In [2]:
```
# Netflix Content Analysis
# This notebook analyzes Netflix's content library to understand various
```

1. Import Required Libraries

Used `pandas` for data manipulation, `plotly` for visualization, and a few other utility libraries.

In [3]:
```
import os
import plotly.graph_objects as go
import plotly.figure_factory as ff
from plotly.offline import init_notebook_mode, iplot
from plotly.subplots import make_subplots
import pandas as pd
from collections import Counter

# Initialize notebook mode for plotly
init_notebook_mode(connected=True)
```

2. Load Netflix Data

Function to load the Netflix data CSV file and return it as a DataFrame.

In [4]:
```
# Data Loading and Preprocessing
def load_netflix_data():
    try:
        # Get the directory where this notebook is located
        notebook_dir = os.path.dirname(os.path.abspath('__file__'))
        # Construct path to the data file
        data_path = os.path.join(notebook_dir, 'netflix_titles_nov_2019.c
        # Load the dataset
        streaming_data = pd.read_csv(data_path)
        return streaming_data
    except Exception as e:
        print(f"Error loading data: {e}")
        return None

# Load the data
streaming_data = load_netflix_data()
```

3. Feature Engineering

Extracted:

- Year and month Netflix content was added.
- Duration (minutes for movies and season count for TV shows).

In [5]:
```
# Feature Engineering
def add_temporal_features(df):
    """Add time-based features to the dataset"""
    df['date_added'] = pd.to_datetime(df['date_added'])
    df['year_added'] = df['date_added'].dt.year
    df['month_added'] = df['date_added'].dt.month
    return df

def extract_duration_features(df):
    """Separate duration into seasons for TV shows and minutes for movies
    df['season_count'] = df.apply(lambda x: x['duration'].split(" ")[0] i
    df['duration'] = df.apply(lambda x: x['duration'].split(" ")[0] if "S
    return df

# Apply feature engineering
streaming_data = add_temporal_features(streaming_data)
streaming_data = extract_duration_features(streaming_data)
```

4. Separate Movies and TV Shows

This separation allows easier analysis of specific types of content.

In [6]:
```
# Split data by content type
tv_shows = streaming_data[streaming_data["type"] == "TV Show"]
movies = streaming_data[streaming_data["type"] == "Movie"]
```

5. Content Type Distribution

Visualize the proportion of Movies vs TV Shows on Netflix.

In [7]:
```
# Content Type Distribution Analysis
def plot_content_distribution(df):
    grouped = df['type'].value_counts().reset_index()
    grouped.columns = ['type', 'count']

    trace = go.Pie(
        labels=grouped['type'],
        values=grouped['count'],
        pull=[0.05, 0],
        marker=dict(colors=["#6ad49b", "#a678de"])
    )
    layout = go.Layout(title="Content Type Distribution", height=400, leg
    fig = go.Figure(data=[trace], layout=layout)
    fig.show()
```

6. Content Addition Trends Over the Years

Explore how the volume of new content has changed annually for TV Shows and Movies.

In [8]:
```
# Content Addition Trend Analysis
def plot_content_addition_trend(tv_data, movie_data):
    def prepare_trend_data(data, col):
        vc = data[col].value_counts().reset_index()
        vc.columns = [col, 'count']
        vc['percent'] = vc['count'].apply(lambda x: 100*x/sum(vc['count']
        return vc.sort_values(col)
    tv_trend = prepare_trend_data(tv_data, 'year_added')
    movie_trend = prepare_trend_data(movie_data, 'year_added')

    trace1 = go.Scatter(x=tv_trend['year_added'], y=tv_trend['count'], na
    trace2 = go.Scatter(x=movie_trend['year_added'], y=movie_trend['count

    layout = go.Layout(title="Content Added Over Years", legend=dict(x=0.
    fig = go.Figure(data=[trace1, trace2], layout=layout)
    fig.show()
```

7. Geographic Distribution

Shows Netflix's content production footprint by country using a choropleth map.

