**National Institute of Technology Durgapur**

**Department of Computer Science and Engineering**



**DATABASE MANAGEMENT SYSTEM LABORATORY**

**MINI PROJECT :**

**‘’A College Attendance Management System.’’**

**SESSION: 2024-25**

**SEMESTER: 4**

**SUBJECT CODE: CSS453**

**GROUP: 17**

**GROUP MEMBERS**

|  |  |
| --- | --- |
| **Roll Number** | **Name** |
| **23CS8082** | **AHINA GANGOPADHYAY** |
| **23CS8083** | **MAKALA SAI TIJIL** |
| **23CS8084** | **PUDI SRAVAN** |

**Acknowledgement**

We would like to express our gratitude to all those who supported and guided us throughout the completion of this project.

First and foremost, we thank our faculty and mentors for their invaluable insights and encouragement. Their guidance helped us understand the concepts better and apply them practically.

We are also thankful to our teammates for their cooperation, effort, and teamwork, which played a key role in bringing this project to life.

Lastly, we acknowledge the tools and resources—especially MySQL and related documentation—that enabled us to build, test, and refine our virtual database system effectively.

This project has been a great learning experience, and we are grateful for the opportunity.

**Contents:**

* **Introduction to the Project**
* **Key Features**
* **Operations Implemented**
* **Technology Used**
* **Future Improvements**
* **Github Repository Link**
* **Implementation of the Project**
* **Advantages of the System**
* **Creating Tables**
* **Inserting Data**
* **Queries and their results**
* **Conclusion**

**Introduction**

**Problem Statement**

Create a database for a college attendance management system.

**Description**

The database is designed to track student attendance records for various subjects. It helps maintain and retrieve attendance details efficiently for students, courses, and instructors.

**Database Design**

**Tables**

1. **Students**
   * StudentID (Primary Key)
   * Name
   * RollNumber
   * Course
2. **Subjects**
   * SubjectID (Primary Key)
   * SubjectName
   * TeacherID
3. **Attendance**
   * AttendanceID (Primary Key)
   * StudentID (Foreign Key → Students)
   * SubjectID (Foreign Key → Subjects)
   * Date
   * Status (Present/Absent)

**Key Features**

* Supports real-time insertion and tracking of attendance.
* Ensures accurate calculation of attendance percentages.
* Highlights defaulters and attendance trends for management use.
* Modular queries and procedures for flexible reporting.
* Data integrity maintained using **foreign key constraints**.

**Assumptions taken**:

A student is considered as present on a day if he/she is present in all classes of that day

**Operations Implemented**

* Fetch attendance records of a specific student.
* Count total present and absent days for a student.
* Retrieve all students who have less than 75% attendance.
* Insert a new attendance record into the system.
* Generate a monthly attendance report (subject-wise).
* Identify students with the highest attendance.
* Retrieve attendance percentage by subject (student-wise and overall).
* List students with more than 3 consecutive absences.
* Fetch subjects with the lowest attendance rate.
* Calculate the average attendance percentage of each course.

**Objective:**

The main goal of this project is to design a digital attendance management system that efficiently tracks student attendance subject-wise and date-wise. This system replaces traditional methods, minimizes manual errors, and helps generate insightful reports like monthly attendance, absentee records, and subject-wise statistics.

**Technology Used:**

We used MySQL to build this project because it's simple to set up, reliable, and handles structured data really well. It made it easy for us to store and manage student and attendance records efficiently. MySQL also supports useful features like triggers, constraints, and complex queries, which helped us automate tasks like calculating attendance percentages and ensuring data accuracy. Overall, it was a good choice for building a system that needs to be both organized and dependable.

**Future Improvements:**

The project successfully delivers a basic, efficient, and scalable attendance tracking system using SQL. It can be extended further to include a frontend interface, user login, notifications, and more advanced analytics for practical use in colleges.

**Why we used MYSQL Command Line Client to create our database?**

For this project, we used the MySQL Command Line Client to set up a database. This method was ideal because:

* It provided a controlled and secure space to test all our SQL queries without affecting any real data.
* We had the flexibility to reset and rebuild the database anytime during development.
* It ensured consistency across our team, as everyone worked on the same structured setup.
* It made the system easy to showcase during evaluations.

Using the MySQL Command Line Client allowed us to run queries directly and get instant feedback, which sped up both development and debugging.

**Github Repository Link for our Project:**

**https://github.com/ahinagangopadhyay/College\_Attendance\_Management\_System**

**Implementation of the Project**

The core of our project revolves around implementing a reliable and efficient student attendance management system using MySQL as the database platform. We began by designing a well-structured schema that includes tables for Students, Subjects, and Attendance. Each table was carefully designed to maintain data integrity, with appropriate primary and foreign key relationships ensuring consistency across the database.

We populated the database with sample data to simulate real-world scenarios. The Attendance table stores individual records for each student per subject, date, and status (Present/Absent). These entries formed the foundation of our reporting logic.

Our implementation relied heavily on SQL queries to extract meaningful insights. Some key features we implemented:

**Monthly and Subject-Wise Reports**: Queries were written to group attendance data by student, subject, and month, calculating attendance percentages accurately.

**Daily Attendance Summary**: We created logic to determine whether a student was fully present on a day by checking if they were marked present in all subjects on that date.

**Consecutive Absence Detection**: This part required us to think beyond simple aggregates. We used date-based logic and row comparisons to detect students who were absent over consecutive class days or across any three successive classes.

**Procedures**: To simplify complex operations, we explored the use of procedures for generating specific reports.

The entire system runs virtually on any machine with MySQL installed, making it accessible and portable. Although this setup is basic, it reflects a strong foundation in database operations and query optimization.

**Advantages of the System:**

1. Centralized Data Storage  
   All attendance records are stored in a structured, organized, and centralized manner, making it easy to retrieve and analyze data.
2. Custom Reporting  
   The system allows for flexible and detailed reports, including subject-wise, student-wise, and day-wise attendance summaries.
3. Accuracy & Consistency  
   With SQL queries and constraints, data accuracy is maintained, and inconsistencies are minimized compared to manual tracking.
4. Automated Calculations  
   Attendance percentages, monthly reports, and identification of students with low or high attendance are automatically calculated—saving time and effort.
5. Easy Maintenance & Scalability  
   The database structure can be easily updated or scaled to accommodate more students, subjects, or features.
6. Testing & Debugging Friendly  
   Running this in a virtual environment using the MySQL CLI ensured safe testing, easy resets, and zero impact on real-world systems.
7. Team Collaboration  
   Since the schema is standardized, all team members could contribute queries, procedures, and data without confusion.
8. Future Integration Possibilities  
   This system can serve as a backend for larger applications like web dashboards, mobile apps, or college portals.

**Creating the tables of the database:**

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

Name VARCHAR(50),

RollNumber VARCHAR(10),

Course VARCHAR(10)

);

CREATE TABLE Subjects (

SubjectID INT PRIMARY KEY,

SubjectName VARCHAR(50),

TeacherID INT

);

CREATE TABLE Attendance (

AttendanceID INT PRIMARY KEY,

StudentID INT,

SubjectID INT,

Date DATE,

Status VARCHAR(10),

FOREIGN KEY (StudentID) REFERENCES Students(StudentID),

FOREIGN KEY (SubjectID) REFERENCES Subjects(SubjectID)

);

**Inserting the data into the tables:**

INSERT INTO Students (StudentID, Name, RollNumber, Course) VALUES

(1, 'Amisha Mehta', '81', 'CSE'),

(2, 'Tanya Nair', '82', 'CSE'),

(3, 'Adrika Das', '83', 'ECE'),

(4, 'Ishita Sen', '84', 'ECE'),

(5, 'Amit Roy', '85', 'CSE');

INSERT INTO Subjects (SubjectID, SubjectName, TeacherID) VALUES

(101, 'DBMS', 1001),

(102, 'Operating Systems', 1002),

(201, 'Signals and Systems', 1003),

(202, 'Digital Circuits', 1004);

INSERT INTO Attendance (AttendanceID, StudentID, SubjectID, Date, Status) VALUES

(1, 1, 101, '2025-02-05', 'Present'),

(2, 1, 102, '2025-02-05', 'Present'),

(3, 1, 101, '2025-02-15', 'Present'),

(4, 1, 102, '2025-02-20', 'Present'),

(5, 1, 101, '2025-03-01', 'Absent'),

(6, 1, 102, '2025-03-05', 'Present'),

(7, 1, 101, '2025-03-15', 'Present'),

(8, 1, 102, '2025-03-25', 'Present'),

(9, 1, 101, '2025-03-25', 'Present'),

(10, 1, 101, '2025-03-20', 'Present'),

(11, 1, 102, '2025-03-20', 'Present'),

(12, 2, 101, '2025-02-05', 'Present'),

(13, 2, 102, '2025-02-05', 'Absent'),

(14, 2, 101, '2025-02-15', 'Present'),

(15, 2, 102, '2025-02-20', 'Absent'),

(16, 2, 101, '2025-03-01', 'Present'),

(17, 2, 102, '2025-03-05', 'Present'),

(18, 2, 101, '2025-03-15', 'Absent'),

(19, 2, 102, '2025-03-25', 'Present'),

(20, 2, 101, '2025-03-25', 'Present'),

(21, 2, 101, '2025-03-20', 'Present'),

(22, 2, 102, '2025-03-20', 'Present'),

(23, 5, 101, '2025-02-05', 'Present'),

(24, 5, 102, '2025-02-05', 'Present'),

(25, 5, 101, '2025-02-15', 'Present'),

(26, 5, 102, '2025-02-20', 'Absent'),

(27, 5, 101, '2025-03-01', 'Present'),

(28, 5, 102, '2025-03-05', 'Present'),

(29, 5, 101, '2025-03-15', 'Absent'),

(30, 5, 102, '2025-03-25', 'Present'),

(31, 5, 101, '2025-03-25', 'Present'),

(32, 5, 101, '2025-03-20', 'Present'),

(33, 5, 102, '2025-03-20', 'Present'),

(34, 3, 201, '2025-02-06', 'Present'),

(35, 3, 202, '2025-02-11', 'Present'),

(36, 3, 201, '2025-02-21', 'Present'),

(37, 3, 202, '2025-03-02', 'Absent'),

(38, 3, 201, '2025-03-12', 'Present'),

(39, 3, 202, '2025-03-18', 'Present'),

(40, 3, 201, '2025-03-25', 'Present'),

(41, 3, 202, '2025-03-25', 'Present'),

(42, 3, 201, '2025-03-20', 'Present'),

(43, 3, 202, '2025-03-20', 'Present'),

(44, 4, 201, '2025-02-06', 'Present'),

(45, 4, 202, '2025-02-11', 'Absent'),

(46, 4, 201, '2025-02-21', 'Absent'),

(47, 4, 202, '2025-03-02', 'Present'),

(48, 4, 201, '2025-03-12', 'Present'),

(49, 4, 202, '2025-03-18', 'Absent'),

(50, 4, 201, '2025-03-25', 'Present'),

(51, 4, 202, '2025-03-25', 'Present'),

(52, 4, 201, '2025-03-20', 'Present'),

(53, 4, 202, '2025-03-20', 'Present'),

(54, 2, 101, '2025-03-27', 'Absent'),

(55, 2, 101, '2025-03-28', 'Absent'),

(56, 2, 101, '2025-03-29', 'Absent'),

(57, 2, 101, '2025-03-30', 'Absent'),

(58, 4, 202, '2025-03-26', 'Absent'),

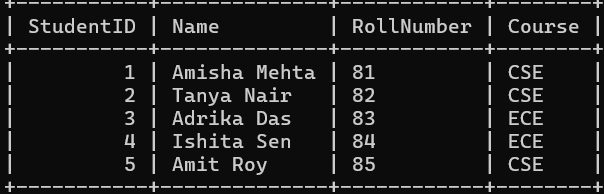
(59, 4, 202, '2025-03-27', 'Absent'),

(60, 4, 202, '2025-03-28', 'Absent'),

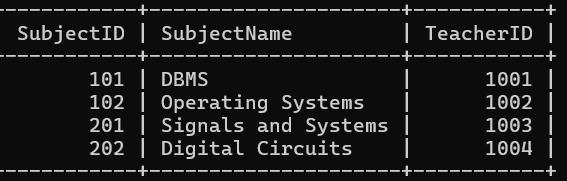
(61, 4, 202, '2025-03-29', 'Absent');

**Tables created:**

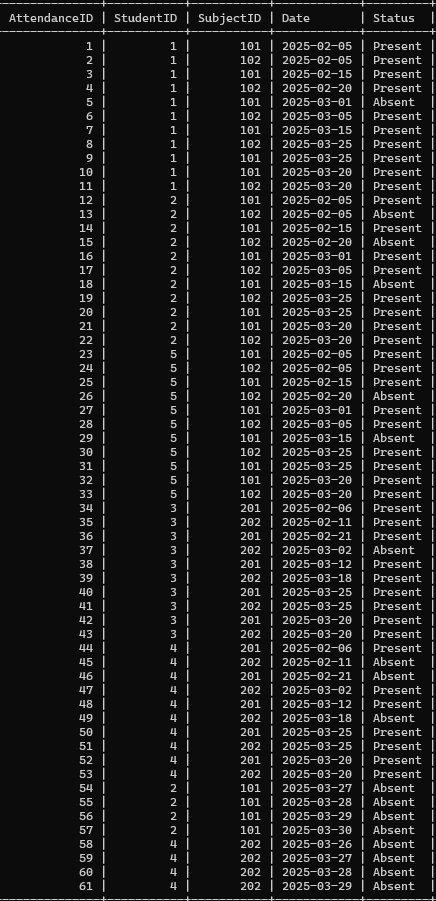
**Students Table**



**Subjects table:**



**Attendance Table:**



**Queries executed and operations performed:**

**Q1) Fetch attendance records of a specific student**

(A specific StudentID was given by us to execute the query and fetch attendance records)

SELECT a.StudentID, s.Name AS StudentName, a.SubjectID, subj.SubjectName, a.Date, a.Status FROM Attendance a

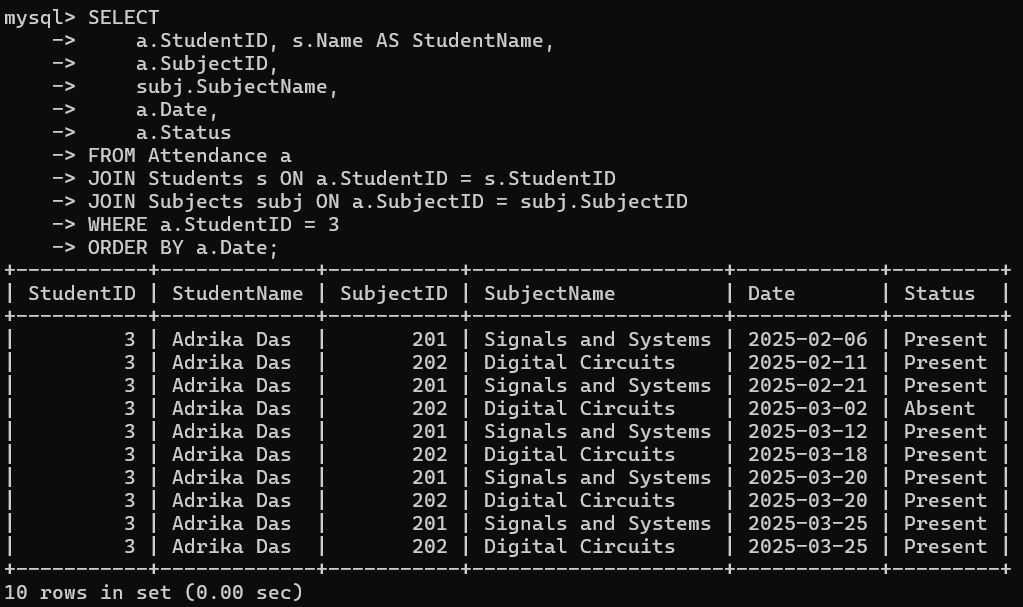
JOIN Students s ON a.StudentID = s.StudentID

JOIN Subjects subj ON a.SubjectID = subj.SubjectID

WHERE a.StudentID = 3

ORDER BY a.Date;

**Output:**



**Q2)** **Count total present and absent days for a student.**

**(Assumption taken by us: A student is considered as present on a day if he/she is present in all classes of that day)**

WITH SubjectTotals AS ( SELECT StudentID, Date, COUNT(\*) AS SubjectsOnDate FROM Attendance GROUP BY StudentID, Date),

PresentTotals AS ( SELECT StudentID, Date, COUNT(\*) AS SubjectsPresent

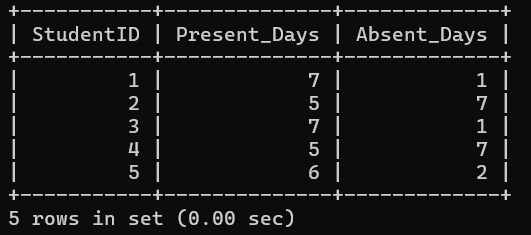
FROM Attendance WHERE Status = 'Present'GROUP BY StudentID,Date),

DailyPresenceStatus AS (SELECT st.StudentID, st.Date,st.SubjectsOnDate, IFNULL(pt.SubjectsPresent, 0) AS SubjectsPresent, CASE WHEN IFNULL(pt.SubjectsPresent, 0) = st.SubjectsOnDate THEN 1 ELSE 0 END AS FullPresence FROM SubjectTotals st LEFT JOIN PresentTotals pt ON st.StudentID = pt.StudentID AND st.Date = pt.Date)

SELECT StudentID, SUM(CASE WHEN FullPresence = 1 THEN 1 ELSE 0 END) AS Present\_Days, SUM(CASE WHEN FullPresence = 0 THEN 1 ELSE 0 END) AS Absent\_Days FROM DailyPresenceStatus GROUP BY StudentID

ORDER BY StudentID;

**OUTPUT:**



**Q.3) Retrieve all students who have less than 75% attendance.**

WITH DailySubjectCounts AS (SELECT StudentID, Date, COUNT(\*) AS Total

FROM Attendance GROUP BY StudentID, Date),

DailyPresentCounts AS ( SELECT StudentID, Date, COUNT(\*) AS Present

FROM Attendance WHERE Status = 'Present' GROUP BY StudentID, Date),

Combined AS ( SELECT d.StudentID, d.Date, d.Total, IFNULL(p.Present, 0) AS Present FROM DailySubjectCounts d

LEFT JOIN DailyPresentCounts p ON d.StudentID = p.StudentID AND d.Date = p.Date),

StatusPerDay AS ( SELECT StudentID, Date,

CASE WHEN Total = Present THEN 'Present' ELSE 'Absent' END AS DayStatus

FROM Combined),

Summary AS (SELECT StudentID, COUNT(\*) AS TotalDays, SUM(CASE WHEN DayStatus = 'Present' THEN 1 ELSE 0 END) AS PresentDays FROM StatusPerDay GROUP BY StudentID)

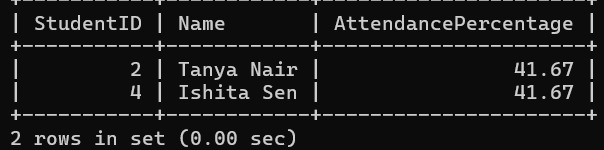
SELECT s.StudentID, st.Name, ROUND(PresentDays \* 100.0 / TotalDays, 2) AS AttendancePercentage

FROM Summary s JOIN Students st ON s.StudentID = st.StudentID

WHERE (PresentDays \* 100.0 / TotalDays) < 75

ORDER BY AttendancePercentage;

**OUTPUT:**

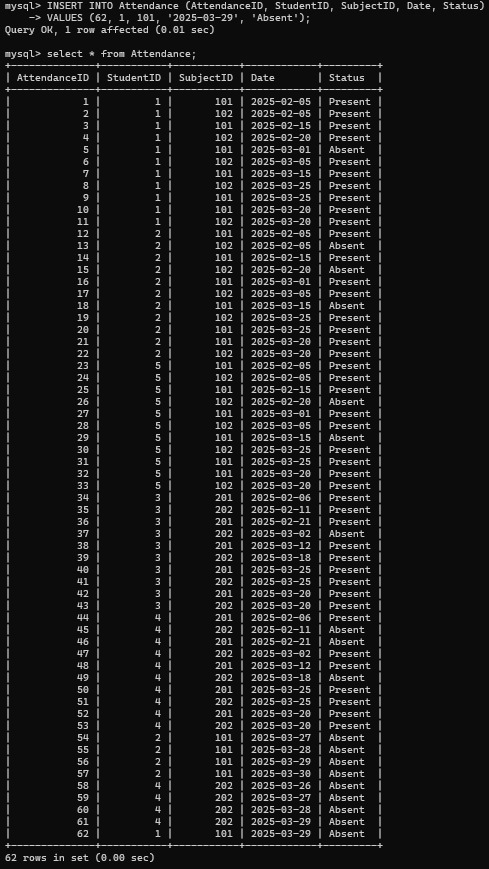


**Q.4) Insert a new attendance record.**

INSERT INTO Attendance (AttendanceID, StudentID, SubjectID, Date, Status)

VALUES (62, 1, 101, '2025-03-29', 'Absent');

**OUTPUT:**



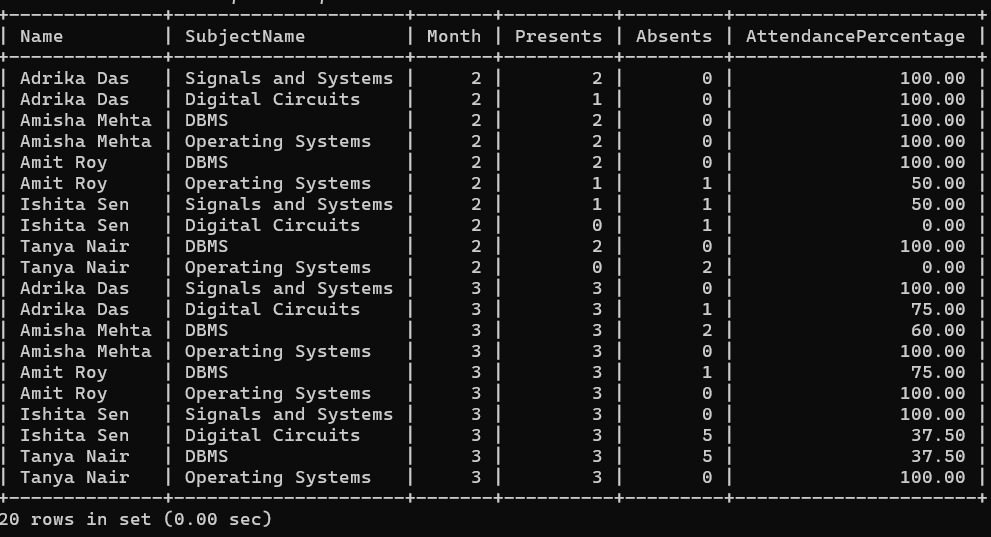
**Q.5 Generate a monthly attendance report (Subjectwise) :**

SELECT S.Name, Sub.SubjectName,EXTRACT(MONTH FROM A.Date) AS Month, SUM(CASE WHEN A.Status = 'Present' THEN 1 ELSE 0 END) AS Presents,

SUM(CASE WHEN A.Status = 'Absent' THEN 1 ELSE 0 END) AS Absents, ROUND(SUM(CASE WHEN A.Status = 'Present' THEN 1 ELSE 0 END) \* 100.0 / COUNT(\*), 2) AS AttendancePercentage FROM Attendance A

JOIN Students S ON A.StudentID = S.StudentID JOIN Subjects Sub ON A.SubjectID = Sub.SubjectID GROUP BY S.Name, Sub.SubjectName, EXTRACT(MONTH FROM A.Date) ORDER BY Month, S.Name;

**OUTPUT:**

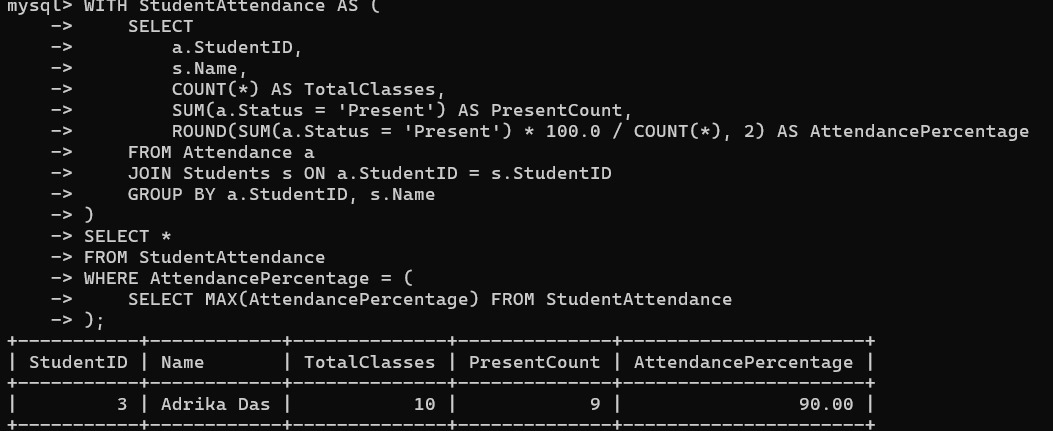


**Q.6) Identify students with the highest attendance.**

WITH StudentAttendance AS ( SELECT a.StudentID, s.Name,COUNT(\*) AS TotalClasses,SUM(a.Status = 'Present') AS PresentCount, ROUND(SUM(a.Status = 'Present') \* 100.0 / COUNT(\*), 2) AS AttendancePercentage FROM Attendance a JOIN Students s ON a.StudentID = s.StudentID GROUP BY a.StudentID, s.Name)

SELECT \* FROM StudentAttendance WHERE AttendancePercentage = (SELECT MAX(AttendancePercentage) FROM StudentAttendance);

**OUTPUT:**

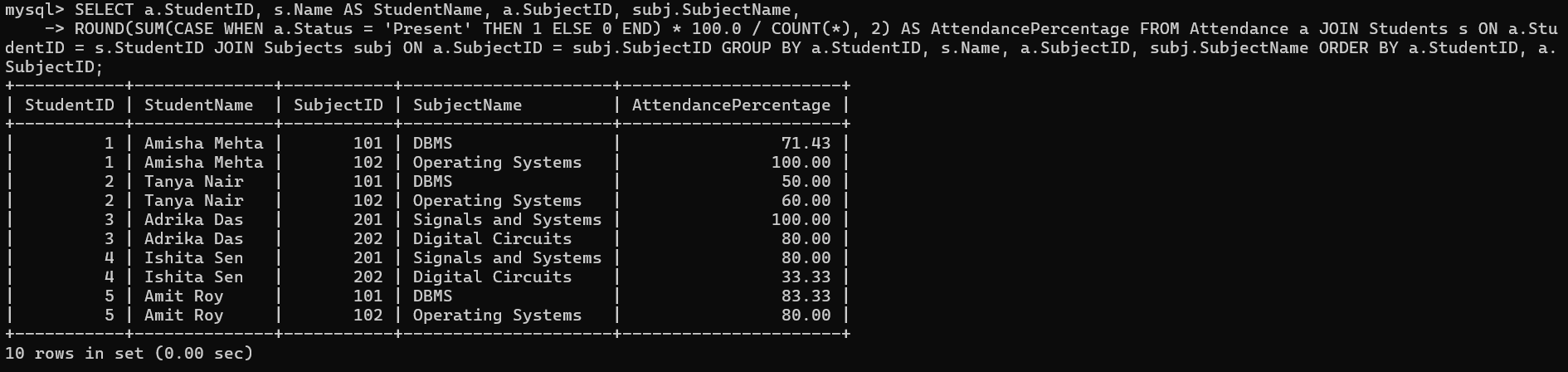


**Q.7) Retrieve attendance percentage by subject.**

SELECT a.StudentID, s.Name AS StudentName, a.SubjectID, subj.SubjectName,

ROUND(SUM(CASE WHEN a.Status = 'Present' THEN 1 ELSE 0 END) \* 100.0 / COUNT(\*), 2) AS AttendancePercentage FROM Attendance a JOIN Students s ON a.StudentID = s.StudentID JOIN Subjects subj ON a.SubjectID = subj.SubjectID GROUP BY a.StudentID, s.Name, a.SubjectID, subj.SubjectName ORDER BY a.StudentID, a.SubjectID;

**OUTPUT**



**Q.8) List students with more than 3 consecutive absences.**

WITH Ranked AS ( SELECT A.StudentID, A.Date, A.Status, ROW\_NUMBER() OVER (PARTITION BY A.StudentID ORDER BY A.Date) AS rn\_all, ROW\_NUMBER() OVER (PARTITION BY A.StudentID, A.Status ORDER BY A.Date) AS rn\_status FROM Attendance A),

Grouped AS ( SELECT StudentID, Date, Status, rn\_all - rn\_status AS grp FROM Ranked)

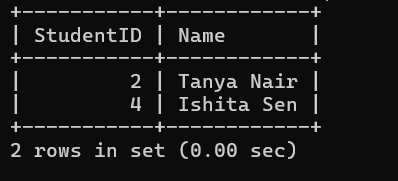
SELECT DISTINCT S.StudentID, S.Name FROM Grouped G

JOIN Students S ON G.StudentID = S.StudentID

WHERE G.Status = 'Absent' GROUP BY G.StudentID, G.grp, S.Name

HAVING COUNT(\*) > 3;

**OUTPUT:**



**Q9) Fetch subjects with the lowest attendance rate.**

DELIMITER $$

CREATE PROCEDURE GetLowestAttendanceSubject()

BEGIN

SELECT a.SubjectID, s.SubjectName, ROUND(SUM(CASE WHEN a.Status = 'Present' THEN 1 ELSE 0 END) \* 100.0 / COUNT(\*), 2) AS AttendancePercentage

FROM Attendance a JOIN Subjects s ON a.SubjectID = s.SubjectID

GROUP BY a.SubjectID, s.SubjectName ORDER BY AttendancePercentage

ASC

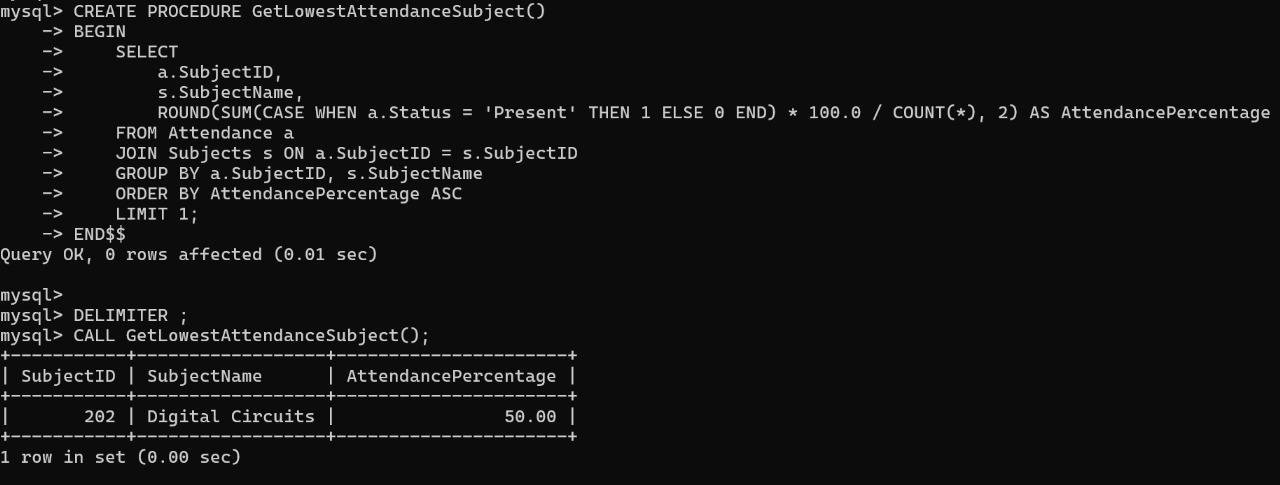
LIMIT 1;

END$$

DELIMITER ;

CALL GetLowestAttendanceSubject();

**OUTPUT:**

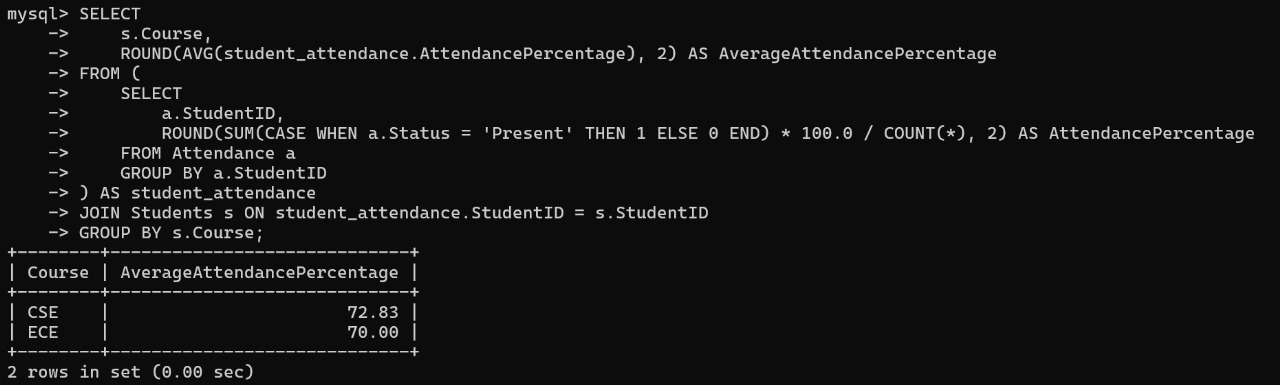


**Q.10) Calculate the average attendance percentage of each course.**

SELECT s.Course, ROUND(AVG(student\_attendance.AttendancePercentage), 2) AS AverageAttendancePercentage FROM ( SELECT a.StudentID, ROUND(SUM(CASE WHEN a.Status = 'Present' THEN 1 ELSE 0 END) \* 100.0 / COUNT(\*), 2) AS AttendancePercentage FROM Attendance a

