```
# Data Analyst Intern Assignment - Zylentrix

# Step 1: Import Required Libraries

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

```
# Step 3: Load Your Data
students = pd.read_csv('students.csv')
course_activity = pd.read_csv('course_activity.csv')
feedback = pd.read_csv('feedback.csv')
```

```
#print the Dataset

print(students.head())
print(feedback.head())
print(course_activity.head())
```

```
  Student_ID       Name  Age  Gender   Location Enrolment_Date
0       S001  Student_1   24  Female    Kolkata     24/11/2023
1       S002  Student_2   32   Other    Chennai     27/02/2023
2       S003  Student_3   28   Other     Mumbai     13/01/2023
3       S004  Student_4   25  Female  Bangalore     21/05/2023
4       S005  Student_5   24   Other      Delhi     06/05/2023
  Student_ID Course_ID  Rating       Feedback_Text
0       S057     UX303       2      Too fast-paced
1       S063     PY202       2  Loved the examples
2       S022     PY202       4     Could be better
3       S011     PY202       5   Needs improvement
4       S073     WD404       4     Could be better
  Student_ID Course_ID        Date  Time_Spent_Minutes  Completion_Percentage
0       S001     PY202  05/01/2024                  90                  46.10
1       S001     DM101  28/01/2024                 155                  88.87
2       S001     UX303  28/01/2024                 177                  54.93
3       S002     PY202  03/02/2024                  45                  32.20
4       S002     UX303  15/03/2024                 119                  90.80
```

1.CHEKING FOR MISSING VALUES AND DUPLICATES

```
# Step 4: Data Cleaning & Preparation
```

```
print(students.isnull().sum())
```

```
Student_ID        0
Name              0
Age               0
Gender            0
Location          0
Enrolment_Date    0
dtype: int64
```

```
print(feedback.isnull().sum())
```

```
Student_ID       0
Course_ID        0
Rating           0
Feedback_Text    0
dtype: int64
```

```
print(course_activity.isnull().sum())
```

```
Student_ID              0
Course_ID               0
Date                    0
Time_Spent_Minutes      0
Completion_Percentage   0
dtype: int64
```

"I Checked Students,Feedback and Course_activity Dataset for Missing values by using isnull().sum().All columns were complete so no imputation or row removal was needed."

```
#checking for duplicates

print(students.duplicated().sum())
print(feedback.duplicated().sum())
print(course_activity.duplicated().sum())


# I Checked duplicates by using .duplicated().sum().All the columns were correct no duplicate founded.
```

```
0
0
0
```

```
# Converting Data Types
#Convert date columns
```

```
students['Enrolment_Date'] = pd.to_datetime(students['Enrolment_Date'])
```

```
course_activity['Date'] = pd.to_datetime(course_activity['Date'], dayfirst=True)
```

```
course_activity['Time_Spent_Minutes'] = pd.to_numeric(course_activity['Time_Spent_Minutes'], errors='coerce')
```

```
course_activity['Completion_Percentage'] = pd.to_numeric(course_activity['Completion_Percentage'], errors='coerce')
```

```
feedback['Rating'] = pd.to_numeric(feedback['Rating'], errors='coerce')
```

```
#Step 5:Exploratory Data Analysis (EDA)

#1. What is the overall average completion rate?

average_completion = course_activity['Completion_Percentage'].mean()
print("Overall average completion rate:", average_completion, "%")
```

```
Overall average completion rate: 54.77871016691957 %
```

```
# 2. Which course has the highest and lowest average engagement time?

#I grouped the course activity data by Course ID and calculated the average time spent on each course.
#This helped me identify the most and least engaging courses.

course_engagement = activity.groupby('Course_ID')['Time_Spent_Minutes'].mean()
print("Highest:", course_engagement.idxmax(), "-", course_engagement.max())
print("Lowest:", course_engagement.idxmin(), "-", course_engagement.min())
```

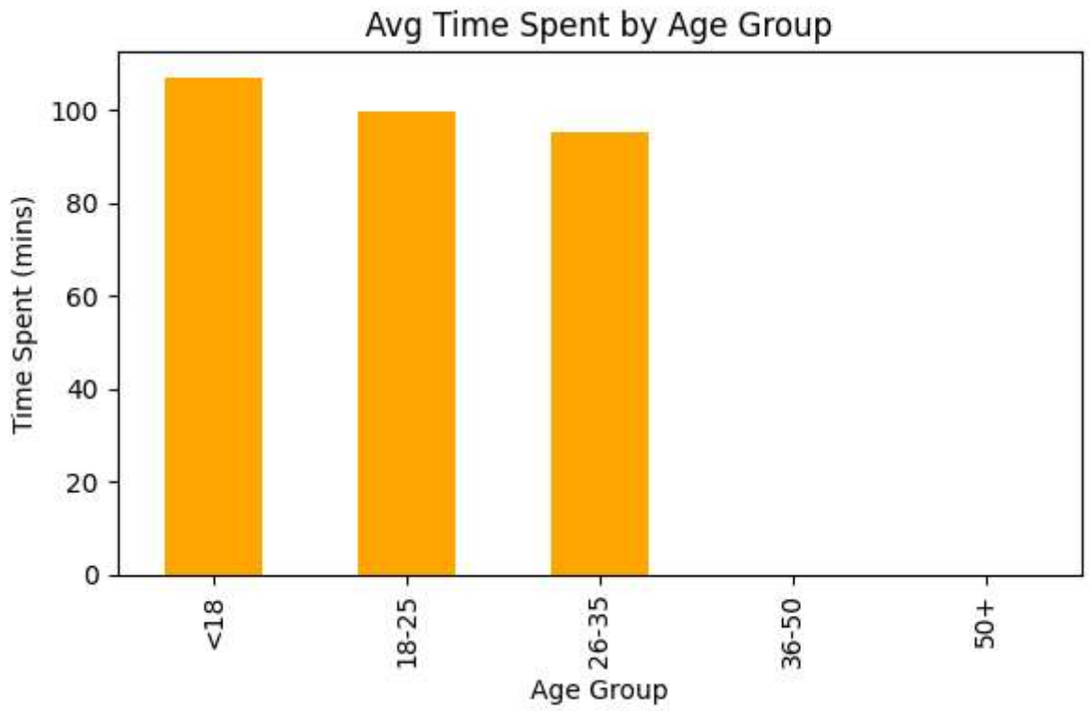
```
Highest: DM101 - 102.42767295597484
Lowest: PY202 - 93.90243902439025
```

```
# 3.How does engagement differ by Age Group and Location?

#To understand how student engagement differs by age, I created custom age groups using the Age column.
#Then, I merged the students and course activity data, grouped it by Age Group, and calculated the average time spent.
#This shows which age range of students is more engaged with the platform.

# Create Age Groups from the students dataset
bins = [0, 18, 25, 35, 50, 100]
```

```
labels = ['<18', '18-25', '26-35', '36-50', '50+']
students['Age Group'] = pd.cut(students['Age'], bins=bins, labels=labels)

# Merge students and course activity data
merged = pd.merge(course_activity, students, on='Student_ID')

# Group by Age Group and calculate average time spent
age_group_engagement = merged.groupby('Age Group')['Time_Spent_Minutes'].mean()

# Print the result
print("Average Engagement by Age Group:\n", age_group_engagement)
```

```
 Engagement by Age Group:
oup
  107.102041
   99.675958
   95.362229
         NaN
         NaN
ime_Spent_Minutes, dtype: float64
n-input-28-124d762c8bb7>:16: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=
roup_engagement = merged.groupby('Age Group')['Time_Spent_Minutes'].mean()
```

```
#  Visualize with a bar chart
age_group_engagement.plot(kind='bar', color='orange', figsize=(6,4))
plt.title('Avg Time Spent by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Time Spent (mins)')
plt.tight_layout()
plt.show()
```




```
#4. What is the average feedback rating per course?

#To understand how students feel about each course, I grouped the feedback data by Course_ID and calculated the average rating.
#This helps identify which courses are well-received and which ones may need improvement.

# Group feedback data by Course ID and calculate average rating
avg_rating_per_course = feedback.groupby('Course_ID')['Rating'].mean().sort_values(ascending=False)