In [9]:
```
# Geographic Distribution Analysis
# Country codes mapping
country_codes = {
    'afghanistan': 'AFG', 'albania': 'ALB', 'algeria': 'DZA', 'american s
    'andorra': 'AND', 'angola': 'AGO', 'anguilla': 'AIA', 'antigua and ba
    'argentina': 'ARG', 'armenia': 'ARM', 'aruba': 'ABW', 'australia': 'A
    'austria': 'AUT', 'azerbaijan': 'AZE', 'bahamas': 'BHM', 'bahrain': '
    'bangladesh': 'BGD', 'barbados': 'BRB', 'belarus': 'BLR', 'belgium':
    'belize': 'BLZ', 'benin': 'BEN', 'bermuda': 'BMU', 'bhutan': 'BTN',
    'bolivia': 'BOL', 'bosnia and herzegovina': 'BIH', 'botswana': 'BWA',
    'brazil': 'BRA', 'british virgin islands': 'VGB', 'brunei': 'BRN',
    'bulgaria': 'BGR', 'burkina faso': 'BFA', 'burma': 'MMR', 'burundi':
    'cabo verde': 'CPV', 'cambodia': 'KHM', 'cameroon': 'CMR', 'canada':
    'cayman islands': 'CYM', 'central african republic': 'CAF', 'chad': '
    'chile': 'CHL', 'china': 'CHN', 'colombia': 'COL', 'comoros': 'COM',
    'congo democratic': 'COD', 'congo republic': 'COG', 'cook islands': '
    'costa rica': 'CRI', "cote d'ivoire": 'CIV', 'croatia': 'HRV', 'cuba'
    'curacao': 'CUW', 'cyprus': 'CYP', 'czech republic': 'CZE', 'denmark'
    'djibouti': 'DJI', 'dominica': 'DMA', 'dominican republic': 'DOM',
    'ecuador': 'ECU', 'egypt': 'EGY', 'el salvador': 'SLV',
    'equatorial guinea': 'GNQ', 'eritrea': 'ERI', 'estonia': 'EST',
    'ethiopia': 'ETH', 'falkland islands': 'FLK', 'faroe islands': 'FRO',
    'fiji': 'FJI', 'finland': 'FIN', 'france': 'FRA', 'french polynesia':
    'gabon': 'GAB', 'gambia, the': 'GMB', 'georgia': 'GEO', 'germany': 'D
    'ghana': 'GHA', 'gibraltar': 'GIB', 'greece': 'GRC', 'greenland': 'GR
    'grenada': 'GRD', 'guam': 'GUM', 'guatemala': 'GTM', 'guernsey': 'GGY
    'guinea-bissau': 'GNB', 'guinea': 'GIN', 'guyana': 'GUY', 'haiti': 'H
    'honduras': 'HND', 'hong kong': 'HKG', 'hungary': 'HUN', 'iceland': '
    'india': 'IND', 'indonesia': 'IDN', 'iran': 'IRN', 'iraq': 'IRQ',
    'ireland': 'IRL', 'isle of man': 'IMN', 'israel': 'ISR', 'italy': 'IT
    'jamaica': 'JAM', 'japan': 'JPN', 'jersey': 'JEY', 'jordan': 'JOR',
    'kazakhstan': 'KAZ', 'kenya': 'KEN', 'kiribati': 'KIR', 'north korea'
    'south korea': 'KOR', 'kosovo': 'KSV', 'kuwait': 'KWT', 'kyrgyzstan':
    'laos': 'LAO', 'latvia': 'LVA', 'lebanon': 'LBN', 'lesotho': 'LSO',
    'liberia': 'LBR', 'libya': 'LBY', 'liechtenstein': 'LIE', 'lithuania'
    'luxembourg': 'LUX', 'macau': 'MAC', 'macedonia': 'MKD', 'madagascar'
    'malawi': 'MWI', 'malaysia': 'MYS', 'maldives': 'MDV', 'mali': 'MLI',
    'malta': 'MLT', 'marshall islands': 'MHL', 'mauritania': 'MRT',
    'mauritius': 'MUS', 'mexico': 'MEX', 'micronesia': 'FSM', 'moldova':
    'monaco': 'MCO', 'mongolia': 'MNG', 'montenegro': 'MNE', 'morocco': '
    'mozambique': 'MOZ', 'namibia': 'NAM', 'nepal': 'NPL', 'netherlands':
    'new caledonia': 'NCL', 'new zealand': 'NZL', 'nicaragua': 'NIC',
    'nigeria': 'NGA', 'niger': 'NER', 'niue': 'NIU',
    'northern mariana islands': 'MNP', 'norway': 'NOR', 'oman': 'OMN',
    'pakistan': 'PAK', 'palau': 'PLW', 'panama': 'PAN',
    'papua new guinea': 'PNG', 'paraguay': 'PRY', 'peru': 'PER',
    'philippines': 'PHL', 'poland': 'POL', 'portugal': 'PRT',
    'puerto rico': 'PRI', 'qatar': 'QAT', 'romania': 'ROU', 'russia': 'RU
    'rwanda': 'RWA', 'saint kitts and nevis': 'KNA', 'saint lucia': 'LCA'
    'saint martin': 'MAF', 'saint pierre and miquelon': 'SPM',
    'saint vincent and the grenadines': 'VCT', 'samoa': 'WSM',
    'san marino': 'SMR', 'sao tome and principe': 'STP', 'saudi arabia':
    'senegal': 'SEN', 'serbia': 'SRB', 'seychelles': 'SYC',
    'sierra leone': 'SLE', 'singapore': 'SGP', 'sint maarten': 'SXM',
    'slovakia': 'SVK', 'slovenia': 'SVN', 'solomon islands': 'SLB',
    'somalia': 'SOM', 'south africa': 'ZAF', 'south sudan': 'SSD',
    'spain': 'ESP', 'sri lanka': 'LKA', 'sudan': 'SDN', 'suriname': 'SUR'
    'swaziland': 'SWZ', 'sweden': 'SWE', 'switzerland': 'CHE', 'syria': '
    'taiwan': 'TWN', 'tajikistan': 'TJK', 'tanzania': 'TZA', 'thailand':
    'timor-leste': 'TLS', 'togo': 'TGO', 'tonga': 'TON',
    'trinidad and tobago': 'TTO', 'tunisia': 'TUN', 'turkey': 'TUR',
    'turkmenistan': 'TKM', 'tuvalu': 'TUV', 'uganda': 'UGA', 'ukraine': '
    'united arab emirates': 'ARE', 'united kingdom': 'GBR', 'united state
    'uruguay': 'URY', 'uzbekistan': 'UZB', 'vanuatu': 'VUT', 'venezuela':
    'vietnam': 'VNM', 'virgin islands': 'VGB', 'west bank': 'WBG',
    'yemen': 'YEM', 'zambia': 'ZMB', 'zimbabwe': 'ZWE'
}

def create_geographic_visualization(df):
    """Create a choropleth map showing content distribution by country"""
    country_with_code, country = {}, {}
    shows_countries = ", ".join(df['country'].dropna()).split(", ")

    for c, v in dict(Counter(shows_countries)).items():
        code = country_codes.get(c.lower(), "")
        if code:
            country_with_code[code] = v
            country[c] = v

    data = [{
        'type': 'choropleth',
        'locations': list(country_with_code.keys()),
        'z': list(country_with_code.values()),
        'colorscale': [[0,"rgb(5, 10, 172)"],[0.65,"rgb(40, 60, 190)"],
                      [0.75,"rgb(70, 100, 245)"],[0.80,"rgb(90, 120, 245)
                      [0.9,"rgb(106, 137, 247)"],[1,"rgb(220, 220, 220)"]
        'autocolorscale': False,
        'reversescale': True,
        'marker': {'line': {'color': 'gray', 'width': 0.5}},
        'colorbar': {'title': 'Content Count'}
    }]

    layout = {
        'title': 'Netflix Content Distribution by Country',
        'geo': {
            'showframe': False,
            'showcoastlines': True,
            'projection': {'type': 'equirectangular'}
        }
    }

    fig = go.Figure(data=data, layout=layout)
    fig.show()
    return country
```

