In [155]:
```
        ###      L O A D I N G   A N D   I N S P E C T I N G   D A T A     #

#importing all libraries and loading the data set using pandas:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid", palette="deep")


df = pd.read_csv("C:\\Users\\DELL\\Documents\\For Flexisaf\\attendance_logs.csv
```

In [156]:
```
df.head()  #previewing rows
```

1	2	202	1202	English Literature	602	James
2	3	203	1203	Physics 101	603	Priya
3	4	204	1201	Math 101	604	Victor
4	5	205	1204	History 201	605	Elena

In [157]:
```
print(df)  # printing all data
```

	attendance_id	session_id	class_id	class_name	student_id
0	1	201	1201	Math 101	601
1	2	202	1202	English Literature	602
2	3	203	1203	Physics 101	603
3	4	204	1201	Math 101	604
4	5	205	1204	History 201	605
..	...	...	...	...	...
195	196	259	1212	Advanced Programming	846
196	197	260	1213	Physical Education	847
197	198	261	1207	Computer Science 101	848
198	199	262	1204	Physics 101	849
199	200	263	1209	Chemistry 202	850

	student_first_name	student_last_name	student_email
0	Aisha	Khan	aisha.khan@mathacademy.org
1	James	Nguyen	james.nguyen@literaturehub.edu
2	Priya	Singh	priya.singh@physicslearn.com
3	Victor	Chen	victor.chen@mathacademy.org
4	Elena	Sharma	elena.sharma@historyzone.net
..	...	...	...
195	Marta	Santos	marta.santos@progstudy.org
196	Ivan	Petrov	ivan.petrov@physed.edu
197	Elise	Martin	elise.martin@compsci.edu
198	Leila	Benali	leila.benali@physicszone.org
199	Paolo	Romano	paolo.romano@chemistrylab.org

	attendance_date	session_start_time	session_end_time
0	5/6/2024	2024-05-06T08:00:00	2024-05-06T09:30:00
1	5/7/2024	2024-05-07T13:00:00	2024-05-07T14:30:00
2	5/8/2024	2024-05-08T10:00:00	2024-05-08T12:00:00
3	5/9/2024	2024-05-09T09:30:00	2024-05-09T11:00:00
4	5/10/2024	2024-05-10T15:00:00	2024-05-10T16:30:00
..	...	...	...
195	11/30/2023	2023-11-30T15:00:00	2023-11-30T16:30:00
196	12/1/2023	2023-12-01T07:00:00	2023-12-01T08:00:00
197	12/1/2023	2023-12-01T09:00:00	2023-12-01T10:30:00
198	12/1/2023	2023-12-01T13:00:00	2023-12-01T14:30:00
199	12/2/2023	2023-12-02T09:00:00	2023-12-02T10:30:00

	attendance_status	join_time	leave_time
0	Present	2024-05-06T08:01:40	2024-05-06T09:30:10
1	Absent	NaN	NaN
2	Late	2024-05-08T10:29:14	2024-05-08T12:00:00
3	Present	2024-05-09T09:32:12	2024-05-09T11:01:25
4	Excused	NaN	NaN
..	...	...	...
195	Present	2023-11-30T15:00:00	2023-11-30T16:30:00
196	Present	2023-12-01T07:00:15	2023-12-01T08:00:00
197	Present	2023-12-01T09:02:06	2023-12-01T10:30:00
198	Excused	NaN	NaN
199	Present	2023-12-02T09:00:00	2023-12-02T10:30:00

	engagement_score	notes
0	0.91	Actively participated in group problem solving...
1	NaN	Absent without prior notification, follow-up n...
2	0.36	Joined half an hour late, minimal engagement d...
3	0.79	Answered multiple questions, showed clear unde...

```
4                NaN       Excused for illness, documentation received.
..               ...                                                  ...
195             0.89  Created a sample web app as part of the exercise.
196             0.88   Participated in all physical fitness activities.
197             0.83  Demonstrated an algorithm and answered questions.
198              NaN             Excused for competition participation.
199             0.92  Led experiment demonstration, encouraged class...

[200 rows x 16 columns]
```

In [16]:
```
df.isnull().values.any() # checking if any missing values exist
```

Out[16]:
```
True
```

In [17]:
```
df.isnull().sum().sum()  # total missing values
```

Out[17]:
```
180
```

In [18]:
```
df.isnull().sum() # missing values per column
```

Out[18]:
```
attendance_id          0
session_id             0
class_id               0
class_name             0
student_id             0
student_first_name     0
student_last_name      0
student_email          0
attendance_date        0
session_start_time     0
session_end_time       0
attendance_status      0
join_time             60
leave_time            60
engagement_score      60
notes                  0
dtype: int64
```

In [19]:
```
df.info()  # dataset summary
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   attendance_id       200 non-null    int64
 1   session_id          200 non-null    int64
 2   class_id            200 non-null    int64
 3   class_name          200 non-null    object
 4   student_id          200 non-null    int64
 5   student_first_name  200 non-null    object
 6   student_last_name   200 non-null    object
 7   student_email       200 non-null    object
 8   attendance_date     200 non-null    object
 9   session_start_time  200 non-null    object
 10  session_end_time    200 non-null    object
 11  attendance_status   200 non-null    object
 12  join_time           140 non-null    object
 13  leave_time          140 non-null    object
 14  engagement_score    140 non-null    float64
 15  notes               200 non-null    object
dtypes: float64(1), int64(4), object(11)
memory usage: 25.1+ KB
```

In [158]:
```
#cleaning the data by dropping rows with any missing data
df_clean = df.dropna()
df_clean.head()
```

Out[158]:

	attendance_id	session_id	class_id	class_name	student_id	student_first_name	student_last_
0	1	201	1201	Math 101	601	Aisha	
2	3	203	1203	Physics 101	603	Priya	
3	4	204	1201	Math 101	604	Victor	
5	6	206	1205	Art & Design	606	Lucas	D
8	9	209	1207	Computer Science 101	609	Sofia	C

In [27]:
```
df_clean.isnull().values.any()   # checking if any missing values still exist
```

Out[27]:
```
False
```

In [160]:
```
#                       P A R S I N G   T I M E  S T A M P S

# Converting join/leave times to datetime objects
df_clean["join_time"] = pd.to_datetime(df_clean["join_time"])
df_clean["leave_time"] = pd.to_datetime(df_clean["leave_time"])

# Computing session duration in minutes
df_clean["duration_minutes"] = (df_clean["leave_time"] - df_clean["join_time"])

df_clean = df_clean[df_clean["leave_time"] >= df_clean["join_time"]]

# Extracting useful time features
df_clean["date"] = df_clean["join_time"].dt.date        # calendar date
df_clean["day_name"] = df_clean["join_time"].dt.day_name()  # weekday name
df_clean["hour"] = df_clean["join_time"].dt.hour        # hour of day (0-23)

# Previewng transformed dataset
df_clean.head()
```

Out[160]:

	attendance_id	session_id	class_id	class_name	student_id	student_first_name	student_last_
0	1	201	1201	Math 101	601	Aisha	
2	3	203	1203	Physics 101	603	Priya	
3	4	204	1201	Math 101	604	Victor	
5	6	206	1205	Art & Design	606	Lucas	D
8	9	209	1207	Computer Science 101	609	Sofia	C

In [229]:
```
#                                   S T U D E N T - L E V E L   M E T R I C S

#Sessions attended & average duration
sessions_per_student = df_clean.groupby("student_id")["session_id"].nunique().r
avg_duration_per_student = df_clean.groupby("student_id")["duration_minutes"].m

student_metrics = pd.concat([sessions_per_student, avg_duration_per_student], a
student_metrics.head()
```

