DATABASE MANAGEMENT SYSTEM MINI PROJECT REPORT

By: Iniyan/ 21172040030049

Project Title: Student Management System with Dashboard Tracking

---

1. Abstract

This project, titled **"Student Management System with Dashboard Tracking,"** aims to create a centralized and efficient database system to manage student academic and administrative data. Its core function is providing students with a **personalized dashboard for progress tracking**. The primary objective is to automate data handling, including student enrollment, attendance, assessment results, and personal information. It uses **MySQL** for robust data storage and SQL queries for efficient record retrieval and manipulation. The key outcomes are an organized, consistent, and secure relational database, enabling **real-time performance tracking for students** and generating essential administrative and academic reports. The system ensures easy access to all relevant information, significantly reducing administrative burden and enhancing communication.

---

2. Objectives

- To design and implement a database for a Student Management System that centralizes student academic and administrative records.
- To maintain **data consistency, integrity, and accuracy** across all student records using Primary and Foreign Keys.
- To create efficient **SQL queries for CRUD** (Create, Read, Update, Delete) operations.
- To develop a student-facing dashboard for **real-time tracking** of attendance, grades, and schedules.
- To generate useful reports for administrators (e.g., class-wise attendance, performance summaries) and students (e.g., transcripts).

---

3. System Analysis

a. Problem Definition

In a traditional or manual academic environment, student data (personal info, grades, attendance, schedule) is often stored across disparate, non-integrated sources like physical files, multiple spreadsheets, or basic departmental records. This fragmented approach leads to **data redundancy, inconsistency, and administrative bottlenecks**, making it difficult for students to get a clear, consolidated view of their progress and for staff to generate timely, accurate reports.

b. Existing System

The current/manual system involves:

- **Paper-based or spreadsheet recording** of attendance and grades, which is time-consuming and error-prone.
- **Decentralized data storage**, leading to difficulty in cross-referencing information and ensuring data integrity.
- Students **lack real-time access** to their performance metrics, requiring them to repeatedly contact administrative staff for updates.
- Report generation is slow, limiting the ability of staff to identify struggling students promptly.

c. Proposed System

The proposed Student Management System provides a **centralized, digital solution** that significantly improves upon the existing system.

- **Automation:** Automates routine tasks like attendance tracking, grade calculation, and report generation, saving staff time and minimizing errors.
- **Reliability & Consistency:** All data is stored in a single, secure database, ensuring high **data integrity** and consistency.
- **Speed & Accessibility:** Provides a customizable dashboard where students can quickly view their attendance percentage, recent grades, upcoming classes, and assignments **in real-time**.
- **Decision-Making:** Enables administrators to access real-time data and analytics for timely interventions and informed decisions.

---

4. System Design

a. Entity Relationship (ER) Diagram

The ER diagram illustrates the major entities: **STUDENT, COURSE, ATTENDANCE, GRADE, and FACULTY**. It shows the relationships between them, such as a one-to-many relationship between **STUDENT** and **GRADE** (one student has many grades), and many-to-many relationships through linking tables (like an ENROLLMENT table, if explicitly modelled). Entities include attributes like `Student_ID` (**Primary Key**), `Name`, `Course_ID`, and `Marks`.

b. Schema Diagram

The Schema Diagram details the logical structure of the database, showing all tables with their columns, Primary Keys (PK), and Foreign Keys (FK).

Table Name	Key	Field Name	Data Type	Relationship
STUDENT	PK	student_id	INT	Unique ID for student
		name	VARCHAR(50)	Student's full name
		email	VARCHAR(100)	Student's unique email address
COURSE	PK	course_id	VARCHAR(10)	Unique ID for a course
		course_name	VARCHAR(100)	Full name of the course
GRADE	PK	grade_id	INT	Unique ID for a grade entry
	FK	student_id	INT	References **STUDENT** table
	FK	course_id	VARCHAR(10)	References **COURSE** table

		marks	INT	Marks obtained
ATTENDANCE	PK	atten_id	INT	Unique ID for record
	FK	student_id	INT	References **STUDENT** table
		date	DATE	Date of the session