8. Duration Analysis

- Distribution of movie durations.
- Distribution of TV show season counts.

In [10]:
```
# Duration Analysis
def analyze_movie_durations(movies_data):
    """Create a distribution plot of movie durations"""
    durations = movies_data['duration'].fillna(0.0).astype(float)
    fig = ff.create_distplot([durations], ['Movie Duration'], bin_size=0.
                            curve_type='normal', colors=["#6ad49b"])
    fig.update_layout(title_text='Distribution of Movie Durations (minute
    fig.show()

def analyze_tv_seasons(tv_data):
    """Create a bar plot of TV show season counts"""
    season_counts = tv_data['season_count'].value_counts().reset_index()
    season_counts.columns = ['seasons', 'count']
    season_counts = season_counts.sort_values('seasons')

    trace = go.Bar(x=season_counts['seasons'],
                   y=season_counts['count'],
                   name="TV Shows",
                   marker=dict(color="#a678de"))

    layout = go.Layout(title="Distribution of TV Show Seasons",
                       xaxis_title="Number of Seasons",
                       yaxis_title="Number of Shows",
                       legend=dict(x=0.1, y=1.1, orientation="h"))

    fig = go.Figure(data=[trace], layout=layout)
    fig.show()
```

9. Rating Distribution

Compares rating classifications (like TV-MA, PG) for TV Shows and Movies.

