STUDENT ACADEMIC PERFORMANCE EVALUATION WITH PANDAS JOINS

Project Description:

The Student Analytics Web App is a comprehensive web-based platform built with Flask that allows educators and administrators to analyze and visualize student performance data. Users can upload multiple CSV datasets (students, courses, enrollments, grades), which are then cleaned, merged, and analyzed using Pandas. The system generates actionable insights such as student rankings, departmental performance, pass/fail trends, and subject difficulty levels. The app offers a responsive Bootstrap frontend with dynamic charts powered by Chart.js, enabling stakeholders to make informed academic decisions quickly and effectively.

Submitted By :

NAME : RUCHITHA S

USN : 4GW23CI045

Email : ruchithashivaswamy03@gmail.com

Date : 02/09/2025

TABLE OF CONTENTS

Project Objective :

1. **Centralize Student Performance Data**
    - Provide a single platform to upload and process CSV datasets for students, courses, enrollments, and grades.

2. **Automate Data Cleaning and Integration**
    - Automatically detect missing values, remove duplicates, and merge multiple datasets to create a clean, unified dataset.

3. **Enable Comprehensive Analytics**
    - Generate insights such as student rankings, department-level statistics, pass/fail trends, and subject-wise performance.

4. **Offer Data Visualization**
    - Use interactive charts and tables to simplify the interpretation of complex academic data.

5. **Improve Decision-Making**
    - Empower educators and administrators with actionable insights for curriculum planning, student support, and departmental evaluation.

6. **Support Downloadable Reports**
    - Allow users to export analysis results in JSON or Excel format for offline review and reporting.

7. **Deliver a User-Friendly Web Interface**
    - Provide a responsive dashboard using Bootstrap and Chart.js for easy navigation and clear data presentation.

Dataset Description :

The Student Analytics Web App uses **four main CSV datasets** that collectively capture student, course, enrollment, and grade information. These datasets are uploaded through the web interface and processed in the backend using **Pandas** for analysis.

1. **Students.csv :** StudentID, Name, Department

   Contains unique records of all students, their names, and their department information.

2. **Courses.csv :** CourseID, CourseName, Department

   Holds details of all courses offered, along with the department responsible for each course.

3. **Enrollments.csv :** StudentID, CourseID

   Maps students to the courses they are enrolled in.

4. **Grades.csv :** StudentID, CourseID, Grade

   Stores students' grades for each course, forming the core dataset for analytics.

Data Characteristics:

- **Format:** CSV files uploaded via the web dashboard
- **Data Cleaning:** Missing values filled, duplicates removed
- **Data Size:** Scales to thousands of records with efficient Pandas processing
- **Relationships:**
  - StudentID links students.csv, enrollments.csv, and grades.csv
  - CourseID links courses.csv, enrollments.csv, and grades.csv

Detailed Explanation :

Overview

The **Student Analytics Web App** is a Flask-based data analytics platform designed to help educational institutions analyze student academic performance. Users can upload **four CSV files**—students.csv, courses.csv, enrollments.csv, and grades.csv.
The backend, built with **Flask** and **Pandas**, processes these datasets to perform:

- Data cleaning (handling missing values, removing duplicates)
- Data merging (linking students, courses, and grades)
- Calculations (totals, averages, pass/fail counts, rankings)
- Visualization (department-level and subject-level performance charts)

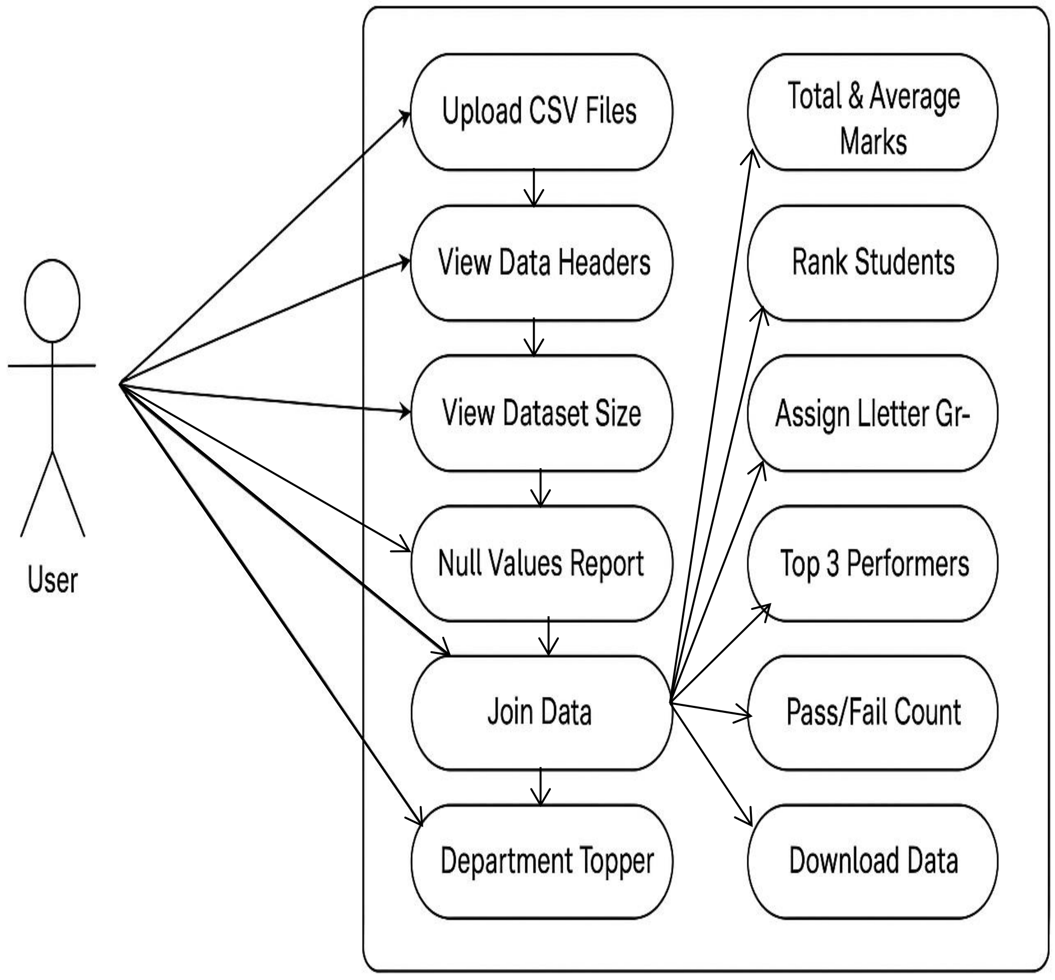
The **frontend** uses **Bootstrap** for a clean, responsive layout and **Chart.js** for interactive graphs.
Users can view results in tables, download them as JSON/Excel, and gain actionable insights on student performance.

Use Case Explanations :

1. Upload	Allows users to upload four CSV files containing all academic
2. View Data Headers	Displays the first 5 rows of each dataset for verification.
3. View Dataset Size	Shows number of rows and columns in each dataset.
4. Null Values Report	Counts missing values in each dataset.
5. Join Data (Inner, Left, Right)	Demonstrates join types for linking students with grades.
6. Total & Average Marks	Calculates total and average marks per student.
7. Rank Students	Sorts students by their average marks in descending order.
8. Assign Letter Grades	Converts numerical grades into A, B, C, D, F.
9. Top 3 Performers	Displays top 3 students overall.
10. Pass/Fail Count	Counts students passing vs. failing each subject.
11. Top Subject	Finds the subject with the highest average marks.
12. Department Topper	Displays topper of each department.
13. Toughest Subject	Finds subject with the most failures.
14. Department Analysis	Shows average, highest, lowest marks, and pass/fail counts
15. Download Data	Allows downloading analysis results as JSON/Excel.

UML Diagram :

USE CASE DIAGRAM :- Actor on the left and Use cases on Right.



Front-End (Interface) Design :

The front-end of this project is a **web-based dashboard** built using **HTML, CSS (Bootstrap 4), and JavaScript**. It provides a simple, intuitive interface for uploading datasets, visualizing analytics, and downloading reports.

Key Features:

- ❖ **Responsive Dashboard Layout**
  - Clean, mobile-friendly design using **Bootstrap**.
  - Sections are arranged as **cards** for clarity.
- ❖ **CSV Upload Section**
  - File inputs for **students**, **courses**, **enrollments**, and **grades** CSVs.
  - A "**Upload & Process**" button triggers the Flask /upload API.
- ❖ **Analytics Cards**
  - Each card represents a feature (e.g., Null Values, Rankings, Pass/Fail).
  - Includes buttons:
    - ➢ **View Result:** Displays table data
    - ➢ **View Graph:** Shows Chart.js graphs (if applicable)
    - ➢ **Download Excel:** Exports results in JSON/Excel format
- ❖ **Interactive Tables**
  - Tables are scrollable using table-responsive.
  - Sticky headers for large datasets.
- ❖ **Data Visualizations**
  - Charts are rendered dynamically with **Chart.js**.
  - Graphs highlight **department performance**, **pass/fail trends**, and **subject-wise statistics**.

Technologies Used:

- **HTML5**: Structure of the dashboard
- **Bootstrap 4.6**: Styling, responsiveness, prebuilt components
- **Chart.js**: Interactive and responsive data visualizations
- **JavaScript (ES6)**: Front-end logic, API calls

Code Walkthrough :

1. Importing Required Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
from flask import Flask, request, jsonify, render_template
```

- **pandas:** Handles reading, cleaning, and analyzing data from CSV files.
- **matplotlib.pyplot:** Creates plots and charts for data visualization.
- **flask:** Provides a web framework to build routes, handle requests, and serve a frontend.

2. Flask App Setup and Global Variables

```
app = Flask(__name__)

students = courses = enrollments = grades = None
merge = total = average = course_result = CourseDept = StudentDept = toppers =
ranking = None
```

- Initializes the Flask app.
- Declares **global DataFrames** (students, courses, enrollments, grades) to store uploaded CSV data.
- merge, total, average, and course_result hold analytics results computed later.

3. Helper Functions

These functions prevent repeating logic across routes.

```
def compute_totals_if_needed():
    """
    Computes total marks, average marks, and per-course stats
    only if they haven't been computed yet.
    """
```

- Avoids recalculating totals/averages repeatedly.
- Uses pandas.groupby() to compute **student totals**, **averages**, and **course-level stats**.

```
def safe_jsonify_from_df(df):
    """
    Converts a DataFrame to JSON. Returns a safe response
    even if the DataFrame is empty or invalid.
    """
```

- Converts DataFrame objects to a JSON-friendly structure for the frontend.
- Ensures the frontend always gets at least one row of data.

4. Uploading and Preprocessing Data

```
@app.route('/upload', methods=['POST'])
def upload_files():
    """
    Accepts CSV files, cleans data, merges into a single DataFrame,
    and prepares analytics.
    """
```

- **Input:** Four files: students.csv, courses.csv, enrollment.csv, grade.csv.
- **Steps:**
    1. Checks if all required files are uploaded.
    2. Loads them into Pandas DataFrames.
    3. Fills missing values with 0 and removes duplicates.
    4. Renames Department columns to avoid conflicts.
    5. Performs **multiple joins** to merge all files into one merge DataFrame.
    6. Computes totals, averages, and per-course stats.
- **Output:** JSON response confirming data upload success.

5. Ensuring Data Availability

```
def ensure_data_loaded():
    if merge is None:
        return False, jsonify({"status": "error", "message": "Please upload data first."}),
400
    return True, None, None
```

- Prevents analytics routes from running without data.
- Returns an error if users try to access features before uploading files.

6. Analytics Routes Overview

The app offers **RESTful API endpoints** that return JSON responses.

7. Example Route in Detail

Example: Rank Students by Average Marks

```
@app.route('/rank_students', methods=['GET'])
def rank_students():
    ok, resp, code = ensure_data_loaded()
    if not ok:
        return resp, code

    compute_totals_if_needed()
    ranking = average.sort_values(by="AverageMarks",
ascending=False).reset_index(drop=True)
    return jsonify(ranking.to_dict(orient='records'))
```

- Checks if data is uploaded.
- Calls compute_totals_if_needed() to ensure averages are ready.
- Sorts students in descending order of AverageMarks.
- Returns the rankings as JSON.

8. Frontend Integration

```
@app.route('/')
def home():
    return render_template('index.html')
```

- Loads the dashboard built with Bootstrap and Chart.js.
- Users can upload files, view analytics, and download reports.

9. Running the App

```
if __name__ == '__main__':
    app.run(debug=True)
```

- Runs the Flask app locally in debug mode.
- URL: http://127.0.0.1:5000

Key Workflow Summary:

1. User uploads four CSV files.
2. Backend merges, cleans, and computes statistics.
3. Flask routes provide **JSON data** for tables and charts.
4. Frontend displays interactive analytics.

Technical Architecture :

1. Presentation Layer (Frontend)

- Built with **HTML5, CSS (Bootstrap 4)** for styling and responsive UI.
- **JavaScript & Chart.js** for dynamic data visualizations.
- Users can:
  - Upload CSV files (Students, Courses, Enrollments, Grades).
  - View analytics via tables and graphs.
  - Download processed data (Excel/JSON).

2. Application Layer (Backend)

- Powered by **Flask** (Python micro-framework).
- Handles:
  - File upload and validation.
  - Data processing and cleaning with **pandas**.
  - Business logic like ranking, grading, and subject/department analysis.
- JSON responses are returned to the frontend for display.

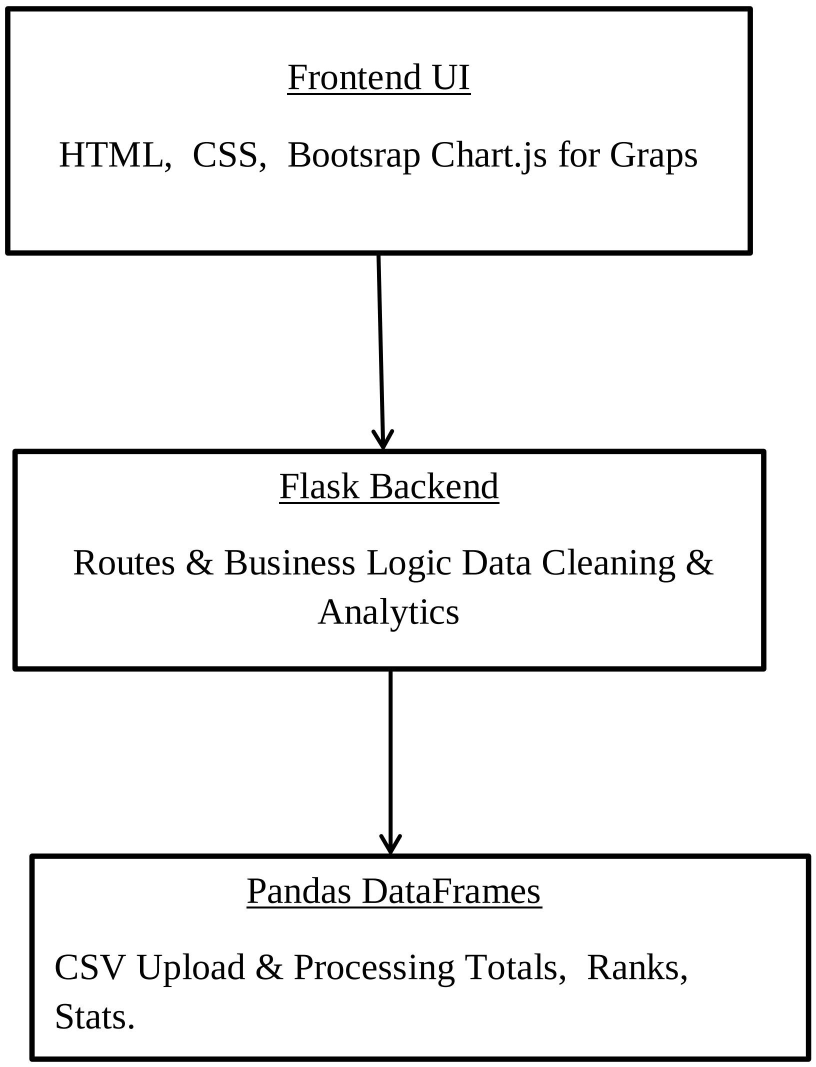
3. Data Layer (Processing & Storage)

- **CSV files** are uploaded dynamically; no database is required.
- **pandas** is used to:
  - Read and merge datasets.
  - Clean null values, remove duplicates.
  - Compute totals, averages, pass/fail counts, and other statistics.

4. Visualization Layer

- **Chart.js** is used for creating interactive charts.
- **Matplotlib** for generating static plots when needed.
- Provides insights like:
  - Top performers
  - Hardest subjects
  - Department-level analytics

Architecture Diagram :


Frontend UI
HTML, CSS, Bootsrap Chart.js for Graps
Flask Backend
Routes & Business Logic Data Cleaning & Analytics
Pandas DataFrames
CSV Upload & Processing Totals, Ranks, Stats.

Setup Instructions :

1. Prerequisites

- **Python 3.8+** (Recommended: Python 3.10)
- **pip** (Python package manager)
- A code editor (VS Code, PyCharm, etc.)
- A modern web browser (Chrome, Edge, Firefox)

2. Project Structure

```
project/
├── app.py               # Main Flask application
├── templates/
│   └── index.html       # Frontend dashboard
├── students.csv       ┐
├── courses.csv        ├  Datasets
├── enrollments.csv    │
├── grades.csv         ┘
├── base_code.py         #No frontend, only one run gives all output
└── menu_driven.py       #Menu driven version of same project
```

3. Clone or Download the Project

```
git clone <repo-link>
cd project
```

4. Install Dependencies

```
pip install flask
pip install pandas
pip install matplotlib
```

5. Run the Application

```
        python app.py (for other two versions : python base_code.py,
python menu_driven.py students.py courses.py enrollments.py grades.py)
```

You will see output like:

```
 * Running on http://127.0.0.1:5000/
 (press ctrl and left button on this link in terminal to navigate to webpage)
```

6. Open the Dashboard

1. Navigate to: `http`://127.0.0.1:5000

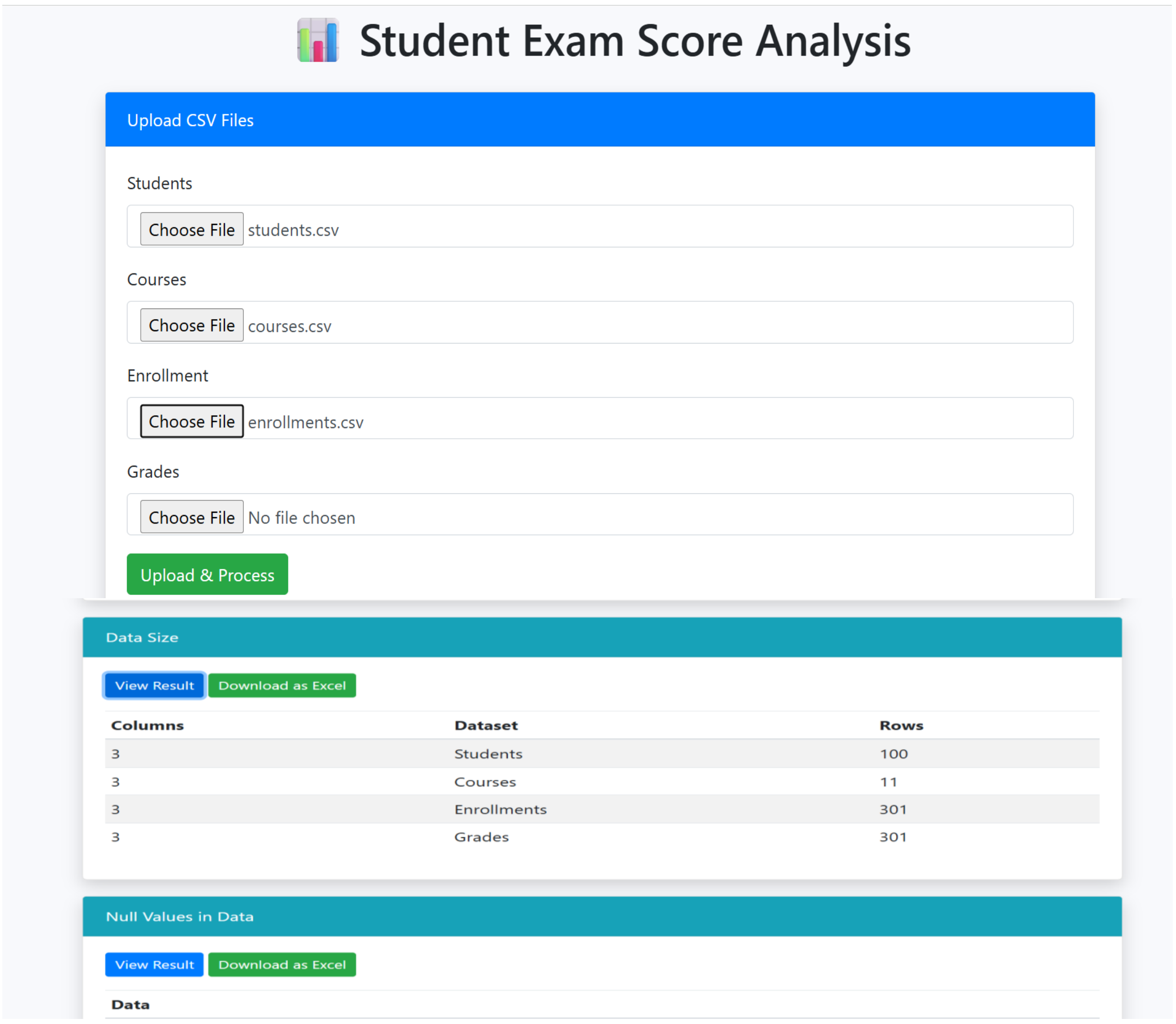
2. Upload your CSV files and start exploring analytics!

8. Sample Data

To test quickly, use the sample CSV files provided in the `project/` folder:

- `students.csv`
- `courses.csv`
- `enrollments.csv`
- `grades.csv`

Screenshots of the Dashboard :



Student Exam Score Analysis

Upload CSV Files

Students: Choose File students.csv

Courses: Choose File courses.csv

Enrollment: Choose File enrollments.csv

Grades: Choose File No file chosen

Upload & Process

Data Size

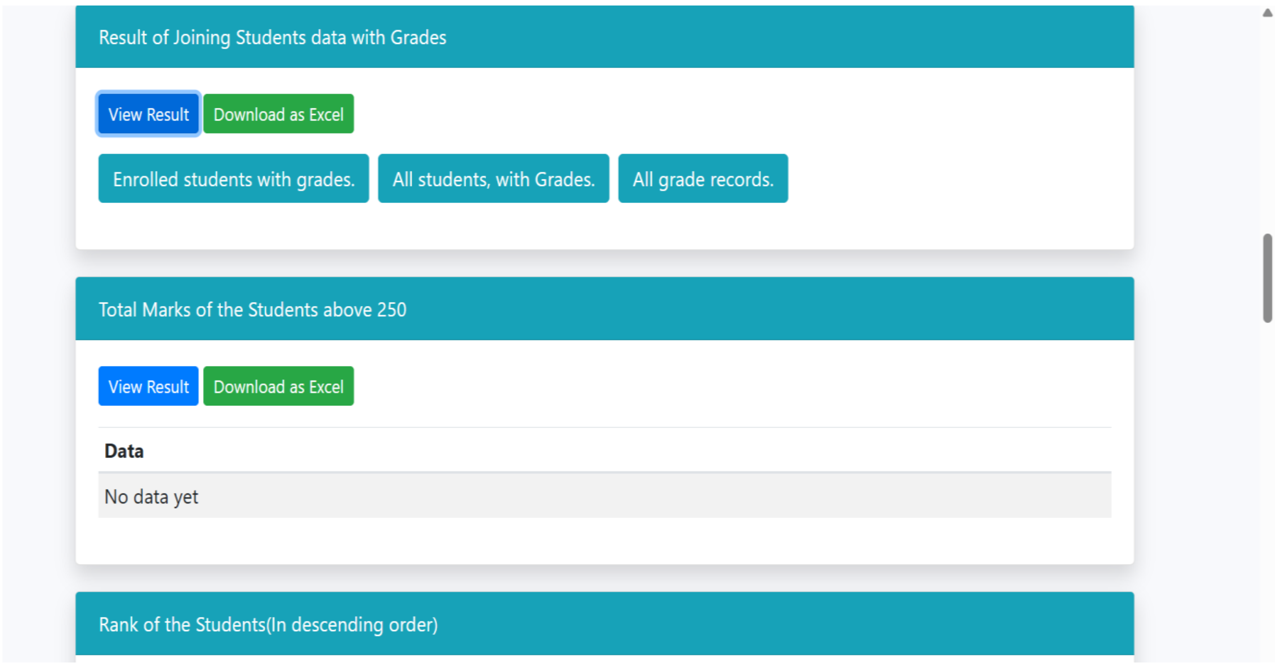
View Result | Download as Excel

Columns	Dataset	Rows
3	Students	100
3	Courses	11
3	Enrollments	301
3	Grades	301

Null Values in Data

View Result | Download as Excel

Data

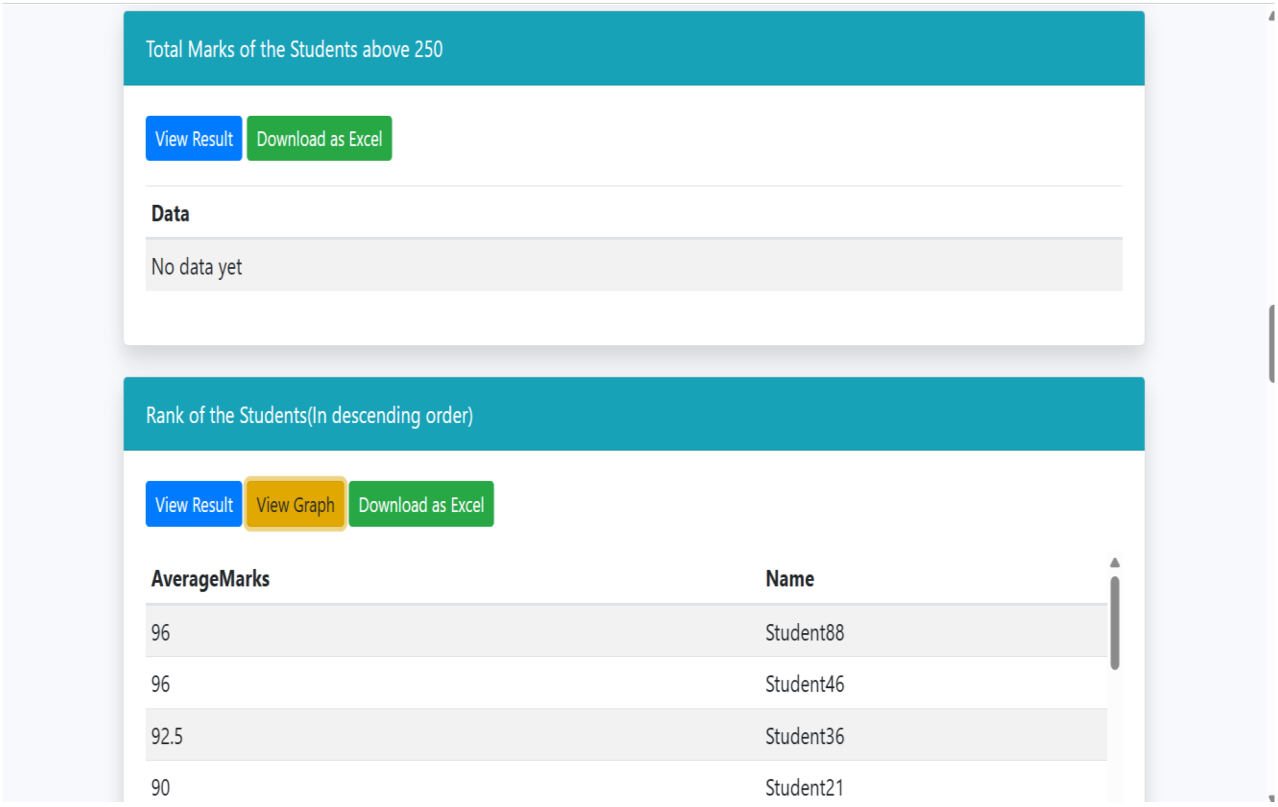


Result of Joining Students data with Grades

Total Marks of the Students above 250

Data
No data yet

Rank of the Students(In descending order)

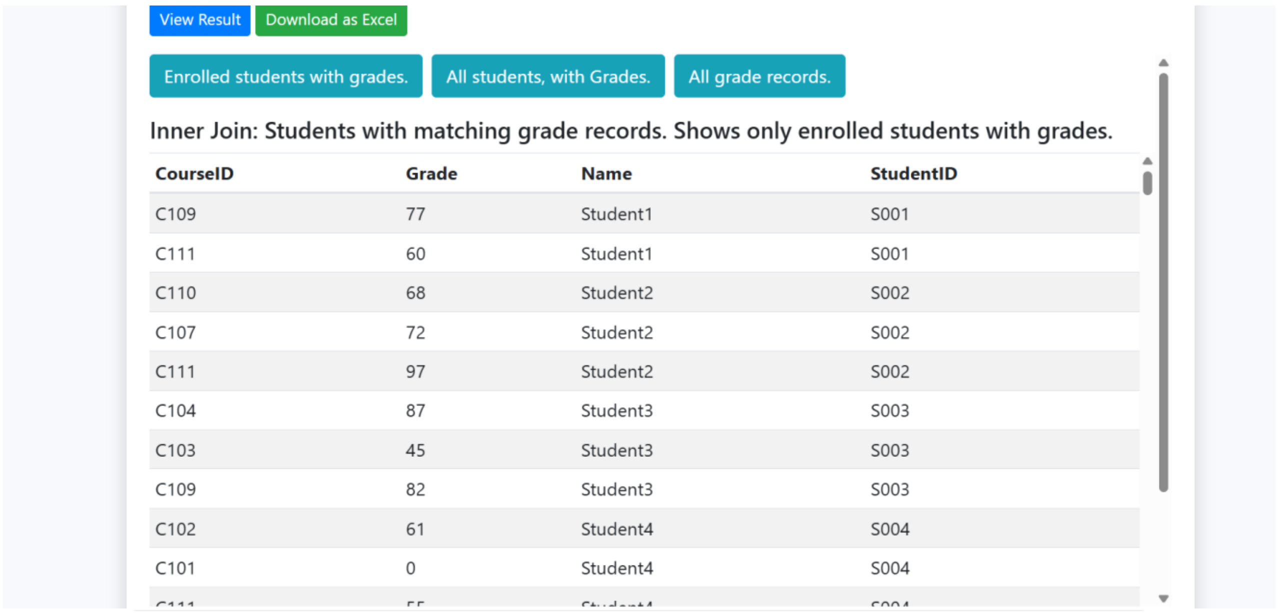


Total Marks of the Students above 250

Data
No data yet

Rank of the Students(In descending order)

AverageMarks	Name
96	Student88
96	Student46
92.5	Student36
90	Student21

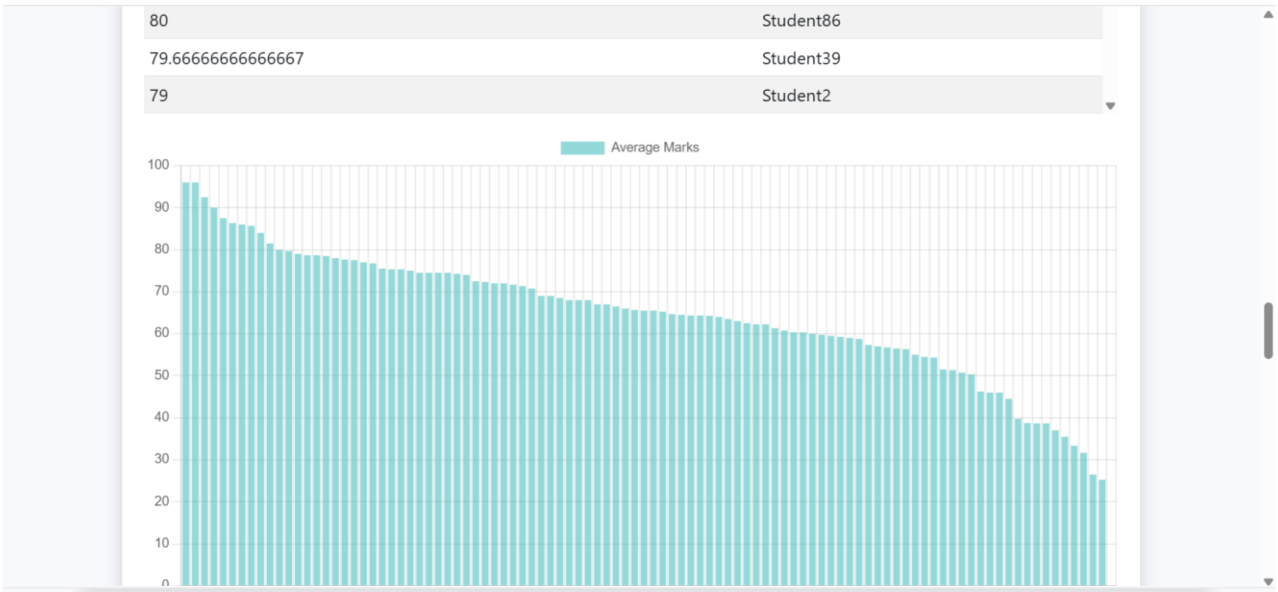


View Result | Download as Excel

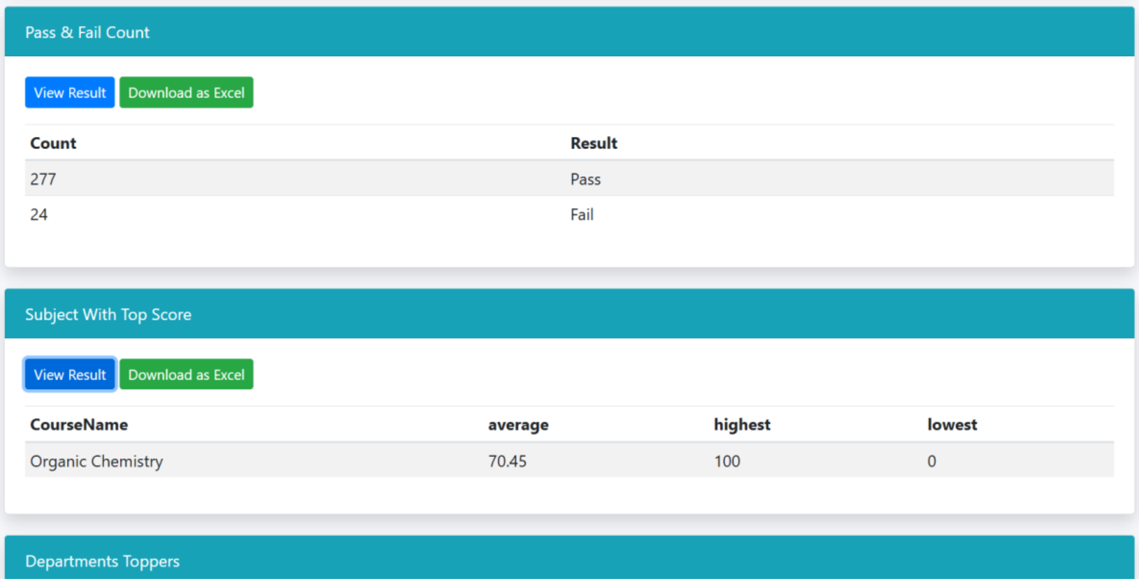
Enrolled students with grades. | All students, with Grades. | All grade records.

Inner Join: Students with matching grade records. Shows only enrolled students with grades.

CourseID	Grade	Name	StudentID
C109	77	Student1	S001
C111	60	Student1	S001
C110	68	Student2	S002
C107	72	Student2	S002
C111	97	Student2	S002
C104	87	Student3	S003
C103	45	Student3	S003
C109	82	Student3	S003
C102	61	Student4	S004
C101	0	Student4	S004



80	Student86
79.66666666666667	Student39
79	Student2



Pass & Fail Count

View Result | Download as Excel

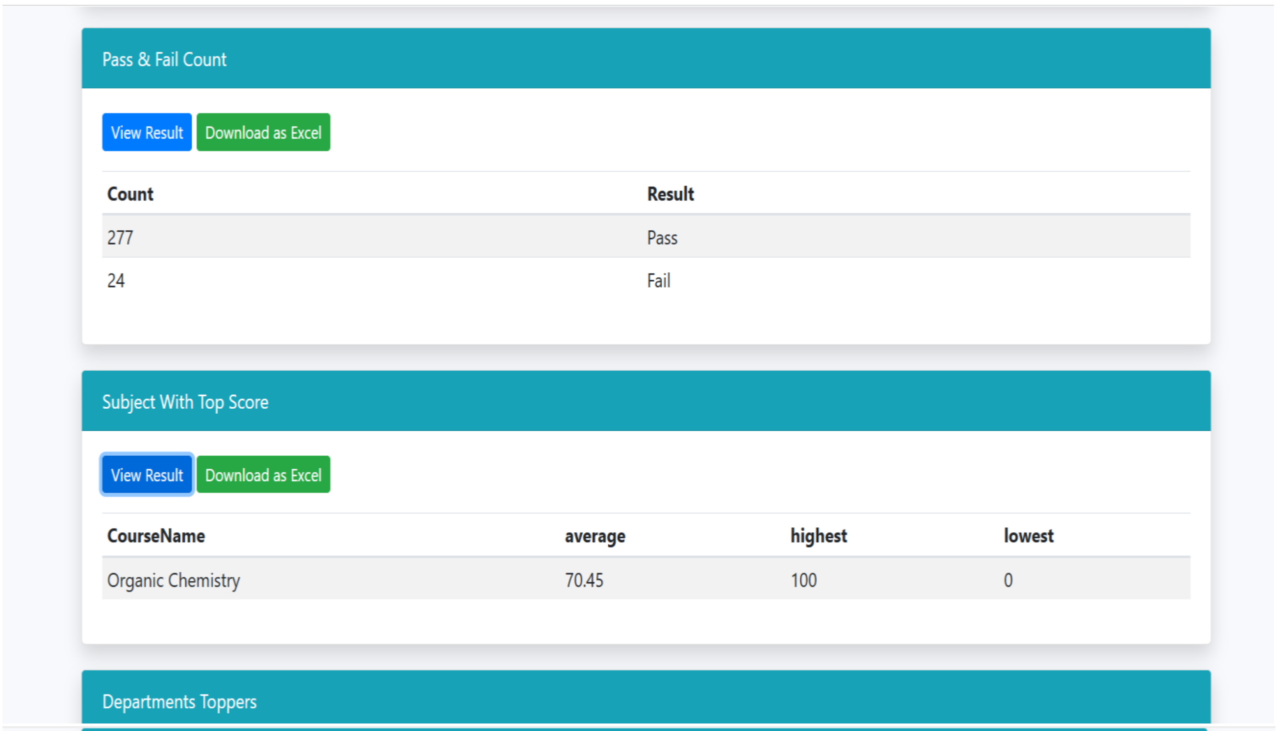
Count	Result
277	Pass
24	Fail

Subject With Top Score

View Result | Download as Excel

CourseName	average	highest	lowest
Organic Chemistry	70.45	100	0

Departments Toppers



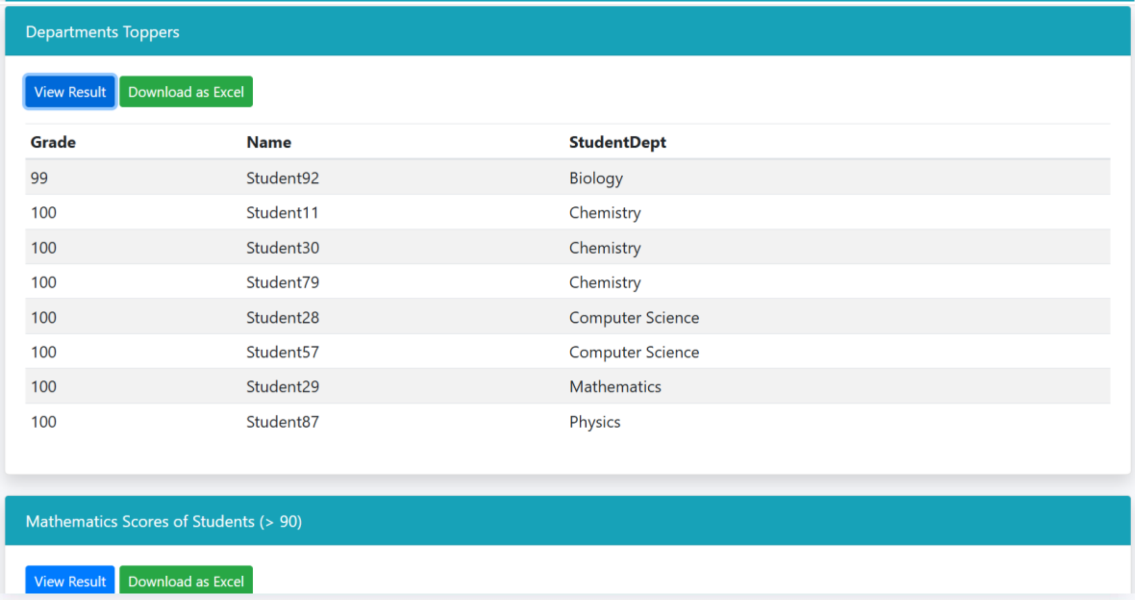
Pass & Fail Count

Count	Result
277	Pass
24	Fail

Subject With Top Score

CourseName	average	highest	lowest
Organic Chemistry	70.45	100	0

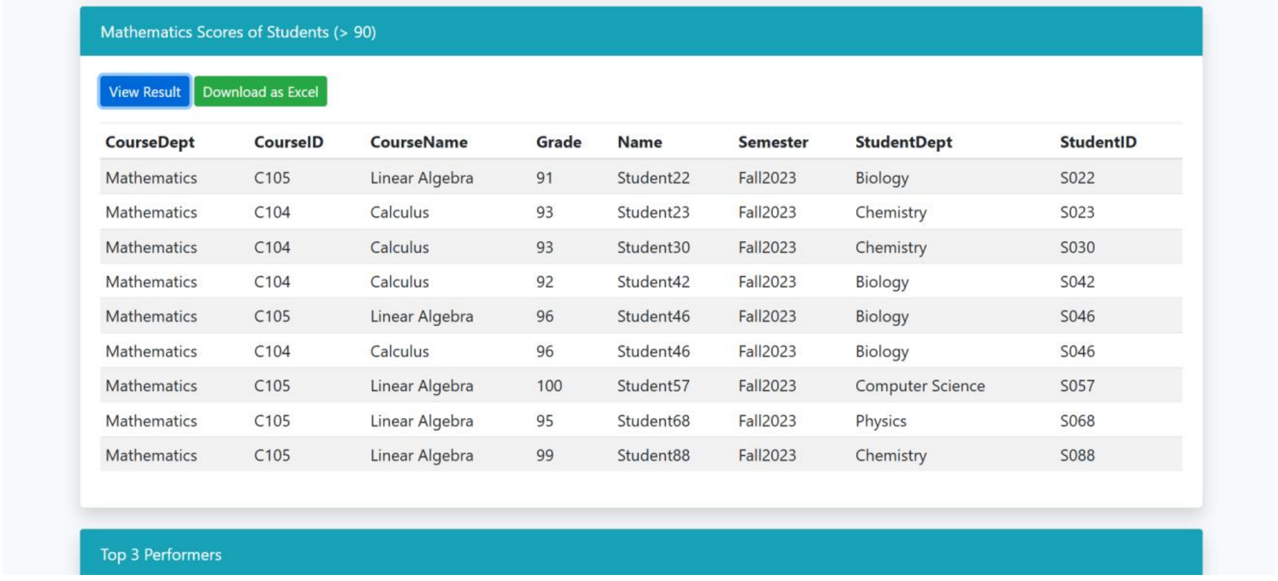
Departments Toppers



Departments Toppers

Grade	Name	StudentDept
99	Student92	Biology
100	Student11	Chemistry
100	Student30	Chemistry
100	Student79	Chemistry
100	Student28	Computer Science
100	Student57	Computer Science
100	Student29	Mathematics
100	Student87	Physics

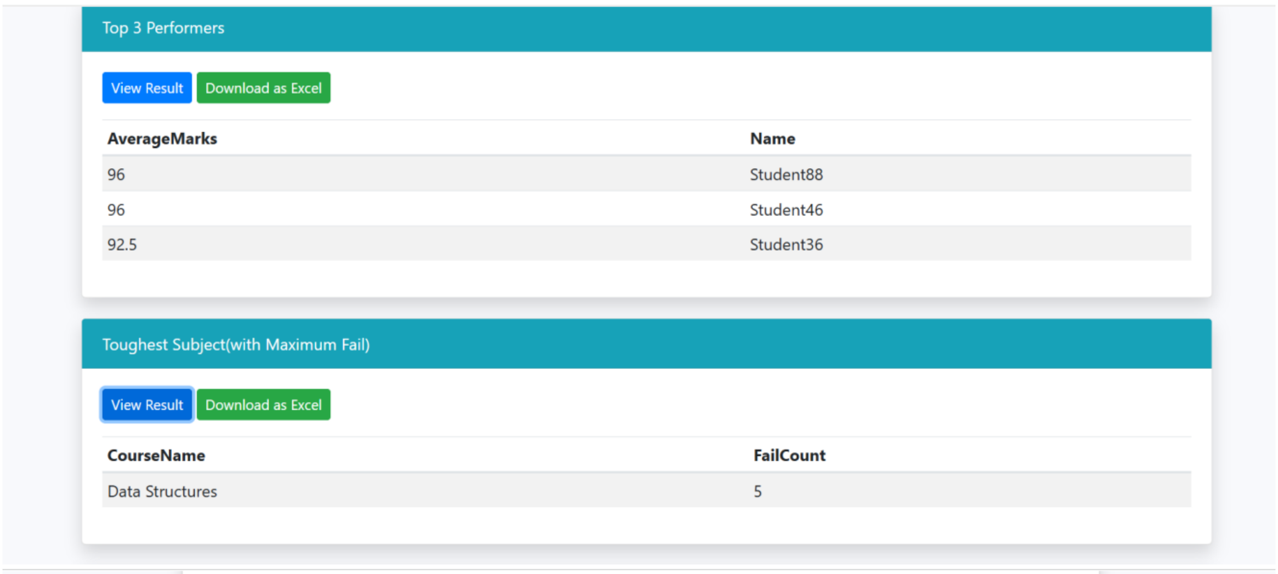
Mathematics Scores of Students (> 90)



Mathematics Scores of Students (> 90)

CourseDept	CourseID	CourseName	Grade	Name	Semester	StudentDept	StudentID
Mathematics	C105	Linear Algebra	91	Student22	Fall2023	Biology	S022
Mathematics	C104	Calculus	93	Student23	Fall2023	Chemistry	S023
Mathematics	C104	Calculus	93	Student30	Fall2023	Chemistry	S030
Mathematics	C104	Calculus	92	Student42	Fall2023	Biology	S042
Mathematics	C105	Linear Algebra	96	Student46	Fall2023	Biology	S046
Mathematics	C104	Calculus	96	Student46	Fall2023	Biology	S046
Mathematics	C105	Linear Algebra	100	Student57	Fall2023	Computer Science	S057
Mathematics	C105	Linear Algebra	95	Student68	Fall2023	Physics	S068
Mathematics	C105	Linear Algebra	99	Student88	Fall2023	Chemistry	S088

Top 3 Performers

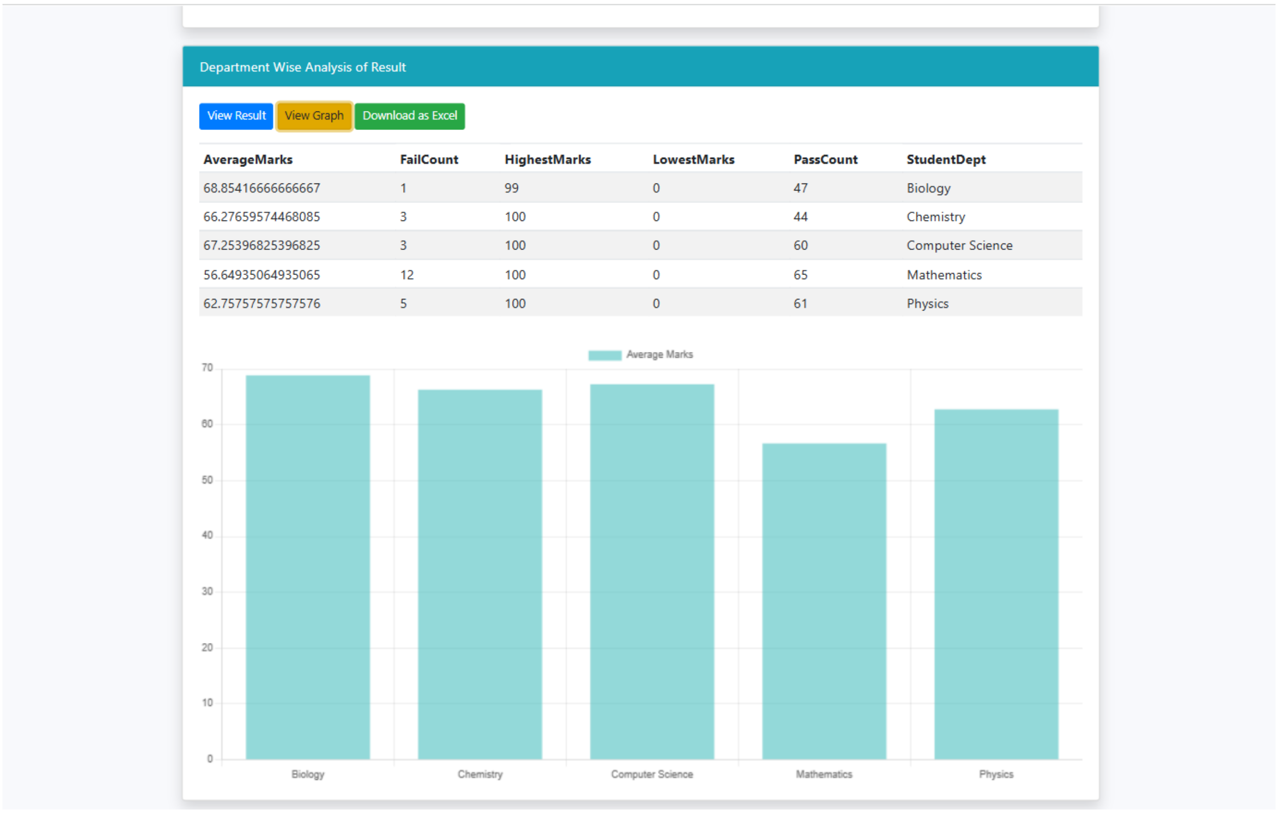


Top 3 Performers

AverageMarks	Name
96	Student88
96	Student46
92.5	Student36

Toughest Subject(with Maximum Fail)

CourseName	FailCount
Data Structures	5



Department Wise Analysis of Result

AverageMarks	FailCount	HighestMarks	LowestMarks	PassCount	StudentDept
68.85416666666667	1	99	0	47	Biology
66.27659574468085	3	100	0	44	Chemistry
67.25396825396825	3	100	0	60	Computer Science
56.64935064935065	12	100	0	65	Mathematics
62.757575757575 76	5	100	0	61	Physics

Closure :

The **Student Analytics Web Application** successfully demonstrates a lightweight, data-driven analytics system built using **Python, Flask, and Pandas**.
It allows users to upload CSV datasets, process them dynamically, and gain insights through interactive visualizations.
This project highlights:

- The importance of **data preprocessing** (cleaning, merging, null handling).
- The ability to perform **real-time analytics** without heavy database dependencies.
- A simple yet scalable architecture that can be extended to include authentication, databases, or advanced reporting features in the future.

This project also emphasizes best practices in:

- **Modular coding** (routes, helpers, analytics separation).
- **RESTful API design** for data services.
- **Visualization** with libraries like Chart.js and Matplotlib.

Bibliography :

1. **Flask Documentation** – https://flask.palletsprojects.com/en/stable/
2. **Pandas Documentation** – https://pandas.pydata.org/docs/
3. **Matplotlib Documentation** – https://www.w3schools.com/python/matplotlib_pyplot.asp
4. **Bootstrap Framework** – https://getbootstrap.com/
5. TutorialsPoint – *Python Flask Tutorial*
6. W3Schools – *HTML, CSS, and JavaScript Basics*
7. GeeksforGeeks – *Data Analysis and Visualization Tutorials*