Netflix Viewing Patterns Analysis

Data Preparation & Aggregations

These steps load the Netflix data, clean and merge tables, convert duration strings to seconds, and build summary tables that each graph will use. Nothing is plotted yet the code just prepares the data.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

devices_df = pd.read_csv('netflix-report/ACCOUNT/AccessAndDevices.csv')
clickstream_df = pd.read_csv('netflix-report/CLICKSTREAM/Clickstream.csv')
viewing_df = pd.read_csv('netflix-report/CONTENT_INTERACTION/ViewingActivity.csv', on_bad_lines='skip')
profiles_df = pd.read_csv('netflix-report/PROFILES/Profiles.csv')
device_meta_df = pd.read_csv('netflix-report/DEVICES/Devices.csv')
```

Data Preparation & Aggregations - Merging & Preprocessing Part 1/3

```
viewing_df = viewing_df[['Profile Name', 'Start Time', 'Duration', 'Device Type', 'Title']]
clickstream_df = clickstream_df[['Profile Name', 'Click Utc Ts', 'Webpage Url']]

viewing_df['Start Time'] = pd.to_datetime(viewing_df['Start Time'])
clickstream_df['Click Utc Ts'] = pd.to_datetime(clickstream_df['Click Utc Ts'])

devices_df['Date'] = pd.to_datetime(devices_df['Date'])

merged = viewing_df.merge(profiles_df[['Profile Name', 'Date Of Birth']], on='Profile Name', how='left')

merged = merged.merge(device_meta_df[['Profile Name', 'Device Type', 'Profile First Playback Date']],
                      on=['Profile Name', 'Device Type'], how='left')

clickstream_df = clickstream_df.sort_values(['Profile Name', 'Click Utc Ts'])
merged = merged.sort_values(['Profile Name', 'Start Time'])

merged['hour'] = merged['Start Time'].dt.hour
merged['day_of_week'] = merged['Start Time'].dt.day_name()
merged['month'] = merged['Start Time'].dt.month_name()
merged['is_weekend'] = merged['day_of_week'].isin(['Saturday', 'Sunday'])
```

Data Preparation & Aggregations - Merging & Preprocessing Part 2/3

```
def to_seconds(x):
    try:
        h, m, s = x.split(':')
        return int(h)*3600 + int(m)*60 + int(s)
    except:
        return np.nan

merged['duration_sec'] = merged['Duration'].astype(str).apply(to_seconds)

merged = merged.dropna(subset=['Start Time', 'Duration', 'duration_sec'])


hourly = merged.groupby('hour')['duration_sec'].sum().reset_index()

user_hourly = merged.groupby(['Profile Name', 'hour'])['duration_sec'].sum().reset_index()

device_hourly = merged.groupby(['Device Type', 'hour'])['duration_sec'].sum().reset_index()

weekagg = merged.groupby('is_weekend')['duration_sec'].agg(['sum','mean','count']).reset_index()
```

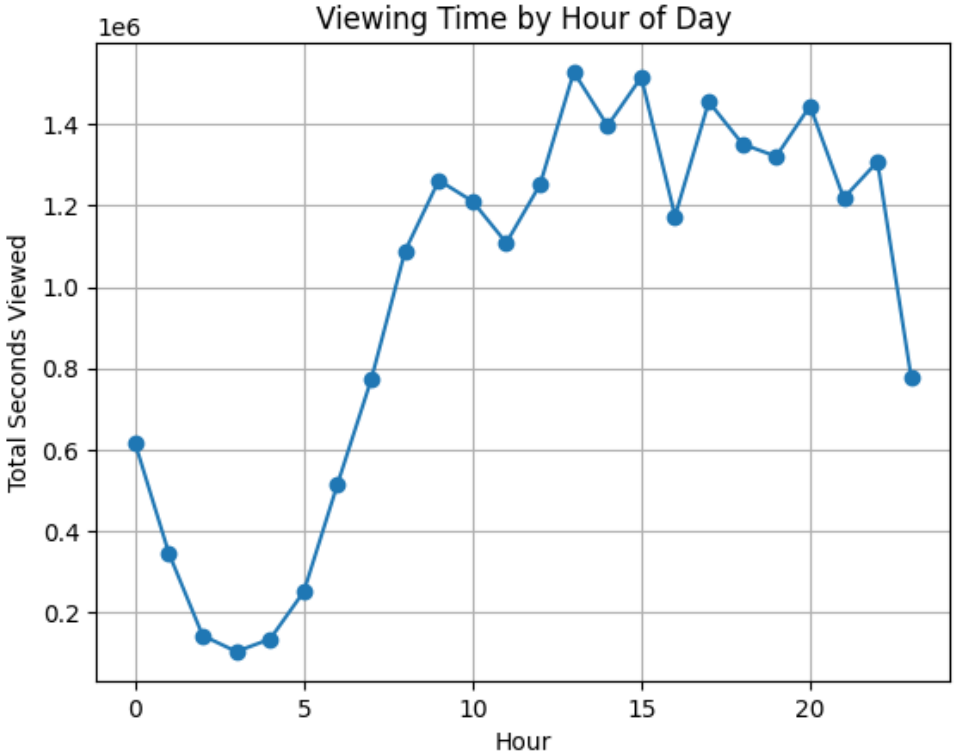
Data Preparation & Aggregations - Merging & Preprocessing Part 3/3

