```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import pandas as pd
from google.colab import files
uploaded = files.upload()
df = pd.read_csv("/content/2025-06-02_country_weekly.csv", sep="\t")
display(df.head())
```

Choose Files 3 files

2025-06-02_country_weekly.csv(text/csv) - 24420042 bytes, last modified: 9/5/2025 - 100% done
2025-06-02_global_alltime.csv(text/csv) - 2945 bytes, last modified: 9/5/2025 - 100% done
2025-06-02_global_weekly.csv(text/csv) - 1119844 bytes, last modified: 9/5/2025 - 100% done
Saving 2025-06-02_country_weekly.csv to 2025-06-02_country_weekly.csv
Saving 2025-06-02_global_alltime.csv to 2025-06-02_global_alltime.csv
Saving 2025-06-02_global_weekly.csv to 2025-06-02_global_weekly.csv

	country_name	country_iso2	week	category	weekly_rank	show_title	season_title	cumulative_weeks_in_top_10
0	Argentina	AR	2025-05-25	Films	1	Puss in Boots: The Last Wish	NaN	1
1	Argentina	AR	2025-05-25	Films	2	Nonnas	NaN	3
2	Argentina	AR	2025-05-25	Films	3	Fear Street: Prom Queen	NaN	1

```
df.to_csv("2025-06-02_country_weekly_output.csv", index=False)
```

```
df.head()
```

	country_name	country_iso2	week	category	weekly_rank	show_title	season_title	cumulative_weeks_in_top_10
0	Argentina	AR	2025-05-25	Films	1	Puss in Boots: The Last Wish	NaN	1
1	Argentina	AR	2025-05-25	Films	2	Nonnas	NaN	3
2	Argentina	AR	2025-05-25	Films	3	Fear Street: Prom Queen	NaN	1

```
df.tail()
```

	country_name	country_iso2	week	category	weekly_rank	show_title	season_title	cumulative_weeks_in_top_10
380135	Vietnam	VN	2021-07-04	TV	6	Reply 1988	Reply 1988: Season 1	1
380136	Vietnam	VN	2021-07-04	TV	7	Nevertheless,	Nevertheless,: Limited Series	1
380137	Vietnam	VN	2021-07-04	TV	8	Too Hot to Handle	Too Hot to Handle: Season 2	1

```
df.sample(5)
```

	country_name	country_iso2	week	category	weekly_rank	show_title	season_title	cumulative_weeks_in_top_10
293264	Saudi Arabia	SA	2022-08-21	Films	5	The Next 365 Days	NaN	1
256778	Panama	PA	2021-10-03	TV	9	Baki Hanma	Baki Hanma: Season 1	1
261113	Paraguay	PY	2021-07-04	TV	4	Pablo Escobar, el patrón del mal	Pablo Escobar, el patrón del mal: Season 1	1
27800	Belgium	BE	2022-03-20	Films	1	The Adam Project	NaN	2
28558	Belgium	BE	2021-	TV	9	Elite	Elite: Season 1	1

```
df.shape
```

```
(380140, 8)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 380140 entries, 0 to 380139
Data columns (total 8 columns):
 #   Column                     Non-Null Count   Dtype 
---  ------                     --------------   ----- 
 0   country_name               380140 non-null  object
 1   country_iso2               380140 non-null  object
 2   week                       380140 non-null  object
 3   category                   380140 non-null  object
 4   weekly_rank                380140 non-null  int64 
 5   show_title                 380140 non-null  object
 6   season_title               185450 non-null  object
 7   cumulative_weeks_in_top_10  380140 non-null  int64 
dtypes: int64(2), object(6)
memory usage: 23.2+ MB
```

```
# Check number of missing values per column
df.isnull().sum()
```

	0
country_name	0
country_iso2	0
week	0
category	0
weekly_rank	0
show_title	0
season_title	194690
cumulative_weeks_in_top_10	0

dtype: int64

```
# Check percentage of missing values
df.isnull().mean() * 100
```

	0
country_name	0.000000
country_iso2	0.000000
week	0.000000
category	0.000000
weekly_rank	0.000000
show_title	0.000000
season_title	51.215342
cumulative_weeks_in_top_10	0.000000

dtype: float64

```
# Count duplicate rows
df.duplicated().sum()
```

```
np.int64(0)
```

```
df.columns
```

```
Index(['country_name', 'country_iso2', 'week', 'category', 'weekly_rank',
       'show_title', 'season_title', 'cumulative_weeks_in_top_10'],
      dtype='object')
```

```
df.dtypes
```

	0
country_name	object
country_iso2	object
week	object
category	object
weekly_rank	int64
show_title	object
season_title	object
cumulative_weeks_in_top_10	int64

dtype: object

```
df.nunique()
```

	0
country_name	94
country_iso2	94
week	204
category	2
weekly_rank	10
show_title	9430
season_title	3423
cumulative_weeks_in_top_10	127

dtype: int64

```
df["country_name"]
```

	country_name
0	Argentina
1	Argentina
2	Argentina
3	Argentina
4	Argentina
...	...
380135	Vietnam
380136	Vietnam
380137	Vietnam
380138	Vietnam
380139	Vietnam

380140 rows × 1 columns

dtype: object

```
df.iloc[45]
```

	45
country_name	Argentina
country_iso2	AR
week	2025-05-11
category	Films
weekly_rank	6
show_title	Karol G: Tomorrow was Beautiful
season_title	NaN
cumulative_weeks_in_top_10	1

dtype: object

```
df.iloc[:,0]
```

	country_name
0	Argentina
1	Argentina
2	Argentina
3	Argentina
4	Argentina
...	...
380135	Vietnam
380136	Vietnam
380137	Vietnam
380138	Vietnam
380139	Vietnam

380140 rows × 1 columns

dtype: object

```
df.iloc[0:3,0:5]
```

	country_name	country_iso2	week	category	weekly_rank
0	Argentina	AR	2025-05-25	Films	1
1	Argentina	AR	2025-05-25	Films	2
2	Argentina	AR	2025-05-25	Films	3

```
df.isnull().sum()
```

	0
country_name	0
country_iso2	0
week	0
category	0
weekly_rank	0
show_title	0
season_title	194690
cumulative_weeks_in_top_10	0

dtype: int64

```
df.loc[df["country_name"] == "India"]
```

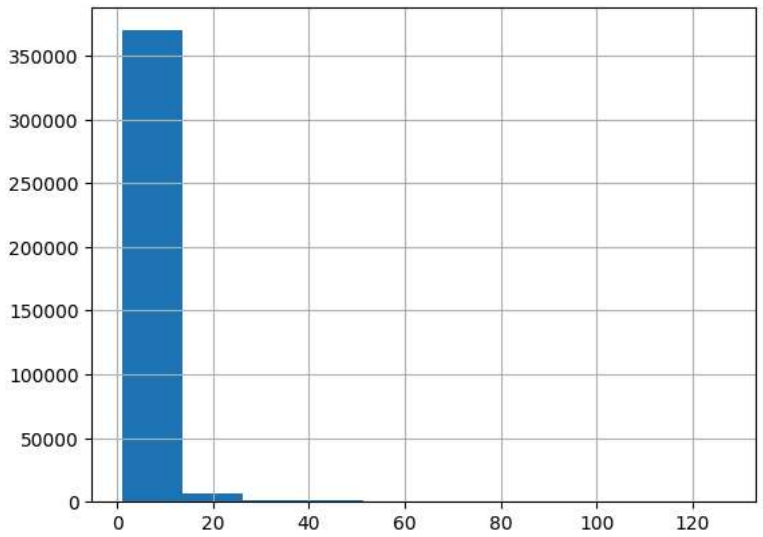
	country_name	country_iso2	week	category	weekly_rank	show_title	season_title	cumulative_weeks_in_top_10
134640	India	IN	2025-05-25	Films	1	The Diplomat	NaN	3
134641	India	IN	2025-05-25	Films	2	Jack	NaN	3
134642	India	IN	2025-05-25	Films	3	Sikandar	NaN	1
134643	India	IN	2025-05-25	Films	4	Good Bad Ugly	NaN	3
134644	India	IN	2025-05-25	Films	5	Mission: Impossible - Dead Reckoning	NaN	5
...	...	...	...	...	...	...	...	...
138715	India	IN	2021-07-04	TV	6	Lucifer	Lucifer: Season 3	1
138716	India	IN	2021-	TV	7	Lucifer	Lucifer: Season	1

```
df.loc[df["weekly_rank"]<3]
```

