Marks Analyzer and Grading System

November 23, 2025

1 Introduction and Project Goal

1.1 Project Overview

This project involves developing a robust script to process raw academic marks data, calculate performance metrics, and generate a consolidated report. The primary goal is to automate the tedious tasks of totaling scores, calculating percentages, and assigning corresponding letter grades for a class of students, culminating in a comprehensive output file and summary statistics.

1.2 Objective

The main objectives of the script are:

1. Read student marks from a standardized input file (`marks.csv`).
2. Calculate the total score and overall percentage for each student across five specified subjects.
3. Apply a predefined grading policy to assign a letter grade.
4. Generate key class-level statistics (average, topper, lowest performer).
5. Persist the final results to a new, structured output file (`report.csv`).

2 Approach and Implementation

The solution is implemented in Python, leveraging the standard `csv` library for file handling. The approach follows a three-stage pipeline: **Data Ingestion**, **Data Processing**, and **Report Generation**.

2.1 Data Ingestion (Reading `marks.csv`)

The script uses `csv.DictReader` to ingest the input data. This allows each row to be treated as a dictionary, making mark retrieval easy and robust using subject column names (e.g., `row["Calculus"]`).

2.2 Data Processing and Grading Logic

2.2.1 Performance Calculation

For each student, the script performs the following calculations:

1. **Total Score:** Sums the scores from the five defined subjects.

2. **Percentage:** Calculated as Total Score$/($Number of Subjects$\times 100)\times 100$. Since all subjects are scored out of 100, the total maximum possible score is 500. The result is rounded to two decimal places for readability.

2.2.2 Grading Function

A dedicated function, `get_grade(percentage)`, implements the grading policy using a series of cascaded `if-elif` statements.

$$\text{Grade} = \begin{cases} A+ & \text{if Percentage} \geq 90 \\ A & \text{if } 80 \leq \text{Percentage} < 90 \\ B & \text{if } 70 \leq \text{Percentage} < 80 \\ C & \text{if } 60 \leq \text{Percentage} < 70 \\ D & \text{if } 50 \leq \text{Percentage} < 60 \\ F & \text{if Percentage} < 50 \end{cases}$$

2.3 Class Statistics Calculation

After processing all students, the script calculates summary statistics:

- **Class Average Percentage:** The arithmetic mean of all students' percentages.
- **Topper:** Determined using the `max()` function with a lambda key based on the `Percentage` field.
- **Lowest Performer:** Determined using the `min()` function with a lambda key based on the `Percentage` field.

3 Results and Analysis

3.1 Console Output (Simulated Results)

The script generates a detailed printout to the console for immediate review:

```
===== CLASS REPORT =====
101 | Alice Smith | Total: 458 | %: 91.6 | Grade: A+
102 | Bob Johnson | Total: 387 | %: 77.4 | Grade: B
103 | Charlie Brown | Total: 330 | %: 66.0 | Grade: C
104 | Diana Prince | Total: 487 | %: 97.4 | Grade: A+
105 | Eve Adams | Total: 262 | %: 52.4 | Grade: D

===== CLASS STATISTICS =====
Class Average Percentage: 77.96
Topper: Diana Prince -- 97.4\%
Lowest: Eve Adams -- 52.4\%

Report saved to file: report.csv
```

3.2 Generated Report File (`report.csv`)

A new file, `report.csv`, is generated containing the structured output, which includes the calculated total, percentage, and final grade for each student. This output is suitable for archiving or further analysis in spreadsheet software.

Structure of `report.csv`:

Table 1: Example Data from the Generated Report File

Roll	Name	Total	Percentage	Grade
101	Alice Smith	458	91.60	A+
104	Diana Prince	487	97.40	A+
...	...	...	...	...

3.3 Data Analysis

The class statistics provide a quick, high-level understanding of the overall performance.

- **Class Average (77.96%):** Indicates a generally strong performance, falling within the 'B' grade range according to the defined policy.
- **Performance Distribution:** The data shows a healthy spread, with top performers achieving A+ status (Alice Smith, Diana Prince) and the lowest score still managing a 'D' grade (Eve Adams). The two A+ grades significantly elevate the class average.

4 Conclusion and Future Enhancements

The script successfully fulfills all project objectives, providing an efficient and reliable tool for processing student marks and generating a comprehensive report. The use of clear functions and standard libraries ensures maintainability and portability.

4.1 Recommended Future Enhancements

1. **Weighted Grading:** Modify the calculation logic to support different weights for subjects (e.g., Calculus counting for 2x the weight of English).
2. **Error Handling:** Implement `try-except` blocks to handle non-numeric data in the subject columns or missing input files gracefully, preventing runtime crashes.
3. **Data Persistence:** Extend the script to save the generated statistics (Class Average, Topper name) into a separate summary file for trend analysis over multiple reporting periods.
4. **Flexible Subject List:** Allow the script to dynamically determine the subject columns based on the CSV header, rather than hardcoding the `subject_columns` list, improving flexibility for different datasets.