c. Data Dictionary

Table Name	**Field Name**	**Data Type**	**Size**	**Description**
STUDENT	**student_id**	INT	10	Unique ID for student (**PK**)
STUDENT	name	VARCHAR	50	Student's full name
STUDENT	email	VARCHAR	100	Student's unique email address
COURSE	**course_id**	VARCHAR	10	Unique ID for a course (**PK**)
COURSE	course_name	VARCHAR	100	Full name of the

				course
GRADE	**grade_id**	INT	10	Unique ID for a grade entry (**PK**)
GRADE	student_id	INT	10	**FK** to STUDENT table
GRADE	course_id	VARCHAR	10	**FK** to COURSE table
GRADE	marks	INT	3	Marks obtained in the assessment
ATTENDANCE	**atten_id**	INT	10	Unique ID for attendance record (**PK**)
ATTENDANCE	student_id	INT	10	**FK** to STUDENT table
ATTENDANCE	date	DATE	-	Date of the class/session

---

5. Implementation

a. Software & Hardware Requirements

Category	**Requirement**

Software	**MySQL** (Database Server), SQLyog or MySQL Workbench (Database Management Tool)
Hardware	Minimum 4GB RAM, 1.6GHz processor

b. SQL Queries Used

Table Creation (DDL)

```sql
CREATE TABLE STUDENT (
    student_id INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE
);

CREATE TABLE COURSE (
    course_id VARCHAR(10) PRIMARY KEY,
    course_name VARCHAR(100) NOT NULL
);

CREATE TABLE GRADE (
    grade_id INT PRIMARY KEY,
    student_id INT,
    course_id VARCHAR(10),
    marks INT,
    FOREIGN KEY (student_id) REFERENCES STUDENT(student_id),
    FOREIGN KEY (course_id) REFERENCES COURSE(course_id)
);
```

Data Insertion (DML)

```sql
INSERT INTO STUDENT (student_id, name, email) VALUES (101, 'Alice Johnson', 'alice@uni.edu');
INSERT INTO COURSE (course_id, course_name) VALUES ('CS101', 'Intro to Databases');
INSERT INTO GRADE (grade_id, student_id, course_id, marks) VALUES (1, 101, 'CS101', 85);
```

Joins (for Dashboard View)

This query is essential for the dashboard as it combines student identity, course details, and grades.

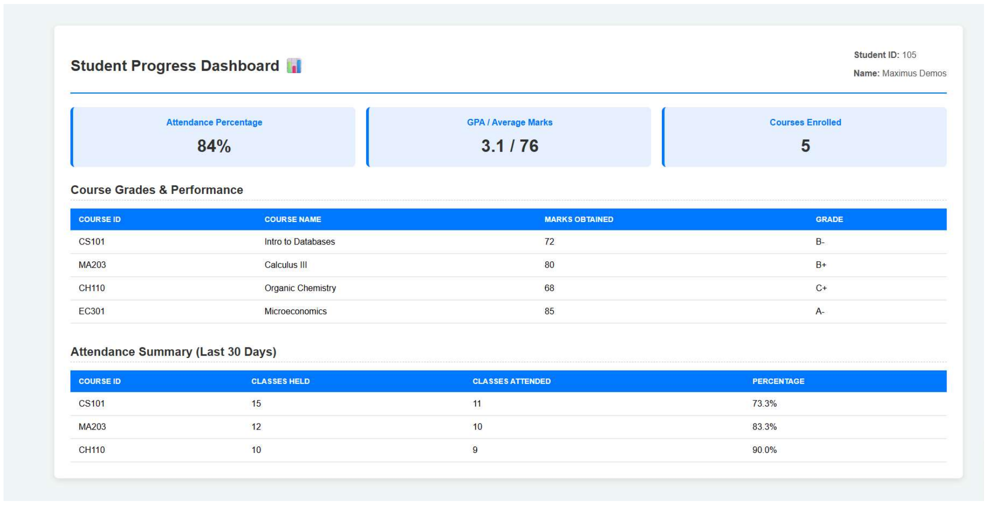
```sql
SELECT
  S.name,
  C.course_name,
  G.marks
FROM
  STUDENT S
JOIN
  GRADE G ON S.student_id = G.student_id
JOIN
  COURSE C ON G.course_id = C.course_id
WHERE
  S.student_id = 101;
```

