

Task

Data Analyst Intern Assignment: Analyse User Engagement on an Online Learning Platform

Background Scenario: Zylentrix runs an online training platform where students enrol in different learning tracks (e.g., Digital Marketing, Python, UI/UX, etc.). The platform records student behaviour, such as logins, course completions, time spent on lessons, and feedback. You've been given 3 CSV files:

1. students.csv – Contains student info (ID, Name, Age, Gender, Location, Enrolment Date)
2. course_activity.csv – Tracks user activity (Student ID, Course ID, Date, Time Spent [mins], Completion %)
3. feedback.csv – Records feedback (Student ID, Course ID, Rating [1- 5], Feedback Text) Tasks & Deliverables
4. Data Cleaning & Preparation Identify and handle missing, duplicate, or inconsistent data Convert data types where necessary (e.g., dates, numeric fields) Prepare a clean dataset ready for analysis
5. Exploratory Data Analysis (EDA) Answer the following using Python (preferably with pandas/matplotlib/seaborn/plotly) or any other analytics tool of your choice: What is the overall average completion rate across courses? Which course has the highest and lowest average engagement time? How does engagement differ by age group or location? What is the average feedback rating per course? Is there a correlation between completion rate and feedback rating? Identify top 3 student segments based on engagement and satisfaction.
6. Visualisations Create at least 3–5 visualisations (bar charts, heatmaps, scatter plots, etc.) to: Show engagement patterns by course and demographic Compare feedback ratings Highlight trends over time
7. Insights & Recommendations Summarise your top 5 insights from the analysis Suggest 3 data-driven recommendations to improve student engagement or course performance Format your findings in a short slide deck (PDF or Google Slides, max 5 slides) or include it in a structured report Submission Format Submit either: o A

Here is all the data you need: "course_activity.csv" "students.csv" "feedback.csv"

Data loading

Subtask:

Load the provided CSV files into pandas DataFrames.

Reasoning: Load the three CSV files into pandas DataFrames and display the first 5 rows of each to verify.

```
import pandas as pd

# Load the datasets into pandas DataFrames.
try:
    df_students = pd.read_csv('students.csv')
    df_course_activity = pd.read_csv('course_activity.csv')
    df_feedback = pd.read_csv('feedback.csv')

    # Display the first 5 rows of each DataFrame.
    display(df_students.head())
    display(df_course_activity.head())
    display(df_feedback.head())
except FileNotFoundError:
    print("One or more CSV files not found.")
except pd.errors.ParserError:
    print("Error parsing one or more CSV files.")
except Exception as e:
    print(f"An unexpected error occurred: {e}")

{"summary": "{\n  \"name\": \"    print(f\\\"\\\"\\\"An unexpected error\noccurred: {e}\\\"\\\"\\\")\\\", \n  \"rows\": 5, \n  \"fields\": [\n    {\n      \"column\": \"Student_ID\", \n      \"properties\": {\n        \"dtype\": \"string\", \n        \"num_unique_values\": 5, \n        \"samples\": [\n          \"S002\", \n          \"S005\", \n          \"S003\", \n          ], \n        \"semantic_type\": \"\", \n        \"description\": \"\", \n        \"column\":\n        \"Name\", \n        \"properties\": {\n          \"dtype\": \"string\", \n          \"num_unique_values\": 5, \n          \"samples\": [\n            \"Student_2\", \n            \"Student_5\", \n            \"Student_3\", \n            ], \n            \"semantic_type\": \"\", \n            \"description\": \"\", \n            }, \n            {\n              \"column\": \"Age\", \n              \"properties\": {\n                \"dtype\": \"number\", \n                \"std\": 3, \n                \"min\":\n                24, \n                \"max\": 32, \n                \"num_unique_values\": 4, \n                \"samples\": [\n                  32, \n                  25, \n                  24,\n                  ], \n                  \"semantic_type\": \"\", \n                  \"description\": \"\", \n                  {\n                    \"column\":\n                    \"Gender\", \n                    \"properties\": {\n                      \"dtype\":\n                      \"category\", \n                      \"num_unique_values\": 2, \n                      \"samples\":\n                      [\n                        \"Other\", \n                        \"Female\", \n                        ], \n                        \"semantic_type\": \"\", \n                        \"description\": \"\", \n                        }, \n                        {\n                          \"column\": \"Location\", \n                          \"properties\":\n                          {\n                            \"dtype\": \"string\", \n                            \"num_unique_values\": 5, \n                            \"samples\": [\n                              \"Chennai\", \n                              \"Delhi\", \n                              ], \n                              \"semantic_type\": \"\", \n                              \"description\": \"\", \n                              {\n                                \"column\":\n                                \"Enrolment_Date\", \n                                \"properties\": {\n                                  \"dtype\":\n                                  \"object\", \n                                  \"num_unique_values\": 5, \n                                  \"samples\":
```

```

[\n      \n"27/02/2023\", \n      \n"06/05/2023\" \n      ], \n
\"semantic_type\": \"\", \n      \"description\": \"\" \n      } \n
n      ] \n }\", \"type\": \"dataframe\"}

{ \"summary\": \"{\n  \"name\": \"      print(f\\\"\\\"\\\"An unexpected error
occurred: {e}\\\"\\\"\\\")\", \n  \"rows\": 5, \n  \"fields\": [\n    {\n
\"column\": \"Student_ID\", \n    \"properties\": {\n
\"dtype\": \"category\", \n    \"num_unique_values\": 2, \n
\"samples\": [\n      \"S002\", \n      \"S001\" \n    ], \n
\"semantic_type\": \"\", \n    \"description\": \"\" \n    } \n
n    }, \n    {\n      \"column\": \"Course_ID\", \n
\"properties\": {\n      \"dtype\": \"string\", \n
\"num_unique_values\": 3, \n      \"samples\": [\n
\"PY202\", \n      \"DM101\" \n    ], \n
\"semantic_type\": \"\", \n    \"description\": \"\" \n    } \n
n    }, \n    {\n      \"column\": \"Date\", \n      \"properties\": {\n
\"dtype\": \"string\", \n      \"num_unique_values\": 4, \n
\"samples\": [\n      \"28/01/2024\", \n      \"15/03/2024\" \n    ], \n
\"semantic_type\": \"\", \n    \"description\": \"\" \n    } \n
} \n    }, \n    {\n      \"column\": \"Time_Spent_Minutes\", \n
\"properties\": {\n      \"dtype\": \"number\", \n      \"std\":
52, \n      \"min\": 45, \n      \"max\": 177, \n
\"num_unique_values\": 5, \n      \"samples\": [\n      155, \n
119 \n    ], \n      \"semantic_type\": \"\", \n
\"description\": \"\" \n    } \n    }, \n    {\n      \"column\":
\"Completion_Percentage\", \n      \"properties\": {\n
\"dtype\": \"number\", \n      \"std\": 26.175321392487238, \n
\"min\": 32.2, \n      \"max\": 90.8, \n      \"num_unique_values\":
5, \n      \"samples\": [\n      88.87, \n      90.8 \n
n      ], \n      \"semantic_type\": \"\", \n
\"description\": \"\" \n    } \n    } \n  ] \n }\", \"type\": \"dataframe\"}

{ \"summary\": \"{\n  \"name\": \"      print(f\\\"\\\"\\\"An unexpected error
occurred: {e}\\\"\\\"\\\")\", \n  \"rows\": 5, \n  \"fields\": [\n    {\n
\"column\": \"Student_ID\", \n    \"properties\": {\n
\"dtype\": \"string\", \n    \"num_unique_values\": 5, \n
\"samples\": [\n      \"S063\", \n      \"S073\", \n
\"S022\" \n    ], \n      \"semantic_type\": \"\", \n
\"description\": \"\" \n    } \n    }, \n    {\n      \"column\":
\"Course_ID\", \n      \"properties\": {\n      \"dtype\":
\"string\", \n      \"num_unique_values\": 3, \n      \"samples\":
[\n      \"UX303\", \n      \"PY202\", \n      \"WD404\" \n    ], \n
\"semantic_type\": \"\", \n    \"description\": \"\" \n    } \n
} \n    }, \n    {\n      \"column\": \"Rating\", \n      \"properties\":
{\n      \"dtype\": \"number\", \n      \"std\": 1, \n
\"min\": 2, \n      \"max\": 5, \n      \"num_unique_values\": 3, \n
\"samples\": [\n      2, \n      4, \n      5 \n    ], \n
\"semantic_type\": \"\", \n    \"description\": \"\" \n    } \n
n    }, \n    {\n      \"column\": \"Feedback_Text\", \n
\"properties\": {\n      \"dtype\": \"string\", \n

```

```
\ "num_unique_values\ ": 4,\n          \ "samples\ ": [\n          \ "Loved
the examples\ ",\n          \ "Needs improvement\ ",\n          \ "Too
fast-paced\ "\n          ],\n          \ "semantic_type\ ": \ "\",\n
\ "description\ ": \ "\ "\n          }\n          }\n          ]\n }", "type": "dataframe"}
```

Data cleaning

Subtask:

Clean the three dataframes: `df_students`, `df_course_activity`, and `df_feedback`.