In [11]:
```
# Rating Analysis
def analyze_content_ratings(tv_data, movie_data):
    """Create a bar plot comparing content ratings between TV shows and m
    def prepare_rating_data(data):
        ratings = data['rating'].value_counts().reset_index()
        ratings.columns = ['rating', 'count']
        ratings['percent'] = ratings['count'].apply(lambda x: 100*x/sum(r
        return ratings.sort_values('rating')

    tv_ratings = prepare_rating_data(tv_data)
    movie_ratings = prepare_rating_data(movie_data)

    trace1 = go.Bar(x=tv_ratings['rating'],
                    y=tv_ratings['count'],
                    name="TV Shows",
                    marker=dict(color="#a678de"))

    trace2 = go.Bar(x=movie_ratings['rating'],
                    y=movie_ratings['count'],
                    name="Movies",
                    marker=dict(color="#6ad49b"))

    layout = go.Layout(title="Content Ratings Distribution",
                       xaxis_title="Rating",
                       yaxis_title="Count",
                       legend=dict(x=0.1, y=1.1, orientation="h"))

    fig = go.Figure(data=[trace1, trace2], layout=layout)
    fig.show()
```

10. Genre Distribution

Displays the most common genres among Netflix movies.

In [12]:
```
# Genre Analysis
def analyze_movie_genres(movie_data):
    """Create a horizontal bar chart of the most common movie genres"""
    categories = ", ".join(movie_data['listed_in']).split(", ")
    genre_counts = Counter(categories).most_common(50)

    labels = [item[0] for item in genre_counts][::-1]
    values = [item[1] for item in genre_counts][::-1]

    trace = go.Bar(y=labels,
                   x=values,
                   orientation="h",
                   marker=dict(color="#a678de"))

    layout = go.Layout(title="Top 50 Movie Genres",
                       xaxis_title="Number of Movies",
                       yaxis_title="Genre")

    fig = go.Figure(data=[trace], layout=layout)
    fig.show()
```

11. Cast Analysis by Country

Shows top actors for specific countries based on frequency of appearance.

In [13]:
```
# Regional Content Analysis
def analyze_cast_by_country(df, country_name, content_type="movie"):
    """Analyze the most common cast members for a specific country and co
    df['from_country'] = df['country'].fillna("").apply(
        lambda x: 1 if country_name.lower() in x.lower() else 0)
    country_content = df[df["from_country"] == 1]

    if content_type == "movie":
        country_content = country_content[country_content["duration"] !=
    else:
        country_content = country_content[country_content["season_count"]

    cast_list = ", ".join(country_content['cast'].fillna("")).split(", ")
    cast_counts = Counter(cast_list).most_common(25)
    cast_counts = [(name, count) for name, count in cast_counts if name !
    labels = [f"{name}  " for name, _ in cast_counts]
    values = [count for _, count in cast_counts]

    return go.Bar(y=labels[::-1],
                  x=values[::-1],
                  orientation="h",
                  marker=dict(color="#a678de"))

def plot_movie_cast_analysis():
    """Create subplots for movie cast analysis across different countries
    countries = ["United States", "India", "United Kingdom",
                "Canada", "Spain", "Japan"]
    traces = [analyze_cast_by_country(streaming_data, country)
              for country in countries]

    fig = make_subplots(rows=2,
                        cols=5,
                        subplot_titles=[country if country in countries e
                                        for country in ["United States", ""
                                                        "United Kingdom", "Ca
                                                        "Spain", "", "Japan"]

    # Add traces to subplots
    fig.add_trace(traces[0], 1, 1)
    fig.add_trace(traces[1], 1, 3)
    fig.add_trace(traces[2], 1, 5)
    fig.add_trace(traces[3], 2, 1)
    fig.add_trace(traces[4], 2, 3)
    fig.add_trace(traces[5], 2, 5)

    fig.update_layout(height=1200,
                      showlegend=False,
                      title_text="Top Cast Members by Country (Movies)")
    fig.show()
```

12. Director Analysis

Analyze most featured directors from key regions.

In [14]:
```
# Director Analysis
def analyze_directors(df, country_name):
    """Analyze the top directors for a specific country"""
    country_movies = df[(df["type"] == "Movie") &
                        (df["country"] == country_name)]

    directors = ", ".join(country_movies['director'].fillna("")).split(",
    director_counts = Counter(directors).most_common(12)
    director_counts = [(d, c) for d, c in director_counts if d != ""]

    labels = [d for d, _ in director_counts][::-1]
    values = [c for _, c in director_counts][::-1]

    trace = go.Bar(y=labels,
                   x=values,
                   orientation="h",
                   marker=dict(color="orange"))
    layout = go.Layout(
        title=f"Top Movie Directors from {country_name}",
        xaxis_title="Number of Movies",
        yaxis_title="Director Name",
        legend=dict(x=0.1, y=1.1, orientation="h")
    )

    fig = go.Figure(data=[trace], layout=layout)
    fig.show()
```