	country_name	country_iso2	week	category	weekly_rank	show_title	season_title	cumulative_weeks_in_top_10
0	Argentina	AR	2025-05-25	Films	1	Puss in Boots: The Last Wish	NaN	1
1	Argentina	AR	2025-05-25	Films	2	Nonnas	NaN	3
10	Argentina	AR	2025-05-25	TV	1	Secrets We Keep	Secrets We Keep: Limited Series	2

```
df["cumulative_weeks_in_top_10"].hist()
```

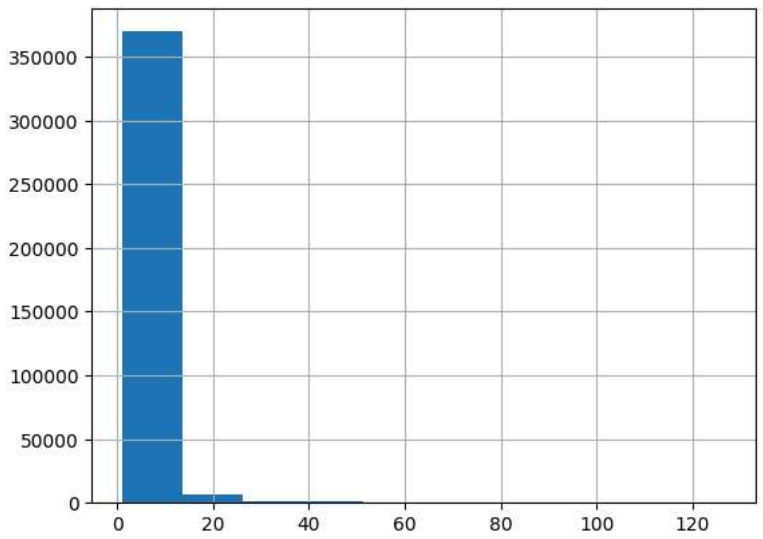
```
<Axes: >
```



	nnas	NaN	2
	...	...	...
	spital aylist	Hospital Playlist: Season 2	2
	Eyes	NaN	1
	Wife	NaN	1
	Mine	Mine: Limited Series	1
	spital aylist	Hospital Playlist: Season 1	1

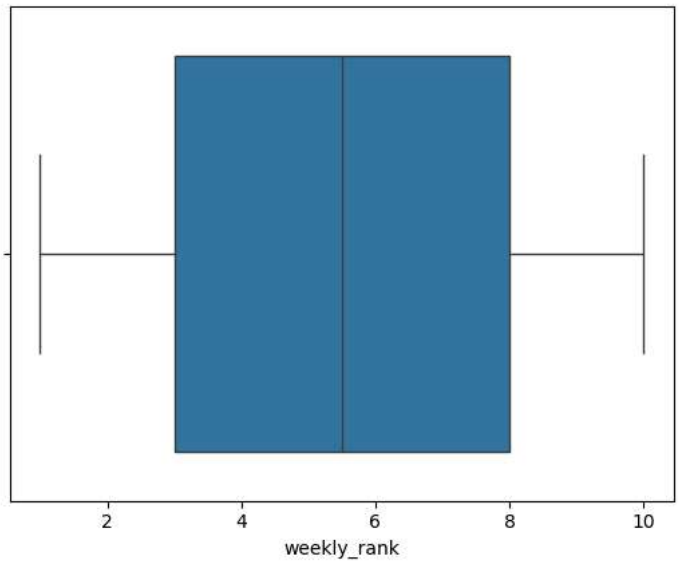
```
df["cumulative_weeks_in_top_10"].hist()
```

```
<Axes: >
```



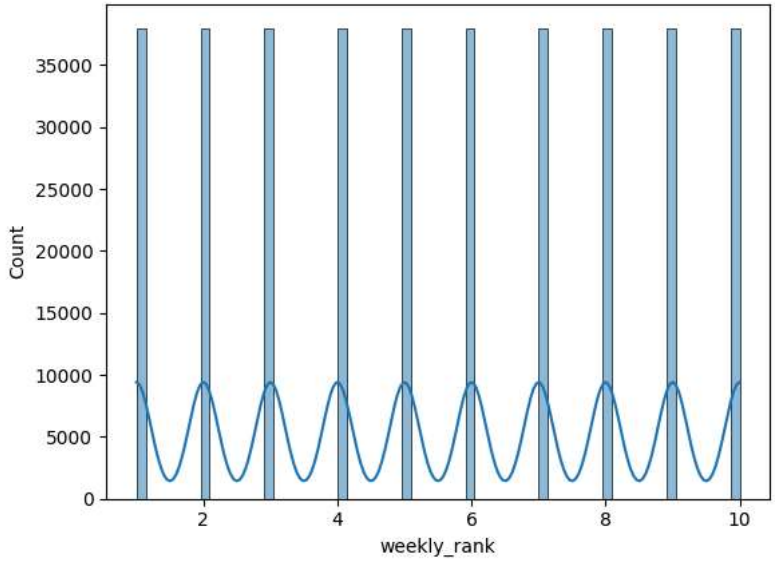

```
sns.boxplot(x=df["weekly_rank"]) # Seaborn Boxplot
```

```
<Axes: xlabel='weekly_rank'>
```



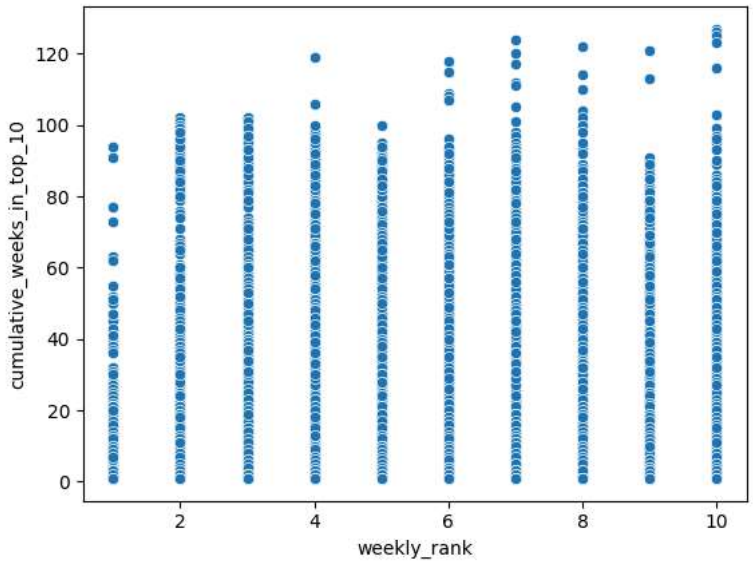

```
sns.histplot(df["weekly_rank"], kde=True) # Histogram with KDE
```

```
<Axes: xlabel='weekly_rank', ylabel='Count'>
```




```
sns.scatterplot(x=df["weekly_rank"], y=df["cumulative_weeks_in_top_10"]) # Scatter plot
```

