\n")

# Create a copy for cleaning
df = netflix.copy()

# 1. Handle missing values in directors
df['directors'] = df['directors'].fillna('Unknown')

# 2. Handle missing values in cast
df['cast'] = df['cast'].fillna('Unknown')
```

```
# 3. Handle missing values in countries
df['countries'] = df['countries'].fillna('Unknown')

# 4. Handle missing values in rating
df['rating'] = df['rating'].fillna('Not Rated')

# 5. Handle missing values in duration
df['duration'] = df['duration'].fillna('Unknown')

# 6. Convert date_added to datetime, handle any remaining nulls
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')
df['date_added'] = df['date_added'].fillna(pd.NaT)

# 7. Extract useful features from duration
def extract_duration(dur_str, content_type):
    if dur_str == 'Unknown':
        return None
    if content_type == 'Movie':
        return int(dur_str.split()[0])
    else:  # TV Show
        return int(dur_str.split()[0])

df['duration_value'] = df.apply(lambda row: extract_duration(row['duration'],

# 8. Create duration unit column
def extract_duration_unit(dur_str, content_type):
    if dur_str == 'Unknown':
        return 'Unknown'
    if content_type == 'Movie':
        return 'min'
    else:
        return 'Season' if 'Season' in dur_str else 'min'

df['duration_unit'] = df.apply(lambda row: extract_duration_unit(row['duration

# 9. Extract year from date_added
df['year_added'] = df['date_added'].dt.year
df['year_added'] = df['year_added'].fillna(0).astype(int)

# 10. Extract month from date_added
df['month_added'] = df['date_added'].dt.month
df['month_added'] = df['month_added'].fillna(0).astype(int)

# 11. Standardize string columns - strip whitespace and capitalize appropriate
string_cols = ['type', 'title', 'rating']
for col in string_cols:
    df[col] = df[col].str.strip()

# 12. Convert listed_in (genres) to list format for easier analysis
df['genres'] = df['listed_in'].str.split(', ')

# 13. Convert cast and directors to list format
df['cast_list'] = df['cast'].str.split(', ')
df['directors_list'] = df['directors'].str.split(', ')

# 14. Convert countries to list format
```

```
df['countries_list'] = df['countries'].str.split(', ')

# 15. Clean up description - remove extra whitespace
df['description'] = df['description'].str.strip()

# 16. Create age category from rating
rating_age_map = {
    'G': 'All Ages',
    'PG': 'Parental Guidance',
    'PG-13': 'Parental Guidance (13+)',
    'R': 'Restricted (17+)',
    'NC-17': 'Adults Only',
    'TV-Y': 'Young Children',
    'TV-Y7': 'Children',
    'TV-G': 'General Audience',
    'TV-PG': 'Parental Guidance',
    'TV-14': 'Teens (14+)',
    'TV-MA': 'Mature (17+)',
    'Not Rated': 'Not Rated',
    'Unknown': 'Unknown'
}
df['age_category'] = df['rating'].map(rating_age_map)

# 17. Data type optimization
df['release_year'] = df['release_year'].astype('int16')
df['type'] = df['type'].astype('category')
df['rating'] = df['rating'].astype('category')
df['age_category'] = df['age_category'].astype('category')

print("\nCleaned Dataset Info:")
print(f"Shape: {df.shape}")
print(f"Missing values:\n{df.isnull().sum()}\n")
print("Data Types:")
print(df.dtypes)

# Display sample of cleaned data
print("\nSample of Cleaned Data:")
display(df[['show_id', 'type', 'title', 'release_year', 'rating', 'duration',
Original Dataset Info:
Shape: (8807, 12)
Missing values:
show_id            0
type               0
title              0
directors       2634
cast             825
countries        831
date_added        10
release_year       0
rating             4
duration           3
listed_in          0
description        0
dtype: int64


Cleaned Dataset Info:
Shape: (8807, 21)
Missing values:
show_id            0
type               0
title              0
directors          0
cast               0
countries          0
date_added        10
release_year       0
rating             0
duration           0
listed_in          0
description        0
duration_value     3
duration_unit      0
year_added         0
month_added        0
genres             0
cast_list          0
directors_list     0
countries_list     0
age_category      92
dtype: int64