View (to simplify student dashboard data access)

Views simplify complex reporting/dashboard queries by storing the join operation.

```sql
CREATE VIEW Student_Performance_View AS
SELECT
  S.student_id,
  S.name,
  C.course_name,
  G.marks
FROM
  STUDENT S
JOIN
  GRADE G ON S.student_id = G.student_id
JOIN
  COURSE C ON G.course_id = C.course_id;
```

6. Sample Outputs (Front End)



Student Progress Dashboard 📊

Student ID: 105
Name: Maximus Demos

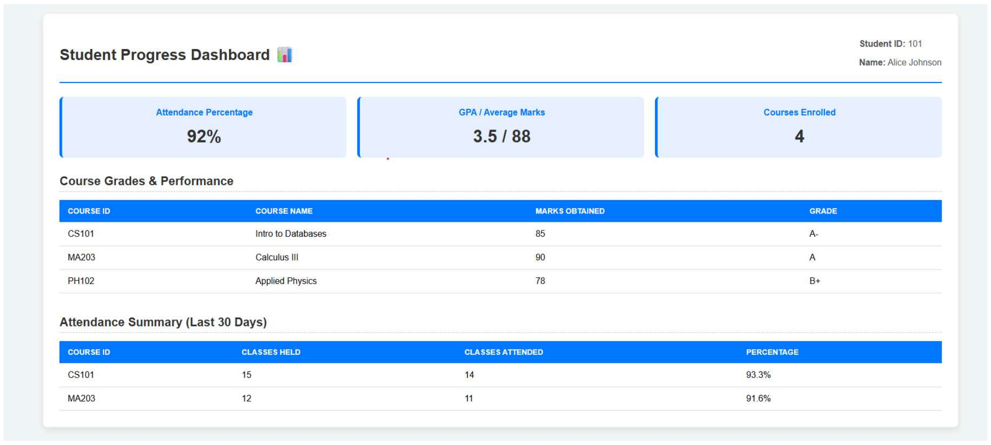
Attendance Percentage	GPA / Average Marks	Courses Enrolled
84%	3.1 / 76	5

Course Grades & Performance

COURSE ID	COURSE NAME	MARKS OBTAINED	GRADE
CS101	Intro to Databases	72	B-
MA203	Calculus III	80	B+
CH110	Organic Chemistry	68	C+
EC301	Microeconomics	85	A-

Attendance Summary (Last 30 Days)

COURSE ID	CLASSES HELD	CLASSES ATTENDED	PERCENTAGE
CS101	15	11	73.3%
MA203	12	10	83.3%
CH110	10	9	90.0%



Student Progress Dashboard 📊

Student ID: 101
Name: Alice Johnson

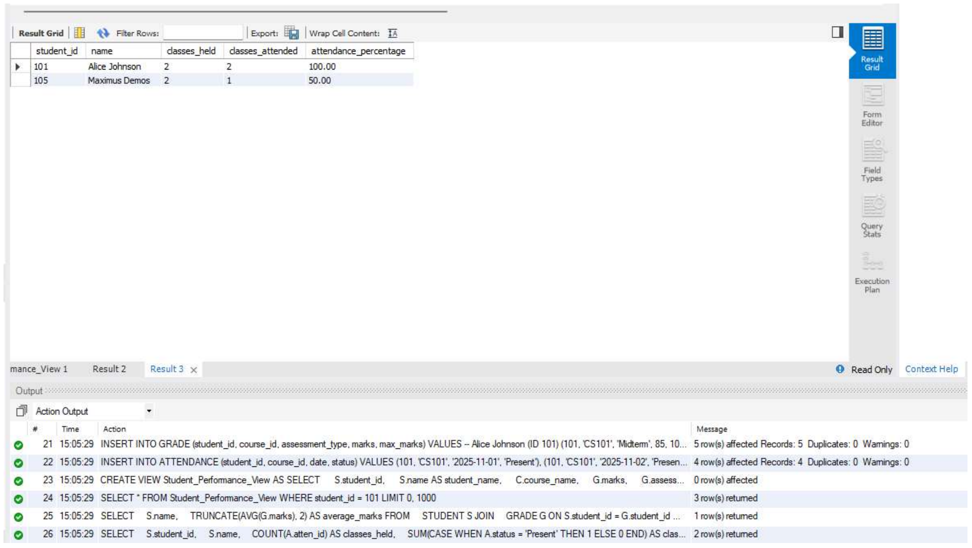
Attendance Percentage	GPA / Average Marks	Courses Enrolled
92%	3.5 / 88	4

