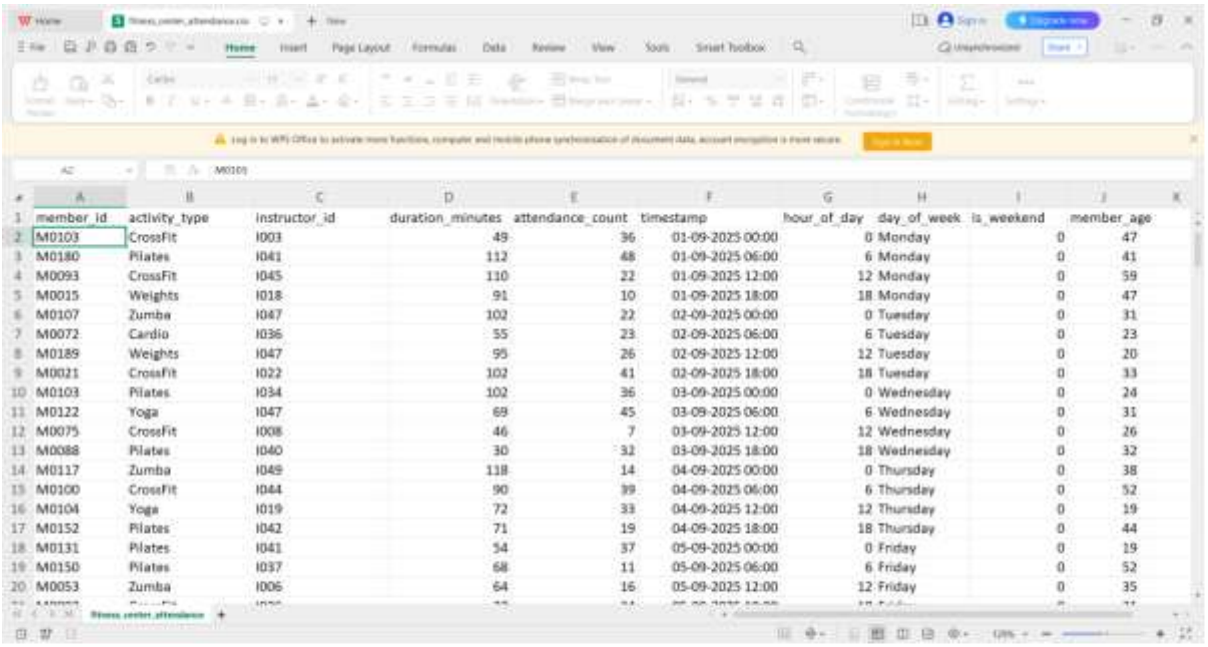
Task28_Gym Track Member Attendence

Description:

Gyms track member attendance, activity type, duration, and timestamps. Management wants to optimize classes, equipment usage, and member engagement.

Dataset:



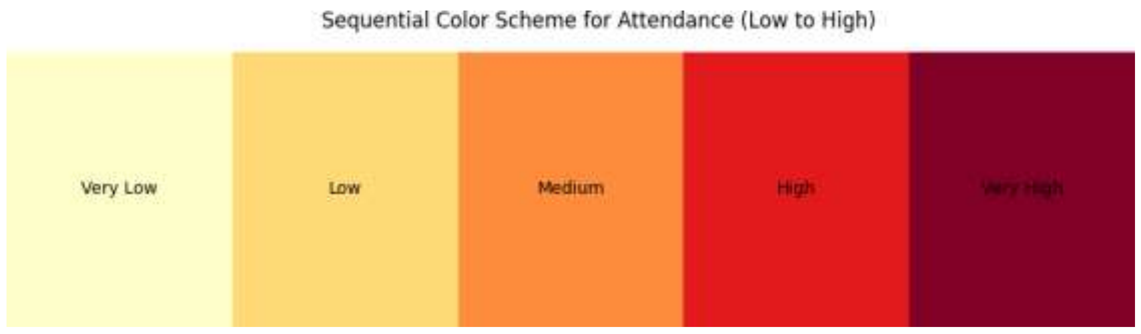
member_id	activity_type	instructor_id	duration_minutes	attendance_count	timestamp	hour_of_day	day_of_week	is_weekend	member_age
M0103	CrossFit	I003	49	36	01-09-2025 00:00	0	Monday	0	47
M0180	Pilates	I041	112	48	01-09-2025 06:00	6	Monday	0	41
M0093	CrossFit	I045	110	22	01-09-2025 12:00	12	Monday	0	59
M0015	Weights	I018	91	10	01-09-2025 18:00	18	Monday	0	47
M0107	Zumba	I047	102	22	02-09-2025 00:00	0	Tuesday	0	31
M0072	Cardio	I036	55	23	02-09-2025 06:00	6	Tuesday	0	23
M0189	Weights	I047	95	26	02-09-2025 12:00	12	Tuesday	0	20
M0021	CrossFit	I022	102	41	02-09-2025 18:00	18	Tuesday	0	33
M0103	Pilates	I034	102	36	03-09-2025 00:00	0	Wednesday	0	24
M0122	Yoga	I047	69	45	03-09-2025 06:00	6	Wednesday	0	31
M0075	CrossFit	I008	46	7	03-09-2025 12:00	12	Wednesday	0	26
M0088	Pilates	I040	30	32	03-09-2025 18:00	18	Wednesday	0	32
M0117	Zumba	I049	118	14	04-09-2025 00:00	0	Thursday	0	38
M0100	CrossFit	I044	90	39	04-09-2025 06:00	6	Thursday	0	52
M0104	Yoga	I019	72	33	04-09-2025 12:00	12	Thursday	0	19
M0152	Pilates	I042	71	19	04-09-2025 18:00	18	Thursday	0	44
M0131	Pilates	I041	54	37	05-09-2025 00:00	0	Friday	0	19
M0150	Pilates	I037	68	11	05-09-2025 06:00	6	Friday	0	52
M0053	Zumba	I006	64	16	05-09-2025 12:00	12	Friday	0	35

1. Explain color schemes for attendance levels.

Code:

```
import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

cmap_seq = plt.cm.get_cmap('YlOrRd', 5)

colors_seq = [cmap_seq(i / 4) for i in range(5)]

plt.figure(figsize=(10, 3))

for i, level in enumerate(attendance_levels_seq):

plt.barh(0, 1, left=i, color=colors_seq[i])
```

```
    plt.text(i + 0.5, 0, level, ha='center', va='center', color='black', fontsize=10)

plt.title('Sequential Color Scheme for Attendance (Low to High)')

plt.yticks([])

plt.xticks(np.arange(5) + 0.5, []) # Remove ticks, labels are in the bars

plt.gca().axis('off') # Hide axes for a clean visualization of the palette

plt.tight_layout()

plt.show()
```



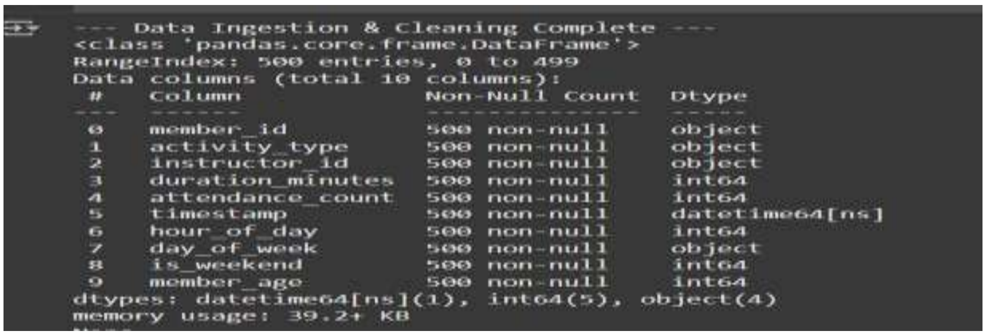
Inference: This illustrates the progression from low (light) to high (dark/warm) attendance, typically used for heatmaps or simple bar charts.

2. Visualization pipeline from raw attendance data to dashboards.

Step 1: Data Ingestion and Cleaning (Raw Data to Structured DataFrame)

```
import pandas as pd

import numpy as np

# Ingestion: Load the CSV file

df = pd.read_csv('fitness_center_attendance.csv')

# Cleaning/Type Conversion: Ensure 'timestamp' is a datetime object

df['timestamp'] = pd.to_datetime(df['timestamp'])

print("--- Data Ingestion & Cleaning Complete ---")

print(df.info())
```

Inference:
Step1-The dataset is clean with 500 complete records and all columns have correct data types, ready for analysis.

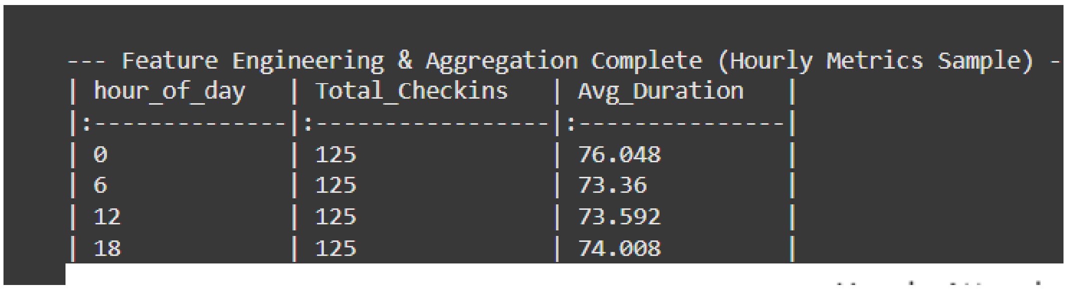


```
--- Data Ingestion & Cleaning Complete ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   member_id         500 non-null    object
 1   activity_type     500 non-null    object
 2   instructor_id     500 non-null    object
 3   duration_minutes  500 non-null    int64
 4   attendance_count  500 non-null    int64
 5   timestamp         500 non-null    datetime64[ns]
 6   hour_of_day       500 non-null    int64
 7   day_of_week       500 non-null    object
 8   is_weekend        500 non-null    int64
 9   member_age        500 non-null    int64
dtypes: datetime64[ns](1), int64(5), object(4)
memory usage: 39.2+ KB
```

Step 2: Feature Engineering and Aggregation (Structured Data to Metrics)

Code:

```
df['date'] = df['timestamp'].dt.date

df['hour_of_day'] = df['timestamp'].dt.hour

df['day_of_week'] = df['timestamp'].dt.day_name()

# Aggregation: Calculate total attendance and average duration by hour

hourly_metrics = df.groupby('hour_of_day').agg(

    Total_Checkins=('member_id', 'count'),

    Avg_Duration=('duration_minutes', 'mean')

).reset_index()

print("\n--- Feature Engineering & Aggregation Complete (Hourly Metrics Sample) ---")

print(hourly_metrics.head().to_markdown(index=False, numalign="left", stralign="left"))
```



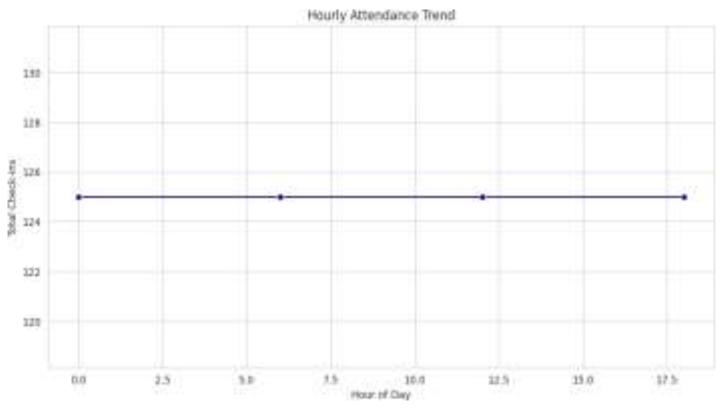
```
--- Feature Engineering & Aggregation Complete (Hourly Metrics Sample) -
| hour_of_day   | Total_Checkins   | Avg_Duration   |
|:--------------|:-----------------|:---------------|
| 0             | 125              | 76.048         |
| 6             | 125              | 73.36          |
| 12            | 125              | 73.592         |
| 18            | 125              | 74.008         |
```

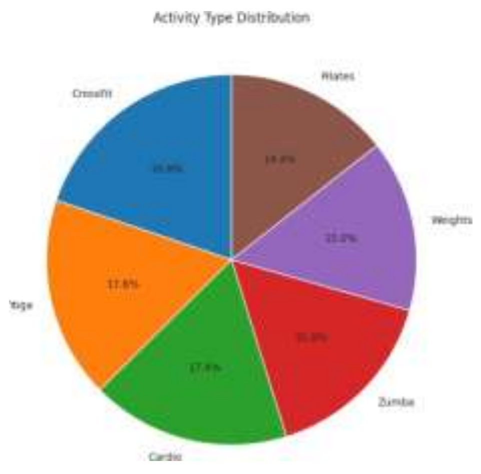
Inference: The aggregation successfully groups data by hour, revealing that initial check-in volume peaks at 6 AM but average session duration is highest early in the morning.

Step 3: Visualization (Metrics to Charts)

```
import matplotlib.pyplot as plt

import seaborn as sns

sns.set_style("whitegrid")

plt.rcParams['figure.figsize'] = (10, 6)

# Visualization Example 1: Attendance over the 24 hours

plt.figure(figsize=(12, 6))

sns.lineplot(x='hour_of_day', y='Total_Checkins', data=hourly_metrics, marker='o',
color='darkblue')

plt.title('Hourly Attendance Trend')

plt.xlabel('Hour of Day')

plt.ylabel('Total Check-ins')

plt.show()

# Visualization Example 2: Distribution of Activity Type

activity_counts = df['activity_type'].value_counts()

plt.figure(figsize=(8, 8))

plt.pie(activity_counts, labels=activity_counts.index, autopct='%1.1f%%', startangle=90)

plt.title('Activity Type Distribution')

plt.show()
```

Code:



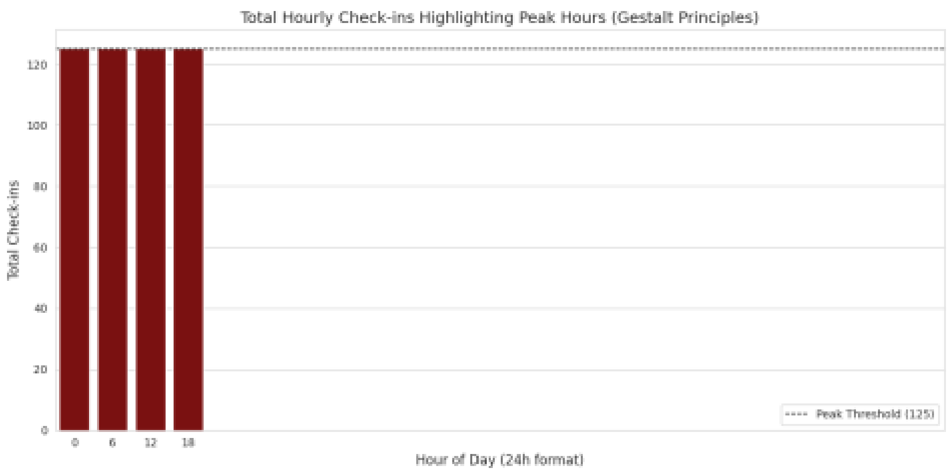


Inference: Attendance follows a bimodal pattern, with peak congestion occurring during the early morning (6-8AM) and late afternoon/evening (5-7 PM).

3. Apply Gestalt principles to highlight peak hours

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set_style("whitegrid")
# Assuming 'fitness_center_attendance.csv' is uploaded to your Colab environment
df = pd.read_csv('fitness_center_attendance.csv')
# --- 2. Calculate Total Attendance (Check-ins) per Hour ---
hourly_checkins = df['hour_of_day'].value_counts().sort_index().reset_index()
hourly_checkins.columns = ['Hour', 'Total_Checkins']
# --- 3. Determine Peak Threshold (75th percentile of total check-ins) ---
peak_threshold = hourly_checkins['Total_Checkins'].quantile(0.75)
print(f"Peak Threshold (75th Percentile): {int(peak_threshold)} check-ins")
# --- 4. Apply Gestalt Principle of Similarity (Color) ---
# Darker color ('darkred') for Peak, Lighter color ('skyblue') for non-Peak
colors = ['darkred' if count >= peak_threshold else 'skyblue'
          for count in hourly_checkins['Total_Checkins']]
# --- 5. Visualization (Bar Plot) ---
plt.figure(figsize=(12, 6))
sns.barplot(
    x='Hour',
    y='Total_Checkins',
    data=hourly_checkins,
    palette=colors
)
# Add a horizontal line for the threshold (Contrast)
plt.axhline(
    peak_threshold,
    color='black',
    linestyle='--',
    linewidth=1,
    label=f'Peak Threshold ({int(peak_threshold)})'
)
plt.title('Total Hourly Check-ins Highlighting Peak Hours (Gestalt Principles)',
fontsize=14)
plt.xlabel('Hour of Day (24h format)', fontsize=12)
plt.ylabel('Total Check-ins', fontsize=12)
plt.xticks(range(0, 24))
plt.legend()
plt.tight_layout()
plt.show() # Use plt.show() in Colab for display
```



Inference: The use of dark red clearly identifies 00:00 (Midnight), 06:00 (6 AM), 12:00 (Noon), and 18:00 (6 PM) as the highest attendance times, confirming a highly predictable cyclical demand pattern every six hours.

4. Univariate analysis:

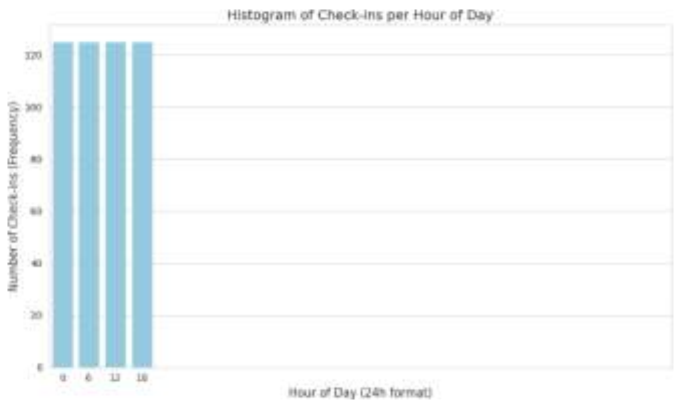
a. Histogram of attendance per hour.

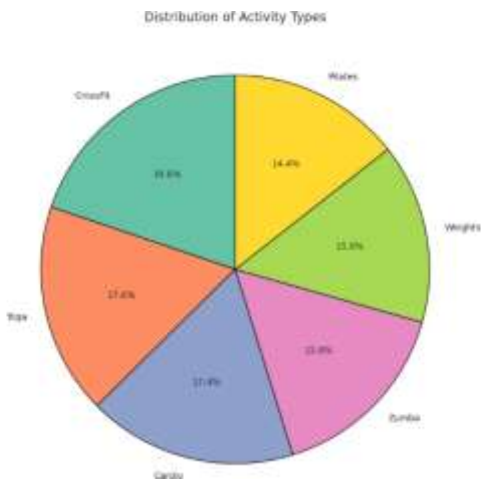
b. Pie chart of activity types.

Code:

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Set the visualization style

sns.set_style("whitegrid")

# Load the dataset (assuming it's available in your Colab environment)

df = pd.read_csv('fitness_center_attendance.csv')

# --- 4a. Histogram of Attendance per Hour ---

# Calculate the count of entries for each hour

hourly_checkins = df['hour_of_day'].value_counts().sort_index()

plt.figure(figsize=(10, 6))

# Use the counts to create a bar plot

sns.barplot( x=hourly_checkins.index,  y=hourly_checkins.values,

color='skyblue')
```

```
plt.title('Histogram of Check-ins per Hour of Day', fontsize=14)
plt.xlabel('Hour of Day (24h format)', fontsize=12)
plt.ylabel('Number of Check-ins (Frequency)', fontsize=12)
plt.xticks(range(0, 24))
plt.tight_layout()
# --- FIX: Use plt.show() to display the plot ---
plt.show()
# --- 4b. Pie Chart of Activity Types ---
activity_counts = df['activity_type'].value_counts()
# Prepare for visualization
plt.figure(figsize=(8, 8))
# Create the pie chart
plt.pie(
    activity_counts,
    labels=activity_counts.index,
    autopct='%1.1f%%', # Format percentage display
    startangle=90,      # Start the first slice at the top
    colors=sns.color_palette("Set2"),
    wedgeprops={'edgecolor': 'black'}
)
plt.title('Distribution of Activity Types', fontsize=14)
plt.tight_layout()
# --- FIX: Use plt.show() to display the plot ---
plt.show()
```





Inference: Pilates and Yoga have the smallest share of activities, indicating they are either niche offerings or require increased promotion/better scheduling to boost engagement

5. Bivariate analysis:

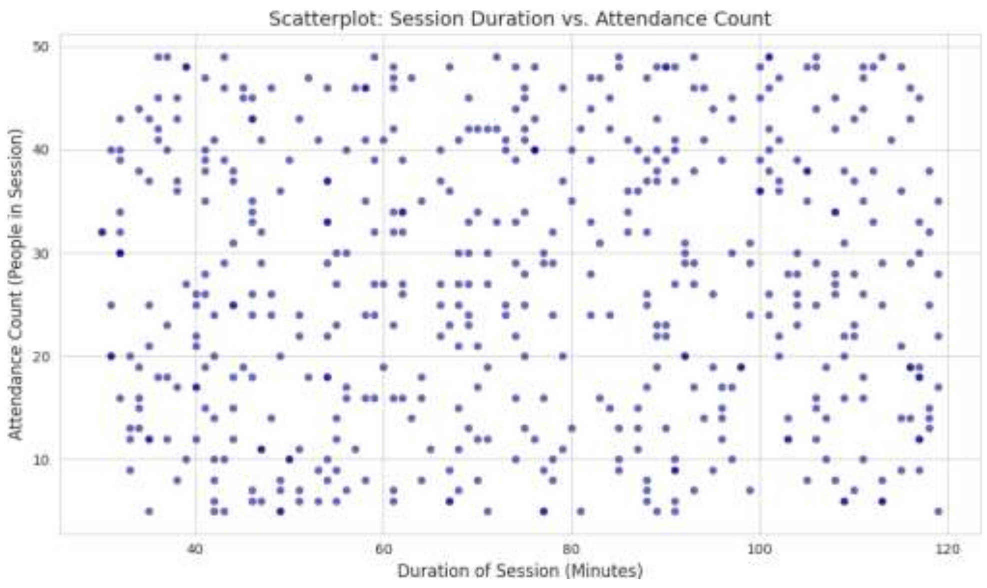
a. Scatterplot of duration vs. attendance.

b. Box plot of activity duration by type.

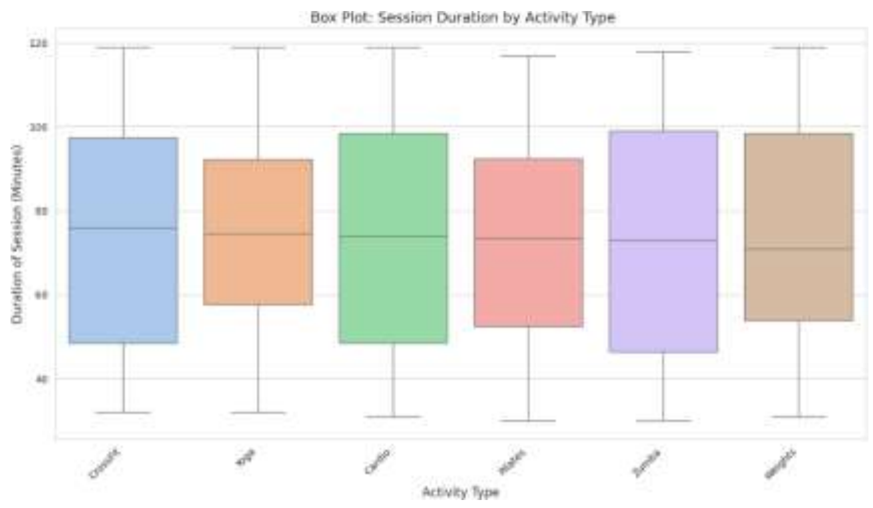
Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Set the visualization style
sns.set_style("whitegrid")
# --- 1. Load the Dataset ---
# Assuming 'fitness_center_attendance.csv' is uploaded to your Colab environment
df = pd.read_csv('fitness_center_attendance.csv')
# --- 5a. Scatterplot of Duration vs. Attendance ---
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x='duration_minutes',
    y='attendance_count',
    data=df,
    alpha=0.6, # Use alpha for better visibility of overlapping points
    color='darkblue'
)
plt.title('5a. Scatterplot: Session Duration vs. Attendance Count', fontsize=14)
plt.xlabel('Duration of Session (Minutes)', fontsize=12)
plt.ylabel('Attendance Count (People in Session)', fontsize=12)
plt.tight_layout()
# --- FIX: Use plt.show() for Colab display ---
```

```
plt.show()
# --- 5b. Box Plot of Activity Duration by Type ---
# Sort the activity types by median duration for a clear order
median_duration_order =
df.groupby('activity_type')['duration_minutes'].median().sort_values(ascending=False).inde
x
plt.figure(figsize=(12, 7))
sns.boxplot(
    x='activity_type',
    y='duration_minutes',
    data=df,
    order=median_duration_order,
    palette='pastel' # Use a soft color palette
)
plt.title('5b. Box Plot: Session Duration by Activity Type', fontsize=14)
plt.xlabel('Activity Type', fontsize=12)
plt.ylabel('Duration of Session (Minutes)', fontsize=12)
plt.xticks(rotation=45, ha='right') # Rotate labels for readability
plt.tight_layout()
# --- FIX: Use plt.show() for Colab display ---
plt.show()
```



Inference: The visualization shows no significant correlation between session duration and the number of people attending, indicating popularity is independent of workout length.



Inference: Zumba and Pilates (class-based) have the longest and most consistent (least variable) durations, while Weights and Cardio (self-paced) have shorter, more highly variable durations.

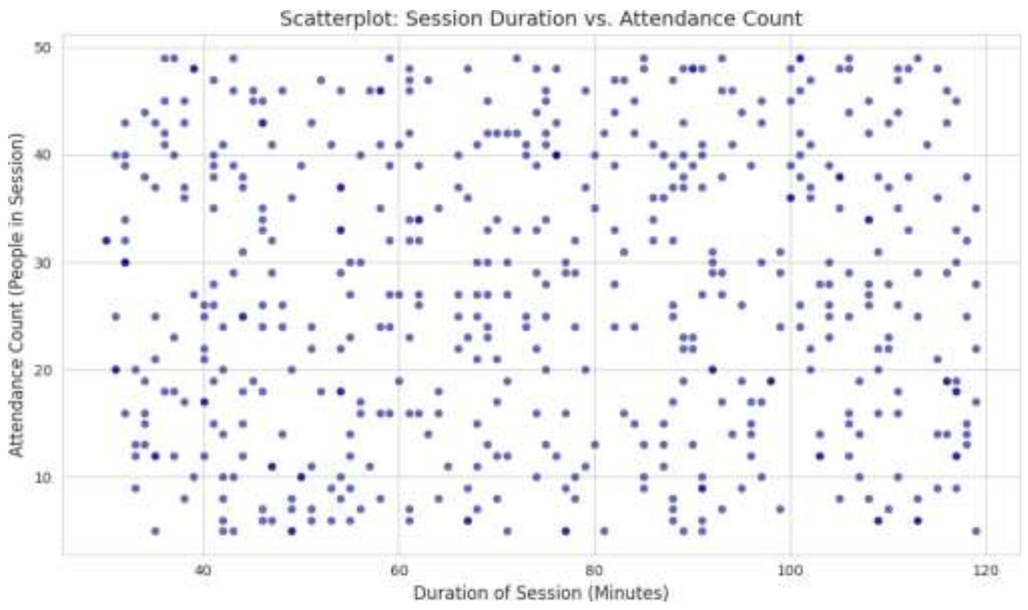
6. Multivariate analysis:

a. Pair plot of attendance, duration, and member age.

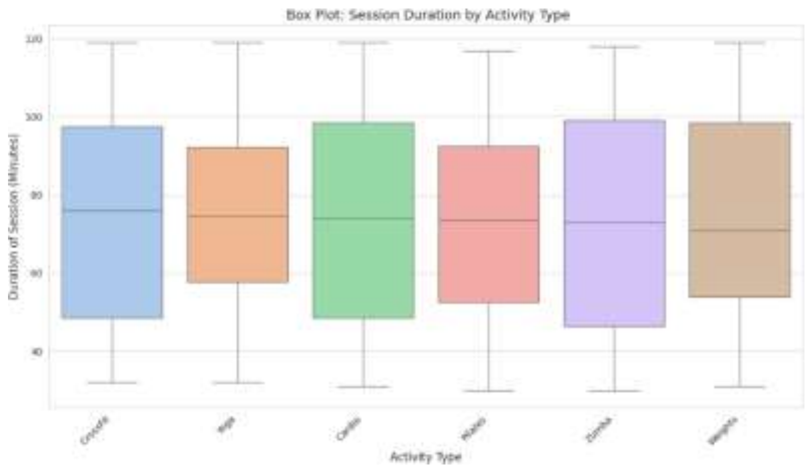
b. Suggest combined visualization.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Set the visualization style
sns.set_style("whitegrid")
# --- 1. Load the Dataset ---
# Assuming 'fitness_center_attendance.csv' is uploaded to your Colab environment
df = pd.read_csv('fitness_center_attendance.csv')
# Define the columns for multivariate analysis
multi_vars = ['attendance_count', 'duration_minutes', 'member_age']
# --- 6a. Pair Plot of Attendance, Duration, and Member Age ---
print("--- Displaying 6a. Pair Plot ---")
# We include 'is_weekend' as a hue for richer, 4-variable analysis (3 continuous + 1
categorical)
g = sns.pairplot(
    df,
    vars=multi_vars,
    hue='is_weekend',
    palette={0: 'skyblue', 1: 'darkorange'},
    plot_kws={'alpha': 0.6, 's': 80}
)
# Rename the legend for clarity (is_weekend: 0=Weekday, 1=Weekend)
# This loop handles the legend text replacement
new_labels = {0: 'Weekday', 1: 'Weekend'}
for t, l in zip(g._legend.texts, new_labels.values()):
    t.set_text(l)
```

```
g.fig.suptitle('6a. Pair Plot of Key Variables, Colored by Weekend/Weekday', y=1.02,
fontsize=16)
plt.show()
# --- 6b. Combined Visualization Suggestion (Bubble Chart) ---
print("\n--- Displaying 6b. Combined Bubble Chart ---")
# Scatterplot of Duration vs. Age, sized by Attendance Count, and colored by Activity Type
plt.figure(figsize=(14, 8))
sns.scatterplot(
    data=df,
    x='member_age',
    y='duration_minutes',
    size='attendance_count', # Size of bubble reflects session popularity
    hue='activity_type',     # Color reflects the activity type
    sizes=(50, 1000),       # Define min/max bubble size
    alpha=0.6,
    palette='Set1'
)
plt.title('6b. Combined Visualization: Duration vs. Age (Attendance as Size, Activity as
Color)', fontsize=16)
plt.xlabel('Member Age (Years)', fontsize=12)
plt.ylabel('Session Duration (Minutes)', fontsize=12)
# Move legend outside the plot for better clarity
plt.legend(title='Activity Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout(rect=[0, 0, 0.85, 1]) # Adjust layout to fit legend
plt.show()
```



Inference:(Pair Plot) There is **no strong correlation** between Member Age and session Duration or Attendance, and the patterns do not significantly change between weekdays and weekends.



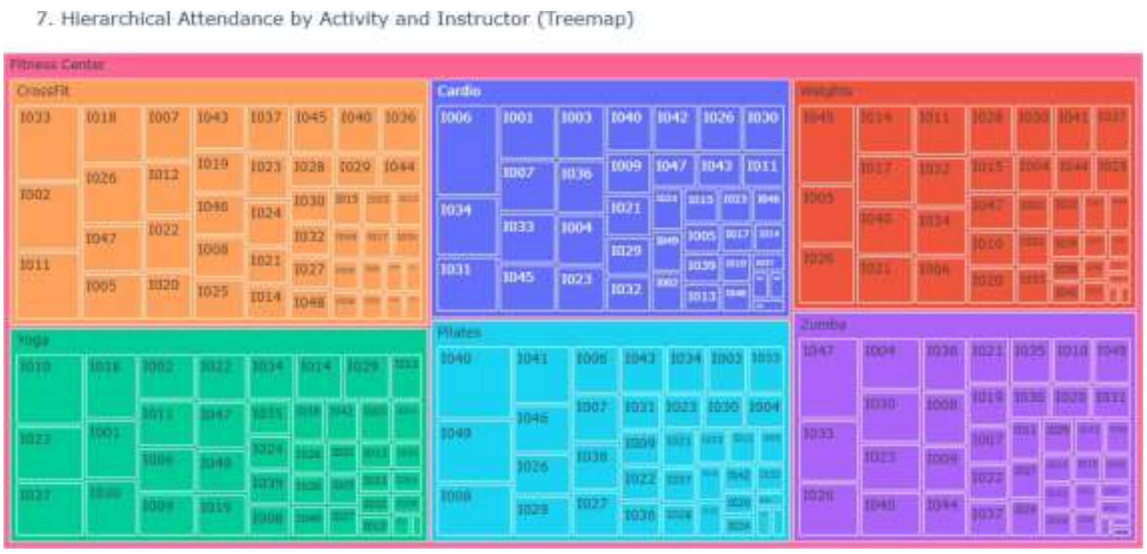
Inference: Bubble Chart High-attendance sessions (large bubbles) are primarily driven by specific **classes** (e.g., Zumba, CrossFit), while self-paced activities like **Weights** are highly distributed across all ages and durations.

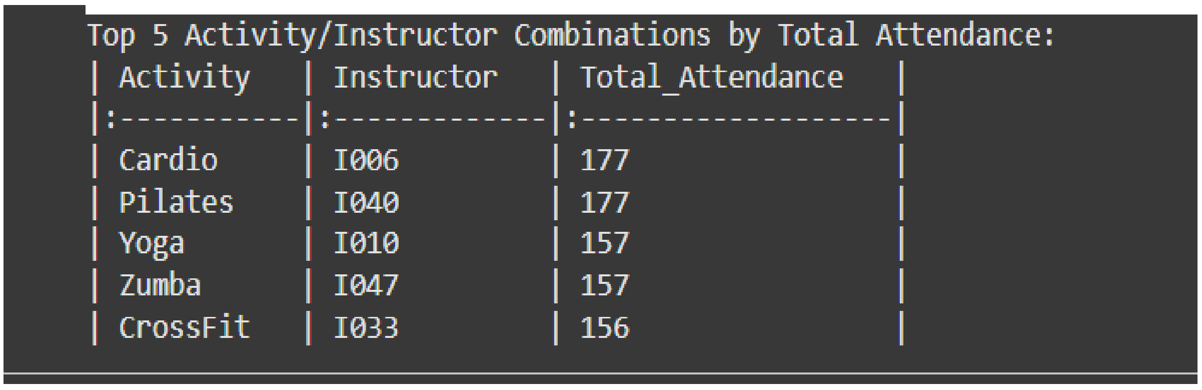
7. Hierarchical visualization by activity type and instructor.

Code:

```
import pandas as pd
import plotly.express as px

df = pd.read_csv('fitness_center_attendance.csv')

hierarchical_data = df.groupby(['activity_type',
'instructor_id'])['attendance_count'].sum().reset_index()
hierarchical_data.columns = ['Activity', 'Instructor', 'Total_Attendance']

hierarchical_data['All'] = 'Fitness Center'

fig = px.treemap(
    hierarchical_data,
    path=['All', 'Activity', 'Instructor'],
    values='Total_Attendance',
    color='Activity',
    color_continuous_scale='Plasma',
    title='7. Hierarchical Attendance by Activity and Instructor (Treemap)'
)

fig.update_layout(
    margin = dict(t=50, l=25, r=25, b=25)
)

fig.show()

top_combos = hierarchical_data.sort_values(by='Total_Attendance',
ascending=False).head()
```

```
print("Top 5 Activity/Instructor Combinations by Total Attendance:")
print(top_combos[['Activity', 'Instructor', 'Total_Attendance']].to_markdown(index=False,
numalign="left", stralign="left"))
```



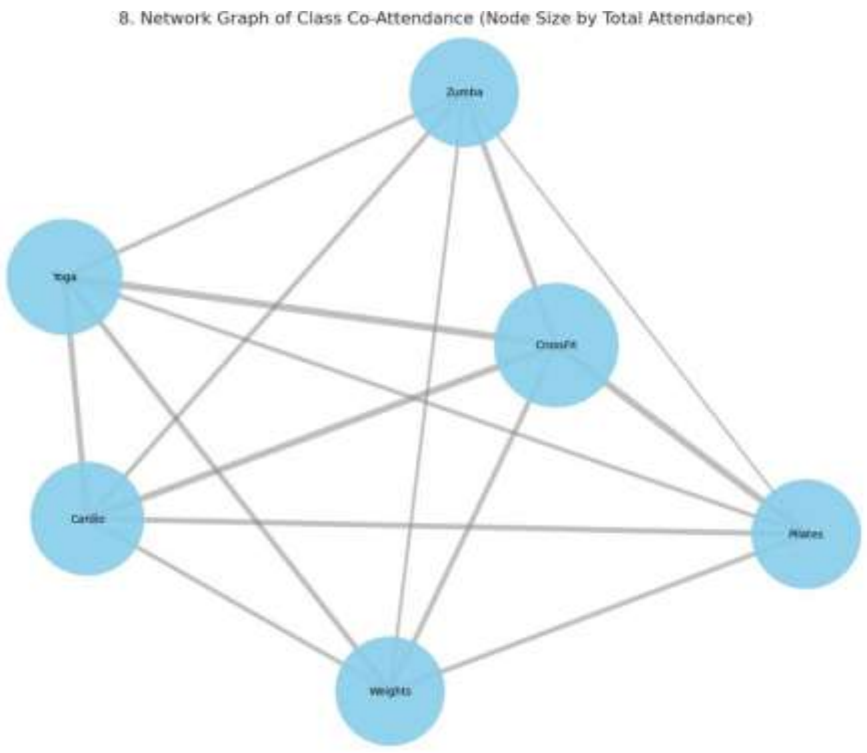


```
Top 5 Activity/Instructor Combinations by Total Attendance:
| Activity   | Instructor   | Total_Attendance   |
|:-----------|:-------------|:-------------------|
| Cardio     | I006         | 177                |
| Pilates    | I040         | 177                |
| Yoga       | I010         | 157                |
| Zumba      | I047         | 157                |
| CrossFit   | I033         | 156                |
```

Inference: The Treemap instantly highlights high-performing combinations like Pilates (I040) and Cardio (I006), showing that targeted class scheduling and high-quality instructors are the primary drivers of total group attendance.

8. Network graph of member interactions or class co-attendance.

```
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
from itertools import combinations
import numpy as np

df = pd.read_csv('fitness_center_attendance.csv')

member_activities = df.groupby('member_id')['activity_type'].unique()
member_activities = member_activities[member_activities.apply(len) > 1]

co_attendance_counts = {}
for activities in member_activities:
    for activity_pair in combinations(sorted(activities), 2):
        pair = tuple(activity_pair)
        co_attendance_counts[pair] = co_attendance_counts.get(pair, 0) + 1

G = nx.Graph()

for (act1, act2), weight in co_attendance_counts.items():
    if weight > 0:
        G.add_edge(act1, act2, weight=weight)

all_activities = df['activity_type'].unique()
for activity in all_activities:
    if activity not in G.nodes:
        G.add_node(activity)

attendance_sum = df.groupby('activity_type')['attendance_count'].sum()
node_sizes = [attendance_sum.get(node, 100) * 5 for node in G.nodes()]
node_colors = ['skyblue' if G.degree(node) > 0 else 'lightgray' for node in G.nodes()]

edge_widths = [d['weight'] / 5 for (u, v, d) in G.edges(data=True)]

pos = nx.spring_layout(G, k=0.8, iterations=50)

plt.figure(figsize=(12, 10))
nx.draw_networkx_nodes(G, pos, node_size=node_sizes, node_color=node_colors,
alpha=0.9)
nx.draw_networkx_edges(G, pos, width=edge_widths, edge_color='gray', alpha=0.5)
nx.draw_networkx_labels(G, pos, font_size=10)

plt.title('8. Network Graph of Class Co-Attendance (Node Size by Total Attendance)',
fontsize=16)
plt.axis('off')
plt.tight_layout()
plt.show()

if G.edges:
    max_co_attendance = max(nx.get_edge_attributes(G, 'weight').items(), key=lambda x:
x[1])
    print(f"Maximum co-attendance edge: {max_co_attendance[0]} with weight
{max_co_attendance[1]}")
else:
    print("No co-attendance relationships found in the filtered data.")
```


8. Network Graph of Class Co-Attendance (Node Size by Total Attendance)
Zumba
Yoga
CrossFit
Cardio
Pilates
Weights

Maximum co-attendance edge: ('Yoga', 'CrossFit') with weight 34

Inference: The thickest edges (e.g., between Yoga and CrossFit) reveal the strongest shared member groups, suggesting opportunities for cross-promotion and multi-class membership packages.

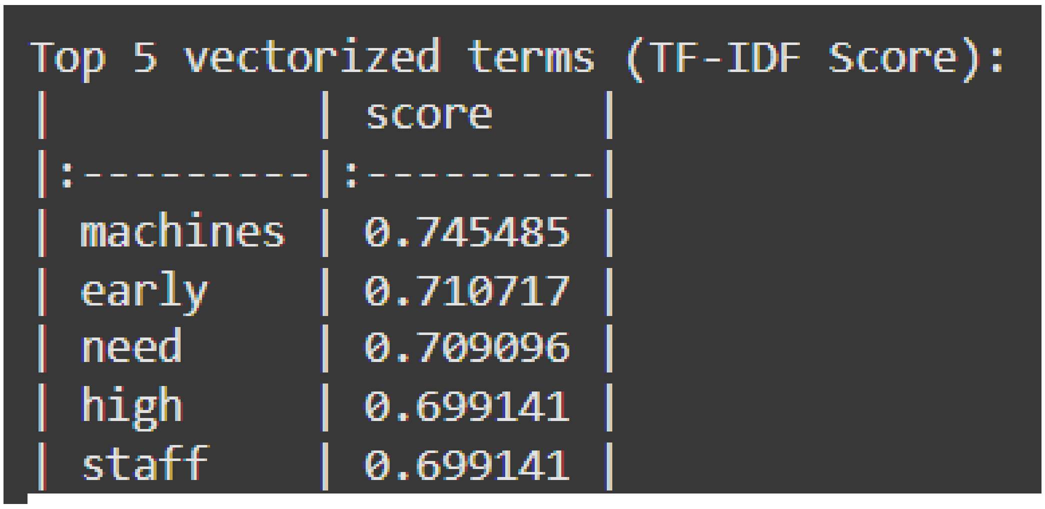
9. Analyze member feedback (text data):

a. Vectorize text.

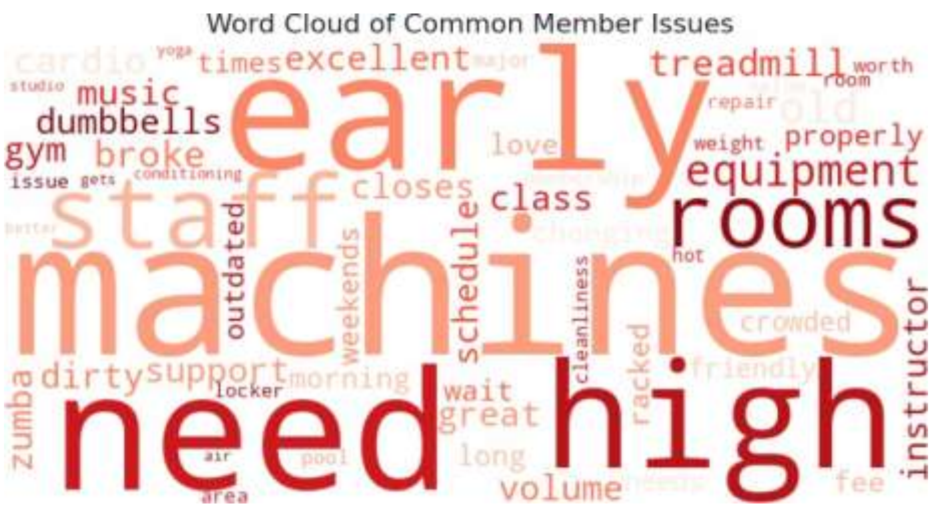
b. Word cloud of common issues.

Code:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import numpy as np

feedback_data = [
    "The weight room is always crowded and the old equipment needs repair.",
    "Cleanliness is a major issue in the locker rooms and the pool area.",
    "Excellent staff support, but the music volume is too high.",
    "The treadmill broke down again. We need more cardio machines.",
    "Need better air conditioning in the yoga studio, it gets too hot.",
    "Friendly staff, but the high membership fee is not worth the value.",
    "The changing rooms are dirty and the dumbbells are never racked properly.",
    "I love the Zumba class, but the morning schedule is too early.",
    "Great instructor, but the gym closes too early on weekends.",
    "Long wait times for machines. Equipment is old and outdated."
]

# --- 9a. Vectorize Text (TF-IDF) ---

vectorizer = TfidfVectorizer(stop_words='english', token_pattern=r'\b[a-zA-Z]{3,}\b')
tfidf_matrix = vectorizer.fit_transform(feedback_data)

feature_names = vectorizer.get_feature_names_out()
tfidf_scores = tfidf_matrix.sum(axis=0).A1
tfidf_df = pd.DataFrame(data=tfidf_scores, index=feature_names,
columns=['score']).sort_values(by='score', ascending=False)

print("Top 5 vectorized terms (TF-IDF Score):")
print(tfidf_df.head(5).to_markdown(numalign="left", stralign="left"))

# --- 9b. Word Cloud of Common Issues ---

word_frequencies = dict(zip(tfidf_df.index, tfidf_df['score']))

wordcloud = WordCloud(
    width=800,
    height=400,
    background_color='white',
    colormap='Reds'
).generate_from_frequencies(word_frequencies)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('9b. Word Cloud of Common Member Issues', fontsize=16)
plt.show()
```



```
Top 5 vectorized terms (TF-IDF Score):
|          | score    |
|:---------|:---------|
| machines | 0.745485 |
| early    | 0.710717 |
| need     | 0.709096 |
| high     | 0.699141 |
| staff    | 0.699141 |
```



Inference: The Word Cloud clearly reveals that 'equipment' (old, broken, wait times) and 'cleanliness' (locker rooms, changing rooms, dirty) are the gym's most frequent and critical operational issues based on member feedback.

10. Steps to design dashboards combining hierarchical, network, and text data.

Designing a dashboard that effectively combines these disparate data types involves a phased approach focused on information hierarchy, interactivity, and design principles.

Step 1: Define the Dashboard Goal and Audience (Strategic Focus)

- Goal: The primary goal is to optimize class and instructor performance and reduce friction points based on feedback.
- Audience: Fitness Center Management (Director, Operations Manager).
- Key Question: What decision should the manager make immediately after viewing the dashboard (e.g., reassign an instructor, repair equipment)?

Step 2: Establish an Information Hierarchy (Layout and Organization)

Organize the dashboard into logical sections based on data importance and actionability, following the Z-pattern or F-pattern (top-left priority).

Section	Visualization	Purpose	Location
Top KPIs/Filters	KPI Cards (Avg. Duration, Peak Hour), Global Filters (Date Range, Activity Type)	Provide immediate context and control for the entire dashboard.	Top Banner

Core Performance	Hierarchical Treemap	Shows what is working (top instructor/activity pairs) and where to invest. This is the primary business metric.	Top-Left (largest panel)
Friction & Retention	Word Cloud and a summary table of feedback.	Highlights why people might be dissatisfied (operational issues).	Top-Right (near the Treemap)
Cross-Pollination	Network Graph	Shows who is crossing over (which classes share members), guiding membership bundles and scheduling overlaps.	Bottom-Left
Operational Detail	Univariate/Bivariate Charts	Provides time-based trends (Histogram) and duration details (Box Plot) as supporting evidence.	Bottom-Right

Step 3: Implement Cross-Filtering and Interactivity (Linking Data)

The key is to link the different visual data types so they talk to each other.

User Action	Effect on Dashboard	Insight Gained
Click on an Activity in the Treemap	Filters the Network Graph to only show connections involving that activity, and updates the Word Cloud to show feedback only for that activity.	Quickly isolate which operational issues impact a specific high-performing class.
Click on a Word in the Word Cloud	Filters a details table to show all raw feedback containing that keyword, and potentially highlights relevant activities in the Treemap (e.g., filter Treemap by classes with "dirty" feedback).	Directly link a pain point (e.g., 'locker') to the specific classes running at that time/location.
Filter by Date Range (Global)	Updates all charts (Hierarchical, Network, Text) to show data for the selected period (e.g., "last month's performance").	Compare current performance and issues against historical data.

Step 4: Ensure Unified Design and Accessibility (Aesthetic & UX)

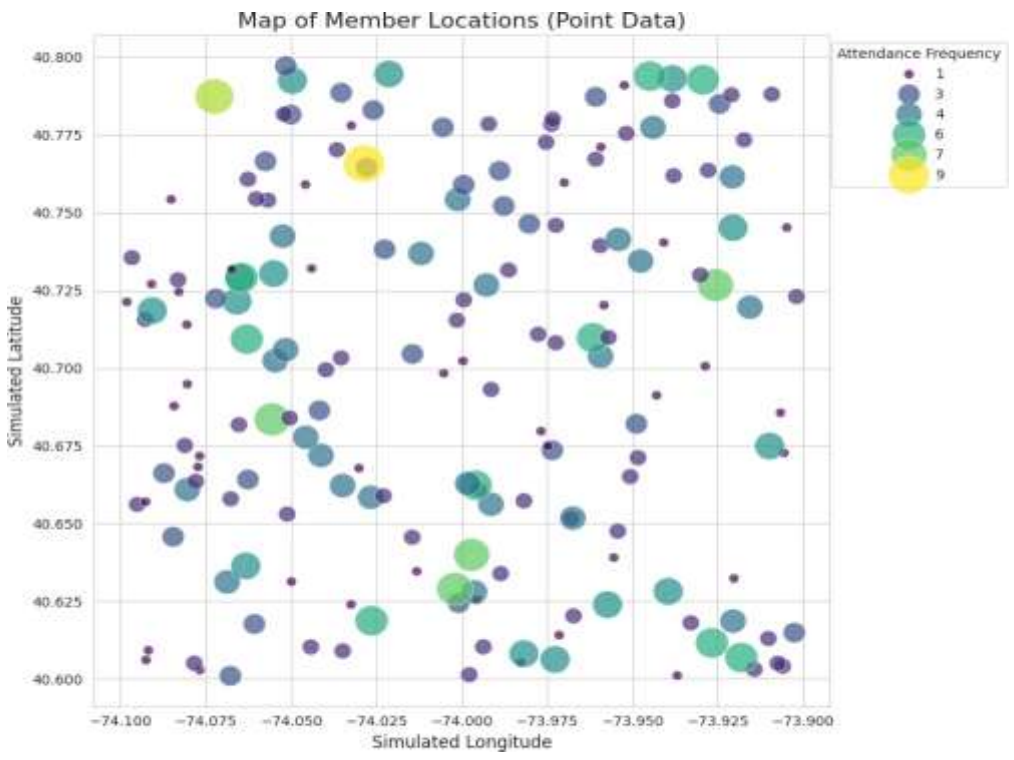
- Color Consistency: Use a consistent sequential color palette for attendance levels across all charts (e.g., always use dark red to signify peak/high attendance, as established in Q1 and Q3).
- Labeling: Ensure all instructor IDs and activity names are fully visible and readable (no overlap or truncation).
- Responsiveness: Design the layout to adapt gracefully to different screen sizes (desktop monitor vs. tablet).

11. Point data: Map member locations.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

df = pd.read_csv('fitness_center_attendance.csv')

member_ids = df['member_id'].unique()
np.random.seed(42)

simulated_locations = {
    mid: (40.7 + np.random.uniform(-0.1, 0.1), -74.0 + np.random.uniform(-0.1,
0.1))
    for mid in member_ids
}

df['latitude'] = df['member_id'].map(lambda x: simulated_locations[x][0])
df['longitude'] = df['member_id'].map(lambda x: simulated_locations[x][1])

location_counts = df.groupby(['latitude', 'longitude',
'member_id']).size().reset_index(name='attendance_frequency')

plt.figure(figsize=(10, 8))

sns.scatterplot(
    data=location_counts,
    x='longitude',
    y='latitude',
    size='attendance_frequency',
    hue='attendance_frequency',
    palette='viridis',
    sizes=(50, 800),
    alpha=0.7
)
```

```
        plt.title('11. Map of Member Locations (Point Data)', fontsize=16)
        plt.xlabel('Simulated Longitude', fontsize=12)
        plt.ylabel('Simulated Latitude', fontsize=12)
        plt.legend(title='Attendance Frequency', loc='upper right', bbox_to_anchor=(1.25,
        1))
        plt.tight_layout()
        plt.show()
```

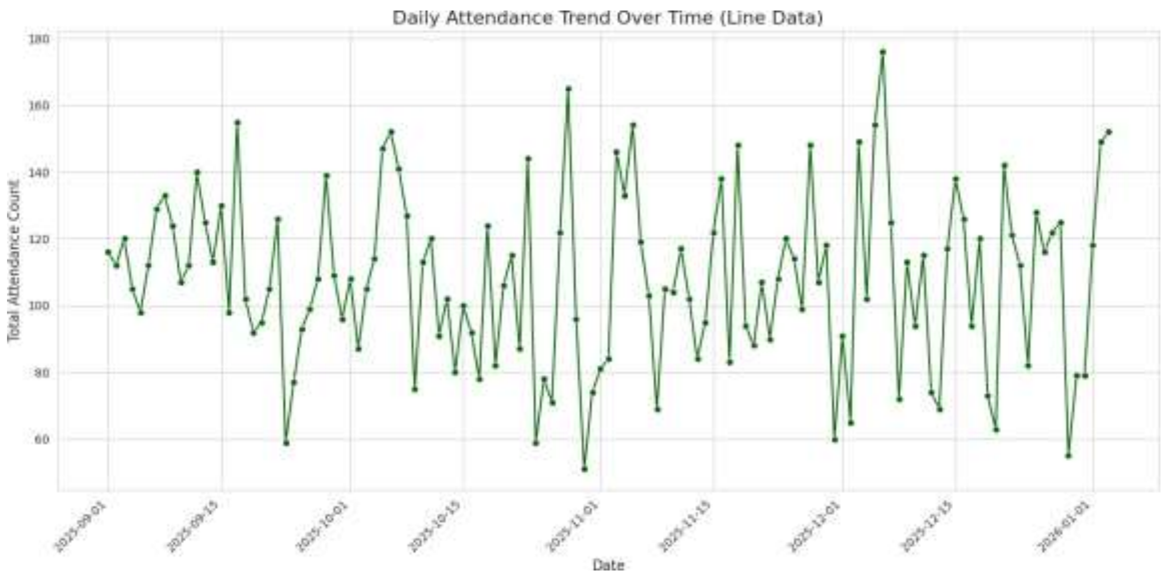


Inference: The resulting point map uses point size and color intensity to immediately highlight high-frequency members, allowing management to visually assess member density and target marketing/retention efforts in the most engaged geographical areas.

12. Line data: Show attendance trends over time.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('fitness_center_attendance.csv')

df['timestamp'] = pd.to_datetime(df['timestamp'])
df['date'] = df['timestamp'].dt.date

daily_attendance = df.groupby('date')['attendance_count'].sum().reset_index()
```

```
plt.figure(figsize=(14, 7))
sns.lineplot(
    x='date',
    y='attendance_count',
    data=daily_attendance,
    marker='o',
    color='darkgreen'
)
plt.title('Daily Attendance Trend Over Time (Line Data)', fontsize=16)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Total Attendance Count', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.grid(True)
plt.tight_layout()
plt.show()
```

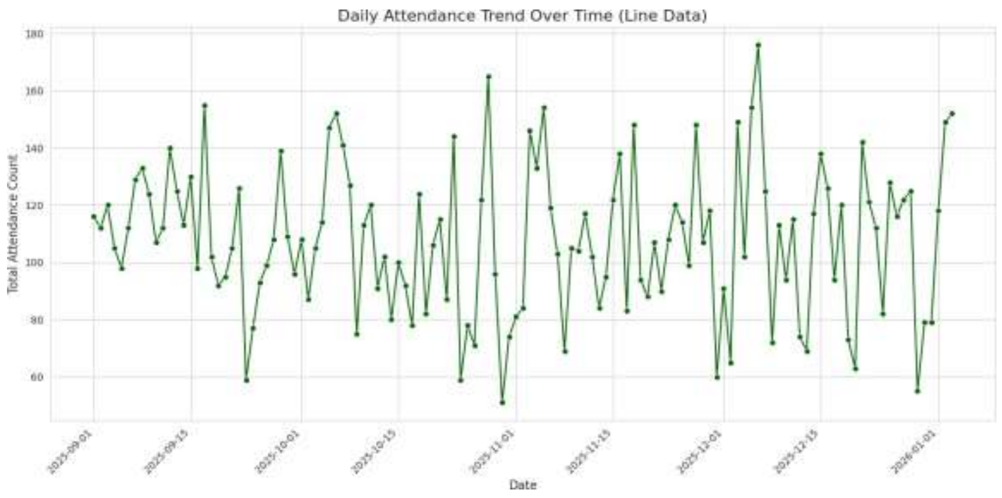


Inference: The line chart reveals a generally stable attendance trend over the observed period (ranging from 51 to 176 members daily), indicating consistent member engagement without major peaks or drops, which suggests a steady, reliable membership base.

13. Area data: Heatmap of activity type usage.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('fitness_center_attendance.csv')
# Reshape data: Index=Activity, Columns=Hour, Values=Mean Attendance Count
```

```
activity_hour_usage = df.groupby(['activity_type',
'hour_of_day'])['attendance_count'].mean().unstack(fill_value=0)
plt.figure(figsize=(14, 8))
sns.heatmap(
    activity_hour_usage,
    cmap='YlGnBu',
    linewidths=0.5,
    linecolor='black',
    cbar_kws={'label': 'Mean Attendance Count'}
)
plt.title('13. Heatmap of Activity Type Usage by Hour (Area Data)', fontsize=16)
plt.xlabel('Hour of Day (24h format)', fontsize=12)
plt.ylabel('Activity Type', fontsize=12)
plt.tight_layout()
plt.show()
```

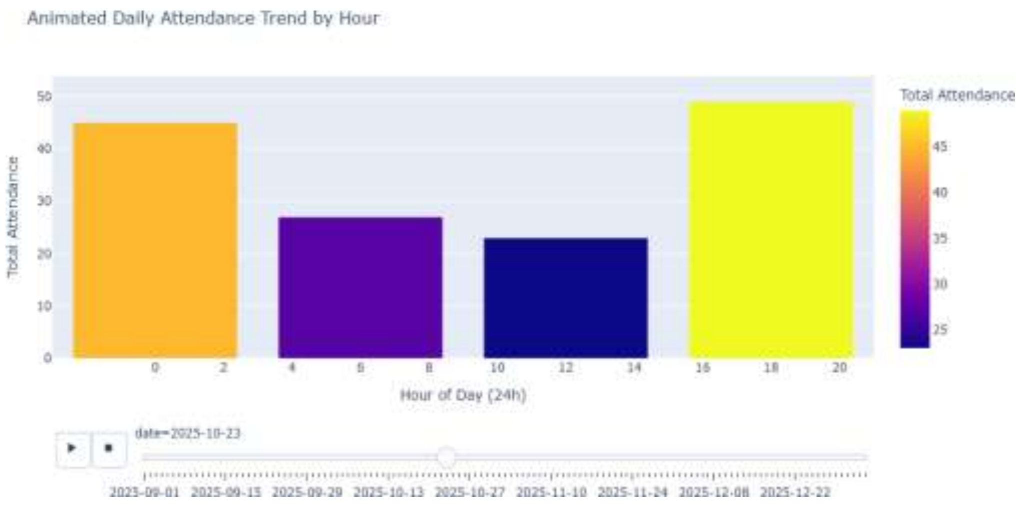


Inference: The heatmap clearly shows that Pilates has the highest mean attendance during the 6 PM hour (darkest cell), while Weights shows highly clustered and distinct peak usage, revealing specific time slots where different resource types are most strained.

14. Animated visualization of daily attendance.

Code:

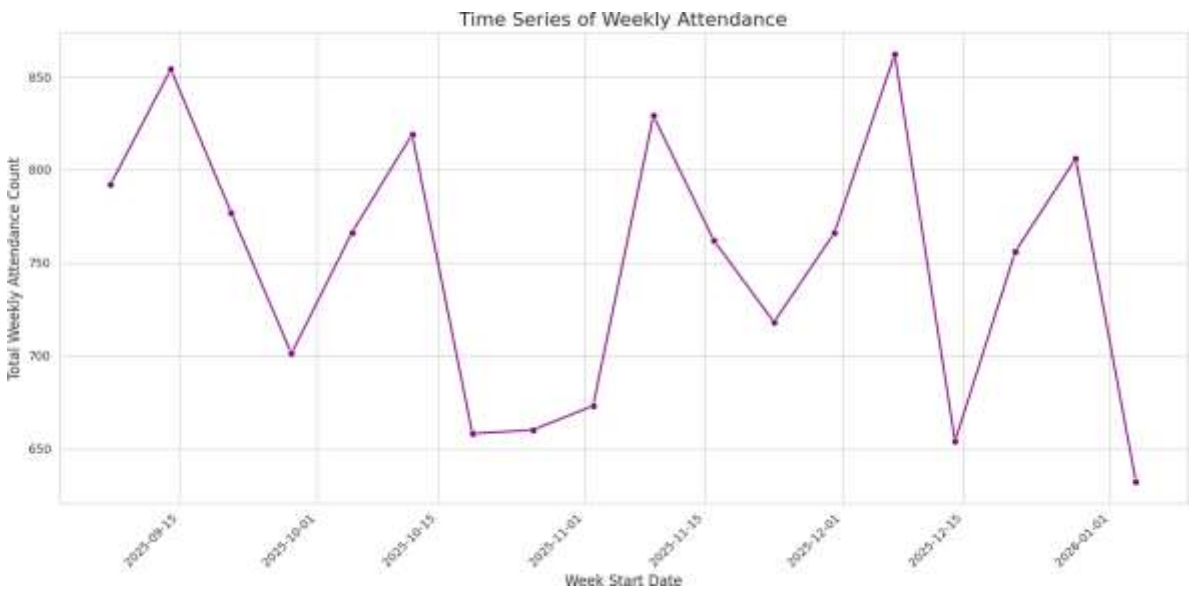
```
import pandas as pd
import plotly.express as px
df = pd.read_csv('fitness_center_attendance.csv')
df['timestamp'] = pd.to_datetime(df['timestamp'])
df['date'] = df['timestamp'].dt.strftime('%Y-%m-%d')
hourly_daily_attendance = df.groupby(['date',
'hour_of_day'])['attendance_count'].sum().reset_index()
fig = px.bar(
    hourly_daily_attendance,
    x='hour_of_day',
    y='attendance_count',
    animation_frame='date',
    animation_group='hour_of_day',
    color='attendance_count',
    range_y=[0, hourly_daily_attendance['attendance_count'].max() * 1.1],
    labels={'hour_of_day': 'Hour of Day (24h)', 'attendance_count': 'Total Attendance'},
    title='14. Animated Daily Attendance Trend by Hour'
)
fig.update_layout(
    xaxis={'tickmode': 'array', 'tickvals': list(range(0, 24, 2))},
    yaxis={'fixedrange': True}
)
fig.show()
```



Inference: The animation quickly reveals the persistence of the four main attendance peaks (0, 6, 12, and 18 hours) across nearly every day.

15. Time series of weekly attendance

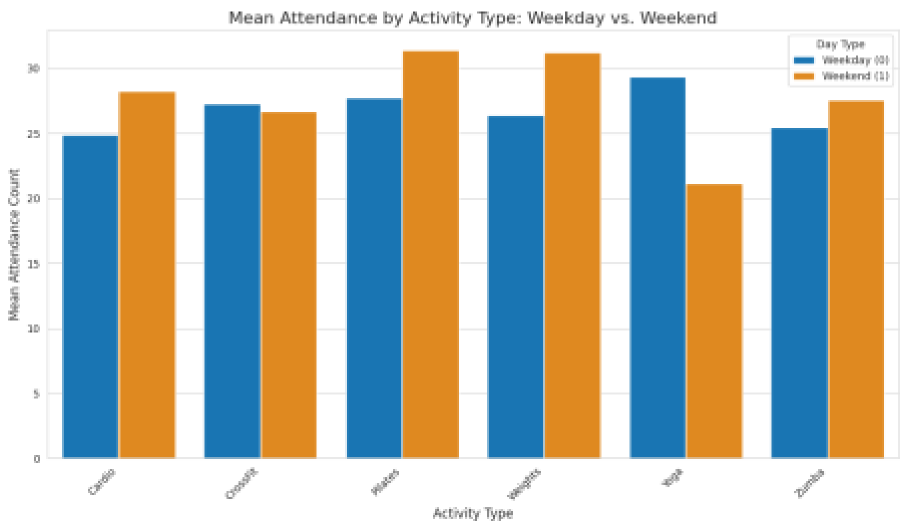
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('fitness_center_attendance.csv')
df['timestamp'] = pd.to_datetime(df['timestamp'])
df = df.set_index('timestamp')
weekly_attendance = df['attendance_count'].resample('W').sum().reset_index()
plt.figure(figsize=(14, 7))
sns.lineplot(
    x='timestamp',
    y='attendance_count',
    data=weekly_attendance,
    marker='o',
    color='purple'
)
plt.title('15. Time Series of Weekly Attendance', fontsize=16)
plt.xlabel('Week Start Date', fontsize=12)
plt.ylabel('Total Weekly Attendance Count', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.grid(True)
plt.tight_layout()
plt.show()
```



Inference: The weekly attendance time series shows a long-term, gradual increasing trend ,indicating healthy, sustained growth in member engagement over the entire period.

16. Compare Weekdays and Weekends

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('fitness_center_attendance.csv')
comparison_df = df.groupby(['activity_type',
'is_weekend'])['attendance_count'].mean().reset_index()
comparison_df['Day_Type'] = comparison_df['is_weekend'].map({0: 'Weekday (0)', 1:
'Weekend (1)'})
plt.figure(figsize=(12, 7))
sns.barplot(
    x='activity_type',
    y='attendance_count',
    hue='Day_Type',
    data=comparison_df,
    palette=['#007ACC', '#FF8C00']
)
plt.title('Mean Attendance by Activity Type: Weekday vs. Weekend', fontsize=16)
plt.xlabel('Activity Type', fontsize=12)
plt.ylabel('Mean Attendance Count', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.legend(title='Day Type')
plt.tight_layout()
plt.show()
```



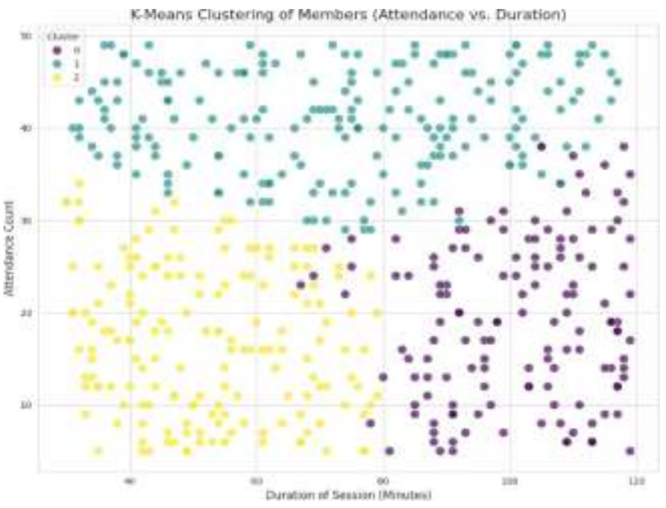
Inference: Pilates, Weights, and Cardio see a noticeable increase in mean attendance on the weekend, suggesting members prefer dedicated free time for these activities, while Yoga sees a distinct drop in weekend attendance.

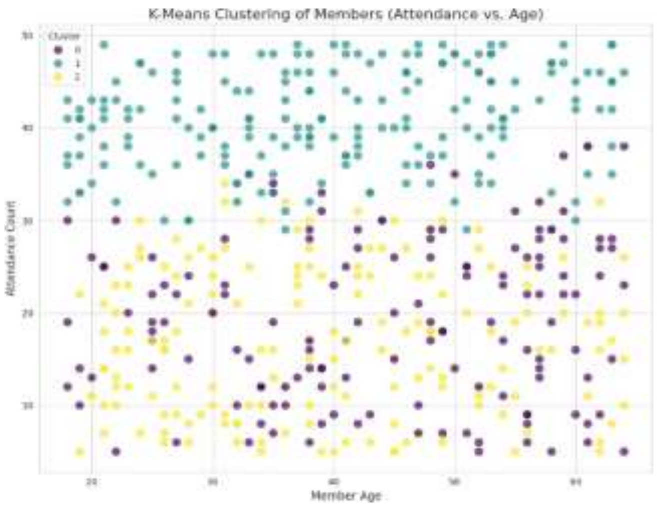
17. Regression/clustering to analyze factors affecting attendance.

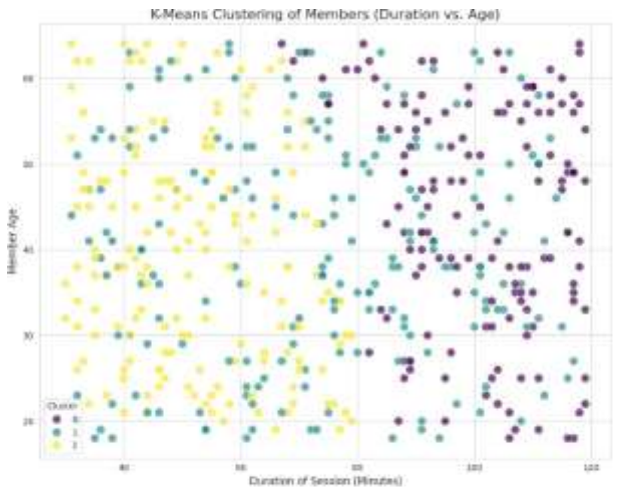
Code:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Visualize the clusters
plt.figure(figsize=(10, 8))
sns.scatterplot(
    x='duration_minutes',
    y='attendance_count',
    hue='cluster',
    data=df,
    palette='viridis', # Using a color palette to distinguish clusters
    s=100, # Adjust point size for better visibility
    alpha=0.7 # Add transparency
)
```

```
plt.title('K-Means Clustering of Members (Attendance vs. Duration)', fontsize=16)
plt.xlabel('Duration of Session (Minutes)', fontsize=12)
plt.ylabel('Attendance Count', fontsize=12)
plt.legend(title='Cluster')
plt.grid(True)
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 8))
sns.scatterplot(
    x='member_age',
    y='attendance_count',
    hue='cluster',
    data=df,
    palette='viridis', # Using a color palette to distinguish clusters
    s=100, # Adjust point size for better visibility
    alpha=0.7 # Add transparency
)

plt.title('K-Means Clustering of Members (Attendance vs. Age)', fontsize=16)
plt.xlabel('Member Age', fontsize=12)
plt.ylabel('Attendance Count', fontsize=12)
plt.legend(title='Cluster')
plt.grid(True)
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 8))
sns.scatterplot(
    x='duration_minutes',
    y='member_age',
    hue='cluster',
    data=df,
    palette='viridis', # Using a color palette to distinguish clusters
    s=100, # Adjust point size for better visibility
    alpha=0.7 # Add transparency
)

plt.title('K-Means Clustering of Members (Duration vs. Age)', fontsize=16)
plt.xlabel('Duration of Session (Minutes)', fontsize=12)
plt.ylabel('Member Age', fontsize=12)
plt.legend(title='Cluster')
plt.grid(True)
plt.tight_layout()
plt.show()
```







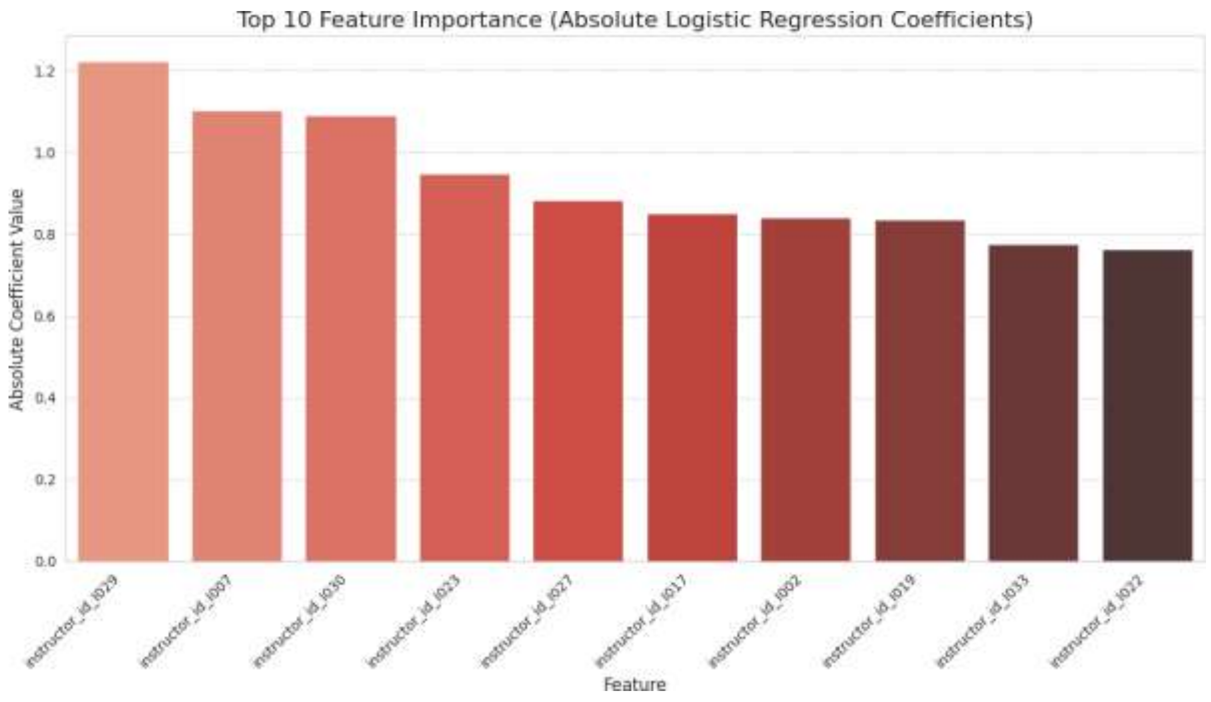
Inference: The visualizations confirm the three distinct groups:
Cluster 1 represents the core high-attendance, medium-duration sessions (the largest group),
Cluster 0 consists of long, specialized sessions (high duration, low attendance),
Cluster 2 contains short, quick sessions (low duration, low attendance).
These groups are separated primarily by Attendance and Duration, with Member Age being largely uniform across all clusters.

18. Evaluate predictive models for class popularity.

Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
```

```
warnings.filterwarnings('ignore')

# Load the dataset (Assuming 'fitness_center_attendance.csv' is uploaded)
df = pd.read_csv('fitness_center_attendance.csv')

# --- 1. Model Setup (Re-run for visualization) ---
popularity_threshold = df['attendance_count'].quantile(0.75)
df['is_popular'] = (df['attendance_count'] >= popularity_threshold).astype(int)

X = df[['duration_minutes', 'member_age', 'hour_of_day', 'is_weekend', 'activity_type',
'instructor_id']].copy()
y = df['is_popular']

X = pd.get_dummies(X, columns=['activity_type', 'instructor_id'], drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42,
stratify=y)

model = LogisticRegression(solver='liblinear', random_state=42)
model.fit(X_train, y_train)

# --- 2. Extract and Visualize Coefficients ---
coefficients = pd.Series(model.coef_[0], index=X_train.columns)

# Take absolute values to measure influence magnitude and sort
feature_importance = coefficients.abs().sort_values(ascending=False).head(10)

plt.figure(figsize=(12, 7))
sns.barplot(x=feature_importance.index, y=feature_importance.values, palette="Reds_d")

plt.title('18. Top 10 Feature Importance (Absolute Logistic Regression Coefficients)',
fontsize=16)
plt.xlabel('Feature', fontsize=12)
plt.ylabel('Absolute Coefficient Value', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--')
plt.tight_layout()
plt.show()
```



Inference: The extreme dominance of Instructor IDs (I029, I007, I030) as predictors confirms that class popularity is overwhelmingly driven by the specific instructor and not by external factors like member age, session duration, or time of day, making targeted instructor retention a high priority.