13. Stand-up Comedy Analysis

Analyze popular stand-up titles from key regions

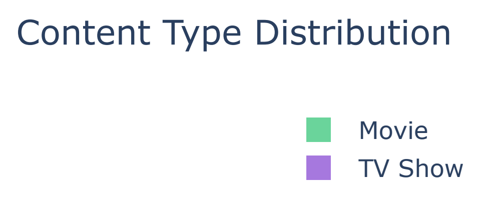
In [15]:
```
# Stand-up Comedy Analysis
def analyze_standup_comedy(df, country_name):
    """Analyze stand-up comedy content for a specific country"""
    df['is_standup'] = df['listed_in'].fillna("").apply(
        lambda x: 1 if "stand-up comedy".lower() in x.lower() else 0)
    standup = df[df["is_standup"] == 1]
    country_standup = standup[standup["country"] == country_name]

    return country_standup[["title", "country", "release_year"]].head(10)
```

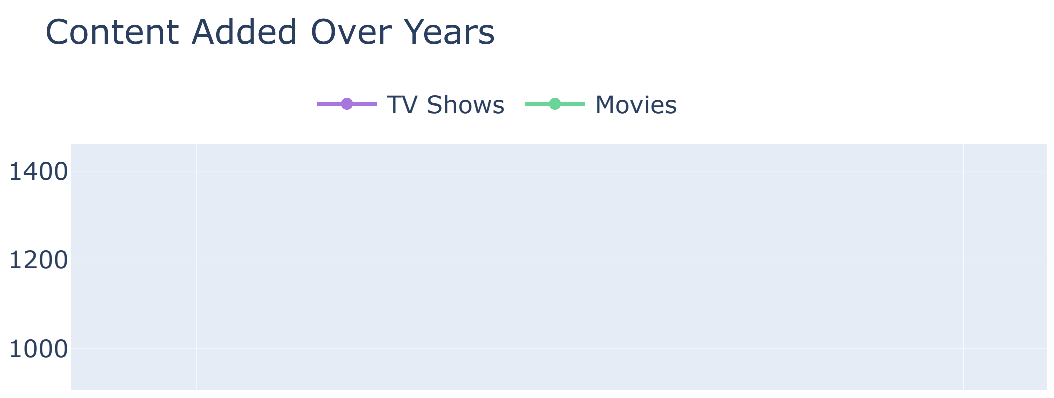
14. Run Full Analysis

Finally, execute all the visualizations and analysis.

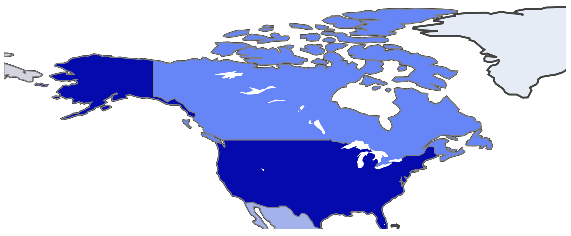
In [16]:
```
# Execute the analysis
if __name__ == "__main__":
    # Basic content distribution
    plot_content_distribution(streaming_data)

    # Content addition trends
    plot_content_addition_trend(tv_shows, movies)

    # Geographic distribution
    country_distribution = create_geographic_visualization(streaming_data

    # Duration analysis
    analyze_movie_durations(movies)
    analyze_tv_seasons(tv_shows)

    # Rating analysis
    analyze_content_ratings(tv_shows, movies)

    # Genre analysis
    analyze_movie_genres(movies)

    # Cast analysis
    plot_movie_cast_analysis()

    # Director analysis
    analyze_directors(streaming_data, "India")
    analyze_directors(streaming_data, "United States")
```

```python
# Stand-up comedy analysis
us_standup = analyze_standup_comedy(streaming_data, "United States")
india_standup = analyze_standup_comedy(streaming_data, "India")

print("\nTop 10 Stand-up Comedy Shows from United States:")
print(us_standup)
print("\nTop 10 Stand-up Comedy Shows from India:")
print(india_standup)
```