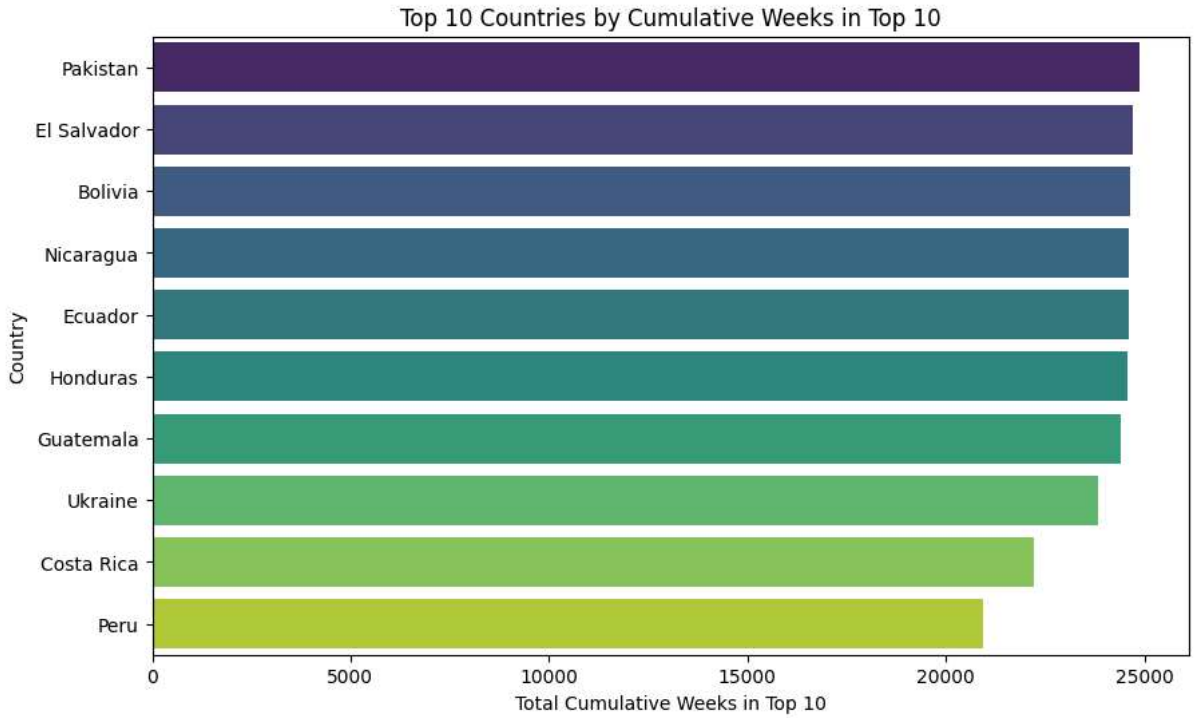
```
<Axes: xlabel='weekly_rank', ylabel='cumulative_weeks_in_top_10'>
```




```
plt.show() # Display plot
```

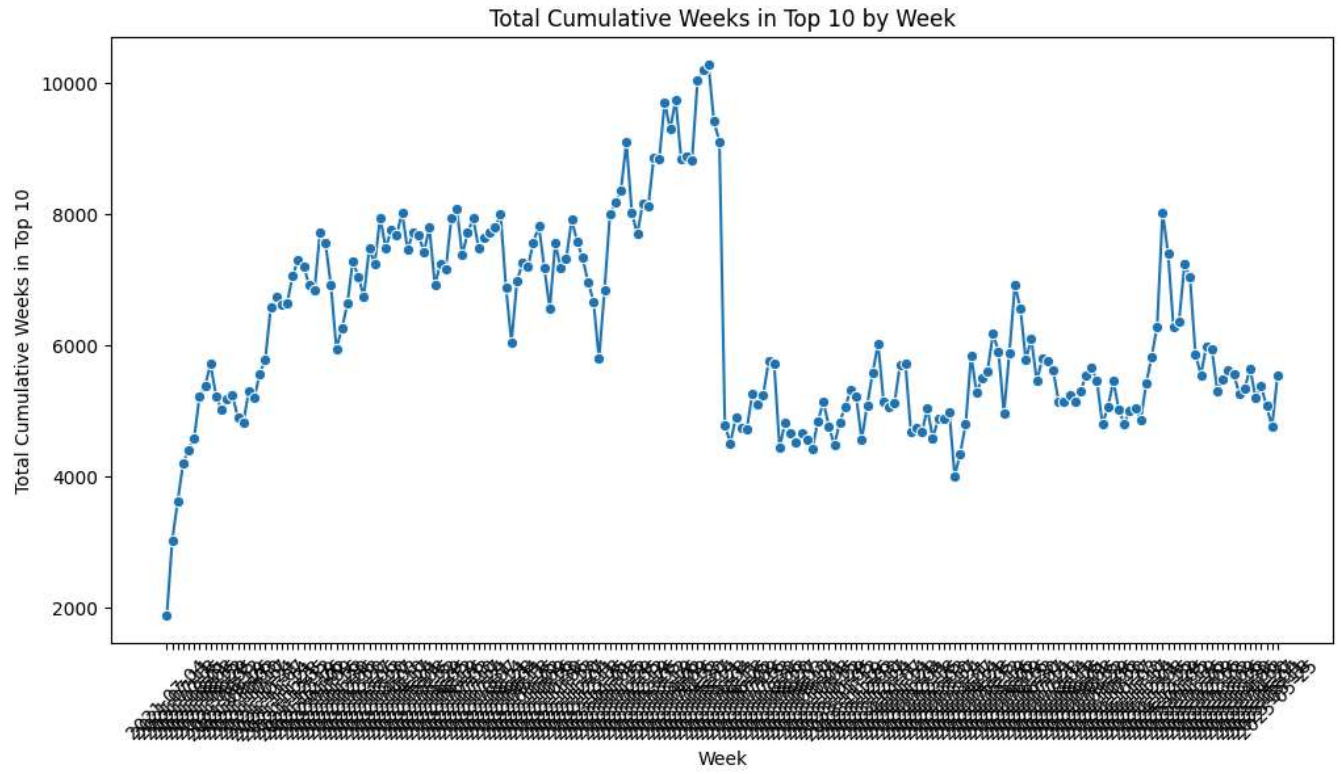
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,6))
top_countries = df.groupby('country_name')['cumulative_weeks_in_top_10'].sum().nlargest(10)
sns.barplot(x=top_countries.values, y=top_countries.index, palette="viridis")
plt.title("Top 10 Countries by Cumulative Weeks in Top 10")
plt.xlabel("Total Cumulative Weeks in Top 10")
plt.ylabel("Country")
plt.show()
```

```
/tmp/ipython-input-2469140324.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `l

  sns.barplot(x=top_countries.values, y=top_countries.index, palette="viridis")
```

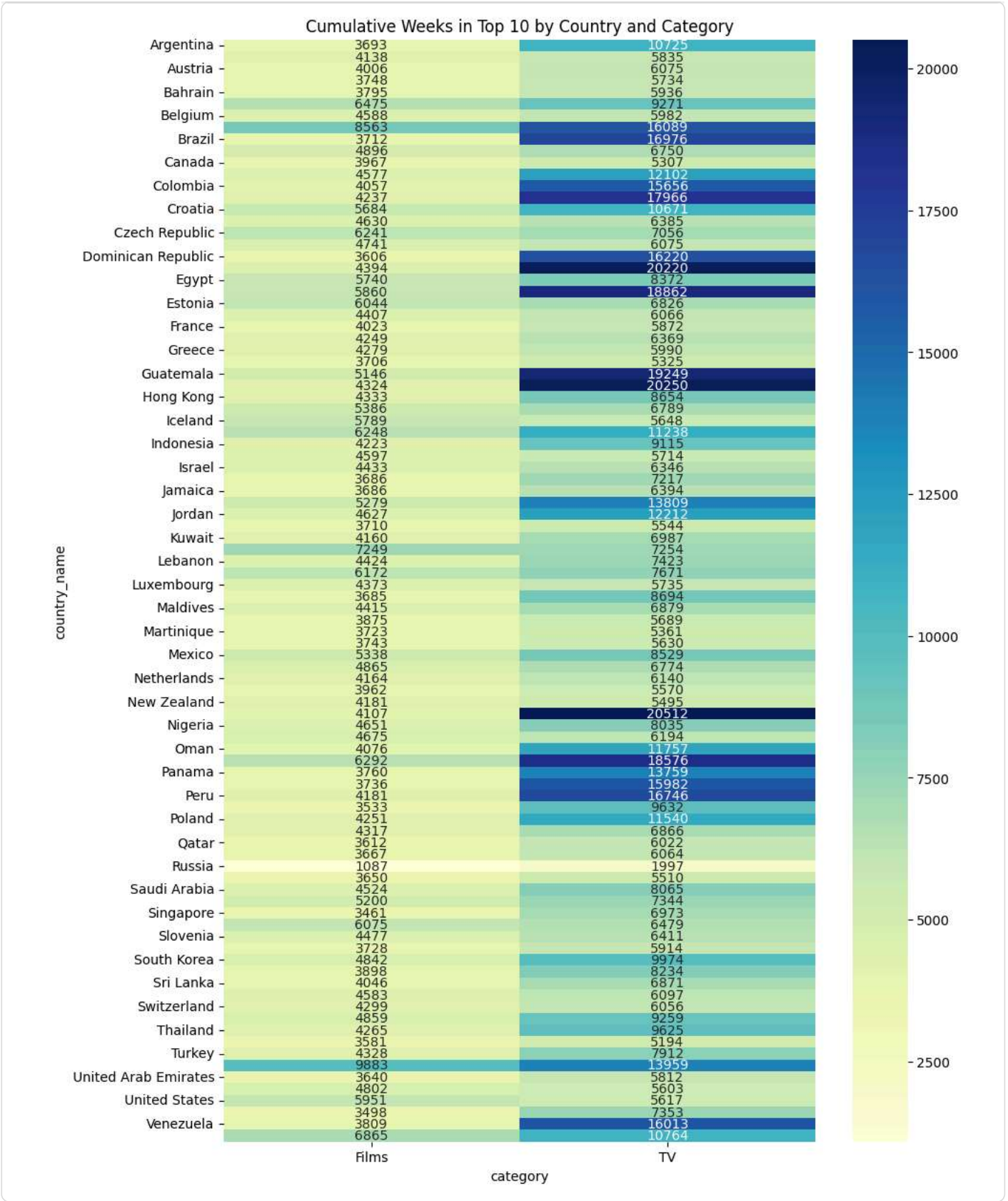



```
plt.figure(figsize=(12,6))
weekly_trend = df.groupby('week')['cumulative_weeks_in_top_10'].sum().reset_index()
```

```
sns.lineplot(data=weekly_trend, x='week', y='cumulative_weeks_in_top_10', marker='o')
plt.title("Total Cumulative Weeks in Top 10 by Week")
plt.xlabel("Week")
plt.ylabel("Total Cumulative Weeks in Top 10")
plt.xticks(rotation=45)
plt.show()
```

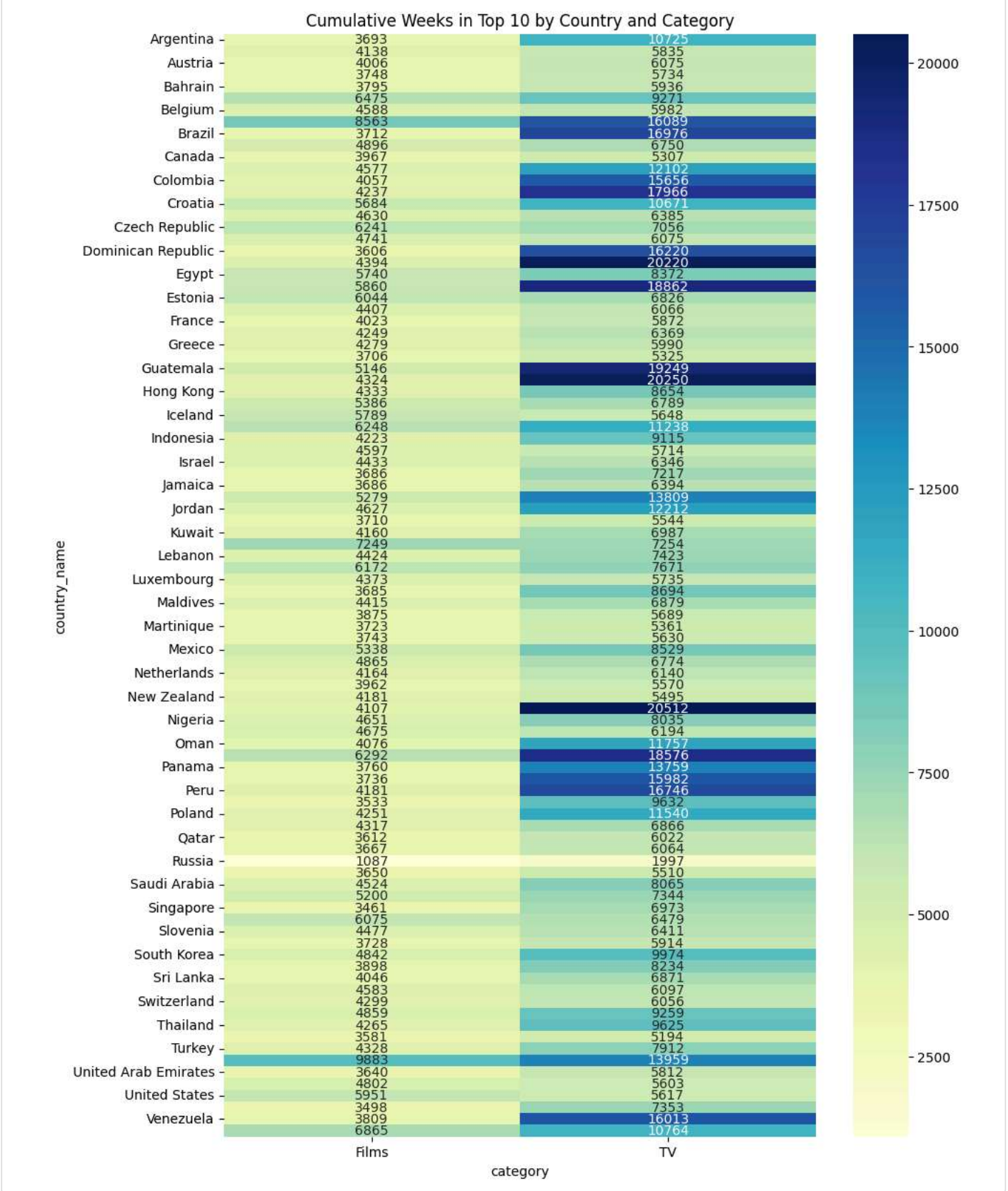



```
pivot = df.pivot_table(
    values='cumulative_weeks_in_top_10',
    index='country_name',
    columns='category',
    aggfunc='sum',
    fill_value=0
)

plt.figure(figsize=(10,15)) # Increased figure size for better readability with many countries
sns.heatmap(pivot, annot=True, fmt=".0f", cmap="YlGnBu")
plt.title("Cumulative Weeks in Top 10 by Country and Category")
plt.show()
```




```
pivot = df.pivot_table(
    values='cumulative_weeks_in_top_10',
    index='country_name',
    columns='category',
    aggfunc='sum',
    fill_value=0
)

plt.figure(figsize=(10,15)) # Increased figure size for better readability with many countries
sns.heatmap(pivot, annot=True, fmt=".0f", cmap="YlGnBu")
```

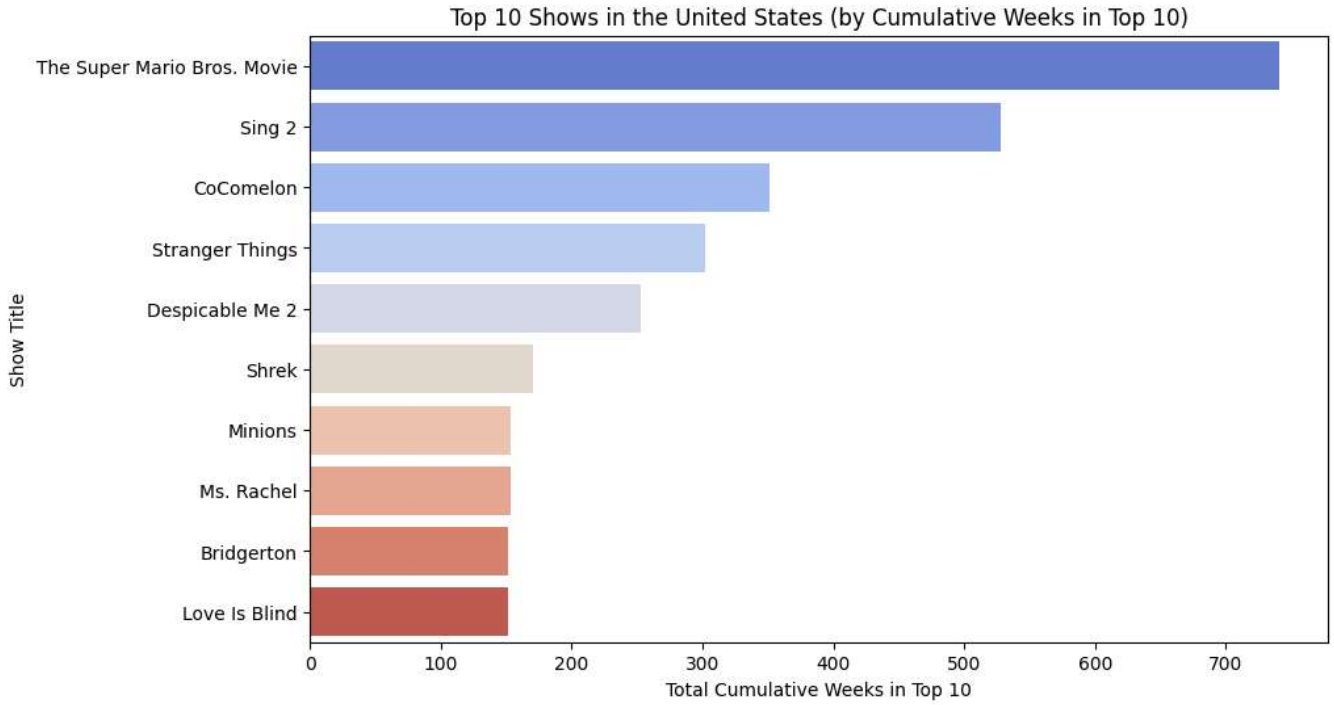
```
plt.title("Cumulative Weeks in Top 10 by Country and Category")
plt.show()
```



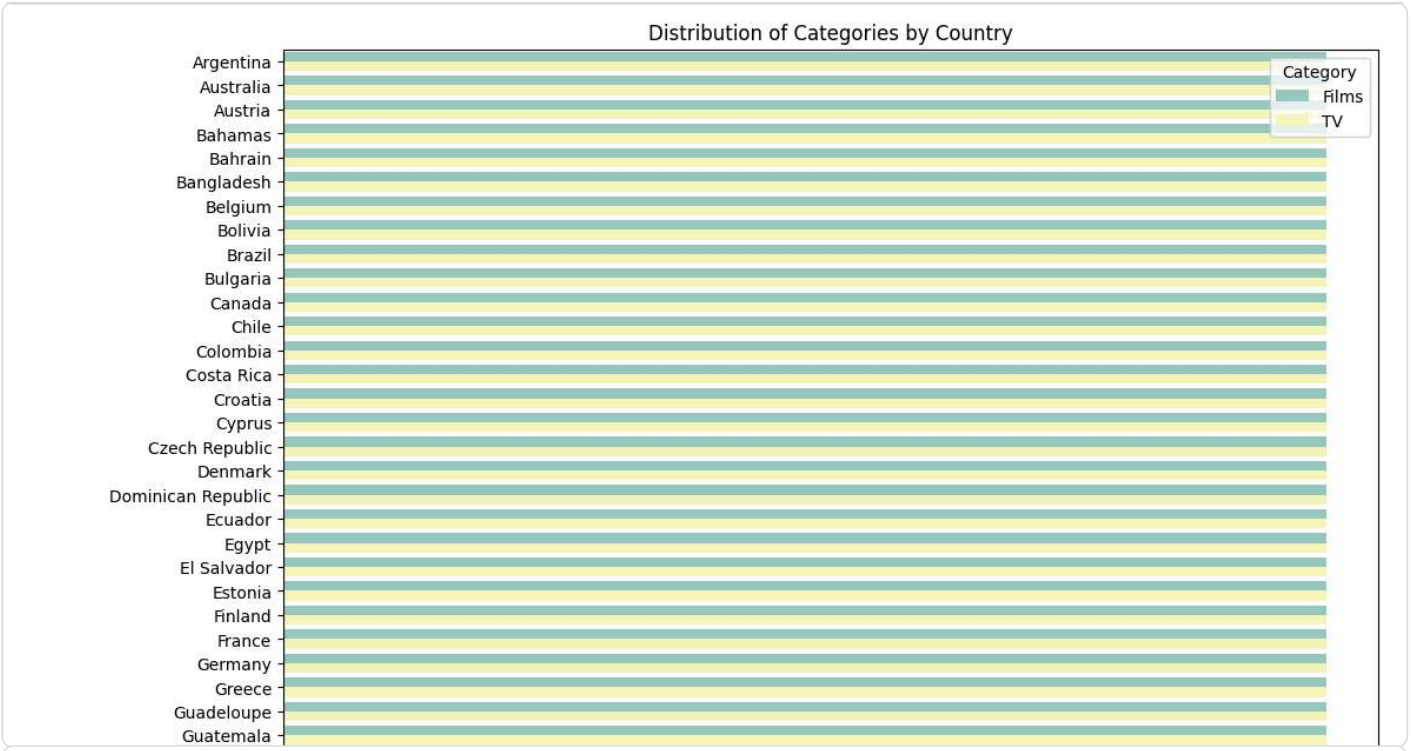

```
top_shows = (df[df['country_name'] == "United States"]
             .groupby('show_title')['cumulative_weeks_in_top_10']
             .sum()
             .nlargest(10))

plt.figure(figsize=(10,6))
sns.barplot(x=top_shows.values, y=top_shows.index, palette="coolwarm")
```

```
plt.title("Top 10 Shows in the United States (by Cumulative Weeks in Top 10)")
plt.xlabel("Total Cumulative Weeks in Top 10")
plt.ylabel("Show Title")
plt.show()
```

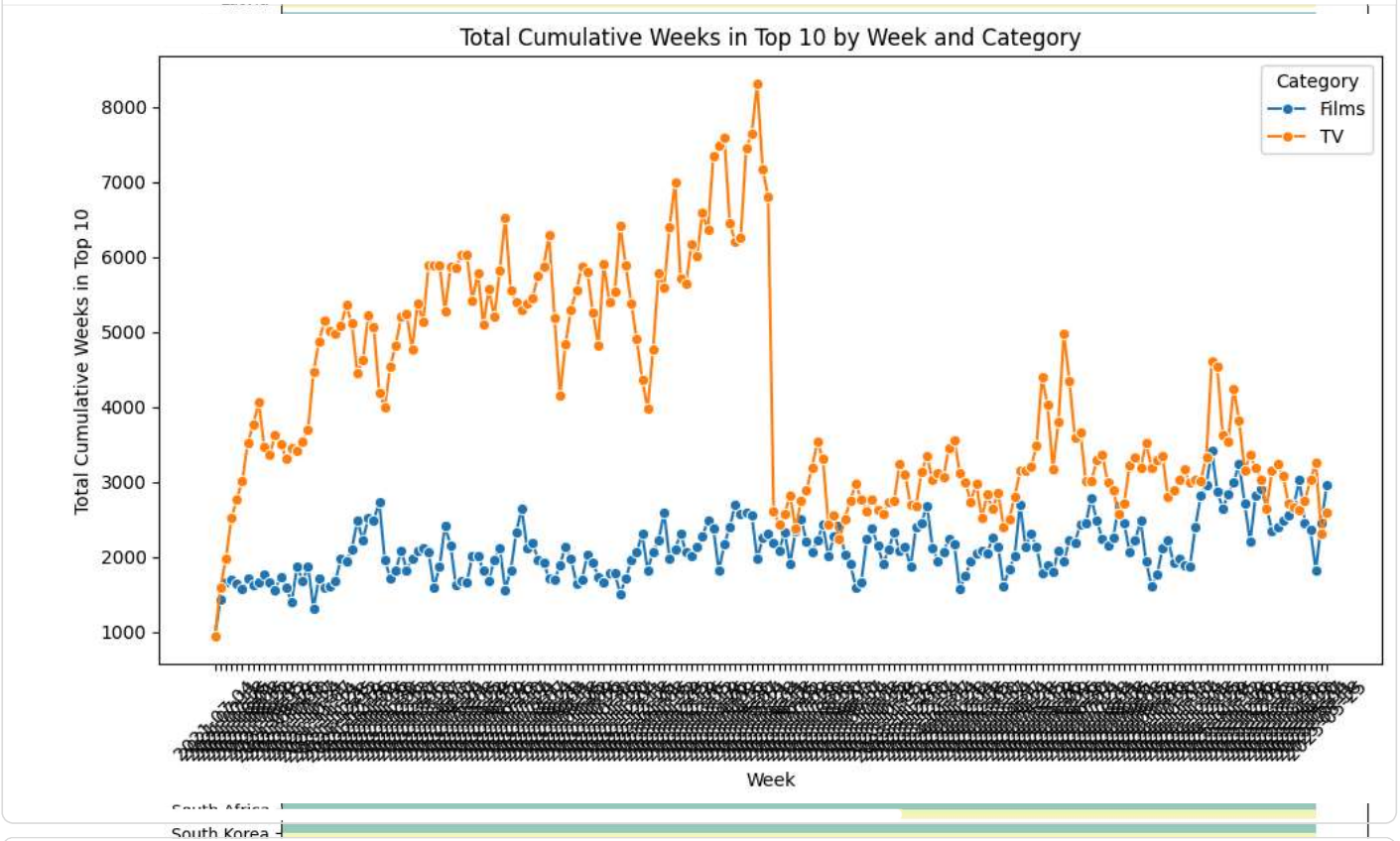
```
/tmp/ipython-input-596161115.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `l

  sns.barplot(x=top_shows.values, y=top_shows.index, palette="coolwarm")
```



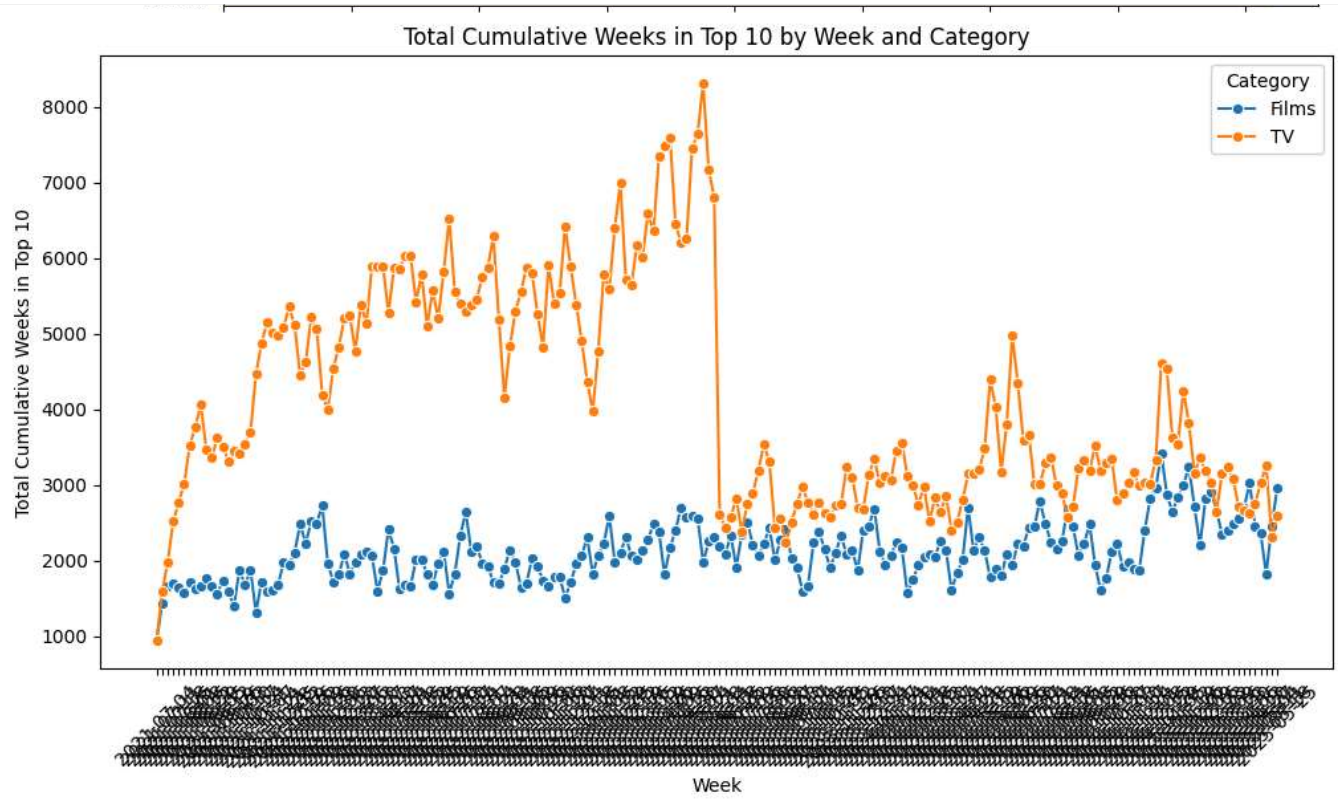

```
plt.figure(figsize=(12,25)) # Increased figure height for better readability with many countries
sns.countplot(data=df, y='country_name', hue='category', palette="Set3")
plt.title("Distribution of Categories by Country")
plt.xlabel("Count of Appearances in Top 10")
plt.ylabel("Country")
plt.legend(title="Category")
plt.show()
```




```
category_trend = (df.groupby(['week','category'])
                  ['cumulative_weeks_in_top_10']
                  .sum()
                  .reset_index())

plt.figure(figsize=(12,6))
sns.lineplot(data=category_trend, x='week', y='cumulative_weeks_in_top_10', hue='category', marker='o')
plt.title("Total Cumulative Weeks in Top 10 by Week and Category")
plt.xlabel("Week")
plt.ylabel("Total Cumulative Weeks in Top 10")
plt.xticks(rotation=45)
plt.legend(title="Category")
plt.show()
```

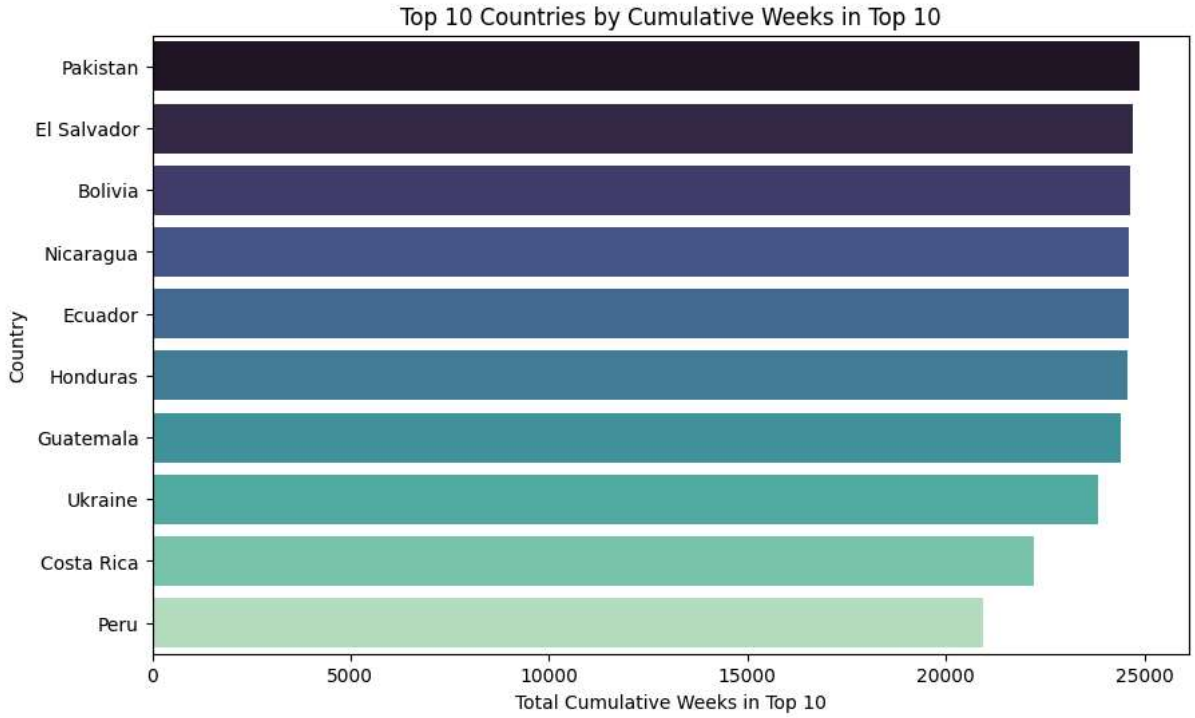


```
category_trend = (df.groupby(['week','category'])
                  ['cumulative_weeks_in_top_10']
                  .sum()
                  .reset_index())

plt.figure(figsize=(12,6))
sns.lineplot(data=category_trend, x='week', y='cumulative_weeks_in_top_10', hue='category', marker='o')
plt.title("Total Cumulative Weeks in Top 10 by Week and Category")
plt.xlabel("Week")
plt.ylabel("Total Cumulative Weeks in Top 10")
plt.xticks(rotation=45)
plt.legend(title="Category")
plt.show()
```




```
top_countries = df.groupby('country_name')['cumulative_weeks_in_top_10'].sum().nlargest(10).reset_index()

plt.figure(figsize=(10,6))
sns.barplot(data=top_countries, x='cumulative_weeks_in_top_10', y='country_name', palette="mako")
plt.title("Top 10 Countries by Cumulative Weeks in Top 10")
plt.xlabel("Total Cumulative Weeks in Top 10")
plt.ylabel("Country")
plt.show()
```

```
/tmp/ipython-input-1507523507.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `l

  sns.barplot(data=top_countries, x='cumulative_weeks_in_top_10', y='country_name', palette="mako")
```




```
# Count missing values
print("Missing values before handling:")
print(df.isnull().sum())

# Categorical columns → fill with custom placeholder for 'season_title'
df['season_title'] = df['season_title'].fillna('Unknown Season')

# Verify missing values are handled
print("\nMissing values after handling:")
print(df.isnull().sum())
```

```
Missing values before handling:
country_name                    0
country_iso2                    0
week                            0
category                        0
weekly_rank                     0
show_title                      0
season_title               194690
cumulative_weeks_in_top_10      0
dtype: int64

Missing values after handling:
country_name               0
country_iso2               0
week                       0
category                   0
weekly_rank                0
show_title                 0
season_title               0
cumulative_weeks_in_top_10 0
dtype: int64
```

```
# Count duplicate rows
print("Duplicates:", df.duplicated().sum())

# Remove duplicates
df.drop_duplicates(inplace=True)
```

```
Duplicates: 0
```

```
def remove_outliers_iqr(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    return data[(data[column] >= lower) & (data[column] <= upper)]

# Example for cumulative_weeks_in_top_10
df = remove_outliers_iqr(df, 'cumulative_weeks_in_top_10')
display(df.head())
```

	country_name	country_iso2	week	category	weekly_rank	show_title	season_title	cumulative_weeks_in_top_10
0	Argentina	AR	2025-05-25	Films	1	Puss in Boots: The Last Wish	Unknown Season	1
1	Argentina	AR	2025-05-25	Films	2	Nonnas	Unknown Season	3
2	Argentina	AR	2025-05-25	Films	3	Fear Street: Prom Queen	Unknown Season	1

```
# Standardize category names
df['category'] = df['category'].str.strip().str.title()

# Create numeric transformations if needed
# The following line caused a KeyError because 'views_first_91_days' column does not exist.
# df['views_in_millions'] = df['views_first_91_days'] / 1_000_000
```

```
df.to_csv("cleaned_country_weekly.csv", index=False)
```

```
# Convert categorical columns to numeric
categorical_cols = df.select_dtypes(include=['object']).columns
df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
```

```
# Define Features and Target
# Replace 'target_column' with your actual target column
target_column = "cumulative_weeks_in_top_10" # Replace with your actual target column
X = df.drop(target_column, axis=1)
y = df[target_column]
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.metrics import (
    mean_squared_error, mean_absolute_error, r2_score,
    accuracy_score, classification_report, confusion_matrix, roc_auc_score
)
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv("/content/2025-06-02_country_weekly.csv", sep="\t")
df = pd.read_csv("/content/2025-06-02_global_alltime.csv", sep="\t")
df = pd.read_csv("/content/2025-06-02_global_weekly.csv", sep="\t")

print("Dataset Shape:", df.shape)
print(df.head())
print(df.info())
print(df.describe())
```

```
1                         4               False
2                         3               False
3                         1               False
4                         2               False

                      runtime_override_flag episode_launch_dtls
0  [{is_staggered_launch=0, is_live_title=0}]                 NaN
1  [{is_staggered_launch=0, is_live_title=0}]                 NaN
2  [{is_staggered_launch=0, is_live_title=0}]                 NaN
3  [{is_staggered_launch=0, is_live_title=0}]                 NaN
4  [{is_staggered_launch=0, is_live_title=0}]                 NaN
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8160 entries, 0 to 8159
Data columns (total 12 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   week                        8160 non-null   object
 1   category                    8160 non-null   object
 2   weekly_rank                 8160 non-null   int64
 3   show_title                  8160 non-null   object
 4   season_title                3948 non-null   object
 5   weekly_hours_viewed         8160 non-null   int64
 6   runtime                     4080 non-null   float64
 7   weekly_views                4080 non-null   float64
 8   cumulative_weeks_in_top_10  8160 non-null   int64
 9   is_staggered_launch         8160 non-null   bool
 10  runtime_override_flag       4080 non-null   object
 11  episode_launch_dtls         133 non-null    object
dtypes: bool(1), float64(2), int64(3), object(6)
memory usage: 709.3+ KB
None
       weekly_rank  weekly_hours_viewed      runtime  weekly_views  \
count  8160.000000         8.160000e+03  4080.000000  4.080000e+03
mean      5.500000         1.786910e+07     3.631789  4.716054e+06
std       2.872457         2.550419e+07     3.111281  5.349054e+06
min       1.000000         7.000000e+05     0.000000  6.000000e+05
25%       3.000000         6.000000e+06     1.683300  1.900000e+06
50%       5.500000         1.091000e+07     2.191650  3.100000e+06
75%       8.000000         2.010000e+07     5.150000  5.300000e+06
max      10.000000         5.717600e+08    47.583300  6.800000e+07

       cumulative_weeks_in_top_10
count                 8160.000000
mean                     3.104412
std                      3.245464
min                      1.000000
25%                      1.000000
50%                      2.000000
75%                      4.000000
max                     30.000000
```