Out[229]:

	student_id	sessions_attended	avg_duration_min
0	551	1	91.8
1	552	1	70.9
2	553	1	89.2
3	555	1	88.8
4	557	1	88.8

In [230]:
```
#attendance rate
total_sessions = df_clean["session_id"].nunique()
student_metrics["attendance_rate_pct"] = (student_metrics["sessions_attended"]
student_metrics.head()
```

Out[230]:

	student_id	sessions_attended	avg_duration_min	attendance_rate_pct
0	551	1	91.8	1.2
1	552	1	70.9	1.2
2	553	1	89.2	1.2
3	555	1	88.8	1.2
4	557	1	88.8	1.2

In [231]:
```
#                    D A Y   &   S E S S I O N - L E V E L   M E T R I C S

#activity by day
activity_by_day = df_clean.groupby("day_name").agg(
    total_joins=("student_id","count"),
    total_minutes=("duration_minutes","sum")
).reset_index()

activity_by_day.sort_values("total_joins", ascending=False)
```

Out[231]:

	day_name	total_joins	total_minutes
6	Wednesday	25	2054.100000
2	Saturday	23	1968.216667
5	Tuesday	23	1955.050000
0	Friday	21	1702.133333
4	Thursday	19	1496.900000
1	Monday	17	1397.950000
3	Sunday	12	1103.866667

In [232]:
```
#session metrics

session_metrics = df_clean.groupby("session_id").agg(
    unique_students=("student_id","nunique"),
    total_minutes=("duration_minutes","sum"),
    avg_duration_min=("duration_minutes","mean")
).reset_index()

session_metrics.head()
```

Out[232]:

	session_id	unique_students	total_minutes	avg_duration_min
0	121	2	162.65	81.325
1	122	1	89.25	89.250
2	123	1	88.75	88.750
3	124	1	88.75	88.750
4	125	1	89.80	89.800

In [233]:
```
#most active days of the week
# Count how many students attended on each weekday
attendance_by_weekday = df_clean.groupby("day_name").size()

attendance_by_weekday = attendance_by_weekday.sort_values(ascending=False)
print("Most Active Days of the Week:")
print(attendance_by_weekday)
```

```
Most Active Days of the Week:
day_name
Wednesday    25
Saturday     23
Tuesday      23
Friday       21
Thursday     19
Monday       17
Sunday       12
dtype: int64
```

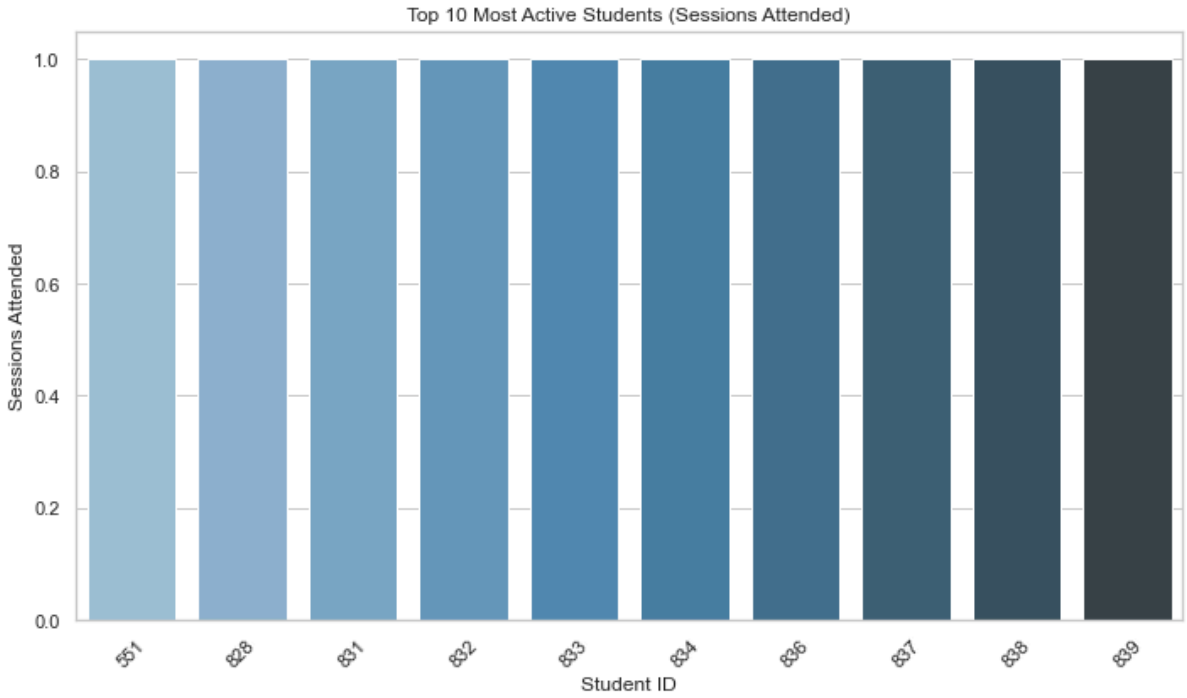
In [234]:
```
#              F L A G G I N G   L O W  - E N G A G E M E N T S
# applying thresholds
ATTENDANCE_THRESHOLD = 60.0
DURATION_THRESHOLD = 20.0

student_metrics["flag_low_attendance"] = student_metrics["attendance_rate_pct"]
student_metrics["flag_low_duration"] = student_metrics["avg_duration_min"] < DU
student_metrics["low_engagement_flag"] = student_metrics["flag_low_attendance"]

flagged_students = student_metrics[student_metrics["low_engagement_flag"]]
flagged_students.sort_values(["attendance_rate_pct","avg_duration_min"]).head(1
```

Out[234]:

	student_id	sessions_attended	avg_duration_min	attendance_rate_pct	flag_low_attendance
88	828	1	22.8	1.2	True
74	807	1	38.6	1.2	True
76	810	1	44.6	1.2	True
15	574	1	44.7	1.2	True
139	4050	1	45.0	1.2	True
40	612	1	45.6	1.2	True
102	845	1	47.5	1.2	True
99	841	1	51.4	1.2	True
28	594	1	56.1	1.2	True
39	610	1	56.5	1.2	True

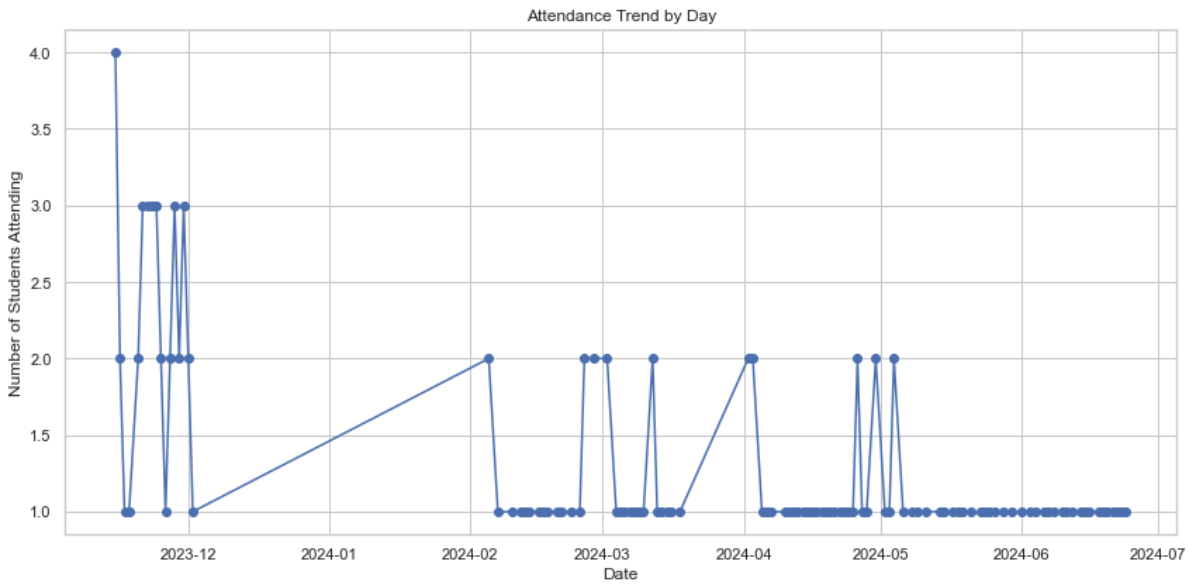
In [237]:
```
#                       V I S U A L I Z A T I O N S

#   BAR CHART OF TOP ACTIVE STUDENTS BY SESSIONS ATTENDED
# Top 10 most active students
active_students = student_metrics.sort_values("sessions_attended", ascending=Fa
print(active_students)
plt.figure(figsize=(10,6))
sns.barplot(data=active_students, x="student_id", y="sessions_attended", palett
plt.title("Top 10 Most Active Students (Sessions Attended)")
plt.xlabel("Student ID")
plt.ylabel("Sessions Attended")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
    student_id  sessions_attended  avg_duration_min  attendance_rate_pct  \
0          551                  1              91.8                  1.2
96         838                  1              88.7                  1.2
90         831                  1              88.3                  1.2
91         832                  1              88.9                  1.2
92         833                  1              90.0                  1.2
93         834                  1              89.2                  1.2
94         836                  1              90.0                  1.2
95         837                  1              88.9                  1.2
97         839                  1              90.0                  1.2
88         828                  1              22.8                  1.2

    flag_low_attendance  flag_low_duration  low_engagement_flag
0                  True              False                 True
96                 True              False                 True
90                 True              False                 True
91                 True              False                 True
92                 True              False                 True
93                 True              False                 True
94                 True              False                 True
95                 True              False                 True
97                 True              False                 True
88                 True              False                 True
```


Top 10 Most Active Students (Sessions Attended)
Sessions Attended
1.0
0.8
0.6
0.4
0.2
0.0
S51
S28
S31
S32
S33
S34
S36
S37
S38
S39
Student ID

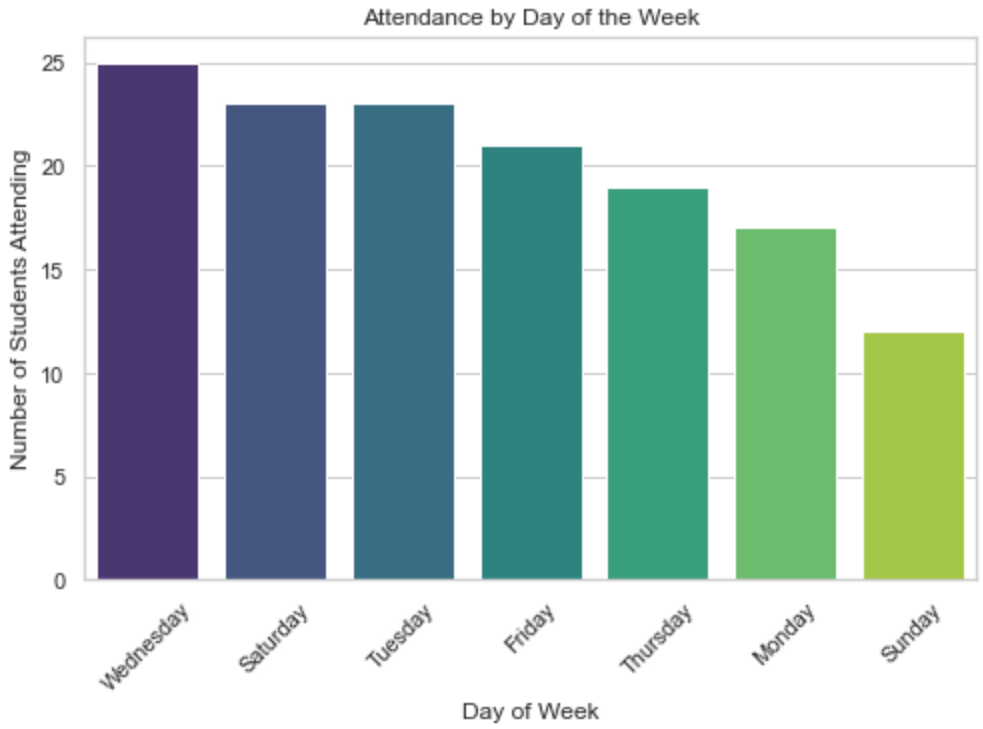
In [238]:
```
#LINE CHART OF DAILY ENGAGEMENT
# Group attendance logs by date
attendance_by_day = df_clean.groupby(df_clean["join_time"].dt.date).size()
print(attendance_by_day)

plt.figure(figsize=(12,6))
attendance_by_day.plot(kind="line", marker="o")
plt.title("Attendance Trend by Day")
plt.xlabel("Date")
plt.ylabel("Number of Students Attending")
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
join_time
2023-11-15    4
2023-11-16    2
2023-11-17    1
2023-11-18    1
2023-11-20    2
             ..
2024-06-20    1
2024-06-21    1
2024-06-22    1
2024-06-23    1
2024-06-24    1
Length: 109, dtype: int64
```



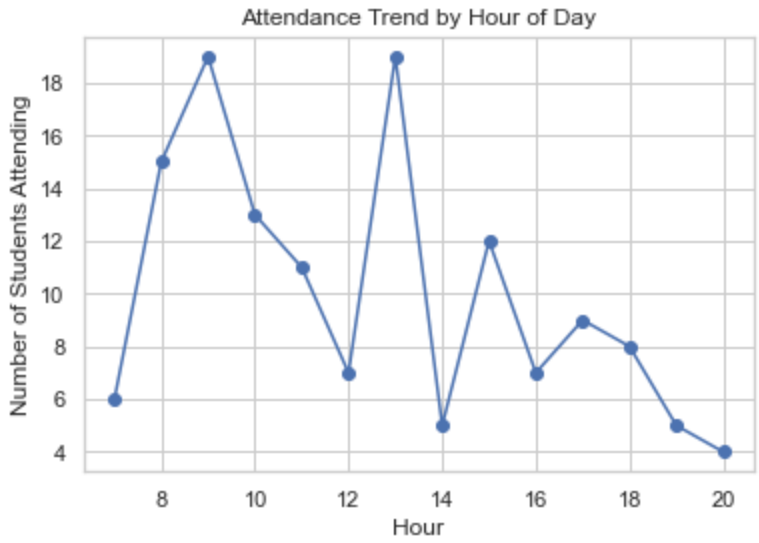
In [239]:
```
#BAR CHART OF MOST ACTIVE DAYS
attendance_by_weekday = df_clean.groupby("day_name").size()
attendance_by_weekday = attendance_by_weekday.sort_values(ascending=False)
print("Most Active Days of the Week:")
print(attendance_by_weekday)

plt.figure(figsize=(8,5))
sns.barplot(x=attendance_by_weekday.index, y=attendance_by_weekday.values, pale
plt.title("Attendance by Day of the Week")
plt.xlabel("Day of Week")
plt.ylabel("Number of Students Attending")
plt.xticks(rotation=45)
plt.show()
```

```
Most Active Days of the Week:
day_name
Wednesday    25
Saturday     23
Tuesday      23
Friday       21
Thursday     19
Monday       17
Sunday       12
dtype: int64
```