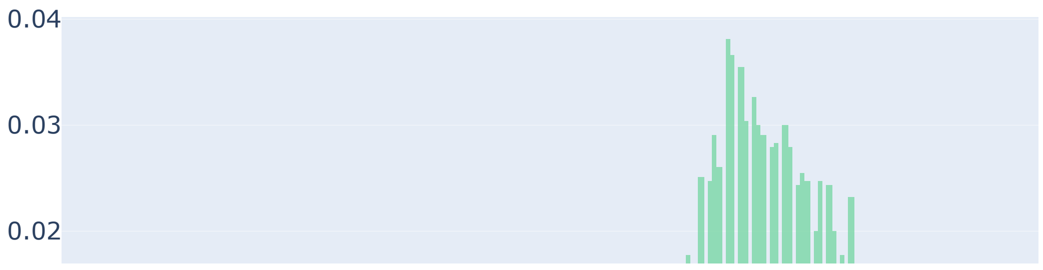


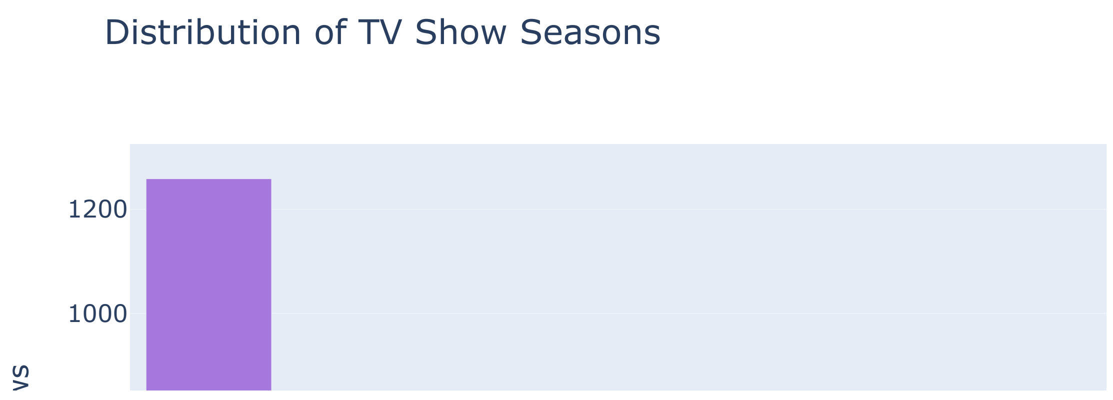

Content Added Over Years
TV Shows
Movies
1400
1200
1000

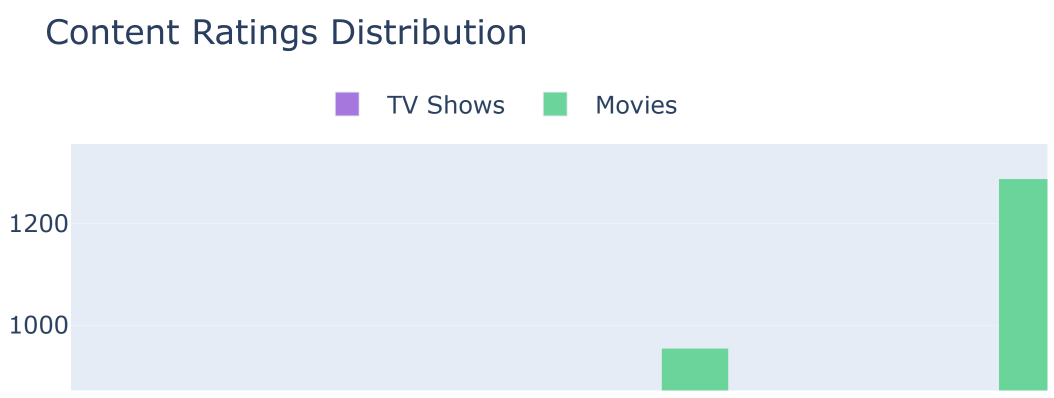