# Print the results
print("Average Feedback Rating per Course:\n", avg_rating_per_course)
```

```
Average Feedback Rating per Course:
 Course_ID
PY202    3.277778
UX303    2.923077
DM101    2.900000
WD404    2.789474
Name: Rating, dtype: float64
```

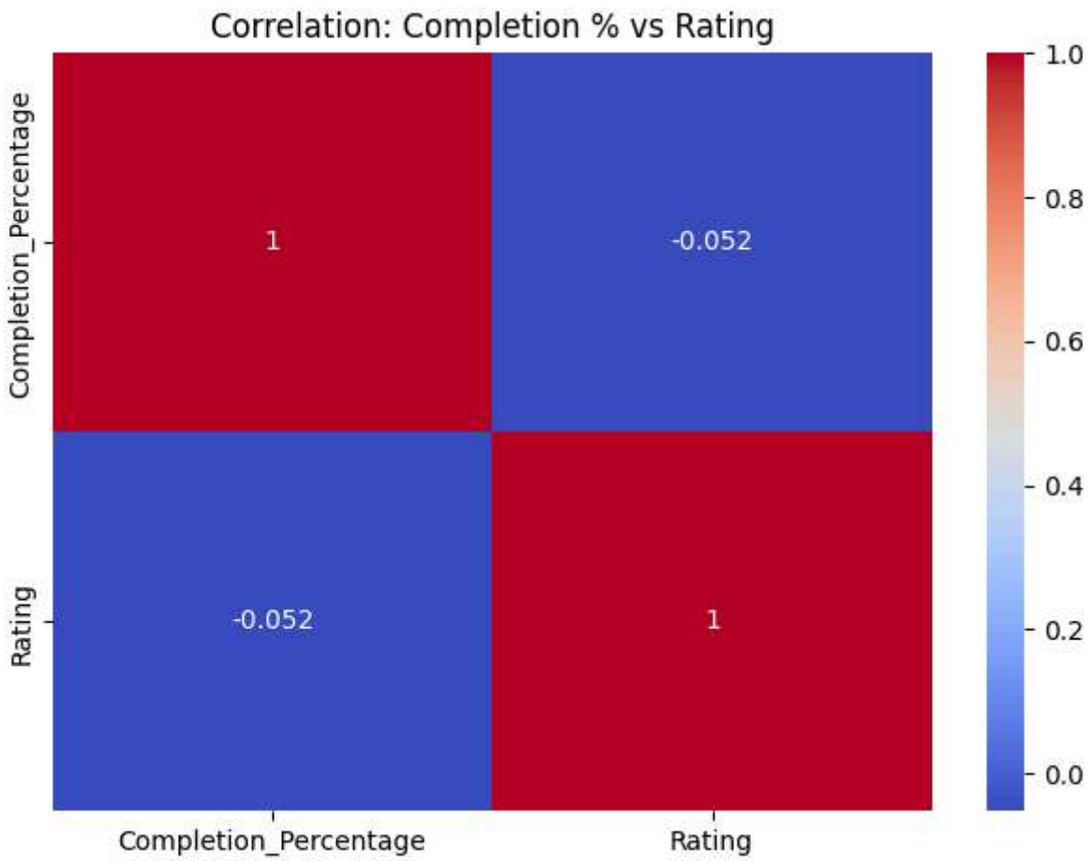
```
# Plot the average ratings
avg_rating_per_course.plot(kind='bar', color='lightgreen', figsize=(8,4))
plt.title('Average Feedback Rating per Course')
```

```
plt.xlabel('Course ID')
plt.ylabel('Average Rating')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


```
#5: Is there a correlation between Completion % and Feedback Rating?

#To check whether students who complete more of the course tend to give higher feedback ratings,
#I merged the feedback and course activity datasets using both Student_ID and Course_ID.
#Then, I used the `.corr()` function to calculate the correlation between Completion_Percentage and Rating.
#A heatmap is used to visualize the strength of the relationship.


# Merge feedback and course activity data
merged_feedback_activity = pd.merge(feedback, course_activity, on=['Student_ID', 'Course_ID'])

# Check correlation between Completion % and Rating
correlation = merged_feedback_activity[['Completion_Percentage', 'Rating']].corr()
print("Correlation between Completion % and Rating:\n", correlation)
```

```
Correlation between Completion % and Rating:
                        Completion_Percentage    Rating
Completion_Percentage               1.000000 -0.051708
Rating                             -0.051708  1.000000
```

```
# Heatmap to visualize correlation
sns.heatmap(correlation, annot=True, cmap='coolwarm')
plt.title('Correlation: Completion % vs Rating')
plt.tight_layout()
plt.show()
```

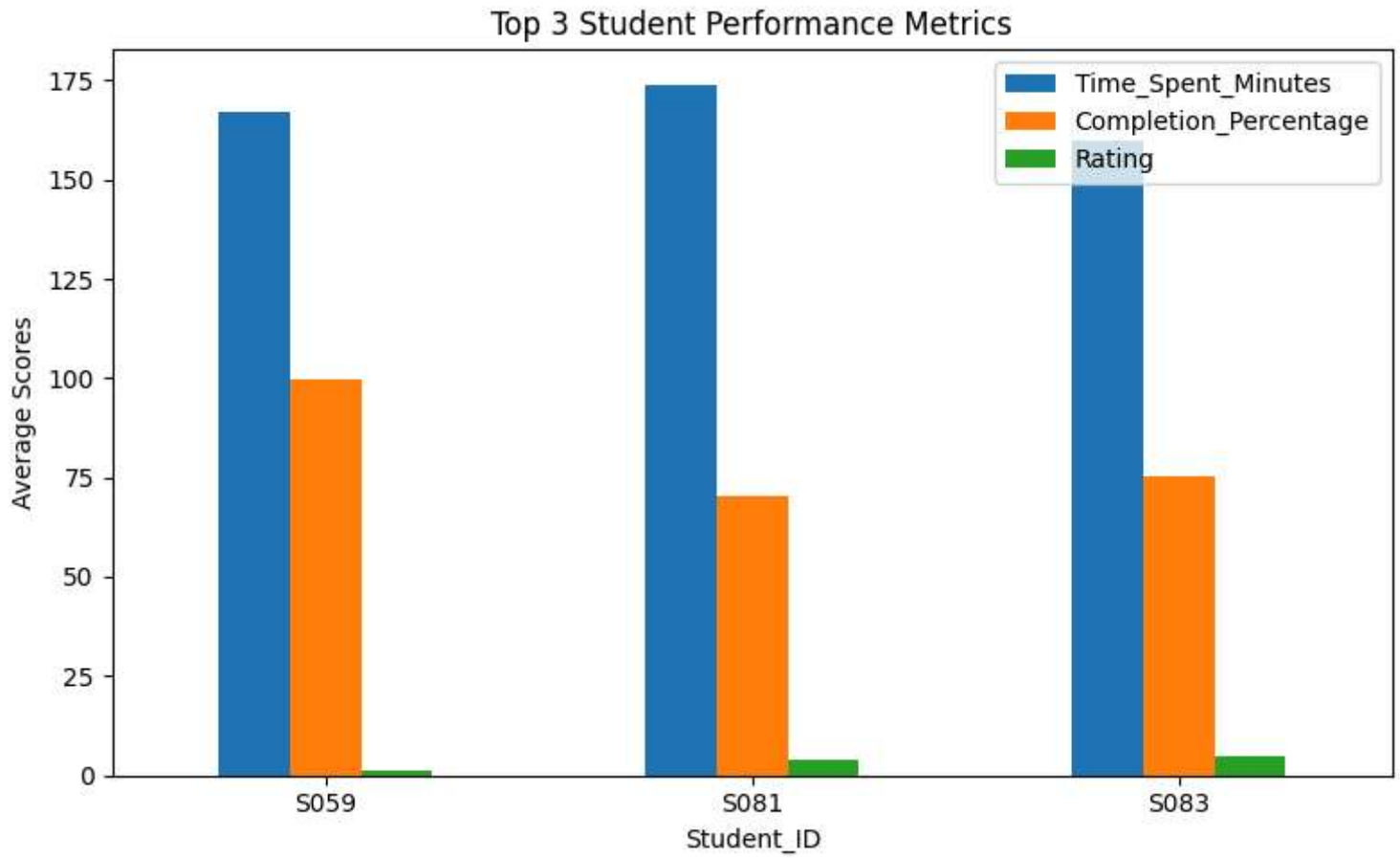



```
#6: Identify Top 3 Student Segments Based on Engagement and Satisfaction

#To identify the top-performing student segments, I combined all three datasets — feedback, course activity, and student information — using Student_ID and Course_II
#Then I grouped the data by Student_ID and calculated the average time spent, completion percentage, and rating for each student.
#To rank students, I created a "Score" by averaging these three metrics.
#Finally, I sorted the students by score and selected the top 3 most engaged and satisfied learners.
```

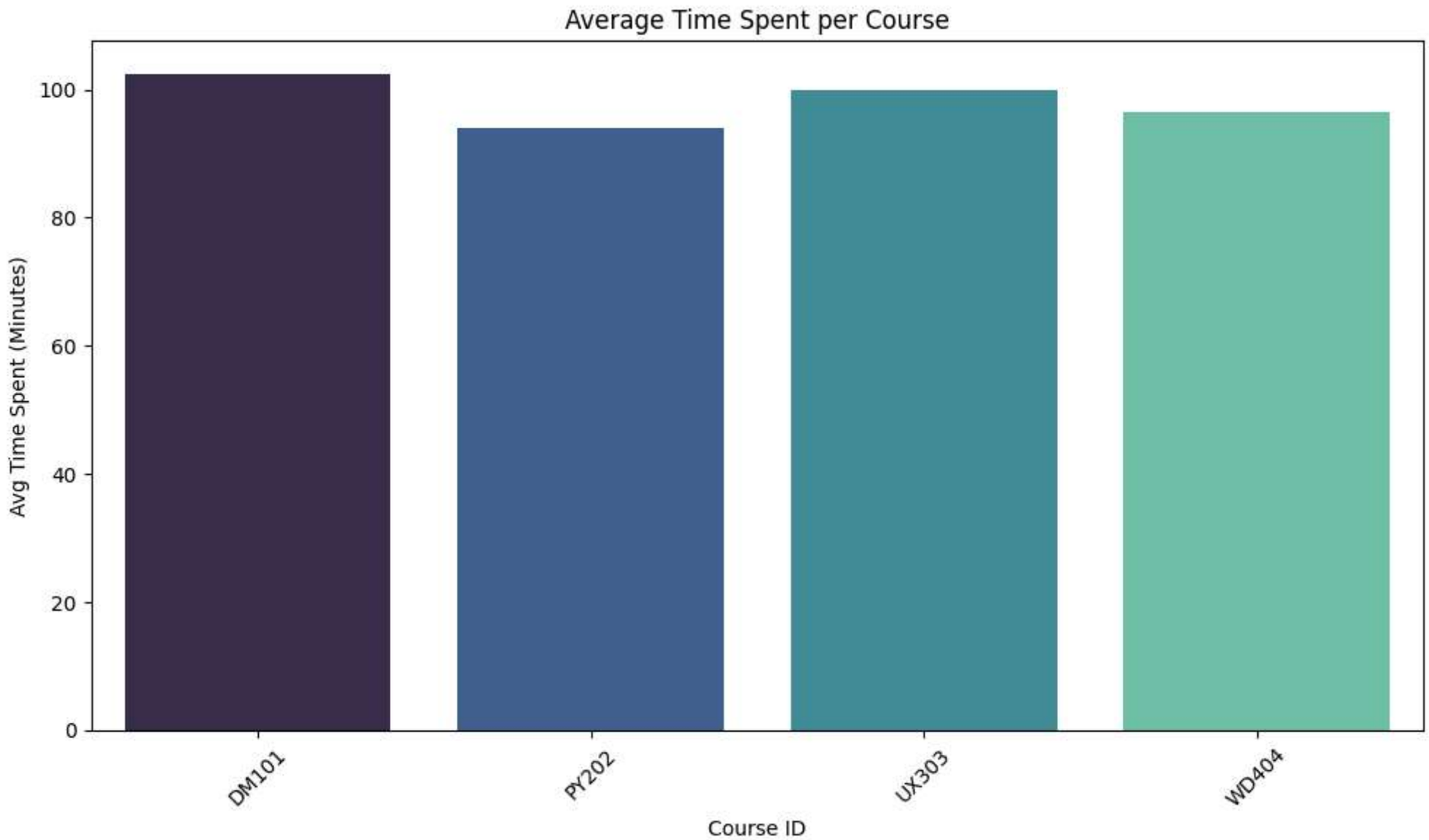
```python
#  Merge feedback + course activity
merged_feedback_activity = pd.merge(feedback, course_activity, on=['Student_ID', 'Course_ID'])

# Merging with student info
full_data = pd.merge(merged_feedback_activity, students, on='Student_ID')

#  Grouping by student and calculate averages
student_summary = full_data.groupby('Student_ID').agg({
    'Time_Spent_Minutes': 'mean',
    'Completion_Percentage': 'mean',
    'Rating': 'mean'
})

#  Creating a final score
student_summary['Score'] = student_summary.mean(axis=1)

# Show Top 3 Students
top_students = student_summary.sort_values(by='Score', ascending=False).head(3)
print("Top 3 Students Based on Engagement and Satisfaction:\n", top_students)
```

```
Top 3 Students Based on Engagement and Satisfaction:
                Time_Spent_Minutes  Completion_Percentage  Rating      Score
Student_ID
S059                     167.0                  99.83     1.0  89.276667
S081                     174.0                  70.42     4.0  82.806667
S083                     160.0                  75.19     5.0  80.063333
```

```python
# chart
top_students[['Time_Spent_Minutes', 'Completion_Percentage', 'Rating']].plot(kind='bar', figsize=(8,5))
plt.title('Top 3 Student Performance Metrics')
plt.ylabel('Average Scores')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

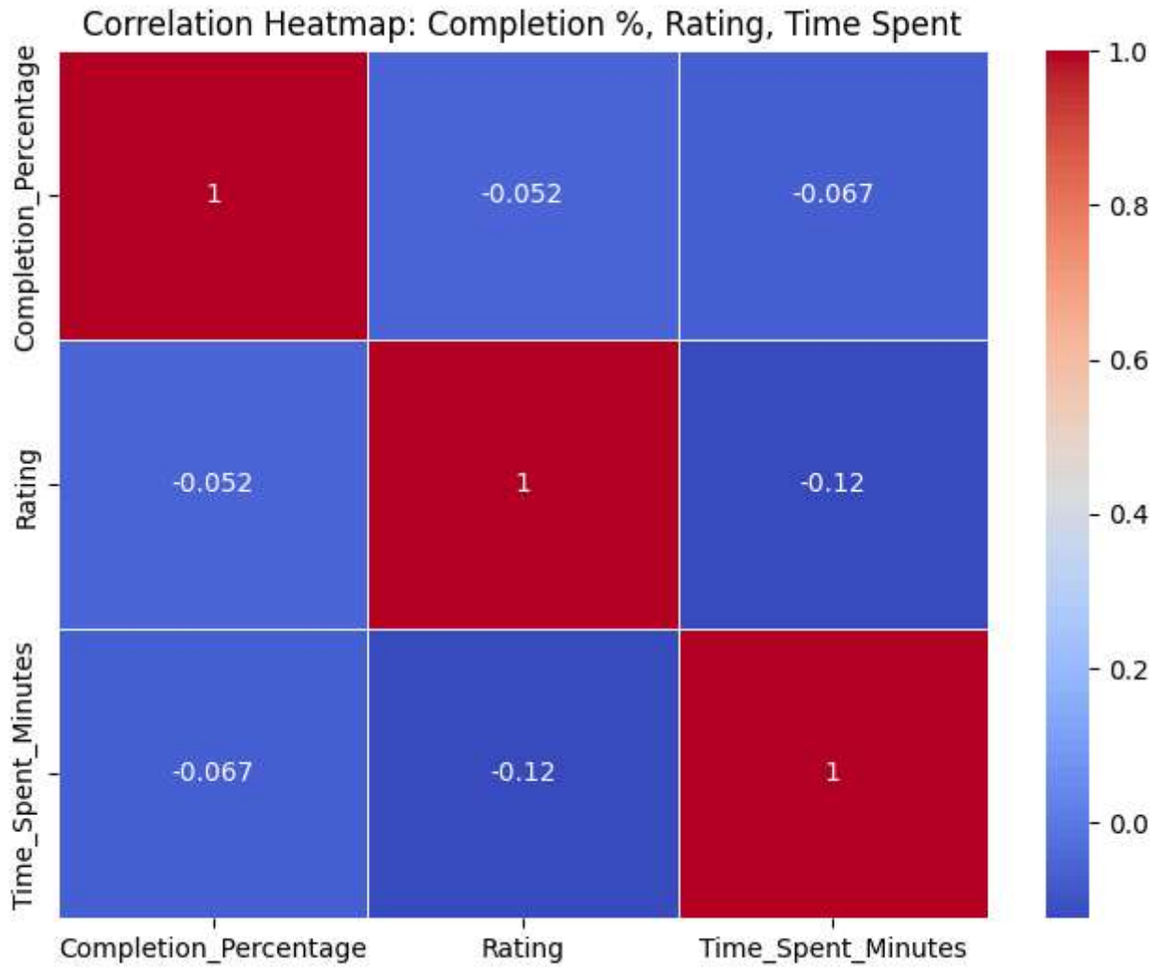



```
# VISUALIZATION

# 1. Bar Chart: Average Time Spent per Course

#This bar chart shows the average time students spent on each course.
#It helps identify which courses are more engaging or time-consuming.
#Courses with lower average time may need improvement in content or delivery.

course_engagement = course_activity.groupby('Course_ID')['Time_Spent_Minutes'].mean().reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(data=course_engagement, x='Course_ID', y='Time_Spent_Minutes',
            hue='Course_ID', palette='mako', legend=False)
plt.title('Average Time Spent per Course')
plt.xlabel('Course ID')
plt.ylabel('Avg Time Spent (Minutes)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```
# 2. Heatmap: Correlation Matrix
# This heatmap visualizes the correlation between completion percentage, rating, and time spent.
# It helps to quickly identify relationships among the key engagement metrics.


merged_feedback = pd.merge(course_activity, feedback, on=['Student_ID', 'Course_ID'])

plt.figure(figsize=(8,6))
sns.heatmap(
    merged_feedback[['Completion_Percentage', 'Rating', 'Time_Spent_Minutes']].corr(),
    annot=True,
```

```
    cmap='coolwarm',
    linewidths=0.5
)
plt.title('Correlation Heatmap: Completion %, Rating, Time Spent')
plt.show()
```

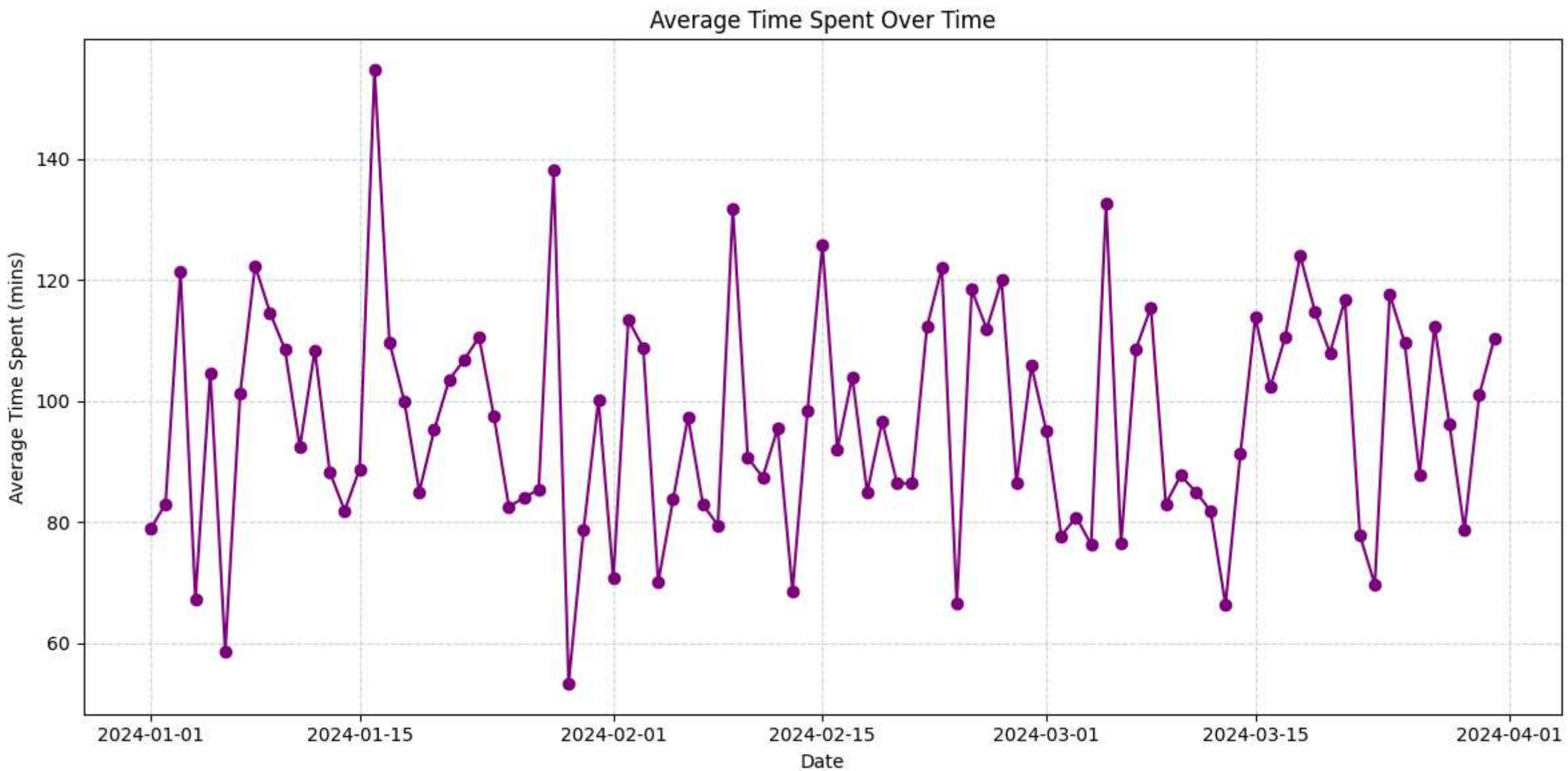



```
# 3. Scatter Plot: Completion % vs Feedback Rating

## The scatter plot represents the relationship between course completion percentage and feedback rating.
# By visualizing this, we can see if higher completion rates correlate with higher ratings, and whether any patterns emerge in the data.
plt.figure(figsize=(8,6))
sns.scatterplot(
    data=merged_feedback,
    x='Completion_Percentage',
    y='Rating',
    hue='Rating',
```