In [224]:
```
#LINE PLOT OF TIME OF DAY TRENDS BY HOUR
#line plot of hourly pattern
attendance_by_hour = df_clean.groupby(df_clean["join_time"].dt.hour).size()
print(attendance_by_hour)

attendance_by_hour.plot(kind="line", marker="o")
plt.title("Attendance Trend by Hour of Day")
plt.xlabel("Hour")
plt.ylabel("Number of Students Attending")
plt.show()
```

```
join_time
7      6
8     15
9     19
10    13
11    11
12     7
13    19
14     5
15    12
16     7
17     9
18     8
19     5
20     4
dtype: int64
```



In [168]:
```
#                R E S U L T S   E X P O R T
#saving metrics
student_metrics.to_csv("student_metrics.csv", index=False)
flagged_students.to_csv("flagged_students.csv", index=False)
session_metrics.to_csv("session_metrics.csv", index=False)
```

In [256]:
```
#       A I   F E A T U R E:   R U L E - B A S E D   E N G A G E M E N T   F L
#Defining thresholds: Students attending < 50% of sessions = low engagement.
#Compute percentage attendance

total_sessions = df_clean["session_id"].nunique()
attendance_rate = sessions_per_student / total_sessions
low_engagement = attendance_rate[attendance_rate < 0.5]
```

In [257]:
```
#OUTPUT FLAGGED STUDENTS
print("Low engagement students:", low_engagement.index.tolist())
```

```
Low engagement students: [551, 552, 553, 555, 557, 558, 560, 561, 563, 564, 5
66, 567, 569, 571, 573, 574, 575, 576, 578, 579, 581, 582, 585, 586, 587, 58
9, 591, 593, 594, 595, 596, 598, 599, 600, 601, 603, 604, 606, 609, 610, 612,
613, 614, 616, 618, 619, 620, 621, 623, 625, 627, 629, 630, 632, 633, 634, 63
6, 637, 638, 640, 641, 642, 644, 645, 646, 647, 648, 649, 650, 801, 802, 803,
804, 806, 807, 808, 810, 816, 817, 818, 819, 820, 822, 823, 824, 825, 826, 82
7, 828, 830, 831, 832, 833, 834, 836, 837, 838, 839, 840, 841, 842, 844, 845,
846, 847, 848, 850, 4001, 4002, 4004, 4005, 4007, 4008, 4010, 4013, 4014, 401
5, 4017, 4019, 4021, 4022, 4024, 4025, 4027, 4028, 4030, 4031, 4033, 4035, 40
37, 4038, 4039, 4040, 4041, 4043, 4044, 4046, 4047, 4049, 4050]
```

In [258]:
```
#Class Distribution Counts to show flagged low-engagement
print(student_metrics["low_engagement_flag"].value_counts())
```

```
True    140
Name: low_engagement_flag, dtype: int64
```

In [259]:

```
# # df_clean contains: student_id, session_id, join_time, leave_time, duration_

# Thresholds for flagging
ATTENDANCE_THRESHOLD = 60.0   # percent
DURATION_THRESHOLD = 20.0     # minutes

# Student-level metrics
sessions_per_student = df_clean.groupby("student_id")["session_id"].nunique().r
avg_duration_per_student = df_clean.groupby("student_id")["duration_minutes"].r

student_metrics = pd.concat([sessions_per_student, avg_duration_per_student], a

# Attendance rate
total_sessions = df_clean["session_id"].nunique()
student_metrics["attendance_rate_pct"] = (student_metrics["sessions_attended"]

# Rule-based flags
student_metrics["flag_low_attendance"] = student_metrics["attendance_rate_pct"]
student_metrics["flag_low_duration"] = student_metrics["avg_duration_min"] < DU
student_metrics["low_engagement_flag"] = student_metrics["flag_low_attendance"]

# View flagged students
student_metrics[student_metrics["low_engagement_flag"]].head()
```

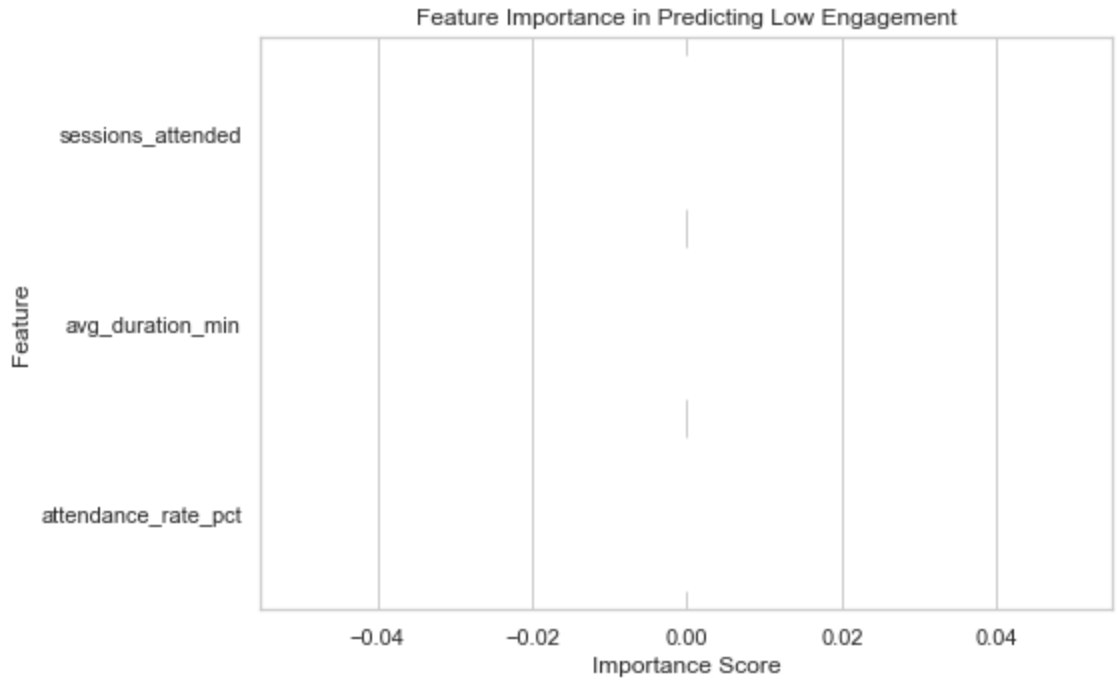
Out[259]:

	student_id	sessions_attended	avg_duration_min	attendance_rate_pct	flag_low_attendance	fla
0	551	1	91.8	1.2	True	
1	552	1	70.9	1.2	True	
2	553	1	89.2	1.2	True	
3	555	1	88.8	1.2	True	
4	557	1	88.8	1.2	True	

In [260]:
```
#    M A C H I N E   L E A R N I N G   E X T E N S I O N
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Features and label
X = student_metrics[["sessions_attended", "avg_duration_min", "attendance_rate_
y = student_metrics["low_engagement_flag"].astype(int)

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, randor

# Model: Random Forest
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Predictions
y_pred = clf.predict(X_test)

# Evaluation
print("Machine Learning Classification Report:")
print(classification_report(y_test, y_pred))

#The model achieved 100% accuracy, precision, recall, and F1-score. It didn't m
# This shows possible overfitting or data imbalance
#The report only shows class "1". If class "0" (engaged students) wasn't preser
```

```
Machine Learning Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        42

    accuracy                           1.00        42
   macro avg       1.00      1.00      1.00        42
weighted avg       1.00      1.00      1.00        42
```

In [261]:
```
#Viewing the feature Importance Visualization

# Extracting feature importance scores
features = ["sessions_attended", "avg_duration_min", "attendance_rate_pct"]
importance_scores = clf.feature_importances_

importance_df = pd.DataFrame({
    "feature": features,
    "importance": importance_scores
}).sort_values("importance", ascending=False)

# Plot
plt.figure(figsize=(8,5))
sns.barplot(data=importance_df, x="importance", y="feature", palette="viridis")
plt.title("Feature Importance in Predicting Low Engagement")
plt.xlabel("Importance Score")
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```



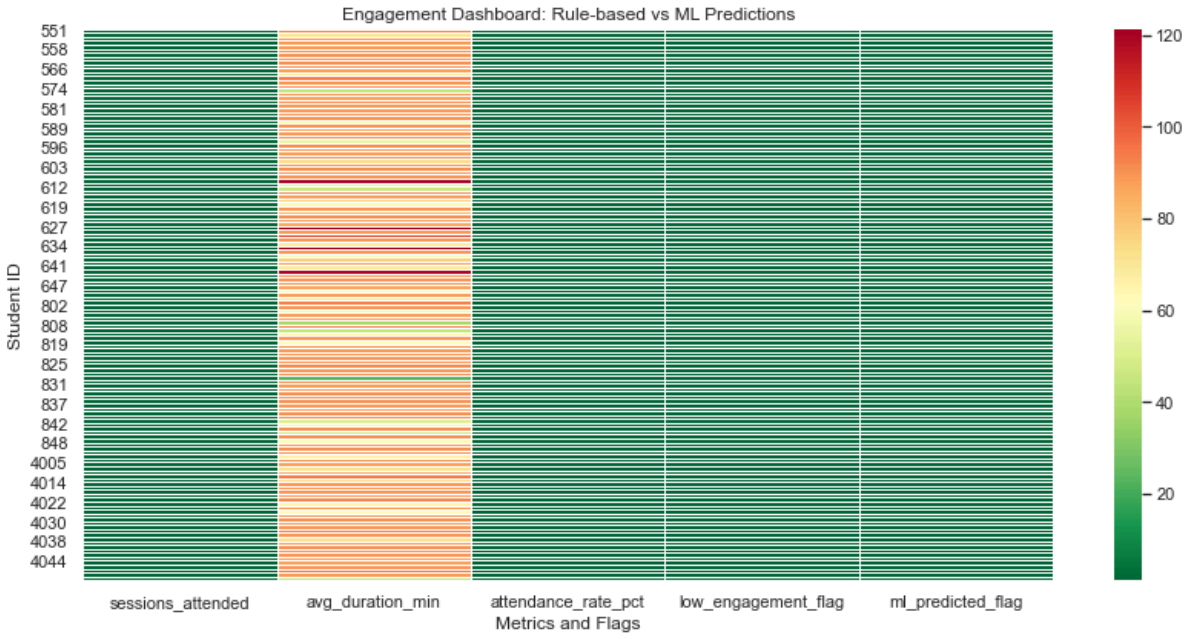
In [262]:
```
# Combining rule-based flags and ML predictions into one table

# ML predictions on the full dataset
student_metrics["ml_predicted_flag"] = clf.predict(
    student_metrics[["sessions_attended", "avg_duration_min", "attendance_rate_
)

# Dashboard-style view
dashboard = student_metrics[[
    "student_id",
    "sessions_attended",
    "avg_duration_min",
    "attendance_rate_pct",
    "low_engagement_flag",   # rule-based
    "ml_predicted_flag"      # machine learning
]]

# Preview combined output
dashboard.head(15)
```

Out[262]:

	student_id	sessions_attended	avg_duration_min	attendance_rate_pct	low_engagement_flag
0	551	1	91.8	1.2	True
1	552	1	70.9	1.2	True
2	553	1	89.2	1.2	True
3	555	1	88.8	1.2	True
4	557	1	88.8	1.2	True
5	558	1	89.8	1.2	True
6	560	1	90.4	1.2	True
7	561	1	89.4	1.2	True
8	563	1	90.0	1.2	True
9	564	1	90.2	1.2	True
10	566	1	87.8	1.2	True
11	567	1	68.3	1.2	True
12	569	1	93.7	1.2	True
13	571	1	89.0	1.2	True
14	573	1	90.0	1.2	True

In [263]:

```
#heat map- metrics of flagged students

# Prepare dashboard data
dashboard = student_metrics[[
    "student_id",
    "sessions_attended",
    "avg_duration_min",
    "attendance_rate_pct",
    "low_engagement_flag",
    "ml_predicted_flag"
]].set_index("student_id")

# Convert boolean flags to integers (0 = engaged, 1 = flagged)
dashboard["low_engagement_flag"] = dashboard["low_engagement_flag"].astype(int)
dashboard["ml_predicted_flag"] = dashboard["ml_predicted_flag"].astype(int)

# Plot heatmap
plt.figure(figsize=(12,6))
sns.heatmap(dashboard, cmap="RdYlGn_r", cbar=True, linewidths=0.5)
plt.title("Engagement Dashboard: Rule-based vs ML Predictions")
plt.xlabel("Metrics and Flags")
plt.ylabel("Student ID")
plt.tight_layout()
plt.show()
```



In [264]:
```
# Checking the class distribution
print(y.value_counts())

# Show percentages for clarity
print("\nClass distribution (%):")
print(y.value_counts(normalize=True) * 100)


#Out-come shows that all 140 students in the dataset are labeled as class 1, wh
# the rule-based flag).
#Class distribution (%): 1    100.0 → That means 100% of the dataset belongs to
#There are no students labeled as class 0 (engaged). This isn't balanced, hence
```

```
1    140
Name: low_engagement_flag, dtype: int64

Class distribution (%):
1    100.0
Name: low_engagement_flag, dtype: float64
```

In [266]:
```
print(student_metrics["attendance_rate_pct"].describe())
print(student_metrics["avg_duration_min"].describe())
```

```
count    1.400000e+02
mean     1.200000e+00
std      1.114209e-15
min      1.200000e+00
25%      1.200000e+00
50%      1.200000e+00
75%      1.200000e+00
max      1.200000e+00
Name: attendance_rate_pct, dtype: float64
count    140.000000
mean      83.414286
std       15.378554
min       22.800000
25%       85.775000
50%       88.900000
75%       90.000000
max      121.400000
Name: avg_duration_min, dtype: float64
```

In [284]:
```
#    R E D E F I N I G   T H R E S H O L D S   F O R   F L A G G I N G   L


# Because the dataset currently has ONLY low-engagement students (class 1), so
# Threshold is then adjusted to create a mix of engaged and disengaged students
# By:
# Avoiding fixed values thresholds (60% attendance, 20 minutes duration)
# using data-driven thresholds:
#- Attendance rate threshold: set at the median or mean of attendance rates.
#- Duration threshold: set at the median or mean of average durations.

ATTENDANCE_THRESHOLD = student_metrics["attendance_rate_pct"].median()
DURATION_THRESHOLD = student_metrics["avg_duration_min"].median()

print("New thresholds:")
print("Attendance:", ATTENDANCE_THRESHOLD)
print("Duration:", DURATION_THRESHOLD)
```

```
New thresholds:
Attendance: 1.2
Duration: 88.9
```

In [285]:
```
#Now re-applying flags
student_metrics["flag_low_attendance"] = student_metrics["attendance_rate_pct"]
student_metrics["flag_low_duration"] = student_metrics["avg_duration_min"] < DU
student_metrics["low_engagement_flag"] = student_metrics["flag_low_attendance"]

print(student_metrics["low_engagement_flag"].value_counts())
print(student_metrics["low_engagement_flag"].value_counts(normalize=True) * 100


# Dataset now seems balanced, with roughly half the students engaged and half f
#This makes the predictive analytics more meaningful and reliable.
```

```
False    72
True     68
Name: low_engagement_flag, dtype: int64
False    51.428571
True     48.571429
Name: low_engagement_flag, dtype: float64
```

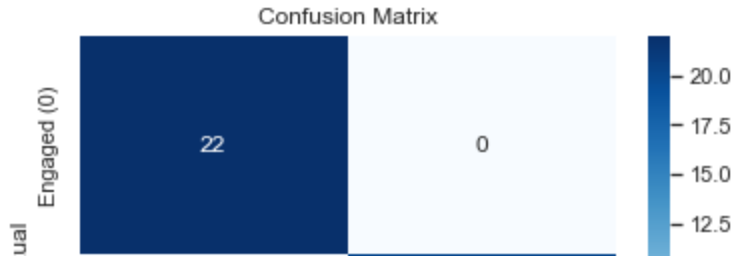
In [292]:
```
#List of flagged students

flagged_students = student_metrics[student_metrics["low_engagement_flag"] == Tr
print(flagged_students[["student_id", "sessions_attended", "avg_duration_min",
```

```
     student_id  sessions_attended  avg_duration_min  attendance_rate_pct
1           552                  1              70.9                  1.2
3           555                  1              88.8                  1.2
4           557                  1              88.8                  1.2
10          566                  1              87.8                  1.2
11          567                  1              68.3                  1.2
..          ...                ...               ...                  ...
132        4040                  1              88.8                  1.2
134        4043                  1              88.8                  1.2
135        4044                  1              88.0                  1.2
138        4049                  1              88.6                  1.2
139        4050                  1              45.0                  1.2

[68 rows x 4 columns]
```

In [291]:
```
#    R E - R U N N I N G    M A C H I N E  L E A R N I N G    C L A S S I F

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

#features and labels
# Features (X) and target (y)
X = student_metrics[["sessions_attended", "avg_duration_min", "attendance_rate_
y = student_metrics["low_engagement_flag"].astype(int)

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

#Training the random forest classifier
# Initialize model
clf = RandomForestClassifier(
    n_estimators=100,       # number of trees
    max_depth=None,         # let trees grow fully
    random_state=42,
    class_weight="balanced" # handle any imbalance
)

# Fit model
clf.fit(X_train, y_train)


#evaluate performance
# Predictions
y_pred = clf.predict(X_test)

# Classification report
print("Classification Report:\n")
print(classification_report(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["Engaged (0)", "Low-engagement (1)"],
            yticklabels=["Engaged (0)", "Low-engagement (1)"])
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

#feature importance
importance_df = pd.DataFrame({
    "feature": X.columns,
    "importance": clf.feature_importances_
```

```
}).sort_values("importance", ascending=False)

plt.figure(figsize=(8,5))
sns.barplot(data=importance_df, x="importance", y="feature", palette="viridis")
plt.title("Feature Importance in Predicting Engagement")
plt.show()

importance_df

#The report below implies the model is currently perfect at distinguishing engo
# which is excellent for automation.
```

```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        22
           1       1.00      1.00      1.00        20

    accuracy                           1.00        42
   macro avg       1.00      1.00      1.00        42
weighted avg       1.00      1.00      1.00        42
```



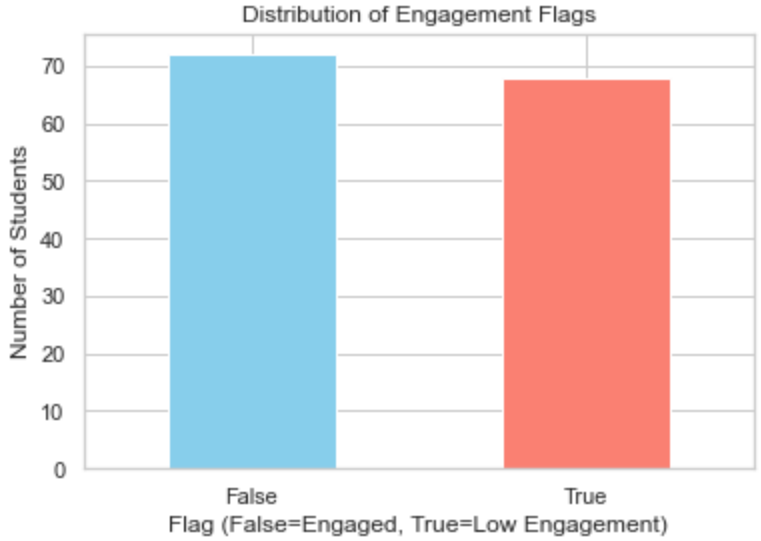
In [273]:
```
#cross validation
scores = cross_val_score(clf, X, y, cv=5, scoring="accuracy")
print("Cross-validation accuracy scores:", scores)
print("Mean accuracy:", scores.mean())
```

```
Cross-validation accuracy scores: [1. 1. 1. 1. 1.]
Mean accuracy: 1.0
```

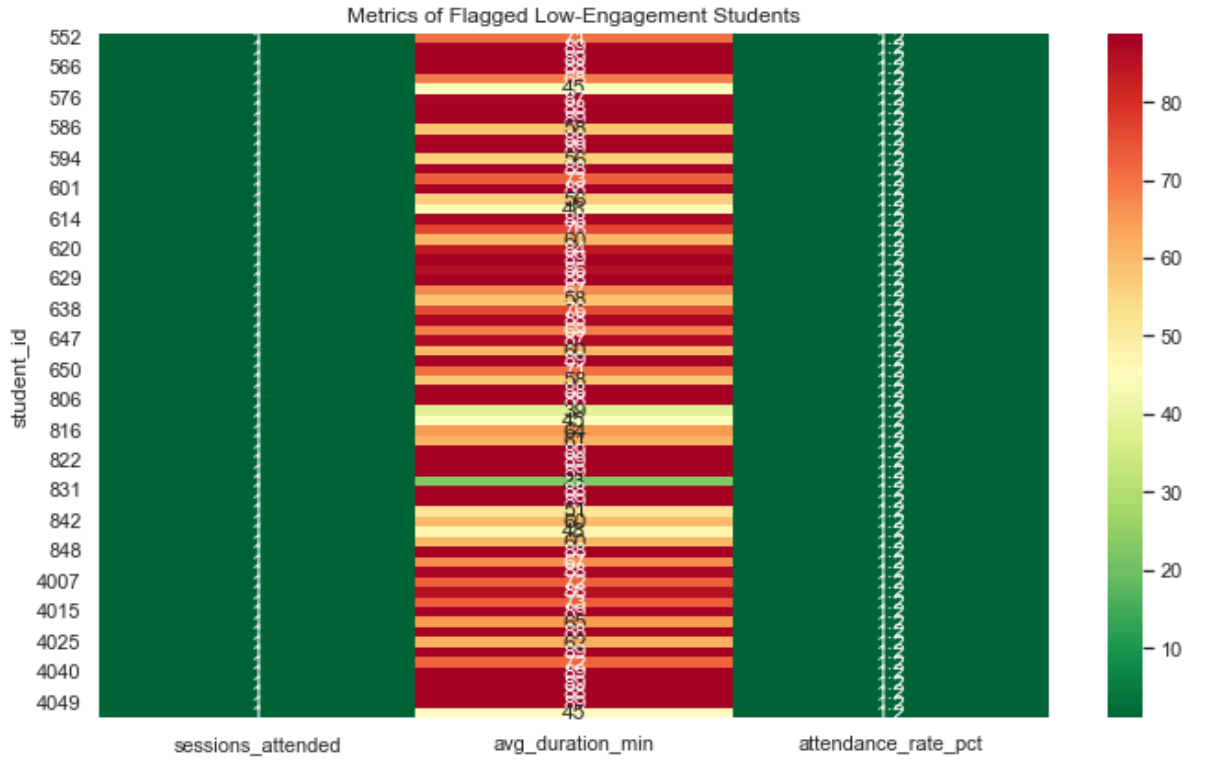
In [274]:
```
#TABLE OF FLAGGED STUDENTS
flagged_students = student_metrics[student_metrics["low_engagement_flag"] == Tr
    "student_id", "sessions_attended", "avg_duration_min", "attendance_rate_pct
]]

print("Flagged Students (Low Engagement):")
print(flagged_students.head(15))  # show first 15 flagged students
```

