Import the CSV file

```
from google.colab import files
import pandas as pd

uploaded = files.upload()
```

Choose Files no files selected Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving all-weeks-countries.csv to all-weeks-countries.csv
```

```
file_name = list(uploaded.keys())[0]
top10 = pd.read_csv(file_name)

top10 = pd.DataFrame(top10)

top10.info()
top10.describe()
```

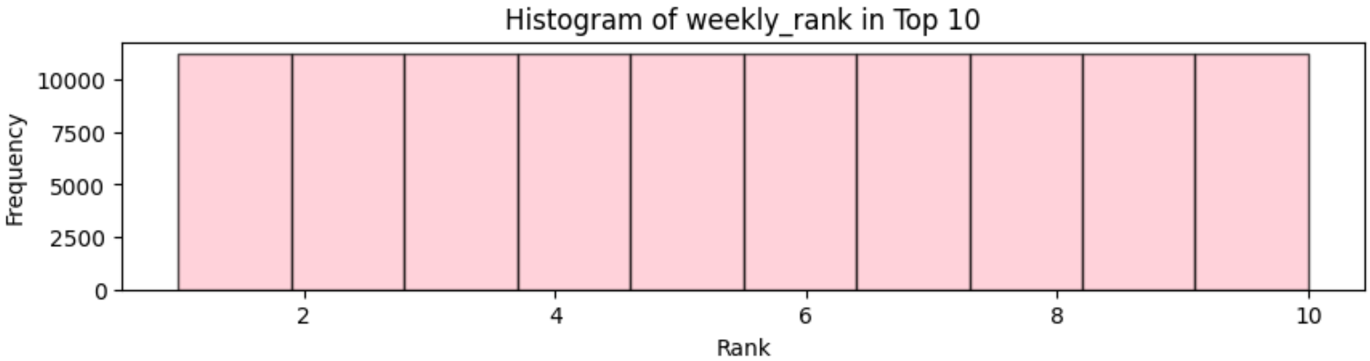
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112300 entries, 0 to 112299
Data columns (total 8 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   country_name                112300 non-null  object
 1   country_iso2                112300 non-null  object
 2   week                        112300 non-null  object
 3   category                    112300 non-null  object
 4   weekly_rank                 112300 non-null  int64 
 5   show_title                  112300 non-null  object
 6   season_title                54668 non-null   object
 7   cumulative_weeks_in_top_10  112300 non-null  int64 
dtypes: int64(2), object(6)
memory usage: 6.9+ MB
```

	weekly_rank	cumulative_weeks_in_top_10
count	112300.000000	112300.000000
mean	5.500000	3.468281
std	2.872294	5.518189
min	1.000000	1.000000
25%	3.000000	1.000000
50%	5.500000	2.000000
75%	8.000000	3.000000
max	10.000000	60.000000

Preliminary analysis

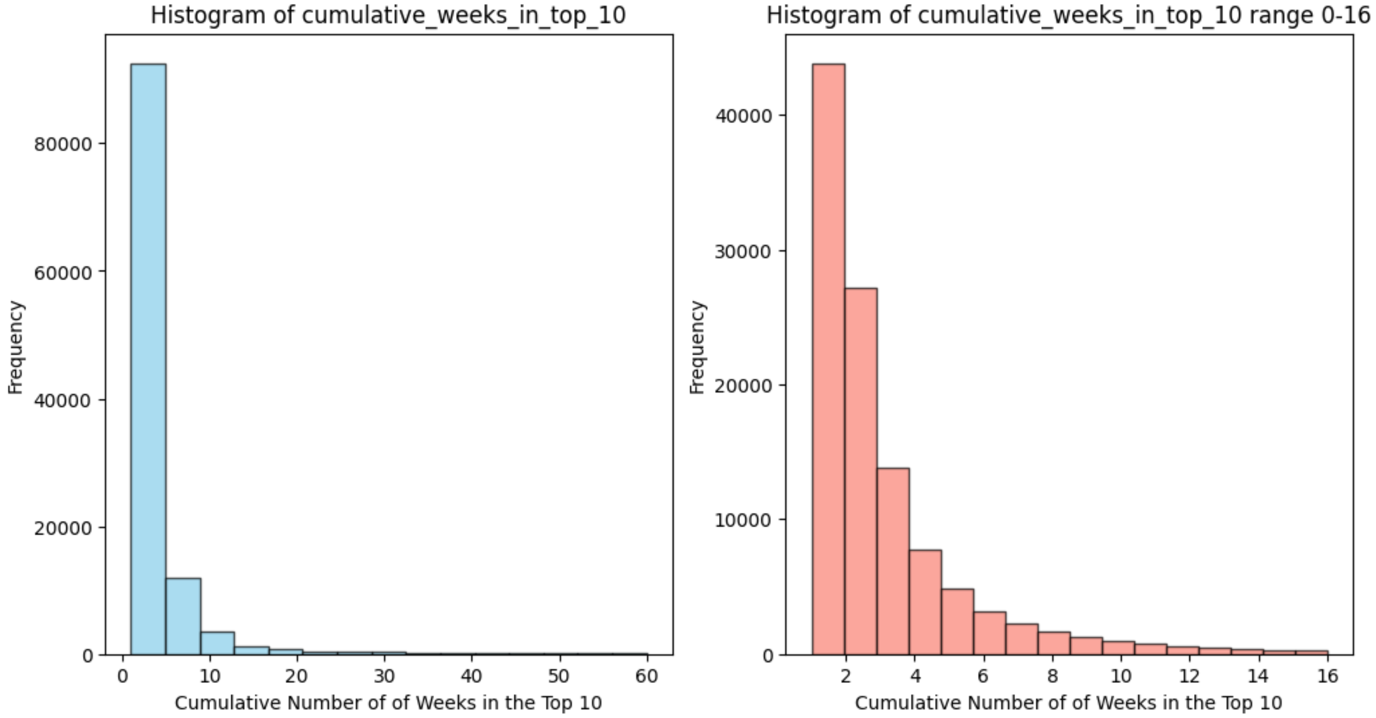
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 2))
plt.hist(top10['weekly_rank'], bins=10, edgecolor='black', alpha=0.7, color='p
plt.title('Histogram of weekly_rank in Top 10')
plt.xlabel('Rank')
plt.ylabel('Frequency')

```

```
Text(0, 0.5, 'Frequency')
```



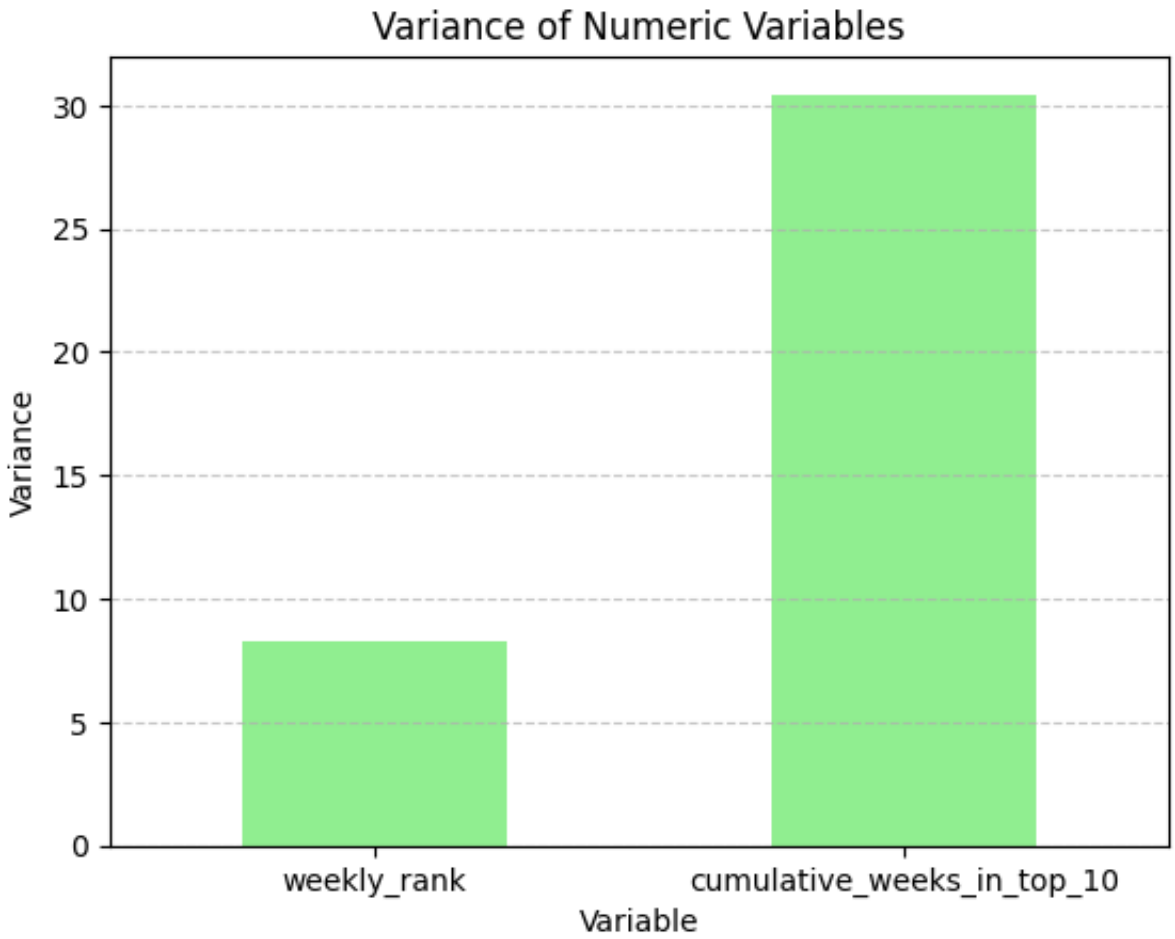
```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

axes[0].hist(top10['cumulative_weeks_in_top_10'], bins=15, edgecolor='black', 
axes[0].set_title('Histogram of cumulative_weeks_in_top_10')
axes[0].set_xlabel('Cumulative Number of of Weeks in the Top 10')
axes[0].set_ylabel('Frequency')

axes[1].hist(top10['cumulative_weeks_in_top_10'], bins=16, range=(1, 16), edge
axes[1].set_title('Histogram of cumulative_weeks_in_top_10 range 0-16')
axes[1].set_xlabel('Cumulative Number of of Weeks in the Top 10')
axes[1].set_ylabel('Frequency')

```

```
Text(0, 0.5, 'Frequency')
```

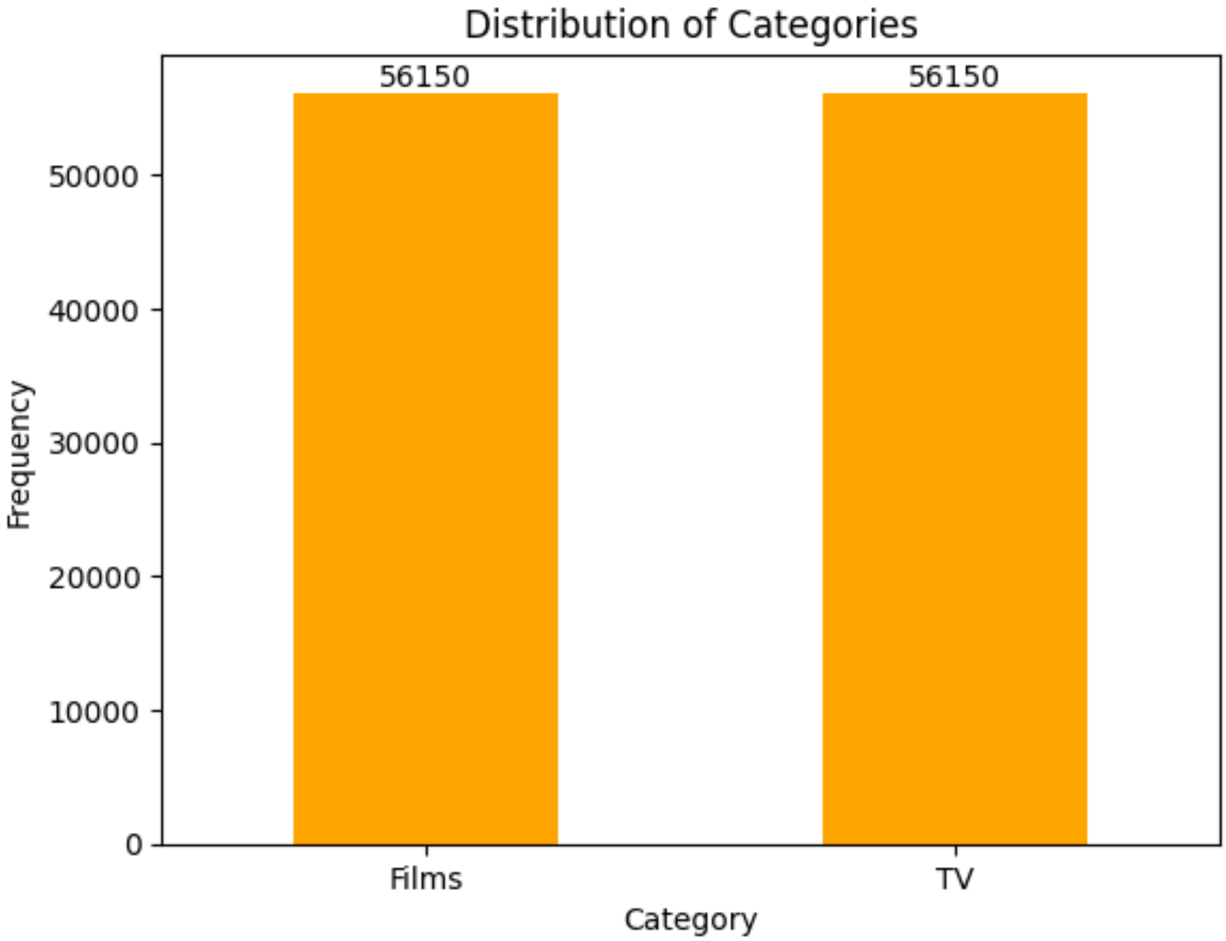


```
variances = top10.var()

variances.plot(kind='bar', color='lightgreen')
plt.title('Variance of Numeric Variables')
plt.xlabel('Variable')
plt.ylabel('Variance')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

```
<ipython-input-5-4c8b6dacff89>:1: FutureWarning: The default value of numeric_
  variances = top10.var()
```

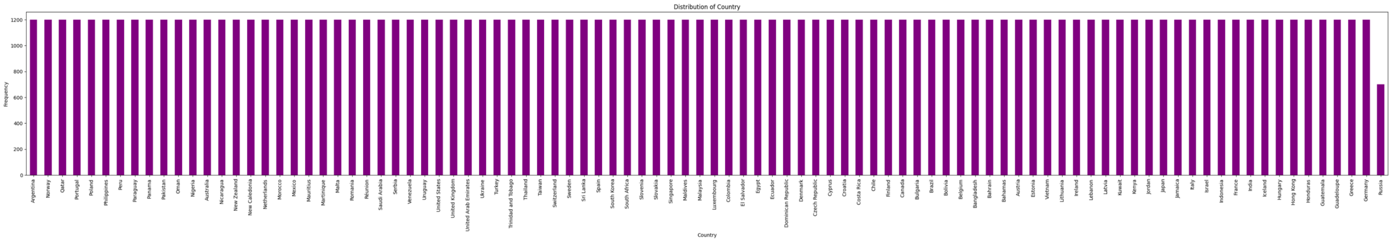


```
category_counts = top10['category'].value_counts()
category_counts.plot(kind='bar', color='orange')

for i, count in enumerate(category_counts):
    plt.text(i, count, str(count), ha='center', va='bottom')

plt.title('Distribution of Categories')
plt.xlabel('Category')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()
```



```
plt.figure(figsize=(50, 6))
top10['country_name'].value_counts().plot(kind='bar', color='purple')
plt.title('Distribution of Countries')
plt.xlabel('Country')
plt.ylabel('Frequency')

print(top10['country_name'].nunique())

plt.title('Distribution of Country')
plt.xticks(rotation=90)
plt.show()
```

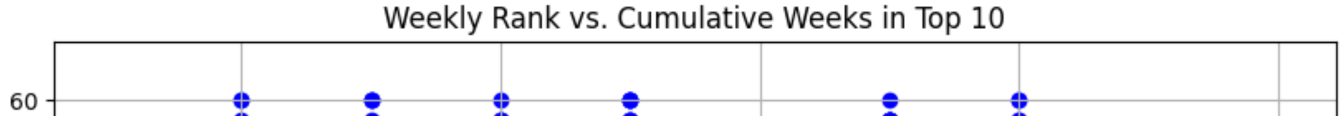
```
94
```

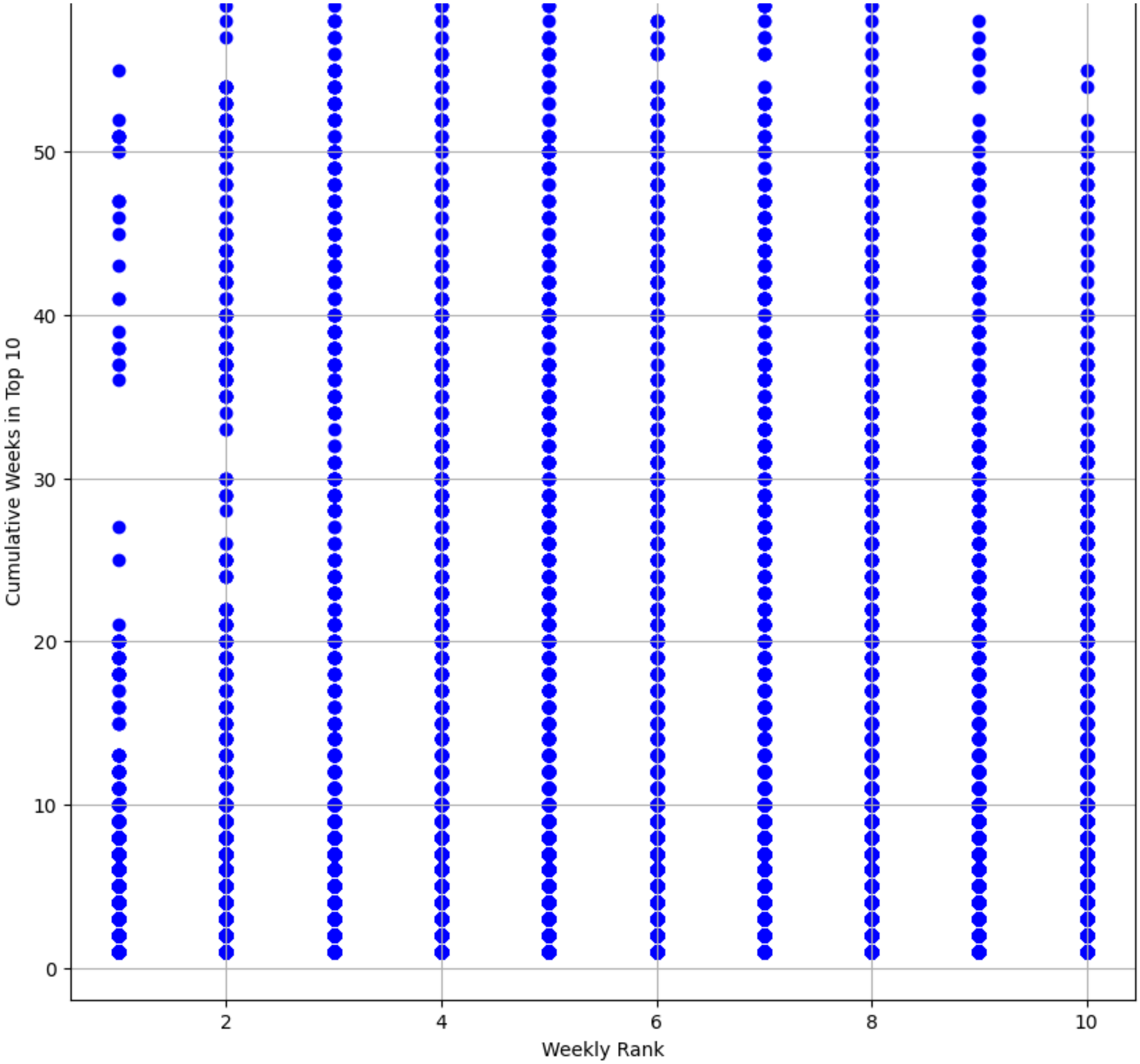


```
spearman_corr = top10[['weekly_rank', 'cumulative_weeks_in_top_10']].corr(meth
print(spearman_corr)
```

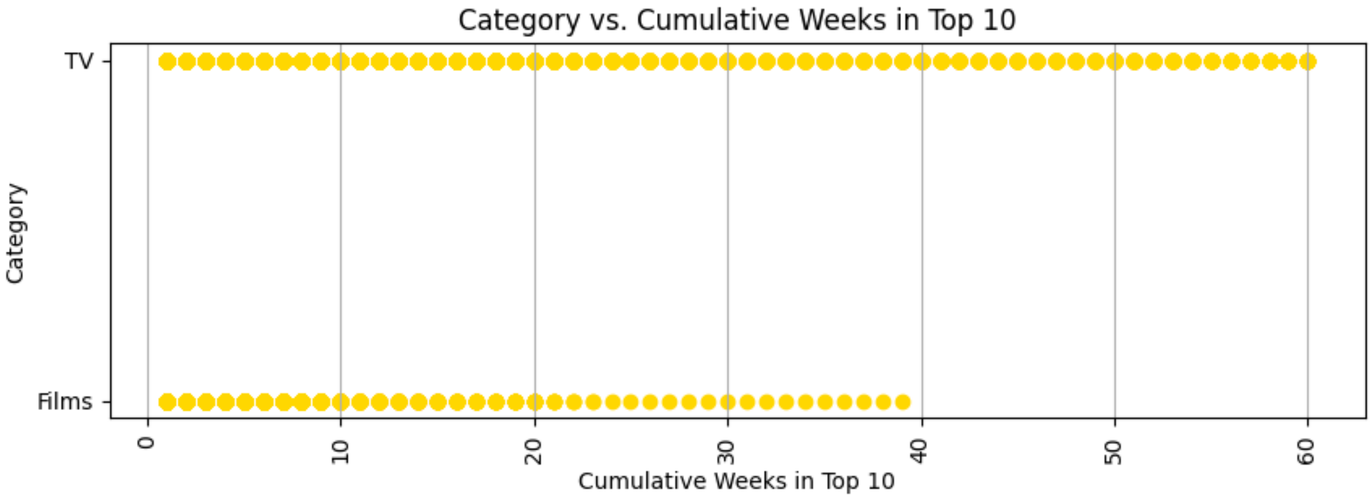
```
                          weekly_rank  cumulative_weeks_in_top_10
weekly_rank                  1.000000                    0.028064
cumulative_weeks_in_top_10   0.028064                    1.000000
```

```
plt.figure(figsize=(10, 10))
plt.scatter(top10['weekly_rank'], top10['cumulative_weeks_in_top_10'], color='
plt.title('Weekly Rank vs. Cumulative Weeks in Top 10')
plt.xlabel('Weekly Rank')
plt.ylabel('Cumulative Weeks in Top 10')
plt.grid(True)
plt.show()
```

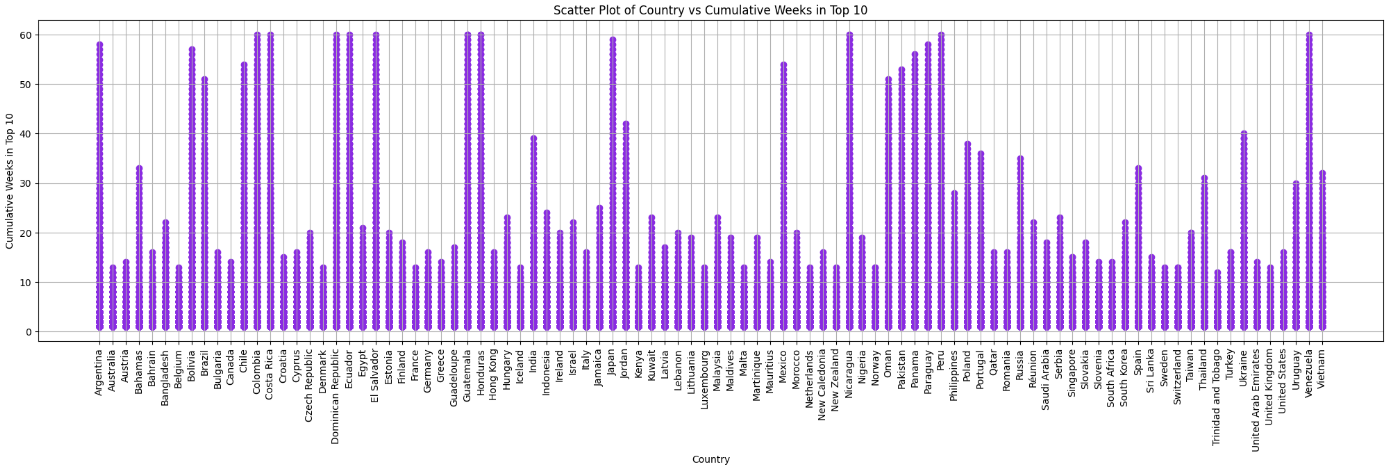



Cumulative Weeks in Top 10
0
10
20
30
40
50
2
4
6
8
10
Weekly Rank

```
plt.figure(figsize=(10, 3))
plt.scatter(top10['cumulative_weeks_in_top_10'], top10['category'], color='gol
plt.xlabel('Cumulative Weeks in Top 10')
plt.ylabel('Category')
plt.title('Category vs. Cumulative Weeks in Top 10')
plt.xticks(rotation=90)
plt.grid(axis='x')
plt.show()
```



```
plt.figure(figsize=(25, 6))
plt.scatter(top10['country_name'], top10['cumulative_weeks_in_top_10'], color=
plt.xlabel('Country')
plt.ylabel('Cumulative Weeks in Top 10')
plt.title('Scatter Plot of Country vs Cumulative Weeks in Top 10')
plt.xticks(rotation=90)
plt.grid(True)
plt.show()
```



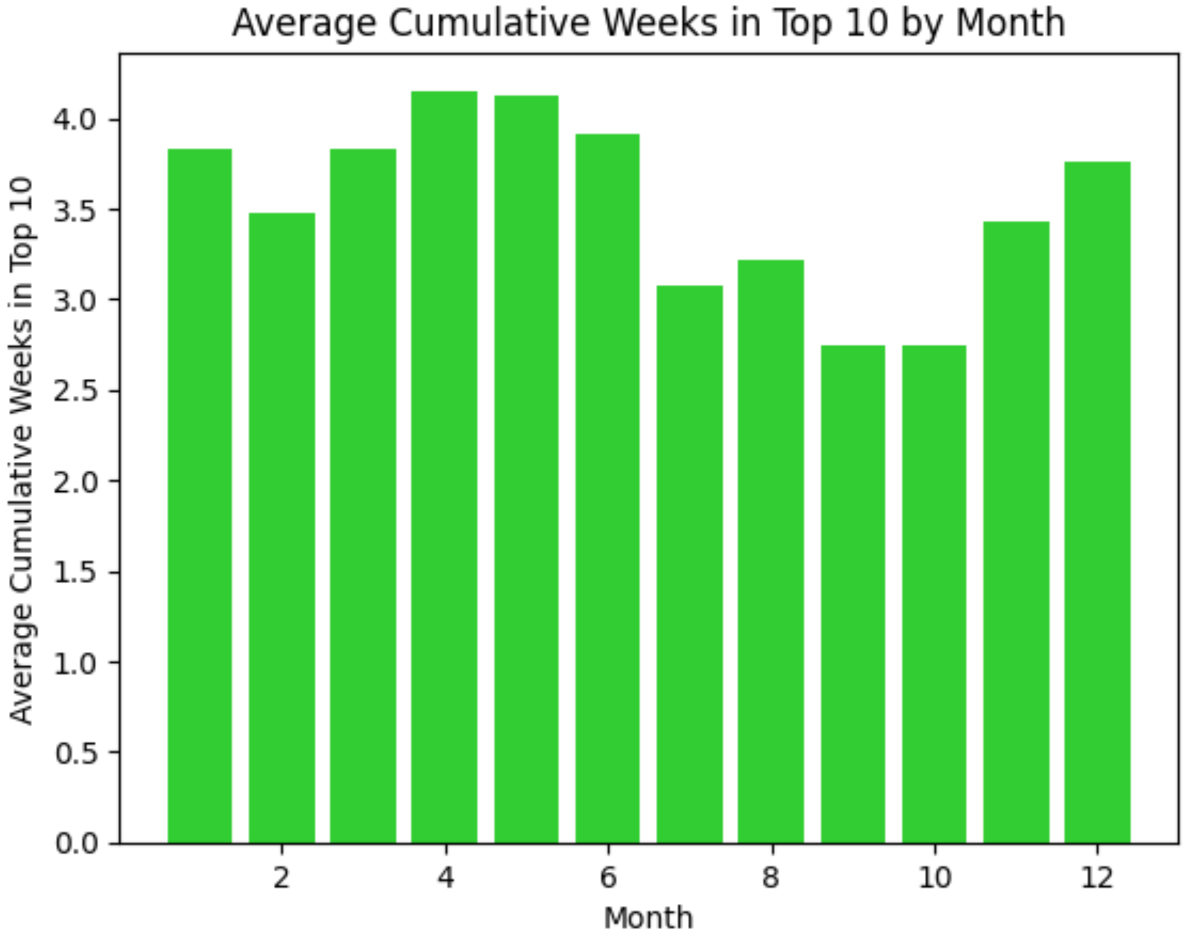
Initial Results and Code

Does date correlate with shows that have larger cumulative weeks in Top 10?

```
#Calculate the average cumulative_weeks_in_top_10 for each month
top10['week'] = pd.to_datetime(top10['week'])
top10['month'] = top10['week'].dt.month

average_weeks_top_10_monthly = top10.groupby('month')['cumulative_weeks_in_top
```

```
print(average_weeks_top_10_monthly)

#view distribution of the average cumulative_weeks_in_top_10 for each month
plt.bar(average_weeks_top_10_monthly.index, average_weeks_top_10_monthly.value
plt.xlabel('Month')
plt.ylabel('Average Cumulative Weeks in Top 10')
plt.title('Average Cumulative Weeks in Top 10 by Month')
plt.show()

#The data is non-normally distributed thus we must use non-parametric statisti
#determine the significance of the relationship between month and cumulative w
from scipy.stats import kruskal

data_by_month = [group.values for name, group in top10.groupby('month')['cumul
print(data_by_month)

kruskal_result = kruskal(*data_by_month)
print(kruskal_result)
```

```
month
1     3.827766
2     3.475399
3     3.834274
4     4.151882
5     4.122366
6     3.916801
7     3.080856
8     3.216422
9     2.750665
10    2.745319
11    3.424202
12    3.754388
Name: cumulative_weeks_in_top_10, dtype: float64
```



```
[array([ 2,  3,  1, ..., 21,  1,  3]), array([1, 2, 1, ..., 3, 1, 3]), array([
KruskalResult(statistic=798.6599409998698, pvalue=3.674882741765935e-164)
```

```
from itertools import combinations
from scipy.stats import ttest_ind

```

```
months = top10['month'].unique()

#Assuming the dataset is sufficiently large, we can perform t-tests on the
#average cumualtive weeks in top 10 between each month

''' from these results we can infer which months have significant relationship
whether there is a TV consumption trend'''

for month1, month2 in combinations(months, 2):
    data_month1 = top10[top10['month'] == month1]['cumulative_weeks_in_top_10'
    data_month2 = top10[top10['month'] == month2]['cumulative_weeks_in_top_10'

    t_statistic, p_value = ttest_ind(data_month1, data_month2)

    print(f"t-test between month {month1} and month {month2}:")
    print("t-statistic:", t_statistic)
    print("p-value:", p_value)
    print()
```

```
t-test between month 8 and month 7:
t-statistic: 2.2826475001277426
p-value: 0.022457711439931027

t-test between month 8 and month 6:
t-statistic: -8.345107815884798
p-value: 7.52103500068677e-17

t-test between month 8 and month 5:
t-statistic: -11.433355372990585
p-value: 3.4038920819053535e-30

t-test between month 8 and month 4:
t-statistic: -11.723393867829902
p-value: 1.1965784751814039e-31

t-test between month 8 and month 3:
t-statistic: -8.15789419545575
p-value: 3.5866989719926787e-16

t-test between month 8 and month 2:
t-statistic: -3.6105253323563606
p-value: 0.00030624142085900005

t-test between month 8 and month 1:
t-statistic: -9.479776916863766
p-value: 2.7728148746402483e-21

t-test between month 8 and month 12:
```

```
t-statistic: -8.000897214186281
p-value: 1.294415927300952e-15

t-test between month 8 and month 11:
t-statistic: -3.2330023507875816
p-value: 0.0012267304830323638

t-test between month 8 and month 10:
t-statistic: 8.483800036374486
p-value: 2.3016125552583263e-17

t-test between month 8 and month 9:
t-statistic: 7.776981868394483
p-value: 7.745411536924857e-15

t-test between month 7 and month 6:
t-statistic: -9.456778357490943
p-value: 3.454243130848794e-21

t-test between month 7 and month 5:
t-statistic: -12.607706363093271
p-value: 2.4437547961980922e-36

t-test between month 7 and month 4:
t-statistic: -12.612229835380301
p-value: 2.351690185059873e-36

t-test between month 7 and month 3:
t-statistic: -9.239488276367874
p-value: 2.674174241822577e-20
```

Are shows that make the top 10 in the United States more likely to make the top 10 in other countries? Western countries? English speaking countries?

```
# A list of every country for which top 10 data is included in the dataset
countries = top10["country_name"].unique()
print(countries)

print("\n")

num_countries = len(countries)
print(num_countries)
```

```
['Argentina' 'Australia' 'Austria' 'Bahamas' 'Bahrain' 'Bangladesh'
 'Belgium' 'Bolivia' 'Brazil' 'Bulgaria' 'Canada' 'Chile' 'Colombia'
 'Costa Rica' 'Croatia' 'Cyprus' 'Czech Republic' 'Denmark'
 'Dominican Republic' 'Ecuador' 'Egypt' 'El Salvador' 'Estonia' 'Finland'
 'France' 'Germany' 'Greece' 'Guadeloupe' 'Guatemala' 'Honduras'
 'Hong Kong' 'Hungary' 'Iceland' 'India' 'Indonesia' 'Ireland' 'Israel'
 'Italy' 'Jamaica' 'Japan' 'Jordan' 'Kenya' 'Kuwait' 'Latvia' 'Lebanon'
 'Lithuania' 'Luxembourg' 'Malaysia' 'Maldives' 'Malta' 'Martinique'
 'Mauritius' 'Mexico' 'Morocco' 'Netherlands' 'New Caledonia'
 'New Zealand' 'Nicaragua' 'Nigeria' 'Norway' 'Oman' 'Pakistan' 'Panama'
 'Paraguay' 'Peru' 'Philippines' 'Poland' 'Portugal' 'Qatar' 'Romania'
 'Russia' 'Réunion' 'Saudi Arabia' 'Serbia' 'Singapore' 'Slovakia'
 'Slovenia' 'South Africa' 'South Korea' 'Spain' 'Sri Lanka' 'Sweden'
 'Switzerland' 'Taiwan' 'Thailand' 'Trinidad and Tobago' 'Turkey'
 'Ukraine' 'United Arab Emirates' 'United Kingdom' 'United States'
 'Uruguay' 'Venezuela' 'Vietnam']


94
```

```
#The shows/movies that make the Top 10 in the United States
us_shows = top10[top10['country_name'] == 'United States']

#count the frequency of how many times each show_title appears in the Top 10 f
us_show_frequency = us_shows.groupby('show_title').size()
us_show_frequency_sorted = us_show_frequency.sort_values(ascending=False)

#the top 10 most frequently occuring shows/movies that make the top 10 in the
us_top10 = us_show_frequency_sorted.head(10)
print(us_top10)

```

```
show_title
CoComelon          52
Stranger Things    43
Ozark              23
Manifest           20
All American       18
Virgin River       15
Bridgerton         12
You                11
The Witcher        11
Squid Game         11
dtype: int64
```

```
#The shows/movies that make the Top 10 in Canada
can_shows = top10[top10['country_name'] == 'Canada']

#count the frequency of how many times each show_title appears in the Top 10 f
can_show_frequency = can_shows.groupby('show_title').size()
can_show_frequency_sorted = can_show_frequency.sort_values(ascending=False)

#the top 10 most frequently occuring shows/movies that make the top 10 in Cana
can_top10 = can_show_frequency_sorted.head(10)
print(can_top10)
```

```
show_title
Stranger Things    37
Ozark              21
Blindspot          21
Manifest           14
Young Sheldon      13
Maid               12
Bridgerton         12
The Witcher        11
Love Is Blind      11
You                11
dtype: int64
```

```
'''Count the number of times Top 10 US shows/movies make the Top 10 in other c

top10_frequency_by_country = {}

#Compute frequency of each show in the top 10 for each country that is not the
for country in top10['country_name'].unique():
    if country != 'United States':
        country_shows = top10[top10['country_name'] == country]
        country_show_frequency = country_shows['show_title'].value_counts().he
        top10_frequency_by_country[country] = country_show_frequency

#DataFrame to store contingency table
contingency_table = pd.DataFrame(index=us_top10.index, columns=top10_frequency_

#Fill the contingency table with frequency counts
for country, country_frequency in top10_frequency_by_country.items():
    for show in us_top10.index:
        # Fill in the frequency count for each show in the top 10 for the curr
        contingency_table.loc[show, country] = country_frequency.get(show, 0)

#Fill the contingency table for the US
```

```
for show in us_top10.index:
    contingency_table.loc[show, 'United States'] = us_show_frequency.get(show,

print(contingency_table)
```

```
                Argentina Australia Austria Bahamas Bahrain Bangladesh  \
show_title
CoComelon               0         0       0      17       0          0
Stranger Things        24        41      40      30      36         44
Ozark                   0        14       0      13       0          0
Manifest               15        26      20      16       0         21
All American            0         0       0       0       0          0
Virgin River            0        13       0      12       0          0
Bridgerton              0        14      13      11      13          0
You                     0        12      15       0      11          0
The Witcher             0         0      12       0      12          0
Squid Game             11        11      12       0      16         22

                Belgium Bolivia Brazil Bulgaria  ... Thailand  \
show_title                                       ...
CoComelon             0       0      0        0  ...        0
Stranger Things      41      27     31       46  ...       21
Ozark                 0       0      0        0  ...        0
Manifest             15       0      0       20  ...        0
All American          0       0      0        0  ...        0
Virgin River         12       0      0        0  ...        0
Bridgerton           14       0      0       15  ...        0
You                  12       0      0       15  ...        0
The Witcher          11       0      0       15  ...        0
Squid Game           11       0     10       16  ...        0

                Trinidad and Tobago Turkey Ukraine United Arab Emirates  \
show_title
CoComelon                         0      0       0                    0
Stranger Things                  34     34      46                   34
Ozark                            12      0      28                   10
Manifest                         16      0       0                    0
All American                      0      0       0                    0
Virgin River                     12      0       0                    0
Bridgerton                       13      0      35                   13
You                              13     19      21                   10
The Witcher                      11      0      27                   11
Squid Game                        0     16       0                   14

                United Kingdom Uruguay Venezuela Vietnam United States
show_title
CoComelon                    0       0         0       0          52.0
Stranger Things             46      21        19      25          43.0
Ozark                       17       0         0       0          23.0
```

```
Manifest                   0     16        15       0           20.0
All American               0      0         0       0           18.0
Virgin River              12      0         0       0           15.0
Bridgerton                13      0         0       0           12.0
You                       13     12         0       0           11.0
The Witcher                0      0         0       0           11.0
Squid Game                10      0         0       0           11.0

[10 rows x 94 columns]
```

```
'''Calculate whether the results in the contingency table above are significan

import numpy as np
from scipy.stats import chi2_contingency

p_values = {}

for country in contingency_table.columns:
    # Extract observed frequencies for the current country
    observed_frequencies = contingency_table[country].values.astype(float)
    row_totals = contingency_table.sum(axis=1)
    column_totals = contingency_table.sum(axis=0)
    expected_frequencies = np.outer(row_totals, column_totals) / row_totals.su
    expected_frequencies = expected_frequencies[:, contingency_table.columns.g

    chi2_stat, p_value, _, _ = chi2_contingency([observed_frequencies, expecte

    p_values[country] = p_value


for country, p_value in p_values.items():
    print(f"{country}: {p_value}")


print("\n")

'''Here, significant countries suggests that in these countries, a show/movie 
the Top 10 in the US is likely to make the Top 10 in the listed country. Possi
influence the markets in these other countries.'''

significant_countries = [country for country, p_value in p_values.items() if p
print("List of Significant Countries", significant_countries)
```

```
Argentina: 0.028787031633686188
Australia: 0.014589738469597569
```

```
Austria: 0.09908073729779852
Bahamas: 1.6428018500900734e-07
Bahrain: 0.00040544203937159415
Bangladesh: 0.00016475360623926946
Belgium: 0.07575420324457381
Bolivia: 0.0070270105095409235
Brazil: 0.002305190808831503
Bulgaria: 0.05274111160975641
Canada: 3.7241957280345492e-06
Chile: 0.0027174447198394522
Colombia: 0.004851091032492417
Costa Rica: 0.022772288597190448
Croatia: 0.020101224321794453
Cyprus: 0.036197197212430726
Czech Republic: 1.9325766371444175e-05
Denmark: 0.007930065789479997
Dominican Republic: 0.0206159800194768
Ecuador: 0.03946046413725036
Egypt: 2.577880163511151e-05
El Salvador: 0.06337385755013583
Estonia: 0.00018414750490193212
Finland: 0.03380490569942349
France: 0.023836926402612604
Germany: 0.0017473424514078122
Greece: 0.021830932614843312
Guadeloupe: 0.02470583816573554
Guatemala: 0.04796088964421329
Honduras: 0.07860712593283108
Hong Kong: 0.02204444827856596
Hungary: 1.5212543936266608e-06
Iceland: 0.0004255930132056846
India: 6.65118228025686e-05
Indonesia: 0.012185316340483103
Ireland: 4.3212020022413026e-12
Israel: 0.002209538283476711
Italy: 0.0011210599199247232
Jamaica: 0.05537084550764695
Japan: 0.0019156756229678359
Jordan: 2.9884026444445108e-05
Kenya: 0.0505726389320982
Kuwait: 0.00012518788664030034
Latvia: 0.0036623239495632374
Lebanon: 0.00014233708594856714
Lithuania: 0.01013552331735906
Luxembourg: 0.03257002315611079
Malaysia: 0.014188477714176969
Maldives: 0.004153869248331972
Malta: 0.0008714551433422026
Martinique: 0.034957038135512285
Mauritius: 0.015399724013805837
```

```
Mexico: 0.0027843972455754208
Morocco: 0.000611412614606106
Netherlands: 5.587868575947894e-05
New Caledonia: 0.021102821293318457
New Zealand: 1.9388276955265418e-05
Nicaragua: 0.04409396244769613
Nigeria: 0.0019520571207548723
```

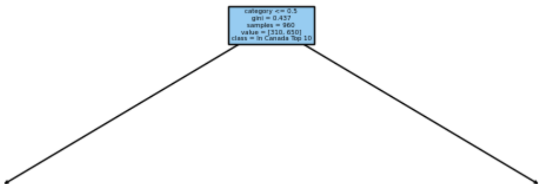
```
print("Number of significant countries: ", len(significant_countries))

#English language countries as per The University of Tennessee Knoxville
#Note this list does not include the UK, Australia but I have added them in

#strong limitation is the definition of english-language country
#not every country is listed and not every country was checked against the 94
'''https://gradschool.utk.edu/future-students/office-of-graduate-admissions/ap
admission-requirements/testing-requirements/countries-with-english-as-official-

english_speaking_countries = [
    'Anguilla', 'Antigua and Barbuda', 'Bahamas', 'Barbados', 'Belize', 'Belgi
    'British Virgin Islands', 'Burundi', 'Cameroon', 'Canada', 'Cayman Islands
    'Dominica', 'Fiji', 'Gambia', 'Ghana', 'Grenada', 'Guyana', 'Hong Kong', '
    'Liberia', 'Malawi', 'Malta', 'Marshall Islands', 'Micronesia', 'Namibia',
    'Nigeria', 'Niue', 'Norfolk Island', 'Northern Mariana Islands', 'Pakistan
    'Philippines', 'Pitcairn Islands', 'Rwanda', 'Saint Kitts and Nevis', 'Sai
    'Sierra Leone', 'Singapore', 'Sint Maarten', 'Solomon Islands', 'Somalia',
    'Swaziland', 'Tanzania', 'Tonga', 'Trinidad and Tobago', 'Turks and Caicos
    'Zimbabwe', 'United Kingdom', 'Australia']

# Of the significant countries, which are english-speaking?
num_significant_english_speaking_countries = 0

for country in significant_countries:
    if country in english_speaking_countries:
        num_significant_english_speaking_countries = num_significant_english_s

print("Number of significant english speaking countries: ", num_significant_en
```

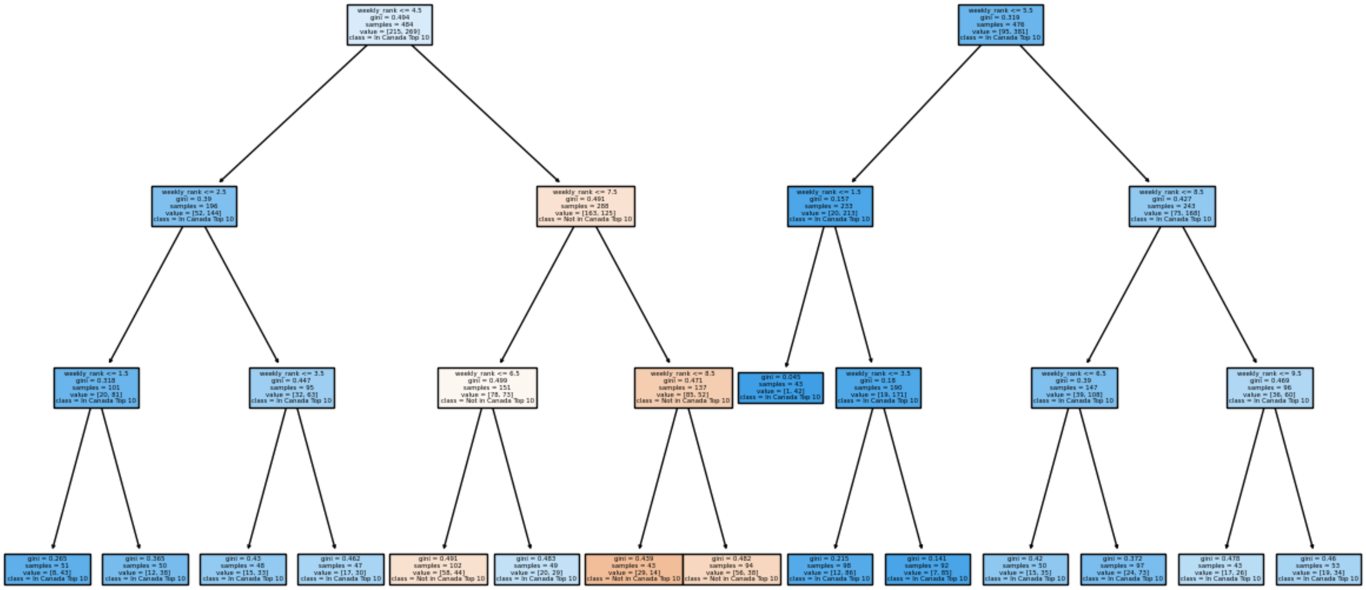
```
Number of significant countries:  52
Number of significant english speaking countries:  10
```

```
'''Can we use classification to predict whether Top 10 US shows make the Top 1

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

us_shows = top10[top10['country_name'] == 'United States']

# Determine target variable indicating whether the show made it to the top 10
target_variable = (us_shows['show_title'].isin(top10[top10['country_name'] ==

# Combine features and target variable into a dataset
prepared_dataset = pd.concat([us_shows[['show_title', 'category', 'weekly_rank
prepared_dataset.columns = ['show_title', 'category', 'weekly_rank', 'country_

print(prepared_dataset)

prepared_dataset['category'] = prepared_dataset['category'].map({'Films': 0, '
X = prepared_dataset[['weekly_rank', 'category']]
y = prepared_dataset['is_top_10_canada']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

# Instantiate the decision tree classifier
classifier = DecisionTreeClassifier(max_depth=4, random_state=42)

# Train the decision tree classifier on the training data
classifier.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = classifier.predict(X_test)

# Print confusion matrix and classification report
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("Confusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(class_report)

# Visualize the decision tree
plt.figure(figsize=(15, 10))
plot_tree(classifier, feature_names=X.columns, class_names=["Not in Canada Top
plt.show()
```

```
50
```

```
                                    show_title category  weekly_rank  \
107500                               Day Shift    Films            1
107501                          Look Both Ways    Films            2
107502  Untold: The Girlfriend Who Didn't Exist    Films            3
107503                               Uncharted    Films            4
107504                                  Sing 2    Films            5
...                                        ...      ...          ...
108695                               CoComelon       TV            6
108696                  Newly Rich, Newly Poor       TV            7
108697                             Sweet Tooth       TV            8
108698                               CoComelon       TV            9
108699                               CoComelon       TV           10

         country_name  is_top_10_canada
107500  United States                 1
107501  United States                 1
107502  United States                 1
107503  United States                 0
107504  United States                 0
...               ...               ...
108695  United States                 0
108696  United States                 0
108697  United States                 0
108698  United States                 0
108699  United States                 0

[1200 rows x 5 columns]
Confusion Matrix:
[[ 35  48]
 [ 26 131]]

Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.42      0.49        83
           1       0.73      0.83      0.78       157

    accuracy                           0.69       240
   macro avg       0.65      0.63      0.63       240
weighted avg       0.68      0.69      0.68       240
```

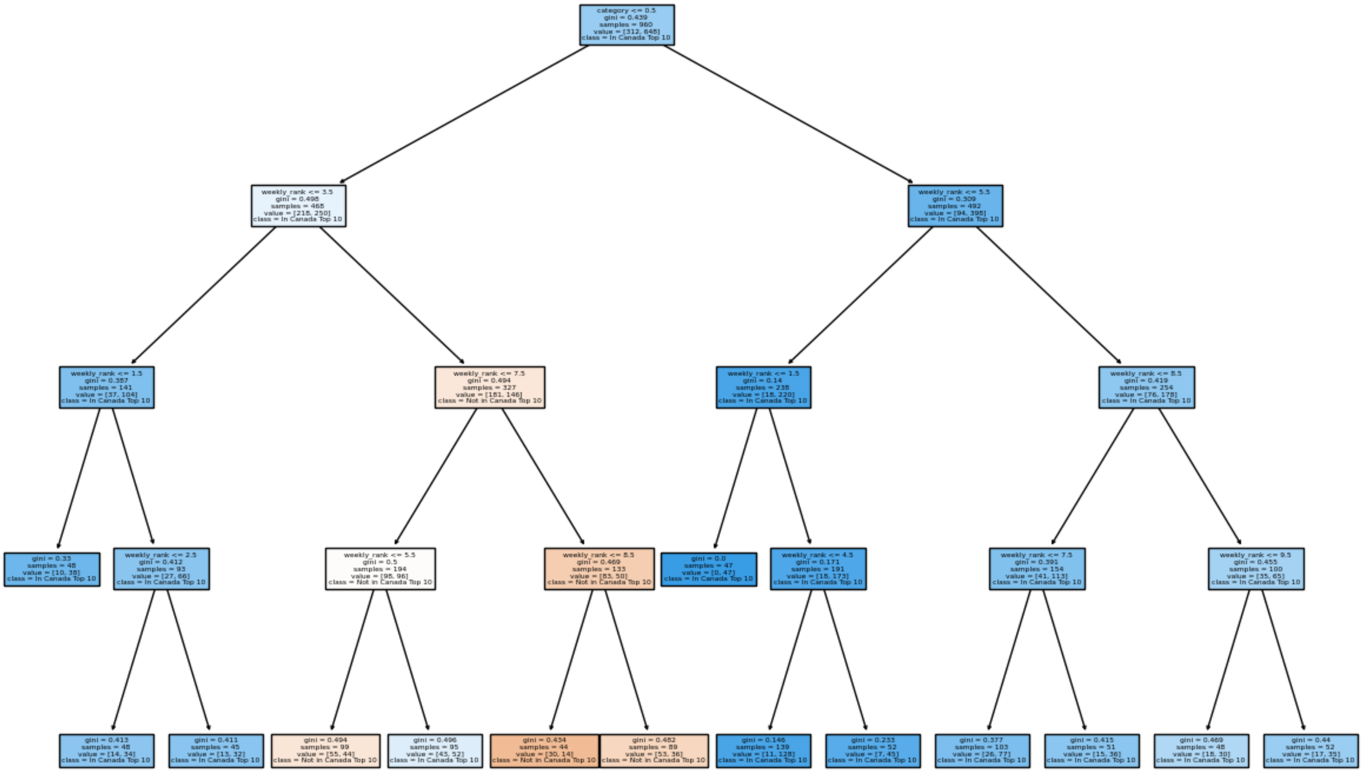



weekly_rank <= 4.5 gini = 0.494 samples = 484 value = [215, 269] class = In Canada Top 10
weekly_rank <= 2.5 gini = 0.39 samples = 196 value = [52, 144] class = In Canada Top 10
weekly_rank <= 7.5 gini = 0.491 samples = 288 value = [163, 125] class = Not in Canada Top 10
weekly_rank <= 1.5 gini = 0.318 samples = 101 value = [20, 81] class = In Canada Top 10
weekly_rank <= 3.5 gini = 0.447 samples = 95 value = [32, 63] class = In Canada Top 10
weekly_rank <= 6.5 gini = 0.499 samples = 151 value = [78, 73] class = Not in Canada Top 10
weekly_rank <= 8.5 gini = 0.471 samples = 137 value = [85, 52] class = Not in Canada Top 10
gini = 0.265 samples = 51 value = [8, 43] class = In Canada Top 10
gini = 0.365 samples = 50 value = [12, 38] class = In Canada Top 10
gini = 0.43 samples = 48 value = [15, 33] class = In Canada Top 10
gini = 0.462 samples = 47 value = [17, 30] class = In Canada Top 10
gini = 0.491 samples = 102 value = [58, 44] class = Not in Canada Top 10
gini = 0.483 samples = 49 value = [20, 29] class = In Canada Top 10
gini = 0.439 samples = 43 value = [29, 14] class = Not in Canada Top 10
gini = 0.482 samples = 94 value = [56, 38] class = Not in Canada Top 10
weekly_rank <= 5.5 gini = 0.319 samples = 476 value = [95, 381] class = In Canada Top 10
weekly_rank <= 1.5 gini = 0.157 samples = 233 value = [20, 213] class = In Canada Top 10
weekly_rank <= 8.5 gini = 0.427 samples = 243 value = [75, 168] class = In Canada Top 10
gini = 0.045 samples = 43 value = [1, 42] class = In Canada Top 10
weekly_rank <= 3.5 gini = 0.18 samples = 190 value = [19, 171] class = In Canada Top 10
weekly_rank <= 6.5 gini = 0.39 samples = 147 value = [39, 108] class = In Canada Top 10
weekly_rank <= 9.5 gini = 0.469 samples = 96 value = [36, 60] class = In Canada Top 10
gini = 0.215 samples = 98 value = [12, 86] class = In Canada Top 10
gini = 0.141 samples = 92 value = [7, 85] class = In Canada Top 10
gini = 0.42 samples = 50 value = [15, 35] class = In Canada Top 10
gini = 0.372 samples = 97 value = [24, 73] class = In Canada Top 10
gini = 0.478 samples = 43 value = [17, 26] class = In Canada Top 10
gini = 0.46 samples = 53 value = [19, 34] class = In Canada Top 10

```
'''Can we use classification to predict whether Top 10 US shows make the Top 1
Using cross validation and different random state'''
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
```

```

# Assuming top10 and us_shows are defined earlier

# Determine target variable indicating whether the show made it to the top 10
target_variable = (us_shows['show_title'].isin(top10[top10['country_name'] ==

# Combine features and target variable into a dataset
prepared_dataset = pd.concat([us_shows[['show_title', 'category', 'weekly_rank
prepared_dataset.columns = ['show_title', 'category', 'weekly_rank', 'country_

prepared_dataset['category'] = prepared_dataset['category'].map({'Films': 0, '
X = prepared_dataset[['weekly_rank', 'category']]
y = prepared_dataset['is_top_10_canada']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

# Instantiate the decision tree classifier
classifier = DecisionTreeClassifier(max_depth=4, random_state=0)

# Perform cross-validation
cv_scores = cross_val_score(classifier, X_train, y_train, cv=5)

print("Cross-validation Scores:", cv_scores)
print("Mean CV Score:", cv_scores.mean())

# Train the decision tree classifier on the training data
classifier.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = classifier.predict(X_test)

# Print confusion matrix and classification report
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("Confusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(class_report)

# Visualize the decision tree
plt.figure(figsize=(15, 10))
plot_tree(classifier, feature_names=X.columns, class_names=["Not in Canada Top
plt.show()
```

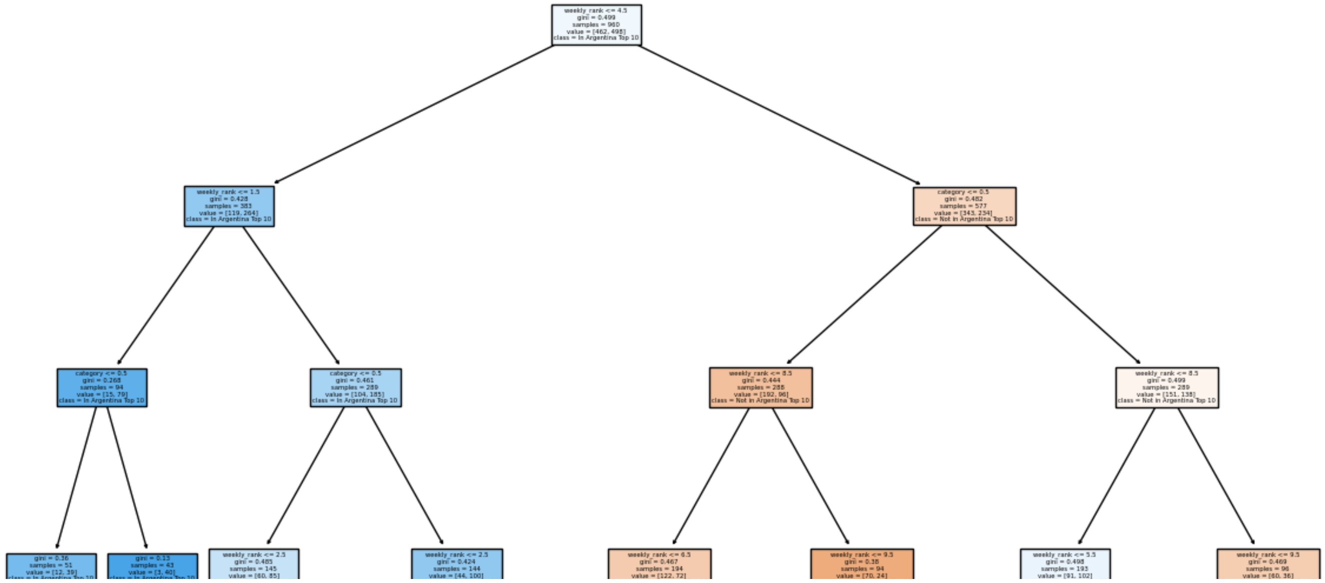
```
Cross-validation Scores: [0.72916667 0.70833333 0.68229167 0.734375   0.671875
Mean CV Score: 0.7052083333333333
Confusion Matrix:
[[ 35  46]
 [ 33 126]]

Classification Report:
              precision    recall  f1-score   support

           0       0.51      0.43      0.47        81
           1       0.73      0.79      0.76       159

    accuracy                           0.67       240
   macro avg       0.62      0.61      0.62       240
weighted avg       0.66      0.67      0.66       240
```



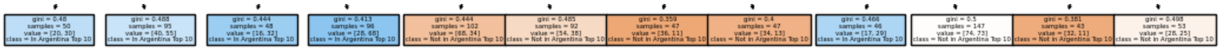
```
'''Can we use classification to predict whether Top 10 US shows make the Top 1

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

us_shows = top10[top10['country_name'] == 'United States']

# Determine target variable indicating whether the show made it to the top 10
target_variable = (us_shows['show_title'].isin(top10[top10['country_name'] ==

# Combine features and target variable into a dataset
prepared_dataset = pd.concat([us_shows[['show_title', 'category', 'weekly_rank
prepared_dataset.columns = ['show_title', 'category', 'weekly_rank', 'country_

print(prepared_dataset)

prepared_dataset['category'] = prepared_dataset['category'].map({'Films': 0, '
X = prepared_dataset[['weekly_rank', 'category']]
y = prepared_dataset['is_top_10_Argentina']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

# Instantiate the decision tree classifier
classifier = DecisionTreeClassifier(max_depth=4, random_state=42)

# Train the decision tree classifier on the training data
classifier.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = classifier.predict(X_test)
```

```

# Print confusion matrix and classification report
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("Confusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(class_report)

# Visualize the decision tree
plt.figure(figsize=(15, 10))
plot_tree(classifier, feature_names=X.columns, class_names=["Not in Argentina
plt.show()

```

```
                                     show_title category  weekly_rank  \
107500                                Day Shift    Films            1
107501                           Look Both Ways    Films            2
107502  Untold: The Girlfriend Who Didn't Exist    Films            3
107503                                Uncharted    Films            4
107504                                   Sing 2    Films            5
...                                         ...      ...          ...
108695                                CoComelon       TV            6
108696                   Newly Rich, Newly Poor       TV            7
108697                              Sweet Tooth       TV            8
108698                                CoComelon       TV            9
108699                                CoComelon       TV           10

         country_name  is_top_10_Argentina
107500  United States                    1
107501  United States                    1
107502  United States                    0
107503  United States                    0
107504  United States                    0
...               ...                  ...
108695  United States                    0
108696  United States                    1
108697  United States                    0
108698  United States                    0
108699  United States                    0

[1200 rows x 5 columns]
Confusion Matrix:
[[80 40]
 [49 71]]

Classification Report:
              precision    recall  f1-score   support
              precision    recall  f1-score   support

           0       0.62      0.67      0.64       120
           1       0.64      0.59      0.61       120

    accuracy                           0.63       240
   macro avg       0.63      0.63      0.63       240
weighted avg       0.63      0.63      0.63       240
```



Do TV shows with more seasons make the top 10 list more often? Have larger number of cumulative weeks in Top 10?



```
#subset data to include rows where season_title is included
subset = top10[top10['season_title'].notnull()]
print(subset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 54668 entries, 10 to 112299
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype         
---  ------                      --------------  -----         
 0   country_name                54668 non-null  object        
 1   country_iso2                54668 non-null  object        
 2   week                        54668 non-null  datetime64[ns]
 3   category                    54668 non-null  object        
 4   weekly_rank                 54668 non-null  int64         
 5   show_title                  54668 non-null  object        
 6   season_title                54668 non-null  object        
 7   cumulative_weeks_in_top_10  54668 non-null  int64         
 8   month                       54668 non-null  int64         
dtypes: datetime64[ns](1), int64(3), object(5)
memory usage: 4.2+ MB
None
```

```
#view format of season_title entries
print(subset['season_title'].head(60))

#add column season_number of the numeric part of season_title
subset['season_number'] = subset['season_title'].str.extract(r'(\d+)')

'''some season_title entries do not include a number'''
print(subset['season_number'].head(60))
```

```
10            Pasión de Gavilanes: Season 2
11                   Another Self: Season 1
12            Pasión de Gavilanes: Season 1
13                       Manifest: Season 1
14                    The Sandman: Season 1
15     Extraordinary Attorney Woo: Season 1
16                      High Heat: Season 1
17                       Manifest: Season 2
18                       Manifest: Season 3
19              Never Have I Ever: Season 3
30            Pasión de Gavilanes: Season 2
31                   Another Self: Season 1
32                    The Sandman: Season 1
33            Pasión de Gavilanes: Season 1
34                       Manifest: Season 1
35     Extraordinary Attorney Woo: Season 1
```

```
36                           Alba: Season 1
37                       Manifest: Season 2
38                       Manifest: Season 3
39              Never Have I Ever: Season 3
50            Pasión de Gavilanes: Season 2
51                       Manifest: Season 1
52                   Another Self: Season 1
53            Pasión de Gavilanes: Season 1
54                           Alba: Season 1
55           Keep Breathing: Limited Series
56                    The Sandman: Season 1
57                       Manifest: Season 2
58                   Virgin River: Season 4
59                       Manifest: Season 3
70            Pasión de Gavilanes: Season 2
71                           Alba: Season 1
72                       Manifest: Season 1
73                   Virgin River: Season 4
74            Pasión de Gavilanes: Season 1
75                       Manifest: Season 2
76                        Stranger Things 4
77           Keep Breathing: Limited Series
78                       Manifest: Season 3
79                   Another Self: Season 1
90                           Alba: Season 1
91            Pasión de Gavilanes: Season 2
92                       Manifest: Season 1
93                   Virgin River: Season 4
94                        Stranger Things 4
95                  Resident Evil: Season 1
96            Pasión de Gavilanes: Season 1
97                       Manifest: Season 2
98        Café con aroma de mujer: Season 1
99                 Rebelde Way: Temporada 1
110                       Stranger Things 4
111                          Alba: Season 1
112                 Resident Evil: Season 1
113             The Longest Night: Season 1
114                      Manifest: Season 1
115           Pasión de Gavilanes: Season 1
116                      Capitani: Season 2
117       Café con aroma de mujer: Season 1
118                       Stranger Things 2
```

```
#subset data to include rows where season_number is included
subset2 = subset[subset['season_number'].notnull()]
print(subset2.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 50640 entries, 10 to 112299
Data columns (total 10 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   country_name                50640 non-null  object
 1   country_iso2                50640 non-null  object
 2   week                        50640 non-null  datetime64[ns]
 3   category                    50640 non-null  object
 4   weekly_rank                 50640 non-null  int64
 5   show_title                  50640 non-null  object
 6   season_title                50640 non-null  object
 7   cumulative_weeks_in_top_10  50640 non-null  int64
 8   month                       50640 non-null  int64
 9   season_number               50640 non-null  object
dtypes: datetime64[ns](1), int64(3), object(6)
memory usage: 4.2+ MB
None
```

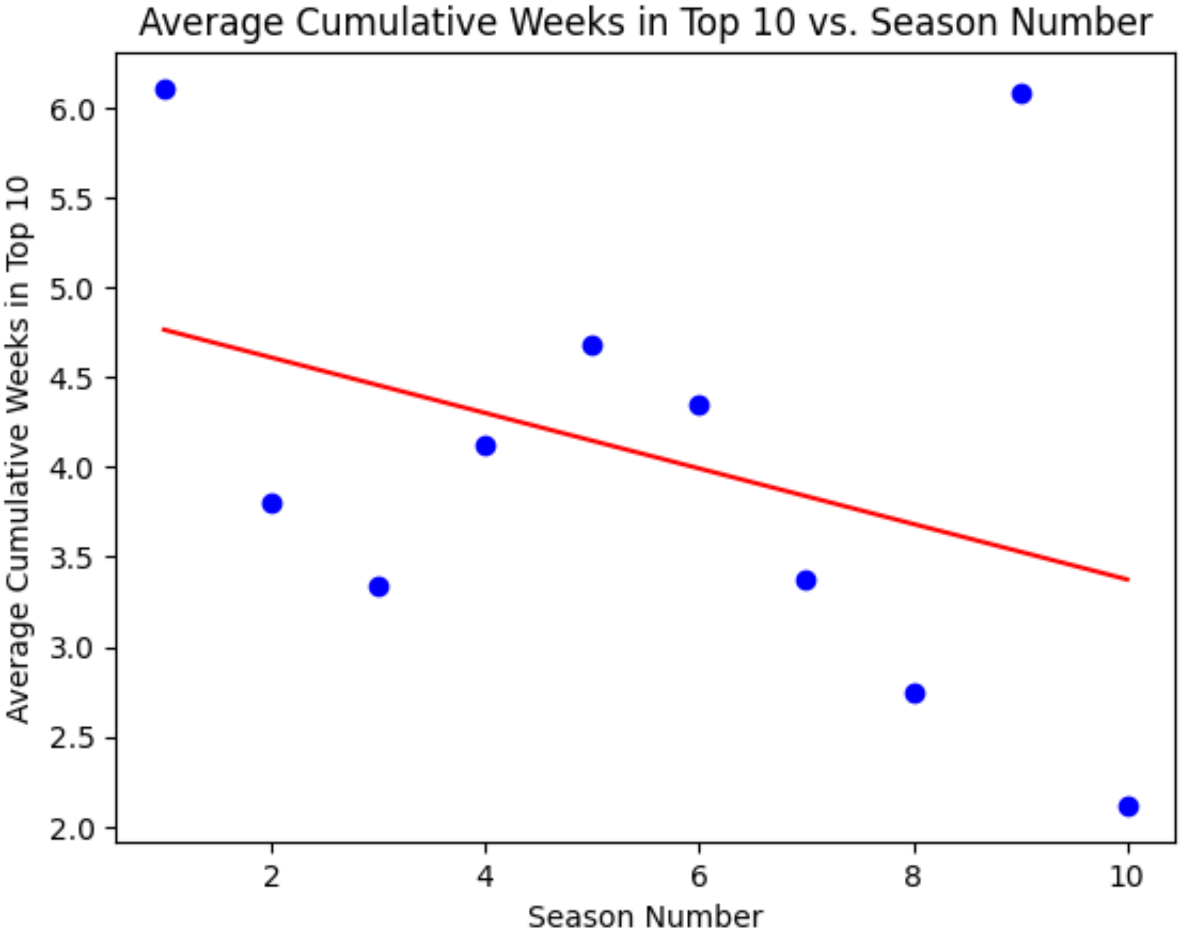
```
'''The method of isolating season_number includes many errors.
For our sake we will look at only entries where season number is 10 or smaller

#many errors exist when isolating season number from season_title
unique_season_numbers = subset2['season_number'].unique()
print(unique_season_numbers)

#convert season_number to int type
subset2.loc[:, 'season_number'] = subset2['season_number'].fillna(-5).astype(i

subset3 = subset2.loc[(subset2['season_number'] > 0) & (subset2['season_number

unique_season_numbers_again = subset3['season_number'].unique()
print(unique_season_numbers_again)
```

```
['2' '1' '3' '4' '6' '5' '42' '81' '17' '100' '8' '99' '7' '15' '11' '9'
 '14' '245' '101' '1988' '18' '24' '10' '2011' '13' '١' '12' '56' '892'
 '2045' '2020' '20' '97' '2022' '2021' '800' '1867' '60']
[ 2  1  3  4  6  5  8  7  9 10]
<ipython-input-31-4c5af1bdabe4>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  subset2.loc[:, 'season_number'] = subset2['season_number'].fillna(-5).astype
<ipython-input-31-4c5af1bdabe4>:9: DeprecationWarning: In a future version, `
  subset2.loc[:, 'season_number'] = subset2['season_number'].fillna(-5).astype
```

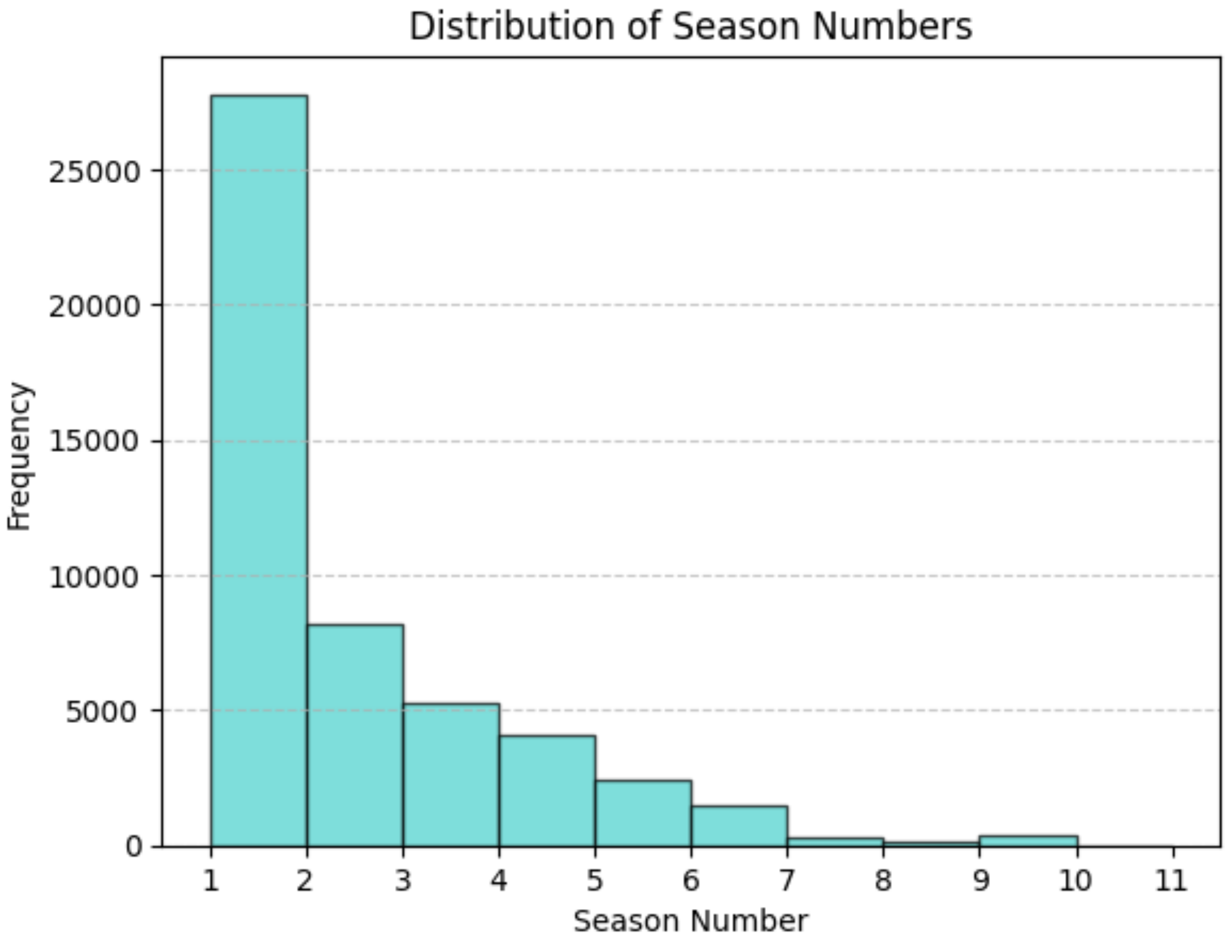
```
'''subset3 is our dataset now'''
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import pearsonr

#average cumulative weeks in top 10 for each season number 1 through 10
average_cumulative_weeks = subset3.groupby('season_number')['cumulative_weeks_
average_cumulative_weeks.columns = ['season_number', 'average_cumulative_weeks_
print(average_cumulative_weeks)

#initiliaze variables for statistical analysis
season_number = average_cumulative_weeks['season_number']
average_cumulative_weeks_in_top_10 = average_cumulative_weeks['average_cumulat

'''In this context season number can be numeric instead of rank because
we want to see if higher season number shows higher average cumulative weeks i

```

```
correlation_coefficient, p_value = pearsonr(season_number, average_cumulative_
print("Pearson correlation coefficient:", correlation_coefficient)
print("P-value:", p_value)

plt.scatter(season_number, average_cumulative_weeks_in_top_10, color='blue', l
plt.plot(season_number, np.poly1d(np.polyfit(season_number, average_cumulative
plt.xlabel('Season Number')
plt.ylabel('Average Cumulative Weeks in Top 10')
plt.title('Average Cumulative Weeks in Top 10 vs. Season Number')
plt.show()
```

```
   season_number  average_cumulative_weeks_in_top_10
0              1                            6.106364
1              2                            3.799366
2              3                            3.333964
3              4                            4.119763
4              5                            4.675585
5              6                            4.346075
6              7                            3.378277
7              8                            2.740964
8              9                            6.077586
9             10                            2.117647
Pearson correlation coefficient: -0.35851553225279903
P-value: 0.3090139342720427
```



```
'''We can view a histogram of season_number to determine which number of seaso

plt.hist(subset3['season_number'], bins=range(1, 12), edgecolor='black', color
plt.xlabel('Season Number')
plt.ylabel('Frequency')
plt.title('Distribution of Season Numbers')
plt.xticks(range(1, 12))
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

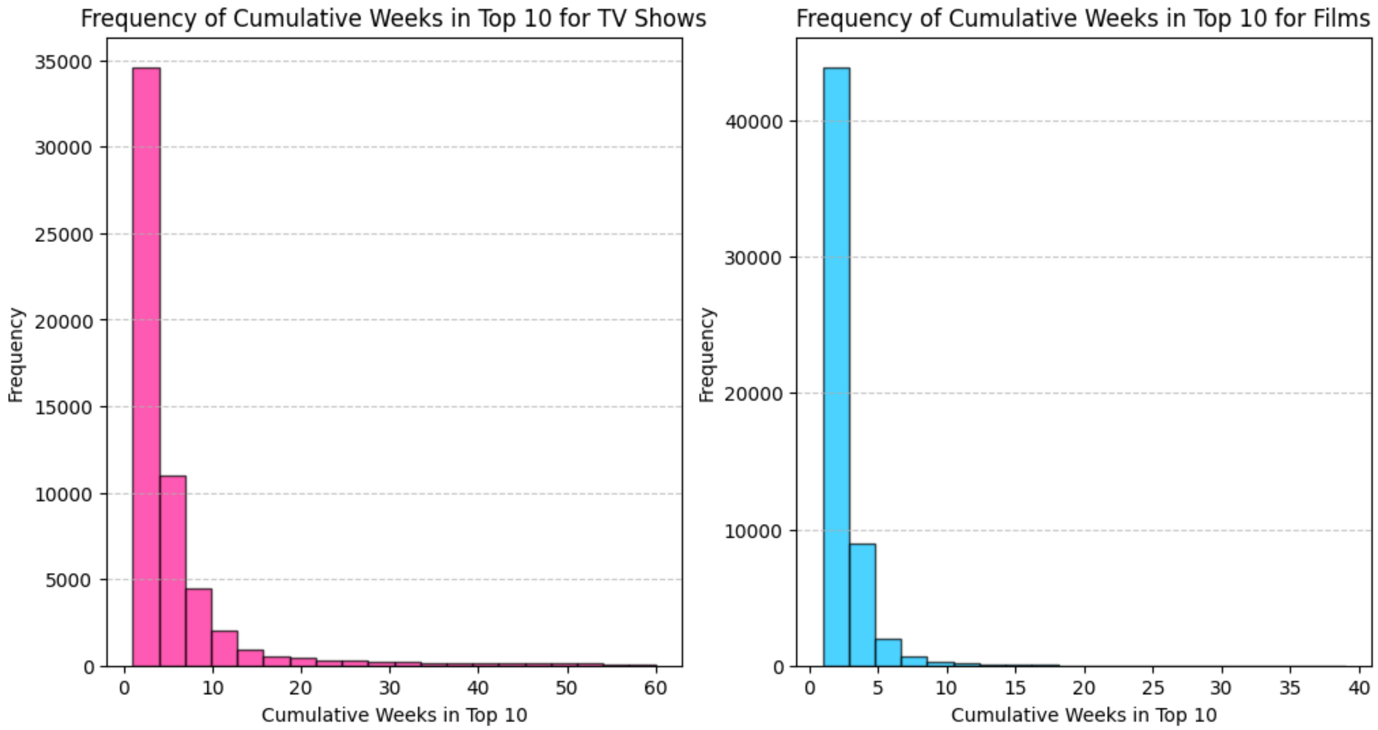


Are TV shows or movies more likely to spend a longer amount of cumulative time on the Top 10 list?

RECALL: Some TV shows made the Top 10 list for up to 60 consecutive weeks, whereas, movies only made the list for up to 40 weeks (yellow scatterplot above)

```
'''lets look at the cumulative weeks on top 10 frequency distribution for TV s
```

```

tv_shows_subset = top10[top10['category'] == 'TV']

movies_shows_subset = top10[top10['category'] == 'Films']

fig, axes = plt.subplots(1, 2, figsize=(12, 6))

axes[0].hist(tv_shows_subset['cumulative_weeks_in_top_10'], bins=20, edgecolor
axes[0].set_xlabel('Cumulative Weeks in Top 10')
axes[0].set_ylabel('Frequency')
axes[0].set_title('Frequency of Cumulative Weeks in Top 10 for TV Shows')
axes[0].grid(axis='y', linestyle='--', alpha=0.7)

axes[1].hist(movies_shows_subset['cumulative_weeks_in_top_10'], bins=20, edgec
axes[1].set_xlabel('Cumulative Weeks in Top 10')
axes[1].set_ylabel('Frequency')
axes[1].set_title('Frequency of Cumulative Weeks in Top 10 for Films')
axes[1].grid(axis='y', linestyle='--', alpha=0.7)
```



Frequency of Cumulative Weeks in Top 10 for TV Shows

Frequency of Cumulative Weeks in Top 10 for Films

```
#Calculate average cumulative weeks in top 10 for films and TV categories
avg_cumulative_weeks_tv = tv_shows_subset['cumulative_weeks_in_top_10'].mean()
print(avg_cumulative_weeks_tv)

avg_cumulative_weeks_films = movies_shows_subset['cumulative_weeks_in_top_10'].m
print(avg_cumulative_weeks_films)

#samples are unpaired and non-normally distributed so we will perform wilcoxon r
from scipy.stats import mannwhitneyu

statistic, p_value = mannwhitneyu(tv_shows_subset['cumulative_weeks_in_top_10'],

print("Wilcoxon Rank Sum test statistic:", statistic)
print("P-value:", p_value)
```

```
4.936420302760463
2.000142475512021
Wilcoxon Rank Sum test statistic: 2152311801.0
P-value: 0.0
```