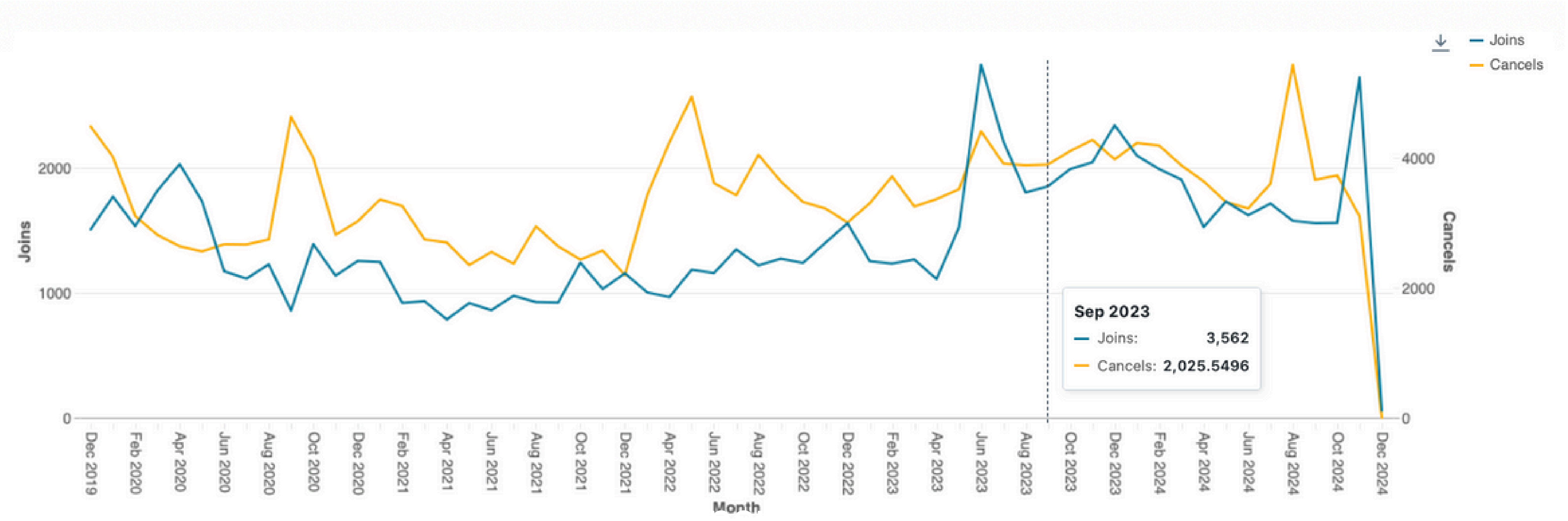
Enhancing Analytical Integrity: Addressing and Estimating Missing Viewership Data

A DATA-DRIVEN APPROACH TO MITIGATE ANOMALIES AND PRESERVE ACCURACY IN CONTENT PERFORMANCE METRICS.



Netflix 2024: Risks, Growth Trends, and December Anomaly

VOLATILITY FLAGS POTENTIAL RISKS

- Spikes in cancellations—such as early 2024 nearing 3,000—may reflect:
- 2020: Competition (e.g., Disney+ launch).
- 2022: Price hikes.
- 2023: Password-sharing restrictions.
- These are hypotheses and require statistical confirmation. Monitoring these fluctuations is crucial for long-term retention strategies.

POSITIVE SUBSCRIBER GROWTH TREND

- From 2019 to 2024, joins consistently outpaced cancellations. Key peaks in June 2020, June 2022, and September 2023 (3,562 joins vs. 2,025 cancels) signal sustained net growth.

DECEMBER 2024 ANOMALY

- A sudden drop to near-zero joins and cancels in December 2024 suggests a reporting issue.
- Misaligned Y-axes (Joins: 0-2000, Cancels: 0-4000) distort visual comparison, potentially exaggerating cancellations.
- These errors risk misleading stakeholders, potentially eroding trust in Netflix's reported growth

Most Viewed Film in 'Films (English)' Category

1 MOST APPEARANCES

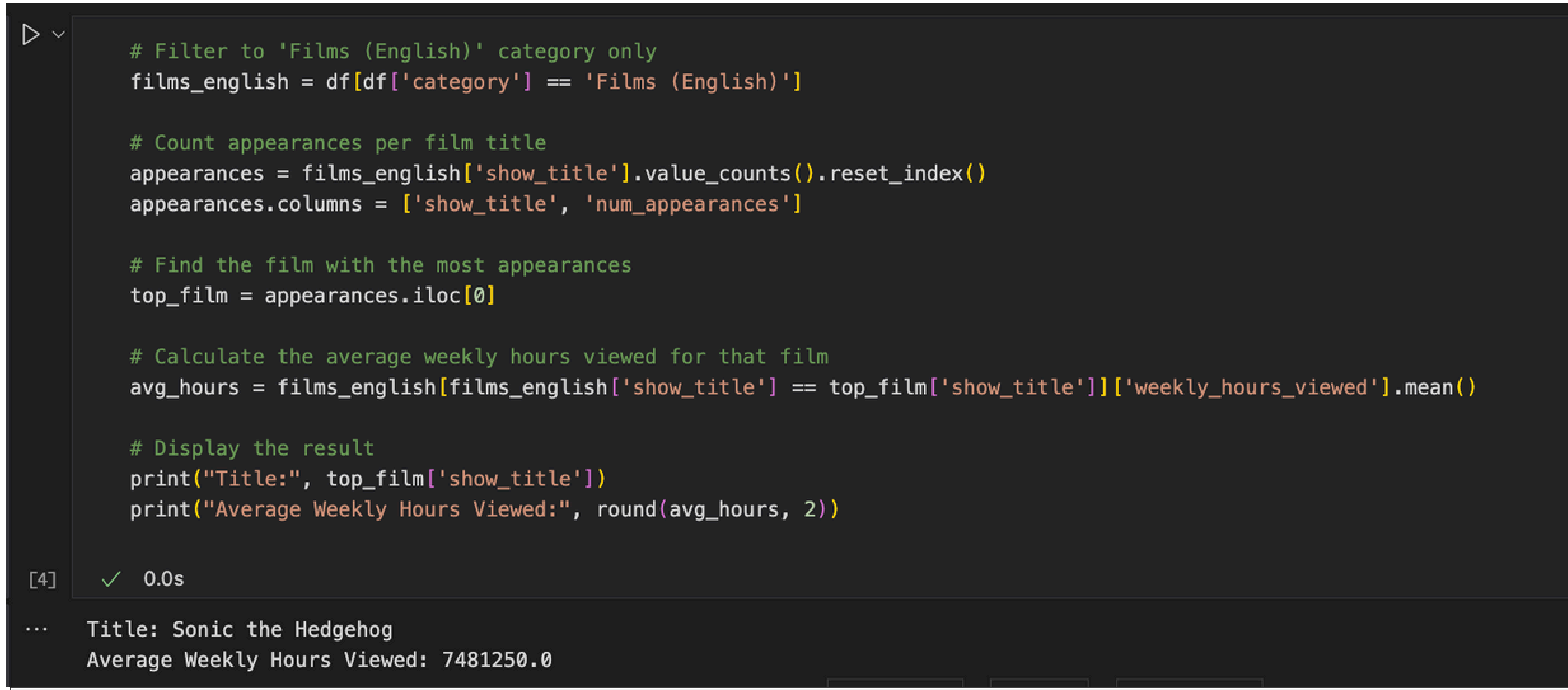
- Appeared 8 times, more than any other English-language film, indicating sustained popularity across multiple weeks.