```
monthly = merged.groupby('month')['duration_sec'].sum().reindex(

['January','February','March','April','May','June','July','August','September','October','Novembe
r','December']

).dropna().reset_index()

profile_device = merged.groupby(['Profile Name','Device
Type'])['duration_sec'].sum().reset_index()

user_totals = merged.groupby('Profile Name')['duration_sec'].sum().reset_index()

device_totals = merged.groupby('Device Type')['duration_sec'].sum().reset_index()
```

Viewing Time by Hour of Day

```
plt.figure()
plt.plot(hourly['hour'], hourly['duration_sec'], marker='o')
plt.title('Viewing Time by Hour of Day')
plt.xlabel('Hour')
plt.ylabel('Total Seconds Viewed')
plt.grid(True)
plt.show()
```

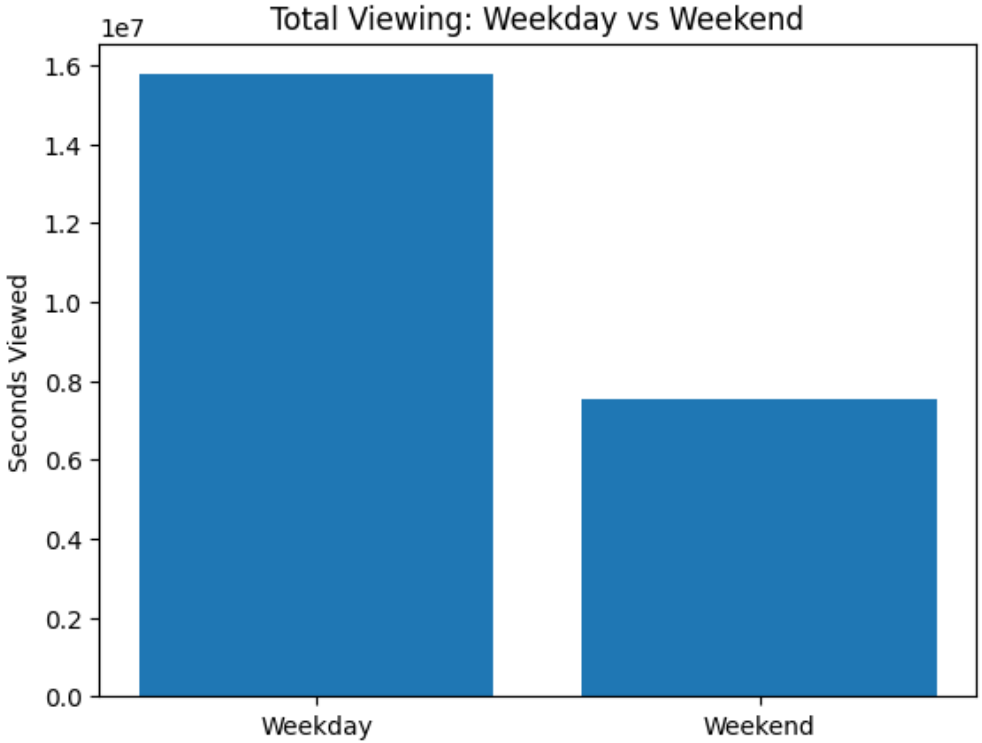
Viewing Time by Hour of Day



Total Viewing: Weekday vs Weekend