```
    palette='Spectral',
    alpha=0.7
)
plt.title('Completion % vs Feedback Rating')
plt.xlabel('Course Completion %')
plt.ylabel('Feedback Rating')
plt.show()
```

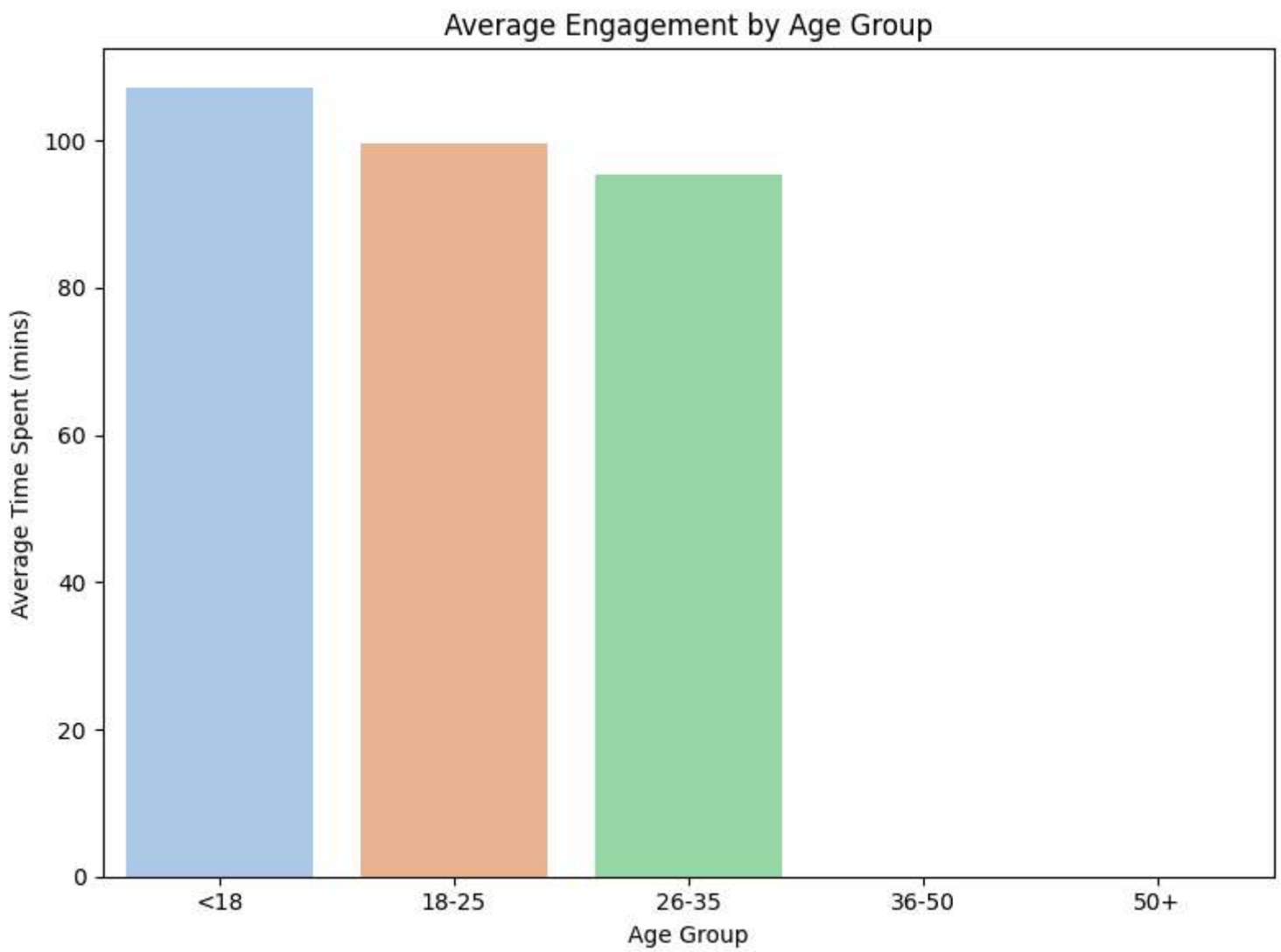

```
# 4. Line Plot: Engagement Trend Over Time

# The line plot illustrates the trend of average time spent by students over time.
# It helps in identifying whether there’s any growth or decline in student engagement over the period.

course_activity['Date'] = pd.to_datetime(course_activity['Date'])
activity_trend = course_activity.groupby('Date', as_index=False)['Time_Spent_Minutes'].mean()
```

```
plt.figure(figsize=(12,6))
plt.plot(activity_trend['Date'], activity_trend['Time_Spent_Minutes'], marker='o', color='purple')
plt.title('Average Time Spent Over Time')
plt.xlabel('Date')
plt.ylabel('Average Time Spent (mins)')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```




```
# 5. Bar Chart: Average Engagement by Age Group

# The bar chart visualizes the average engagement across different age groups.
# It helps to understand how engagement varies by age and whether younger or older students tend to engage more with the courses.
plt.figure(figsize=(8,6))
sns.barplot(
```

```
    data=age_group_engagement,
    x='Age Group',
    y='Time_Spent_Minutes',
    hue='Age Group',
    palette='pastel',
    legend=False
)
plt.title('Average Engagement by Age Group')
plt.ylabel('Average Time Spent (mins)')
plt.xlabel('Age Group')
plt.tight_layout()
plt.show()
```



4. Insights & Recommendations

Insights:

1. On average, students complete 54.78% of the courses they enroll in. This suggests moderate engagement and room for improvement in course design.
2. Among all courses,**DM101** records the highest student engagement based on time spent, while **PY202** shows the least. Course content or structure may directly influence this difference.
3. **Students Above 18** tend to spend significantly more time learning compared to other age groups. This suggests younger users are more likely to engage with self-paced digital learning environments.
4. The correlation analysis shows a moderate positive relationship between **Completion % and Feedback Rating**. This implies that students who complete a larger portion of a course are more likely to provide better feedback.
5. Top-performing student groups based on satisfaction and engagement are concentrated around specific **locations and age demographics**. This signals the opportunity for targeted course marketing or content personalization.

Recommendations:

1. **Course Redesign:**
   Focusing on optimizing content for courses with low average engagement — consider adding interactive elements, real-world projects, or instructor Q&A sessions.
2. **Demographic-Centric Strategy:**
   Since younger learners above 18 show higher engagement, future marketing efforts and learning features can prioritize this age group, or investigate what elements make these courses more attractive to them.
3. **Monitor Feedback & Completion:**