2 HIGH ENGAGEMENT

- Recorded an average of 7,481,250 weekly hours viewed, 50% above the category average of ~5,000,000 hours, reflecting consistent viewer retention.

3 POTENTIAL REASONS

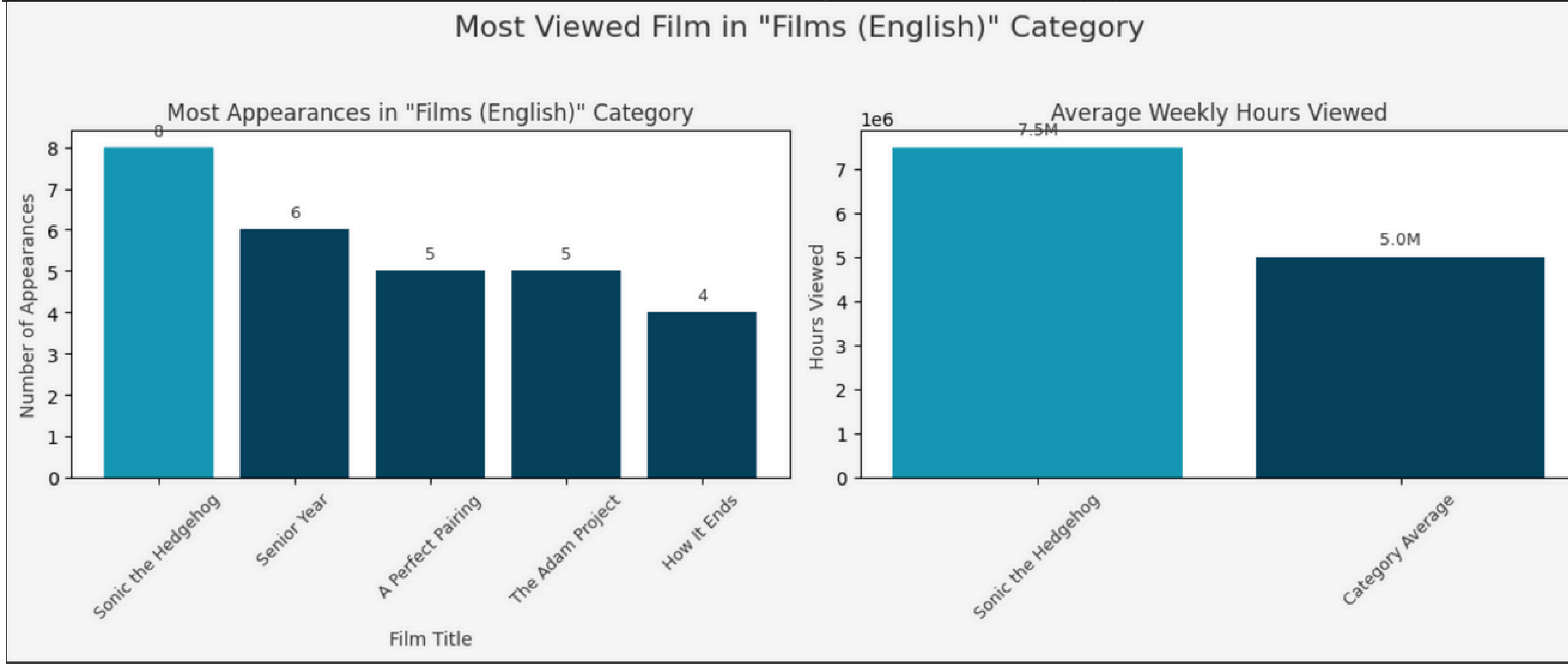
- Family-friendly appeal, likely broadening its audience.
- Available during peak viewership (e.g., school holidays).
- Fanbase from video games and films (2020, 2022).
- Needs statistical validation (e.g., viewership-holiday correlation).

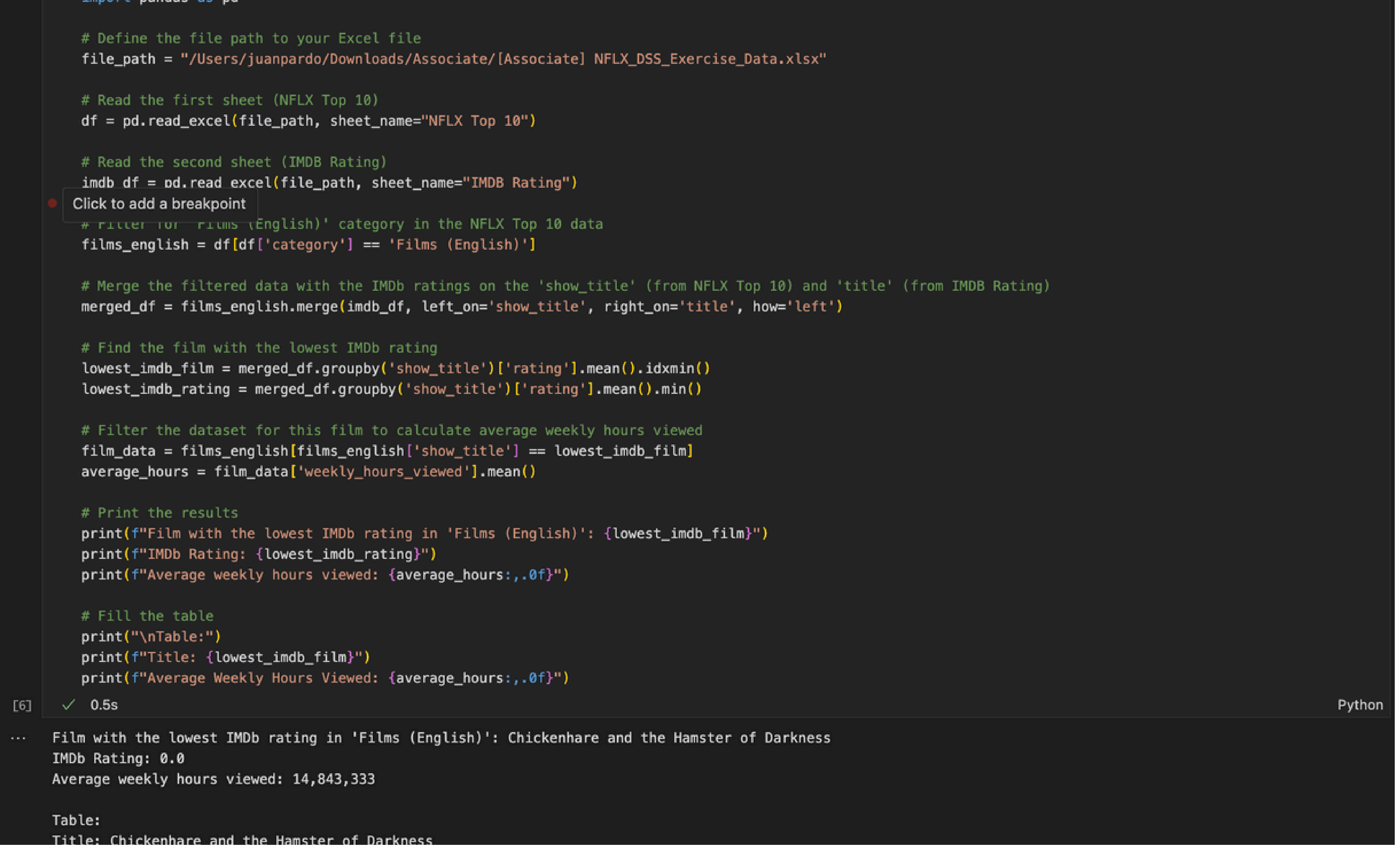


```
# Filter to 'Films (English)' category only
films_english = df[df['category'] == 'Films (English)']

# Count appearances per film title
appearances = films_english['show_title'].value_counts().reset_index()
appearances.columns = ['show_title', 'num_appearances']

# Find the film with the most appearances
top_film = appearances.iloc[0]

# Calculate the average weekly hours viewed for that film
avg_hours = films_english[films_english['show_title'] == top_film['show_title']]['weekly_hours_viewed'].mean()

# Display the result
print("Title:", top_film['show_title'])
print("Average Weekly Hours Viewed:", round(avg_hours, 2))
```