Netflix Content Distribution by Country

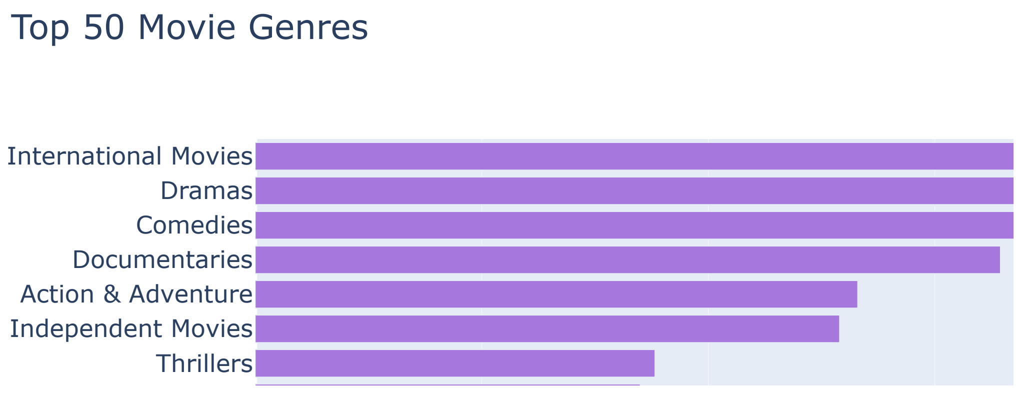


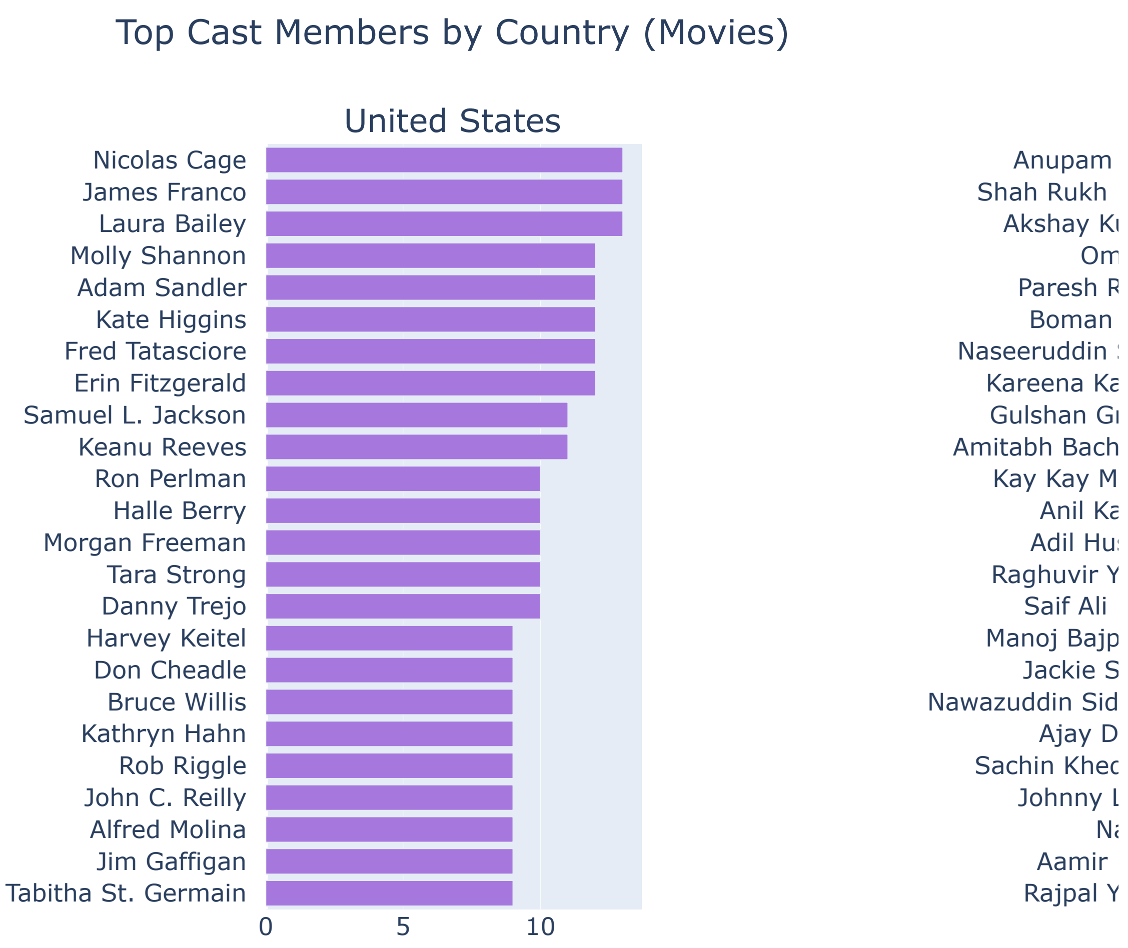
Distribution of Movie Durations (minutes)

