ASSIGNMENT NO.- 1

GradeBook Analyzer – Project Report

NAME- ADITYA CHAUHAN

ROLL NO. – 2501730276

COURSE- BTECH CSE (AIML)

SUBJECT- PROBLEM SOLVING USING PYTHON

FACULTY- SAMEER FAROOQ

Introduction

The GradeBook Analyzer is a Python-based command-line tool that helps lecturers quickly analyze student marks. Instead of calculating averages, grades, and statistics manually, this program automates the entire process. It supports both manual entry of student data and importing marks from CSV files.

---

2. Objectives

The main goals of this project were to:

- Read and store student marks efficiently.
- Perform key statistical calculations such as average, median, highest, and lowest score.
- Automatically assign letter grades based on score ranges.
- Count grade distribution and identify pass/fail students.
- Display results in a clean, formatted table.
- Allow repeated analysis through an interactive CLI loop.

---

3. Methodology

The project was implemented using Python and structured into several modular functions:

Data Input

Users may choose:

- Manual input of names and marks
- Importing data from a CSV file

The marks are stored in a dictionary:
{"Alice": 78, "Bob": 92, ...}

Statistical Functions

Custom functions calculate:

- Average score
- Median score
- Maximum and minimum marks

Python's statistics module was used to simplify median calculation.

Grade Assignment

A grading function assigns letter grades using conditional logic:

- A (90+), B (80–89), C (70–79), D (60–69), F (< 60)

A grade distribution is also generated.

Pass/Fail Analysis

List comprehensions were used to identify:

- Students scoring ≥ 40 (pass)
- Students scoring < 40 (fail)

Output Presentation

The program prints a well-formatted table showing:

- Name
- Mark
- Grade

The user can also export results to a CSV file.

CLI Loop

A while loop allows the user to run multiple analyses until they choose to exit.

4. Results

The program successfully:

- Loads data from both manual input and CSV files
- Computes all required statistics
- Assigns grades and displays grade distribution
- Identifies pass/fail students
- Prints a clear results table
- Allows CSV export for reporting purposes

Tests were performed using five manually-entered students and one CSV file, all functioning correctly.

5. Conclusion

The GradeBook Analyzer meets all requirements of the assignment. It is efficient, user-friendly, and modular. It demonstrates effective use of Python concepts including functions, control flow, list comprehensions, dictionaries, and file handling.

The finished tool can help lecturers save time and reduce errors in evaluating student performance.

CODE:_

```python
import csv
import statistics

def calculate_average(marks):
    return sum(marks.values()) / len(marks) if marks else 0

def calculate_median(marks):
    return statistics.median(marks.values()) if marks else 0

def find_max_score(marks):
    return max(marks.values()) if marks else None

def find_min_score(marks):
    return min(marks.values()) if marks else None

def assign_grade(score):
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
        return "F"

def compute_grades(marks):
    return {name: assign_grade(score) for name, score in marks.items()}

def grade_distribution(grades):
    dist = {"A":0, "B":0, "C":0, "D":0, "F":0}
    for g in grades.values():
        dist[g] += 1
    return dist

def manual_entry():
    marks = {}
    print("\nEnter student names and marks. Type 'done' to finish.\n")
    while True:
        name = input("Student name: ")
        if name.lower() == "done":
            break
        try:
            score = float(input("Mark: "))
            marks[name] = score
        except:
            print("Invalid number.")
    return marks

def load_csv():
    filename = input("Enter CSV filename: ")
    marks = {}
    try:
        with open(filename, "r") as f:
```

```python
def load_csv():
        with open(filename, "r") as f:
            reader = csv.reader(f)
            next(reader, None)
            for row in reader:
                marks[row[0]] = float(row[1])
        print("CSV loaded.\n")
    except Exception as e:
        print("Error loading CSV:", e)
    return marks

def export_to_csv(marks, grades):
    filename = input("Export filename: ")
    with open(filename, "w", newline="") as f:
        writer = csv.writer(f)
        writer.writerow(["Name", "Marks", "Grade"])
        for name in marks:
            writer.writerow([name, marks[name], grades[name]])
    print("Exported.\n")

def print_results_table(marks, grades):
    print("\nName\t\tMarks\tGrade")
    print("-" * 40)
    for name, score in marks.items():
        print(f"{name:15s}{score:<10.1f}{grades[name]}")
    print()

def main():+
    print("==================================")
    print("      GRADEBOOK ANALYZER CLI")
    print("==================================")

    while True:
        print("\n1. Manual entry")
        print("2. Load CSV")
        print("3. Exit")
        choice = input("Enter choice: ")

        if choice == "1":
            marks = manual_entry()
        elif choice == "2":
            marks = load_csv()
        elif choice == "3":
            print("Goodbye!")
            break
        else:
            print("Invalid option.")
            continue

        if not marks:
            print("No data.")
            continue

        avg = calculate_average(marks)
        med = calculate_median(marks)
```

```
        mx = find_max_score(marks)
        mn = find_min_score(marks)

        grades = compute_grades(marks)
        dist = grade_distribution(grades)

        passed = [n for n, s in marks.items() if s >= 40]
        failed = [n for n, s in marks.items() if s < 40]

        print_results_table(marks, grades)

        print("Statistics:")
        print(f"Average: {avg:.2f}")
        print(f"Median: {med:.2f}")
        print(f"Max: {mx}")
        print(f"Min: {mn}\n")

        print("Grade Distribution:")
        for g, c in dist.items():
            print(f"{g}: {c}")

        print("\nPass/Fail:")
        print(f"Passed ({len(passed)}): {passed}")
        print(f"Failed ({len(failed)}): {failed}")

        if input("\nExport to CSV? (y/n): ").lower() == "y":
            export_to_csv(marks, grades)

        if input("\nRun again? (y/n): ").lower() != "y":
            print("Goodbye!")
            break

if __name__ == "__main__":
    main()
```

```
==================================
      GRADEBOOK ANALYZER CLI
==================================

1. Manual entry
2. Load CSV
3. Exit
Enter choice: 1

Enter student names and marks. Type 'done' to finish.

Student name: Aryan
Mark: 76
Student name: ankur
Mark: 87
Student name: done

Name            Marks   Grade
----------------------------------------
Aryan           76.0      C
ankur           87.0      B

Statistics:
Average: 81.50
Median: 81.50
Max: 87.0
Min: 76.0

Grade Distribution:
A: 0
B: 1
C: 1
D: 0
F: 0

Pass/Fail:
Passed (2): ['Aryan', 'ankur']
Failed (0): []
```