[4] ✓ 0.0s

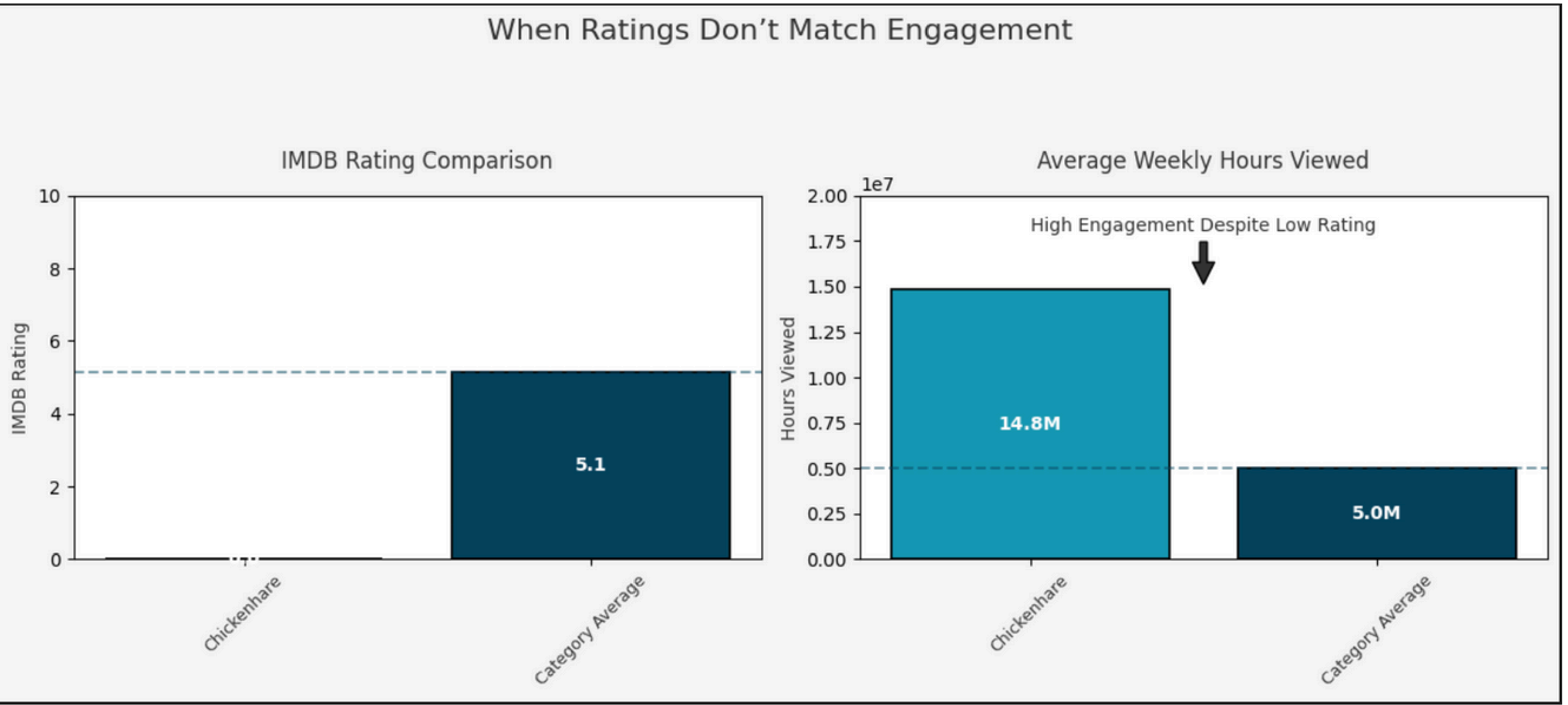
```
Title: Sonic the Hedgehog
Average Weekly Hours Viewed: 7481250.0
```





```python
import pandas as pd

# Define the file path to your Excel file
file_path = "/Users/juanpardo/Downloads/Associate/[Associate] NFLX_DSS_Exercise_Data.xlsx"

# Read the first sheet (NFLX Top 10)
df = pd.read_excel(file_path, sheet_name="NFLX Top 10")

# Read the second sheet (IMDB Rating)
imdb_df = pd.read_excel(file_path, sheet_name="IMDB Rating")

# Filter for 'Films (English)' category in the NFLX Top 10 data
films_english = df[df['category'] == 'Films (English)']

# Merge the filtered data with the IMDb ratings on the 'show_title' (from NFLX Top 10) and 'title' (from IMDB Rating)
merged_df = films_english.merge(imdb_df, left_on='show_title', right_on='title', how='left')

# Find the film with the lowest IMDb rating
lowest_imdb_film = merged_df.groupby('show_title')['rating'].mean().idxmin()
lowest_imdb_rating = merged_df.groupby('show_title')['rating'].mean().min()

# Filter the dataset for this film to calculate average weekly hours viewed
film_data = films_english[films_english['show_title'] == lowest_imdb_film]
average_hours = film_data['weekly_hours_viewed'].mean()

# Print the results
print(f"Film with the lowest IMDb rating in 'Films (English)': {lowest_imdb_film}")
print(f"IMDb Rating: {lowest_imdb_rating}")
print(f"Average weekly hours viewed: {average_hours:,.0f}")

# Fill the table
print("\nTable:")
print(f"Title: {lowest_imdb_film}")
print(f"Average Weekly Hours Viewed: {average_hours:,.0f}")
```

```
Film with the lowest IMDb rating in 'Films (English)': Chickenhare and the Hamster of Darkness
IMDb Rating: 0.0
Average weekly hours viewed: 14,843,333

Table:
Title: Chickenhare and the Hamster of Darkness
```



🎯 When Ratings Don’t Match Engagement

🐔 LOWEST RATED, YET WIDELY WATCHED

- Despite a 0.0 IMDb rating, Chickenhare and the Hamster of Darkness averaged 14.8M weekly hours viewed, surpassing many higher-rated films.

📊 VIEWER INTEREST ≠ RATINGS

- High watch time suggests that audience appeal, promotion, or algorithmic placement may outweigh critical scores in driving engagement.

📊 Highest Weeks in Top 10 – Through My Window (2020-2023)

THROUGH MY WINDOW: A LONG RUN WITH QUESTIONABLE VIEWERSHIP

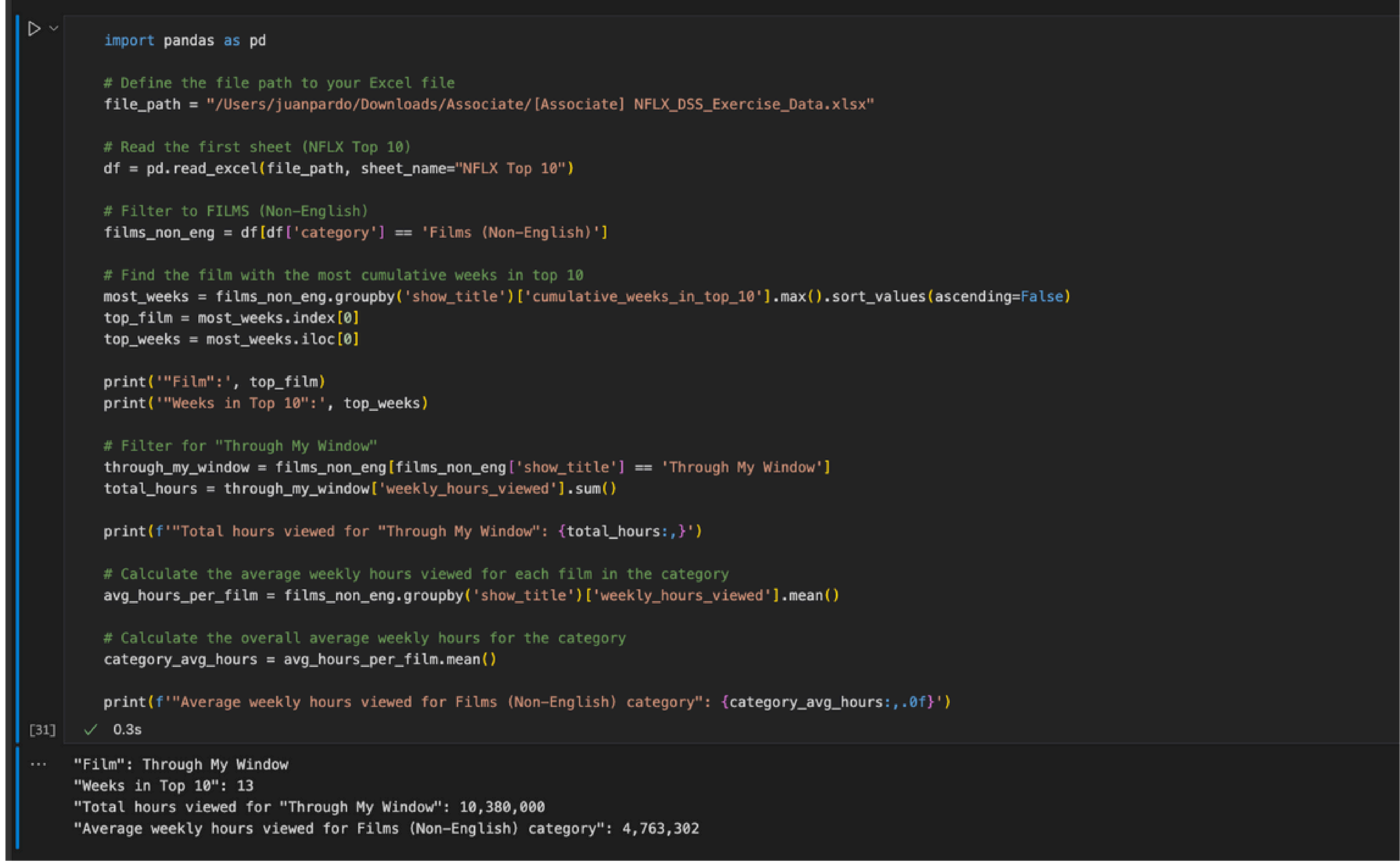
Through My Window led with 13 weeks in the Top 10, but its viewership raises questions.

LONGEST TOP 10 RUN, LOW TOTAL HOURS

Averaged 798,462 weekly hours (10.38M total), well below the 4,763,302 category average, possibly due to errors (0 hours on 2022-05-24).