Reasoning: Clean the dataframes by handling missing values, removing duplicates, converting data types, and checking for inconsistencies.

```
# Handle missing values
for col in df_students.columns:
    if df_students[col].isnull().any():
        if pd.api.types.is_numeric_dtype(df_students[col]):
            df_students[col].fillna(df_students[col].median(),
inplace=True)
        else:
            df_students[col].fillna(df_students[col].mode()[0],
inplace=True)
for col in df_course_activity.columns:
    if df_course_activity[col].isnull().any():
        if pd.api.types.is_numeric_dtype(df_course_activity[col]):
            df_course_activity[col].fillna(df_course_activity[col].median(),
inplace=True)
        else:
            df_course_activity[col].fillna(df_course_activity[col].mode()[0],
inplace=True)
for col in df_feedback.columns:
    if df_feedback[col].isnull().any():
        if pd.api.types.is_numeric_dtype(df_feedback[col]):
            df_feedback[col].fillna(df_feedback[col].median(),
inplace=True)
        else:
            df_feedback[col].fillna(df_feedback[col].mode()[0],
inplace=True)

# Remove duplicates
df_students.drop_duplicates(inplace=True)
df_course_activity.drop_duplicates(inplace=True)
df_feedback.drop_duplicates(inplace=True)

# Data type conversion
df_students['Enrolment_Date'] =
```

```

pd.to_datetime(df_students['Enrolment_Date'], format='%d/%m/%Y',
errors='coerce')
df_course_activity['Time_Spent_Minutes'] =
pd.to_numeric(df_course_activity['Time_Spent_Minutes'],
errors='coerce')
df_course_activity['Completion_Percentage'] =
pd.to_numeric(df_course_activity['Completion_Percentage'],
errors='coerce')
df_course_activity['Completion_Percentage'] =
df_course_activity['Completion_Percentage'].clip(0, 100)
df_feedback['Rating'] = pd.to_numeric(df_feedback['Rating'],
errors='coerce')
df_feedback['Rating'] = df_feedback['Rating'].clip(1, 5)

# Consistency checks (example: Gender)
df_students['Gender'] = df_students['Gender'].str.lower()
df_students['Gender'] = df_students['Gender'].replace({'male': 'Male',
'female': 'Female', 'other': 'Other'})

display(df_students.head())
display(df_course_activity.head())
display(df_feedback.head())

{"summary":{"name": "display(df_feedback", "rows": 5, "
"fields": [{"column": "Student_ID", "
"properties": {"dtype": "string", "
"num_unique_values": 5, "samples": [{"
"S002", "S005", "S003"}], "
"semantic_type": "", "description": ""}
}, {"column": "Name", "properties": {"
"dtype": "string", "num_unique_values": 5, "
"samples": [{"Student_2", "Student_5", "
Student_3"}], "semantic_type": "", "
"description": ""} }, {"column":
"Age", "properties": {"dtype": "number", "
"std": 3, "min": 24, "max": 32, "
num_unique_values": 4, "samples": [{"
25, 24}, {"semantic_type": "", "
"description": ""} }, {"column":
"Gender", "properties": {"dtype":
"category", "num_unique_values": 2, "samples":
[{"Other", "Female"}], "
"semantic_type": "", "description": ""}
}, {"column": "Location", "properties":
{"dtype": "string", "num_unique_values": 5, "
"samples": [{"Chennai", "Delhi"}
}, {"semantic_type": "", "
"description": ""} }, {"column":
"Enrolment_Date", "properties": {"dtype":
"date", "min": "2023-01-13 00:00:00", "max":

```

```

\ "2023-11-24 00:00:00\ ",\n          \ "num_unique_values\ ": 5,\n
\ "samples\ ": [\n          \ "2023-02-27 00:00:00\ ",\n          \ "2023-05-06 00:00:00\ "\n          ],\n          \ "semantic_type\ ": \ "\",\n
\ "description\ ": \ "\\ "\n          }\n          }\n          ]\n          }", "type": "dataframe"}

{ "summary": "{\n  \ "name\ ": \ "display(df_feedback\ ",\n  \ "rows\ ": 5,\n  \ "fields\ ": [\n    {\n      \ "column\ ": \ "Student_ID\ ",\n      \ "properties\ ": {\n        \ "dtype\ ": \ "category\ ",\n        \ "num_unique_values\ ": 2,\n        \ "samples\ ": [\n          \ "S002\ ",\n          \ "S001\ "\n        ],\n        \ "semantic_type\ ": \ "\",\n        \ "description\ ": \ "\\ "\n      }\n    },\n    {\n      \ "column\ ": \ "Course_ID\ ",\n      \ "properties\ ": {\n        \ "dtype\ ": \ "string\ ",\n        \ "num_unique_values\ ": 3,\n        \ "samples\ ": [\n          \ "PY202\ ",\n          \ "DM101\ "\n        ],\n        \ "semantic_type\ ": \ "\",\n        \ "description\ ": \ "\\ "\n      }\n    },\n    {\n      \ "column\ ": \ "Date\ ",\n      \ "properties\ ": {\n        \ "dtype\ ": \ "string\ ",\n        \ "num_unique_values\ ": 4,\n        \ "samples\ ": [\n          \ "28/01/2024\ ",\n          \ "15/03/2024\ "\n        ],\n        \ "semantic_type\ ": \ "\",\n        \ "description\ ": \ "\\ "\n      }\n    },\n    {\n      \ "column\ ": \ "Time_Spent_Minutes\ ",\n      \ "properties\ ": {\n        \ "dtype\ ": \ "number\ ",\n        \ "std\ ": 52,\n        \ "min\ ": 45,\n        \ "max\ ": 177,\n        \ "num_unique_values\ ": 5,\n        \ "samples\ ": [\n          119,\n          155,\n          119\n        ],\n        \ "semantic_type\ ": \ "\",\n        \ "description\ ": \ "\\ "\n      }\n    },\n    {\n      \ "column\ ": \ "Completion_Percentage\ ",\n      \ "properties\ ": {\n        \ "dtype\ ": \ "number\ ",\n        \ "std\ ": 26.175321392487238,\n        \ "min\ ": 32.2,\n        \ "max\ ": 90.8,\n        \ "num_unique_values\ ": 5,\n        \ "samples\ ": [\n          88.87,\n          90.8\n        ],\n        \ "semantic_type\ ": \ "\",\n        \ "description\ ": \ "\\ "\n      }\n    }\n  ],\n  \ "type": "dataframe"}

{ "summary": "{\n  \ "name\ ": \ "display(df_feedback\ ",\n  \ "rows\ ": 5,\n  \ "fields\ ": [\n    {\n      \ "column\ ": \ "Student_ID\ ",\n      \ "properties\ ": {\n        \ "dtype\ ": \ "string\ ",\n        \ "num_unique_values\ ": 5,\n        \ "samples\ ": [\n          \ "S063\ ",\n          \ "S073\ ",\n          \ "S022\ "\n        ],\n        \ "semantic_type\ ": \ "\",\n        \ "description\ ": \ "\\ "\n      }\n    },\n    {\n      \ "column\ ": \ "Course_ID\ ",\n      \ "properties\ ": {\n        \ "dtype\ ": \ "string\ ",\n        \ "num_unique_values\ ": 3,\n        \ "samples\ ": [\n          \ "UX303\ ",\n          \ "PY202\ ",\n          \ "WD404\ "\n        ],\n        \ "semantic_type\ ": \ "\",\n        \ "description\ ": \ "\\ "\n      }\n    },\n    {\n      \ "column\ ": \ "Rating\ ",\n      \ "properties\ ": {\n        \ "dtype\ ": \ "number\ ",\n        \ "std\ ": 1,\n        \ "min\ ": 2,\n        \ "max\ ": 5,\n        \ "num_unique_values\ ": 3,\n        \ "samples\ ": [\n          2,\n          4,\n          5\n        ],\n        \ "semantic_type\ ": \ "\",\n        \ "description\ ": \ "\\ "\n      }\n    },\n    {\n      \ "column\ ": \ "Feedback_Text\ ",\n      \ "properties\ ": {\n        \ "dtype\ ": \ "string\ ",\n

```

```
\ "num_unique_values\ ": 4,\n          \ "samples\ ": [\n          \ "Loved
the examples\ ",\n          \ "Needs improvement\ ",\n          \ "Too
fast-paced\ "\n          ],\n          \ "semantic_type\ ": \ "\",\n
\ "description\ ": \ "\ "\n          }\n          }\n          ]\n }", "type": "dataframe"}
```

Data preparation

Subtask:

Merge the cleaned dataframes: `df_students`, `df_course_activity`, and `df_feedback` into a single dataframe called `df_merged`. Create new features as needed.

Reasoning: Merge the dataframes, create new features, and display the result.

```
# Merge df_students and df_course_activity
df_temp = pd.merge(df_students, df_course_activity, on='Student_ID',
how='inner')

# Merge df_temp and df_feedback
df_merged = pd.merge(df_temp, df_feedback, on=['Student_ID',
'Course_ID'], how='inner')

# Create Age_Group feature
bins = [18, 25, 35, 45, float('inf')]
labels = ['18-24', '25-34', '35-44', '45+']
df_merged['Age_Group'] = pd.cut(df_merged['Age'], bins=bins,
labels=labels, right=False)

# Create Engagement_Level feature
df_merged['Engagement_Level'] = 'Medium' # Default value
df_merged.loc[(df_merged['Time_Spent_Minutes'] >
df_merged['Time_Spent_Minutes'].quantile(0.75)) |
(df_merged['Completion_Percentage'] >
df_merged['Completion_Percentage'].quantile(0.75)),
'Engagement_Level'] = 'High'
df_merged.loc[(df_merged['Time_Spent_Minutes'] <
df_merged['Time_Spent_Minutes'].quantile(0.25)) |
(df_merged['Completion_Percentage'] <
df_merged['Completion_Percentage'].quantile(0.25)),
'Engagement_Level'] = 'Low'

# Display the first few rows and shape of df_merged
display(df_merged.head())
print(df_merged.shape)

{"summary":{"\n  \ "name\ ": \ "print(df_merged\ ",\n  \ "rows\ ": 5,\n  \ "fields\ ": [\n    {\n      \ "column\ ": \ "Student_ID\ ",\n      \ "properties\ ": {\n        \ "dtype\ ": \ "string\ ",\n
```



```

{"column": "Feedback_Text", "description": "Feedback text", "dtype": "category", "num_unique_values": 2, "properties": {"dtype": "category", "num_unique_values": 2, "samples": ["Loved the examples", "Disliked the examples"], "semantic_type": "Text"}, "samples": ["Loved the examples", "Disliked the examples"], "semantic_type": "Text"}, {"column": "Age_Group", "description": "Age group", "dtype": "category", "num_unique_values": 2, "properties": {"dtype": "category", "num_unique_values": 2, "samples": ["18-24", "25-34"], "semantic_type": "Text"}, "samples": ["18-24", "25-34"], "semantic_type": "Text"}, {"column": "Engagement_Level", "description": "Engagement level", "dtype": "category", "num_unique_values": 3, "properties": {"dtype": "category", "num_unique_values": 3, "samples": ["Low", "Medium", "High"], "semantic_type": "Text"}, "samples": ["Low", "Medium", "High"], "semantic_type": "Text"}], "type": "dataframe"}
(122, 14)

```

Data exploration

Subtask:

Explore the merged dataset, `df_merged`, to understand its characteristics and identify potential patterns.

Reasoning: I need to explore the merged dataset `df_merged` by calculating descriptive statistics, visualizing data distributions, examining categorical variable frequencies, and performing correlation analysis, as well as checking for missing values and validating data types.

```

# Save the cleaned dataset to a new CSV file
df_merged.to_csv('cleaned_dataset.csv', index=False)

import matplotlib.pyplot as plt
import seaborn as sns

# 1. Data Shape and Summary Statistics
print("Shape of df_merged:", df_merged.shape)
print("\nDescriptive Statistics:")
display(df_merged[['Age', 'Time_Spent_Minutes', 'Completion_Percentage', 'Rating']].describe())

# 2. Data Distributions
plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
sns.histplot(df_merged['Age'], kde=True)
plt.title('Distribution of Age')

plt.subplot(2, 2, 2)
sns.histplot(df_merged['Time_Spent_Minutes'], kde=True)
plt.title('Distribution of Time Spent (Minutes)')

```

```

plt.subplot(2, 2, 3)
sns.histplot(df_merged['Completion_Percentage'], kde=True)
plt.title('Distribution of Completion Percentage')

plt.subplot(2, 2, 4)
sns.histplot(df_merged['Rating'], kde=True)
plt.title('Distribution of Rating')
plt.tight_layout()
plt.show()

# 3. Categorical Variable Frequencies
categorical_cols = ['Gender', 'Location', 'Age_Group',
'Engagement_Level', 'Course_ID']
plt.figure(figsize=(15, 10))
for i, col in enumerate(categorical_cols):
    plt.subplot(3, 2, i+1)
    df_merged[col].value_counts().plot(kind='bar')
    plt.title(f'Frequency of {col}')
plt.tight_layout()
plt.show()

# 4. Correlation Analysis
correlation_matrix = df_merged[['Age', 'Time_Spent_Minutes',
'Completion_Percentage', 'Rating']].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Matrix of Numerical Variables')
plt.show()

# 5. Missing Values Check
print("\nMissing Values in df_merged:")
print(df_merged.isnull().sum())

# 6. Data Type Validation
print("\nData Types in df_merged:")
print(df_merged.dtypes)

Shape of df_merged: (122, 14)

```

Descriptive Statistics:

```

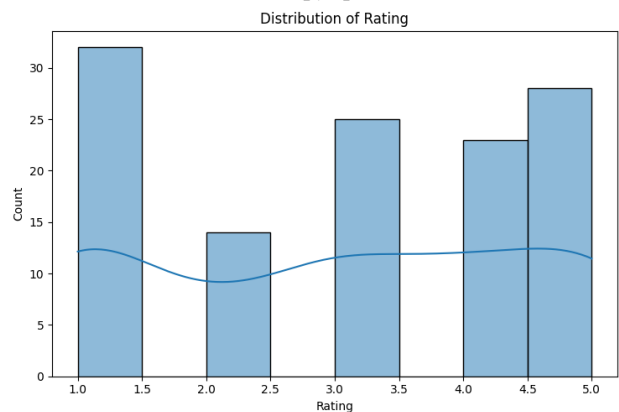
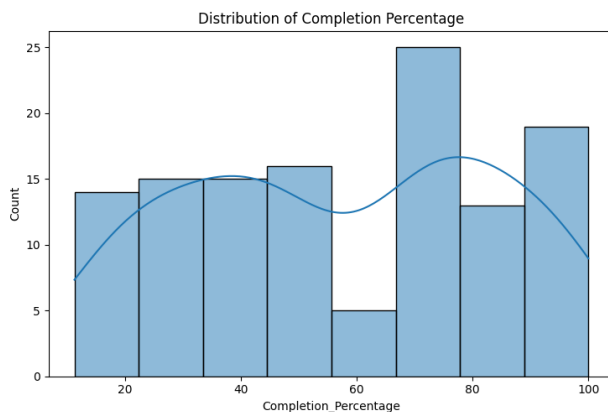
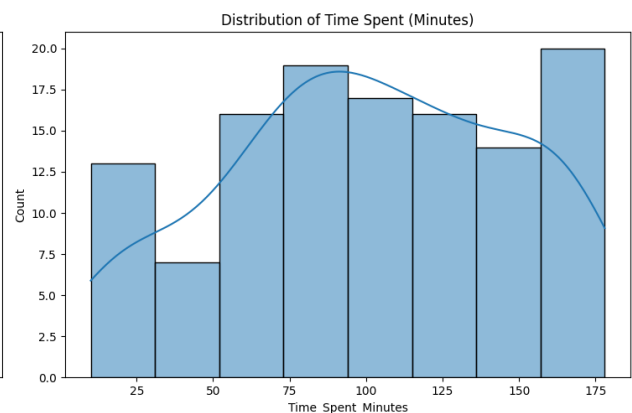
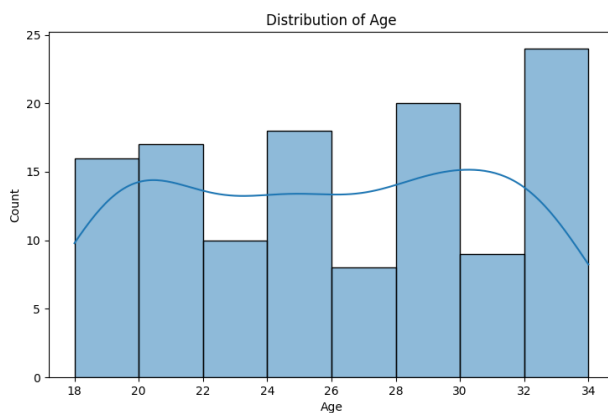
{"summary": "{\n  \"name\": \"print(df_merged\\\", \n  \"rows\": 8, \n
\"fields\": [\n    {\n      \"column\": \"Age\\\", \n
\"properties\": {\n        \"dtype\": \"number\\\", \n        \"std\":
36.16681890337821, \n        \"min\": 4.954863332205501, \n
\"max\": 122.0, \n        \"num_unique_values\": 8, \n
\"samples\": [\n          25.860655737704917, \n          25.5, \n
122.0\n        ], \n        \"semantic_type\": \"\", \n

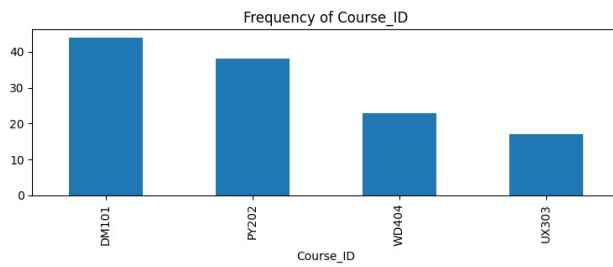
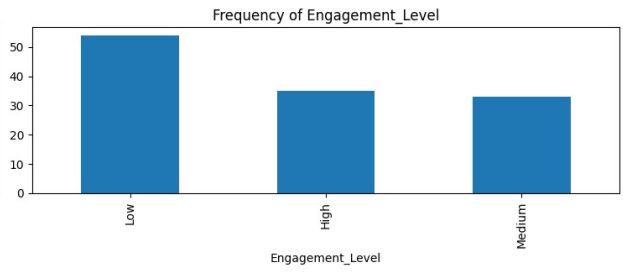
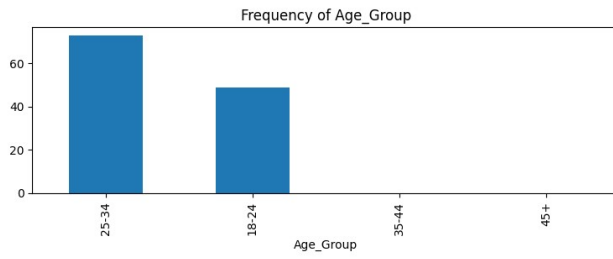
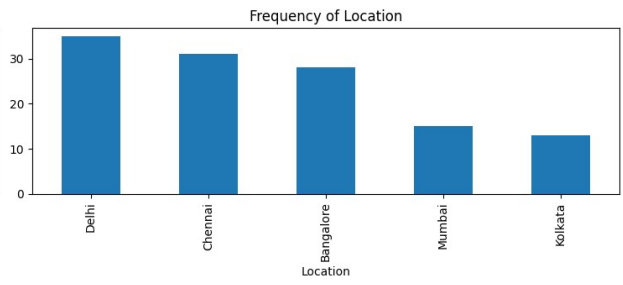
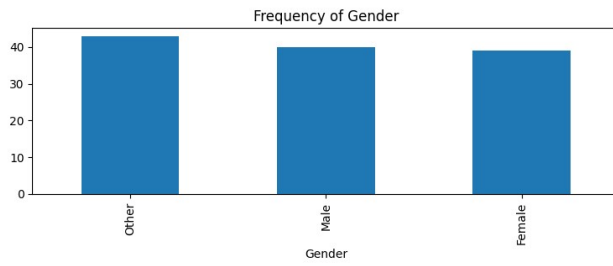
```