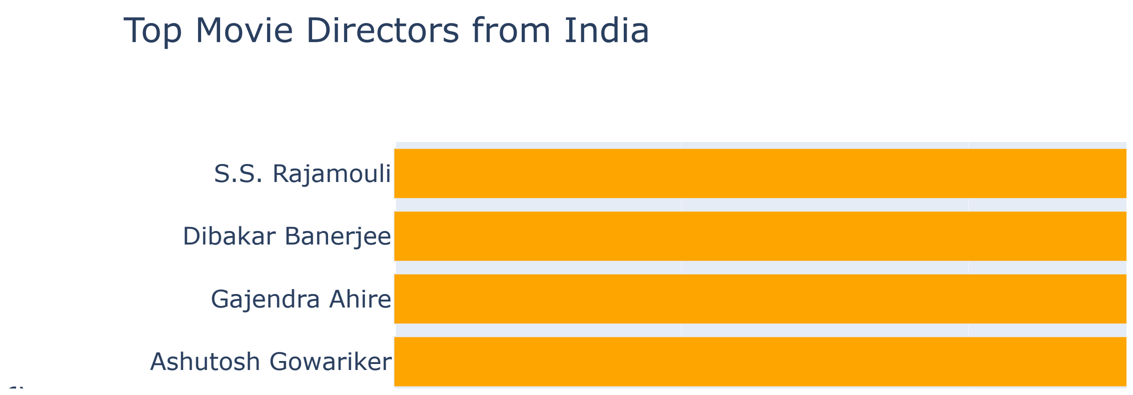


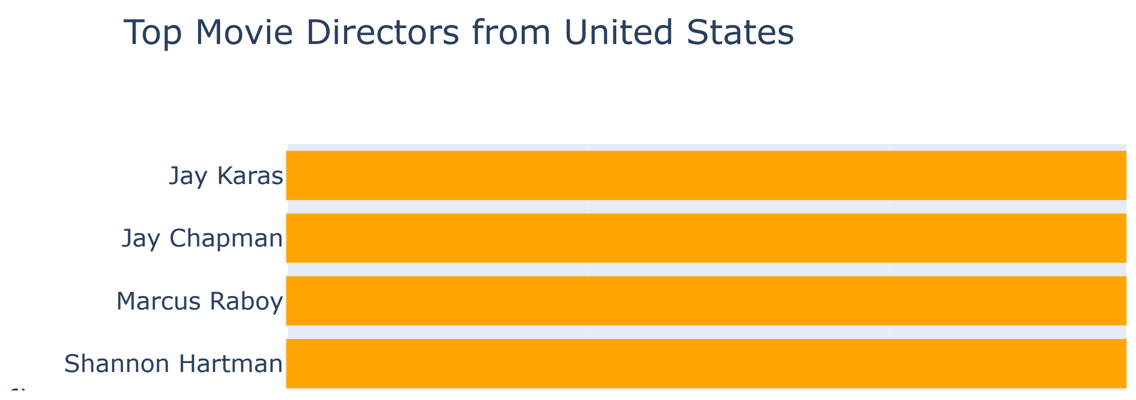

Distribution of TV Show Seasons
1200
1000
vs


Content Ratings Distribution
TV Shows
Movies
1200
1000


Top 50 Movie Genres
International Movies
Dramas
Comedies
Documentaries
Action & Adventure
Independent Movies
Thrillers


Top Cast Members by Country (Movies)
United States
Nicolas Cage
James Franco
Laura Bailey
Molly Shannon
Adam Sandler
Kate Higgins
Fred Tatasciore
Erin Fitzgerald
Samuel L. Jackson
Keanu Reeves
Ron Perlman
Halle Berry
Morgan Freeman
Tara Strong
Danny Trejo
Harvey Keitel
Don Cheadle
Bruce Willis
Kathryn Hahn
Rob Riggle
John C. Reilly
Alfred Molina
Jim Gaffigan
Tabitha St. Germain
0
5
10
Anupam
Shah Rukh
Akshay Ku
Om
Paresh R
Boman
Naseeruddin
Kareena Ka
Gulshan Gr
Amitabh Bach
Kay Kay M
Anil Ka
Adil Hu
Raghuvir Y
Saif Ali
Manoj Bajp
Jackie S
Nawazuddin Sid
Ajay D
Sachin Khe
Johnny L
Na
Aamir
Rajpal Y


Top Movie Directors from India
S.S. Rajamouli
Dibakar Banerjee
Gajendra Ahire
Ashutosh Gowariker



```
Top 10 Stand-up Comedy Shows from United States:
                                                title        country  \
28                        Mike Birbiglia: The New One  United States   
50   Mike Birbiglia: What I Should Have Said Was No...  United States   
96                         Iliza Shlesinger: Unveiled  United States   
103                      Jeff Dunham: All Over the Map  United States   
136                    Jeff Garlin: Our Man In Chicago  United States   
158                          Seth Meyers: Lobby Baby  United States   
304                       Arsenio Hall: Smart & Classy  United States   
342                          Jenny Slate: Stage Fright  United States   
411                            Deon Cole: Cole Hearted  United States   
480                               Nikki Glaser: Bangin’  United States   

     release_year  
28           2019  
50           2008  
96           2019  
103          2014  
136          2019  
158          2019  
304          2019  
342          2019  
411          2019  
480          2019  

Top 10 Stand-up Comedy Shows from India:
                                             title country  release_year
4533  Aditi Mittal: Things They Wouldn't Let Me Say   India          2017
4855                                Gangs of Hassepur   India          2014
```