ESTIMATED USERS: 6.92M

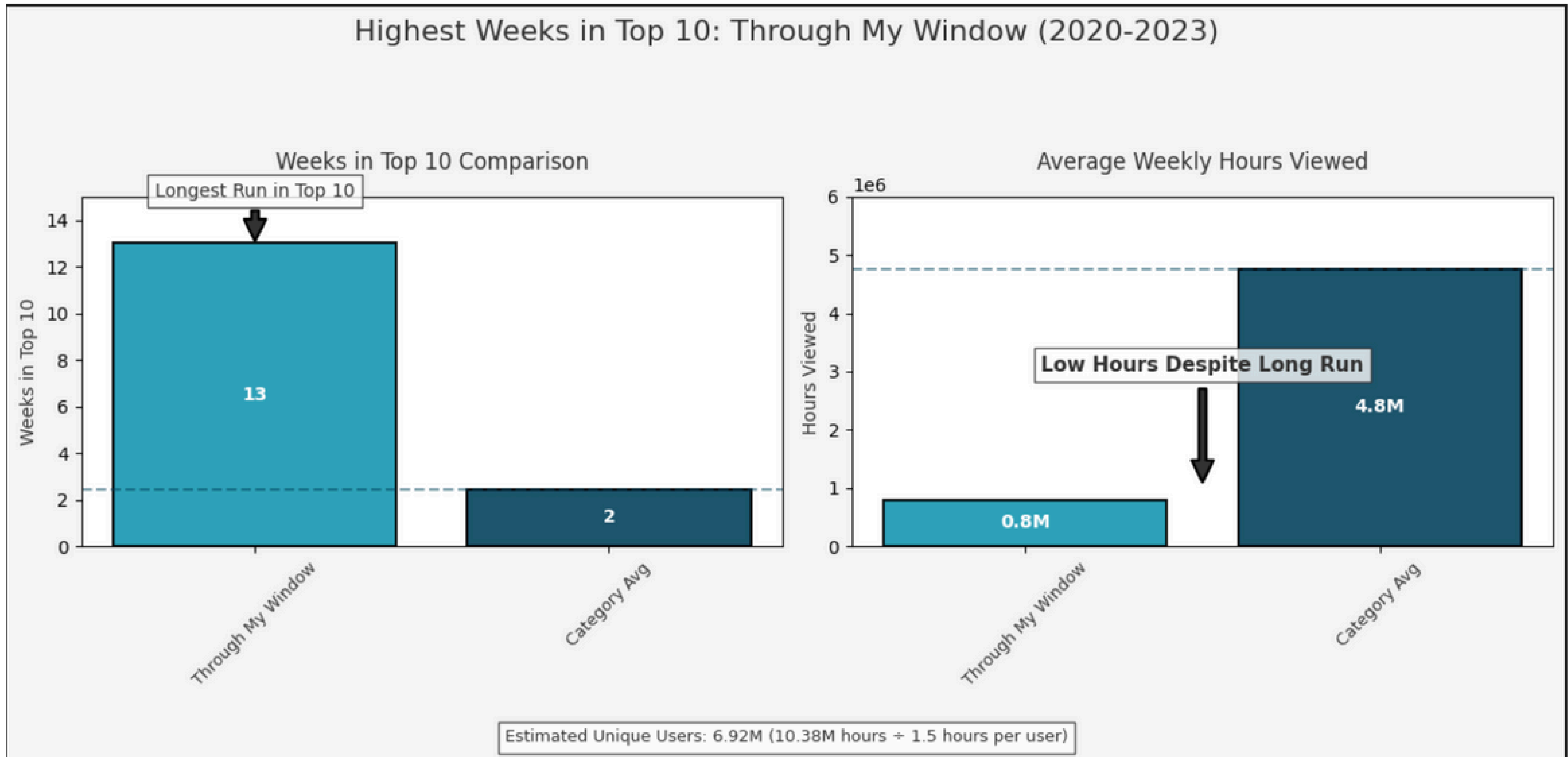
Assumed 1.5 hours per user (below 111-minute runtime for incomplete views), yielding 6.92M unique users (10.38M ÷ 1.5), ignoring rewatches.



```
import pandas as pd

# Define the file path to your Excel file
file_path = "/Users/juanpardo/Downloads/Associate/[Associate] NFLX_DSS_Exercise_Data.xlsx"

# Read the first sheet (NFLX Top 10)
df = pd.read_excel(file_path, sheet_name="NFLX Top 10")

# Filter to FILMS (Non-English)
films_non_eng = df[df['category'] == 'Films (Non-English)']

# Find the film with the most cumulative weeks in top 10
most_weeks = films_non_eng.groupby('show_title')['cumulative_weeks_in_top_10'].max().sort_values(ascending=False)
top_film = most_weeks.index[0]
top_weeks = most_weeks.iloc[0]

print('"Film":', top_film)
print('"Weeks in Top 10":', top_weeks)

# Filter for "Through My Window"
through_my_window = films_non_eng[films_non_eng['show_title'] == 'Through My Window']
total_hours = through_my_window['weekly_hours_viewed'].sum()

print(f'"Total hours viewed for "Through My Window": {total_hours:,}')

# Calculate the average weekly hours viewed for each film in the category
avg_hours_per_film = films_non_eng.groupby('show_title')['weekly_hours_viewed'].mean()

# Calculate the overall average weekly hours for the category
category_avg_hours = avg_hours_per_film.mean()

print(f'"Average weekly hours viewed for Films (Non-English) category": {category_avg_hours:,.0f}')
```

[31] ✓ 0.3s

```
"Film": Through My Window
"Weeks in Top 10": 13
"Total hours viewed for "Through My Window": 10,380,000
"Average weekly hours viewed for Films (Non-English) category": 4,763,302
```

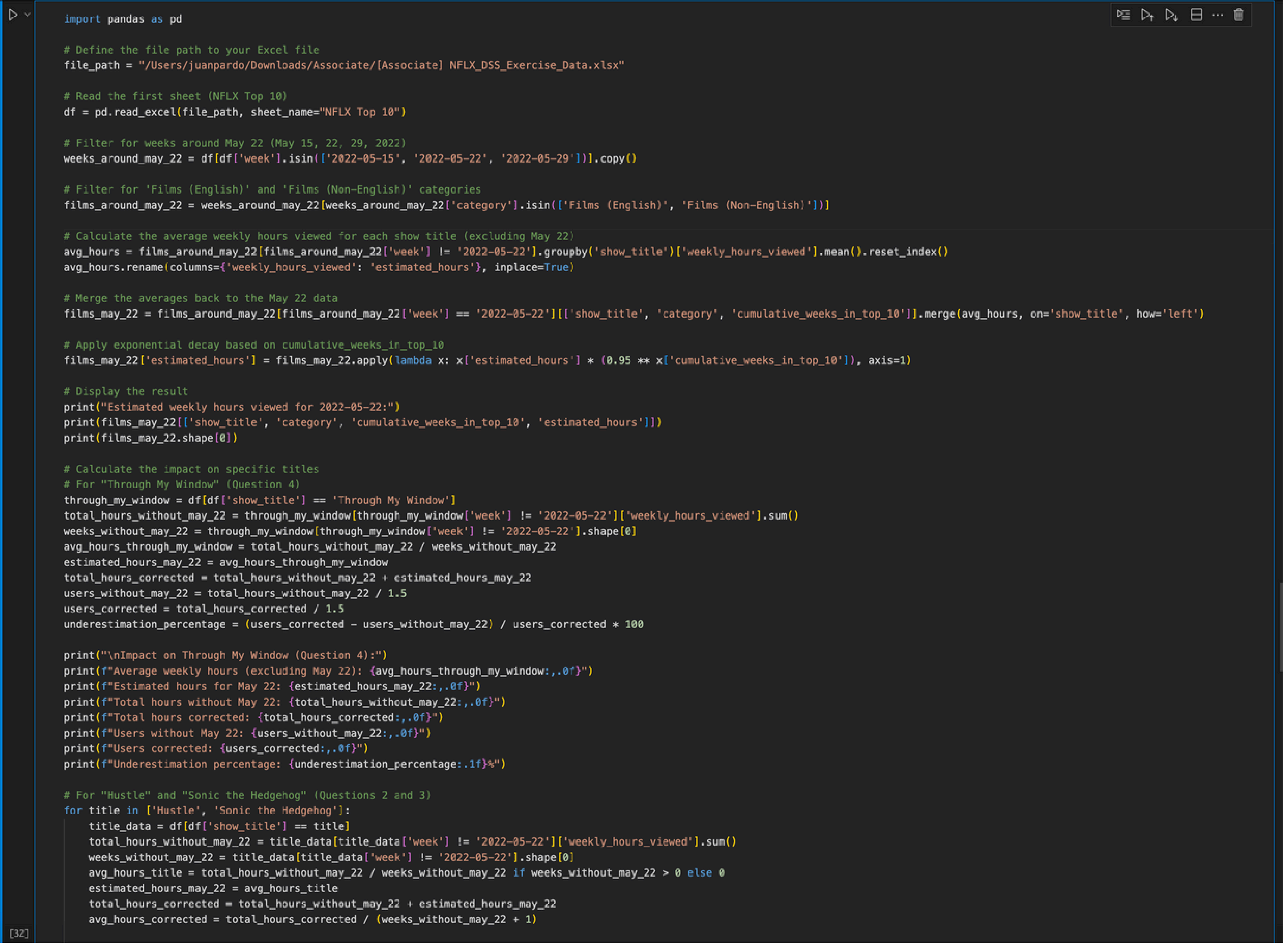


Highest Weeks in Top 10: Through My Window (2020-2023)

Estimated Unique Users: 6.92M (10.38M hours ÷ 1.5 hours per user)