```
# Handle missing values

# Fill missing values in numeric columns with the mean of each column
numeric_cols = df.select_dtypes(include=np.number).columns
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())

# Fill missing values in non-numeric columns (like 'season_title') with a placeholder
# You can adjust this strategy based on the specific column and your needs
if 'season_title' in df.columns:
    df['season_title'] = df['season_title'].fillna('Unknown Season')

# Verify missing values are handled
print("Missing values after handling:")
print(df.isnull().sum())
```

```
Missing values after handling:
week                              0
category                          0
weekly_rank                       0
show_title                        0
season_title                      0
weekly_hours_viewed               0
runtime                           0
weekly_views                      0
cumulative_weeks_in_top_10        0
is_staggered_launch               0
runtime_override_flag          4080
episode_launch_dtls            8027
dtype: int64
```

```
# Identify categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns
```

```
# Encode categorical variables
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
```

```
# Replace 'target_column' with your column
target_column = "cumulative_weeks_in_top_10"
X = df.drop(target_column, axis=1)
y = df[target_column]
```

```
# Detect type of problem: regression or classification
if y.nunique() <= 10 and y.dtype in [np.int64, np.int32]:  # small number of classes → classification
    problem_type = "classification"
else:
    problem_type = "regression"

print(f"\nDetected problem type: {problem_type}")
```

```
Detected problem type: regression
```

```
# Train-Test Split
# ==============================
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Train & Evaluate Model

if problem_type == "regression":
    # Regression models
    model = LinearRegression()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Metrics
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print("\nRegression Model Evaluation:")
    print(f"Mean Squared Error (MSE): {mse:.2f}")
    print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
    print(f"Mean Absolute Error (MAE): {mae:.2f}")
    print(f"R-squared (R2): {r2:.2f}")

elif problem_type == "classification":
    # Classification models
    # Example: RandomForestClassifier
    model = RandomForestClassifier(random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Metrics
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    matrix = confusion_matrix(y_test, y_pred)
    # auc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1]) # For binary classification

    print("\nClassification Model Evaluation:")
    print(f"Accuracy: {accuracy:.2f}")
    print("Classification Report:")
    print(report)
    print("Confusion Matrix:")
    print(matrix)
    # print(f"AUC: {auc:.2f}")
```

```
Regression Model Evaluation:
Mean Squared Error (MSE): 8.72
Root Mean Squared Error (RMSE): 2.95
Mean Absolute Error (MAE): 1.96
R-squared (R2): 0.03
```

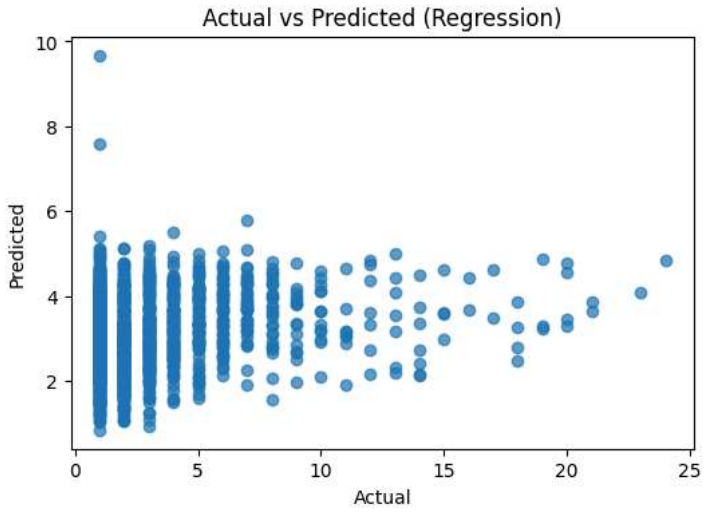
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np

# 7️⃣ Train & Evaluate Model
# ============================
if problem_type == "regression":
    # Regression models
    model = LinearRegression()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    # Metrics
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print("\nRegression Model Evaluation:")
    print(f"Mean Squared Error (MSE): {mse:.2f}")
    print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
    print(f"Mean Absolute Error (MAE): {mae:.2f}")
    print(f"R-squared (R2): {r2:.2f}")
```

```
Regression Model Evaluation:
Mean Squared Error (MSE): 8.72
Root Mean Squared Error (RMSE): 2.95
Mean Absolute Error (MAE): 1.96
R-squared (R2): 0.03
```

```
# 7️⃣ Train & Evaluate Model
# ============================
if problem_type == "regression":
    # Regression models
    model = LinearRegression()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Metrics
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print("\n--- Regression Metrics ---")
    print(f"MSE: {mse}")
    print(f"RMSE: {rmse}")
    print(f"MAE: {mae}")
    print(f"R2: {r2}")

    # Plot Actual vs Predicted
    plt.figure(figsize=(6,4))
    plt.scatter(y_test, y_pred, alpha=0.7)
    plt.xlabel("Actual")
    plt.ylabel("Predicted")
    plt.title("Actual vs Predicted (Regression)")
    plt.show()

else:
    # Classification models
    model = LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:,1] if len(y.unique()) == 2 else None

    # Metrics
    print("\n--- Classification Metrics ---")
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
    print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

    # ROC-AUC for binary classification
    if y_prob is not None:
        roc_auc = roc_auc_score(y_test, y_prob)
        print(f"\nROC-AUC Score: {roc_auc}")

    # Confusion matrix heatmap
    sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d", cmap="Blues")
    plt.title("Confusion Matrix")
    plt.show()
```

```
--- Regression Metrics ---
MSE: 8.720254893868914
RMSE: 2.953007770709199
MAE: 1.9624193864984256
R2: 0.025842880485723474
```




```
print(df.head())         # see transformed columns
print(df.dtypes)         # all should be numeric (int or float)
```

```
   week  category  weekly_rank  show_title  season_title  weekly_hours_viewed  \
0   203         0            1         709          1115             16000000
1   203         0            2        1606          1115             14800000
2   203         0            3        1480          1115             16000000
3   203         0            4        2487          1115              7300000
4   203         0            5        1022          1115             10600000

   runtime  weekly_views  cumulative_weeks_in_top_10  is_staggered_launch  \
0   1.5000    10700000.0                           1                False
1   1.7000     8700000.0                           4                False
2   1.9000     8400000.0                           3                False
3   1.0667     6800000.0                           1                False
4   1.9833     5300000.0                           2                False

   runtime_override_flag  episode_launch_dtls
0                      0                  133
1                      0                  133
2                      0                  133
3                      0                  133
4                      0                  133
week                          int64
category                      int64
weekly_rank                   int64
show_title                    int64
season_title                  int64
weekly_hours_viewed           int64
runtime                     float64
weekly_views                float64
cumulative_weeks_in_top_10    int64
is_staggered_launch            bool
runtime_override_flag         int64
episode_launch_dtls           int64
dtype: object
```