```
pip install pandas matplotlib seaborn
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/d
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: mkl_random in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: mkl in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: tbb4py in /usr/local/lib/python3.11/dist-packages (fr
Requirement already satisfied: mkl-service in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: onemkl-license==2025.3.0 in /usr/local/lib/python3.11
Requirement already satisfied: intel-openmp<2026,>=2024 in /usr/local/lib/python3.11
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: intel-cmplr-lib-rt in /usr/local/lib/python3.11/dist-
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in /usr/local/lib/python
Note: you may need to restart the kernel to use updated packages.
```

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

GRADE_POINTS = {
    'A': 5,
    'B': 4,
    'C': 3,
    'D': 2,
    'E': 1,
    'F': 0
}

def get_point(grade):
    """Converts letter grade to point value. Handles invalid inputs."""
    return GRADE_POINTS.get(grade.upper().strip(), 0)
```

```
Start coding or generate with AI.
```

```
def load_and_analyze_data(filepath):

    try:
        df = pd.read_csv('/kaggle/working/transcript.csv')
    except FileNotFoundError:
        return None, "File not found. Please check the path."


    if df.isnull().values.any():
        print("Warning: Missing values detected. Dropping incomplete rows...")
        df = df.dropna()


    df['Point'] = df['Grade'].apply(get_point)


    df['Weighted_Point'] = df['Unit'] * df['Point']


    semester_stats = df.groupby('Semester').apply(
        lambda x: pd.Series({
            'Total_Units': x['Unit'].sum(),
            'Total_Points': x['Weighted_Point'].sum(),
            'GPA': x['Weighted_Point'].sum() / x['Unit'].sum()
        })
    ).reset_index()


    semester_stats['Cumulative_Units'] = semester_stats['Total_Units'].cumsum()
    semester_stats['Cumulative_Points'] = semester_stats['Total_Points'].cumsum()
    semester_stats['CGPA'] = semester_stats['Cumulative_Points'] / semester_stats['

    return df, semester_stats


df, stats = load_and_analyze_data('results.csv')
print("--- Semester Analysis ---")
print(stats[['Semester', 'GPA', 'CGPA']])
```

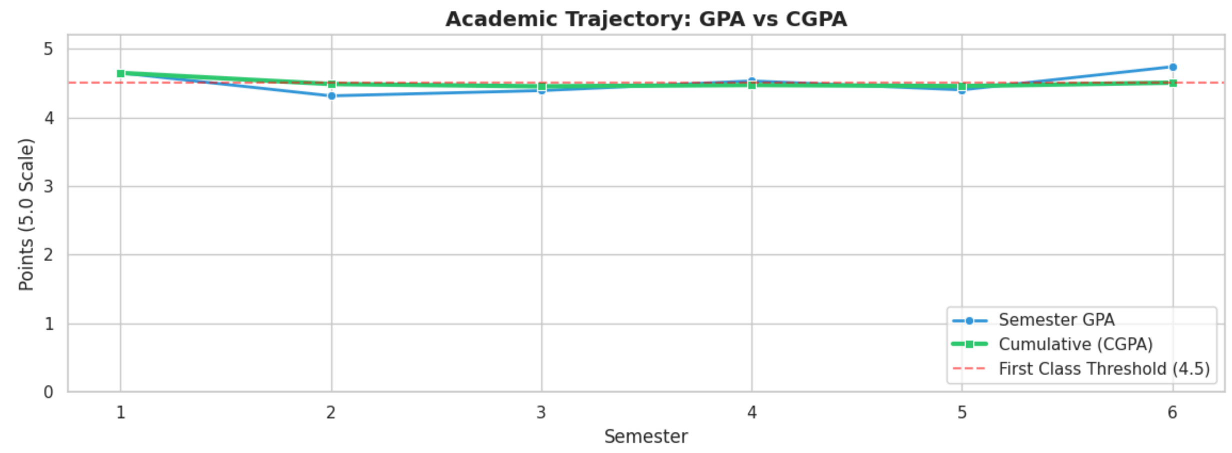
```
--- Semester Analysis ---
   Semester       GPA      CGPA
0         1  4.647059  4.647059
1         2  4.312500  4.484848
2         3  4.388889  4.450980
3         4  4.529412  4.470588
4         5  4.400000  4.454545
5         6  4.736842  4.504673
/tmp/ipykernel_111/2194594193.py:20: DeprecationWarning: DataFrameGroupBy.apply oper
  semester_stats = df.groupby('Semester').apply(
```

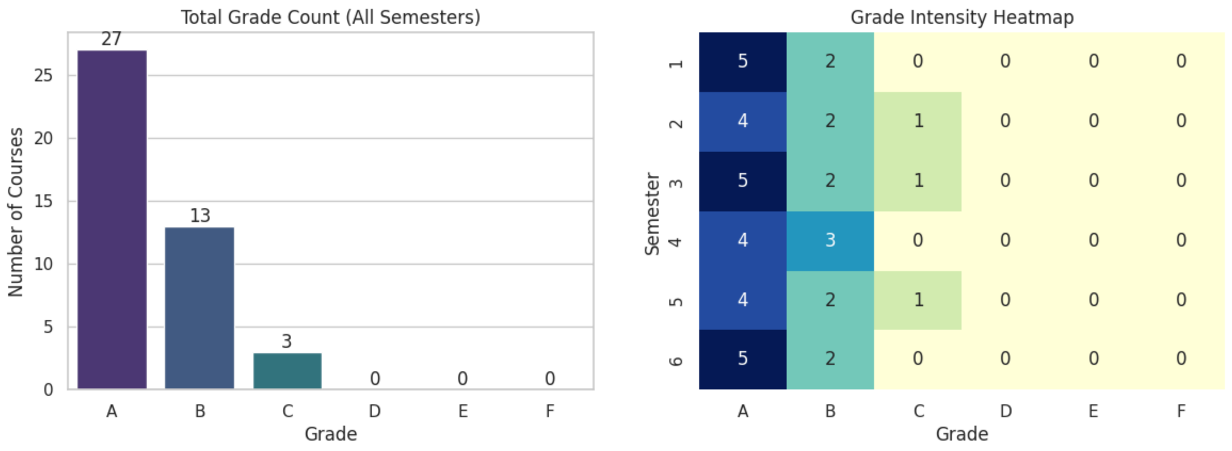
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

```
def plot_advanced_dashboard(df, stats):

    sns.set_theme(style="whitegrid")


    fig = plt.figure(figsize=(14, 10))
    grid = plt.GridSpec(2, 2, height_ratios=[1, 1], hspace=0.3)


    ax1 = fig.add_subplot(grid[0, :]) # Span entire top row

    sns.lineplot(data=stats, x='Semester', y='GPA', marker='o', label='Semester GPA
    sns.lineplot(data=stats, x='Semester', y='CGPA', marker='s', label='Cumulative


    ax1.axhline(y=4.5, color='red', linestyle='--', alpha=0.5, label='First Class T

    ax1.set_title('Academic Trajectory: GPA vs CGPA', fontsize=14, fontweight='bold
    ax1.set_ylim(0, 5.2)
    ax1.set_ylabel('Points (5.0 Scale)')
    ax1.legend(loc='lower right')


    ax2 = fig.add_subplot(grid[1, 0])


    grade_order = ['A', 'B', 'C', 'D', 'E', 'F']
    sns.countplot(data=df, x='Grade', order=grade_order, palette='viridis', ax=ax2)

    ax2.set_title('Total Grade Count (All Semesters)', fontsize=12)
    ax2.set_ylabel('Number of Courses')
    ax2.bar_label(ax2.containers[0]) # Show numbers on top of bars


    ax3 = fig.add_subplot(grid[1, 1])

    heatmap_data = pd.crosstab(df['Semester'], df['Grade'])

    for g in grade_order:
        if g not in heatmap_data.columns:
            heatmap_data[g] = 0
    heatmap_data = heatmap_data[grade_order] # Sort columns

    sns.heatmap(heatmap_data, annot=True, cmap="YlGnBu", cbar=False, ax=ax3, fmt='d

    ax3.set_title('Grade Intensity Heatmap', fontsize=12)
    ax3.set_ylabel('Semester')

    plt.show()

if 'df' in locals() and 'stats' in locals():
    plot_advanced_dashboard(df, stats)
```

```
else:
    print("ERROR, run the code blocks ahead first na")
```

```
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119: FutureWarning: use
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119: FutureWarning: use
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119: FutureWarning: use
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119: FutureWarning: use
  with pd.option_context('mode.use_inf_as_na', True):
```





	A	B	C	D	E	F
1	5	2	0	0	0	0
2	4	2	1	0	0	0
3	5	2	1	0	0	0
4	4	3	0	0	0	0
5	4	2	1	0	0	0
6	5	2	0	0	0	0

```
def scenario_predictor(current_cum_points, current_cum_units, hypothetical_courses)
    """
    hypothetical_courses: List of tuples -> [('CourseName', Unit, 'TargetGrade'), .
    """
    projected_points = 0
    projected_units = 0

    print("\n--- Scenario Prediction ---")
    for course, unit, grade in hypothetical_courses:
        points = get_point(grade)
        projected_units += unit
        projected_points += (unit * points)
        print(f"Target: {course} ({unit} units) -> Grade {grade}")
    new_total_units = current_cum_units + projected_units
    new_total_points = current_cum_points + projected_points
    predicted_cgpa = new_total_points / new_total_units

    print(f"Projected Units Added: {projected_units}")
    print(f"Predicted New CGPA: {predicted_cgpa:.2f}")

    return predicted_cgpa


current_cum_points = stats['Cumulative_Points'].iloc[-1]
current_cum_units = stats['Cumulative_Units'].iloc[-1]

next_semester_plan = [
    ('CSC401', 3, 'A'),
    ('CSC402', 3, 'A'),
    ('GED400', 2, 'B')
]

scenario_predictor(current_cum_points, current_cum_units, next_semester_plan)
```

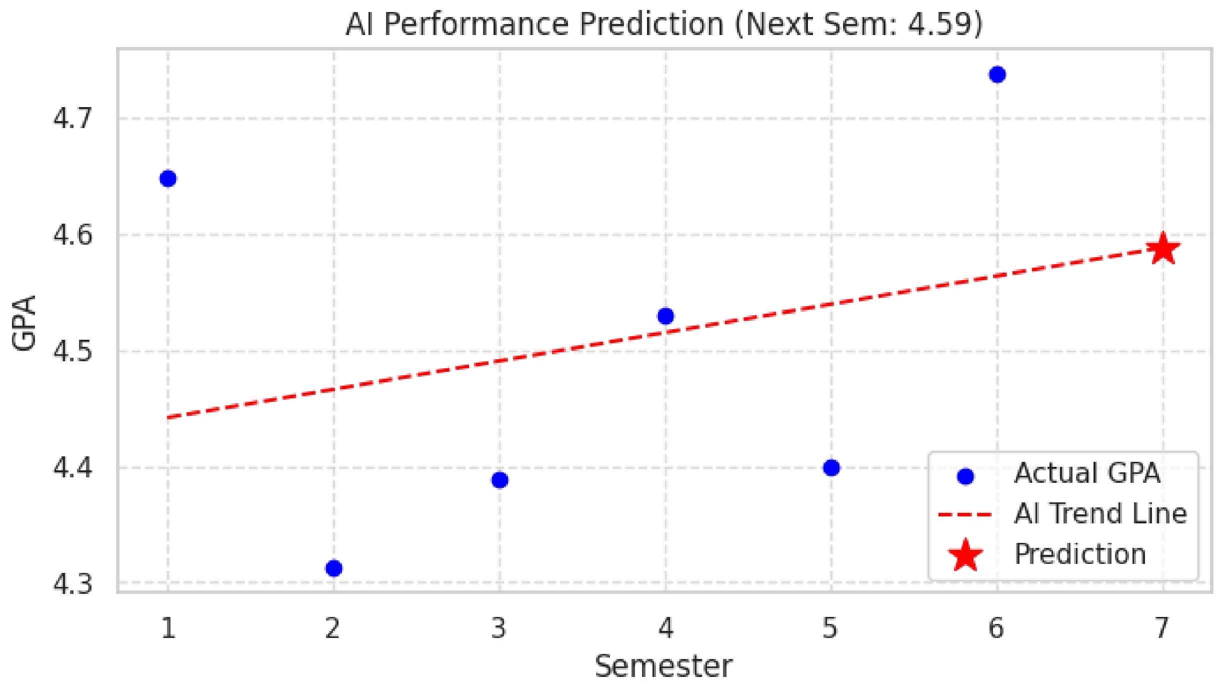
```
--- Scenario Prediction ---
Target: CSC401 (3 units) -> Grade A
Target: CSC402 (3 units) -> Grade A
Target: GED400 (2 units) -> Grade B
Projected Units Added: 8
Predicted New CGPA: 4.52
4.521739130434782
```

```
!pip install scikit-learn huggingface_hub
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: huggingface_hub in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.11/dis
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: mkl_random in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.11/dist-packages 
Requirement already satisfied: mkl in /usr/local/lib/python3.11/dist-packages (from 
Requirement already satisfied: tbb4py in /usr/local/lib/python3.11/dist-packages (fr
Requirement already satisfied: mkl-service in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.11
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: onemkl-license==2025.3.0 in /usr/local/lib/python3.11
Requirement already satisfied: intel-openmp<2026,>=2024 in /usr/local/lib/python3.11
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: intel-cmplr-lib-rt in /usr/local/lib/python3.11/dist-
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in /usr/local/lib/python
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

def analyze_trend_and_predict(stats):

    X = stats['Semester'].values.reshape(-1, 1)
    y = stats['GPA'].values


    if len(X) < 2:
        print("Not enough data to predict trends (need at least 2 semesters).")
        return


    model = LinearRegression()
    model.fit(X, y)


    next_semester = stats['Semester'].max() + 1
    predicted_gpa = model.predict([[next_semester]])[0]


    predicted_gpa = max(0.0, min(5.0, predicted_gpa))

    print(f"--- AI Trend Analysis ---")
    print(f"Based on your current trajectory, your predicted GPA for Semester {next


    plt.figure(figsize=(8, 4))


    plt.scatter(X, y, color='blue', label='Actual GPA')


    line_x = np.array([X.min(), next_semester]).reshape(-1, 1)
    line_y = model.predict(line_x)
    plt.plot(line_x, line_y, color='red', linestyle='--', label='AI Trend Line')


    plt.scatter([next_semester], [predicted_gpa], color='red', marker='*', s=200, l

    plt.title(f'AI Performance Prediction (Next Sem: {predicted_gpa:.2f})')
    plt.xlabel('Semester')
    plt.ylabel('GPA')
    plt.legend()
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.show()


if 'stats' in locals():
    analyze_trend_and_predict(stats)
else:
    print("Please run the Data Loading step first!")
```

```
--- AI Trend Analysis ---
Based on your current trajectory, your predicted GPA for Semester 7 is: 4.59
```



```
from huggingface_hub import InferenceClient
import re

# PASTE YOUR TOKEN HERE
HF_TOKEN = "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"

def clean_response(text):
    text = re.sub(r'\[.*?\]', '', text)
    return text.strip()

def get_ai_advice(stats):
    if "hf_" not in HF_TOKEN:
        print("⚠ Please paste a valid Hugging Face token.")
        return

    # --- 1. DYNAMIC NAME INPUT ---
    # This asks the user for their name at runtime!
    print("------------------------------------------------")
    user_name = input("Enter your name (Press Enter to skip): ").strip()
    # Default to "Scholar" if they don't type anything
    if not user_name:
        user_name = "Scholar"
    print("------------------------------------------------")

    # 2. Gather Context
    current_cgpa = stats['CGPA'].iloc[-1]
    last_gpa = stats['GPA'].iloc[-1]

    # 3. Determine Vibe
    if last_gpa >= current_cgpa:
        vibe = "Positive. The user is improving."
    else:
        vibe = "Encouraging. The user slipped a bit."

    # 4. Prompt with Dynamic Name
    prompt_content = f"""
    You are an expert academic coach speaking to {user_name}.

    User Stats:
    - Current CGPA: {current_cgpa:.2f}
    - Last Term GPA: {last_gpa:.2f}
    - Trend: {vibe}

    Instructions:
    1. Start by addressing {user_name} directly.
    2. Give a warm, specific compliment about their efforts.
    3. Provide exactly 3 high-impact, actionable steps to reach a First Class.
    4. End with a short motivational line.

    CONSTRAINTS:
    - Keep it conversational and supportive.
    - Total length: UNDER 150 words.
    - No scripts. No [Tags].
    """

    print(f"Thinking... (Coach is analyzing stats for {user_name})...")

    try:
        client = InferenceClient(token=HF_TOKEN)

        response = client.chat_completion(
            model="HuggingFaceH4/zephyr-7b-beta",
            messages=[
                {"role": "system", "content": "You are a helpful, concise academic
                {"role": "user", "content": prompt_content}
            ],
            max_tokens=1024,
            temperature=0.7
        )

        raw_advice = response.choices[0].message.content
        final_advice = clean_response(raw_advice)

        print("\n--- 🧠 Coach's Feedback ---")
        print(final_advice)

    except Exception as e:
        print(f"Error connecting to AI: {e}")

# Run it
if 'stats' in locals():
    get_ai_advice(stats)
```

```
------------------------------------------------
Enter your name (Press Enter to skip):  ilori
------------------------------------------------
Thinking... (Coach is analyzing stats for ilori)...

--- 🧠 Coach's Feedback ---
Hi, ilori! Your efforts have paid off this term as you've improved your CGPA to a st
    1. Prioritize time management. Create a calendar with deadlines for assignments,
    2. Apply active learning techniques, like summarizing notes and teaching a frien
    3. Seek help early for any confusion. Don't wait until it's too late. Remember,
```