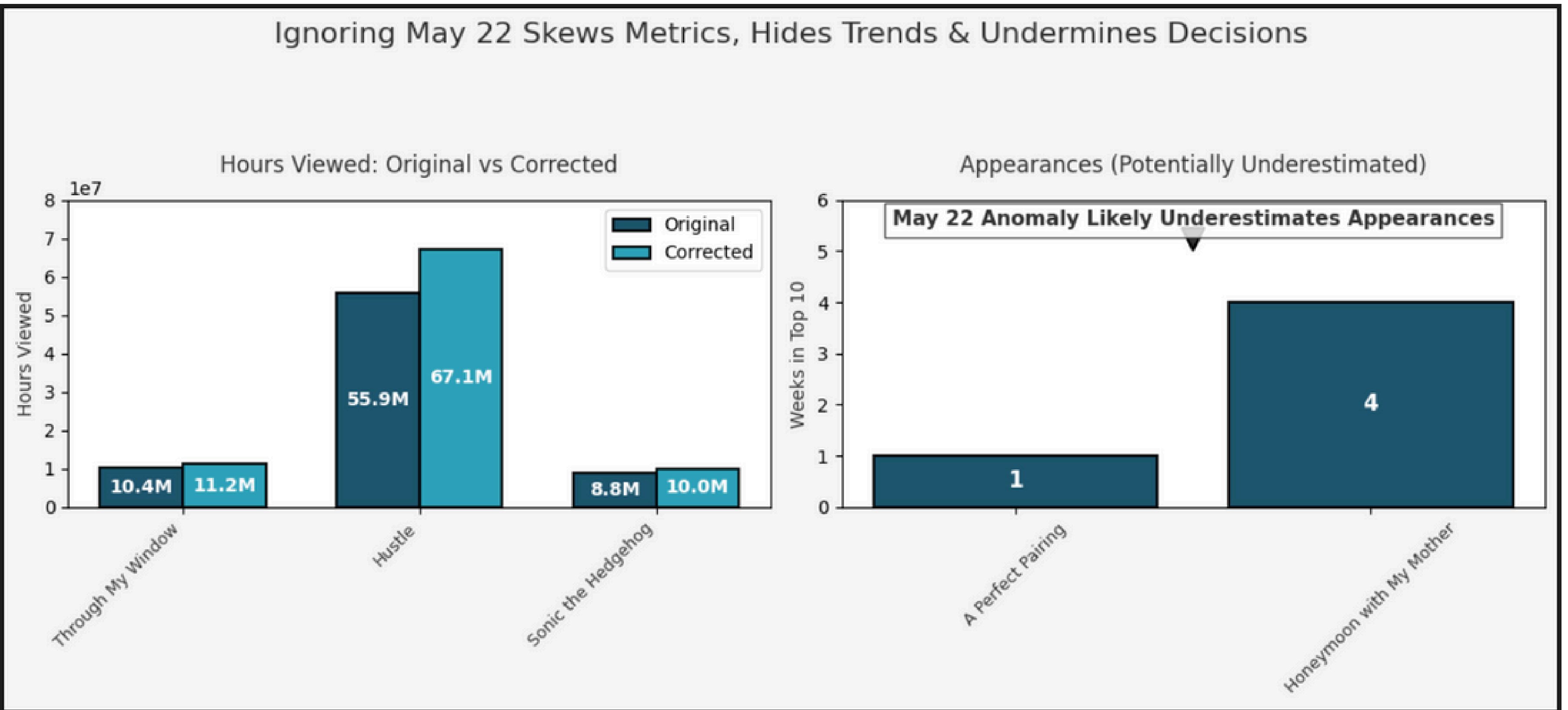
Ignoring May 22 Skews Metrics, Hides Trends & Undermines Decisions

An anomaly on the week of May 22 (20 titles showing 0 viewing hours) creates significant distortions in viewership and key metrics.



```
import pandas as pd

# Define the file path to your Excel file
file_path = "/Users/juanpardo/Downloads/Associate/[Associate] NFLX_DSS_Exercise_Data.xlsx"

# Read the first sheet (NFLX Top 10)
df = pd.read_excel(file_path, sheet_name="NFLX Top 10")

# Filter for weeks around May 22 (May 15, 22, 29, 2022)
weeks_around_may_22 = df[df['week'].isin(['2022-05-15', '2022-05-22', '2022-05-29'])].copy()

# Filter for 'Films (English)' and 'Films (Non-English)' categories
films_around_may_22 = weeks_around_may_22[weeks_around_may_22['category'].isin(['Films (English)', 'Films (Non-English)'])]

# Calculate the average weekly hours viewed for each show title (excluding May 22)
avg_hours = films_around_may_22[films_around_may_22['week'] != '2022-05-22'].groupby('show_title')['weekly_hours_viewed'].mean().reset_index()
avg_hours.rename(columns={'weekly_hours_viewed': 'estimated_hours'}, inplace=True)

# Merge the averages back to the May 22 data
films_may_22 = films_around_may_22[films_around_may_22['week'] == '2022-05-22'][['show_title', 'category', 'cumulative_weeks_in_top_10']].merge(avg_hours, on='show_title', how='left')

# Apply exponential decay based on cumulative_weeks_in_top_10
films_may_22['estimated_hours'] = films_may_22.apply(lambda x: x['estimated_hours'] * (0.95 ** x['cumulative_weeks_in_top_10']), axis=1)

# Display the result
print("Estimated weekly hours viewed for 2022-05-22:")
print(films_may_22[['show_title', 'category', 'cumulative_weeks_in_top_10', 'estimated_hours']])
print(films_may_22.shape[0])

# Calculate the impact on specific titles
# For "Through My Window" (Question 4)
through_my_window = df[df['show_title'] == 'Through My Window']
total_hours_without_may_22 = through_my_window[through_my_window['week'] != '2022-05-22']['weekly_hours_viewed'].sum()
weeks_without_may_22 = through_my_window[through_my_window['week'] != '2022-05-22'].shape[0]
avg_hours_through_my_window = total_hours_without_may_22 / weeks_without_may_22
estimated_hours_may_22 = avg_hours_through_my_window
total_hours_corrected = total_hours_without_may_22 + estimated_hours_may_22
users_without_may_22 = total_hours_without_may_22 / 1.5
users_corrected = total_hours_corrected / 1.5
underestimation_percentage = (users_corrected - users_without_may_22) / users_corrected * 100

print("\nImpact on Through My Window (Question 4):")
print(f"Average weekly hours (excluding May 22): {avg_hours_through_my_window:,.0f}")
print(f"Estimated hours for May 22: {estimated_hours_may_22:,.0f}")
print(f"Total hours without May 22: {total_hours_without_may_22:,.0f}")
print(f"Total hours corrected: {total_hours_corrected:,.0f}")
print(f"Users without May 22: {users_without_may_22:,.0f}")
print(f"Users corrected: {users_corrected:,.0f}")
print(f"Underestimation percentage: {underestimation_percentage:.1f}%")

# For "Hustle" and "Sonic the Hedgehog" (Questions 2 and 3)
for title in ['Hustle', 'Sonic the Hedgehog']:
    title_data = df[df['show_title'] == title]
    total_hours_without_may_22 = title_data[title_data['week'] != '2022-05-22']['weekly_hours_viewed'].sum()
    weeks_without_may_22 = title_data[title_data['week'] != '2022-05-22'].shape[0]
    avg_hours_title = total_hours_without_may_22 / weeks_without_may_22 if weeks_without_may_22 > 0 else 0
    estimated_hours_may_22 = avg_hours_title
    total_hours_corrected = total_hours_without_may_22 + estimated_hours_may_22
    avg_hours_corrected = total_hours_corrected / (weeks_without_may_22 + 1)
```

