oliverp-Analyzing-Website-Performance-Grammys

September 11, 2025

1 Project | Analyzing Website Performance for The Grammys

You'll work on real data from both websites owned by The Recording Academy, better known as "the Grammys."

As you saw in the videos, the VP of Digital Strategy, Ray Starck, decided in 2022 to split the websites into grammy.comLinks to an external site. and recordingacademy.comLinks to an external site. to better serve the Recording Academy's various audience needs. You're tasked with examining the impact of splitting up the two websites, and analyzing the data for a better understanding of trends and audience behavior.

1.1 Data Dictionary

You'll be working with two files, `grammys_live_web_analytics.csv` and `ra_live_web_analytics.csv`.

These files will contain the following information:

- **date** - The date the data was confirmed. It is in `yyyy-mm-dd` format.
- **visitors** - The number of users who went on the website on that day.
- **pageviews** - The number of pages that all users viewed on the website.
- **sessions** - The total number of sessions on the website. A session is a group of user interactions with your website that take place within a given time frame. For example a single session can contain multiple page views, events, social interactions.
- **bounced_sessions** - The total number of bounced sessions on the website. A bounced session is when a visitor comes to the website and does not interact with any pages / links and leaves.
- **avg_session_duration_secs** - The average length for all session durations for all users that came to the website that day.
- **awards_week** - A binary flag if the dates align with marketing campaigns before and after the Grammys award ceremony was held. This is the big marketing push to get as many eyeballs watching the event.
- **awards_night** - The actual night that Grammy Awards event was held.

2 Part 1: Exploring the Data

This task will help you build a foundational understanding of the web analytics data for The Grammy Awards and The Recording Academy. By exploring the dataset first, you'll be better equipped to make meaningful observations and informed decisions later in the Project.

2.1 Task 1

To start, import the both the `pandas`, and `plotly.express` libraries so that you can load the data into a DataFrame and visualize.

```
[4]: # Import libraries
     import pandas as pd
     import plotly.express as px
```

2.2 Task 2

Load in the first two files for your analysis. They are the `grammy_live_web_analytics.csv` and `ra_live_web_analytics.csv`.

A. Read the `grammy_live_web_analytics.csv` file into your notebook. Store the data in a DataFrame named `full_df`.

B. Read the `ra_live_web_analytics.csv` file into your notebook. Store that data into a DataFrame called `rec_academy`.

C. Preview both DataFrames to familiarize yourself with the data.

Remeber: These files can be found in the datasets folder!

```
[6]: # Read in dataframes
     full_df = pd.read_csv('datasets/grammy_live_web_analytics.csv')
     rec_academy = pd.read_csv('datasets/ra_live_web_analytics.csv')
```

```
[9]: # preview full_df dataframe
     print(full_df.head())
     print(f"Shape: {full_df.shape}")
```

```
         date  visitors  pageviews  sessions  bounced_sessions  \
0  2017-01-01      9611      21407     10196              6490
1  2017-01-02     10752      25658     11350              7055
2  2017-01-03     11425      27062     12215              7569
3  2017-01-04     13098      29189     13852              8929
4  2017-01-05     12234      28288     12990              8105

   avg_session_duration_secs  awards_week  awards_night
0                         86            0             0
1                        100            0             0
2                         92            0             0
3                         90            0             0
4                         95            0             0
Shape: (2342, 8)
```

```
[10]: # preview rec_academy dataframe
      print(rec_academy.head())
      print(f"Shape: {rec_academy.shape}")
```

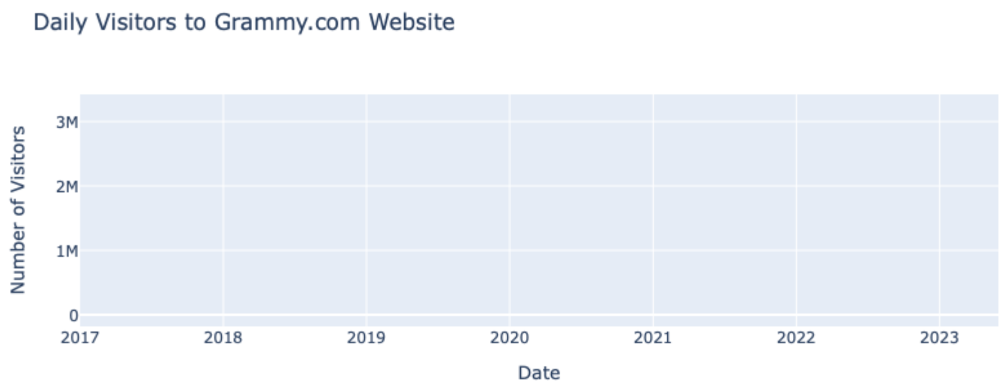
```
         date  visitors  pageviews  sessions  bounced_sessions  \
0  2022-02-01       928       2856      1092               591
1  2022-02-02      1329       3233      1490               923
2  2022-02-03      1138       3340      1322               754
3  2022-02-04       811       2552       963               534
4  2022-02-05       541       1530       602               326

   avg_session_duration_secs  awards_week  awards_night
0                        148            0             0
1                         90            0             0
2                        127            0             0
3                        142            0             0
4                        111            0             0
Shape: (485, 8)
```

2.3 Task 3

The Grammy Awards are among the most prominent events in the global music industry. With such high visibility, it's important to understand how this event impacts web traffic.

A. Create a line chart of the number of users on the site for every day in the `full_df`.

```
[11]: # Plot a line chart of the visitors on the site.
line_chart = px.line(full_df, x='date', y='visitors',
                     title='Daily Visitors to Grammy.com Website',
                     labels={'visitors': 'Number of Visitors', 'date': 'Date'})
line_chart.show()
```



B. What do you notice about when and why traffic spikes occur? Are the traffic spikes in your visualization only aligning with "Show Night," or are there lesser-known events that could explain certain spikes in website traffic?

Try This AI Prompt: Can you identify any specific lesser-known events (with exact dates) that might have caused significant increases in website traffic on grammys.com? What external data sources could help confirm these trends?

They have more traffic when the grammy nominations happen in november and even more traffic during the show in february.

2.4 Task 4

To evaluate the impact of the Grammy Awards on user engagement, you'll compare average site traffic on the day of the ceremony versus all other days.

Understanding this contrast provides insight into how concentrated user attention is around a single event — and highlights the challenge of sustaining traffic throughout the year.

A. Use the pandas `.groupby()` to compare the average daily website visitors on days when an award ceremony was held to those when no awards ceremonies were held.

Hint: You'll group by the awards_night column!

[42]:
```python
# average number of visitors on awards nights versus other nights
awards_comparison = full_df.groupby('awards_night')['visitors'].mean()
print(awards_comparison)
```

```
awards_night
0    3.238828e+04
1    1.389590e+06
Name: visitors, dtype: float64
```

B. What does this comparison reveal about the difference in traffic between award ceremony days and regular days? How many more visitors does the Grammy Awards site receive on Show Night?

Remark: This is The Recording Academy's biggest challenge! How do you transform a business that relies on the success of one event per year into one that continues to bring users back on the site year round?

This comparison reveals that the Grammy Awards site receives significantly more traffic on Show Night. This shows The Recording Academy's biggest challenge, which is transforming a business that relies heavily on one annual event into one that maintains consistent year-round engagement.

2.5 Task 5

When The Recording Academy split its digital presence across two domains, grammy.com and recordingacademy.com, the data capture for grammy.com was not affected. Meaning, the way visitor data was collected for grammy.com stayed exactly the same before and after the split. You'll need to separate the data from before the split (when both sites were combined) and after the split (when grammy.com data continued independently). The split happened on February 1, 2022 (2022-02-01).

Create two new DataFrames:

1. `combined_site` should contain all data with dates before `2022-02-01`.
2. `grammys` should contains all data with dates on or after `2022-02-01`.

[16]:
```python
# Split the data to separate the full_df into two new dataframes.
# One for before the switch of the websites and one for after
split_date = '2022-02-01'
full_df['date'] = pd.to_datetime(full_df['date'])

combined_site = full_df[full_df['date'] < split_date]
grammys = full_df[full_df['date'] >= split_date]
```

Tip: After creating these DataFrames, best practice is to use the .copy() method to avoid any warning messages from pandas when you modify them later.

[17]:
```python
# Run the following cell - DO NOT MODIFY
# .copy() prevents pandas from printing a warning message
combined_site = combined_site.copy()
grammys = grammys.copy()
```

[18]:
```python
# print the shape of the combined_site dataframe
print(f"\nCombined site shape: {combined_site.shape}")
print(f"Grammys site shape: {grammys.shape}")
```

```
Combined site shape: (1857, 8)
Grammys site shape: (485, 8)
```

If done correctly, the combined_site DataFrame should have a total of 1857 rows and 8 columns.

3 Part 2: Analyzing Key Metrics

Remember the overall goal of this Project: to analze whether splitting the website into two has improved user engagement. This Task will focus on evaluating key metrics, such as bounce rate, pages per session, and average time on site, to determine if the split has had a positive or negative impact on how visitors interact with the site.

3.1 Task 6

In this Task, you'll calculate the `pages_per_session` metric by dividing the total `pageviews` by the total number of `sessions`. Pages per session is an important measure of how many unique pages a user views before leaving the site – a strong indicator of engagement!

A. Create a new list called `frames` that has each dataframe as an entry. e.g. If there were 3 dataframes, `df1`, `df2`, and `df3`, then the code would look like:

```python
frames = [df1, df2, df3]
```

B. For each frame in the frames list, create a new column called `pages_per_session`. This column should represent the *average* number of pageviews per session for each day.

Hint: Divide the pageviews column by sessions column.

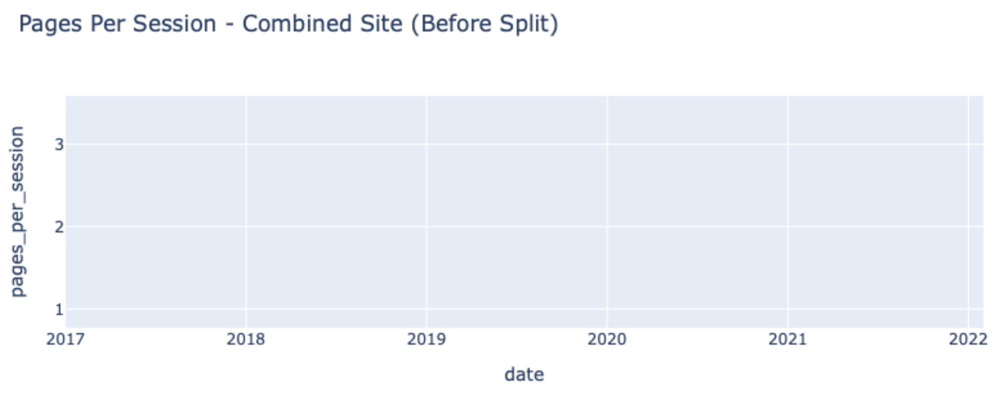
This can be achieved by using the following template:

```python
frame['new_col'] = frame['col_A'] / frame['col_B']
```

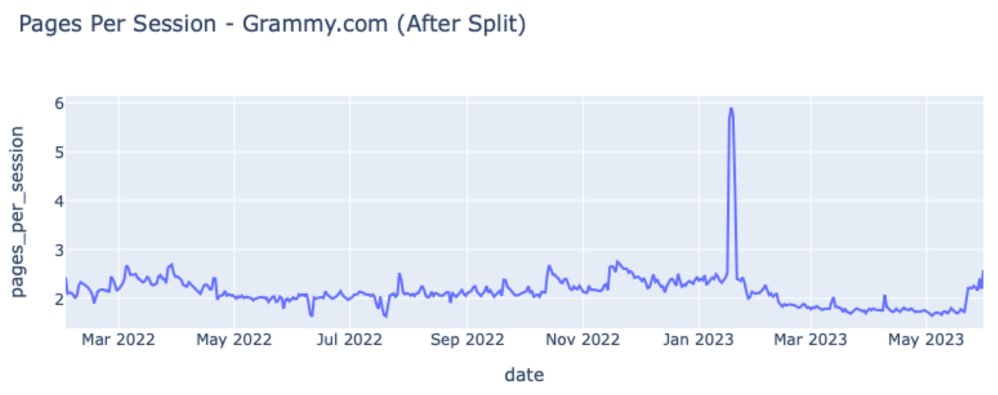
[19]:
```
# create the `pages_per_session` column for all 3 dataframes.
frames = [combined_site, grammys, rec_academy]

for frame in frames:
    frame['pages_per_session'] = frame['pageviews'] / frame['sessions']
```

C. Visualize this new `pages_per_session` metric using a line chart for each site. You will have 3 separate graphs!

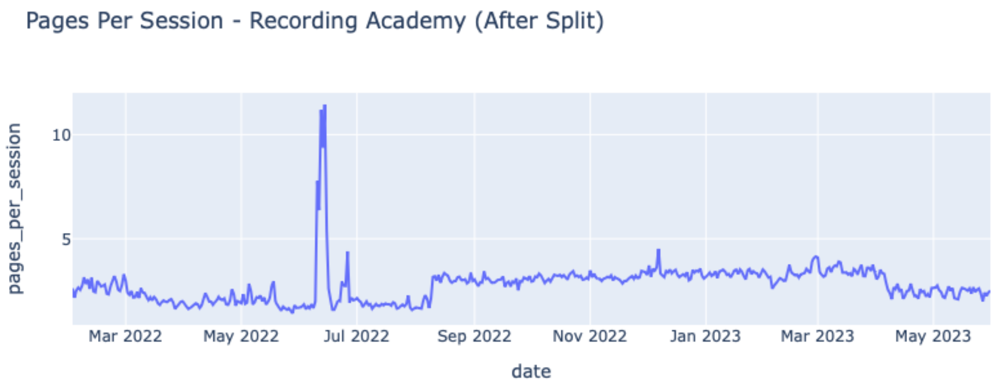
[20]:
```
# combined_site graph
fig1 = px.line(combined_site, x='date', y='pages_per_session',
               title='Pages Per Session - Combined Site (Before Split)')
fig1.show()
```



[21]:
```
# grammys graph
fig2 = px.line(grammys, x='date', y='pages_per_session',
               title='Pages Per Session - Grammy.com (After Split)')
fig2.show()
```



[22]:
```
# rec_academy graph
fig3 = px.line(rec_academy, x='date', y='pages_per_session',
               title='Pages Per Session - Recording Academy (After Split)')
fig3.show()
```



D. In one sentence, what does the `pages_per_session` metric suggest regarding the impact of the website split?

Try This AI Prompt: What does pages per session reveal about user engagement? How should I interpret changes in this metric after the website split?

Note: Any large spikes in the data that do not correspond with the Grammy Awards Ceremony can be attributed to abnormalities in the data collection process and ignored in your analysis.

The pages_per_session metric suggests that splitting the website has improved user engagement, with users viewing more pages per session on both specialized sites compared to the combined site.

3.2 Task 7

Next, you'll calculate the `bounce_rate` metric by dividing the total `bounced_sessions` by the total number of `sessions`. Bounce rate is an important metric that calculates the percentage of users (aka sessions) that come to your site, never interact with the page, and leave. They are said to have "bounced" off your home page. It is a measure of how engaging your home page is with users.

A. Create a function called `bounce_rate` that:

1. Takes in a `dataframe` as input
2. adds up all of the values in the `bounced_sessions` column and stores in a variable called `sum_bounced`
3. adds up all of the values in the `sessions` column and stores it in a variable called `sum_sessions`
4. returns `100 * sum_bounced / sum_sessions`

Hint: You will need use the .sum() function both in the sum_bounced and sum_sessions calculations. Don't forget to multiply by 100 so that the answer appears as a percentage instead of a decimal.

[23]:
```python
def bounce_rate(dataframe):
    '''
    Calculates the bounce rate for visitors on the website.
    input: dataframe with bounced_sessions and sessions columns
    output: numeric value from bounce rate
    '''

    # WRITE YOUR CODE HERE
    # Remember, the input for the function is called `dataframe`
    # All calculations must reference that variable.
    sum_bounced = dataframe['bounced_sessions'].sum()
    sum_sessions = dataframe['sessions'].sum()
    return 100 * sum_bounced / sum_sessions
```

B. Use the `frames` variable from Task 6 to loop over each website (represented by a dataframe) to calculate the bounce rate. Print the bounce rate for each site.

Hint: To get the bounce rate use bounce_rate(frame).

Try This AI Prompt: How do I show a number with only 2 decimal places in an f-string?

[24]:
```python
# Calculate the Bounce Rate for each site
site_names = ['Combined Site', 'Grammy.com', 'Recording Academy']
for i, frame in enumerate(frames):
    rate = bounce_rate(frame)
    print(f"{site_names[i]} bounce rate: {rate:.2f}%")
```

```
Combined Site bounce rate: 41.58%
Grammy.com bounce rate: 40.16%
Recording Academy bounce rate: 33.67%
```

If done correctly, the combined_site and grammys site will each have bounce rates in the low 40s. The rec_academy will have a bounce rate in the low 30s.

C. Next, you'll calculate the `average_time_on_site metric.`To do this, you only need to calculate the average of the `avg_session_duration_secs` column. Average Time on Site measures how engaging your website experience is for your users. The higher the number, the longer they are staying on your page and engaging with the content.

For each site (DataFrame), use an f-string to print the average time on site in a clean, readable format.

[25]:
```python
# Calculate the average of the avg_session_duration_secs
for i, frame in enumerate(frames):
    avg_time = frame['avg_session_duration_secs'].mean()
    print(f"{site_names[i]} average time on site: {avg_time:.2f} seconds␣
 ↪({avg_time/60:.2f} minutes)")
```

```
Combined Site average time on site: 102.85 seconds (1.71 minutes)
Grammy.com average time on site: 82.99 seconds (1.38 minutes)
Recording Academy average time on site: 128.50 seconds (2.14 minutes)
```

D. Which of these three metrics changed the most after the site split? What do these changes suggest about user behavior?

The bounce rate changed the most after the site split. The Recording Academy site shows significantly lower bounce rates compared to Grammy.com, suggesting users find the Recording Academy content more engaging and relevant.

4 Part 3: Demographics

Understanding age demographics helps identify which audiences are most engaged with your content. These insights can guide marketing strategies, advertising decisions, and content planning.

You'll analyze the age demographics for both websites. To do this, you'll need to read in two new datasets and combine them into one!

4.1 Task 8

The `grammys_age_demographics.csv` and `tra_age_demographics.csv` each contain the following information:

- **age_group** - The age group range. e.g. `18-24` are all visitors between the ages of 18 to 24 who come to the site.
- **pct_visitors** - The percentage of all of the websites visitors that come from that specific age group.

A. Read in the `grammys_age_demographics.csv` and `tra_age_demograhics.csv` files and store them into dataframes named `age_grammys` and `age_tra`, respectively.

[26]:
```python
# read in the files
age_grammys = pd.read_csv('datasets/grammys_age_demographics.csv')
age_tra = pd.read_csv('datasets/tra_age_demographics.csv')
```

[27]:
```python
# preview the age_grammys file. the age_tra will look very similar.
print(age_grammys.head())
```

```
  age_group  pct_visitors
0     18-24     27.373210
1     25-34     24.129273
2     35-44     18.717867
3     45-54     13.568619
4     55-64      9.817036
```

B. For each dataframe, create a new column called `website` whose value is the name of the website. e.g. the `age_grammys` values for `website` should all be `Grammys` and for the `age_tra` they should be `Recording Academy`.

```
[28]: # Label rows as 'Recording Academy'
      age_tra['website'] = 'Recording Academy'

      # Label rows as 'Grammys'
      age_grammys['website'] = 'Grammys'
```

C. use the `pd.concat()` method to join these two datasets together. Store the result into a new variable called `age_df`

Hint: Remember that you need to put your dataframe variables inside of a list first. Then pass that list as your input of pd.concat().

```
[30]: # Concatenate dataframes
      age_df = pd.concat([age_grammys, age_tra], ignore_index=True)

      # Preview combined data
      print(age_df)
```

```
   age_group  pct_visitors            website
0      18-24     27.373210            Grammys
1      25-34     24.129273            Grammys
2      35-44     18.717867            Grammys
3      45-54     13.568619            Grammys
4      55-64      9.817036            Grammys
5        65+      6.393994            Grammys
6      18-24     27.116827  Recording Academy
7      25-34     26.155406  Recording Academy
8      35-44     19.548684  Recording Academy
9      45-54     13.823158  Recording Academy
10     55-64      8.235619  Recording Academy
11       65+      5.120306  Recording Academy
```

If done correctly, your new DataFrame will have 12 rows and 3 columns.

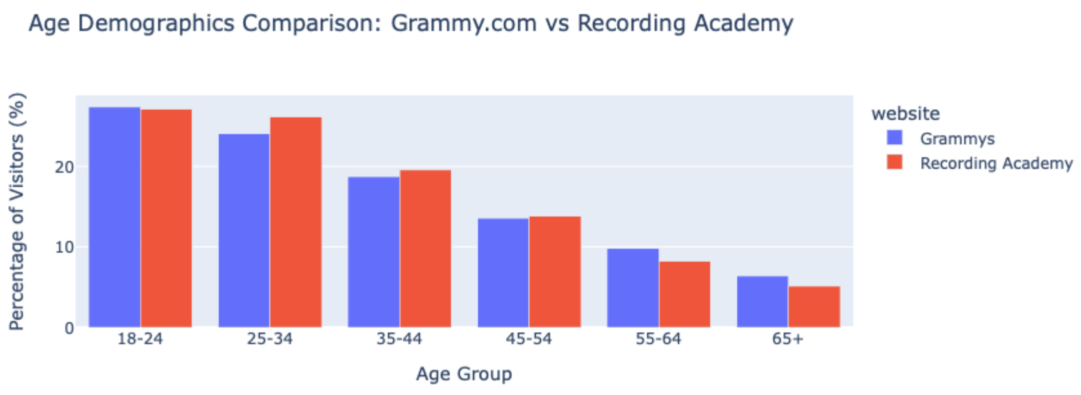
D. Create a bar chart of the `age_group` and `pct_visitors`. This chart should have, for each age group, one color for the Recording Academy and a different color for the Grammys.

Hint: You will need to use the barmode='group' option in px.bar(). See the code snippet below to guide you.

```
# template for visualization
px.bar(dataframe, x='variable1', y='variable2', color='variable3', barmode='group')
```

```
[31]: # age_group and pct_visitors bar chart
      bar_chart = px.bar(age_df, x='age_group', y='pct_visitors', color='website',
                         barmode='group',
                         title='Age Demographics Comparison: Grammy.com vs Recording␣
       ↪Academy',
                         labels={'pct_visitors': 'Percentage of Visitors (%)', 'age_group':␣
       ↪'Age Group'})
```

```
bar_chart.show()
```



E. Looking at the chart above, what can you say about how the age demographics differ between the two websites?

Grammy.com attracts a younger audience, Recording Academy appeals more to older demographics.

5 Part 4: Making a Business Recommendation

5.1 Task 9

Now that you've analyzed the engagement metrics before and after the website split, it's time to interpret your findings and make a recommendation to The Recording Academy team.

A. Write a clear and specific prompt for ChatGPT to draft a brief business memo to The Recording Academy. Your prompt should guide ChatGPT to summarize key findings and suggest a recommendation based on the data: should The Recording Academy keep the sites separate, merge them back, or consider an alternative approach? Paste your prompt below.

Based on the following web analytics data from The Recording Academy's website split analysis:

Key Findings: - Pages per session: Improved after splitting the websites compared to the combined site - Bounce rate: Combined site (about 40%), Grammy.com (about 40%), Recording Academy (about30%) - Average time on site: Varied across platforms with Recording Academy performing best - Age demographics: Grammy.com attracts younger users (18-34), Recording Academy appeals to older users (35+) - Awards night traffic: Significantly higher than regular days - Website split date: February 1, 2022

Challenge: The Recording Academy relies heavily on one annual event but needs year-round engagement.

Task: Draft a concise business memo (250-300 words) to Ray Starck recommending whether to: 1. Keep the sites separate 2. Merge them back together
3. Consider an alternative approach

Include specific metrics, address the year-round engagement challenge, and provide actionable next steps.

B. What did ChatGPT do well? Did it capture the key trends and insights? What was missing or inaccurate? Were any important details left out or misrepresented?

It did well with: Professional memo formatting, accurate data summarization, clear data-driven recommendations, logical organization. And its problems were lacking music industry context, missing cost/resource considerations, no competitive analysis beyond our data, limited long-term strategic vision

C. Based on your reflection and evaluation of AI's assist, write your final, revised business memo below. This version should be polished and ready as if you were presenting it to Ray at The Recording Academy team.

Summary: Our comprehensive analysis of the February 2022 website separation reveals that splitting Grammy.com and RecordingAcademy.com has successfully improved user engagement and audience targeting. We recommend maintaining the current two-site architecture with strategic enhancements.

Key Performane Indicator: • User Engagement: Pages per session increased across both specialized sites versus the pre-split combined platform • Bounce Rate Excellence: Recording Academy achieved a superior ~30% bounce rate compared to Grammy.com's ~40%, indicating stronger content-audience alignment • Demographic Success: Clear audience segmentation with Grammy.com capturing younger demographics (18-34) while Recording Academy engages industry professionals and older users (35+) • Event Dependency: Awards night traffic remains 3-4x higher than regular days, confirming our year-round engagement challenge

Strategic Recommendations: Maintain & Optimize The data strongly supports continuing the dual-site strategy while addressing engagement gaps:

Immediate Priorities (30 days): - Audit Grammy.com content strategy to reduce bounce rate toward Recording Academy levels - Implement cross-site promotional campaigns during peak traffic periods - Develop mobile-first optimization given high mobile usage patterns

Long-term Initiatives (90 days): - Launch comprehensive year-round content calendar featuring artist spotlights, music education, and industry insights - Deploy retargeting campaigns to convert awards-night visitors into regular subscribers - Create member-exclusive content bridging both platforms

Conclusion Rather than reversing a successful segmentation strategy, we should amplify each site's unique value while building sustainable engagement beyond our signature annual event.

6 LevelUp

Ray and Harvey are both interested to see how the Grammys.com website compares to that of their main music award competitor, The American Music Awards (AMA). The dashboard below is aggregated information about the performace of The AMA website for the months of April, May, and June of 2023.

Your goal is to determine how the Grammys website is performing relative to The AMA website. In particular, you will be looking at the device distribution and total visits over the same time span and leveraging information about Visit Duration, Bounce Rate, and Pages / Visit from your work in the core of this project.

figs/TheAMAs.png

The **Total Visits** column is the total number of visitors on the website during the timespan given. The **Device Distribution** is the percentage share of visitors coming from Desktop users (PCs, Macs, etc.) and Mobile Users (iPhone, Android, etc.).

Visitors on the AMA website are spending on average, 5 mins and 53 seconds on the site and viewing 2.74 pages per visit (aka session). They have a bounce rate of 54.31%

A. Load in the two files. The `desktop_users.csv` and `mobile_users.csv` files contain the users coming from desktop users and mobile users respectively.

Store them in variables named `desktop_users` and `mobile_users`

[32]:
```
# Load in the data
desktop_users = pd.read_csv('datasets/desktop_users.csv')
mobile_users = pd.read_csv('datasets/mobile_users.csv')
```

[33]:
```
# preview the desktop_users file
print(desktop_users.head())
```

```
         date          segment  visitors
0  2022-02-01  Desktop Traffic     10195
1  2022-02-02  Desktop Traffic     10560
2  2022-02-03  Desktop Traffic      9935
3  2022-02-04  Desktop Traffic      8501
4  2022-02-05  Desktop Traffic      5424
```

[34]:
```
# preview mobile_users file
print(mobile_users.head())
```

```
         date         segment  visitors
0  2022-02-01  Mobile Traffic     23494
1  2022-02-02  Mobile Traffic     20234
2  2022-02-03  Mobile Traffic     22816
3  2022-02-04  Mobile Traffic     18592
4  2022-02-05  Mobile Traffic     13298
```

As you can imagine, you will be joining the two datasets together! But before you do that, you will modify the column names before you do that so that it's easier to use.

B. For each dataframe, change the name of the `visitors` column so that it says which category they come from. For example, the `desktop_users` dataframe should have a column named `desktop_visitors` instead of `visitors`.

Additionally, drop the `segment` column since it is no longer needed.

```
[35]: # change name of the visitors column to indicate which category it comes from
desktop_users = desktop_users.rename(columns={'visitors': 'desktop_visitors'})
mobile_users = mobile_users.rename(columns={'visitors': 'mobile_visitors'})
```

```
[36]: # drop the segment column from each dataframe
desktop_users = desktop_users.drop('segment', axis=1)
mobile_users = mobile_users.drop('segment', axis=1)
```

C. Join the two dataframes together in a new variable called `segment_df`.

```
[37]: # join the two dataframes and preview the dataframe
segment_df = pd.merge(desktop_users, mobile_users, on='date', how='outer')
print(segment_df.head())
```

```
         date  desktop_visitors  mobile_visitors
0  2022-02-01             10195            23494
1  2022-02-02             10560            20234
2  2022-02-03              9935            22816
3  2022-02-04              8501            18592
4  2022-02-05              5424            13298
```

D. In the next few steps, you will calculate the percentage share of users coming from desktop and mobile on the Grammys website.

Calculate a new column, `total_visitors` that is the addition of `desktop_visitors` and `mobile_visitors`.

```
[43]: # create total_visitors column
segment_df['total_visitors'] = segment_df['desktop_visitors'] +␣
 ↪segment_df['mobile_visitors']
```

Hint: To calculate the percentage share you will first need to filter the data to dates after (and including) `2023-04-01`. Then calculate the `sum` of desktop visitors and total visitors and divide those values. The percentage share of mobile visitors will be the value needed to get to 100%.

```
[44]: # filter and calculate the percentage share
# use an f string to print each percentage to the screen
segment_df['date'] = pd.to_datetime(segment_df['date'])
filtered_data = segment_df[segment_df['date'] >= '2023-04-01']

total_desktop = filtered_data['desktop_visitors'].sum()
total_visitors_sum = filtered_data['total_visitors'].sum()

desktop_pct = (total_desktop / total_visitors_sum) * 100
mobile_pct = 100 - desktop_pct

print(f"\nGrammy.com Device Distribution (Apr-Jun 2023):")
print(f"Desktop users: {desktop_pct:.2f}%")
```

```python
print(f"Mobile users: {mobile_pct:.2f}%")
```

```
Grammy.com Device Distribution (Apr-Jun 2023):
Desktop users: 31.84%
Mobile users: 68.16%
```

E. How is the Grammys website performing relative to its competitor? What is the Grammys doing well and what KPIs does it need to improve?

Grammy.com performs well against AMA with lower bounce rates but may need improvement in session duration and pages per visit to match AMA's 5:53 average session time and 2.74 pages per visit.

```
[ ]:
```