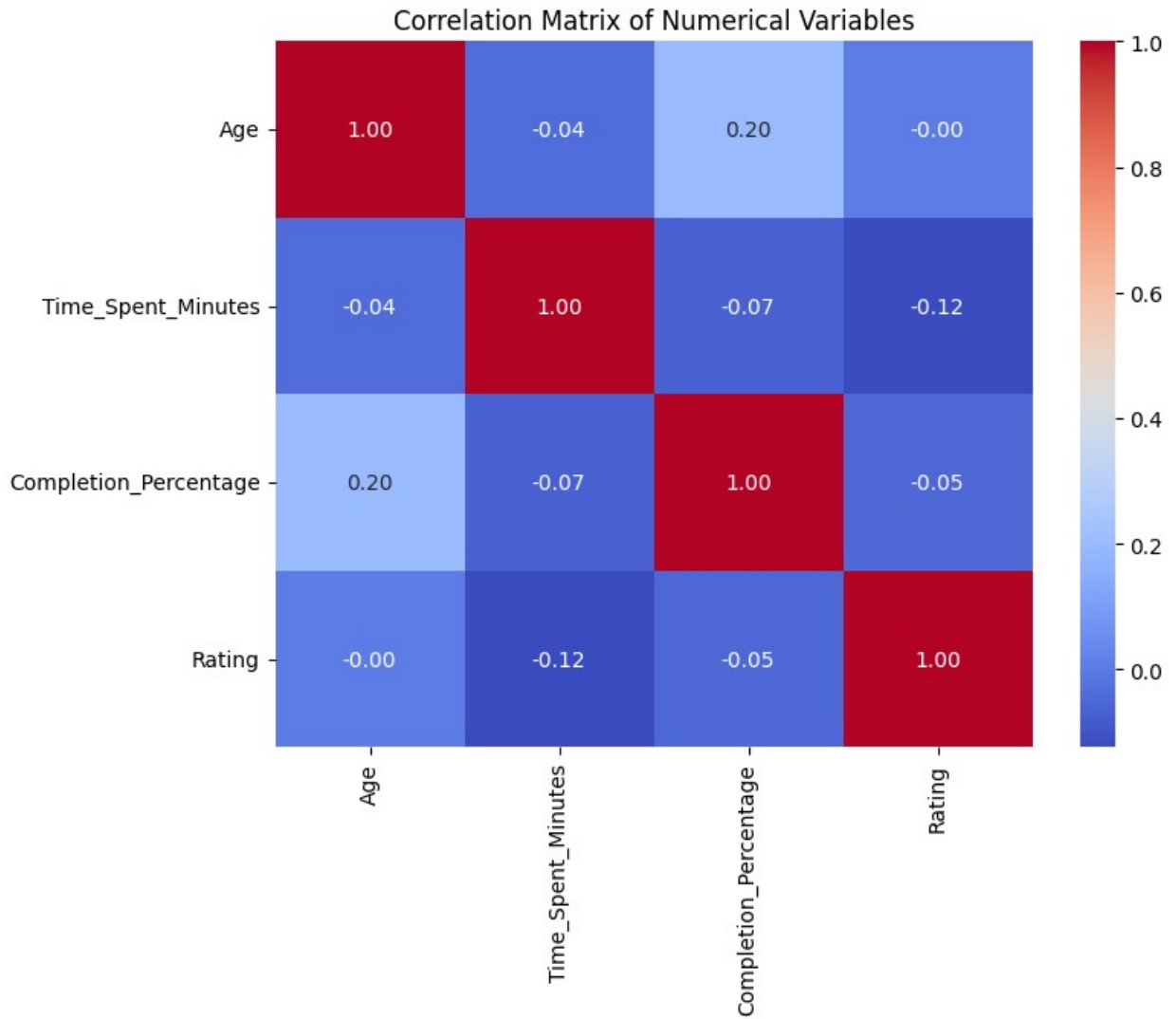
```

{"description\":"\", \"column\":"\",
\"Time_Spent_Minutes\", \"properties\": {\"dtype\":
\"number\", \"std\": 53.27596805673643, \"min\":
10.0, \"max\": 178.0, \"num_unique_values\": 8, \"
\"samples\": [100.9672131147541, 98.5,
122.0], \"semantic_type\": \"\",
\"description\": \"\", \"column\":
\"Completion_Percentage\", \"properties\": {\"
\"dtype\": \"number\", \"std\": 37.713697510689705,
\"min\": 11.25, \"max\": 122.0,
\"num_unique_values\": 8, \"samples\": [
57.3838524590164, 55.655, 122.0],
\"semantic_type\": \"\", \"description\": \"\",
\", \"column\": \"Rating\", \"properties\":
{\"dtype\": \"number\", \"std\":
42.22230684820655, \"min\": 1.0, \"max\": 122.0,
\"num_unique_values\": 7, \"samples\": [
3.0081967213114753, 4.0],
\"semantic_type\": \"\", \"description\": \"\",
\", \"column\": \"\", \"type\": \"dataframe\"}

```







Missing Values in df_merged:

Student_ID	0
Name	0
Age	0
Gender	0
Location	0
Enrolment_Date	0
Course_ID	0
Date	0
Time_Spent_Minutes	0
Completion_Percentage	0
Rating	0
Feedback_Text	0
Age_Group	0
Engagement_Level	0

dtype: int64

```
Data Types in df_merged:
Student_ID          object
Name                object
Age                 int64
Gender              object
Location            object
Enrolment_Date      datetime64[ns]
Course_ID           object
Date                object
Time_Spent_Minutes  int64
Completion_Percentage float64
Rating              int64
Feedback_Text       object
Age_Group            category
Engagement_Level    object
dtype: object
```

Data analysis

Subtask:

Perform the required data analysis on the merged dataset `df_merged`.

Reasoning: Calculate the overall average completion rate, highest/lowest average engagement time, engagement by age/location, average feedback rating per course, and the correlation between completion rate and feedback rating. Also identify the top 3 student segments based on engagement and satisfaction.

```
# 1. Overall Average Completion Rate
overall_avg_completion = df_merged['Completion_Percentage'].mean()
print(f"Overall Average Completion Rate: {overall_avg_completion:.2f}%")

# 2. Highest and Lowest Average Engagement Time
avg_engagement_by_course = df_merged.groupby('Course_ID')
['Time_Spent_Minutes'].mean()
highest_engagement_course = avg_engagement_by_course.idxmax()
lowest_engagement_course = avg_engagement_by_course.idxmin()
print(f"\nCourse with Highest Average Engagement Time:
{highest_engagement_course} ({avg_engagement_by_course.max():.2f}
minutes)")
print(f"Course with Lowest Average Engagement Time:
{lowest_engagement_course} ({avg_engagement_by_course.min():.2f}
minutes)")

# 3. Engagement by Age Group and Location
engagement_by_age = df_merged.groupby('Age_Group')
[['Time_Spent_Minutes', 'Completion_Percentage']].mean()
```

```

print("\nEngagement by Age Group:")
display(engagement_by_age)

engagement_by_location = df_merged.groupby('Location')
[['Time_Spent_Minutes', 'Completion_Percentage']].mean()
print("\nEngagement by Location:")
display(engagement_by_location)

# 4. Average Feedback Rating per Course
avg_rating_by_course = df_merged.groupby('Course_ID')['Rating'].mean()
print("\nAverage Feedback Rating per Course:")
display(avg_rating_by_course)

# 5. Correlation between Completion Rate and Feedback Rating
correlation =
df_merged['Completion_Percentage'].corr(df_merged['Rating'])
print(f"\nCorrelation between Completion Rate and Feedback Rating:
{correlation:.2f}")

# 6. Top 3 Student Segments (using Engagement_Level and Rating)
student_segments = df_merged.groupby(['Engagement_Level', 'Rating'])
['Student_ID'].count().reset_index(name='Student_Count')
top_segments = student_segments.sort_values(by='Student_Count',
ascending=False).head(3)
print("\nTop 3 Student Segments:")
display(top_segments)

```

Overall Average Completion Rate: 57.38%

Course with Highest Average Engagement Time: UX303 (115.35 minutes)
Course with Lowest Average Engagement Time: PY202 (85.95 minutes)

Engagement by Age Group:

<ipython-input-5-87c4e16d9136>:13: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.

```

engagement_by_age = df_merged.groupby('Age_Group')
[['Time_Spent_Minutes', 'Completion_Percentage']].mean()

```

```

{"summary": "{\n  \"name\": \"engagement_by_age\",\n  \"rows\": 4,\n  \"fields\": [\n    {\n      \"column\": \"Age_Group\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 4,\n        \"samples\": [\n          \"25-34\",\n          \"45+\",\n          \"18-24\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Time_Spent_Minutes\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.7330030597259465,\n        \"min\": 100.34693877551021,\n        \"max\": 101.38356164383562,\n        \"num_unique_values\": 2,\n      }\n    }\n  ]\n}"

```

```

\"samples\": [\n          101.38356164383562,\n          100.34693877551021\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\",\n        \"column\": \"Completion_Percentage\",\n        \"properties\": {\n          \"dtype\": \"number\",\n          \"std\": 5.794114409826985,\n          \"min\": 52.48081632653061,\n          \"max\": 60.67493150684931,\n          \"num_unique_values\": 2,\n          \"samples\": [\n            60.67493150684931,\n            52.48081632653061\n          ],\n          \"semantic_type\": \"\",\n          \"description\": \"\"\n        }\n      ],\n      \"type\": \"dataframe\", \"variable_name\": \"engagement_by_age\"

```

Engagement by Location:

```

{\"summary\": {\n  \"name\": \"engagement_by_location\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n      \"column\": \"Location\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 5,\n        \"samples\": [\n          \"Chennai\",\n          \"Mumbai\",\n          \"Delhi\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      },\n      \"column\": \"Time_Spent_Minutes\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 9.35464335372532,\n        \"min\": 89.35483870967742,\n        \"max\": 110.89285714285714,\n        \"num_unique_values\": 5,\n        \"samples\": [\n          89.35483870967742,\n          92.46666666666667,\n          104.97142857142858\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      },\n      \"column\": \"Completion_Percentage\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 4.493063477333171,\n        \"min\": 52.65857142857143,\n        \"max\": 65.054,\n        \"num_unique_values\": 5,\n        \"samples\": [\n          56.534838709677416,\n          65.054,\n          58.12571428571429\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    ]\n  },\n  \"type\": \"dataframe\", \"variable_name\": \"engagement_by_location\"

```

Average Feedback Rating per Course:

```

Course_ID
DM101    3.113636
PY202    3.236842
UX303    2.764706
WD404    2.608696
Name: Rating, dtype: float64

```

Correlation between Completion Rate and Feedback Rating: -0.05

Top 3 Student Segments:


```
{
  "summary": {
    "name": "top_segments",
    "rows": 3,
    "fields": [
      {
        "column": "Engagement_Level",
        "properties": {
          "dtype": "category",
          "num_unique_values": 1,
          "samples": [
            "Low"
          ],
          "semantic_type": "",
          "description": ""
        }
      },
      {
        "column": "Rating",
        "properties": {
          "dtype": "number",
          "std": 2,
          "min": 1,
          "max": 5,
          "num_unique_values": 3,
          "samples": [
            5
          ],
          "semantic_type": "",
          "description": ""
        }
      },
      {
        "column": "Student_Count",
        "properties": {
          "dtype": "number",
          "std": 3,
          "min": 11,
          "max": 17,
          "num_unique_values": 3,
          "samples": [
            17
          ],
          "semantic_type": "",
          "description": ""
        }
      }
    ]
  },
  "type": "dataframe",
  "variable_name": "top_segments"
}
```

Data visualization

Subtask:

Create visualizations to communicate the key findings from the data analysis.

Reasoning: Create visualizations based on the analysis results.

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

# 1. Engagement Patterns by Course and Demographic
plt.figure(figsize=(12, 6))
sns.barplot(x='Course_ID', y='Time_Spent_Minutes', hue='Age_Group',
            data=df_merged, errorbar=None)
plt.title('Engagement (Time Spent) by Course and Age Group')
plt.show()

plt.figure(figsize=(12, 6))
sns.barplot(x='Course_ID', y='Completion_Percentage', hue='Location',
            data=df_merged, errorbar=None)
plt.title('Engagement (Completion Percentage) by Course and Location')
plt.show()

# 2. Feedback Ratings Comparison
plt.figure(figsize=(10, 6))
sns.boxplot(x='Course_ID', y='Rating', data=df_merged)
plt.title('Feedback Rating Comparison across Courses')
plt.show()