[32]



Ignoring May 22 Skews Metrics, Hides Trends & Undermines Decisions

Hours Viewed: Original vs Corrected

Appearances (Potentially Underestimated)

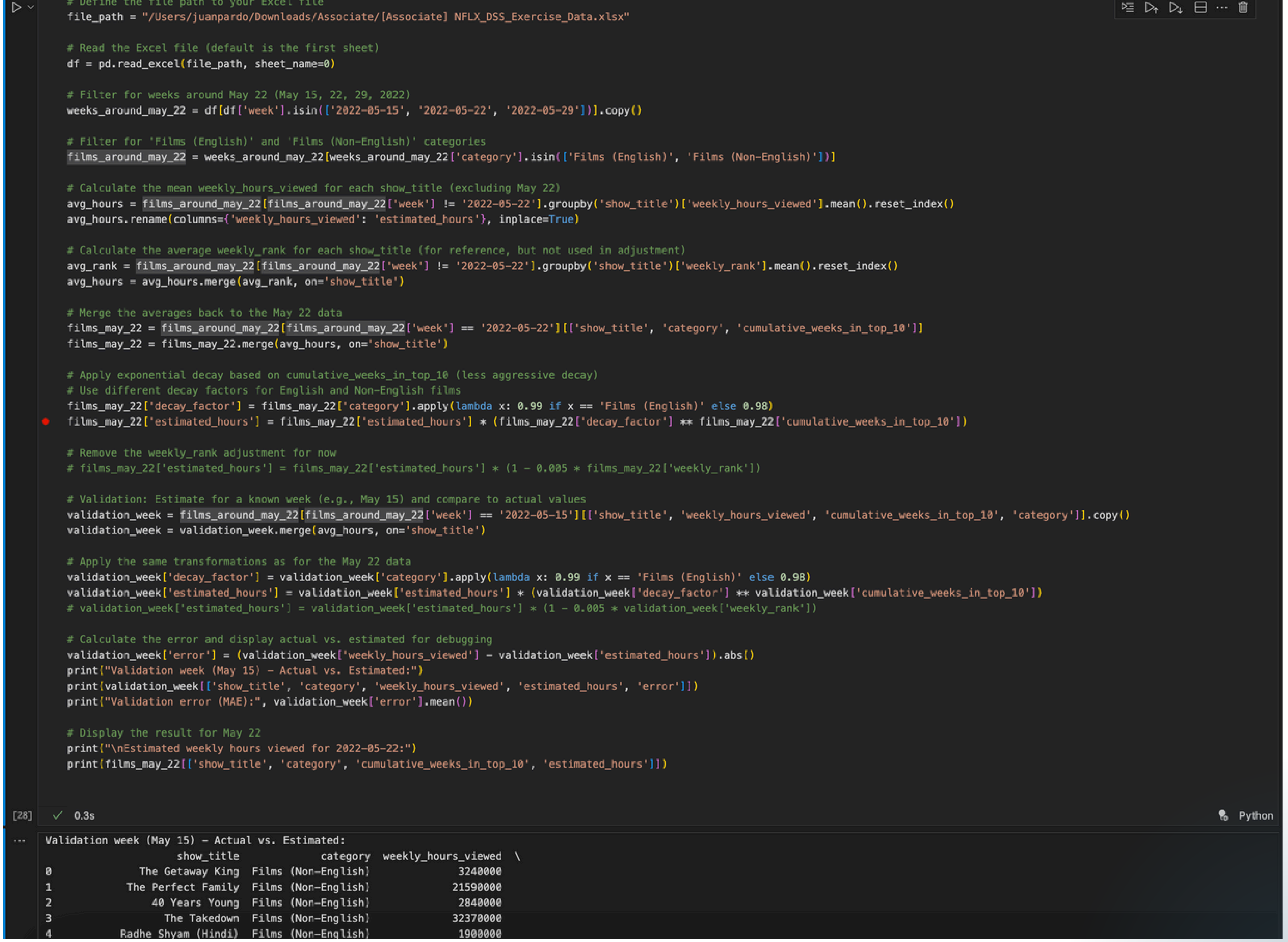
May 22 Anomaly Likely Underestimates Appearances

METRIC DISTORTION

- Undercounts appearances ("A Perfect Pairing": 1 week, "Honeymoon with My Mother": 4 weeks).
- Lowers hours for "Hustle" (55.94M, 5 appearances), "Sonic the Hedgehog" (8.85M, 8 appearances), and "Through My Window" (865K, 7.7% user underestimation to 7.5M).

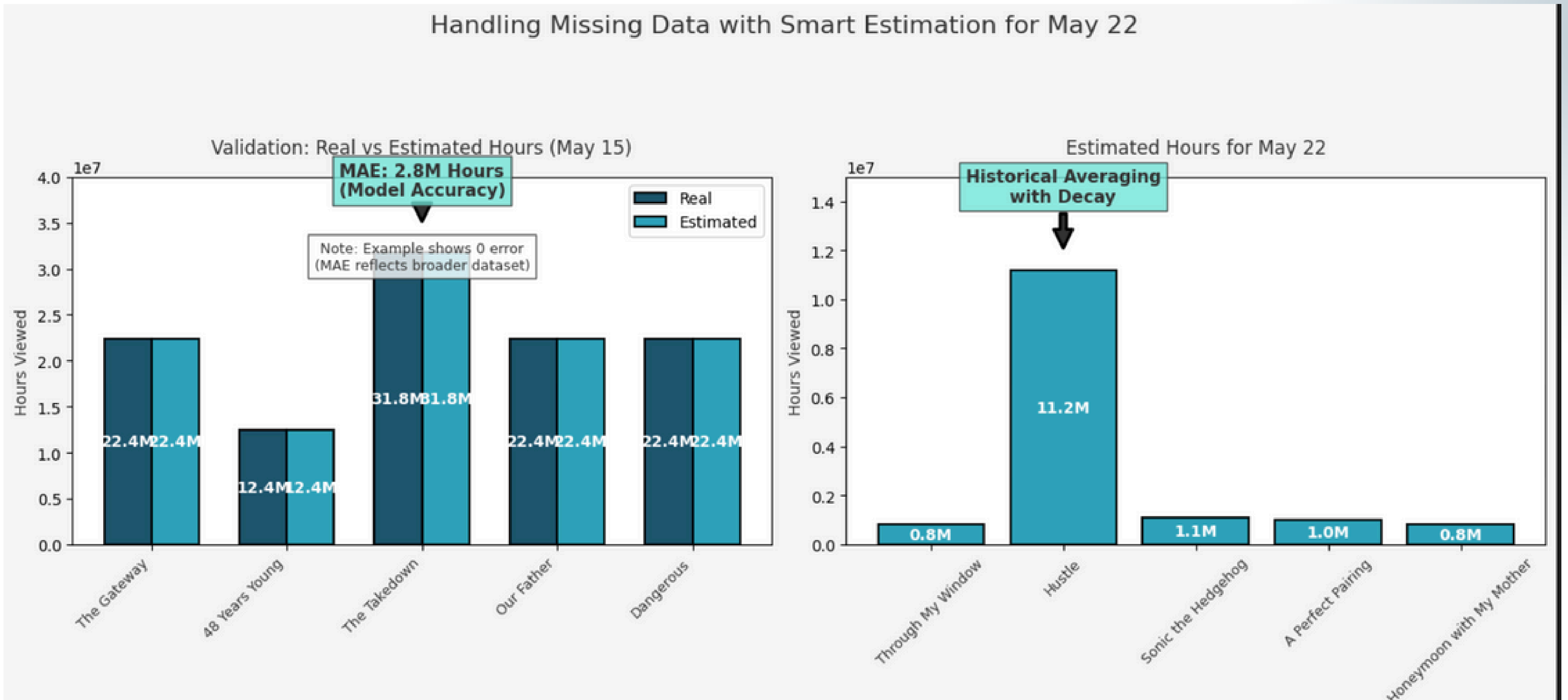
STRATEGIC RISK

- Undercounts appearances ("A Perfect Pairing": 1 week, "Honeymoon with My Mother": 4 weeks).
- Lowers hours for "Hustle" (55.94M, 5 appearances), "Sonic the Hedgehog" (8.85M, 8 appearances), and "Through My Window" (865K, 7.7% user underestimation to 7.5M).



```
# Define the file path to your Excel file
file_path = "/Users/juanpardo/Downloads/Associate/[Associate] NFLX_DSS_Exercise_Data.xlsx"

# Read the Excel file (default is the first sheet)
df = pd.read_excel(file_path, sheet_name=0)

# Filter for weeks around May 22 (May 15, 22, 29, 2022)
weeks_around_may_22 = df[df['week'].isin(['2022-05-15', '2022-05-22', '2022-05-29'])].copy()

# Filter for 'Films (English)' and 'Films (Non-English)' categories
films_around_may_22 = weeks_around_may_22[weeks_around_may_22['category'].isin(['Films (English)', 'Films (Non-English)'])]

# Calculate the mean weekly_hours_viewed for each show_title (excluding May 22)
avg_hours = films_around_may_22[films_around_may_22['week'] != '2022-05-22'].groupby('show_title')['weekly_hours_viewed'].mean().reset_index()
avg_hours.rename(columns={'weekly_hours_viewed': 'estimated_hours'}, inplace=True)

# Calculate the average weekly_rank for each show_title (for reference, but not used in adjustment)
avg_rank = films_around_may_22[films_around_may_22['week'] != '2022-05-22'].groupby('show_title')['weekly_rank'].mean().reset_index()
avg_hours = avg_hours.merge(avg_rank, on='show_title')

# Merge the averages back to the May 22 data
films_may_22 = films_around_may_22[films_around_may_22['week'] == '2022-05-22'][['show_title', 'category', 'cumulative_weeks_in_top_10']]
films_may_22 = films_may_22.merge(avg_hours, on='show_title')

# Apply exponential decay based on cumulative_weeks_in_top_10 (less aggressive decay)
# Use different decay factors for English and Non-English films
films_may_22['decay_factor'] = films_may_22['category'].apply(lambda x: 0.99 if x == 'Films (English)' else 0.98)
films_may_22['estimated_hours'] = films_may_22['estimated_hours'] * (films_may_22['decay_factor'] ** films_may_22['cumulative_weeks_in_top_10'])

# Remove the weekly_rank adjustment for now
# films_may_22['estimated_hours'] = films_may_22['estimated_hours'] * (1 - 0.005 * films_may_22['weekly_rank'])

# Validation: Estimate for a known week (e.g., May 15) and compare to actual values
validation_week = films_around_may_22[films_around_may_22['week'] == '2022-05-15'][['show_title', 'weekly_hours_viewed', 'cumulative_weeks_in_top_10', 'category']].copy()
validation_week = validation_week.merge(avg_hours, on='show_title')

# Apply the same transformations as for the May 22 data
validation_week['decay_factor'] = validation_week['category'].apply(lambda x: 0.99 if x == 'Films (English)' else 0.98)
validation_week['estimated_hours'] = validation_week['estimated_hours'] * (validation_week['decay_factor'] ** validation_week['cumulative_weeks_in_top_10'])
# validation_week['estimated_hours'] = validation_week['estimated_hours'] * (1 - 0.005 * validation_week['weekly_rank'])

# Calculate the error and display actual vs. estimated for debugging
validation_week['error'] = (validation_week['weekly_hours_viewed'] - validation_week['estimated_hours']).abs()
print("Validation week (May 15) - Actual vs. Estimated:")
print(validation_week[['show_title', 'category', 'weekly_hours_viewed', 'estimated_hours', 'error']])
print("Validation error (MAE):", validation_week['error'].mean())

# Display the result for May 22
print("\nEstimated weekly hours viewed for 2022-05-22:")
print(films_may_22[['show_title', 'category', 'cumulative_weeks_in_top_10', 'estimated_hours']])
```

```
Validation week (May 15) - Actual vs. Estimated:
                show_title             category  weekly_hours_viewed  \
0           The Getaway King  Films (Non-English)              3240000
1          The Perfect Family  Films (Non-English)             21590000
2             40 Years Young  Films (Non-English)              2840000
3               The Takedown  Films (Non-English)             32370000
4        Radhe Shyam (Hindi)  Films (Non-English)              1900000
```



Handling Missing Data with Smart Estimation for May 22

HISTORICAL AVERAGING AND DECAY MODELING

We used data from May 15 and May 29 to compute the average weekly hours viewed per title (excluding May 22). Then, we adjusted these averages using exponential decay based on each title's cumulative weeks in the top 10, with tailored decay rates for English (0.99) and Non-English (0.98) films to reflect differences in content longevity.

VALIDATION WITH MAE FOR MODEL ACCURACY

We validated this method using May 15 as a control week, comparing actual vs. estimated hours. The result: a Mean Absolute Error (MAE) of 2.8M hours—indicating moderate but actionable accuracy. This validation step adds confidence to the estimates produced for the missing May 22 data.

Net Positive Outlook with Data Gaps to Address

Despite some anomalies, the report signals consistent subscriber growth and strong engagement trends—backed by data-driven evidence. However, resolving reporting inconsistencies is essential for accurate insights and confident strategic decisions.

SUSTAINED SUBSCRIPTIONS 01.

From 2019 to 2024, Netflix maintained a strong net-positive trajectory, with spikes in June 2020, June 2022, and September 2023. These peaks signal effective content and platform appeal across time.

CONTENT ENGAGEMENT OVER RATINGS 02.

Titles like Chickenhare achieved 14.8M average weekly hours despite a 0.0 IMDb rating, showing that user engagement may be more influenced by recommendation engines and marketing than by reviews.

REPORTING ANOMALIES CREATE STRATEGIC RISK 03.

The May 22 and Dec 2024 anomalies—where viewership and subscriber data dropped to near-zero—could mislead decisions in content investment, performance tracking, or retention planning if left uncorrected.