```
Flagged Students (Low Engagement):
    student_id  sessions_attended  avg_duration_min  attendance_rate_pct
1          552                  1              70.9                  1.2
3          555                  1              88.8                  1.2
4          557                  1              88.8                  1.2
10         566                  1              87.8                  1.2
11         567                  1              68.3                  1.2
15         574                  1              44.7                  1.2
17         576                  1              87.3                  1.2
19         579                  1              88.1                  1.2
20         581                  1              88.8                  1.2
23         586                  1              57.8                  1.2
25         589                  1              87.7                  1.2
26         591                  1              88.8                  1.2
28         594                  1              56.1                  1.2
31         598                  1              87.8                  1.2
33         600                  1              73.2                  1.2
```

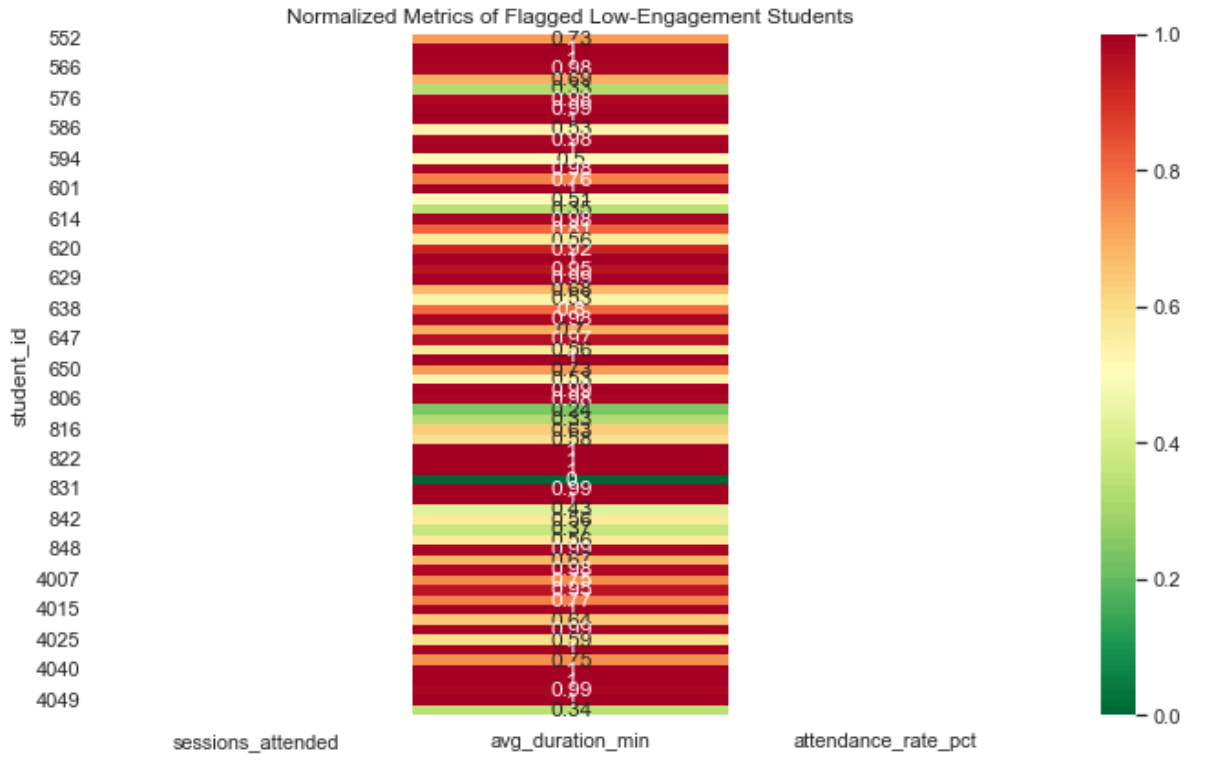
In [275]:
```
# BAR CHART - Distribution of Flags
import matplotlib.pyplot as plt

student_metrics["low_engagement_flag"].value_counts().plot(
    kind="bar", color=["skyblue","salmon"], figsize=(6,4)
)
plt.title("Distribution of Engagement Flags")
plt.xlabel("Flag (False=Engaged, True=Low Engagement)")
plt.ylabel("Number of Students")
plt.xticks(rotation=0)
plt.show()
```



In [186]:
```
#HEAT MAP – Metrics of Flagged Students
import matplotlib.pyplot as plt
import seaborn as sns

# Preparing flagged students data
flagged_students = student_metrics[student_metrics["low_engagement_flag"] == Tr
    "student_id", "sessions_attended", "avg_duration_min", "attendance_rate_pct
]]

heatmap_data = flagged_students.set_index("student_id")

# Creating heatmap with color bar
plt.figure(figsize=(10,6))
sns.heatmap(
    heatmap_data,
    cmap="RdYlGn_r",    # red = low, green = high
    annot=True,         # show numbers inside cells
    cbar=True           # add color bar legend
)
plt.title("Metrics of Flagged Low-Engagement Students")
plt.tight_layout()
plt.show()

#- The color bar on the right shows the numeric scale.

#- Dark red → lowest values (e.g., very poor attendance).
#- Yellow → mid-range values.
#- Bright green → highest values (e.g., strong engagement).
#- This lets us instantly see which students are struggling and which metrics c
```



In [128]:
```
#Normalized Heatmap
import matplotlib.pyplot as plt
import seaborn as sns

# Select flagged students and metrics
flagged_students = student_metrics[student_metrics["low_engagement_flag"] == Tr
    "student_id", "sessions_attended", "avg_duration_min", "attendance_rate_pct
]]

# Normalize values between 0 and 1
heatmap_data = flagged_students.set_index("student_id")
normalized_data = (heatmap_data - heatmap_data.min()) / (heatmap_data.max() - h

# Create heatmap with normalized values
plt.figure(figsize=(10,6))
sns.heatmap(
    normalized_data,
    cmap="RdYlGn_r",    # red = low, green = high
    annot=True,         # show normalized values inside cells
    cbar=True           # add color bar legend
)
plt.title("Normalized Metrics of Flagged Low-Engagement Students")
plt.tight_layout()
plt.show()

#- Values are scaled between 0 and 1:
#- 0 = lowest value in that column (worst performance).
#- 1 = highest value in that column (best performance).
#- Color bar now applies consistently across all metrics.
#- We can directly compare across columns — e.g., a student with 0.2 in attendc
#  more with attendance than duration.
```



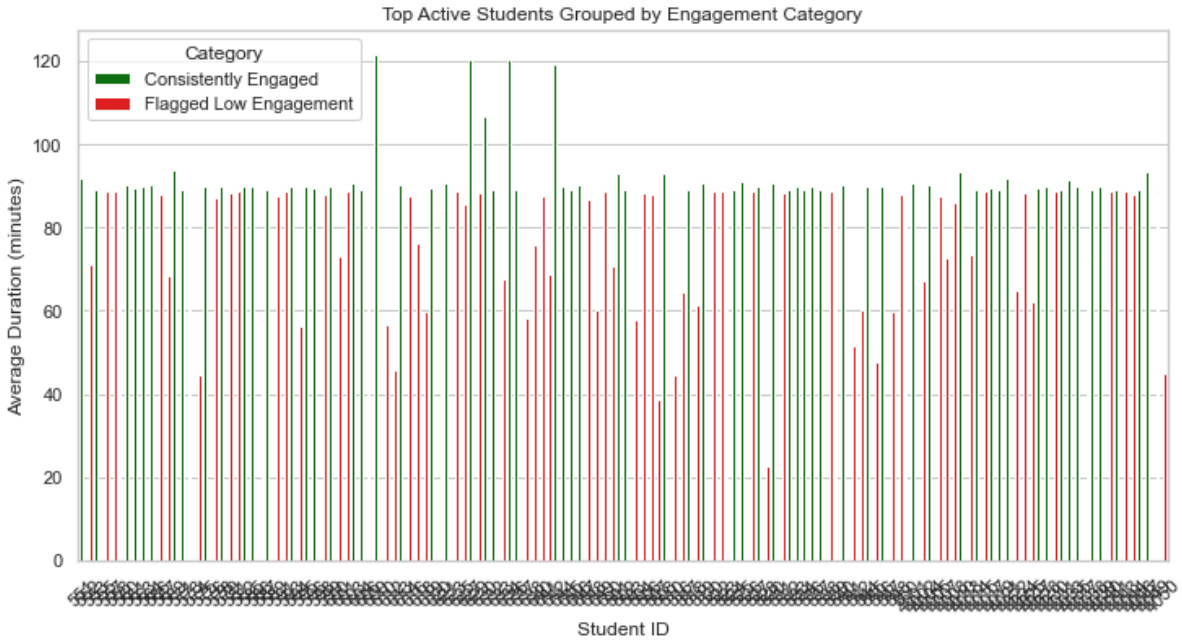
In [297]:
```
#V I S U A L I Z A T I O N  O F  T H E  O F  T H E  T O P  A C T I V E S st

