netflix_analysis

June 11, 2025

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
from sklearn.preprocessing import MultiLabelBinarizer
from datetime import datetime
import warnings

#Configuration
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns', None)
import seaborn as sns
sns.set_theme(style="whitegrid", palette="muted")
sns.set_theme(style="whitegrid", palette="muted")
COLOR_PALETTE = ['#E50914', '#221F1F', '#B81D24', '#F5F5F1', '#000000']


#DATA LOADING & CLEANING
def load_and_clean_data():
    """Load and preprocess Netflix data with comprehensive cleaning"""
    try:
        df = pd.read_csv('netflix.csv')
        print(" Data loaded successfully (rows: {}, cols: {})".format(*df.
 ↪shape))
    except FileNotFoundError:
        raise FileNotFoundError(" Error: File 'netflix.csv' not found")

    # Date processing
    df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')
    df['year_added'] = df['date_added'].dt.year
    df['month_added'] = df['date_added'].dt.month_name()

    # Country processing (extract primary country)
    df['country'] = df['country'].fillna('Unknown').apply(
        lambda x: x.split(',')[0].strip() if isinstance(x, str) else 'Unknown')
```

```
    # Duration processing
    df['duration'] = df['duration'].fillna('0')
    df['duration_min'] = df.apply(
        lambda x: int(x['duration'].split()[0]) if 'min' in x['duration'] else␣
 ↪np.nan, axis=1)
    df['duration_seasons'] = df.apply(
        lambda x: int(x['duration'].split()[0]) if 'Season' in x['duration']␣
 ↪else np.nan, axis=1)

    # Genre processing
    df['genres'] = df['listed_in'].str.split(', ').fillna('').apply(
        lambda x: [genre.strip() for genre in x])

    # Director/actor processing
    df['directors'] = df['director'].str.split(', ').fillna('')
    df['actors'] = df['cast'].str.split(', ').fillna('')

    # Content rating standardization
    rating_map = {
        'TV-MA': 'Adult',
        'TV-14': 'Teen',
        'TV-PG': 'Older Kids',
        'R': 'Adult',
        'PG-13': 'Teen',
        'PG': 'Older Kids',
        'G': 'Kids',
        'NC-17': 'Adult',
        'TV-Y7': 'Older Kids',
        'TV-Y7-FV': 'Older Kids',
        'TV-Y': 'Kids',
        'NR': 'Adult',
        'UR': 'Adult'
    }
    df['rating_category'] = df['rating'].map(rating_map).fillna('Other')
    return df
df = load_and_clean_data()

#EXPLORATORY ANALYSIS
def comprehensive_eda(df):
    print("\n  COMPREHENSIVE EXPLORATORY ANALYSIS")

    # Basic statistics
    print("\n  BASIC STATISTICS:")
    print(f"- Dataset spans {df['release_year'].min()} to {df['release_year'].
 ↪max()}")
    print(f"- Latest addition: {df['date_added'].max().strftime('%Y-%m-%d')}")
    print(f"- {df['type'].value_counts(normalize=True).to_string()}")
```

```
    # Missing values analysis
    print("\n MISSING VALUES:")
    missing = df.isnull().sum().sort_values(ascending=False)
    print(missing[missing > 0].to_string())

    # Temporal analysis
    print("\n TEMPORAL PATTERNS:")
    print(f"- Peak release year: {df['release_year'].mode()[0]}")
    print(f"- Peak addition year: {df['year_added'].mode()[0]}")

    # Content characteristics
    print("\n CONTENT CHARACTERISTICS:")
    print(f"- Avg movie duration: {df['duration_min'].mean():.1f} minutes")
    print(f"- Avg show seasons: {df['duration_seasons'].mean():.1f}")
    return df
df = comprehensive_eda(df)

#ADVANCED VISUALIZATIONS
def create_advanced_visualizations(df):
    """Generate sophisticated visualizations with insights"""
    plt.figure(figsize=(20, 18))

    # 1. Content Type Evolution Over Time
    plt.subplot(3, 2, 1)
    type_year = df.groupby(['release_year', 'type']).size().unstack()
    type_year.plot(kind='area', stacked=True, color=['#E50914', '#221F1F'],␣
 ↪alpha=0.8, ax=plt.gca())
    plt.title('Content Type Evolution (1925-2021)', fontsize=12)
    plt.xlabel('Release Year')
    plt.ylabel('Count')
    plt.legend(title='Content Type')

    # 2. Monthly Addition Patterns
    plt.subplot(3, 2, 2)
    month_order = ['January', 'February', 'March', 'April', 'May', 'June',
                   'July', 'August', 'September', 'October', 'November',␣
 ↪'December']
    monthly = df['month_added'].value_counts().reindex(month_order)
    sns.barplot(x=monthly.index, y=monthly.values, palette='viridis')
    plt.title('Monthly Addition Patterns', fontsize=12)
    plt.xlabel('Month')
    plt.ylabel('Titles Added')
    plt.xticks(rotation=45)