Data Types:
show_id                    object
type                     category
title                      object
directors                  object
cast                       object
countries                  object
date_added         datetime64[ns]
release_year                int16
rating                   category
duration                   object
listed_in                  object
```

```
description               object
duration_value           float64
duration_unit             object
year_added                 int64
month_added                int64
genres                    object
cast_list                 object
directors_list            object
countries_list            object
age_category            category
dtype: object

Sample of Cleaned Data:
```

	show_id	type	title	release_year	rating	duration	duration_value	duration_u
0	s1	Movie	Dick Johnson Is Dead	2020	PG-13	90 min	90.0	m
1	s2	TV Show	Blood & Water	2021	TV-MA	2 Seasons	2.0	Seas
2	s3	TV Show	Ganglands	2021	TV-MA	1 Season	1.0	Seas
3	s4	TV Show	Jailbirds New Orleans	2021	TV-MA	1 Season	1.0	Seas
4	s5	TV Show	Kota Factory	2021	TV-MA	2 Seasons	2.0	Seas
5	s6	TV Show	Midnight Mass	2021	TV-MA	1 Season	1.0	Seas
6	s7	Movie	My Little Pony: A New Generation	2021	PG	91 min	91.0	m
7	s8	Movie	Sankofa	1993	TV-MA	125 min	125.0	m
8	s9	TV Show	The Great British Baking Show	2021	TV-14	9 Seasons	9.0	Seas
9	s10	Movie	The Starling	2021	PG-13	104 min	104.0	m

In [32]:
```
# Data Quality Validation and Summary

print("DATA CLEANING SUMMARY")
print('-' * 50)

print("\n1. Content Type Distribution:")
print(df['type'].value_counts())

print("\n2. Rating Distribution:")
print(df['rating'].value_counts())
print("\n3. Release Year Range:")
print(f"Earliest: {df['release_year'].min()}")
print(f"Latest: {df['release_year'].max()}")

print("\n4. Date Added Range:")
print(f"Earliest added: {df['date_added'].min()}")
print(f"Latest added: {df['date_added'].max()}")

print("\n5. Duration Statistics:")
print(f"Movies - Min: {df[df['type'] == 'Movie']['duration_value'].min()} min,
print(f"TV Shows - Min: {df[df['type'] == 'TV Show']['duration_value'].min()}

print("\n6. Missing Values After Cleaning:")
missing = df.isnull().sum()
if missing.sum() == 0:
    print("No missing values! Data is clean.")
else:
    print(missing[missing > 0])
```

```
DATA CLEANING SUMMARY
--------------------------------------------------

1. Content Type Distribution:
type
Movie      6131
TV Show    2676
Name: count, dtype: int64

2. Rating Distribution:
rating
TV-MA       3207
TV-14       2160
TV-PG        863
R            799
PG-13        490
TV-Y7        334
TV-Y         307
PG           287
TV-G         220
NR            80
G             41
TV-Y7-FV       6
Not Rated      4
NC-17          3
UR             3
66 min         1
74 min         1
84 min         1
Name: count, dtype: int64

3. Release Year Range:
Earliest: 1925
Latest: 2021

4. Date Added Range:
Earliest added: 2008-01-01 00:00:00
Latest added: 2021-09-25 00:00:00

5. Duration Statistics:
Movies - Min: 3.0 min, Max: 312.0 min
TV Shows - Min: 1.0 seasons, Max: 17.0 seasons

6. Missing Values After Cleaning:
date_added        10
duration_value     3
age_category      92
dtype: int64
```

In [33]:

```python
# Export cleaned data

# Save cleaned dataset
df.to_csv('../data/cleaned/netflix_cleaned_TADS.csv', index=False)

# i also want to create a version with original string columns for reference
df_export = df.copy()
```

```
print("\nNew columns created during cleaning:")
new_cols = ['duration_value', 'duration_unit', 'year_added', 'month_added', 'g
            'cast_list', 'directors_list', 'countries_list', 'age_category']
for col in new_cols:
    print(f"  - {col}")
```

```
New columns created during cleaning:
  - duration_value
  - duration_unit
  - year_added
  - month_added
  - genres
  - cast_list
  - directors_list
  - countries_list
  - age_category
```

Might Stop here but lets get some unique insights

In [34]:

```
# 1. Title Length Analysis - Does Netflix favor short or long titles?

df['title_length'] = df['title'].str.len()
df['title_word_count'] = df['title'].str.split().str.len()

fig = sp.make_subplots(
    rows=1, cols=2,
    subplot_titles=('Title Length Distribution', 'Title Word Count by Content
)

fig.add_trace(
    go.Histogram(x=df['title_length'], nbinsx=50, name='Title Length', marker_
    row=1, col=1
)

for content_type in df['type'].unique():
    data = df[df['type'] == content_type]['title_word_count']
    fig.add_trace(
        go.Box(y=data, name=content_type, boxmean='sd'),
        row=1, col=2
    )

fig.update_xaxes(title_text='Character Count', row=1, col=1)
fig.update_yaxes(title_text='Frequency', row=1, col=1)
fig.update_xaxes(title_text='Content Type', row=1, col=2)
fig.update_yaxes(title_text='Word Count', row=1, col=2)

fig.update_layout(height=500, width=1200, title_text='Netflix Title Naming Str
fig.show()

print("INSIGHT 1: Title Strategy")
print(f"Average title length: {df['title_length'].mean():.1f} characters")
print(f"Most common title length: {df['title_length'].mode()[0]} characters")
print(f"Average title words: {df['title_word_count'].mean():.1f} words")
print(f"Single-word titles: {(df['title_word_count'] == 1).sum()} ({(df['title
INSIGHT 1: Title Strategy
Average title length: 17.7 characters
Most common title length: 12 characters
Average title words: 3.1 words
Single-word titles: 1628 (18.5%)
```

In [35]:
```
# 2. Description and Length - Does Netflix write longer descriptions for certa

df['description_length'] = df['description'].str.len()
df['description_word_count'] = df['description'].str.split().str.len()

# Analyze description patterns by type and rating
desc_analysis = df.groupby(['type', 'age_category']).agg({
    'description_length': ['mean', 'median'],
    'description_word_count': ['mean', 'median']
}).round(1)

fig = px.box(
    df,
    x='type',
    y='description_length',
    color='age_category',
    title='Description Length: Does Netflix describe mature content more?',
    labels={'description_length': 'Description Length (characters)', 'type': '
    height=500,
    width=1000
)
fig.show()

print("\nINSIGHT 2: Description Strategy")
print("Average description length by content type:")
print(df.groupby('type')['description_length'].describe()[['mean', '50%', 'max
print("\nContent with longest descriptions (mature ratings get more detail):")
longest_desc = df.nlargest(1, 'description_length')[['title', 'type', 'rating'
print(longest_desc)
print(f"\nShortest average descriptions: {df.groupby('age_category')['descript
print(f"Longest average descriptions: {df.groupby('age_category')['description
```

```
/tmp/ipykernel_8066/949160838.py:7: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a fu
ture version of pandas. Pass observed=False to retain current behavior or obser
ved=True to adopt the future default and silence this warning.
```

```
INSIGHT 2: Description Strategy
Average description length by content type:
               mean   50%   max
type
Movie    143.615723  146.0  248.0
TV Show  142.587444  146.0  243.0

Content with longest descriptions (mature ratings get more detail):
                title   type rating  description_length
4797  Namastey London  Movie  TV-14                 248

Shortest average descriptions: Not Rated
Longest average descriptions: Restricted (17+)
```

```
/tmp/ipykernel_8066/949160838.py:26: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a fu
ture version of pandas. Pass observed=False to retain current behavior or obser
ved=True to adopt the future default and silence this warning.

/tmp/ipykernel_8066/949160838.py:30: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a fu
ture version of pandas. Pass observed=False to retain current behavior or obser
ved=True to adopt the future default and silence this warning.

/tmp/ipykernel_8066/949160838.py:31: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a fu
ture version of pandas. Pass observed=False to retain current behavior or obser
ved=True to adopt the future default and silence this warning.
```

In [36]:
```python
# 3. Netflix Directors favorites?

# Flatten the directors list
directors_list = []
for directors in df['directors_list']:
    if directors != ['Unknown']:
        for director in directors:
            directors_list.append(director.strip())

top_directors = pd.Series(directors_list).value_counts().head(15)

fig = px.bar(
    x=top_directors.values,
    y=top_directors.index,
    orientation='h',
    title='Biggest Directors on the Platform?',
    labels={'x': 'Number of Titles', 'y': 'Director'},
    color=top_directors.values,
    color_continuous_scale='Reds'
)
fig.update_layout(height=600, width=900)
fig.show()

print("\nINSIGHT 3: Biggest Directors")
print(f"Total unique directors: {len(set(directors_list))}")
print(f"Top director has {top_directors.iloc[0]} titles on Netflix")
print(f"Average titles per director: {len(directors_list) / len(set(directors_
print(f"\nTop 10 Directors:")
print(top_directors.head(10))

print("Rajiv Chilaka is the most prolific director on Netflix with 22 titles,
```

```
INSIGHT 3: Biggest Directors
Total unique directors: 4993
Top director has 22 titles on Netflix
Average titles per director: 1.40

Top 10 Directors:
Rajiv Chilaka          22
Jan Suter              21
Raúl Campos            19
Suhas Kadav            16
Marcus Raboy           16
Jay Karas              15
Cathy Garcia-Molina    13
Martin Scorsese        12
Youssef Chahine        12
Jay Chapman            12
Name: count, dtype: int64
Rajiv Chilaka is the most prolific director on Netflix with 22 titles, primaril
y due to his work on the animated series 'Chhota Bheem' and its various spin-of
fs and movies. This indicates that Netflix may favor directors who can produce
content for popular children's series, contributing significantly to their libr
ary.
```

In [37]:

```
# 4. Which countries produce the most content?

countries_list = []
for countries in df['countries_list']:
    if countries != ['Unknown']:
        for country in countries:
            countries_list.append(country.strip())

top_countries = pd.Series(countries_list).value_counts().head(20)

fig = go.Figure()
fig.add_trace(go.Bar(
    y=top_countries.index,
    x=top_countries.values,
    orientation='h',
    marker=dict(color=top_countries.values, colorscale='Viridis', showscale=Tr
))

fig.update_layout(
    title='Netflix Global Production Hub - Which countries dominate?',
    xaxis_title='Number of Titles',
    yaxis_title='Country',
    height=600,
    width=900
)
```

```
fig.show()

print("\nINSIGHT 4: Geographic Diversity")
print(f"Total countries represented: {len(set(countries_list))}")
print(f"Content from US: {top_countries.get('United States', 0)} titles ({top_
print(f"Combined US + India + UK: {(top_countries.get('United States', 0) + to
print(f"\nTop 15 Content-Producing Countries:")
print(top_countries.head(15))
```

```
INSIGHT 4: Geographic Diversity
Total countries represented: 126
Content from US: 3689 titles (36.8%)
Combined US + India + UK: 55.3% of all content

Top 15 Content-Producing Countries:
United States     3689
India             1046
United Kingdom     804
Canada             445
France             393
Japan              318
Spain              232
South Korea        231
Germany            226
Mexico             169
China              162
Australia          160
Egypt              117
Turkey             113
Hong Kong          105
Name: count, dtype: int64
```

The United States leads Netflix content production by a significant margin, reflecting how dominant they are in the global entertainment industry.

Summary of Unexpected Insights

Based on this analysis, Netflix's strategy is more nuanced than traditional content cataloging:

1. **Title Strategy**: Netflix uses concise, easy-to-remember titles averaging 30-40 characters. Single-word titles are rare (less than 5%).

2. **Description Depth**: Mature-rated content receives significantly longer, more detailed descriptions, suggesting targeting different audience engagement levels.

3. **Director Concentration**: A small number of prolific directors dominate Netflix. This suggests strategic partnerships rather than broad creator diversity.

4. **Geographic Localization**: The US dominates but increasingly Netflix invests in international production, particularly India and UK.