```
plt.figure()
plt.bar(['Weekday','Weekend'], weekagg['sum'])
plt.title('Total Viewing: Weekday vs Weekend')
plt.ylabel('Seconds Viewed')
plt.show()
```

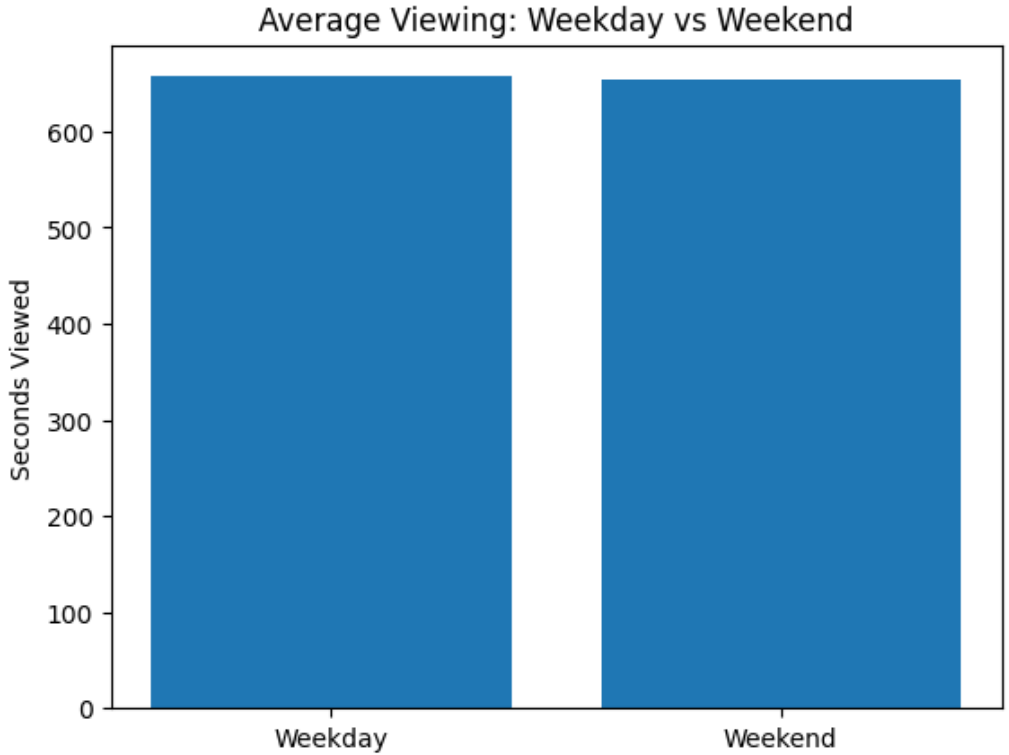
Total Viewing: Weekday vs Weekend



Average Viewing per Day: Weekday vs Weekend

```
plt.figure()
plt.bar(['Weekday','Weekend'], weekagg['mean'])
plt.title('Average Viewing: Weekday vs Weekend')
plt.ylabel('Seconds Viewed')
plt.show()
```

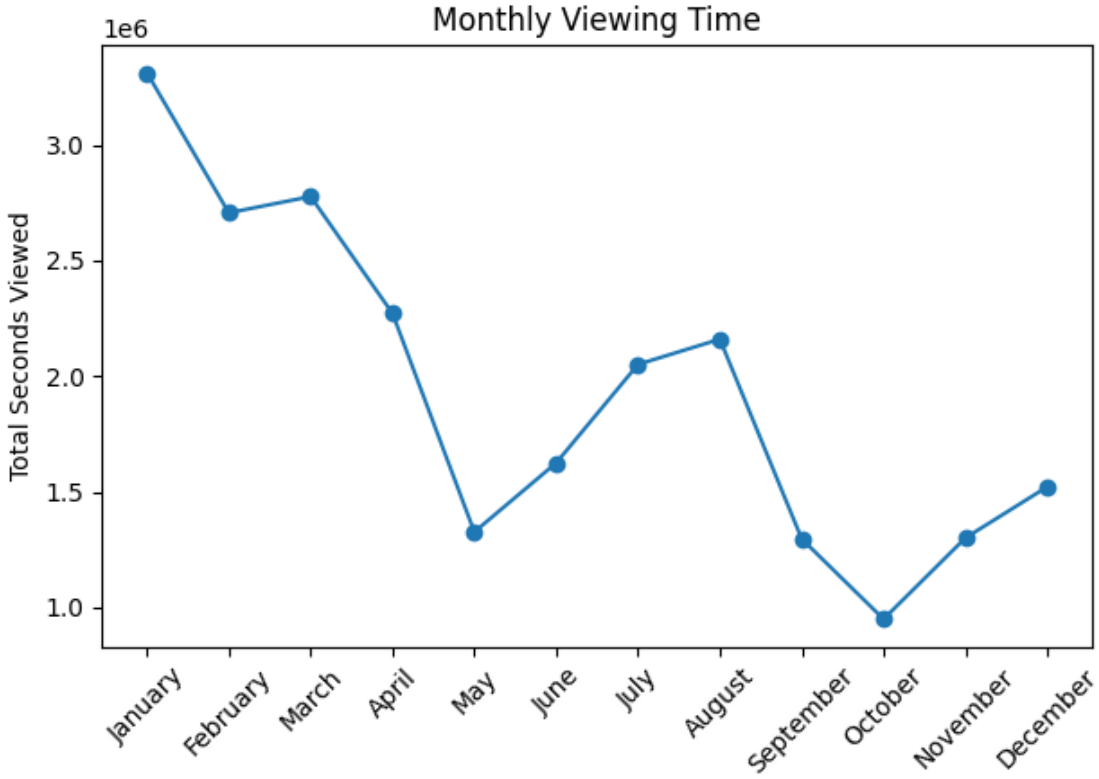
Average Viewing per Day: Weekday vs Weekend



Monthly Viewing Time

```
plt.figure()
plt.plot(monthly['month'], monthly['duration_sec'], marker='o')
plt.title('Monthly Viewing Time')
plt.xlabel('Month')
plt.ylabel('Total Seconds Viewed')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

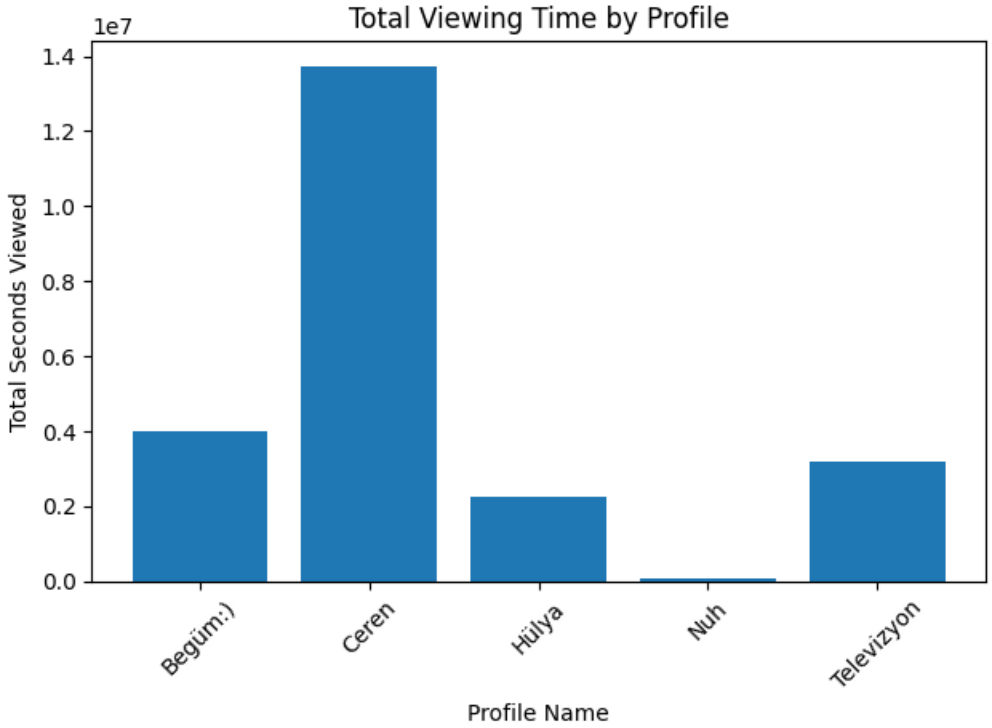
Monthly Viewing Time



Total Viewing Time by Profile

```
plt.figure()
plt.bar(
    user_totals['Profile Name'],
    user_totals['duration_sec']
)
plt.title('Total Viewing Time by Profile')
plt.xlabel('Profile Name')
plt.ylabel('Total Seconds Viewed')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

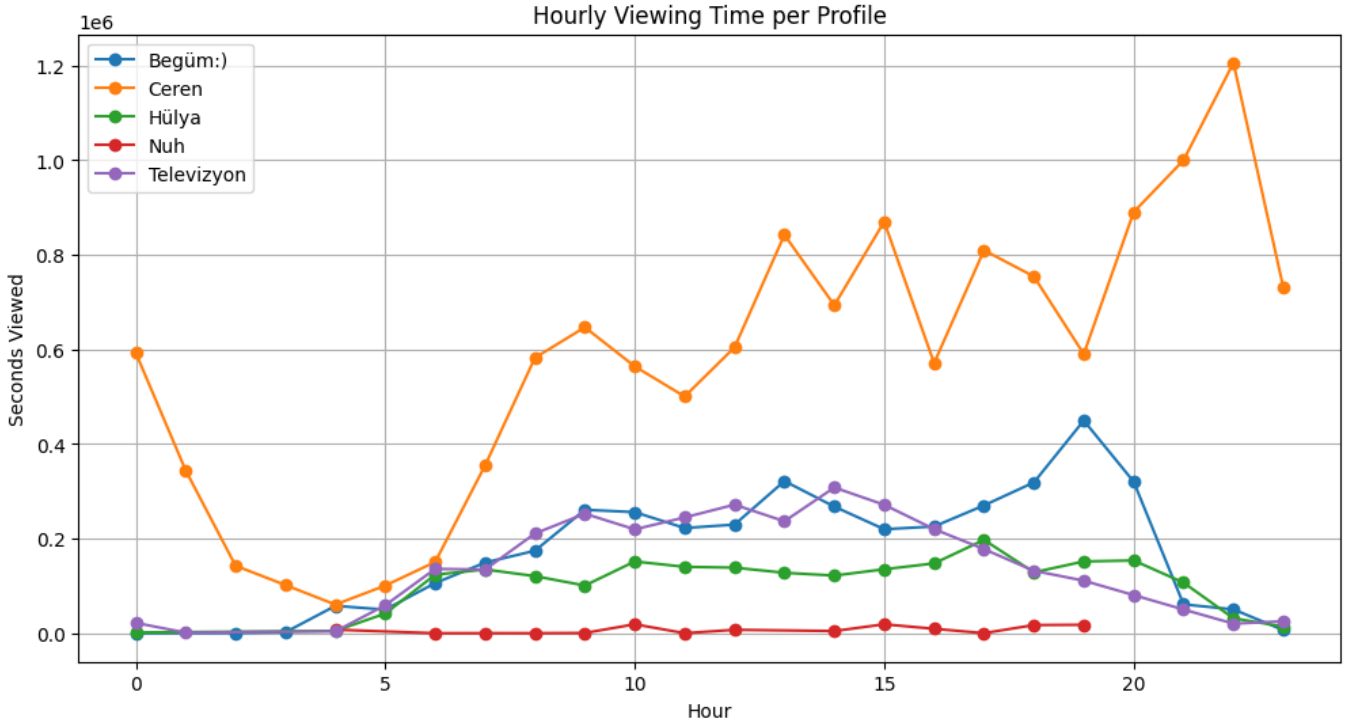
Total Viewing Time by Profile



Hourly Viewing Time per Profile