    # 3. Movie Duration Distribution
    plt.subplot(3, 2, 3)
```

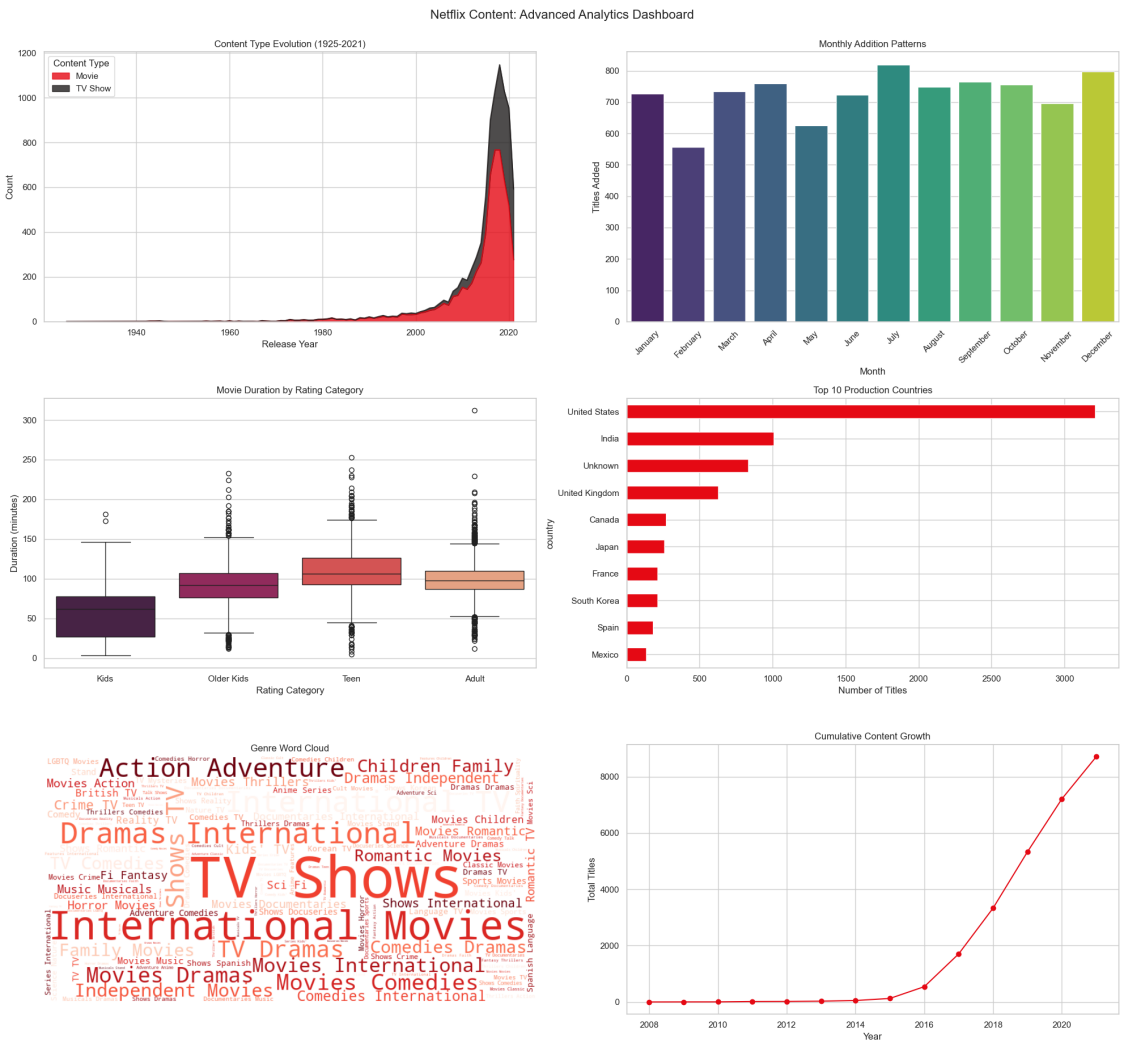
```
    sns.boxplot(x='rating_category', y='duration_min', data=df,
                palette='rocket', order=['Kids', 'Older Kids', 'Teen', 'Adult'])
    plt.title('Movie Duration by Rating Category', fontsize=12)
    plt.xlabel('Rating Category')
    plt.ylabel('Duration (minutes)')