GROUP BY a.StudentID) AS student\_attendance JOIN Students s ON student\_attendance.StudentID = s.StudentID GROUP BY s.Course;

**OUTPUT:**



**TRIGGERS (OPTIONAL)**

If we wish to prevent further insertion of new records we may create a trigger as well

--Additionally this trigger can be implemented to show an error message everytime a new insertion is attempted if we want to prevent further insertion

DELIMITER $$

CREATE TRIGGER NotifyNewAttendanceInsert

BEFORE INSERT ON Attendance

FOR EACH ROW

BEGIN

DECLARE msg TEXT;

SET msg = CONCAT('New attendance record cannot be inserted: StudentID ',

CAST(NEW.StudentID AS CHAR), ', SubjectID ', CAST(NEW.SubjectID AS CHAR), ', Date ', CAST(NEW.Date AS CHAR) ); SIGNAL SQLSTATE '45000'

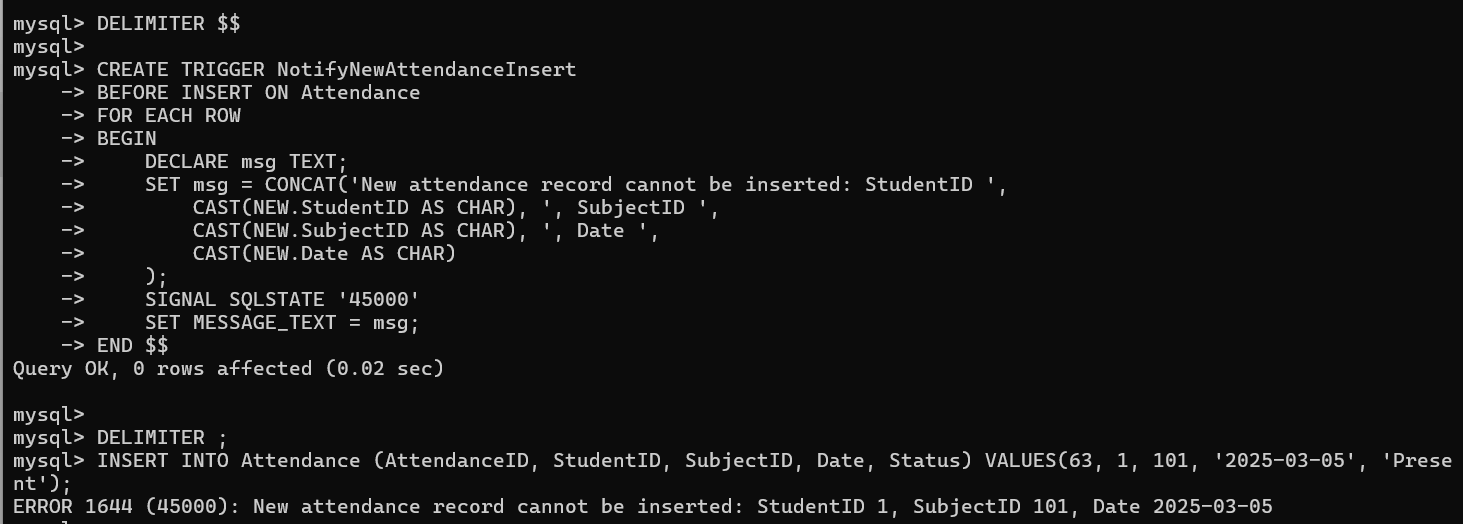
SET MESSAGE\_TEXT = msg;

END $$

DELIMITER ;

**OUTPUT:**

**Error message will be displayed by trigger if new record is inserted**



**Conclusion**

In conclusion, the development of this attendance management system using a virtual MySQL database provided a meaningful opportunity to apply theoretical knowledge to a practical problem. Throughout the course of this project, we designed and implemented a system capable of recording, tracking, and analyzing student attendance efficiently and accurately. The use of structured queries allowed us to extract useful insights such as monthly attendance reports, subject-wise analysis, and identification of students with low attendance.

Beyond technical skills, this project strengthened our problem-solving abilities, attention to detail, and collaborative teamwork. The system not only meets the functional requirements but is also scalable and adaptable for future enhancements. We believe this project lays a strong foundation for more advanced systems that can integrate with real-time biometric inputs or mobile applications for even greater efficiency.

This experience reinforced the value of database systems in academic and administrative environments and highlighted how technology can support transparency, accountability, and better decision-making in educational institutions.

**Github Repository Link for our Project:**

**https://github.com/ahinagangopadhyay/College\_Attendance\_Management\_System**