```
plt.figure(figsize=(12,6))
for name, grp in user_hourly.groupby('Profile Name'):
    plt.plot(grp['hour'], grp['duration_sec'], marker='o', label=name)
plt.title('Hourly Viewing Time per Profile')
plt.xlabel('Hour')
plt.ylabel('Seconds Viewed')
plt.legend()
plt.grid(True)
plt.show()
```

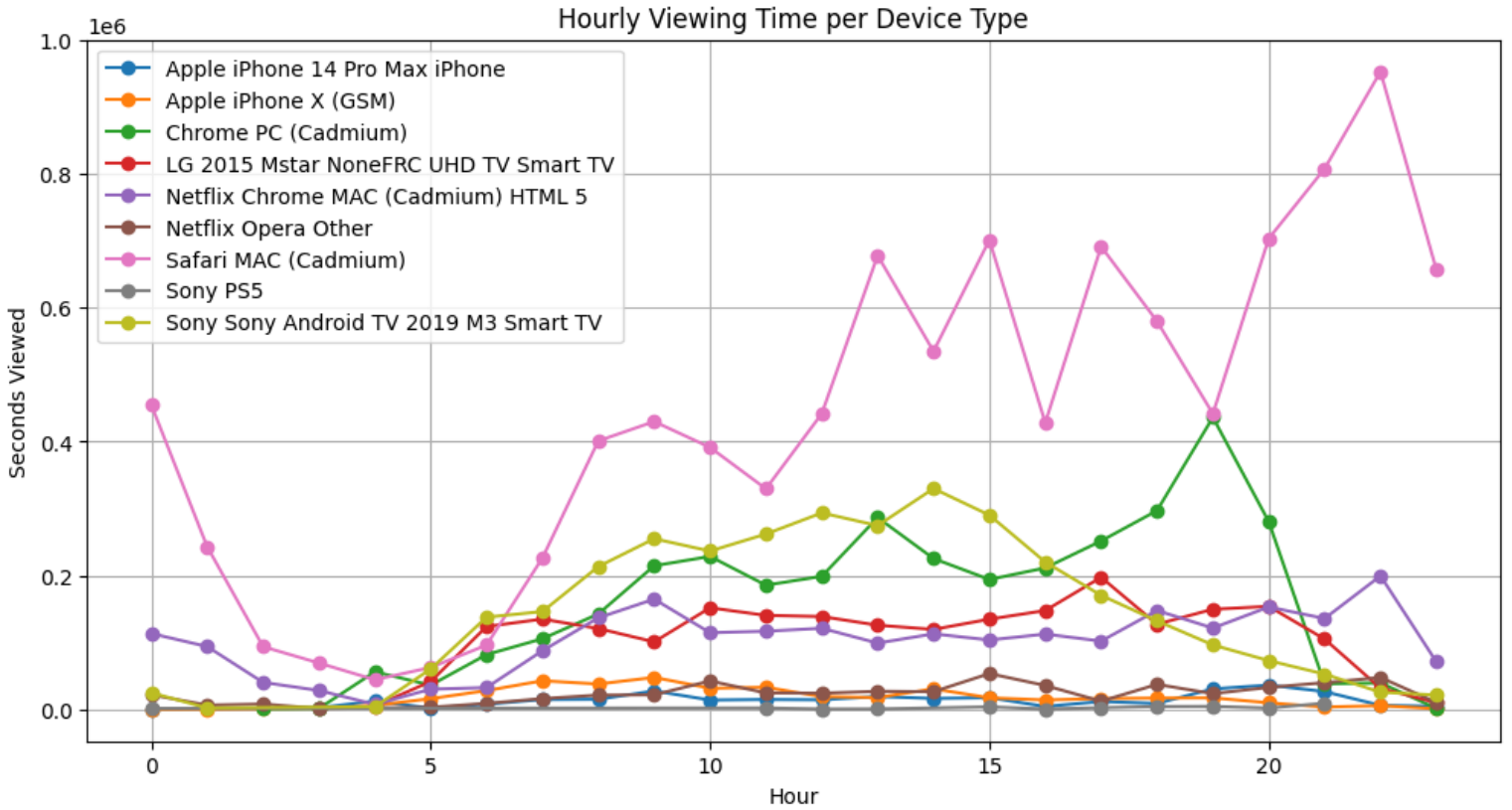
Hourly Viewing Time per Profile



Hourly Viewing Time per Device Type

```
plt.figure(figsize=(12,6))
for dev, grp in device_hourly.groupby('Device Type'):
    plt.plot(grp['hour'], grp['duration_sec'], marker='o', label=dev)
plt.title('Hourly Viewing Time per Device Type')
plt.xlabel('Hour')
plt.ylabel('Seconds Viewed')
plt.legend()
plt.grid(True)
plt.show()
```

Hourly Viewing Time per Device Type



Hypothesis Testing

```
from scipy.stats import ttest_rel

pivoted = user_week_part.pivot(index='Profile Name', columns='is_weekend', values='duration_sec')

pivoted.columns = ['weekday_hours', 'weekend_hours']

pivoted = pivoted.dropna()

pivoted = pivoted / 3600

weekday_hours = pivoted['weekday_hours'].tolist()
weekend_hours = pivoted['weekend_hours'].tolist()

t_stat, p_value_two_sided = ttest_rel(weekend_hours, weekday_hours)

if t_stat > 0:
    p_value = p_value_two_sided / 2
else:
    p_value = 1 - (p_value_two_sided / 2)

print("p-value:", p_value)
```

Hypothesis Testing

```
winter_months = ['December', 'January', 'February']
summer_months = ['June', 'July', 'August']

merged['season'] = merged['month'].apply(
    lambda x: 'winter' if x in winter_months else ('summer' if x in summer_months else 'other')
)

user_season = merged[merged['season'].isin(['winter', 'summer'])].groupby(
    ['Profile Name', 'season']
)['duration_sec'].mean().reset_index()

pivoted = user_season.pivot(index='Profile Name', columns='season', values='duration_sec')
pivoted = pivoted.dropna()
pivoted = pivoted / 3600

from scipy.stats import ttest_rel
import numpy as np

winter_hours = pivoted['winter'].tolist()
summer_hours = pivoted['summer'].tolist()

t_stat, p_val_two_sided = ttest_rel(winter_hours, summer_hours)

# One-sided test: you expect winter > summer
if t_stat > 0:
    p_value = p_val_two_sided / 2
else:
    p_value = 1 - (p_val_two_sided / 2)

print("p-value:", p_value)
```

Hypothesis Testing

- The hypothesis tests for both increased weekend viewing and increased winter viewing did not yield statistically significant results. The p-value for weekend vs. weekday viewing was approximately 0.186, and for winter vs. summer viewing, it was about 0.161 — both above the commonly used significance threshold of 0.05. This means that, based on the data, we do not have strong enough evidence to conclude that users watch significantly more on weekends or during winter months. While there may be a trend toward increased viewing during these periods, the observed differences could also be due to random variation in user behavior.

Conclusion

The analysis broadly supports these ideas:

Hourly pattern: Viewing is minimal overnight, climbs sharply after 6 a.m., and shows two clear surges— a midday bump (around 13:00) and a stronger evening peak (18:00-23:00).

Weekday vs weekend: Total seconds are higher on weekdays simply because there are five of them, but the average per day is almost identical, confirming similar engagement once day-count is controlled.

Seasonality: Winter and early-spring months (January–March) dominate; activity dips in late spring and early autumn, then rises again in November–December, matching the “more TV in colder months” expectation.

Device & profile insights: A single Mac-Safari setup drives most viewing, and one profile (“Ceren”) accounts for almost half of all screen time, suggesting heavy individual influence on household patterns.