Course Grades & Performance

COURSE ID	COURSE NAME	MARKS OBTAINED	GRADE
CS101	Intro to Databases	85	A-
MA203	Calculus III	90	A
PH102	Applied Physics	78	B+

Attendance Summary (Last 30 Days)

COURSE ID	CLASSES HELD	CLASSES ATTENDED	PERCENTAGE
CS101	15	14	93.3%
MA203	12	11	91.6%



student_id	name	classes_held	classes_attended	attendance_percentage
101	Alice Johnson	2	2	100.00
105	Maximus Demos	2	1	50.00

```
#   Time      Action                                                                                                                        Message
21  15:05:29  INSERT INTO GRADE (student_id, course_id, assessment_type, marks, max_marks) VALUES -- Alice Johnson (ID 101) (101, 'CS101', 'Midterm', 85, 10...   5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0
22  15:05:29  INSERT INTO ATTENDANCE (student_id, course_id, date, status) VALUES (101, 'CS101', '2025-11-01', 'Present'), (101, 'CS101', '2025-11-02', 'Presen...   4 row(s) affected Records: 4  Duplicates: 0  Warnings: 0
23  15:05:29  CREATE VIEW Student_Performance_View AS SELECT    S.student_id,    S.name AS student_name,    C.course_name,    G.marks,    G.assess...   0 row(s) affected
24  15:05:29  SELECT * FROM Student_Performance_View WHERE student_id = 101 LIMIT 0, 1000                                                            3 row(s) returned
25  15:05:29  SELECT    S.name,    TRUNCATE(AVG(G.marks), 2) AS average_marks FROM    STUDENT S JOIN    GRADE G ON S.student_id = G.student_id ...        1 row(s) returned
26  15:05:29  SELECT    S.student_id,    S.name,    COUNT(A.atten_id) AS classes_held,    SUM(CASE WHEN A.status = 'Present' THEN 1 ELSE 0 END) AS clas...   2 row(s) returned
```

7. Results and Discussion

The Student Management System successfully meets its primary objective by providing a **centralized and relational database** for all student information. The use of **Primary and Foreign Keys** ensures high data integrity and consistency, preventing orphaned records. The implementation of the **JOIN operation** and the `Student_Performance_View` query demonstrates the system's ability to efficiently retrieve consolidated data, which is the core requirement for the student dashboard. The system significantly simplifies the process of generating complex reports, which previously required manual data consolidation.

Possible Improvements:

- **Triggers:** Implementing a real-time notification system (via triggers) to alert students about low attendance or new grade postings.
- **Normalization:** Further normalization to include dedicated tables for Assessment Types and Faculty-Course assignment.
- **Leave Module:** Integrating a more comprehensive Leave Management module that interacts with the attendance table.

8. Conclusion

The Student Management System project has achieved its goal of designing and implementing a **robust, centralized, and efficient database** to manage core student academic records. The key achievement is the ability to support the creation of a dynamic, comprehensive **student dashboard** through well-structured relational tables and efficient SQL views. This system provides clear, real-time data access for students and streamlines administrative reporting, moving beyond the limitations of manual or spreadsheet-based systems. In the future, the project can be extended to include faculty management, a financial/fee collection module, and the integration of a full-fledged web-based user interface.

9. References

- "Database System Concepts" by Korth & Silberschatz
- The MySQL documentation: https://www.mysql.com
- Online Tutorials on Student Management Systems (e.g., GeeksforGeeks, YouTube resources)