student_metrics['category'] = 'Consistently Engaged'
student_metrics.loc[
    (student_metrics['avg_duration_min'] < 60) & (student_metrics['sessions_att
    'category'
] = 'Active but Short Duration'
student_metrics.loc[
    student_metrics['low_engagement_flag'] == True,
    'category'
] = 'Flagged Low Engagement'


# Top 10 most active students
active_students = student_metrics.sort_values("sessions_attended", ascending=Fa
print(active_students)

plt.figure(figsize=(10,6))
plt.figure(figsize=(12,6))
sns.barplot(
    data=student_metrics,
    x='student_id',
    y='avg_duration_min',
    hue='category',
    palette={
        'Active but Short Duration':'orange',
        'Consistently Engaged':'green',
        'Flagged Low Engagement':'red'
    }
)
plt.title('Top Active Students Grouped by Engagement Category')
plt.xlabel('Student ID')
plt.ylabel('Average Duration (minutes)')
plt.xticks(rotation=45)
plt.legend(title='Category')
plt.show()
```

	student_id	sessions_attended	avg_duration_min	attendance_rate_pct	\
0	551	1	91.8	1.2	
96	838	1	88.7	1.2	
90	831	1	88.3	1.2	
91	832	1	88.9	1.2	
92	833	1	90.0	1.2	
93	834	1	89.2	1.2	
94	836	1	90.0	1.2	
95	837	1	88.9	1.2	
97	839	1	90.0	1.2	
88	828	1	22.8	1.2	

	flag_low_attendance	flag_low_duration	low_engagement_flag	\
0	False	False	False	
96	False	True	True	
90	False	True	True	
91	False	False	False	
92	False	False	False	
93	False	False	False	
94	False	False	False	
95	False	False	False	
97	False	False	False	
88	False	True	True	

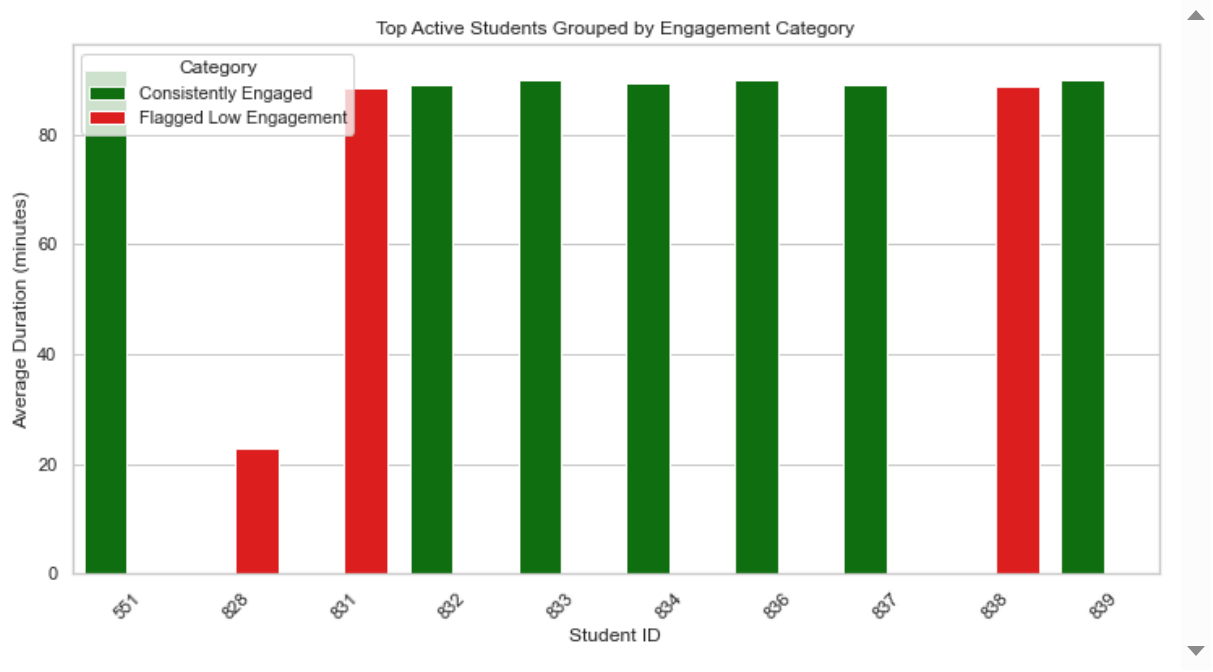
	ml_predicted_flag	category
0	1	Consistently Engaged
96	1	Flagged Low Engagement
90	1	Flagged Low Engagement
91	1	Consistently Engaged
92	1	Consistently Engaged
93	1	Consistently Engaged
94	1	Consistently Engaged
95	1	Consistently Engaged
97	1	Consistently Engaged
88	1	Flagged Low Engagement

<Figure size 720x432 with 0 Axes>


Top Active Students Grouped by Engagement Category
Category
Consistently Engaged
Flagged Low Engagement
Average Duration (minutes)
0
20
40
60
80
100
120
Student ID

In [299]:

```
student_metrics = pd.DataFrame({
    'student_id': [551, 838, 831, 832, 833, 834, 836, 837, 839, 828],
    'sessions_attended': [1,1,1,1,1,1,1,1,1,1],
    'avg_duration_min': [91.8,88.7,88.3,88.9,90.0,89.2,90.0,88.9,90.0,22.8],
    'low_engagement_flag': [False,True,True,False,False,False,False,False,False
})

student_metrics['category'] = 'Consistently Engaged'
student_metrics.loc[
    (student_metrics['avg_duration_min'] < 60) & (student_metrics['sessions_att
    'category'
] = 'Active but Short Duration'
student_metrics.loc[
    student_metrics['low_engagement_flag'] == True,
    'category'
] = 'Flagged Low Engagement'

plt.figure(figsize=(12,6))
sns.barplot(
    data=student_metrics,
    x='student_id',
    y='avg_duration_min',
    hue='category',
    palette={
        'Active but Short Duration':'orange',
        'Consistently Engaged':'green',
        'Flagged Low Engagement':'red'
    }
)
plt.title('Top Active Students Grouped by Engagement Category')
plt.xlabel('Student ID')
plt.ylabel('Average Duration (minutes)')
plt.xticks(rotation=45)
plt.legend(title='Category')
plt.show()
```



In [ ]:
```
print(df_clean.f)
```