# 3. Trends Over Time
# Convert 'Date' to datetime and handle errors
```

```

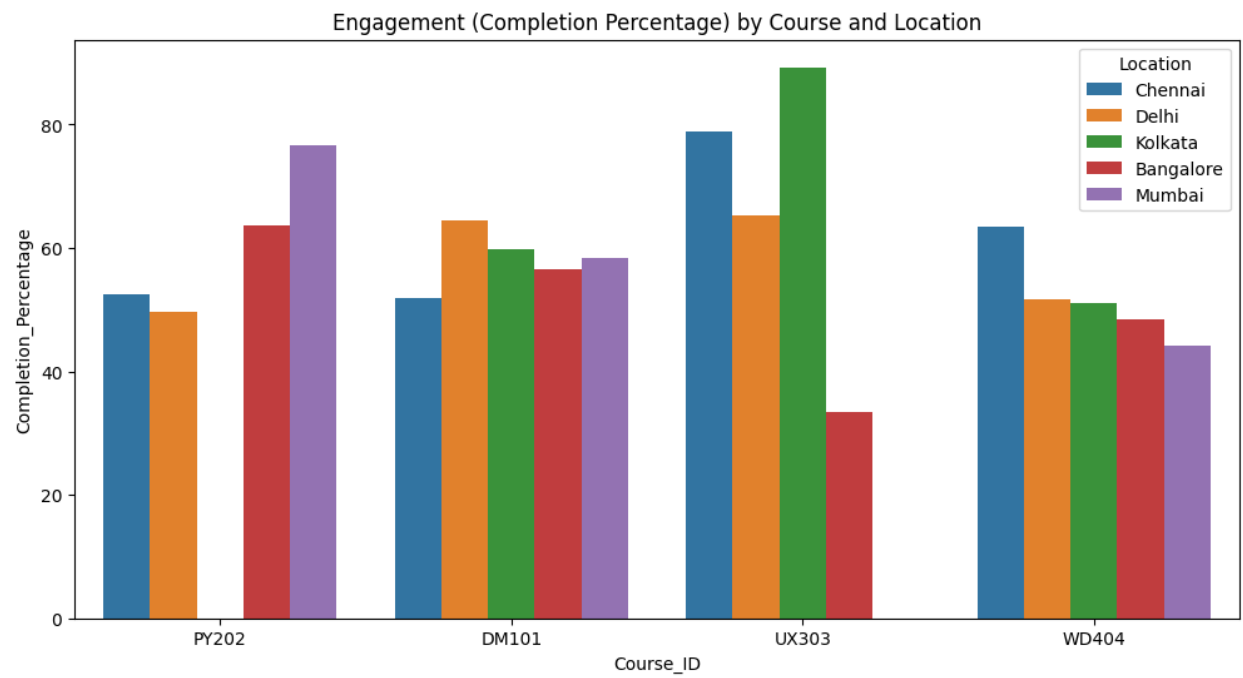
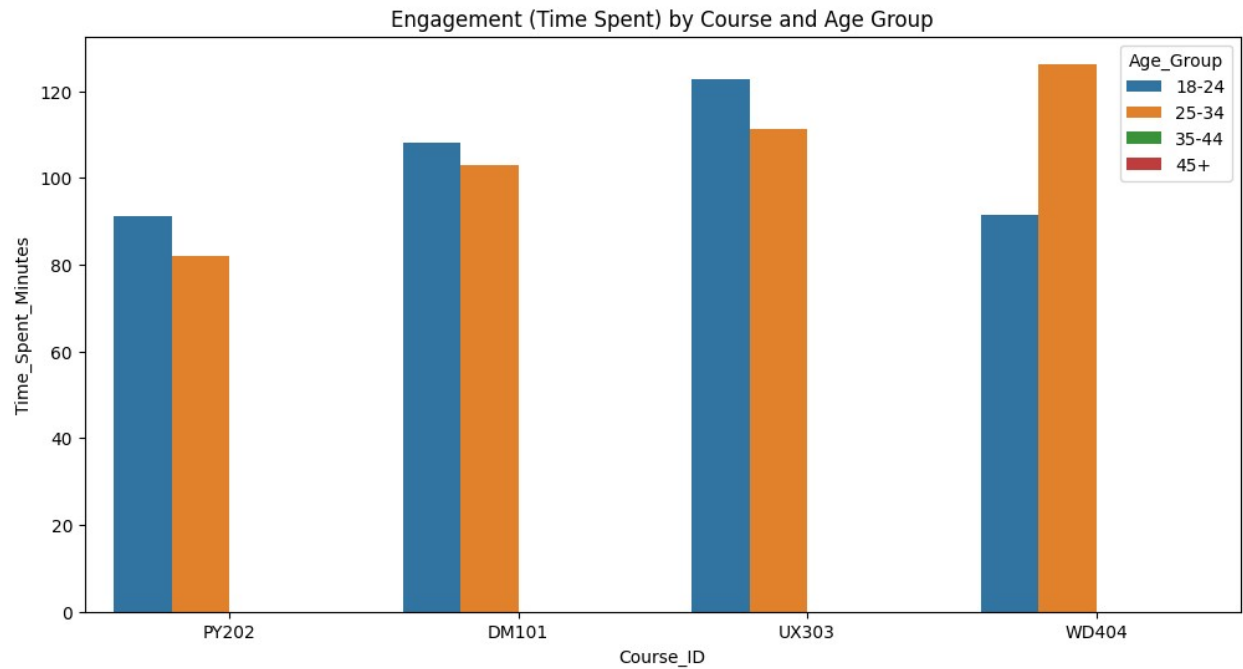
df_merged['Date'] = pd.to_datetime(df_merged['Date'], errors='coerce')
# Group and aggregate the data
time_series_data = df_merged.groupby(['Course_ID', 'Date'])
[['Time_Spent_Minutes', 'Completion_Percentage']].mean().reset_index()
# Plotting
plt.figure(figsize=(12, 6))
for course in time_series_data['Course_ID'].unique():
    course_data = time_series_data[time_series_data['Course_ID'] ==
    course]
    plt.plot(course_data['Date'], course_data['Time_Spent_Minutes'],
    label=f'Course {course}')
plt.title('Average Time Spent Over Time')
plt.xlabel('Date')
plt.ylabel('Average Time Spent (Minutes)')
plt.legend()
plt.show()

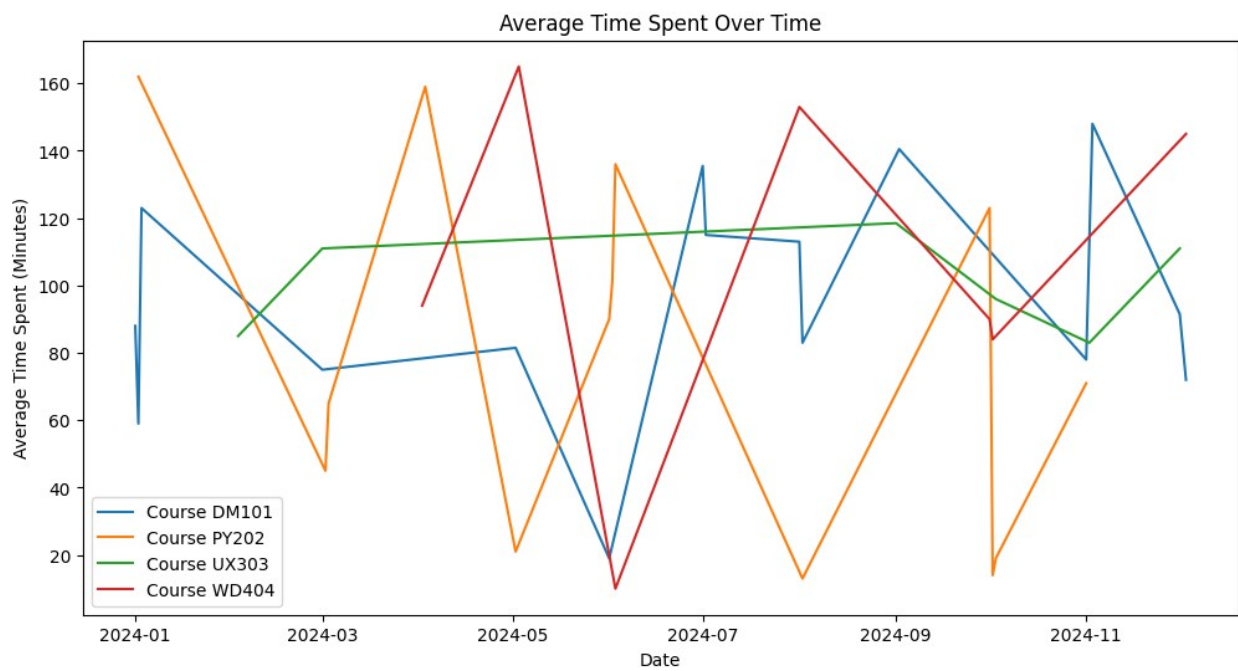
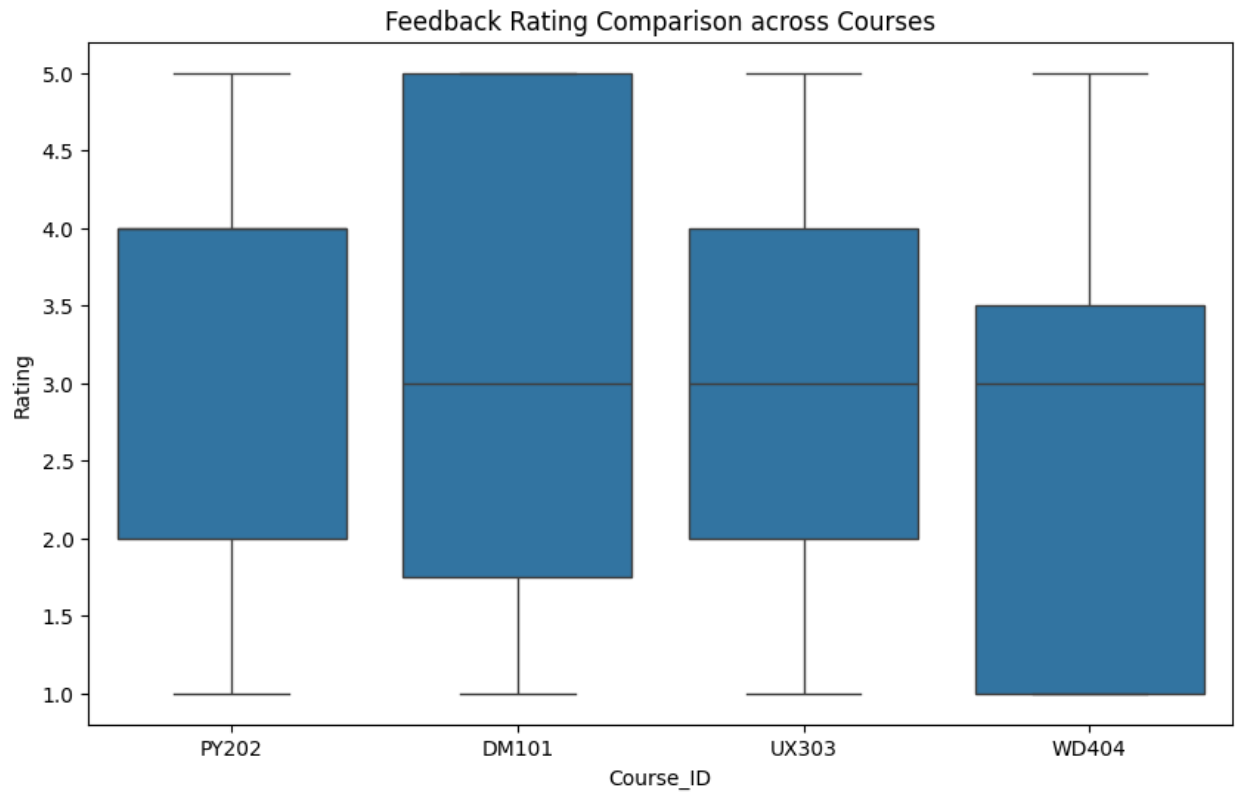
plt.figure(figsize=(12, 6))
for course in time_series_data['Course_ID'].unique():
    course_data = time_series_data[time_series_data['Course_ID'] ==
    course]
    plt.plot(course_data['Date'],
    course_data['Completion_Percentage'], label=f'Course {course}')
plt.title('Average Completion Percentage Over Time')
plt.xlabel('Date')
plt.ylabel('Average Completion Percentage')
plt.legend()
plt.show()

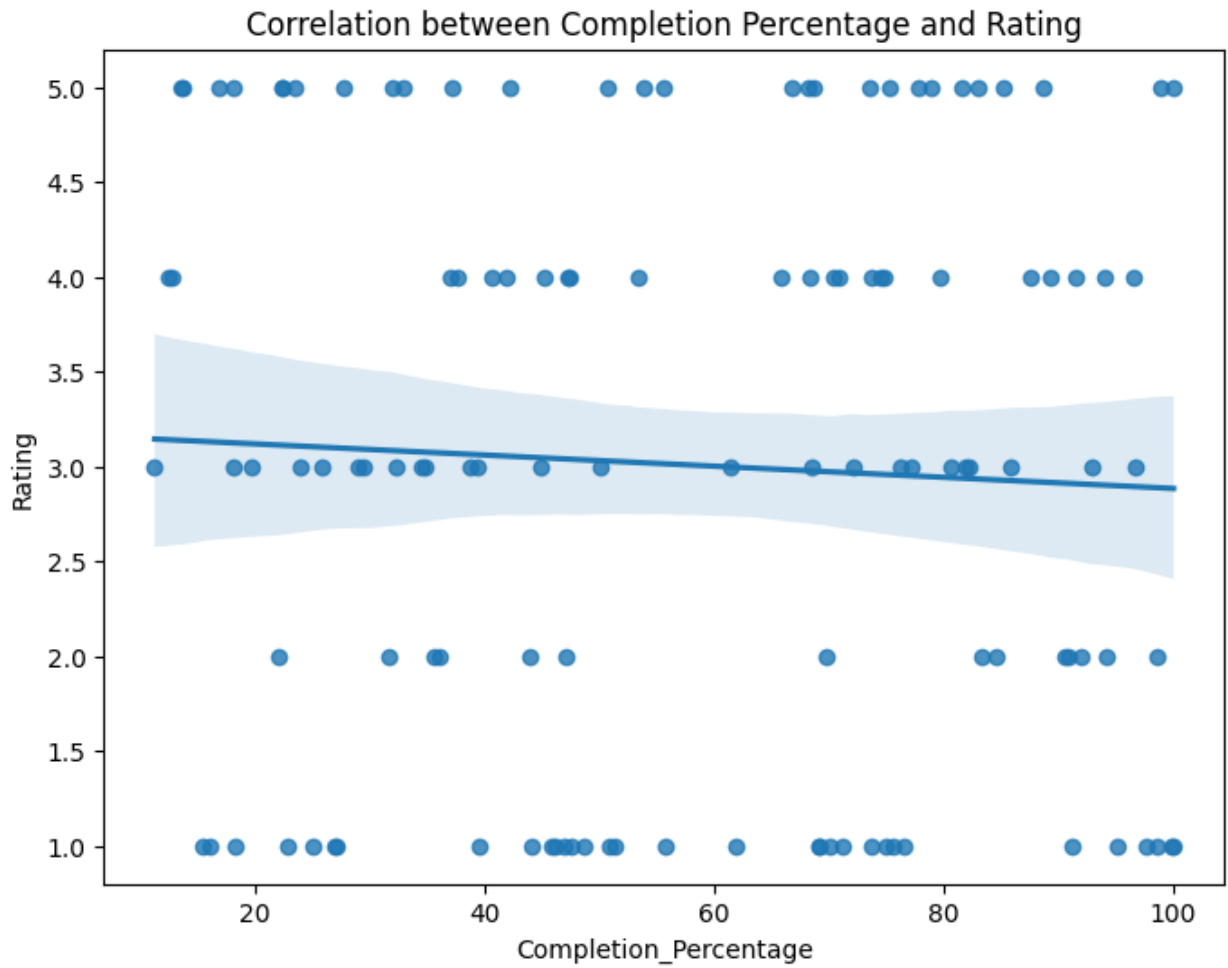
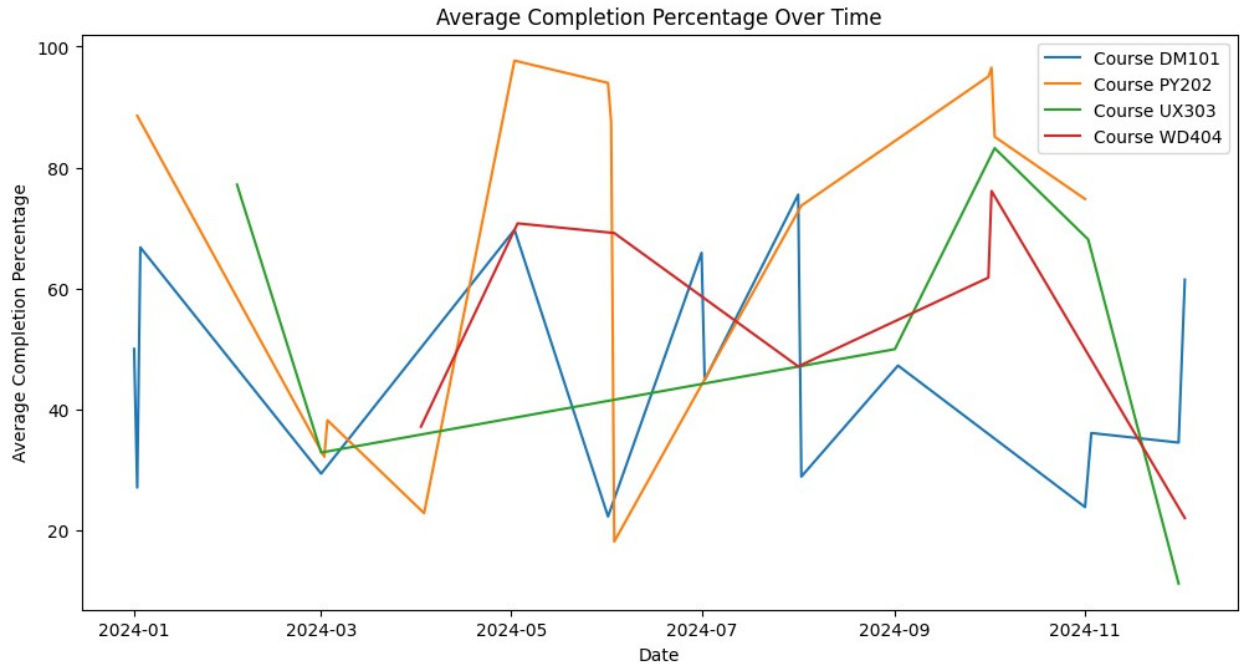
# 4. Correlation Visualization
plt.figure(figsize=(8, 6))
sns.regplot(x='Completion_Percentage', y='Rating', data=df_merged)
plt.title('Correlation between Completion Percentage and Rating')
plt.show()

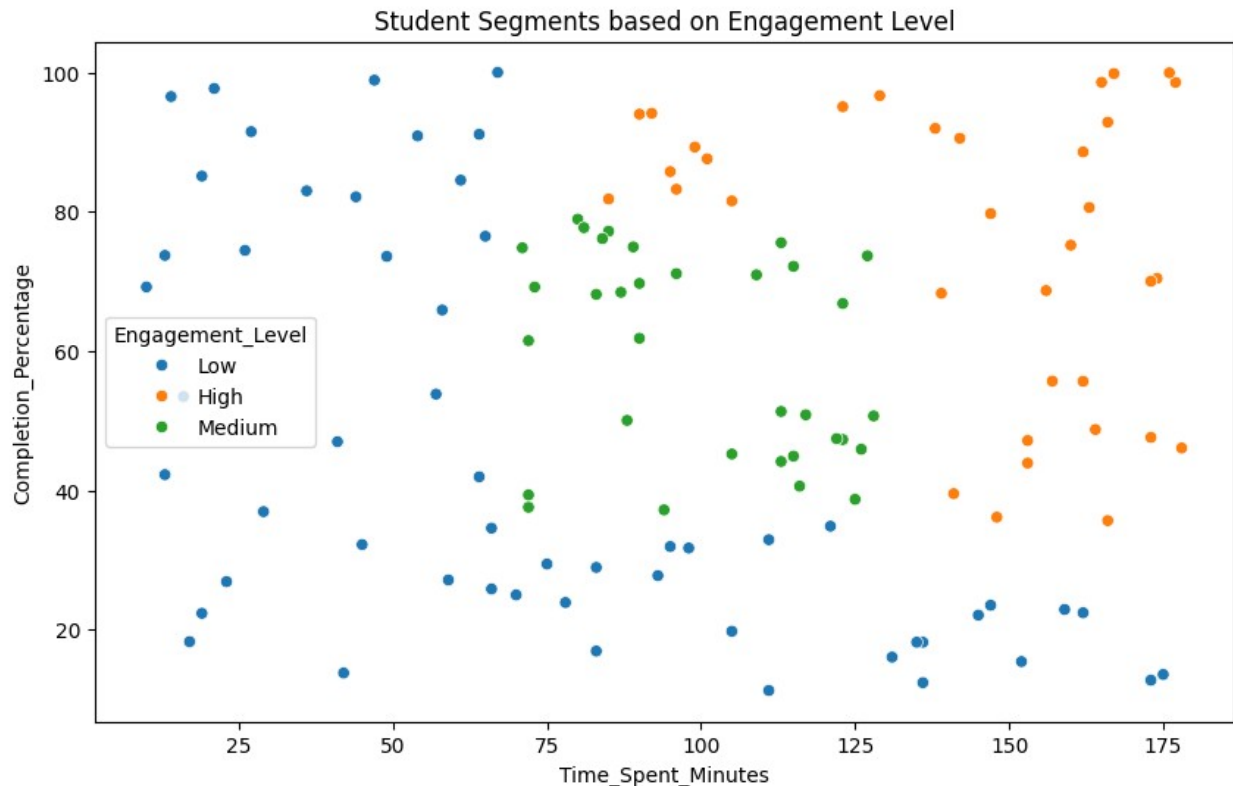
# 5. Student Segment Visualization (using Engagement_Level and
Time_Spent_Minutes)
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Time_Spent_Minutes', y='Completion_Percentage',
hue='Engagement_Level', data=df_merged)
plt.title('Student Segments based on Engagement Level')
plt.show()

```









Summary:

1. Q&A

- **Overall average completion rate across courses:** 57.38%
- **Course with highest average engagement time:** UX303 (115.35 minutes)
- **Course with lowest average engagement time:** PY202 (85.95 minutes)
- **Engagement differences by age group/location:** The analysis provided the average time spent and completion percentage for each age group and location. Note that age groups 35-44 and 45+ had no data. Detailed breakdowns are available in the provided output.
- **Average feedback rating per course:** The analysis calculated and displayed the average feedback rating for each course. Specific ratings are included in the execution history.
- **Correlation between completion rate and feedback rating:** A weak negative correlation of -0.05 was found.
- **Top 3 student segments based on engagement and satisfaction:** The top three segments, based on student counts, are:
 - Low Engagement, Rating 5 (17 students)
 - Low Engagement, Rating 1 (13 students)
 - Low Engagement, Rating 3 (11 students)

2. Data Analysis Key Findings

- **Course Performance Discrepancy:** Significant differences in average engagement time exist across courses, with UX303 showing the highest (115.35 minutes) and PY202 the

lowest (85.95 minutes). This suggests a need for further investigation into course content and delivery methods.

- **Weak Negative Correlation between Completion and Feedback:** A correlation of -0.05 was observed between completion rate and feedback rating, indicating a very weak negative relationship. This is unexpected and warrants further analysis to understand the underlying factors.
- **Low Engagement Segment Dominance:** The top three student segments are all characterized by "Low Engagement," highlighting a potential area of concern. Further analysis should explore the reasons behind low engagement, particularly among students providing high (5) and low (1) ratings. The observation suggests that engagement might be a separate factor than satisfaction.

3. Insights or Next Steps

- **Investigate Low Engagement:** Prioritize understanding the root causes of low engagement across different courses and student demographics. A/B testing different instructional methods for low-performing courses may be beneficial. Qualitative feedback analysis within the low engagement segments is crucial.
- **Re-examine Completion/Feedback Relationship:** The weak negative correlation between completion rate and feedback rating is counterintuitive and needs further investigation. Explore potential confounding variables or biases in data collection. Consider changing the feedback mechanism to collect more in-depth data.

```
pip install reportlab
```

```
Collecting reportlab
```

```
  Downloading reportlab-4.4.0-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pillow>=9.0.0 in
/usr/local/lib/python3.11/dist-packages (from reportlab) (11.1.0)
Requirement already satisfied: chardet in
/usr/local/lib/python3.11/dist-packages (from reportlab) (5.2.0)
Downloading reportlab-4.4.0-py3-none-any.whl (2.0 MB)
----- 2.0/2.0 MB 20.0 MB/s eta
```

```
0:00:00
```

```
import pandas as pd
from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer,
Table, TableStyle
from reportlab.lib.styles import getSampleStyleSheet
```

```
-----
-----
ModuleNotFoundError                                Traceback (most recent call
last)
<ipython-input-8-83f0174a2757> in <cell line: 0>()
      1 import pandas as pd
----> 2 from reportlab.lib.pagesizes import letter
      3 from reportlab.platypus import SimpleDocTemplate, Paragraph,
Spacer, Table, TableStyle
```

```
4 from reportlab.lib.styles import getSampleStyleSheet
```

```
ModuleNotFoundError: No module named 'reportlab'
```

```
-----  
-----  
NOTE: If your import is failing due to a missing package, you can  
manually install dependencies using either !pip or !apt.  
  
To view examples of installing some common dependencies, click the  
"Open Examples" button below.  
-----  
-----
```

```
def generate_report(df, report_type="pdf", filename="report"):  
    """  
    Generates a report on the cleaned dataset.  
  
    Args:  
        df: The cleaned pandas DataFrame.  
        report_type: The type of report to generate ("pdf" or "csv").  
        filename: The name of the output file.  
    """  
    if report_type == "pdf":  
        doc = SimpleDocTemplate(f"{filename}.pdf", pagesize=letter)  
        styles = getSampleStyleSheet()  
        story = []  
  
        # Add title  
        story.append(Paragraph("Data Analysis Report", styles['h1']))  
        story.append(Spacer(1, 12))  
  
        # Add summary statistics  
        summary_stats = df[['Age', 'Time_Spent_Minutes',  
        'Completion_Percentage', 'Rating']].describe()  
        summary_data = [summary_stats.columns.tolist()] +  
summary_stats.values.tolist()  
        table = Table(summary_data)  
        table.setStyle(TableStyle([('BACKGROUND', (0, 0), (-1, 0),  
        '#CCCCC'),  
        ('TEXTCOLOR', (0, 0), (-1, 0),  
        '#000000'),  
        ('ALIGN', (0, 0), (-1, -1),  
        'CENTER'),  
        ('FONTNAME', (0, 0), (-1, 0),  
        'Helvetica-Bold'),  
        ('BOTTOMPADDING', (0, 0), (-1, 0),  
        12),  
        ('GRID', (0, 0), (-1, -1), 1,
```



```

'#000000']]))
    story.append(Paragraph("Summary Statistics", styles['h2']))
    story.append(table)
    story.append(Spacer(1, 24))

    # Add other sections (e.g., key findings, insights) as needed
    # ...

    doc.build(story)

elif report_type == "csv":
    df.to_csv(f"{filename}.csv", index=False)

else:
    print("Invalid report type. Please choose 'pdf' or 'csv'.")

from reportlab.lib.pagesizes import letter
from reportlab.lib.styles import getSampleStyleSheet

from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer,
Table, TableStyle

# Load the cleaned dataset
df_cleaned = pd.read_csv('cleaned_dataset.csv')

# Generate a PDF report
generate_report(df_cleaned, report_type="pdf",
filename="data_analysis_report")

# Generate a CSV report (optional)
generate_report(df_cleaned, report_type="csv",
filename="data_analysis_data")

!pip install reportlab==3.6.12

Collecting reportlab==3.6.12
  Downloading reportlab-3.6.12-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (1.2 kB)
Requirement already satisfied: pillow>=9.0.0 in
/usr/local/lib/python3.11/dist-packages (from reportlab==3.6.12)
(11.1.0)
Downloading reportlab-3.6.12-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.8 MB)
----- 2.8/2.8 MB 25.2 MB/s eta
0:00:00
pting uninstall: reportlab
  Found existing installation: reportlab 4.4.0
  Uninstalling reportlab-4.4.0:
    Successfully uninstalled reportlab-4.4.0
Successfully installed reportlab-3.6.12

```