    # 4. Top Production Countries
    plt.subplot(3, 2, 4)
    countries = df['country'].value_counts().nlargest(10).sort_values()
    countries.plot(kind='barh', color='#E50914')
    plt.title('Top 10 Production Countries', fontsize=12)
    plt.xlabel('Number of Titles')

    # 5. Genre Word Cloud
    plt.subplot(3, 2, 5)
    genres_text = ' '.join([' '.join(genres) for genres in df['genres']])
    wordcloud = WordCloud(width=800, height=400, background_color='white',
                          colormap='Reds').generate(genres_text)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title('Genre Word Cloud', fontsize=12)
    plt.axis('off')

    # 6. Content Addition Growth
    plt.subplot(3, 2, 6)
    additions = df.groupby('year_added').size().cumsum()
    additions.plot(color='#E50914', marker='o')
    plt.title('Cumulative Content Growth', fontsize=12)
    plt.xlabel('Year')
    plt.ylabel('Total Titles')
    plt.tight_layout()
    plt.suptitle('Netflix Content: Advanced Analytics Dashboard', y=1.02,␣
 ↪fontsize=16)
    plt.savefig('netflix_advanced_dashboard.png', dpi=300, bbox_inches='tight')
    plt.show()

create_advanced_visualizations(df)

#DEEP DIVE ANALYSIS
def perform_deep_dive_analysis(df):
    """Execute sophisticated analytical techniques"""
    print("\n DEEP DIVE ANALYSIS")

    # Genre Analysis
    mlb = MultiLabelBinarizer()
    genre_matrix = pd.DataFrame(mlb.fit_transform(df['genres']),
                                columns=mlb.classes_,
                                index=df.index)
```

```
    top_genre_combos = genre_matrix.sum().sort_values(ascending=False).head(10)

    # Content Longevity Analysis
    df['content_age'] = df['year_added'] - df['release_year']
    longevity_stats = df.groupby('type')['content_age'].describe()

    # Seasonal Analysis
    seasonal = df.groupby(['month_added', 'type']).size().unstack()

    # Print insights
    print("\n TOP GENRES:")
    print(top_genre_combos.to_string())
    print("\n CONTENT LONGEVITY (years between release and addition):")
    print(longevity_stats.to_string())
    print("\n SEASONAL ADDITION PATTERNS:")
    print(seasonal.idxmax().to_string())
    return {
        'genre_matrix': genre_matrix,
        'longevity_stats': longevity_stats,
        'seasonal_patterns': seasonal
    }
analysis_results = perform_deep_dive_analysis(df)

#TREND ANALYSIS & FORECASTING
def analyze_trends_and_forecast(df):
    """Perform time series analysis and simple forecasting"""
    print("\n TREND ANALYSIS & FORECASTING")

    # Prepare time series data
    ts = df.groupby(['year_added', 'type']).size().unstack().fillna(0)
    ts['Total'] = ts.sum(axis=1)

    # Calculate growth rates
    ts['Growth_Rate'] = ts['Total'].pct_change() * 100
    ts['3Yr_Avg_Growth'] = ts['Growth_Rate'].rolling(3).mean()

    # Simple forecasting (linear extrapolation of 3-year growth)
    last_year = int(ts.index.max())  #  FIXED
    forecast_years = 3
    forecast = pd.DataFrame(index=range(last_year, last_year + forecast_years +␣
 ↪1))
    forecast['Total'] = [ts.loc[last_year, 'Total'] *
                         (1 + ts.loc[last_year, '3Yr_Avg_Growth']/100)**i
                         for i in range(forecast_years + 1)]

    # Plot historical and forecasted data
    plt.figure(figsize=(12, 6))
```

```
    plt.plot(ts.index, ts['Total'], 'o-', label='Historical', color='#E50914')
    plt.plot(forecast.index, forecast['Total'], 's--', label='Forecast',␣
 ↪color='#221F1F')
    plt.fill_between(forecast.index, forecast['Total']*0.9, forecast['Total']*1.
 ↪1,
                     color='gray', alpha=0.1)
    plt.title('Netflix Content Growth: Historical & Forecast', fontsize=14)
    plt.xlabel('Year')
    plt.ylabel('Total Titles')
    plt.legend()
    plt.grid(True, alpha=0.3)
    plt.savefig('netflix_growth_forecast.png', dpi=300, bbox_inches='tight')
    plt.show()
    print(f"\n INSIGHT: Based on 3-year average growth of {ts.loc[last_year,␣
 ↪'3Yr_Avg_Growth']:.1f}%")
    print(forecast.to_string())
    return ts, forecast

ts_data, forecast = analyze_trends_and_forecast(df)

#SAVE RESULTS
def save_analysis_results(df, analysis_results):
    """Save all analysis outputs systematically"""
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")

    # Save cleaned data
    df.to_csv(f'netflix_cleaned_{timestamp}.csv', index=False)

    # Save analysis results
    with pd.ExcelWriter(f'netflix_analysis_{timestamp}.xlsx') as writer:
        df.describe().to_excel(writer, sheet_name='Summary Stats')
        analysis_results['genre_matrix'].sum().sort_values(ascending=False).
 ↪to_excel(
            writer, sheet_name='Genre Analysis')
        analysis_results['longevity_stats'].to_excel(
            writer, sheet_name='Content Longevity')
        ts_data.to_excel(writer, sheet_name='Time Series Data')
        forecast.to_excel(writer, sheet_name='Growth Forecast')
    print(f"\n Results saved with timestamp: {timestamp}")
save_analysis_results(df, analysis_results)
print("\n Analysis completed successfully! ")
```

```
Data loaded successfully (rows: 8807, cols: 12)

COMPREHENSIVE EXPLORATORY ANALYSIS

BASIC STATISTICS:
```

```
- Dataset spans 1925 to 2021
- Latest addition: 2021-09-25
- type
Movie      0.696151
TV Show    0.303849

  MISSING VALUES:
duration_seasons    6131
duration_min        2679
director            2634
cast                 825
year_added            98
month_added           98
date_added            98
rating                 4

  TEMPORAL PATTERNS:
- Peak release year: 2018
- Peak addition year: 2019.0

  CONTENT CHARACTERISTICS:
- Avg movie duration: 99.6 minutes
- Avg show seasons: 1.8
```



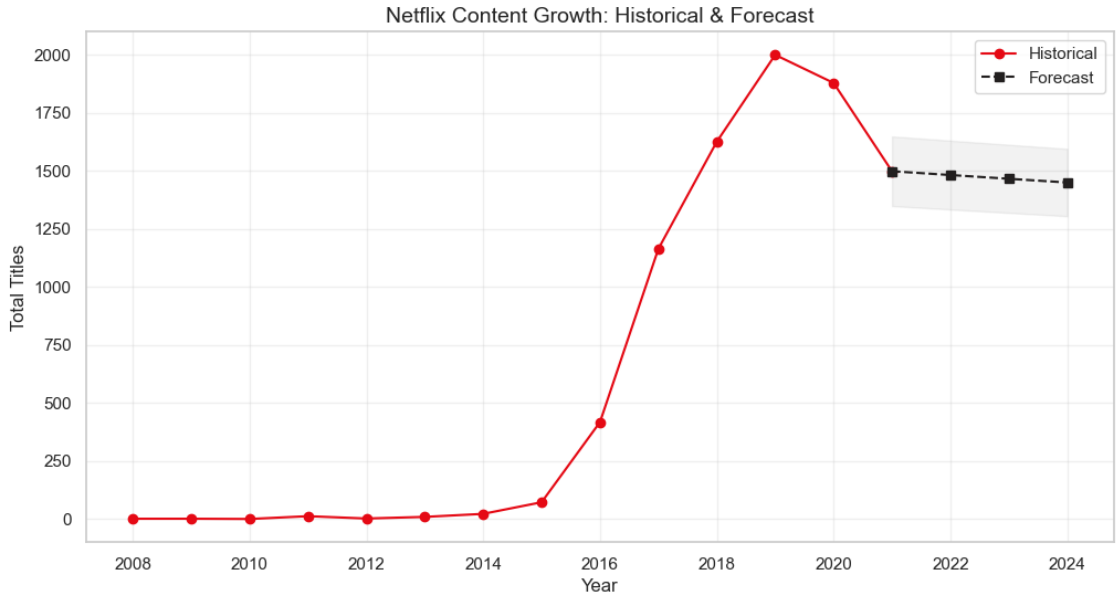
DEEP DIVE ANALYSIS

TOP GENRES:

International Movies	2752
Dramas	2427
Comedies	1674
International TV Shows	1351
Documentaries	869
Action & Adventure	859
TV Dramas	763
Independent Movies	756
Children & Family Movies	641
Romantic Movies	616

CONTENT LONGEVITY (years between release and addition):

```
          count      mean       std  min  25%  50%  75%   max
type
Movie    6131.0  5.727777  9.742631 -1.0  0.0  2.0  7.0  75.0
TV Show  2578.0  2.224981  5.175744 -3.0  0.0  0.0  2.0  93.0

 SEASONAL ADDITION PATTERNS:
type
Movie      July
TV Show    July

 TREND ANALYSIS & FORECASTING
```




```
 INSIGHT: Based on 3-year average growth of -1.1%
            Total
2021  1498.000000
2022  1481.662144
2023  1465.502477
2024  1449.519054

 Results saved with timestamp: 20250509_232136

 Analysis completed successfully!
```

[ ]: