Install and import libraries

In [ ]:
```
!pip install requests
!pip install bs4
!pip install lxml
```

In [1]:
```
import pandas as pd
import requests
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import spearmanr
```

Scrape data from Rotten Tomatoes

In [2]:
```
headers = {"User-Agent":"Mozilla/5.0"}
url = "https://editorial.rottentomatoes.com/guide/best-2025-movies-every-certifi
page = requests.get(url, headers = headers)
if page.status_code == 200:
    html_doc = page.text
else:
    print(f"Error accessing URL: Status Code {page.status_code}")

soup = BeautifulSoup(html_doc)
rottentomatoes_data = []

# Target the parent class wrapper
movie_rows = soup.find_all("div", class_="block-countdown")

for row in movie_rows:
    # Extract the Movie Title
    title_link = row.find("a", class_="meta-title")
    if title_link:
        title = title_link.text.strip()
    else:
        continue

    # Extract the Release Year
    year_span = row.find("span", class_="meta-year")
    # Clean the parentheses off the year string if it exists
    year = year_span.text.replace("(", "").replace(")", "").strip() if year_span

    # Extract the Tomatometer Score
    score_span = row.find("span", class_="tMeterScore")
    score = score_span.text.strip() if score_span else "N/A"

    rottentomatoes_data.append({"Title": title, "Year": year, "Score": score})

# Create and preview the DataFrame
rt_full = pd.DataFrame(rottentomatoes_data)

# Remove all non-2025 entries
rt_filtered = rt_full[rt_full['Year'] == '2025']

# Year is no longer required
rt_short = rt_filtered[["Title", "Score"]]
```

```
# Drop rows with missing titles or ratings
rt = rt_short.dropna()

print(rt.head(5))
print(rt.shape)
```

```
                          Title Score
4                     Cover-Up   98%
5           Deaf President Now!  100%
6  Come See Me in the Good Light  100%
7              The Secret Agent   98%
8         The Perfect Neighbor   99%
(141, 2)
```

Input data from IMDb

In [3]:
```
imdb_movies = pd.read_csv("title.basics.tsv", sep = "\t",
                          usecols = ["tconst", "titleType", "primaryTitle", "orig

imdb_ratings = pd.read_csv("title.ratings.tsv", sep = "\t",
                           usecols = ["tconst", "averageRating"])
imdb_full = pd.merge(imdb_movies, imdb_ratings, on="tconst", how="left")

# Remove all non-2025 entries and non-movies
imdb_filtered = imdb_full[(imdb_full['startYear'] == '2025') & (imdb_full['title

imdb_short = imdb_filtered[['primaryTitle', 'averageRating']]

# Drop rows with missing titles or ratings
imdb = imdb_short.dropna()

print(imdb.head(5))
print(imdb.shape)
```

```
                      primaryTitle  averageRating
150349                 Shyamchi Aai            7.5
331190  Zamaanat: And Justice for All            3.3
457732              Anywhere Anytime            6.7
478661                     The Twits            4.7
774429                     Red Sonja            4.6
(10095, 2)
```

Joining the two dataframes - left Join - Rotten Tomatoes on left

In [4]:
```
combined_file = pd.merge(rt, imdb, left_on="Title", right_on="primaryTitle", how

# Drop rows with missing titles or ratings
combined_file = combined_file.dropna()

print(combined_file.head(5))
print(combined_file.shape)
```

```
                          Title Score                  primaryTitle  \
0                     Cover-Up   98%                     Cover-Up
1           Deaf President Now!  100%           Deaf President Now!
2  Come See Me in the Good Light  100%  Come See Me in the Good Light
3              The Secret Agent   98%              The Secret Agent
4          The Perfect Neighbor   99%          The Perfect Neighbor

   averageRating
0            7.5
1            7.9
2            7.5
3            7.2
4            7.1
(130, 4)
```

Tidying Final File

In [5]:
```
# Remove the "primaryTitle" column
movies = combined_file.drop(columns=['primaryTitle'])

# Rename the ratings columns
movies = movies.rename(columns={
    'Score': 'Rotten',
    'averageRating': 'IMDb',
})

# Convert Rotten to a float
movies['Rotten'] = movies['Rotten'].str.replace('%', '').astype(float)

print(movies)
```

```
                           Title  Rotten  IMDb
0                      Cover-Up    98.0   7.5
1            Deaf President Now!   100.0   7.9
2    Come See Me in the Good Light   100.0   7.5
3                The Secret Agent    98.0   7.2
4            The Perfect Neighbor    99.0   7.1
..                            ...     ...   ...
136                  Mountainhead    73.0   5.4
137               Freakier Friday    73.0   6.2
138                 The Housemaid    74.0   6.7
139                 Black Phone 2    72.0   6.0
140          The Smashing Machine    71.0   6.3

[130 rows x 3 columns]
```

Perform Correlation Analysis

In [6]:
```
# Calculate the Pearson correlation coefficient

correlation = movies['IMDb'].corr(movies['Rotten'])
print(f"Pearson Correlation Coefficient: {correlation:.4f}")

# Calculate rho and p-value
rho, p_value = spearmanr(movies['IMDb'], movies['Rotten'])

print(f"Spearman's rho: {rho:.4f}")
print(f"p-value: {p_value:.4e}")

# Quick significance check
```

```
if p_value < 0.05:
    print("The correlation is statistically significant.")
else:
    print("The correlation is not statistically significant.")
```

```
Pearson Correlation Coefficient: 0.5102
Spearman's rho: 0.5284
p-value: 1.0337e-10
The correlation is statistically significant.
```

Visualise Data on a Scattergraph

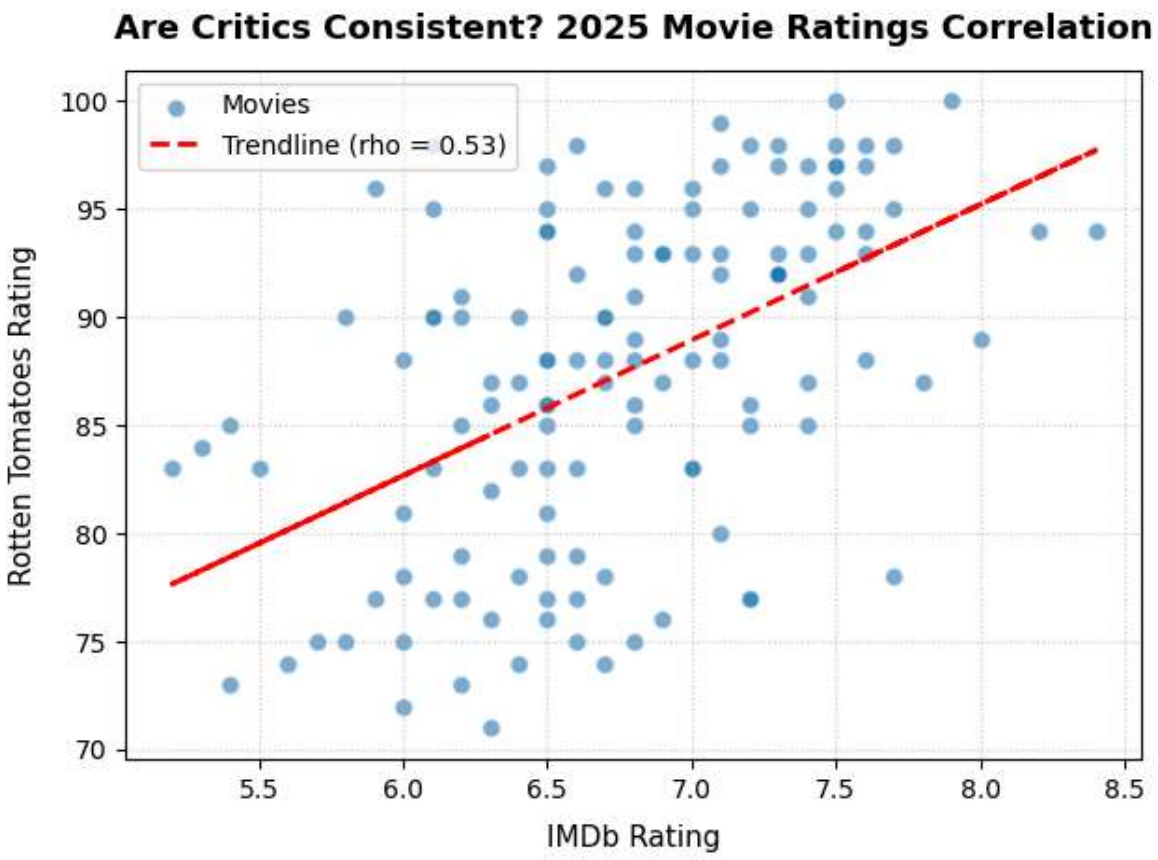
In [7]:
```
# Set up the scatter plot data
x = movies['IMDb']
y = movies['Rotten']

# Create the scatter plot points
plt.scatter(x, y, alpha=0.6, color='#1f77b4', edgecolors='w', s=50, label='Movie

# Calculate and add a line of best fit
slope, intercept = np.polyfit(x, y, 1)
trendline_y = slope * x + intercept
plt.plot(x, trendline_y, color='red', linestyle='--', linewidth=2,
         label=f'Trendline (rho = {rho:.2f})')

# Add labels, title, and styling
plt.xlabel('IMDb Rating', fontsize=11, labelpad=8)
plt.ylabel('Rotten Tomatoes Rating', fontsize=11, labelpad=8)
plt.title('Are Critics Consistent? 2025 Movie Ratings Correlation', fontsize=13,
plt.grid(True, linestyle=':', alpha=0.6)
plt.legend(loc='upper left')

# 5. Perfect the layout and save the image
plt.tight_layout()  # Ensures labels are fully readable and not truncated
plt.savefig('ratings_correlation_scatter.png', dpi=300)
plt.show()
```


Are Critics Consistent? 2025 Movie Ratings Correlation
Movies
Trendline (rho = 0.53)
Rotten Tomatoes Rating
IMDb Rating
100
95
90
85
80
75
70
5.5
6.0
6.5
7.0
7.5
8.0
8.5

In [ ]: